SmartImage Sensor

Installation & User Guide

Part Ordering Number: DOC-USER

Document Control Number: MAN-100 REV A

FrameWork Release 2.6

Welcome to the large family of DVT product users! We are very pleased that you purchased our products and look forward to helping you build a Smart Factory. We at DVT have made a few improvements and changes to the content of this manual to better assist you in your endeavors. We hope that you find this version of the manual both helpful and accommodating. If you have any questions, suggestions, or comments in regards to any of our user manuals, please help us to better help you by contacting our Technical Editor at docs@dvtsensors.com.

Table of Contents

How to Use this Guide:

The FrameWork SmartImage Sensors Installation & User Guide is your basic resource for both our hardware and software platforms. Users of the Series 700 and 800 hardware platforms will find their hardware addressed in a separate Hardware Installation Guide. This document is designed to provide the users with basic ideas about the functionality of the software. There are other sources of documentation for FrameWork. One such source is the SmartImage Sensors Script Reference Guide, a manual for the programming language used in all scripts in FrameWork. Another source is the FrameWork HTML Help. These HTML- based help files, which will be installed on your PC's hard disk as you install FrameWork, are also available from both the Help menu and the DVT web site at http://www.dvtsensors.com under the Support section. The best source for FrameWork training is the DVT VirtualTour. The DVT Virtual Tour contains training modules that can be used for self-paced learning. It is a free tool that users can request from the DVT website at no charge.

NOTE: This guide assumes users are familiar with Microsoft Windows®.

Chapter 1 - Getting Started

This chapter contains important information for both new and experienced users. New users will benefit from general ideas about machine vision as well as from pointers to other sources of information and training. This chapter starts with an introduction to the newest family of SmartImage Sensors, the Legend Series, including technical specifications and a brief description of every SmartImage Sensor in that family including the newest Legend 520, Legend IS (Intelligent Scanner) and the Legend SC (SpectroCam). Also, some technical information about the older systems (Series 600) is included. It then moves on to how to install the software (FrameWork user interface), walks the user through the process of setting up the SmartImage Sensor, and explains how to establish communications with it. The chapter ends with a brief description of the new features of FrameWork 2.6, and a guide to other resources.

What is Machine Vision?

Machine vision, in layman's terms, is exactly that. It can be defined as machines with their own set of eyes. Using their vision, machines inspect products for defects by detecting the presence or absence of parts, taking measurements, reading codes, etc. How it works may seem rather complicated at first glance. However, with a little help from basic optics, it is not. The machine's eye mainly looks for changes in contrast and uses a 2-dimensional plane to do so. Our SoftSensors tell the camera (the machine's eye) how to evaluate changes in contrast and in what direction to evaluate them. The camera then collects this information from the image and evaluates it following the user-defined rules. Because the camera's actions are so dependent upon that change in contrast, it is imperative that you adjust your lighting such that the camera (or machine) can get a consistent and close look at the factory product.

Imaging and Pixels

Every image from the SmartImage Sensors can be considered an array of pixels. For standard resolution systems, the image consists of 640 columns and 480 rows of pixels (for a total of over 300,000 pixels). For high resolution systems, the image contains 1280 columns and 1024 rows of pixels (for a total of over 1.3 million pixels!). Every pixel in the image can be considered a source of information, so from a standard resolution image, the system obtains over 300,000 data points. For grayscale systems, every pixel provides a numerical value from 0 to 255 which is normalized to values from 0 to 100. This value is the intensity level of that pixel. An intensity value of 0 corresponds to pure black, an intensity value of 100 corresponds to pure white, and the values in between correspond to 99 different shades of gray. For color systems, every pixel provides 3 different intensity values: one for red, one for blue and one for green. These three values make up the RGB content of that pixel. Virtually any color can be represented by using the appropriate coefficient for the base colors (Red, Green and Blue). Using this information SmartImage Sensors inspect images to provide the user with feedback regarding the presence or absence of a part, flaw detection, code reading and/or verification, etc.

What is a SmartImage Sensor?

The SmartImage Sensor is your hardware tool for assuring quality in manufacturing, gathering data, and providing information to all levels of your enterprise. This information might take the form of device-level data such as inspection values being sent to a PLC or process-level Statistical Process Control (SPC) data. This information might be as generalized as a PASS or FAIL signal or as specific as position feedback to a motion controller. You can use the FrameWork software to divide and conquer these tasks. FrameWork was designed to be easy to use for anyone.

Newest Hardware Platform: The Legend Series 500

The Legend Series consists of the 500 family of products as well as the Intelligent Scanner and the SpectroCam. The 500 family of products consists of the following SmartImage Sensors:

- Legend 520: lowest-cost machine vision solution for factories that need high-end photoelectric sensors that easily adapt to new applications. The Legend 520 SmartImage Sensor combines the industry's latest CMOS imaging technology with FrameWork firmware to create an easy-to-use but powerful inspection tool. The grayscale CMOS device has a resolution of 640 by 480 pixels which makes this inexpensive SmartImage Sensor an excellent option for basic applications.

- Legend 530: basic SmartImage Sensor with a CCD for image acquisition. This SmartImage Sensor is similar in functionality to the Legend 520 SmartImage Sensor, but delivers more precision replacing the image acquisition device with a CCD. This

SmartImage Sensor is better than the Legend 520 system for precision measurement applications and analysis of parts that are rich in details.

- Legend 540: fastest SmartImage Sensor available. It consists of a 640 by 480 pixel monochrome CCD and a high speed processor. It combines the quality images provided by a CCD with the power of a fast processor to perform delicate inspections at a very high speed.
- Legend 542C: basic color system. It consists of a color CCD of 640 by 480 pixels which makes the SmartImage Sensor the best choice for most color applications.
- Legend 544: high resolution system. It uses a high resolution CCD (1280 by 1024 pixels) to capture images with an unprecedented level of details. Recommended for cases where ultra high precision is a must.
- Legend 544C: high resolution color system. This SmartImage Sensor uses a high resolution color CCD (1280 by 1024) for color applications that require a high level of details.

Every SmartImage Sensor of the 500 family includes the FrameWork software to set up inspections on part presence/absence, positioning, defect detection, precise measurement, 1D and 2D code readers, OCR, Color (can be used for grayscale levels in monochrome systems), and programmable tools to accommodate user needs. Figure 1 shows the Legend Series 530 SmartImage Sensor.

Figure 1: Member of The Legend Series: the Legend 530 SmartImage Sensor.

The other two devices available in the Legend Series are:

- The Intelligent Scanner: This SmartImage Sensor uses the precision of a CCD to read 1D and 2D codes and characters. It is the only industrial barcode reader to include OCR/OCV capabilities. Reading up to 200 codes per second, the Intelligent Scanner is the leader in speed. It can read the following codes: BC412, PDF417, Codabar, Code 39, Code 128, Interleaved 2-of-5, PharmaCode, UPC/EAN, Code 93, RSS-14, POSTNET, Snowflake 2D code and DataMatrix codes ECC200 and older. It can also perform 1D barcode grading, Optical Character Recognition, and Optical Character Verification (OCV). This SmartImage Sensor shares the form factor of the Legend 500 Series of SmartImage Sensors.
- The SpectroCam: This SmartImage Sensor uses a grayscale CCD (with resolution of 640 by 480) and an external spectrograph. It is a spectral line imaging SmartImage Sensor. SpectroCam captures a line image of a target; using prisms and grating optics, it

disperses light from the line image along the spectrum. Each captured image contains line pixels in spatial axis (Y axis) and spectral pixels in spectral axis (X axis). Unlike other color smart cameras using RGB platforms, SpectroCam offers more detailed color information across the entire visible and Near Infrared (NIR) spectrum. This allows the CCD to capture more data for more accurate measurements and detection of very slight color variations. Unlike point color spectrophotometers, SpectroCam provides spatial resolution and simultaneous measurement across a large number of points. Figure 2 shows the SpectroCam. It shares the form factor with the 500 Series SmartImage Sensors but it requires a different piece of hardware to visualize the spectrum of the part being inspected. It should be noted that the SpectroCam does not come with the integrated ring light. It uses an external white light that offers a very high intensity minimizing the exposure time.

Figure 2: Legend Series SpectroCam SmartImage Sensor.

Legend Series System Specifications

Size: 112mm x 60mm x 30mm (not including spectrograph for the SpectroCam which adds an additional 122mm to the 30mm dimension or a lens) + an additional 50mm cable clearance added to 112mm

Mounting: Four M4 threaded holes (on back of unit), 7.9mm depth

Weight: 170g / 6 oz. (without lens)

Power Requirements: A regulated and isolated 24 V DC power supply is needed (sold separately): 210mA for the 520 and 530 systems (a minimum 5W) or 300mA for the rest of the Legend Series (a minimum of 7.5 W.) If DVT LED array lights are used, the power supply must provide an additional 30W required per array.

Operating Temperatures: 0-45° C (32-113° F), non-condensing environment; Maximum 50° C / 122° F

Electronic Shuttering: 10µs – 1s exposure times (520 supports rolling or global shuttering, rolling shuttering should only be used for indexed applications)

Optics: CS mount standard, C mount-capable with an adapter (sold separately)

External Ports: Keyed 10 pin RJ-45 connector (Power and Digital I/O), RJ-45 (10/100 megabit Ethernet communications, TCP/IP protocol connector)

Digital I/O: 24 Volts DC regulated, 8 configurable inputs & outputs, NPN (current sinking) inputs, PNP (current sourcing) outputs, active high signals. Inputs can sink up to 1.5mA and outputs can source a maximum of 50mA.

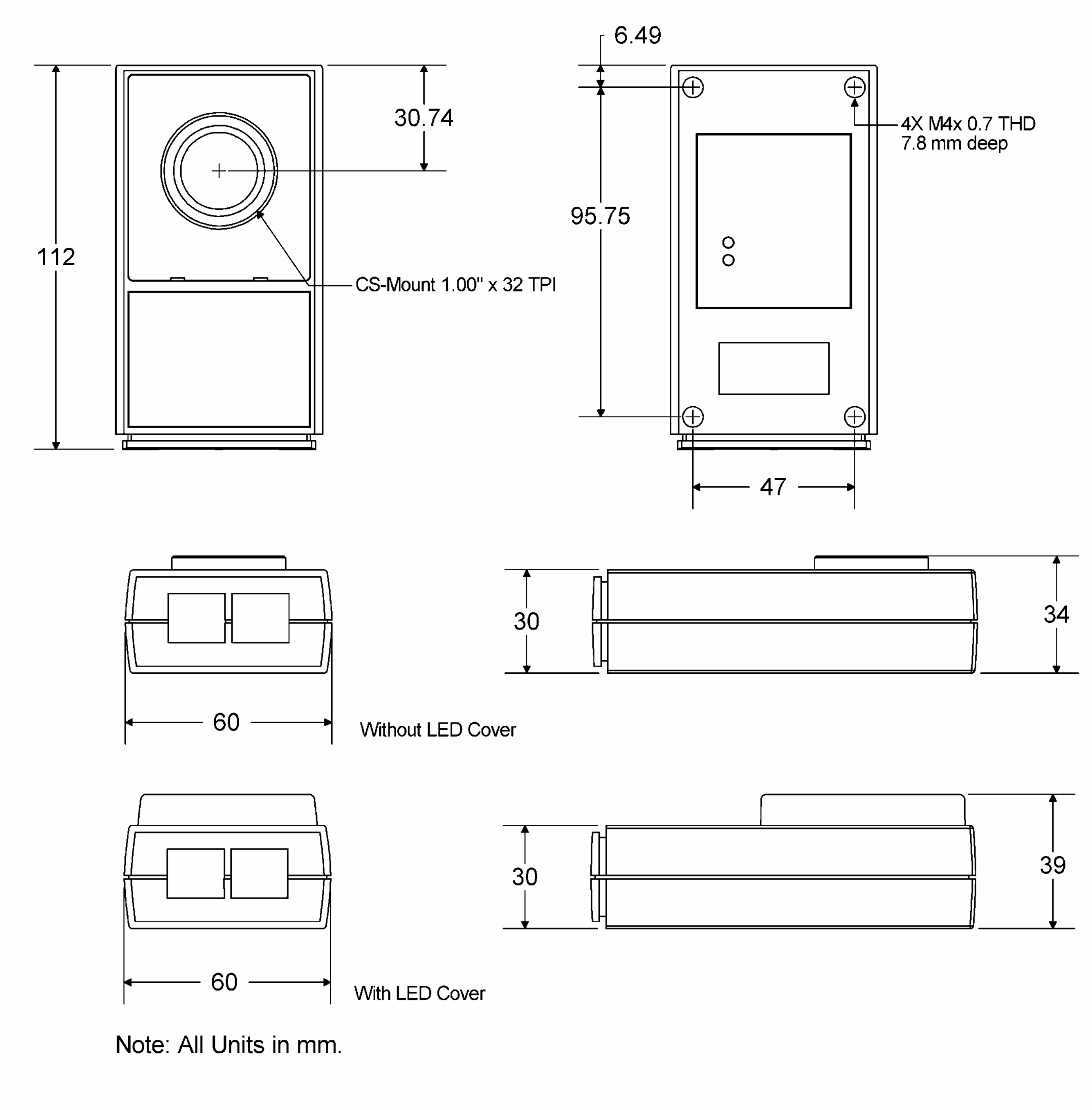
Warning: Users are strongly encouraged to the DVT Digital I/O and Power Cable (sold separately).

Certifications: CE certified

Summary of Image Sensor Information

System	Description	Resolution	CCD Size (Format)
Model 520	Grayscale CMOS	640x480	CMOS 5x3.7mm (1/3")
Model 530	Grayscale CCD	640x480	4.8x3.6 mm (1/3")
Model 540	High Speed Grayscale CCD	640x480	4.8x3.6 mm (1/3")
Model 542C	High Speed Color CCD	640x480	3.2x2.4mm (1/4")
Model 544	High Resolution Grayscale CCD	1280x1024	6.4x4.8 mm (1/2")
Model 544C	High Resolution Color CCD	1280x1024	6.4x4.8 mm (1/2")
Model IS (Intelligent Scanner)	Reader (1D codes, 2D codes, OCR)	640x480	4.8x3.6 mm (1/3")
Model SC (SpectroCam)	Spectrum analyzer for visible and near IR wavelengths	640x480	4.8x3.6 mm (1/3")

Legend Series SmartImage Sensor Dimensions



Series 600 SmartImage Sensors

Figure 3 shows an image of the older family of systems, the Series 600. These systems were replaced by the Legend 530 SmartImage Sensors. The basic functionality is the same except that the newer system requires an external converter for serial communications, has more memory and it is offered at a lower cost. New customers looking to buy a Series 600 SmartImage Sensor should consider purchasing a Legend 530 SmartImage Sensor because of the extra memory space. The latest version of FrameWork, FrameWork 2.6 represents a significant load for the Series 600 SmartImage Sensors due to the extended functionality included in the firmware which increases the size of the file being downloaded to the SmartImage Sensor. However, as covered in the What's New in this Version section, product files will take less space in flash memory in FrameWork 2.6 than they did in FrameWork 2.5. This means that FrameWork 2.6 will adapt better than version 2.5 to the memory limitations of the Series 600. The Legend 530 SmartImage Sensor works optimally with the latest version of the software and will be fully upgradeable with the next release.

Figure 3: The Series 600.

Legend 600 Accessories

System

Power Supply: Regulated and Isolated 24V DC, at a minimum of 5W (supplying 210mA @ 24Volts) power supply is needed. An additional 1.25A (30W) is required if using a DVT LED strobe light.

Dedicated illumination: Standard LED strobe light or optional light source.

Cables

15-pin high-density D-Sub Connector: Carries the Digital I/O and power lines. A fully populated, double shielded cable should be used. A VGA monitor cable will not work for this application.

RJ-45 Ethernet Connector: Supports the Ethernet communications. A Category 5, 6-conductor, twisted pair, shielded (STP), EIA-568 cable should be used. Use straight through-type if connecting the Series 600 to a hub, then a PC. Use a crossover cable or connector if connecting the Series 600 directly to a PC.

RJ-12 cable: Supports the RS-422 communications. A Category 5, 6 conductor, twisted pair, shielded, USOC cable should be used.

Strobe Illumination Cable: Connects an optional strobe light to the Series 600 system. Available from DVT in various lengths.

Series 600 System Specifications

Size: 114mm x 55mm x 40mm (not including lens) + 50mm cable clearance added to the longest dimension.

Mounting: Four M4 threaded holes (on bottom of unit), 7.9mm depth

Weight: 170g / 6 oz. (without lens)

Power Requirements: 24 Volts DC, 210mA at 24 Volts (or minimum 5W supply), regulated and isolated (sold separately). Additional 30W required per DVT LED array light used

Operating Temperatures: 0-45° C, 32-113° F, non-condensing environment

Maximum 50° C / 122° F

Image Sensor: 4.8mm x 3.6mm (1/3" format) CCD, 640 x 480 pixel resolution, 7.4 µm x 7.4 µm, square pixels

Electronic Shuttering: (10µs – 1s exposure times)

Optics: CS mount standard, C mount-capable with included adapter

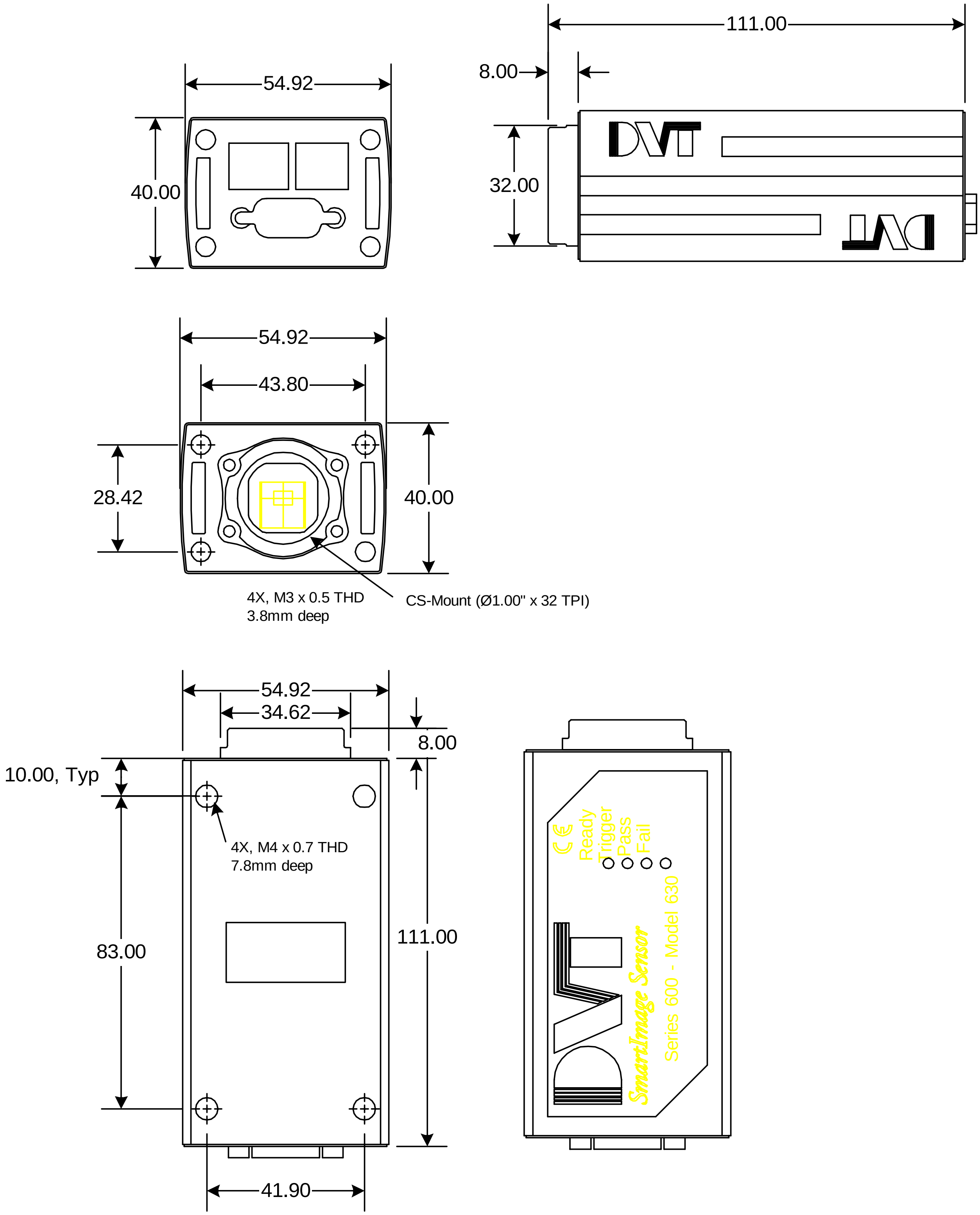
External Ports: 15 pin high density DB Subconnector (Power and Digital I/O), RJ-45 (10/100 megabit Ethernet communications, TCP/IP protocol), RJ-11 (RS-422 serial communications)

Digital I/O: 24 Volts DC regulated, 12 configurable inputs & outputs, current sinking inputs, PNP (current sourcing) outputs, active high signals. Input can sink up to 1.5mA and outputs source a maximum of 100mA.

Warning: Connect with a fully populated, double-shielded, 15-pin high density D-Sub Cable (sold separately)

Certifications: CE certified

Series 600 SmartImage Sensor Dimensions



Note: All units in mm.

Installing FrameWork

While it may be very tempting to immediately insert the FrameWork CD into your computer, please take a moment to ensure that installation will go smoothly. The following is a list of requirements necessary for a proper installation of this version of FrameWork.

PC Requirements

- 400 MHz Pentium-class PC: For best results, we recommend that you use Windows 2000® or a more recent version of Windows® on your PC. However, the following Windows® platforms will work as well: Windows 98® or Windows NT® 4.0. A minimum of 128 MB RAM will be required. A faster processor will directly relate to faster updates of images in the FrameWork user interface. Note: only Windows NT®, 2000, and XP support the new event logging capabilities of FrameWork.

- Available Space: FrameWork requires a minimum 30 MB hard disk space.

- Ethernet Card: 10/100 megabit Ethernet card with integrated RJ-45 jack. For laptop computers, an Ethernet card with CardBus technology is recommended.

Installing the FrameWork User Interface

The steps to follow are:

- Insert the FrameWork CD. The FrameWork installation should automatically start. However, if the CD does not automatically run, follow these steps:
- From the Start menu, choose Run.
- Type D:\setup.exe (substitute the appropriate drive letter of your CD-ROM drive for D) and press Enter.
- Follow the installation instructions on screen.

Hint: The FrameWork installation program leaves intact older versions of FrameWork. You will still have access to FrameWork 2.5, 2.4, 2.3, 2.2, 2.1, 2.0, 1.4, 1.3, or other versions installed in the “c:\Program Files\DVT\” directories.

After installation, FrameWork will reside in the “C:\Program Files\dvt\FrameWork2x” directory (assuming the user accepts the default selection for directory location). The FrameWork User Interface file is named “Fwk.exe” and firmware files end with a numerical extension indicating the DVT series number (the hardware platform with which they are associated). For example, the firmware file for FrameWork 2.6 release 1 for a Legend 542C system is named “fwk261.542C.” These firmware file contains the code that needs to be downloaded to the SmartImage Sensor. They are the part of FrameWork that resides in the SmartImage Sensor where all the calculations take place.

Please take a moment to fill out the registration card included with this manual and return to DVT. Return of this registration card activates your Warranty.

SmartImage Sensor Hardware Setup

In this section, physical connections to the SmartImage Sensor are explained. This includes powering the system, connecting the communications cable, and connecting to the I/O breakout board.

Unpacking and Powering the Legend Series

Connect the keyed RJ-45 Digital I/O and power cable to the SmartImage Sensor (typical part number: CBL5-D10).

Note: The cable used for The Legend Series 500 has a male 15-pin high density DB-sub-connector on one end and a keyed 10 pin RJ-45 connector on the other end.

Connect the RJ-45 Ethernet cable (typical part number: CBL-C10E).

Note: If you are connecting the other end of the Ethernet cable directly to a PC (as opposed to a hub), make sure to use either a crossover Ethernet cable (CBL-C10EX) or a crossover connector (CON-RJ45N).

Connect the opposite end of the digital I/O & power connector (High Density 15 pin D connector [CBL5-D10]) to the DVT I/O breakout board (CON-IBOB). Or, if you do not plan on using the breakout board, cut the connector from the 15-pin cable and wire directly into power and I/O destinations.

Connect a 24V DC power supply to the DVT I/O breakout board using the last three screw-down terminals.

Note: For laboratory use, a power supply is available from DVT. The part number is ACC-24-NA (for North America). This power supply uses the power jack on the DVT I/O breakout board.

The isolation breakout board contains two sets of terminals: one that is directly connected to the SmartImage Sensor, and one that allows for the use of opto-isolation modules. As we saw in the previous steps the port on the right (see diagram) includes I/O, strobe and power screw-down terminals. The I/O lines on this port are to be used when isolation is not needed.

The second port, the one on the left side (see diagram) has the I/O connections to be used in conjunction with isolation modules. These terminals are inactive unless isolation modules are used. Users should decide which lines need isolation and install the corresponding modules (input or output) for those lines only. For more information refer to the following document on the DVT website: DVT Isolation Break Out Board Note.

Apply power to the SmartImage Sensor and wait for the system to boot. The Status light will illuminate with a constant green light when the boot sequence is finished.

Attach a lens to your camera. If you are using a C-mount lens, you will find it necessary to use a 5mm adapter (LDS-C). If you have a CS-mount lens, you will not need the adapter.

Unpacking and Powering the Series 600

Connect the Ethernet cable (typical part number: CBL-C10E).

Note: If you are connecting the other end of the Ethernet cable directly to a PC (as opposed to a hub, make sure to use either a crossover Ethernet cable (CBL-C10EX) or a crossover connector (CON-RJ45N).

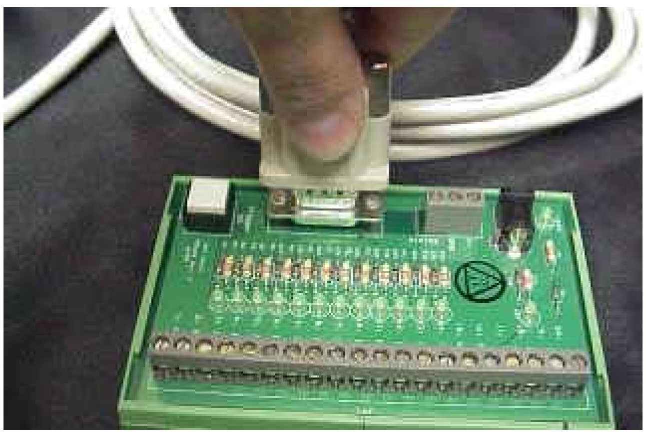
Optional: Connect the Serial cable (typical part number: CB-C104).

The serial port on a Series 600 is an RS-422 port. A converter for RS-422 to RS-232 is available from DVT. The part number is CON-BRK (formerly known as CON-600).

The Series 600 serial port is used for loading new firmware versions into the camera, configuring the TCP/IP settings on a camera, and for communicating with other serial-enabled devices (such as HMI's). The Legend Series can be set up (TCP/IP parameters and firmware download) using Ethernet, which simplifies the process.

Connect the 15-pin power & digital I/O cable (typical part number: CBL6-D10).

The cable used here has a 15-pin high density DB-subconnector on each end (1 male and 1 female).



Connect the opposite end of the 15-pin power & digital I/O cable to the Series 600 I/O breakout board (CON-BRK). Or, if you do not plan on using the I/O breakout board, cut the connector from the 15-pin cable and wire directly into power and I/O destinations.

Connect a 24V DC power supply to the breakout board using the screw-down terminals just above the 15-pin connector.

Note: For laboratory use, a power supply is available from DVT. The part number is ACC-24-NA (North America). This power supply uses the power jack on the I/O breakout board.

Apply power to the SmartImage Sensor and wait for the system to boot. The camera lights the "Ready" LED with a constant green light when the system completes the sequence.

Attach a lens to your camera. If you are using a C-mount lens, you will find it necessary to use a 5mm adapter (LDS-C). If you have a CS-mount lens, you will not need the adapter.

Communicating with the SmartImage Sensor

The DVT SmartImage Sensors can use Ethernet and the TCP/IP and UDP/IP communication protocols. This section outlines how to establish communications with the SmartImage Sensor. In order to communicate with a SmartImage Sensor, you will need the following:

- PC: A Pentium class PC with the following hardware and software:
  - Windows® Operating System: Windows 98®, NT, XP, or 2000. (Windows 2000® or a more recent release is recommended.)
  - FrameWork Software: DVT's FrameWork User Interface.
  - Ethernet Network Card: 10/100 megabit Ethernet card. For laptop computers, an Ethernet card with CardBus technology and integrated RJ-45 jack is recommended. The Ethernet card's appropriate drivers should be installed and the TCP/IP communications network protocol should be installed and setup.
  - Ethernet Switch (Hub): Full Duplex 10/100 megabit Ethernet Switch is necessary for connecting multiple SmartImage sensors in a network.
  - Ethernet Cable: Category 5 Ethernet Cable with RJ-45 connectors. (Note: a cross-over Ethernet cable should be used when connecting directly to a single SmartImage Sensor.)

The RJ-45 connection on your Legend Series 500 or Series 600 is designed to communicate between the SmartImage Sensor and a PC running FrameWork. To do this, the PC must have an Ethernet network card running the TCP/IP protocol. Figure 4 shows where the connectors are located in the SmartImage Sensors.

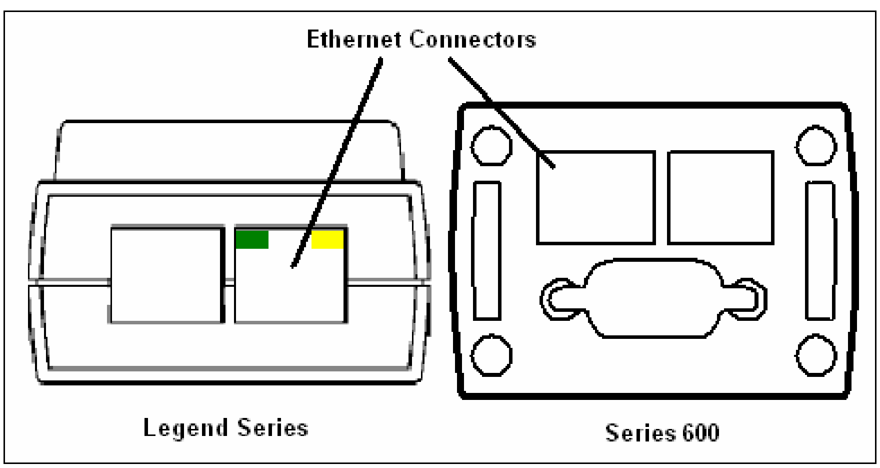


Figure 4: For the Legend Series, the RJ-45 (Ethernet) port is the port on the right with indicator LEDs built into it. The Series 600 SmartImage Sensors do not have the built in status LEDs in this connector.

Assigning an IP Address to a SmartImage Sensor

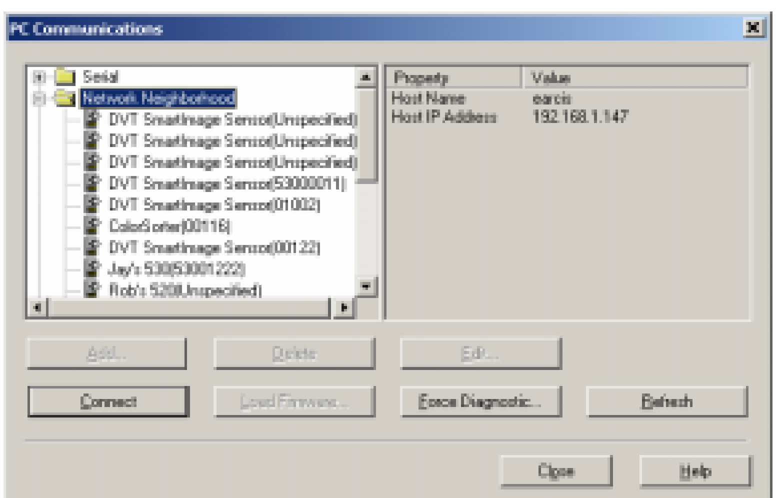
The PC and SmartImage Sensor communicate using the TCP/IP and UDP/IP networking protocols. TCP/IP stands for Transmission Control Protocol / Internet Protocol and is based on identifying elements on a network by unique numbers, called IP addresses. In order for two devices to communicate they must reside on the same Local Area Network (LAN), be part of the same sub network, and have unique IP addresses. UDP stands for User Datagram Protocol and is lightweight, connectionless and efficient but it does not guarantee successful and error-free delivery of data packets. This protocol is used within FrameWork for browsing the network and finding DVT SmartImage Sensors.

All SmartImage Sensor systems come with a default IP address, which must be changed to a number compatible with both your computer and LAN. An IP address is the equivalent to a telephone number. The SmartImage Sensor will assign itself a default IP address. You must then re-assign a new IP address that is compatible with your local network. In other words, you must make sure that you have the right phone number to call the SmartImage Sensor!

Please use the following steps to set up your computer and camera with the correct IP addresses:

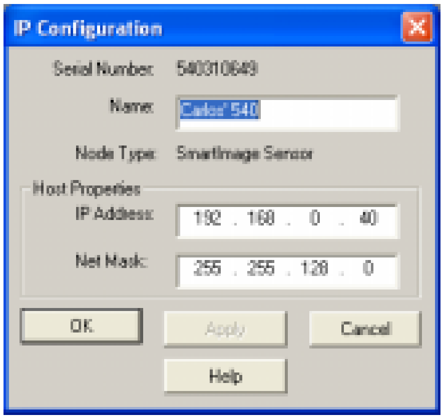
- Start FrameWork.

- Click and Open the "Network Neighborhood". Network Neighborhood allows you to identify cameras on a network. Your computer will search your local network for SmartImage Sensors. When "Network Neighborhood" is highlighted, the right pane will show the current IP address of your computer. Record your computer's (Host) IP address on a piece of paper. If no camera has been found, wait about 10 seconds and search again. There is sometimes a small delay due to the time it takes for your computer to recognize the camera.



- Highlight the SmartImage Sensor with the appropriate serial number.

Note: The Legend Series 500 Systems will have a serial number in parenthesis next their user configurable name. Series 600s will have “Unspecified” for the serial number. It is easiest to identify these units by connecting directly to them or by the unique random IP that is initially assigned to them. This IP will have the form 169.254.xxx.xxx and will be visible in the right pane when the SmartImage Sensor is highlighted.



- Click on the edit button. When you do so, the above window will appear. In the Edit dialog box, you can change the camera’s IP address, subnet mask and the name of the SmartImage Sensor. Set the IP address to be same as your computer’s IP address except for the last number. For instance, computer: 192.168.1.113; SmartImage Sensor: 192.168.1.32. When you hit the apply button, the units information will be permanently stored in the unit’s flash memory. If you are not sure about which IP address to use you should ask your network administrator.

- If the IP address is already being used by another device, you will receive the following message. Simply go back and edit the systems’ IP address once more to a number that is not currently being used.

- Highlight your SmartImage Sensor in the Network Neighborhood and click on the Add button. This will add the SmartImage Sensor to the Ethernet folder, which contains your most commonly used systems.

Hint: After configuring your SmartImage Sensor with its new IP address, you should affix a label to the camera with the Ethernet properties so that you do not forget this important information.

Repeat these steps for each SmartImage Sensor you have. When complete, the PC and all SmartImage Sensors will have unique IP addresses. Connect all of the systems to a hub (or multiple hubs) and the hub to your PC. You now have a "DVT network."

- For more advanced network configurations, you will need to adjust the Sensor Address Mask. For a SmartImage Sensor and PC to communicate, they must be on the same sub-network. This means that the masked portions of the camera's IP address must be identical to the PC's. For example, with the default mask of 255.255.255.0 on your SmartImage Sensor, only the fourth portion of the PC's IP address may be different from your SmartImage Sensor. In this example, a camera at 192.168.0.242 can communicate with a PC at 192.168.0.200, but not with a PC at 192.168.1.200. The following table summarizes the use of the Sensor Address Mask property with Ethernet networks.

Sensor Address Mask	Camera IP Address	Valid Sample PC IP Addresses
255.255.255.0	192.168.0.242	192.168.0.200, 192.168.0.100, 192.168.0.240
255.255.0.0	192.168.0.242	192.168.1.200, 192.168.35.67, 192.168.222.35
255.0.0.0	192.168.0.242	192.25.1.200, 192.222.22.242, 192.0.1.45

Warning: Whenever connecting a node to an existing network, please exercise extreme caution. Connecting a SmartImage Sensor system onto a network without the Network Administrator's knowledge is not recommended.

What's New in This Version

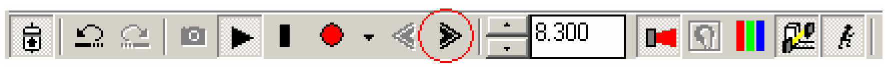
Many enhancements have been added to the latest version of FrameWork (FrameWork 2.6.) The major enhancements are summarized here. For more information about each one of them, please see the appropriate section of this User's Guide.

One interface for all systems

Starting in version 2.6, FrameWork will be the user interface for all DVT SmartImage Sensors. When connecting to a system, FrameWork will auto configure to reflect the functionality of that particular SmartImage Sensor.

Better interface layout and functionality

Now the spectrograph SoftSensor (available for SpectroCam systems) is available from the main SoftSensor toolbar. Also, more functionality has been assigned to the next image button located in the expanded toolbar. When working with the Static Image Window, this button can be used to move to the next image in the sequence. If the system is in external trigger mode and we are utilizing the SID, the button will work as a trigger signal. If the user is connected to a SmartImage Sensor, the system will acquire a new image to inspect; if the emulator is being used, the next image in the sequence is displayed.



Improved memory management system

Memory management was redesigned so that product files will take comparatively less flash memory with FrameWork 2.6 than they did with previous versions. This is due to a compression mechanism applied to products saved in flash memory. The RAM consumption of some SoftSensors, such as the OCR Reader, was also reduced.

Addition of global Coordinate System

Starting in FrameWork 2.6, the Coordinate System SoftSensor can be configured as a global calibration and used from any product. For more information consult the section covering this topic.

Addition of XML support for DataLink

Now DataLink's output data can be formatted using XML tags. XML is becoming more popular for labeling the actual content or meaning of data, and a selection of this option will give FrameWork users the ability to apply this format whenever is needed.

Improved 2D Reader SoftSensors

There are some improvements to the DataMatrix SoftSensors that make these tools more robust, reduce processing time and allow for auto detection of the type of code (dark on light background or light on dark background.) There is also the addition of a SnowFlake reader SoftSensor. SnowFlake is another two-dimensional code.

Enhanced 1D Reader SoftSensors

Starting in FrameWork 2.6, 1-D reader SoftSensors are able to read three new types of 1D codes: POSTNET, RSS-14 and Code 93. There is also the addition of a new area-based SoftSensor. This variation of the 1D Reader SoftSensor would scan a region of interest to find and decode a barcode present anywhere in that area as long as there is not excessive rotation.

New threshold option for Blob Generator SoftSensor

A new threshold option has been added to the Blob Generator SoftSensor: gradient for edge detection. This new threshold works for both grayscale and color images. For more information refer to the section on thresholding.

Improved color learning options

Advanced features for color learning allow users to perform one of several learning methods through the digital relearn signal or manually.

Ability to prevent color SoftSensors from inspecting

After a color is imported from a pixel counting SoftSensor into another tool, the former can become an unnecessary load for the product its results are no longer used. Starting in FrameWork 2.6, an option will be available to prevent the pixel counting SoftSensor from being processed in every inspection. The SoftSensor will only be processed when the digital relearn signal is received, provided that the tool is enabled for digital relearn.

Enhanced script functionality

Starting in FrameWork 2.6 scripts will adhere more to the Java syntax, including the support for new assignment operators and, more importantly, user-defined functions. Also, the number of registers available for data sharing and storage has increased from 4096 to 16384. These registers are indexed 0 through 16383 and just like in FrameWork 2.5 the first 12 registers (0 through 11) are reserved. For more details refer to the Script Guide (Part number DOC –SCR) available for download at:
http://www.dvtsensors.com/support/DownloadsManagerNew.php?KW=Manual

Improved Spectrograph SoftSensor

New features were added to the Spectrograph SoftSensor present in the SpectroCam system. With capabilities ranging from image filtering for noise to calibration in nanometers instead of pixels, this SoftSensor has become a very powerful tool for spectrum analysis.

Register Design Setup for print registration

Users are able to create a customized print registration configuration. This can be found in the “Image Parameters” dialog box under the Images menu. This configuration will then be used by the Segmentation SoftSensor when “Abs. Register” is selected as the rule type under the “Rules” tab of the property dialog box of the SoftSensor.

Ability to restore and backup system files from external applications

Starting in FrameWork 2.6 it is possible to load a system file to a SmartImage Sensor from an application through a command line call to FrameWork's executable. This will start FrameWork in a non-GUI mode. The command is:

fwk.exe -SendSystemFile *IPAddressOfSensor PathToTheSystemFile UserName Password*

It is important to note that this will substitute the previous system file stored in the camera and it will save the new system to flash memory. This operation should only be done sporadically as the user could be in a situation of having to reclaim flash memory frequently.

It is also possible to backup system files from the SmartImage Sensor to the PC. The command line call to FrameWork’s executable to do this is the following:

fwk.exe -CreateSystemFile *IPAddressOfSensor PathToTheSystemFileToCreate*

Getting More From FrameWork

In addition to the FrameWork SmartImage Sensors Installation & User's Guide, you can learn more about FrameWork software from the following resources:

On-line Documentation & Help Files

After FrameWork Software has been installed on a PC, all the necessary documentation is included along with the application software. HTML Help and PDF (Portable Document Format) versions of the written documentation are included.

- HTML Help: The help file is a great source of information about all aspects of FrameWork. There are several ways to access the help file:
  - Click on the Help menu and choose Help File Index or press F1.
  - Use the Help buttons located in most dialog boxes (navigates to the relevant location in the help file).
  - Hold the Shift key and press the F1 key to get the  prompt. From this, click on any part of the user interface for context-sensitive help.
  - The Help file can be viewed independent of a FrameWork session. Simply click on the Windows® Start menu, choose DVT Applications, and then choose "FrameWork Help".
  - Access them directly from the DVT website. These help files are continuously updated by DVT.
- PDF Files: The written documentation provided by DVT in PDF format includes this Installation & User Guide. In addition, the DVT Script Reference Manual is both loaded on your computer and available on-line in PDF format. A hard copy of the Script Reference can also be purchased separately through a DVT Distributor. If you have a Series 700 or 800 SmartImage Sensor, you will also need the SmartImage Sensor Hardware Installation guide. Hard copies of these documents are also included with all SmartImage Sensor shipments from DVT. In addition, users can click on the Help menu and choose the PDF version of the written document to view. The PDF files are created with Adobe® Acrobat® and can be viewed and printed from the free Acrobat Reader® software. Acrobat Reader installs as part of a standard FrameWork installation.
- FrameWork Seminars/Workshops: DVT offers SmartImage Sensor products through a worldwide network of Automation Solution Providers (ASPs). Local seminars and workshops can be found on the DVT website. You can find this information on the web under the Education portion of the DVT website. Also, you can contact your local ASP for dates and locations of the next FrameWork Seminar/Workshop. These workshops offer an introductory view of both the SmartImage Sensor hardware platform and the FrameWork software. The ASP Seminar/Workshop is a perfect way to "get your feet wet" on DVT's inspection systems.
- FrameWork Training Courses: Comprehensive two-day or four-day training courses are taught at DVT Headquarters. These courses are a great way to get a quick start on learning FrameWork. They are free of charge, with users responsible only for their transportation, dinners, and lodging. To learn more and to sign up for a class, please

contact your local Automation Solution Provider (ASP) or go to http://www.dvtsensors.com/training and click on "Classroom training" for the schedule.

- CD-Rom/Self Paced Training: Training is also available on a multimedia CD-Rom, which is available for free from DVT. The information on the CD-Rom is also available on our website. Visit the DVT web site at http://www.dvtsensors.com and click on "Request Information" and follow the directions to the on-line training area.

- Online Training: Weekly 45-minute courses using the latest technology of online conferencing. These courses are free of charge as well; please contact your distributor for more information.

- Subscribe to the SmartList: Join the long list of SmartImage Sensor users who regularly discuss topics via DVT's SmartList server. Discussion topics include (but are not limited to):

    - Application questions: "What light source should I use?"
    - FrameWork software questions: "Which SoftSensor is best for this application?"
    - Integration questions: "Has anyone gotten the Series 600 & a XYZ PLC to communicate?"

  For information on SmartList, visit the DVT web site, http://www.dvtsensors.com and click "Support".

  Users can join the SmartList by visiting the support section of the DVT website.

- DVT Certified Automation Solution Provider (ASP): DVT sells all of its products through a worldwide network of Automation Solution Providers. These ASPs are certified annually based on in-depth knowledge of both FrameWork and the SmartImage Sensor inspection system. If you are unsure of who is your DVT Solution Provider, you can search for that information on the DVT website. Please contact your local ASP with your technical support questions.

- Stay informed about New FrameWork Releases: Visit the DVT Web site for up-to-date information about forthcoming releases of FrameWork. There are two easy ways to reach the web site:

    - From FrameWork, click on the Help menu and choose DVT on the World Wide Web.
    - Start your web browser and type http://www.dvtsensors.com on the address bar.

Chapter 2 - A Closer Look at the System

This chapter describes all the basic principles behind SmartImage Sensors. The internal structure of the SmartImage Sensor is described and a brief description about the parameters available at every level is provided. From system level (unique to every SmartImage Sensor) to SoftSensor level (dependent upon the SoftSensor being used), this chapter illustrates the use of their parameters and the functionality they add to the system. Once the user becomes familiar with the internal structure of SmartImage Sensors, the use of these will become a very simple procedure because FrameWork is designed to be an easy to use interface. The SmartImage Sensor hardware emulator is also explained in this chapter. The hardware emulator is a very powerful tool that simplifies troubleshooting, shortens setup and assists in training.

As mentioned before, SmartImage Sensors can take an image and analyze it to determine whether, based on user-defined parameters, that is a good image or not. Based on this output from the SmartImage Sensor, the user can take the necessary action (e.g. reject a part, fix the production process, etc.). The purpose of this section is to discuss the functionality inside the SmartImage Sensor. Figure 5 shows the hierarchical tree determined by inner components of SmartImage Sensors.

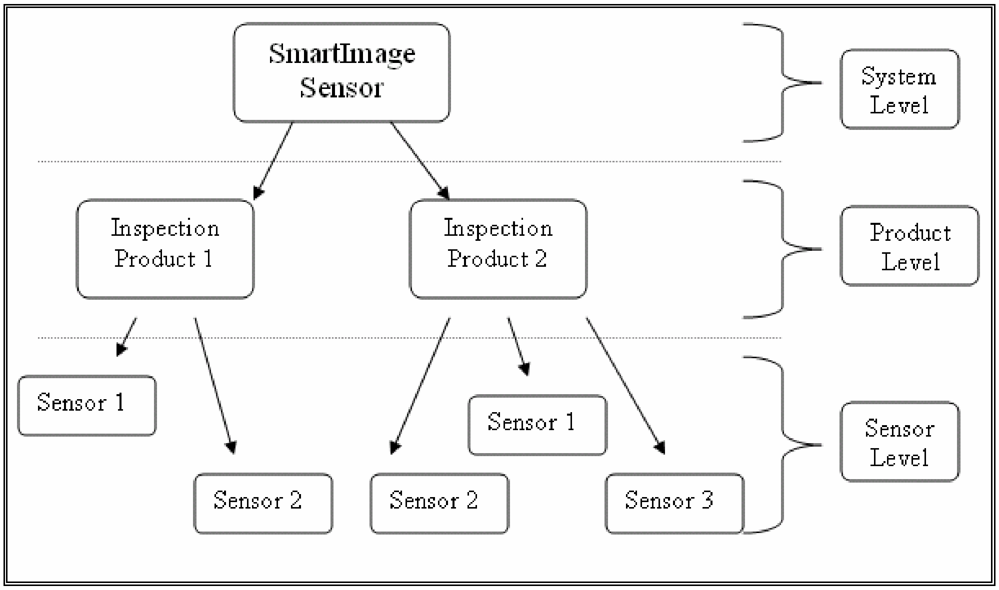


Figure 5. Hierarchical organization within the SmartImage Sensor.

At the top level, we find the system parameters. These parameters are common to every inspection that the SmartImage sensor takes, they affect the overall behavior of the SmartImage Sensor rather than that of a certain inspection. Next, at the product level, the user can change parameters that affect a specific inspection (such as illumination). Essentially, a product is directly associated with an inspection, so product parameters affect only one of the inspections that the SmartImage Sensor is taking. Finally, at the sensor level, the user can set some sensor parameters. Each sensor performs part of an inspection; here is where all the pieces of every inspection are defined. SoftSensors are associated with inspection tasks.

In order to illustrate this functionality let us work with an example. The part in Figure 6 needs to be inspected. First, we need to verify that the drillings on the left end are vertically centered, a certain distance apart, and of a certain radius. After that, we need to verify that the label on the right end is centered, aligned and that it contains the correct code. Finally, the specific data from every inspection has to be sent out of the SmartImage Sensor using Modbus communications. The existence of products inside SmartImage Sensors simplifies this task. The inspections can be separated into two groups, those that take place on the left end of the part and those that take place on the right end of the part. This will allow the user to trigger the SmartImage Sensor twice as the part moves in a conveyor from left to right. The first trigger would occur when the left end is in front of the SmartImage Sensor and the second one when the right end is in front of the SmartImage Sensor.

Figure 6: Sample part to be analyzed in two different inspections.

This would not only add modularity to the process but also would allow for a more precise inspection because now we can zoom in on the part. There is not a need to have the entire part in a single image. The inspections are then separated into products. A product would inspect the left end of the part and another product would inspect the right end. In order to determine if there is a problem with the part, those products would use a number of SoftSensors. For the first product, both Math and Measurement SoftSensors are needed to perform the inspection. For the second product, Math and Measurement SoftSensors are again needed and a label reader SoftSensor (OCR SoftSensor) would also be required in this case. Finally, to send out the data, communications, a system level parameter, needs to be configured. Regardless of the inspection taking place, a data transfer will occur.

System Parameters

As mentioned before, the System level of a SmartImage Sensor controls the overall functionality of the SmartImage Sensor. The main system parameters are inspection mode, password protection, I/O configuration, background scripts, power-on-product, trigger source, FOV balance, and communications setup.

Inspection Mode

A SmartImage Sensor can be taking images for the purpose of inspecting them or simply for displaying them in a user interface. It is important to distinguish between these two modes regarding image acquisition: play mode and inspection mode. Play mode applies to each user interface connected to a SmartImage Sensor and it is independent of the inspection process so its activation does not require the presence of an inspection product. When Real-Time-Feedback is enabled in a user interface play mode becomes active and the Sensor sends images to be displayed by each program. Inspection mode determines whether the SmartImage Sensor is currently inspecting images or not. When inspection mode is active, the SmartImage Sensor sets all configured output signals and sends out any TCP data that the user has set up. The activation of inspection mode requires an inspection product. If a product is selected as the inspection product and the inspection mode is activated, then when the SmartImage Sensor is triggered, it acquires an image and inspects it according to the sensors that the chosen inspection product contains and sets all appropriate outputs.

Password Protection

In order to provide administrators with access control, FrameWork supports the creation of different user accounts. This allows system administrators to give specific control of the system to different users. Each user can be given certain access restrictions, from simply stopping the inspection mode to editing the parameters of every SoftSensor. Administrators can choose which tasks each user is allowed to perform on the system. Furthermore, an inactivity timeout can be set to maximize security. Since version 2.5 of FrameWork, the access control is independent of the PC being used: the access control information is stored in the internal flash memory of the SmartImage Sensor.

Digital I/O Configuration

SmartImage Sensors have a number of configurable I/O lines. Legend Series SmartImage Sensors contain eight I/O lines and older systems (Series 600) contain twelve I/O lines. Having fewer I/O lines allows for the smaller size of newer systems as well as their lower cost. It should be mentioned that the number of I/O lines in the Legend Series can be increased to 32 inputs and 64 outputs by using remote Ethernet I/O devices. The I/O lines can be configured to better suit the user's needs. SmartImage Sensors do not have a concept of input or output lines. Every I/O line can be configured as an input or as an output depending on the user's needs. This assignment is a system parameter because another device will be connected to the I/O lines, and this cannot change between inspections.

Input Functions

Inputs on the SmartImage Sensor can serve one of 5 functions. The following table summarizes these functions:

Input Function	Purpose
Product ID Bits	Each line will represent a digit to make up the binary code that identifies a product based on the assigned Product ID.
Product Select	Toggled when **Product ID** bits are active to select a Product. This signal tells the camera that the Product ID in binary code is ready for reading.
Inspection Trigger	Turned on to start acquisition of an image for inspection when the system is in external trigger mode.
Digital Relearn	Used by Template, ObjectFind, OCR, Code readers, Spectrograph, and Color SoftSensors to automatically relearn or reset SoftSensor parameters. Relearning will occur if the Digital Relearn input is held active during the next inspection and the SoftSensor has its Digital Relearn parameter checked.
Input 18-31	Inputs 18-31 are used in combination with SoftSensor Scripts and Background Scripts to monitor inputs. (See the Script Reference Manual for more details and examples.)

In addition to assigning specific functions to the I/O pins, the user can specify other parameters to ensure that the inputs are correctly read by the system. The use of these parameters depends on the hardware device providing the connection to the SmartImage Sensor. These parameters are polarity inversion and debounce times. Polarity inversion is used to change the system lines from active high to active low. Debounce times are used when the signal received by the SmartImage Sensor is affected by noise. If this is the case, the user can select a period of time during which the SmartImage Sensor does not look at the line after registering a transition in the state of that line.

Output Functions

There are several output functions that can be assigned to the I/O lines. The following table summarizes the available output functions and their meaning. Any function may be assigned to any I/O line.

Output Function	Purpose
Pass	After inspection is completed, **Pass** is set to show that all SoftSensors within the Inspected Product passed inspection.
Warn	After inspection is completed, **Warn** is set to show that any SoftSensor within the Inspected Product met its warn condition and no SoftSensors failed.
Fail	After inspection is completed, **Fail** is set to show that any SoftSensor within the Inspected Product failed inspection.
Busy	**Busy** is set after **Inspection Trigger** is read until the inspection result (**Pass, Warn** or **Fail**) is signaled.
Strobe	When active, the corresponding strobe light source will be on during image exposure.
Strobe 2	When active, the corresponding strobe light source will be on during image exposure.
Strobe 3	When active, the corresponding strobe light source will be on during image exposure.
Resource Conflict	Signals a conflict between inspections trying to use the same system resources. Typically, signals that an **Inspection Trigger** was received while the system was acquiring another image.
Select Pass	While **Select** input is active, **Select Pass** signals that the **Product ID** bits signaled a Product ID found in the system.
Select Fail	While **Select** input is active, **Select Fail** signals that the **Product ID** bits signaled an invalid Product ID.
Acquiring	Signaled after **Inspection Trigger** is read, during exposure time and image digitization. Another **Inspection Trigger** cannot be signaled while **Acquiring** is active.
Inspecting	Signaled after the **Acquiring** signal while an image is being processed for inspection. (Note: **Acquiring + Inspecting = Busy**)
Inspection Toggle	Changes state once after each inspection is performed.
User1 – User16	User definable outputs that go active or inactive based on combination of the results of the SoftSensors being used. User Definable outputs are Product-specific.
Wrong Code	Indicates whether any reader SoftSensor in the product failed because of a string mismatch (high) or because it could not read anything (low).
Output24-Output 31 Output48-Output63	Outputs that can be set through Scripts. See the DVT Script Reference Manual for more details.

Unlike for input functions, in addition to assigning specific functions to the I/O pins, the user can specify other parameters to ensure that the outputs are correctly received by another system. FrameWork offers extra flexibility for the outputs to determine when and how the outputs are made available. The use of these parameters depends on the hardware device providing the connection to the SmartImage Sensor. These configurable parameters are:

- Invert Polarity: any line configured as an output on a SmartImage Sensor system is active in the high state. The “Invert Polarity” option makes it active in the low state.

- Timing: outputs from the SmartImage Sensor can be made available immediately after inspection or at a fixed delay after the trigger signal.
- Pulse Width: outputs can remain high until the next inspection, or simply stay high for a predefined period. The user can also force a separation between consecutive outputs (useful when working at very high speeds).

Other Communications Settings

There are several ways to get data into and out of SmartImage sensors including Ethernet and serial communications. Various drivers for specific robots are also available. For specific information on these methods see Chapter 5 – SmartImage Sensor Integration.

Background Scripts

Background Scripts are system-wide programs that can run independent of the Inspection Product. Some tasks assigned to Background Scripts are Product selection for an inspection, register reading and writing, exposure time control, communications, etc. There are important changes to the overall syntax of scripts in FrameWork 2.6. To find out about these changes and get more information on Background Scripts, consult the DVT Script Reference Manual, available from DVT under the part number DOC-SCR or from the DVT website for free download at http://www.dvtsensors.com/support/DownloadsManagerNew.php?KW=Manual.

Power-on Product

The Power-on Product allows the system to begin running inspections as soon as the camera is turned on and "alive". Only Products that have been saved to flash memory may be assigned as Power-on Products because those that are only in RAM memory will be lost on power-down. The Power-on Product assignment is critical when the SmartImage Sensor is used as a stand-alone system. Once the system is configured with a PC, the Power-on Product must be assigned in order for the sensor to start running inspections (thus signal digital outputs) immediately after cycling the power.

In selecting the Power-on Product at start-up, the SmartImage Sensor actually completes 2 tasks. First, the system selects the named Product for inspections (called the Current Inspection Product). Second, the SmartImage Sensor sets the Inspection State to Running.

Note: A SmartImage Sensor without a Power-on Product will not start running inspections after power-up.

Trigger Source

Another system parameter is the trigger source. This parameter indicates how the system is triggered. The options are external trigger and internal trigger. External trigger requires that the trigger input function be assigned to an I/O line and that an external device provide a signal on that input whenever an image is to be taken. Internal trigger is an automatic mode, in which the camera takes a new image periodically at fixed user-defined intervals or simply at fast as possible.

Note: When working at full speed the SmartImage Sensor may show a slow response to communications because the inspections take priority over other processes.

When external trigger is the option for triggering the system, the user can define a fixed delay after trigger if the detection of the part to be inspected cannot be made at the precise moment that the part is in front of the SmartImage Sensor. This way, the user can have a sensor detecting the position of the part, and triggering the system, but the SmartImage Sensor will not acquire an image until the part is precisely in the correct position.

FOV Balance

Starting in FrameWork 2.6, FOV balance will include two different operations: White Image Normalization and Dark Image Subtraction (available for the SpectroCam.)

White Image Normalization is the process that was known as FOV balance in previous releases. When the illumination is not uniform and the physical setup cannot be improved, doing a White Image Normalization may be helpful. The user can select to calibrate (automatic process) and activate a white normalization of the field of view in a certain region of the image (or over the entire image). This will correct the pixel intensities to make the illumination seem even and well-distributed across the image or the selected region. It should be noted that this normalization alters the information in the pixels, thus the image itself. In general, uneven light should be made even by changing the physical setup (position, source, etc.) instead of using this option. This procedure is not recommended for precision measurement applications, in which the information contained in every pixel may be vital for the success of the inspection.

The Dark Image Subtraction is only available for the SpectroCam Sensor. It is generally used in applications where the exposure time is high. When this is the case, some regions of artificially high intensity levels may appear due to certain characteristics of the CCD. To compensate for this occurrence, Dark Image Subtraction may be applied by covering the lens with a lens cap and pressing “Calibrate”. This calibration can then be used in the Spectrograph SoftSensor by selecting the option “Subtract Black Image”. This will eliminate the aforementioned unwanted effect by subtracting the learned black image from the live region of interest of the SoftSensor. This process does not affect the original image since it is applied just for the processing of a particular SoftSensor

Note: Although the White Image Calibration is a system parameter, its activation is product-specific.

Sensor Gain

The sensor gain adjustment makes your SmartImage Sensor’s CCD more sensitive to light. Gain has a maximum value of 25 and is initially set to 2. A Gain of approximately 2 generally works best for the Legend Series 500 and a value of 4 is appropriate for the Series 600.

Hint: Is your application on a constantly moving conveyor line? Effective use of the gain setting allows you to minimize blur in your images due to relatively long exposure times. With parts moving past the camera, increase the gain, then decrease the exposure time until the images no longer appear blurry.

Reflectance Calibration

The goal of Reflectance Calibration is to have the measured reflectance of the camera match an external device. Color monitoring, Segmentation and 1-D ANSI barcode grading are the sensors that can benefit from this. Note that Reflectance Calibration can be performed for gray scale cameras as well as color.

There are two reasons why you may need to have Reflectance Calibration for color cameras. One reason is to have more accurate relative color measurements with Color Monitoring and/or Segmentation. The absolute color values themselves would not be accurate but the relative ones are closer to reality. Another reason why you may need Reflectance Calibration is to match two cameras to each other. If they are both calibrated with a single target, the color values they measure will match. For more information on this procedure refer to the integration notes section of the DVT website or the FrameWork help file.

Spectrum Calibration

Spectrum Calibration is only available for the SpectroCam. It is the process through which pixel data is translated into wavelength information. When working with the SpectroCam sometimes pixel information is not descriptive enough, so this calibration option is available. The process will involve the use of a part or illumination source with known peak wavelengths. After the pixel location and wavelength is entered for each of these peaks, the system interpolates them to obtain a transformation model for each pixel in the region of interest. After the calibration is done, it will be used by all Spectrograph SoftSensors. For more information on this procedure refer to the FrameWork help file.

Register Design Setup

Through the Register Design Setup process, users can create one or several custom print registration calibrations. A print register design is a group of register marks that follow a certain pattern. After a design is added, it can be used by any Segmentation SoftSensor present in any product inspection in the SmartImage Sensor. The designed pattern can be used by selecting the option “Abs. Register” as the rule type under the “Rules” tab of the property dialog box of the SoftSensor. For more information on this procedure refer to the FrameWork help file.

Product Parameters

The next hierarchical level inside SmartImage Sensors is the Product Level. At the product level, we have independent subsystems that are responsible for specific inspections. These subsystems are called products. All the products in a system use a common set of system parameters (as discussed before). Since every product is responsible for a specific inspection, products need to have control over the imaging process. That is why parameters such as illumination, image window, and exposure time are Product Level parameters. We will now discuss them.

Exposure Time

Exposure time refers to the time that the electronic shuttering mechanism exposes the image-acquiring device (CMOS device for the Legend Series 520, CCD for the other systems) to light. The longer the exposure time, the more light enters the device. Three factors will determine the exposure time you set for an inspection:

- The overall speed of the parts. For high-speed parts, short exposure times are recommended to minimize blur within the images.

- The overall inspection rate (parts per minute). In situations where the overall part rate is high, exposure time should be minimized to optimize the inspection.

- The available light. Brighter lighting fixtures enable shorter exposure time whereas dimmer ones require longer exposure time.

There may be instances in which you must reduce your exposure time. You have the following options to help reduce exposure time and still maintain overall image intensity:

- Increase the light aimed at the inspection area.

- Increase the gain on the CCD. Gain is the ratio of output over input, or, in this case, the overall increase in image brightness. Increasing the gain makes the CCD more sensitive to light. **Caution: high gain settings may produce a grainy image, which may affect sensitive inspections**.

- Use a lens with an adjustable aperture. Many C and CS mount lenses have aperture controls.

Notes: An aperture is the hole that lets light through the lens onto the imaging surface. The larger the hole, the more light can strike the CCD. Opening the aperture may reduce the depth of field. Doing so will make it more difficult to focus on objects if they vary in distance from the lens.

Increasing the exposure time slows down image acquisition process and will increase image blur for moving inspections.

Digitizing Time

This parameter determines the precision of the analog to digital conversions in the image acquisition device. The slower digitizing time allows for more precise conversions. It is recommended to use the default value. Visible changes in the image are not likely to be observed when changing this option. Depending on the system, users can choose between different digitizing times to speed up the process or to get the best possible image. A parameter directly related to the digitizing time is the partial window acquisition (explained below). When partial image acquisition is being used, the acquisition time decreases to a fraction of the selected digitizing time. This fraction is proportional to the size of the image being acquired.

Partial Image Window

Some applications may not need to process the entire image, but just a portion of it (if the part to inspect is consistently located). SmartImage Sensors allow the user to select a specific window within the image so only that window is acquired.

One portion of the total inspection time is for image acquisition. The SmartImage Sensor CCD, with a resolution of 640x480 pixels takes approximately 13ms to transfer the image from the CCD into memory. This time is proportional to the number of pixels being acquired. Reducing the acquisition window to 320x240, for example, would cut the acquisition time to approximately ¼ the original value.

Note: For the 520 system, the digitized area will correspond to the partial image coordinates specified by the user. On the 530 and 540 systems, the area to the left of the image window will still be digitized as it is shown in Figure 7 **with the dotted lines. This will be true for the regions both right and left of the acquisition window entered by the user for the 542C, 544 and 544C systems.**

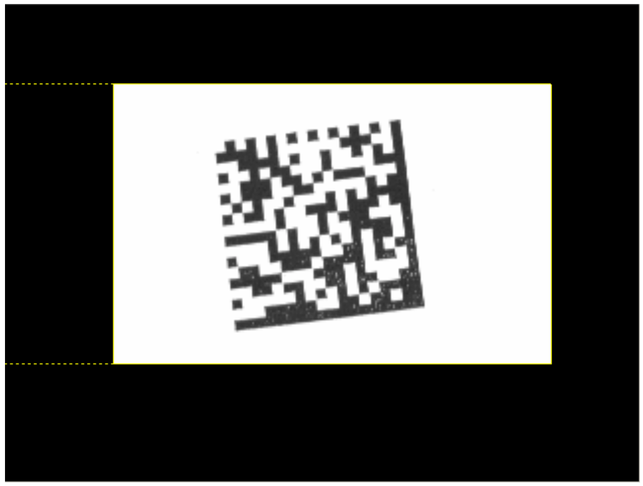


Figure 7: Partial acquisition window for 530 and 540 systems.

Antiblooming

Blooming effect occurs when pixels absorb so much light that they saturate and cause neighbor pixels (mostly in the vertical direction) to saturate even though their original intensity level was below saturation. When Antiblooming is turned on, local spots of overexposure are reduced, thus minimizing streaks and washed out areas of the image. Antiblooming will correct the image but will add to the overall Image Acquisition time as a result. Blooming is not an issue for the Legend 520 system because of light accumulation characteristics. Antiblooming is always enabled for the 542C, 544 and 544C systems.

Warning: Do not enable Antiblooming when working with the Measurement SoftSensor when using sub-pixel computations. Results may be affected by as much as one full pixel when Antiblooming is activated.

Illumination

The illumination signal is a digital output, normally configured on pin 8 in a Legend Series 500 SmartImage Sensor and on pin 12 of a Series 600 SmartImage Sensor. Strobe light sources from DVT, with part numbers of the form IDxx are designed to use the strobe illumination signal (see Figure 8).

Figure 8: A strobe illumination source (IDRA) that uses the strobe output.

Pulse width for the strobe signal is determined by the Product exposure time (which is a Product-specific parameter). These two signals are tied together so the strobe light remains on for the duration of the CCD exposure to light.

For Legend Series SmartImage Sensors, this option of illumination enabled/disabled controls the integrated LED ring as well. This will allow the user to gain some control over illumination without using an I/O line. However, when more specific lighting is needed, up to three external strobe lights (besides the integrated LED ring) can be controlled using the digital I/O lines.

Product Identification

Every product in a SmartImage Sensor can be assigned a number, called the Product Digital ID, which is used to reference the product. This number enables the user to send a digital signal to the SmartImage Sensor to indicate which product to use for a specific inspection. In order to do that, a number of I/O lines must be dedicated to the Product ID bit input function. The number of lines dedicated to this function determines the maximum number of products that can be selected externally. Basically, n lines dedicated to Product ID bits will enable the user to select among 2^n products. A common configuration for I/O setup involves designating 4 lines as Product ID bits (bits 0 through 3). The following table reviews the binary combinations for 16 Product ID numbers (binary 0-15) using these 4 input bits:

Product ID	Binary Pattern (Bit 3, Bit 2, Bit 1, Bit 0)	Product ID	Binary Pattern (Bit 3, Bit 2, Bit 1, Bit 0)	Product ID	Binary Pattern (Bit 3, Bit 2, Bit 1, Bit 0)
0	0,0,0,0	6	0,1,1,0	12	1,1,0,0
1	0,0,0,1	7	0,1,1,1	13	1,1,0,1
2	0,0,1,0	8	1,0,0,0	14	1,1,1,0
3	0,0,1,1	9	1,0,0,1	15	1,1,1,1
4	0,1,0,0	10	1,0,1,0		
5	0,1,0,1	11	1,0,1,1		

Product Selection

Digitally selecting a Product is a simple process. Refer to Figure 9 for a diagram illustrating the following procedure. First, set the binary bit pattern of a Product's ID number on the "Product ID Input" lines. When the Product ID bits are loaded (in this case Product ID 1, which is 01 in 2-digit binary code) the Product Select line should become active. This tells the SmartImage Sensor to read the data bits from the Product ID bit lines. When this signal is sent to the system, a response could be read indicating a successful/unsuccessful operation. Two outputs are dedicated to alerting the user of the success or failure of the digital selection process: Product Select Pass and Product Select Fail. Reading one of these should be enough to determine the success of the product selection.

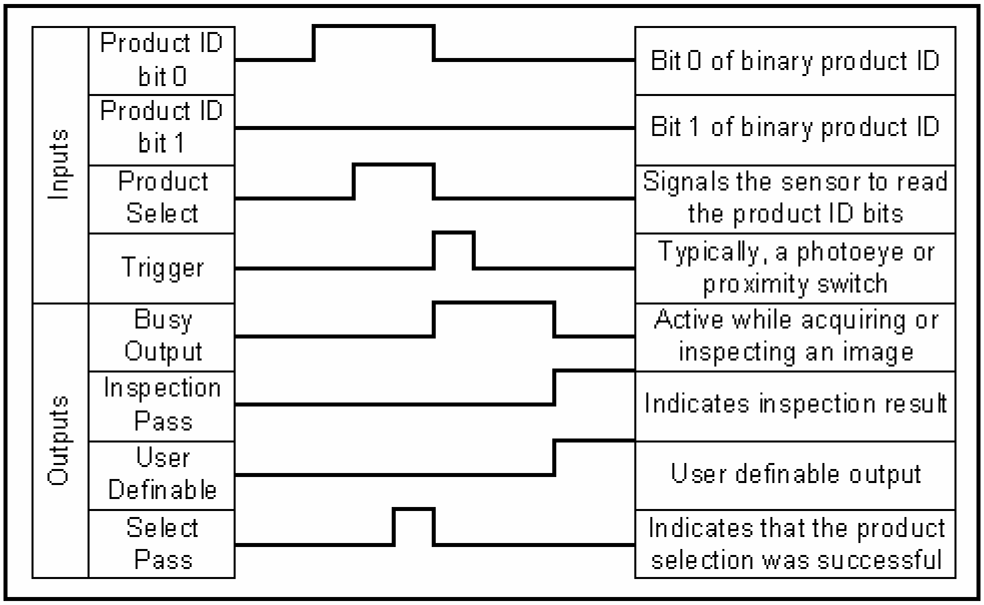


Figure 9: Select a Product using the digital inputs. The above diagram shows the I/O timing of the Product Selection process.

For example if a Product ID of 3 is selected but the camera has only products with ID's 0, 1 and 2, Product Select Fail should be high and Product Select Pass should be low. When the product selection has been completed the system can be triggered for an inspection.

Product Graphs

Product graphs help users troubleshoot systems as well as gather statistical data about the inspections. The graphs available at the Product level are: Pass/Warn/Fail, Result Table, Inspection Statistics, Inspection Times, and Digital I/O. These features are part of the FrameWork user interface; they do not reside in the SmartImage Sensor.

The Pass/Warn/Fail graph keeps count of the inspection results. It Shows the actual counts of Fail, Pass, and Warn outputs as well as an indication of the percentage of the total inspections taken that those counts represent. The graph shows the number of inspections as the abscissa (X axis) and the running percentages of inspection results as the ordinate (Y axis). Figure 10 shows a Pass/Warn/Fail graph.

Figure 10: Sample Pass/Warn/Fail graph.

The Result Table is a list of all the SoftSensors in the product. It consists of three columns: the name of the SoftSensor, the result for that specific SoftSensor and a customizable third field about that SoftSensor. The user can select what to see in that third column from a number of options: Output Value (the default output of the sensors), Count or Percentage of Pass/Fail outputs for that SoftSensor, Cause of Failure (displayed only when SoftSensor fails), and SoftSensor type. Figure 11 shows a sample result table. It indicates the product name, in this case “ORingDetection”, and it shows the SoftSensors being used as well as their outputs.

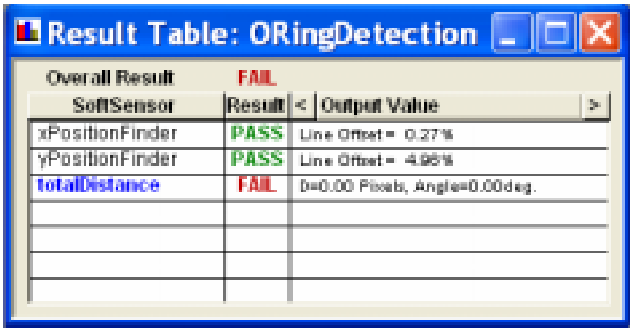


Figure 11: Result Table for a product. It lists all the SoftSensors in it as well as their outputs.

The inspection Statistics graph is particularly useful for troubleshooting. This graph keeps count of total inspections, inspections that timed out (the inspection did not complete in a certain amount of time), and inspections that could not be taken when trigger was received. The last type of count refers to two cases: resource conflict (the SmartImage Sensor is triggered while it is acquiring an image) and missed count (the SmartImage Sensor was triggered while the image buffer was full so the trigger was ignored).These parameters help users determine when there might be a timing issue with the inspections. Figure 12 shows an Inspection Statistics Graph.

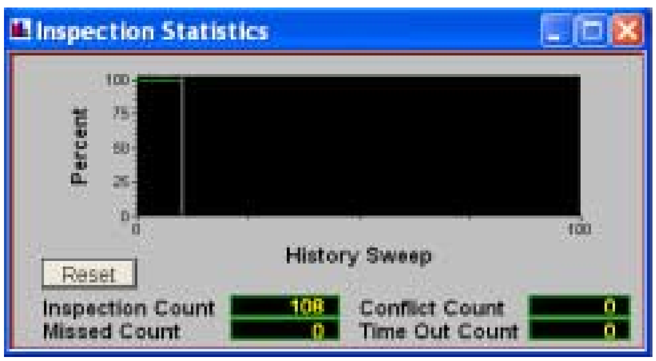


Figure 12: Inspection Statistics graph.

The inspection Times graph is a very important graph. It shows how much time every inspection takes showing the maximum and minimum inspection times. This graph should be used when two or more different methods for inspections are being tested. Usually, the time it takes to perform an inspection is crucial, so after setting up different SoftSensors to perform a certain inspection, this graph will tell the user which method is the fastest one. Figure 13 shows a screen capture of this type of graph.

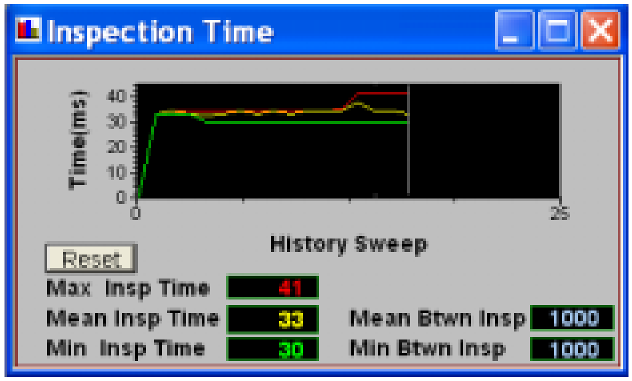


Figure 13: Inspection time graph.

Finally, the digital I/O graph works like an oscilloscope. It uses time as the abscissa (X axis) and eight configurable I/O lines as the ordinate (Y axis) to show a timing diagram that indicates when signals go active or inactive. This graph is very useful for troubleshooting communications. Figure 14 shows a Digital I/O graph.

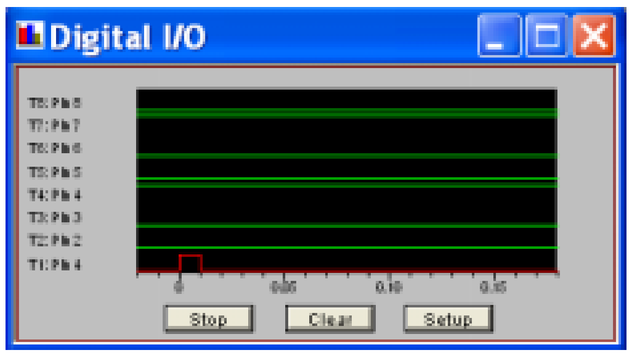


Figure 14: Digital I/O graph. The X axis indicates time in seconds.

SoftSensor Parameters

SoftSensors are the "working class" inside SmartImage Sensors. Each one is assigned a specific task to extract information from the image. The combination of all the tasks performed by the SoftSensors makes up the entire inspection. The main SoftSensor parameters are the threshold, PASS/WARN/FAIL conditions, SoftSensor shape, and processing options.

Threshold

SoftSensors use the information from every pixel to inspect images. Every pixel contains a certain intensity level ranging from 0 to 100. Based on application specifications, the user selects a threshold level to classify the pixels contained in the sensor as dark pixels (below threshold) or light pixels (above threshold). There are several ways to compute a threshold:

- **Intensity Based, Fixed Value:** This option lets the user manually select the threshold level. This threshold has to be a number between 0 and 100 and it is considered a static threshold because it does not change from image to image.
- **Intensity Based, Percent of Path Contrast**: This is a dynamic threshold that places the threshold at some location between the minimum and maximum intensity values based on the value set by the user. This threshold is calculated as follows: $Threshold = (I_{MAX} - I_{MIN}) \times T\% + I_{MIN}$. The "*T%*" represents the threshold percent set by the user, I_{MAX} and I_{MIN} are the maximum and minimum intensity values inside the area that the SoftSensor is set to analyze. This option selects a threshold which can

adapt itself from image to image (dynamic threshold) which makes it more robust that any static threshold.

- **Intensity Based, Percent of Reference Intensities:** This is similar to Percent of Path Contrast threshold, described above. The threshold is calculated as follows: $Threshold = (I_{MAX} - I_{MIN}) \times T\% + I_{MIN}$. However, I_{MAX} and I_{MIN} are not the max and min intensity levels along the SoftSensor path. Instead, the user selects references. This option allows users to name other SoftSensors as high and low intensity references from other regions of the image. When this option is enabled, a new menu prompts the user for four options. Two SoftSensors and two intensity levels from those SoftSensors (such as maximum intensity, minimum intensity, etc.) need to be determined. The same SoftSensor can be used for both options. Note: When using this threshold type, the values reported for the SoftSensor (Contrast, Minimum Intensity, and Maximum Intensity) refer to the selected references, which are the ones used for computation of the threshold.
- **Intensity Based, Auto Bimodal:** This method automatically calculates a single threshold value to use based on the entire histogram. Since it uses all the values in the area to calculate the threshold, the effects of noise and specular reflections are minimized. Auto bimodal thresholding is recommended when there is a distinct foreground and background intensity. The threshold is recalculated for each image.
- **Intensity Based, Adaptive:** The system calculates a specific threshold value for different areas of the image depending on the contrast found in those areas. The user needs to specify a window size in pixels, the software then calculates a threshold (using the percent of path contrast method) for individual windows, making the overall threshold more robust when lighting and contrast are not uniform. This method is the slowest of the thresholding methods.
- **Intensity Based, Dominant Color:**T type of threshold is used in combination with a color SoftSensor. The color SoftSensor learns a color (it could be a shade of gray if using a grayscale system) that is referenced by another SoftSensor to establish a threshold. Basically, pixels of that color, or shade of gray, are considered to be above threshold, the rest of the pixels are considered to be below threshold. This concept will be explained later in the manual.
- **Gradient Based:** This method for calculation of a threshold uses the pixels intensity gradient. That is, it compares pixels with their neighbors in the image to determine local changes in intensity. It then produces a gradient graph that has peaks where the highest changes are located. The user needs to specify a threshold to separate the main edge(s). The methods for determination of the threshold are: Fixed Value, Percent of Path Contrast, and Percent of Reference Intensities. These methods work in the same way as for Intensity based threshold (as explained above). **NOTE: This method uses pixel interpolation to calculate the best position for the edge, providing sub-pixel accuracy.**
- **Gradient Based: Edge Detection:** This threshold type is available for the Blob Generator (see page 90) SoftSensor. It utilizes a method for extracting edges similar to the one used in the Segmentation (see page 103) SoftSensor. Scanning in both directions (horizontally and vertically), it compares pixels with the neighboring pixels in the image to determine local changes in intensity. This allows the SoftSensor to better identify edges and subsequently regions within or outside those boundaries. This type of threshold works for both grayscale and color systems. When using this option, the edges of the blobs found will be considered light blobs, while the regions outside and inside the defined boundaries will be considered dark blobs. Users will have the option to eliminate the background from being considered a blob by importing the color data from a Pixel

Counting SoftSensor and selecting dark blobs. In this case, the edges and the background areas will be considered light blobs and all other regions will correspond to dark blobs.

- **OCR specific:** the OCR SoftSensor has three extra options for calculation of the threshold that will be discussed later on when the OCR SoftSensor is covered.

SoftSensor Graphs

Just like at the Product level, FrameWork offers graphs to display Sensor-level information. This information is sensor-specific; it could be generic information about the sensor (Pass/Fail, etc.) or very specific SoftSensor information. The latter depends on the SoftSensor being used. The first and most common sensor graph is the Pass/Warn/Fail graph. It serves the same purpose as its product-level counterpart, but the data refers to a specific sensor (see Product Graphs for more information). The rest of the SoftSensor graphs could be classified into two main categories: Data Graphs and Image Graphs.

Data graphs display data over time. That is, they have inspections as their abscissa (X axis) and a specific set of data from the sensor as their ordinate. Some examples of these graphs are Measurement, Intensity/Threshold, and Position. Figure 15 shows an Intensity/Threshold graph. Some variation in the values over time can be observed from the graph, but the current values are displayed at the bottom.

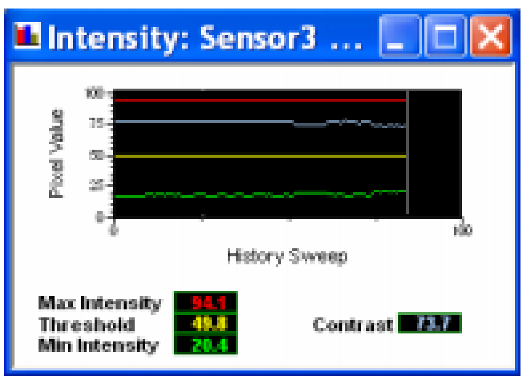


Figure 15: Intensity/Threshold graph showing specific information about the area analyzed by the sensor.

Image graphs are graphical interpretation of the areas analyzed by the SoftSensors that help the user set a threshold. There are three types of image graphs: Pixel Graph, Histogram and Gradient Graph. The Pixel Graph is the simplest way of displaying information about a SoftSensor path. It uses the SoftSensor path as the abscissa, and the intensity of the pixels as the ordinate. That is, the range displayed in the X axis goes from the origin to the end of the line SoftSensor, whereas the range in the Y axis goes from zero intensity (black) to 100 (white). Figure 16 shows the pixel graph for a SoftSensor with a length of 165 pixels.

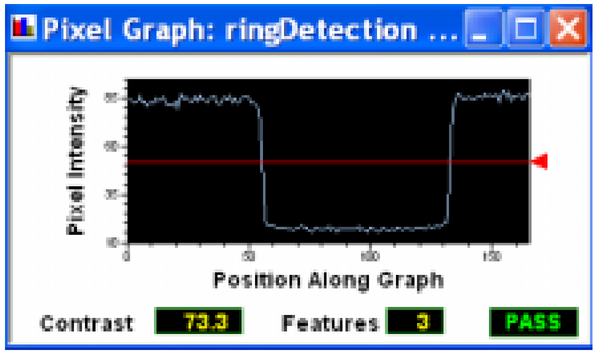


Figure 16: Example of a Pixel Graph for a line SoftSensor of 165 pixels in length.

As the graph illustrates, the pixels near the origin of the SoftSensor are bright (high intensities), then, a dark region is found so the graph drops down to the low intensities and then it goes back up towards the high intensities as it approaches the end of the SoftSensor.

Histograms are also used to display the intensity of the pixels, but in bar chart format. Unlike pixel graphs, histograms use the intensity levels (ranging from 0 to 100) as the abscissa; for the ordinate, they use the percent pixels. That is, histograms display the percentage of the pixels found at every intensity level. Figure 17 shows a histogram for an intensity SoftSensor. The histogram shows two well defined groups of pixels which may represent background (lighter pixels) and the actual part (darker pixels).

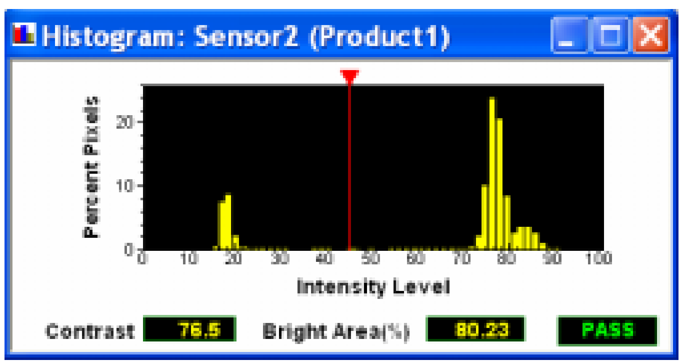


Figure 17: Screen capture of a histogram.

The third image graph is the gradient graph. This type of graph plots changes in intensity values from pixel to pixel along the path of the SoftSensor. It does not show the actual intensity value of the pixels, just the changes. Mathematically, this represents the absolute value of the first derivative of the pixel graph. Figure 18 shows a pixel graph and a gradient graph for a certain SoftSensor.

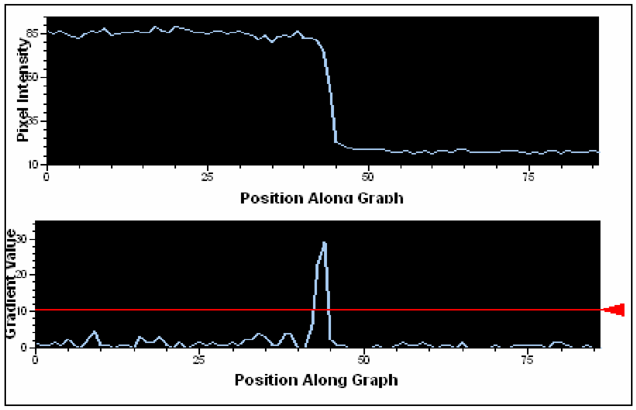


Figure 18: Pixel and gradient graphs for a SoftSensor.

Notice how the gradient graph peaks where the pixel graph shows the biggest change in intensity levels.

SoftSensor Result

SoftSensors can be set to PASS, WARN, or FAIL based on predefined criteria. Usually, the PASS condition is assigned when the parameter being checked is within the specifications. The FAIL output is reserved for cases where the parameter being checked exceeds the specifications. For cases where the feature being checked by the SoftSensor is approaching the limits of the specification, the user may need a signal to correct the production process without rejecting any parts. In those cases the WARN condition should be used. The WARN output will not reject a part, but it will let the operator know that the production process needs attention. Every SoftSensor contains a custom set of rules for the user to set as the PASS/WARN/FAIL criteria. Those rules are related to the specific functionality of the SoftSensor. That is, for measurement SoftSensors the condition would be based on maximum and minimum size, for the barcode reader it would be set based on a matching code, etc. However, there is an option that is common to most SoftSensors: Minimum Contrast. The option of Minimum contrast should be used when a dynamic type of threshold is being used. The use of this option will be illustrated with an example: suppose the user selects Percent of Path Contrast as the method for calculation of a threshold. The SoftSensor (as explained before) will determine the intensity levels of the brightest and the darkest pixel, and based on those numbers it will compute a threshold level. This is a very powerful method, but what happens when the part is not present? Only the background is present, so the SoftSensor sees a very low contrast given by the noise in the background. Based on this contrast the SoftSensor still computes a threshold for the image. Figure 19 shows the pixel graph for a SoftSensor illustrating this case.

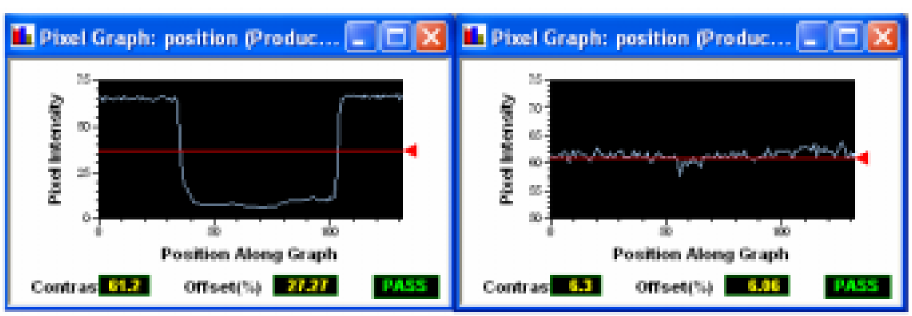


Figure 19: Case where the part being inspected (determined by the group of dark pixels in the graph on the left) is not present in the next image. The graph on the right shows that there is still a threshold being calculated.

The pixel graph on the left shows the presence of the (dark) part. The threshold is correctly calculated at 50% of the difference between the lightest and the darkest pixel. The pixel graph on the right shows the pixel graph for the same SoftSensor for the case where only background is present in the image with intensity levels between 55 and 63. In this case, there is no part, but the SoftSensor still computes a threshold because it can still find a lightest and a darkest pixel. In order to avoid this case, the user can select the "Minimum Contrast" option and set a minimum contrast that must be matched to even compute a threshold. For the second case explained above, a minimum contrast of about 10 would prevent this SoftSensor from computing a threshold and thus will cause it to fail.

SoftSensor Shape

SoftSensor creation is a very simple point and click process. The first step is to draw a certain shape in the image. The available shapes depend on the type of SoftSensor being used, and the selected shape depends on the part being inspected. The two main categories of SoftSensor shapes are Line SoftSensors and Area SoftSensors.

Line SoftSensors are mostly used when edges need to be found. These types of SoftSensor scan along lines analyzing every pixel along them. The different types of lines available in FrameWork are shown in Figure 20. Starting from the upper left corner, the shapes are:

- Horizontal/Vertical Line: to be used when the SoftSensor is to be drawn perfectly aligned with the pixel grid, either horizontally or vertically.
- Line at an angle: allows the user to draw a line at any desired angle
- Polyline: allows the creation of a line defined by straight segments, used for analysis of very complicated shapes
- Freehand Line: the line is drawn following the movement of the mouse, used for analysis of very complicated shapes
- Square: draws four sides of a square, good for analyzing the outside of an object with 90 degree symmetry
- Rectangle: draws four sides of a rectangle, good for analyzing the outside of an object with 180 degree symmetry
- Polygon: similar to polyline but it closes the figure, used for complex shapes
- Circle: draws a circular line, used for analysis of round or highly symmetric shapes
- Ellipse: draws an ellipse, designed for cases where a circle is to be analyzed but perspective errors make circles appear as ellipses
- Circular Arc: draws an incomplete circle, used for cases where only part of the circular region is to be analyzed
- Elliptical Arc: draws an incomplete ellipse, used for cases where only part of the elliptical region is to be analyzed

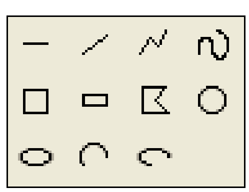


Figure 20: Available shapes for line SoftSensors.

It should be noted that even though some lines are closed, that is, they draw a figure, the SoftSensors are line SoftSensors and they only analyze the edges of those figures. Only area tools analyze areas.

NOTE: For instructions on how to draw a specific type of SoftSensor select the desired shape and refer to the FrameWork status bar for click-by-click instructions.

Area SoftSensors analyze the pixels contained in a specific area of the image. That area is determined when the SoftSensor is drawn. The main types of area SoftSensors are shown in Figure 21.

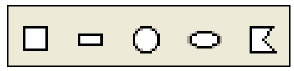


Figure 21: Available shapes for Area SoftSensors.

As the image shows, the shapes are equivalent to those of line SoftSensors. The different shapes are applicable to different cases; thus the user should be familiar with the application and the part to be inspected and then select a shape accordingly. An important difference between line and area SoftSensors is the processing time. While area SoftSensors might be the best choice to make the inspection reliable, they take more processing time than line SoftSensors. Fortunately, FrameWork is designed in a way that this change in processing time is very small and it does not become an issue for most applications.

NOTE: The main difference between icons for line and area SoftSensors is that those for area SoftSensors are white on the inside, highlighting the fact that they analyze the area delimited by the line.

Besides the basic line and area SoftSensors, FrameWork has other specific shapes available for certain applications. Those shapes will be covered as the specific SoftSensors that use them are explained in the next chapter.

Processing Operations

Image quality is always the most important factor for setting up an inspection. Sometimes a minor change in the lighting technique makes a big difference for the image. However, in many cases, users have limited control over the physical setup of the system, so another way to deal with noisy images must be used. For those cases, some SoftSensors have a built-in number of operations that process the image before a threshold is applied. These operations do not actually alter the image; they alter the perception of the image for the selected SoftSensor. Since SoftSensors can overlap, altering the actual image would affect the result of every SoftSensor analyzing it. The available operations are:

- Erode: shrinks shapes in the image a user-specified number of pixels. Useful operation when dealing with background noise
- Dilate: expands shapes in the image a user-specified number of pixels. Useful operation when dealing with noise inside the part
- Open: combination of Erode operation followed by Dilate operation. This option eliminates background noise and does not alter the sizes of the parts
- Close: combination of Dilate operation followed by Erode operation. This option eliminates noise inside the part and does not alter the sizes of the parts

Figure 22 illustrates the use of the Erode and the Open operations. In this case, the dark part being analyzed is in presence of background noise.

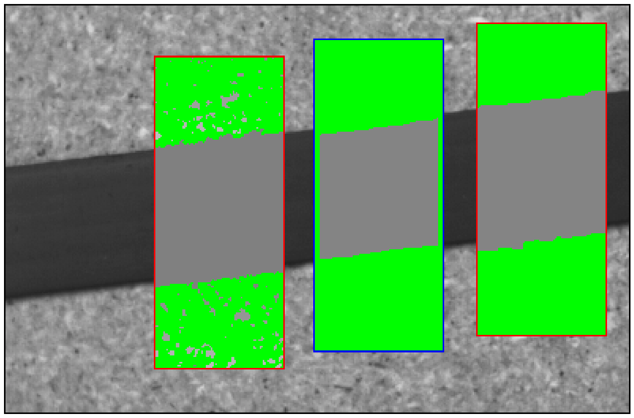


Figure 22: Noisy image illustrates use of Erode and Open operations. The SoftSensor on the left uses neither, the one in the center uses the Erode operation, and the SoftSensor on the right uses the Open operation.

The use of a SoftSensor with no options of image processing would result in an inspection of the part and the noise as shown in the first SoftSensor. The second SoftSensor uses the Erode operation, it shrinks everything (3 pixels in this case) until the noise is gone, but that affects the original size of the part. The third SoftSensor uses the Open operation, that is, after shrinking the parts it expands them. The result is the noise disappearing and only the part to be inspected being expanded to its original size.

Figure 23 shows the inspection of a part with texture. The texture causes some noise inside the part; the first SoftSensor shown does not use any operation, so it picks up all the noise.

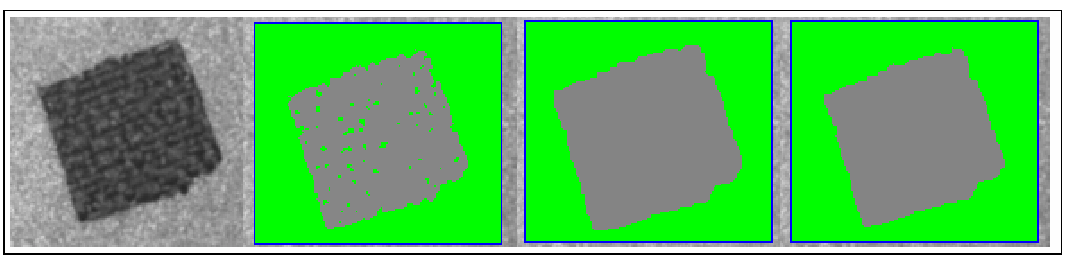


Figure 23: Example where the operations of Dilate and Close are used.

The second SoftSensor uses the operation Dilate, which expands the part in all directions. This helps with the noise but alters the area of the part. The last SoftSensor uses the operation of Close and it successfully eliminates the noise and returns the part to its original size.

The user needs to remember that all these operations add some processing time. The operations of Open and Close add twice as much processing time as the other two operations because they perform the basic operations twice.

Digital Relearn

Digital relearn is an option that is present in many SoftSensors. It refers to cases where the inspection consists of comparing something to a learned version of it. Its use extends to the following SoftSensors: Template Match, ObjectFind, OCR, 1D and 2D Readers, Color, and Spectrograph. When these SoftSensors are created, something is learned (a shape, a code, a color, etc.). When the SoftSensor performs inspections, it uses the learned model. If the application happens to change the elements being used, there is no need for the user to connect to the SmartImage Sensor using FrameWork and reconfigure the SoftSensor. A simple digital signal to the SmartImage Sensor can be used to indicate that a new element is ready to be learned. In that case, the system takes the element (color, code, shape, etc.) from the next image to be inspected and stores it as the new model.

FrameWork Software

Now that the user is familiar with the system, product and sensor level parameters, we will discuss how to access them. This section explains the basics of the FrameWork software. After reading this section the user should be able to access all the parameters explained before and should be more comfortable with the system. FrameWork consists of two main parts. One of them is the FrameWork firmware which resides in the SmartImage Sensor. The FrameWork firmware is the responsible for all the computations that happen inside SmartImage Sensors. The user has no direct control over it. The user only programs the user interface and this one translates everything for the firmware to execute. The second part of FrameWork is the software. The FrameWork software consists of the FrameWork user interface and the SmartImage Sensor hardware emulator. Figure 24 shows the relationship amongst the different components of FrameWork. The user can only access the components that reside in the PC: the FrameWork user interface and the hardware emulator. Using the FrameWork user interface the user connects to a SmartImage Sensor and makes changes to the product or system files loaded in the SmartImage Sensor. In order to do this, the FrameWork user interface queries the SmartImage Sensor for images and reproduces in the PC what the SmartImage Sensor does in its internal memory with those images.

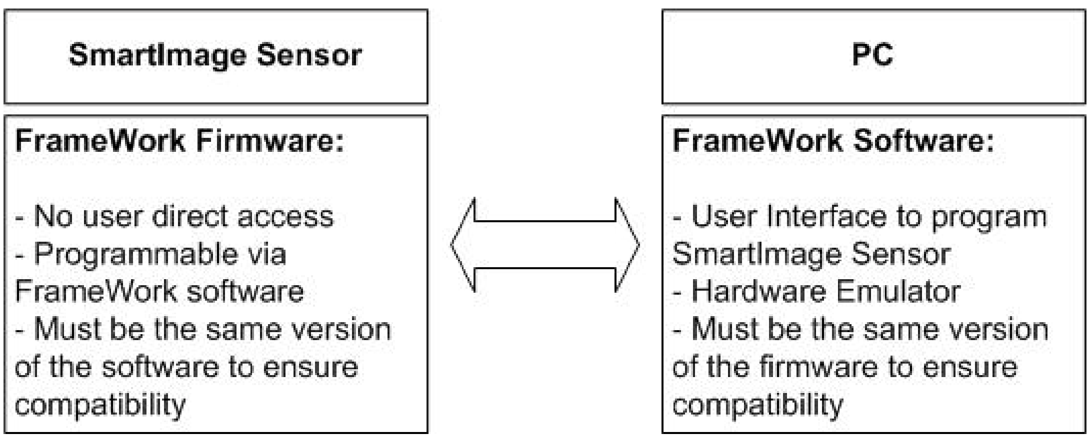


Figure 24: Components of FrameWork and the relationship between them.

FrameWork User Interface

The FrameWork User Interface is a Microsoft Windows® application used for the configuration of SmartImage Sensors. The main areas of the interface are the main menu, the expanded toolbar, the SoftSensor toolbar, the SID, the result panel, and the status bar. Figure 25 shows a screen capture of the FrameWork user interface highlighting these areas.

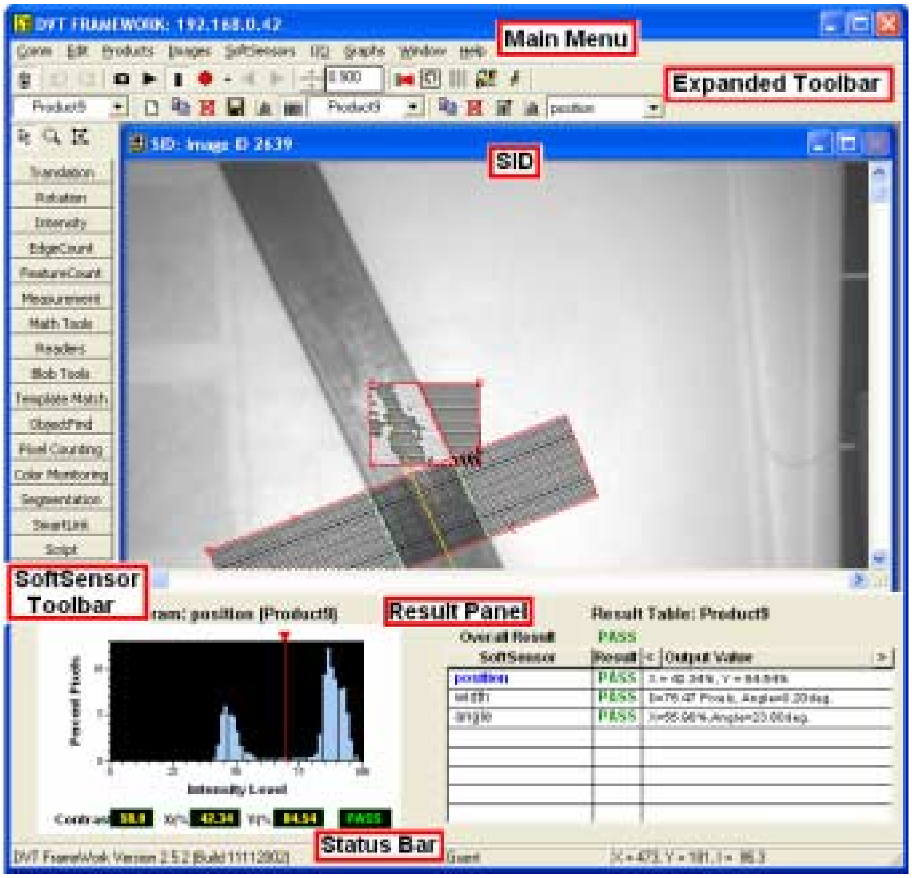


Figure 25: screen capture of the user interface highlighting the most important areas.

The Main Menu contains commands to control most of the parameters discussed so far. Here, we will discuss the functionality of each submenu:

- The "Comm" submenu contains commands to control the connection to the SmartImage Sensor. From here the user can get to the "PC Communications" dialog box to connect to SmartImage Sensors or the Emulator, as well as manage access control (log in, log out, create user accounts, etc.).
- The "Edit" submenu offers access to the following FrameWork features:
    - Background Scripts dialog box: From this dialog box, the user can add, delete, edit, and create background scripts. Please refer to the script documentation for information about background scripts.
    - Terminal Window: Opens an Ethernet connection to the SmartImage Sensor's system terminal (port 3246 for new systems and 5000 for old systems). The camera can receive a number of commands in this port that give the user access to it without using FrameWork. By using this terminal window, the user can execute these functions on the SmartImage Sensor without exiting FrameWork. Refer to the FrameWork help files for a complete list of commands SmartImage Sensors can receive on the system terminal.
    - Script Debug Window: Special window dedicated to receiving Debug statements from scripts that help in troubleshooting the code. Please refer to the script documentation for more information about it.

  - Options: Gives access to advanced features. From this menu the user can enable advanced features for many SoftSensors, set parameters for the SmartImage Sensor itself, and activate/deactivate buttons from the SoftSensor toolbar.

- The "Products" menu lets the user manage the products in the SmartImage Sensor as well as create backup copies of products or the entire system. From this menu the user can select the Power-on Product, assign digital ID's for product selection, store products to flash memory, reclaim flash memory, backup products or systems to the PC's hard drive, and restore them back to the SmartImage Sensor.

- The "Images" menu gives the user control over many imaging parameters. Most options in this menu are very general such as image rotation, display of SoftSensors in the image, copy images from the PC and restore them to the display, go back a number of images (retained images option), and configure image sequences (only applicable to the hardware Emulator). The two most important options are:

  - Display Parameters: This option brings up a dialog box from that enables the user set up recording of images, change the display of images (all images, only failed, etc.), enable/disable image compression for image transfers (to gain speed or concentrate on image quality), and select the type of SoftSensor image marking to display.

  - Image Parameters: This option opens a dialog box that offers more control over the system. From this dialog the user can set the type of trigger (internal or external), the period (in case of internal trigger), the exposure time, the illumination to use (in case the application uses strobe lights), the sensor gain, product gain, the FOV balance, and the size of the image being acquired.

- The "SoftSensor" submenu offers access to SoftSensor parameters. From here the user can edit SoftSensor settings and create SoftSensors.

- The "I/O" submenu allows the user to set up the communications and assign the digital I/O lines to the desired I/O pins. DataLink, a tool available from FrameWork that can be set up to output a configurable string with data after every inspection, is also available from this menu. For more information about communications, see Chapter 5 – SmartImage Sensor Integration.

- The "Graphs" submenu offers access to both product graphs and sensor graphs (explained before).

- The last two options, "Window" and "Help", have standard Windows® functionality. The "Window" option is used to customize the user interface. The "Help" menu brings up the FrameWork help files, which are rich in content and examples.

The Expanded Toolbar contains buttons that offer shortcuts to the most frequently used functions of the software. Every button offers a tool tip, that is, when the cursor is placed over a button, a message is displayed on the screen indicating the functionality of the button. The expanded bar contains three main sections, the upper bar, the product section of the lower bar, and the SoftSensor section of the lower bar. The main controls are in the upper bar. It has a button to connect/disconnect from the system, spin buttons to change the exposure time (which is displayed in milliseconds), a button to control the strobe lights, antiblooming control, white balance control (see Color SoftSensors), and controls for both the SmartImage Sensor and the user interface. The controls for the user interface contain a "Play" button, a "Stop" button, and a "Record" button. The controls for the SmartImage Sensor are the "Trigger" button and the "Run/Stop Inspections" button. These two sets of buttons seem to have similar functionality, but they are very different. The ones for the user interface only determine if the SmartImage Sensor is to send images to the user interface (this is known as Play Mode. The fact that the user interface gets images from the SmartImage Sensor does not mean that the SmartImage Sensor is inspecting them. The only indication that we have of that is the availability of the inspection results from the SmartImage Sensor, and the control the user has over this process is the "Run/Stop Inspections" button. If this is enabled, the SmartImage Sensor is inspecting images, which means that it is making the output signals available. Table 1 summarizes the different cases.

User Interface: Play Mode	Inspection Mode	Real Time Feedback to user interface	Outputs from the SmartImage Sensor
PLAY	OFF	Active (updates images)	Not Available
STOP	OFF	Inactive (stopped)	Not Available
PLAY	ON	Active (updates images)	Available
STOP	ON	Inactive (stopped)	Available

Table 1: Summary of the relationship between play mode and inspections mode.

The user should understand the differences between these two options in order to avoid incorrect configurations. The main reason for the independence between these two functions is that the SmartImage Sensor is a stand alone System. It does not need a user interface connected to it in order to inspect images, so it must not rely on user interface modes to produce inspection results. Likewise, the user does not have to rely on the SmartImage Sensor running inspections to see images, and it does not have to stop the SmartImage Sensor from inspecting to observe a particular image. For example, if the inspection mode was tied to the user interface play mode, the SmartImage Sensor would stop inspecting images every time the user needs to analyze an image in details for which it would be necessary to stop the real time feedback. This would cause a problem when other devices depend on the result from the SmartImage Sensor to accept or reject a part. It should be mentioned that Table 1 summarizes what happens upon triggering the system. That is, when the trigger mode is set to internal with a period, the updating of images/outputs as set by the aforementioned parameters will occur periodically. However, if the trigger mode is set to external, the updating of those parameters will only occur when the SmartImage Sensor receives an external trigger.

In addition to these controls, the upper portion of the Expanded Toolbar contains a Previous Image and a Next Image button. The Previous Image button works only with the Static Image Window. It changes the current image shown to the previous image in the sequence being used.

The Next Image button has two different functions: it can be used in conjunction with the Static Image Window or it can act as an external trigger button. When working with the Static Image Window the next image button changes the current image shown to the next image according to the image sequence. If the system is on external trigger this button is enabled and its functionality affects the SID (thus the Emulator or SmartImage Sensor) by causing it to go to the next image. For the Emulator it means that the next image in the selected sequence is displayed. For the SmartImage Sensor it will trigger the system to obtain a new image.

In order to determine the function of the Next Image button that is used, users need to select the Static Window (for the first function) or the SID (for the second function) by clicking once on it before clicking on the button.

The product section and the sensor section of the lower portion of the Expanded Toolbar contain similar functionality for products and sensors. For products it offers the extra options of showing the result panel (explained below), creating a new product, and saving a product to flash memory. Flash memory is the nonvolatile memory where the product should reside in order to remain in the SmartImage Sensor after the power is cycled. Both sections of the toolbar contain a pull down menu with the existing products in the system and the SoftSensors that the selected product contains. The remaining toolbar buttons are to duplicate, edit parameters, delete, or access the graphs associated with the product/SoftSensor selected from the pull down menu.

The SID, or Sampled Image Display, is where the images obtained from the SmartImage Sensor are displayed. The name describes its functionality; not every image from the SmartImage Sensor is transferred to the user interface. For the SmartImage Sensor the number one priority is to run inspections. After that, and depending on processor usage, it sends images via Ethernet to the user interface. In cases where the SmartImage Sensor is being triggered very fast, the user interface will only receive some images. This is why the display is called sampled image display; it only displays sample images from the SmartImage Sensor. Besides the case of the SmartImage Sensor being busy with inspections, other factors influence the number of images transferred to the user interface such as network traffic, the types of hub being used, the Ethernet card being used, the speed of the processor in the PC, and the type of images being transferred (color, high resolution, etc.). The top bar of the SID displays a number called the Image ID, which is an incremental number assigned to every image. The Image ID is reset on power-up. This number should only be used to monitor the rate at which images are being transferred.

The SoftSensor toolbar is where most SoftSensors are available. This toolbar represents a shortcut to most options available under the SoftSensor menu of FrameWork. Except for the Spectrograph, a recently added SoftSensor, all the SoftSensors can be obtained from the toolbar. If the user prefers to use the main menu, the toolbar can be hidden to maximize the work area by deselecting the “Sensor Tools” option from the Window menu.

The result panel is a combination of two elements that have been discussed in this chapter. The result table and the sensor graph for the SoftSensor that is selected from the result table. Figure 25 shows three SoftSensors in the result table. The selected SoftSensor is the one highlighted in blue, so the sensor graph to the left of the result table corresponds to this SoftSensor (named position).

The Status Bar, the last section of the user interface contains three main sections:

- Instructions section: This is the left side of the bar. In Figure 25 it shows the version of FrameWork being used, but when the user selects to create a new SoftSensor, it displays click-by-click instructions so the user does not need to remember how to draw every SoftSensor.
- User section: This section is related to the SmartImage Sensor administrator, it shows the username under which the current operator is logged on. Figure 25 shows “Guest” because no user has logged on.
- Image information section: This part of the toolbar is used in combination with mouse movements. When the user moves the pointer over the image being inspected, this section of the status bar will show three values for grayscale systems and five values for color systems. The first two values in both cases are the X and Y coordinates of the pixel where the pointer is located; “X” coordinates start at zero on the left edge of the image and increase to the right, and “Y” coordinates start at zero on the top edge of the image and increase downwards. For grayscale systems there is a third value which indicates the intensity of the pixel (between 0 and 100). For color systems there are three values instead of an overall intensity value, these numbers represent the Red, Green, and Blue intensity values of the pixel.

SmartImage Sensor Hardware Emulator

The SmartImage Sensor Hardware Emulator acts like a SmartImage Sensor, but it is a software application that runs on your PC. When the emulator application is running, the FrameWork user interface can connect to the emulator – just like a connection to a SmartImage Sensor. Actually, FrameWork does not recognize a difference between the actual SmartImage Sensor and the Emulator. The main uses for the Emulator are:

- **Education / Training:** With only a PC, users can run the emulator application with FrameWork and practice using the tools to learn how FrameWork operates.
- **Off-line Setup:** Take development of Product files off of the factory floor. Use the emulator to continue setup work on Products even after you've walked away from the machine.
- **Troubleshoot Reliability Problems:** Is your SmartImage Sensor rejecting too many good parts or passing flawed parts? Save a batch of the images and work off-line to find the problem, and then perform the necessary changes to improve the inspection and reload the product file to the SmartImage Sensor.
- **Demonstrations:** Show the power of FrameWork without the SmartImage Sensor hardware present.

FrameWork's functionality is opened up only after a connection is established to a SmartImage Sensor or to the Hardware Emulator. When the application starts, the PC Communications dialog box is opened. Users may opt to connect to a SmartImage Sensor (hardware device) via Ethernet (or serial port for older systems) or to the Emulator. Under Emulator, the user will find a number of options. Each one corresponds to a different model of SmartImage Sensor. The selection of emulator depends on the SmartImage Sensor that the user needs. When the user selects to connect to the Emulator, a new Windows® application launches in the background. The user should not attempt to modify anything in this new application, since all the functionality is available from FrameWork (unlike in older versions). The basic steps to work with the Emulator are:

- Create a backup copy of the system file or product file that resides in the SmartImage Sensor. This step is needed only if the user needs to work on the setup of a SmartImage Sensor. A product file contains all the product parameters (specific for every inspection) whereas the system file contains all the parameters in the System. Figure 5 illustrates the difference between product level and system level.
- Record some images from the SmartImage Sensor. Using the record option, the user can setup FrameWork to record images imported from the SmartImage Sensor to the computer hard drive. Then, when FrameWork connects to the Emulator, the image sequence could be loaded into the Emulator and the user could cycle through those images. (This step is optional, as the user may already have an image sequence to work with.)
- Start Framework and connect to the Emulator. Choose the Emulator that corresponds to the SmartImage Sensor used.
- Perform changes to the system/product files as needed. The Emulator supports any type of changes: the user can add or delete SoftSensors and products, set up communications, etc. The only changes that are not reflected are changes that relate to physical setup such as exposure time, illumination, etc. because the Emulator works with prerecorded images.
- Back up the product/system file to the PC and restore it to the SmartImage Sensor.

For a quick tutorial on this process see Appendix A – Emulator tutorial.

Chapter 3 - SoftSensor Reference

This chapter explains the main concepts of the FrameWork SoftSensors. As it was mentioned before, SoftSensors are the working class inside SmartImage Sensors. Every type of SoftSensor serves a specific purpose, and the combination of the SoftSensor results represents the overall result of the inspection. The main groups of SoftSensors are Presence/Absence SoftSensors, Positioning SoftSensors, and Specific Inspection SoftSensors. Positioning SoftSensors help locate parts to be inspected. In most applications, this type of SoftSensor becomes the key to a successful setup because they locate the part to be inspected and pass an internal reference to the remaining SoftSensors indicating the location of the part. Presence/Absence SoftSensors perform basic checks for detection of features. Specific Inspection SoftSensors include 1D and 2D code readers, OCR readers, measurement sensors, math tools, color sensors, pattern matching sensors, geometry finding sensors, spectrograph for color analysis, and programmable sensors.

Presence / Absence SoftSensors

The first group of SoftSensors is the Presence/Absence SoftSensors. These SoftSensors perform basic checks for detection of features and edges or analyze the intensity level of the pixels in a certain area or linear path. These SoftSensors are:

- EdgeCount SoftSensor: these SoftSensors look for and count transitions from bright pixels to dark pixels and vice versa.
- FeatureCount SoftSensor: these SoftSensors are an enhanced version of the EdgeCount SoftSensors: besides detecting intensity transitions, they also report the size of the regions between such transitions.
- Intensity SoftSensors: these SoftSensors calculate the percentage of bright pixels in a given region of the image.

In all cases, the classification of pixels as "bright" or "dark" is based on a user-defined threshold. For details refer to section "Threshold" on page 52

EdgeCount SoftSensors

EdgeCount SoftSensors are designed to detect and count transitions between bright and dark pixels (or above and below threshold). These SoftSensors basically analyze the image and determine where the pixel graph intersects the threshold line. Those intersections determine an edge, and the user can select the type of edge to locate (bright to dark, dark to bright, or either edge). Figure 26 shows an EdgeCount SoftSensor and its pixel graph.

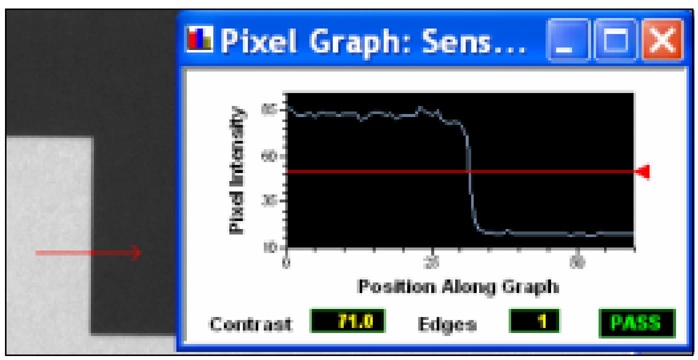


Figure 26: Edge Count SoftSensor and its pixel graph. The SoftSensor starts on the bright area (high intensity) and then goes into the dark area (low intensity).

In this case, the shape of the SoftSensor is a horizontal line, and the direction of scanning is from left to right as indicated by the arrowhead at the right end of the SoftSensor. From the pixel graph the user can determine that the length of the SoftSensor is about 60 pixels and that the edge (transition from bright to dark) is located close to pixel number 32. The output of this type of SoftSensor is the number of edges found; in this case it is one. However, the user must be careful to select the correct type of edge to be located. If in this case the user had selected dark to bright edges only, the sensor would have returned zero edges because there are no such edges in the direction of scanning.

EdgeCount SoftSensors are available only as line SoftSensors because of their purpose. Since they detect edges, they must consist of lines searching for transitions between dark and bright pixels along a certain path. As a consequence, a single dark pixel (like background noise) could make them find extra edges and thus alter the result of the inspection. To avoid this problem, the user can choose to increase the scanning. The option of scanning extends the SoftSensor in any direction to make the inspection more robust. Figure 27 shows an example where this option is used.

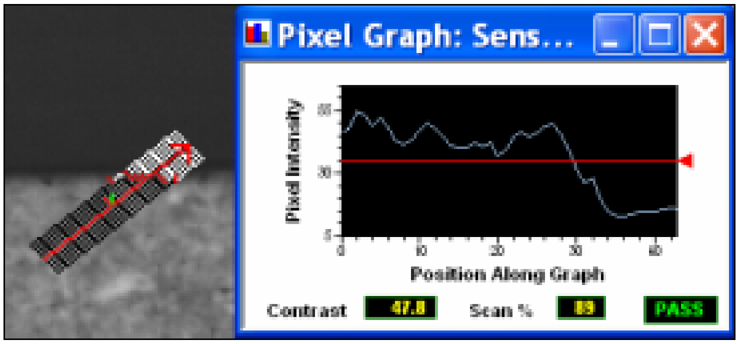


Figure 27: EdgeCount SoftSensor analyzing an image with noisy background.

The background in this case is noisier than the background in Figure 26, so the option of scanning must be used. As the pixel graph shows, some of the noise in the background is very close to the threshold level. If one of those dark pixels in the background reaches the threshold level, a new edge will be found. In order to avoid that problem, the area scanned by the SoftSensor was expanded. To do that, the user needs to specify the direction in which the SoftSensor is to be extended ("X" or "Y") and the number of pixels. Furthermore, a rotation can be indicated to expand the sensor in a very specific direction. In the example shown in Figure 27 the axis angle was set to 45 degrees and the SoftSensor was expanded 5 pixels in the "Y" direction. These settings rotated the main axis (X-axis) 45 degrees counterclockwise and then expanded the sensor 4 pixels in both the positive and negative "Y" directions.

This option makes the SoftSensor more robust and less sensitive to noise because the user now can select to pass or fail based on the number of scanning lines that need to find the desired number of edges. In other words, a few scanning lines could be affected by noise and the inspection can still be successful. Of course, this option adds some processing time to the system. For cases in which processing time is an issue, an additional option is available. The user can select to sample those scanning lines instead of analyzing every one. By indicating the scanning density as a percentage of the total lines, the user can select to skip some lines. In the example of Figure 27 there are 4 lines above and 4 lines below the sensor. If the user selects 50% of scan density, every other line will be used instead every single one.

This type of SoftSensor can be set to pass, warn or fail based on maximum edge count, minimum edge count, percentage of scanning lines finding the appropriate number of edges, and minimum contrast to find edges.

FeatureCount SoftSensors

FeatureCount SoftSensors are designed to detect and count features that make the part being inspected different from the background. A feature, as interpreted by FrameWork, is defined by two consecutive opposite edges. For example, a transition from bright to dark, followed by a transition from dark to bright indicates the presence of a dark object in bright background. That object is considered a dark feature. Likewise, a light feature is defined by a transition from dark to bright followed by a transition from bright to dark. The SoftSensors basically analyze the image and determine where the pixel graph intersects the threshold line (edges), then, group those edges to define features. Figure 28 shows a FeatureCount SoftSensor and its pixel graph. It can be observed that the sensor crosses four dark regions and five light regions.

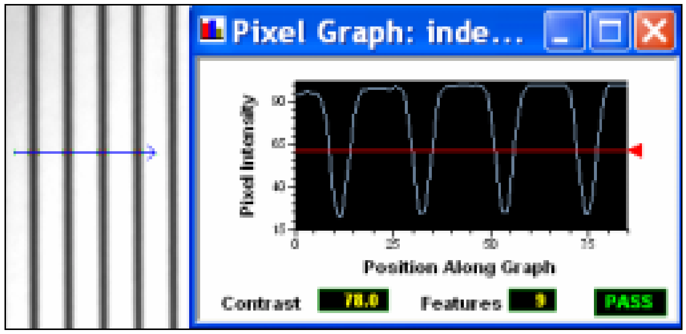


Figure 28: FeatureCount SoftSensor and its pixel graph.

This is reflected in the pixel graph, where four groups of pixels can be identified below threshold and five above threshold. These groups of pixels represent the features. As the pixel graph shows, this sensor is finding nine features (four dark features and five light features). Depending on the application, different approaches can be taken. The user can select which features to count. If the user selected to count only the dark features, the sensor would report four features instead of nine. Likewise, if light features were selected, five features would be reported. The default option(used in this example) is to count both light and dark features. If the image to be inspected is not as good as the one in Figure 28, the user may use some options that will make the SoftSensor less susceptible to noise.

The first option to handle noise is to expand the scanning region, which works the same way as for the EdgeCount SoftSensor (explained before). The second option is a new capability that is unique to SoftSensors that find/count features. It consists of setting a minimum and maximum feature size. When this option is used, the only features that are counted are the ones whose size falls within a specified range. Figure 29 shows a sample image with a sensor in it and a pixel graph. The SoftSensor is trying to detect the presence of the dark mark but the dark dot on the outside is another feature. According to the pixel graph, more than one dark feature should be found. However, it can also be observed that whereas the size of the feature to be located is about 80 pixels, the rest of the features do not exceed 20 pixels in size.

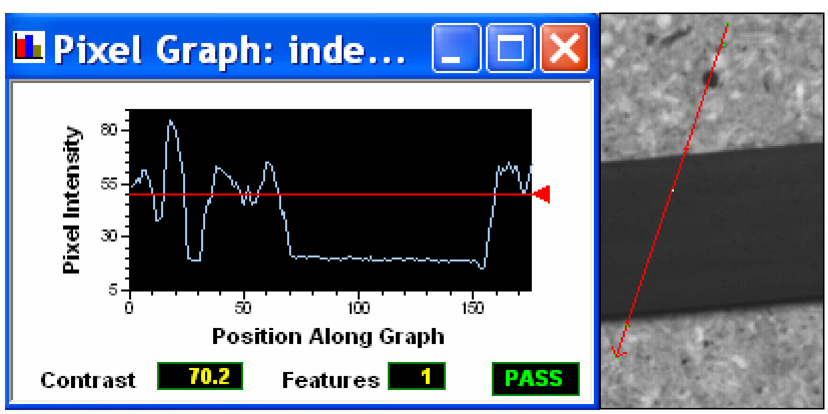


Figure 29: Filtration of noise using feature size.

In this case the user can select a minimum feature size of fifty pixels and only one dark feature will be reported by the sensor. This makes the SoftSensor less susceptible to noise making the inspection more reliable.

This type of SoftSensor can be set to pass, warn or fail based on maximum feature count, minimum feature count, percentage of scanning lines finding the appropriate number of features, and minimum contrast to find features.

Intensity SoftSensors

Intensity SoftSensors calculate the percentage of bright pixels in a given region of the image. It could be a line or an area. The major purpose of this SoftSensor is to determine the presence or absence of image features. The SoftSensor analyzes the pixels and determines how many of them are above threshold (light pixels) and how many of them are below threshold (dark pixels). Figure 30 shows an example of its use.

Figure 30: Use of an Intensity SoftSensor to verify the presence and correct position of a label.

The SoftSensor in this case is detecting the correct placement of the label in the computer disk. This particular example uses a line SoftSensor. The pixels along that line are grouped based on their intensity levels. The histograms, illustrate the pixel analysis of the image. For the first case, the label is present, so most pixels along the line are light. This is reflected in the histogram which shows above ninety percent of the area as bright area. For the second case, all the pixels are dark (below threshold) so the SoftSensor reports no bright area. This example only required the verification that the label reached a certain point, so the line SoftSensor was adequate, but if the inspection consisted on the verification of the entire label, a rectangular area SoftSensor should have been drawn over the label.

This type of SoftSensor can be set to pass, warn, or fail based on the amount of bright area or the level of contrast found in the image. For the example above, a minimum bright area should be established to determine what is acceptable and what is not. A minimum bright area of 90% would cause the SoftSensor to pass when the label is correctly placed and to fail when the label is incorrectly placed.

Simple Positioning SoftSensors

This group of SoftSensors helps to locate parts and features when working with applications that need to account for part movement. These SoftSensors are:

- Translation SoftSensors: These SoftSensors look for edges or features along a certain path or in an area and then provide offsets in the horizontal and/or vertical direction that would serve as position reference for other SoftSensors.
- Rotation SoftSensors: these SoftSensors look for edges or features along circular or elliptical paths (or areas using multiple scanning lines). They would provide angle information that can be used as position reference for other SoftSensors.

It is important to note that these are not the only SoftSensors that could be used for providing a position reference for other tools. They are just the simplest and fastest positioning tools available. For more information on which SoftSensor to choose or how to approach applications where part tracking is needed, consult the section Dealing with Part Movement in Chapter 4

Translation SoftSensors

Translation SoftSensors are designed to locate the absolute position of an element in an image in cases where that element's position changes between inspections. There are two types of Translation SoftSensors: Lines and Fiducial finders.

Translational lines determine the position of an edge or a feature (edges and features were explained under EdgeCount and FeatureCount SoftSensors). The option to locate features should be used when the application allows for it because by selecting a minimum and maximum feature size, the user can make the SoftSensor less sensitive to noise. This type of SoftSensor reports a distance as a percent offset from the origin of the SoftSensor. It searches the pixels from the origin in the direction of scanning (set by the direction in which the SoftSensor was drawn). When it finds the edge/feature that the user indicated, it reports the distance from the origin of the SoftSensor to the location of the edge/feature as a percentage of the total length of the SoftSensor. Figure 31 shows a translation SoftSensor and its pixel graph. The pixel graph shows that the length of the SoftSensor is approximately 110 pixels and the light to dark edge is found at approximately 62% of the length of the SoftSensor from the origin.

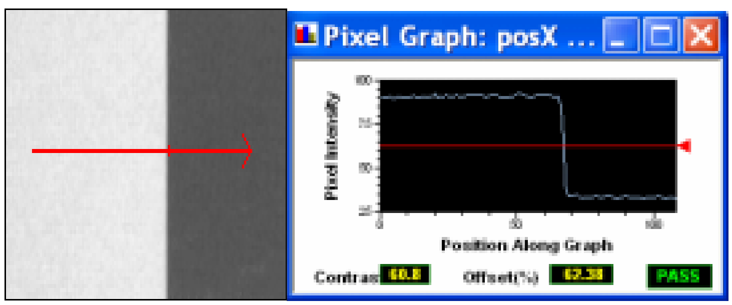


Figure 31: Translation SoftSensor and its pixel graph.

The output of this SoftSensor will be 62.38% of offset, provided that the user specified that this type of edge was an acceptable reference. However, users should tend to use features because of the advantages they offer when dealing with noisy images (see FeatureCount SoftSensor for more details).

This SoftSensor is needed to correct for movement in one direction. However, in most cases the part needs to be tracked in two directions of translation. When that is the case, another SoftSensor of this type needs to be created and a reference between them needs to be assigned. That is, the user needs to create two Translation SoftSensors: one to determine the translation in one direction and a second one to determine the translation in another direction. The second one needs to reference the first one and all inspection SoftSensors should reference the second one to create the chain of references mentioned before.

Fiducial SoftSensors are the area tools of the Translation SoftSensors. They scan areas from the top, bottom, right and/or left looking for a specific region (dark fiducial or light fiducial). The user must specify what type of fiducial needs to be found and which directions of scanning are to be used. Usually, two directions of scanning are enough to determine the position of the part. Figure 32 illustrates the difference between using two scanning lines and four.

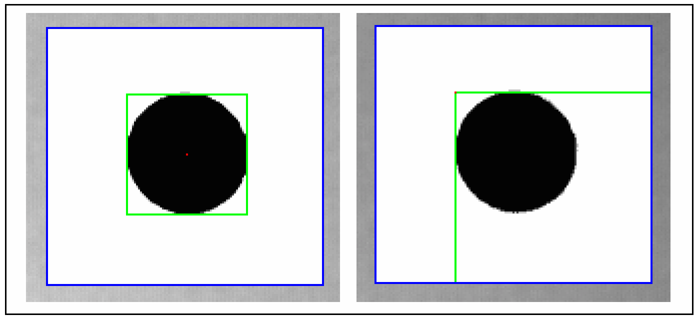


Figure 32: Two different ways to locate a part using the same SoftSensor: scanning from all four sides (first case) or only from the left and the top (second case).

In this example, the use of four scanning lines is redundant because the part can be easily located using two scanning lines. Furthermore, the use of extra scanning lines means more processing time. The output of the SoftSensors changes when different numbers of scanning lines are used. When four scanning lines are used, the reported output is the position of the center of the green bounding box. That bounding box is determined by all four scanning lines. When two scanning directions are selected, the output is the position of the intersection of the two scanning lines. In all cases, the point used as the position is marked with a red dot. Figure 32 also illustrates image marking. The area inside the Fiducial SoftSensor is being binarized. Pixels above threshold appear white, whereas pixels below threshold appear black. This image marking helps the user determine if the threshold needs to be adjusted by letting the user see what the sensor sees.

Another application for fiducial SoftSensors is the replacement of translation lines in a chain of references. Figure 33 shows an example where the vertical position is being tracked.

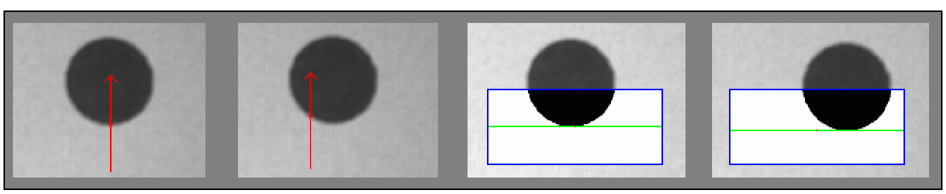


Figure 33: replacement of translation line with translation fiducial SoftSensor to increase reliability.

With the setup of the first image, as the part shifts right or left, the SoftSensor will report a change in vertical position because the intersection of the line SoftSensor and the edge of the part shifts upward. This undesired effect is illustrated in the second image, where the part has shifted to the right and thus intersects the SoftSensor at a different point. This will cause the SoftSensor to report a change in the vertical position. In the third image, a different setup is used where a Fiducial SoftSensor tracks the vertical movement of the part. This allows for some horizontal movement without effect on the vertical position being read. The fourth image shows that this setup is not affected by horizontal movement of the part.

Besides providing a position reference for other SoftSensors to perform the inspection, Translation SoftSensors can be used to inspect a part. In applications where the position of a certain feature decides whether the part passes or fails, translation SoftSensors can be used to actually determine the final result. They can be set to pass or fail based on how much the part has moved in the image.

Rotation SoftSensors

Rotation SoftSensors are designed to compute the angle of rotation of objects in the image. Like Translation SoftSensors, Rotation SoftSensors allow different positions of an object without failing the part for being in an unexpected position or location. In other words, other SoftSensors can reference the output of Rotation SoftSensors in the same manner as with Translation SoftSensors. There are two types of Rotation SoftSensors: arc-based and parallelogram-based.

Arc-Based Rotation SoftSensors are often applied when the position of an object is known to have fixed radius and center of rotation. For example, the ball bearings in a race can be moved around a circle, but are restricted from moving up and down independently. A Rotation Circular Arc can be drawn to locate one bearing and other SoftSensors can reference the position output of the Circular Arc. Basically, the arcs used to track rotations work like translation lines do. They search along the arc for a specific edge or a specific feature. Then, they report the percent offset from the origin of the SoftSensor. The difference with translation lines is that besides that percent offset, there is a center of rotation associated with the angle being reported. That center coincides with the center of the tool as drawn. Thus, when this tool is used to track the rotation of a part, the closer the center of the tool is to the true center of rotation of the part, the more accurate the results. Figure 34 shows an example where the incorrect placement of the rotation tool causes a problem with the inspection. In the upper left image, we see an intensity tool at the tip of an arrow. Since the center of the arc does not coincide with the center of rotation of the arrow, as the arrow changes position, the arc tracks it incorrectly. It finds it but it uses its own center as the center of rotation. In the third image the arc is correctly placed, so as the arrow changes position, the arc tracks it correctly and it passes a good reference for the intensity SoftSensor which performs the inspection at the tip of the arrow as expected.

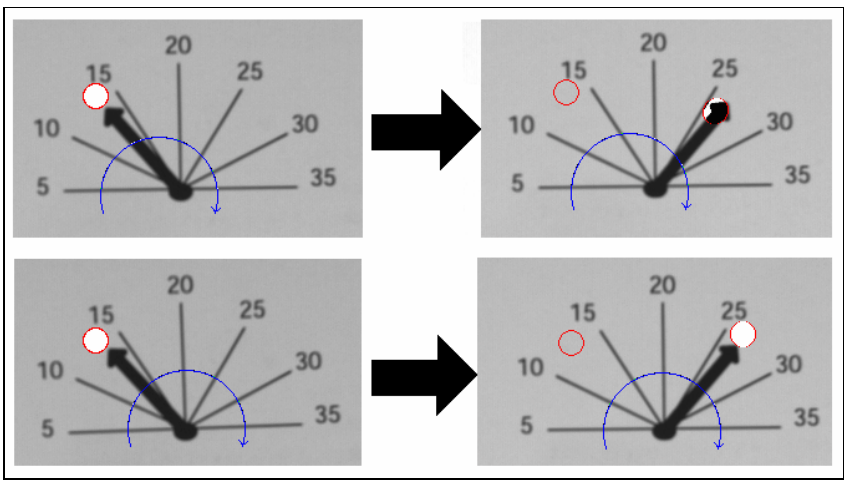


Figure 34: Incorrect and correct uses for a rotational arc.

The second rotation SoftSensor is the Find Edge tool (this is a parallelogram-based SoftSensor). This SoftSensor is used when an object with straight edges is not fixed in orientation. The SoftSensor fits a line along the straight edge and tracks the rotation of the line. A comparison between the arc and the parallelogram is shown in Figure 35.

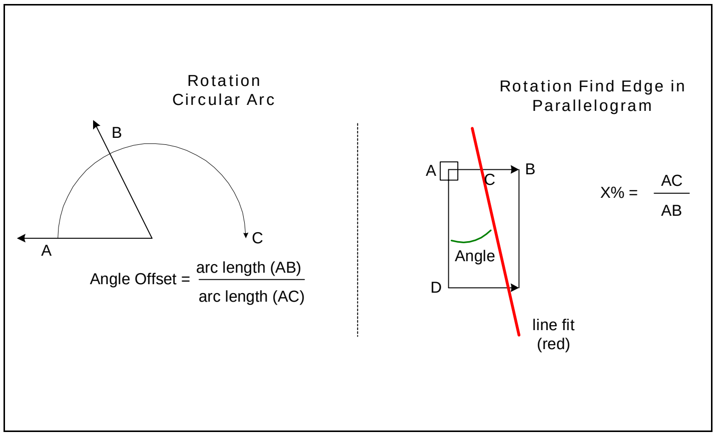


Figure 35: Output calculation for Arc (left) and Parallelogram (right) Rotation SoftSensors.

As the figure indicates, besides the angle of rotation the parallelogram outputs a distance. That distance, illustrated as X% in the figure, is obtained from the origin line of the SoftSensor. That is, the first line that the user draws for this SoftSensor acts as a Translation SoftSensor. Furthermore, the direction in which that line is drawn indicates the direction of the scanning lines. This SoftSensor consists of a number of parallel scanning lines. Those lines look for edges or features (according to user specifications) and each one determines a point where they intersect the selected edge/feature. The SoftSensor then fits a line along those points and tracks its rotation. The intersection of that line with the original SoftSensor line is used for translation output. Figure 36 shows a sample image where the parallelogram tool is used. The square in the upper left corner of the tool indicates where the user started drawing the SoftSensor. The arrows indicate the direction of scanning, so this tool was drawn starting on the upper left corner going right, then it was expanded down to indicate the height.

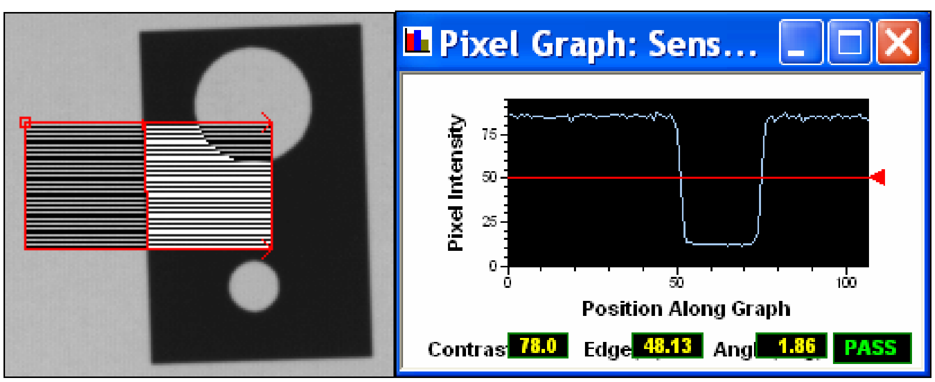


Figure 36: Use of the Parallelogram tool of the Rotation SoftSensors. The pixel graph associated with this SoftSensor illustrates the pixels along the origin line (top edge in this case).

The top line then becomes the origin line and works like a Translation SoftSensor. The SoftSensor pixel graph corresponds to the origin line. As it was discussed before, pixel graphs are only compatible with line SoftSensor; however, in this case we have an area SoftSensor and a pixel graph associated with it. Since the most valuable region of the SoftSensor is the origin line, that is the line used for the pixel graph. The pixel graph then shows a numerical value for both the angle of the part and the position of the edge or feature.

Specific Inspection SoftSensors

The next group of SoftSensors is the Specific Inspection SoftSensors set. These SoftSensors perform many tasks from very specific inspections to a user defined task (programmable). These SoftSensors are:

- Measurement SoftSensors: Take measurements from parts for inspection or simply to gather data.
- Math Tools: Perform mathematical calculations based on information from the measurement SoftSensors.
- Readers: Perform readings of 1D and 2D codes as well as OCR (Optical Character Recognition). They can be used for recognition, verification, and in some cases for grading.
- Blob Tools: Search the image for specific shapes to track their movement or simply count them.
- Template Match: Learn a model and then verify that the model is present in subsequent images, if the image does not match the model it reports the amount of error.
- ObjectFind: Geometry-finding algorithm used to locate, count and classify parts even if they are overlapping.
- Pixel Counting: Color matching tools. Learn colors and determine their presence in subsequent images.
- Color Monitoring: Ultra precise color verification tools.
- Segmentation: Versatile tools used to learn color patterns, detect defects, or count parts.
- SmartLink: Used to create tables of data to send out to SmartLink hardware for monitoring purposes. This SoftSensor will not be covered in this manual; for more information refer to the SmartLink manual.
- Script: Programmable tool that can access other SoftSensors for data gathering, etc.
- Spectrograph: Available in the SpectroCam, this SoftSensor analyzes the full spectrum of a part or light source and can compare it against a list of previously learned spectrums.

Measurement SoftSensors

Measurement SoftSensors are used to take measurements from the parts being inspected. These measurements will be in pixels unless a scale factor is created and calibrated (see Math Tools for details about the scale factor). There are two types of Measurement SoftSensors: line based and area based. In this case, the area based SoftSensors consist of a number of scanning lines. That is, they collect data from many lines in order to produce a more accurate result. Typically, each scan line will produce a data point by finding an edge. The SoftSensor then uses all those data points to perform calculations (e.g. fit a line or a circle). All the Measurement SoftSensors can use gradient based methods for threshold in addition to the intensity based types. When a gradient based threshold is being used, sub-pixel accuracy can be easily achieved and some advanced SoftSensor parameters become available: Edge Width, Use Best Edge, and Hi-Res Mode. These options were added to this version of the software to make it less sensitive to noise. This is how they help the SoftSensor perform the inspection:

- Edge Width: This option lets the user indicate how many pixels determine the edges to be used. If the images are not well focused, the edges will not appear as sharp. In those cases, the user needs to give the SoftSensor an idea of the transition width to better calculate edge positions. Figure 37 illustrates the use of this option. The SoftSensor finding the top edge faces a sharp transition while the one finding the bottom edge faces a blurred transition in a fuzzy image. For the bottom edge, the edge width for the SoftSensor is set much higher.

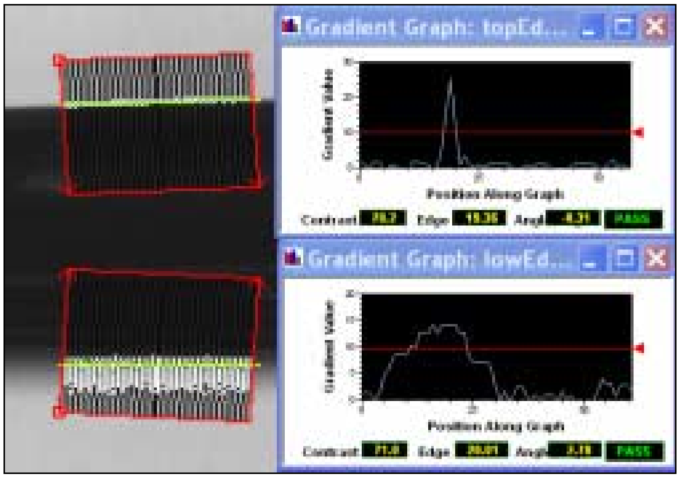


Figure 37: Area Measurement SoftSensors in presence of sharp and fuzzy edges. The gradient graphs indicate the width of the transition.

- Use Best Edge: This option is for noisy images. Sometimes background noise produces minor peaks in the gradient graph. In cases where those peaks happen to reach levels above threshold, this option will discard them and keep only the highest peak. The use of the Best Edge option is illustrated in Figure 38. In this example, if the option was not selected, the SoftSensor would report the edge close to pixel number 20, which is where the first peak (above threshold) is located. The use of this option makes it report the true edge instead because it is a better defined edge. The SoftSensor on the left is not using this advanced option, and the data points generated by the presence of noise cause the output line to be calculated at an incorrect location. The second SoftSensor uses this option and the output line is correctly drawn over the edge of the part.

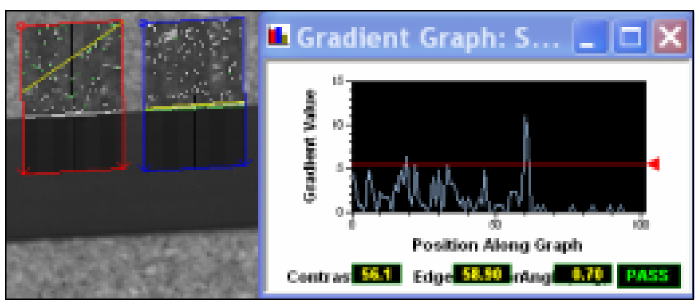


Figure 38: Effect of the use of the best edge option. The SoftSensor on the left fails to locate the edge because of the background noise, whereas the one on the right looks for the best edge (the highest peak in the gradient graph).

- Hi-Res Mode: This option increases the sub-pixel accuracy of the measurements when the scanning lines are not drawn parallel, perpendicular or at a 45-degree angle with respect to the pixel grid. It compensates for angular error given by the shape of the pixel grid. By using this option the user should obtain more repeatability in the measurements.

All these options increase the processing time of the inspection, so if timing is an important issue, the user should first try to avoid them by changing the light or the lens being used to improve the image.

If sub-pixel accuracy is not a requirement, the SoftSensors should use intensity based threshold. This option increases repeatability in the measurement and it adds some advanced options to deal with noisy images as well. The user has the option to select two of the SoftSensor processing operations (Erode, Dilate, Open or Close). For more information on these operations see Processing Operations under SoftSensor Parameters.

Besides the specific parameters discussed so far, some measurement tools include extra options to make the SoftSensor more precise. When area based Measurement SoftSensors are being used, more parameters become available: Scan Density, Max Iterations, and Outlier Distance.

- Scan Density lets the user change the number of scan lines being used for the particular SoftSensor. These scan lines run parallel (for rectangles/parallelograms) or towards the center of the SoftSensor (for circles). The more scan lines are used, the more data points are obtained for the measurement. In some cases, increasing the number of scan lines gives a better estimate of the location of the edge of the part being inspected.
- The Max Iterations option lets the user indicate how many times to recalculate the line/circle being interpolated using the data points. After every calculation, points are assigned a certain weight for the next calculation based on the distance from them to the computed output (line/circle). By doing this, the algorithm puts more emphasis of true edge points and less on noise. Figure 39 illustrates the use of this option. Image (a) shows the output line calculated with no iterations. As the image illustrates the output line is not on the edge because of the imperfection of the part. Images (b) and (c) show the same SoftSensor after 2 and 5 iterations. Image (c) is the only one that finds the edge of the part regardless of the imperfections that the part presents.

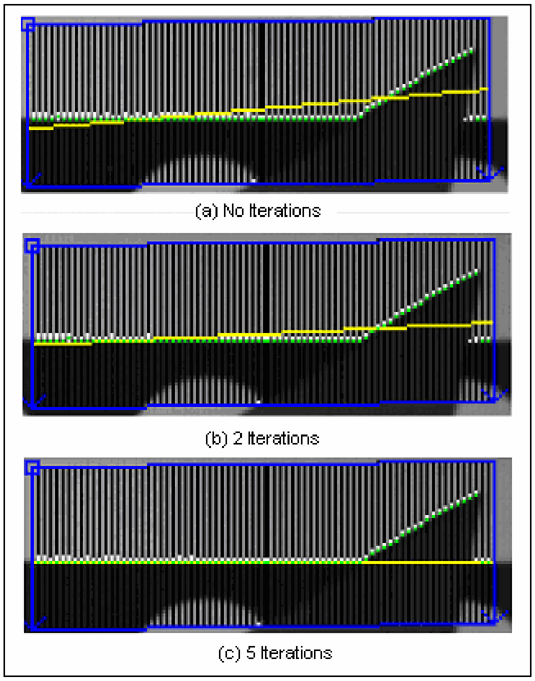


Figure 39: Use of the number of iterations to overcome imperfections/noise in the part.

- The Outlier Distance option is used after the iterations and it discards points that are more than a user defined number of pixels away from the output line/circle. This option helps find the right edge of the part giving a more accurate result.

There are four main measurements that the SoftSensors can take: line offset, caliper, line fit, and calculation of a radius.

The line offset Measurement SoftSensor works just like translational lines do. That is, it looks for a specific characteristic (edge or gradient level) and reports the distance from the origin of the

SoftSensor to the pixel where the characteristic is found. Unlike the Translation SoftSensors, line offset tools report the distance from the origin of the SoftSensor in pixels instead of a percentage. The main advantage of this SoftSensor over the translational line is that since it is a Measurement SoftSensor it can use the gradient based threshold to achieve sub-pixel accuracy.

Caliper-type measurements can be performed using a line or an area SoftSensor. In both cases the SoftSensor can scan inside out or outside in (just like using a caliper). The scanning lines start scanning from the ends of the SoftSensor or from the center until they find the specified characteristic. This characteristic could be a peak value in the gradient graph (when using gradient computation) or a specific type of edge (when the threshold type is intensity based). The output of the SoftSensor is the distance in pixels between the edges of the part. Figure 40 shows the use of both line and area based SoftSensors. Line SoftSensors (called Measure Across Line) can only measure the distance along the line.

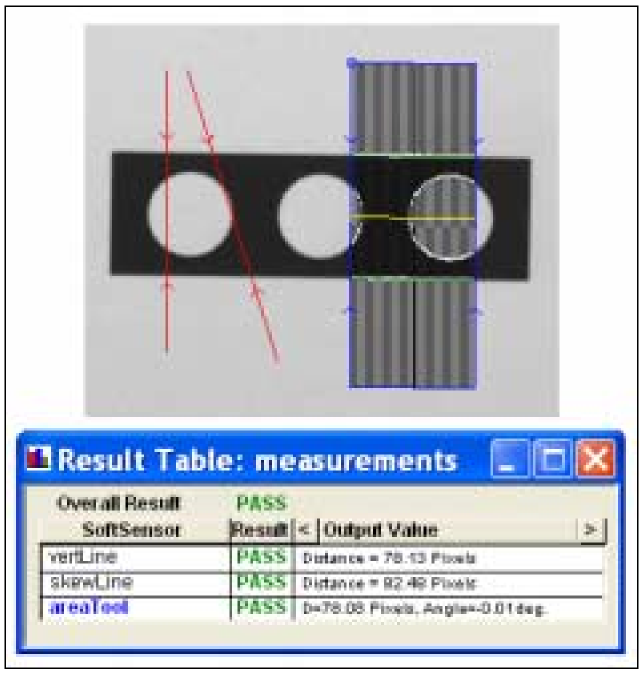


Figure 40: Caliper type SoftSensors measuring the height of a part.

As the figure illustrates, more precision can be achieved using the area SoftSensor (called Measure Across Area). Line SoftSensors require that the SoftSensor be drawn at a perfect 90-degree angle with respect to the edge of the part. None of the line SoftSensors in the image is drawn in such way, so the reported distance is greater than the actual distance. Area SoftSensors on the other hand use many scanning lines and each scanning line provides a data point. The algorithm then fits a line along each set of points and reports the distance between those lines as well as the angle between them. If the angle is other than zero degrees, the algorithm averages the maximum and minimum distances that it calculates from the data points in the SoftSensor. The area SoftSensor requires more processing time because it not only consists of many scanning lines, but it also requires further processing for the line fitting calculation. One way to reduce the processing time is to reduce the scan density. Scan density is reported as a percentage; for 100% every single line in the SoftSensor will be used, for 50% every other line will be used, and so on. The user can select this value anywhere from 0 (only the 2 edges) to 100 (every single line inside the SoftSensor plus the two edges). Higher scan density is associated with more reliable measurements, so the user should only reduce this number if the image contains very little noise, or inspection speed is critical. Note that if the Scan Direction is set to Inside Out and the center line of the SoftSensor is not inside the part, it will fail.

The line fit SoftSensor (called Area Edge Line SoftSensor) is another area SoftSensor. It works just like the area caliper tool only that it scans in one direction. It consists of many scan lines (a number that the user can specify like in the Measure Across Area SoftSensor). Every scan line intersects the edge of the part being inspected and determines the intersection point. The algorithm then interpolates those points into a straight line which is the output of the SoftSensor. This type of tool is the one shown in Figure 37, Figure 38, and Figure 39.

The Measurement SoftSensors that calculate radii are the Find Circle and the Segmented Find Circle. They have the same functionality; they differ only on their applications. A Find Circle tool should be used when the circle to be measured is a complete circle. The segmented version of this tool is used when there is not a full circle. Figure 41 illustrates the use of these two different tools.

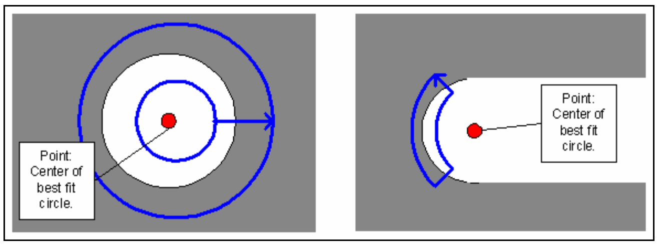


Figure 41: Different uses of Find Circle and Segmented Find Circle tools.

To use these tools the user draws two circles, one inside the part and one outside. Then, scanning lines will go from the first circle drawn towards the second one looking for the edges of the part. The intersections of the scanning lines with the edge of the part determine data points. The algorithm then fits a circle through those points.

Different outputs can be obtained from the area based tools. The user can select minimum, maximum, or average distance (or radius, depending on the tool being used). These tools can be set to pass or fail based on specifications for the distances being measured (min radius, max offset, etc.) when the inspection consists of the determination of a correct size. Another feature that the user can select for quality control is the overall roundness/straightness. Depending on the area tool being used, there will be an extra output that indicates how spread out around the output line/circle the data points are. In other words, straightness (or roundness)is given by the sum of the distances between the data point that is farthest away in one direction and the one farthest away in the other direction of the result line (or circle). Hence, a perfect edge would produce a very small straightness (roundness for a circle) whereas a noisy edge would produce a high straightness (roundness for a circle). In this case, the outlier distance helps a lot in presence of noise. When noise produces data points for the final line, the straightness can increase considerably. That is, one single data point caused by noise far from the edge will increase the straightness dramatically. Outlier distance will discard those so the calculation for straightness concentrates on true edge data points instead of noise. Figure 42 illustrates the use of this option. The figure shows a dark part and a noisy background. The dark pixels of the background cause some scan lines to identify data points far from the edge.

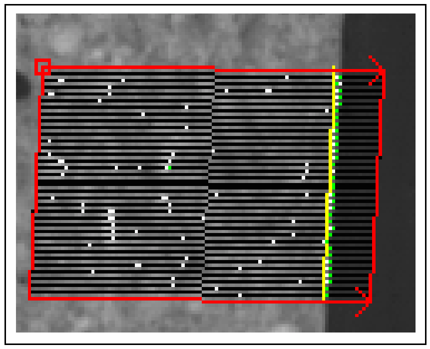


Figure 42: Effect of Outlier distance on Straightness for noisy images

When a number of iterations is selected to correct that effect, the edge is correctly located. However, the data points caused by the noise are still active, so the SoftSensor reports a straightness of 49 pixels. When the outlier distance is used to discard points located more than 5 pixels away from the line the data points produced by the noise are discarded and the straightness drops down to 1.92 pixels. This type of pass/fail condition could be used to verify the quality of a cut or a drilling in a part.

Another use for the Measurement SoftSensors is positioning. Even though FrameWork has a dedicated number of SoftSensors for position reference, Measurement SoftSensors can provide a position reference as well. Since Measurement SoftSensors can use gradient based thresholds and offer a wide range of options to overcome image deficiencies, they can be more precise than the Positioning SoftSensors. In cases where the positioning of a part needs to be very precise, the user can select a Measurement SoftSensor instead of a Positioning SoftSensor as the position reference.

Math Tools

The Math Tools toolbox offers an assortment of SoftSensors. Unlike any other tools (except Scripts, the Math Tools SoftSensors require no additional drawings. Math Tools use references to other SoftSensors that have been previously defined on the object. Usually, they reference Measurement or other Math SoftSensors using their outputs (lines and points) for the calculations. There are three types of Math SoftSensors. The first type performs basic calculations based on other SoftSensors outputs. Table 2 shows a list of these SoftSensors and indicates the type of reference that the SoftSensor needs and the output it produces.

Math Tool	References	Returns
Distance	Two points	The distance between the reference points
	Two lines	The distance between the reference lines
	A point and a line	The perpendicular distance from the reference point to the reference line
Angle	Two lines	The angle between the reference lines
Intersection	Two lines	The coordinates of the intersection of the reference lines
Midpoint	Two points	The coordinates of the midpoint between the reference points

	A point and a line	The coordinates of the midpoint between the reference point and the reference line
Midline	A point and a line	A line parallel to the reference line through the midpoint of the reference point and the reference line
	Two lines	The bisector of the angle determined by the reference lines
Line through two points	Two points	A line through the reference points
Perpendicular Line	A point and a line	The perpendicular to the reference line through the reference point

Table 2: Summary of Math SoftSensors used for basic calculations.

The second type of SoftSensor is the scale factor. For cases where a distance in pixels is not enough, the user can create a scale factor SoftSensor to convert pixels to units. There are two ways to compute a scale factor: static and dynamic. The static scale method computes the scale factor once and uses it for subsequent images. The user needs to create a SoftSensor to measure a known distance in the object being inspected or in a calibration target. That distance is then indicated to the scale factor SoftSensor, which computes a scale factor in user units per pixel based on the distance in pixels that the measurement SoftSensor reports. This conversion factor is used for every image. The dynamic method requires that an object (or calibration target) with a known size remain in front of the SmartImage Sensor as long as images are being inspected. The size of this object in pixels is calculated in every inspection and a new scale factor is computed in every inspection. This makes the system more reliable but it takes more processing time. It should be used in cases where there is variation in lighting or working distance (distance from the lens to the part being inspected).

The third type of math SoftSensor is the coordinate transformation type. These SoftSensors are used to set up a coordinate system different from the pixel grid or for correction of lens distortion. The user needs to select a minimum of 5 points in the image (which can be produced with Measurement or Math SoftSensors). The Coordinate Transformation SoftSensor then uses those points to compute the new coordinate system based on their original coordinate (pixels) and the new coordinates (user defined). After this procedure is completed, any SoftSensor can reference the new coordinate system to output a location based on user defined coordinates. That is, any location in the image will have the original coordinates and the option to report the user defined coordinates. The main application of these tools is interfacing with a robot. Pixels coordinates need to be transformed to robot coordinates before transferring the data.

Readers

FrameWork contains a number of SoftSensors dedicated to reading codes, labels, etc. These are the Reader SoftSensors. These SoftSensors consist of the 1D BarCode Reader, 2D Reader (DataMatrix, Vericode, etc.) and OCR SoftSensor (used for Optical Character Recognition).

One Dimensional Barcode Reader

The 1D BarCode reader consists of two types of SoftSensors: a line SoftSensor (arc or straight) and an area based SoftSensor. The line SoftSensor (arc or straight) must be drawn on top of the barcode. This type of tool uses intensity based thresholds, and it has an extra option to include sub-pixel calculation, making it more powerful when the image quality is not the best and the line reader crosses the barcode at an angle other than 0º or 45º. This option makes the SoftSensor more reliable, but it adds some processing time. The types of barcode that this tool can read are Interleaved 2 of 5, USS-128, USS-39 (Code 39), UPC/EAN (found on most consumer products), Codabar, PharmaCode, non-stacked versions of RSS-14, Code 93, POSTNET, BC412, and PDF417. The last two types of barcode are called stacked barcodes. The SoftSensor can be programmed to auto detect the type of barcode being read or to always search for a specific type of barcode. The automatic option adds some processing time and should be used only if the barcode type is unknown. The second option can be set manually (the user indicates which type of code is to be read) or automatic (sets the type of barcode upon read of the first instance of the barcode). Usually, applications need to read a certain type code which does not change, what changes is the encoded message that the SoftSensor needs to read. Therefore, the usual setup for this SoftSensor is to set the barcode type upon read. Using this option, the user does not need to worry about the type of barcode being used unless it is a stacked barcode. When the type of barcode being read is BC412 or PDF417 (stacked barcode types) or RSS-14 the user must indicate so because the automatic option will not detect the type. Figure 43 shows sample barcodes. PDF417 is the one on the left; it can be observed that it is a very different type of barcode: it is a stacked barcode. When this is the case, the SoftSensor creates more scanning lines to read the different levels of the stacked barcode.

Figure 43: Comparison of PDF417, Code 39, and PharmaCode types. The first one is a stacked barcode type.

The last two barcodes in the image are Code 39 and PharmaCode. For most users it is impossible to visually determine the type of code being used because they have very similar appearance (unless a stacked type is being used). In such cases the user should rely on the SoftSensor to determine the type.

Since this type of SoftSensor consists of a line, it is susceptible to noise. To overcome situations where the noise causes codes to be misread, the SoftSensor contains an option to increase the scan lines. By increasing the scan lines, the user can rely on the output from most of the lines instead of all of them. This will allow some lines to misread the code with the SoftSensor still getting the desired result. This option basically creates a number of copies of the SoftSensor parallel to the original one at a user defined distance and groups all those SoftSensor into one.

When the position of the code is not consistent from image to image, the area-based version of the 1D Reader SoftSensor can be used. This SoftSensor will generate parallel scanning lines starting from the origin line progressively toward the opposite side until it can read the code. If the opposite side of the parallelogram is reached and the code was not read the SoftSensor will fail. This SoftSensor will handle translation of the code and some rotation, but it will not try to determine the orientation of the code.

An option is available to perform grading on USS-128 and UPC/EAN barcodes. This SoftSensor can be used to grade the quality of the barcode being printed. The method follows ANSI/ISO standards and produces a full report for every barcode analyzed. For more information on this procedure, refer to the "Integration Notes" section of the DVT website or the FrameWork help files. This procedure is very sensitive to illumination. Before using the SoftSensor for barcode grading, system reflectance calibration must be performed (see the FrameWork help files for more information on Reflectance Calibration).

Two Dimensional Barcode Reader

There are two basic groups of 2D code Reader SoftSensors: the first is for DataMatrix codes (see Figure 44) and the second is for SnowFlake codes. The DataMatrix reader has the following features:

- Support for ECC200 format DataMatrix codes. ECC 200 codes can be identified by the even number of rows and columns (result is a white pixel in the corner opposite to the intersection of the solid dark edges).
- Automatic detection of code density starting at 10x10 (including rectangular code densities).

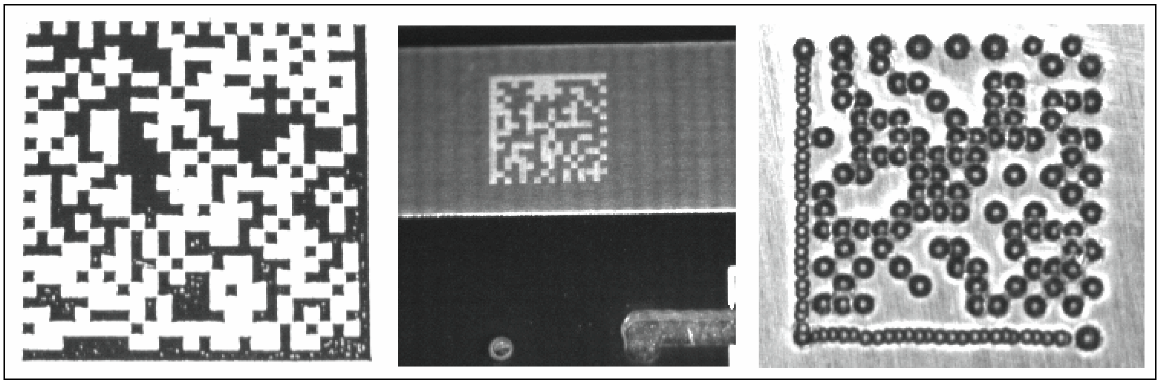


Figure 44: Different examples of DataMatrix applications: labels (left), printed circuit boards (center), and other metal parts (right).

There are two different DataMatrix SoftSensors: fixed and searching. The fixed SoftSensor, which consists of a quadrilateral, must be located just outside the DataMatrix to be read. If the code moves from this location, the SoftSensor will fail. The edges of the quadrilateral must coincide with the edges of the DataMatrix every time. The search SoftSensor (which could be a rectangle, an ellipse or a polygon) searches for a DataMatrix inside the area. That is, the code does not have to be consistently located in the same place every time. This SoftSensor will search for it and decode it as long as it is inside the SoftSensor. This tool takes more processing time because of the searching being performed. In order to reduce the processing time when performing a search, the user can specify known values for the DataMatrix. There are many parameters associated with this type of code (ex: cell size, matrix size, etc.). When some of those parameters are known, they can be indicated in the SoftSensor options so the search is limited to a search for a DataMatrix with those features. This makes the SoftSensor faster and more reliable. Figure 45 shows a screen capture of two different inspections. The SoftSensor on the left is the fixed SoftSensor, which will not search for the DataMatrix code: the code has to be exactly where the SoftSensor is or it will fail. Of course the SoftSensor can use a position reference to inspect the code where it is located, but it needs another SoftSensor to provide that reference. The second SoftSensor is the searching tool. It searches inside the area delimited by (in this case) the outermost rectangle and it finds the DataMatrix regardless of position or orientation as long as it is within that outer rectangle.

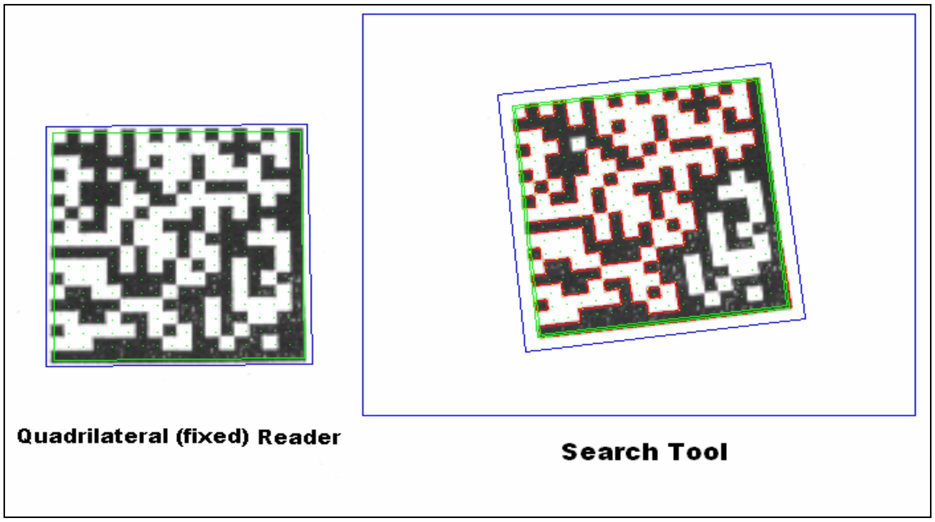


Figure 45: Two different methods to read a DataMatrix code: using a fixed tool or a searching tool.

This type of SoftSensor has intensity based thresholds, so the user can perform image processing operations (Erode, Dilate, Close, and Open) before trying to decode the DataMatrix. This is a helpful feature for cases where the image does not offer the necessary contrast. Figure 46 shows the use of the searching SoftSensor and the image marking that can be obtained from a DataMatrix SoftSensor.

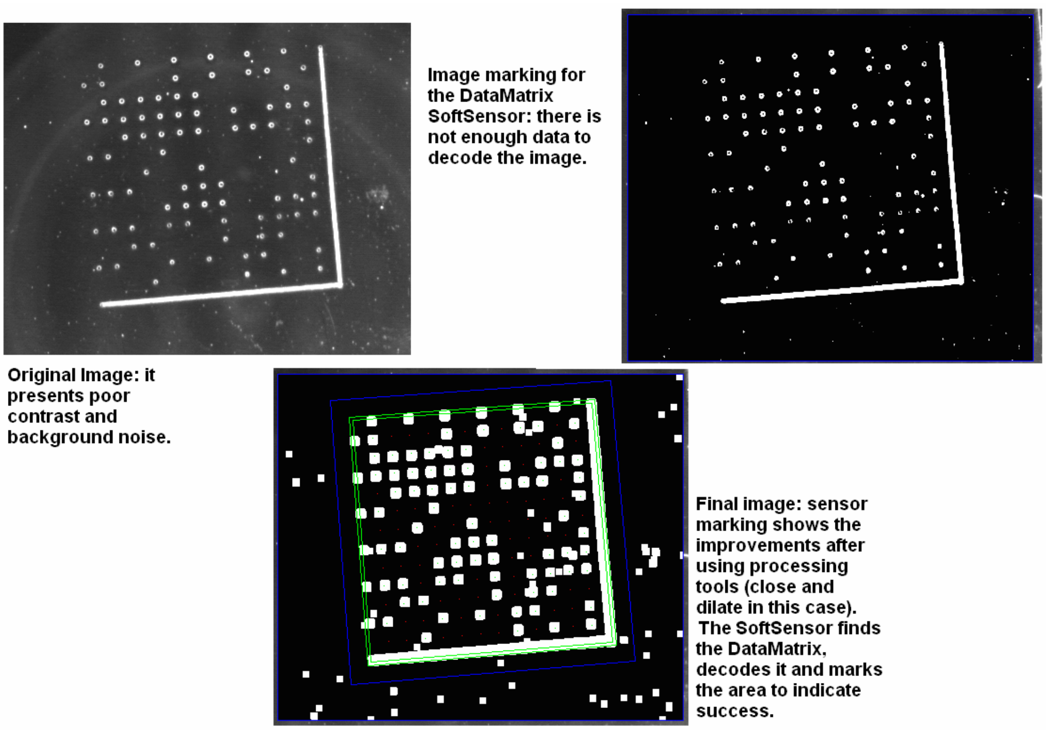


Figure 46: Use of the DataMatrix search SoftSensor to decode a noisy image.

The image marking for this SoftSensor is unique. It marks with double green lines the edges of the DataMatrix code, and it marks with a color dot the center of each cell. Red indicates that the cell is dark, and green indicates that the cell is light. Since this SoftSensor can read the DataMatrix even if up to 30% of it is missing, the user can select to change the marking to green for cells that were correctly identified and red for cells with error.

These SoftSensors can be set up to simply read, decode and output the data, or to verify that a certain string is present in the code. For the second case, the SoftSensor will fail when the string is not matched. It also includes a digital relearn option. That is, if the string to be decoded changes, a signal could be sent to the SmartImage Sensor to relearn the matching string.

Starting in FrameWork 2.6, a SnowFlake reader SoftSensor exists. This SoftSensor is available as a rectangle or as an ellipse. Both versions of the SoftSensor are searching readers; this means that the code does not have to be consistently located in the same place every time. SnowFlake codes are designed for fast printing applications and dot matrix printers are generally used. Figure 47 contains two examples of such codes.

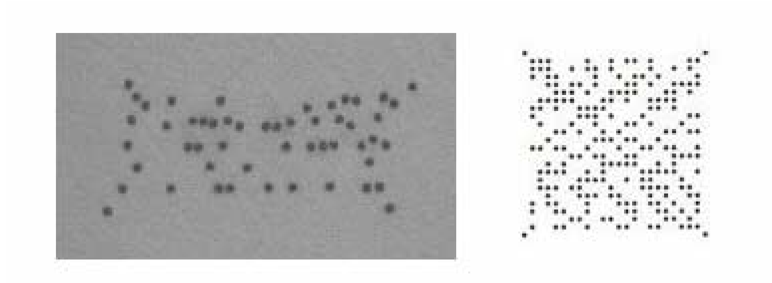


Figure 47: Examples of SnowFlake codes.

Several parameters can be set to improve the processing speed of the SoftSensor. The density, format, and type of error correction can be auto-detected. However, if these parameters are known, they should be manually specified to obtain faster reads. Other parameters that can be set include code symmetry, dot size and color (dark code on light background and vice versa) The SnowFlake Reader SoftSensor can be set to pass based on a match string.

Optical Character Recognition (OCR)

The DVT OCR (Optical Character Recognition) is a powerful industrial reading tool that provides robust character reading and verification capabilities. It is designed for reading lot codes, serial numbers, labels, product IDs, and symbols in the industrial environment. It is not designed to be a page reader that reads scanned-in documents.

The DVT OCR SoftSensors are based on feature extraction that is more powerful and reliable than normal correlation algorithms that are used in most low-end readers. Feature extraction computes the features of a graphical object and compares them to features of the trained or learned character set. Since DVT OCR uses feature extraction, it has several advantages over normal correlation based systems.

The feature extraction is a faster process; it can handle scaled and rotated characters. Feature extraction is fully trainable, which means even symbols, foreign characters, or mixed font sets can be read. This is a very important feature; the SoftSensor is not limited to OCR fonts, which are fonts designed to be read by standard OCR readers. This tool can be used to read anything, the user only needs to train it first (i.e. teach the SoftSensor what needs to be read. In addition, it can be trained on a sequence of characters so that it can be used as an OCV SoftSensor. Instead of simply reading a code, it can also verify that a certain code is present. Unlike normal correlation, it can handle any type of character marking, such as dot matrix, laser etching, laser marking, ink jet, video jet, dot peening, etc.

The OCR SoftSensor can be trained on any type of character with any type of font; however, some font styles work better than others. Special OCR fonts have been developed to make reading easier by making characters with the same width (i.e., II and WW are `II` and `WW`) and by making similar characters as different as possible (i.e., 8 and B are `8` and `B`). Other factors that will help you achieve the most reliable and robust readings are:

- Achieve high contrast between the characters and the background.
- Make sure characters have good separation.
- Maintain a character height of between 20 and 35 pixels.
- Try to minimize noise in your image background.
- Keep characters as similar to the learned characters as possible.

This type of SoftSensor has two different types of threshold: intensity based and OCR. Intensity based threshold methods include the previously discussed types (static, dynamic, and color reference). The OCR thresholds are unique to OCR SoftSensors. This type of threshold is designed to maximize the extraction of the characters for more accurate readings. There are three options for the use of the OCR threshold:

- OCR Computed: Automatic intensity based threshold that makes an intelligent decision as to what is text and what is background, and sets the threshold level to better separate those two. It calculates a single threshold level for the entire area of the SoftSensor.
- OCR Linear: Same as OCR Computed but designed for uniform variations in light. It computes an individual threshold for the first and last characters and applies a linear variation of those values for the characters in between. The user needs to specify how many characters (blocks) the SoftSensor is supposed to read.
- OCR Nonlinear: Same as OCR Linear but for cases where the variation of the illumination is not uniform. The SoftSensor in this case computes one individual threshold for each character. This option requires more processing time than all other mentioned here.

This SoftSensor is very flexible; the user can customize it to the application. A great feature of this SoftSensor is the set of available shapes. The SoftSensor comes in three different shapes: rectangle, parallelogram, and arc. The rectangle is used for text in standard position. The parallelogram is used for text that is inclined, like italics or text written at an angle. The arc is for text that follows a circular arc. Figure 48 shows four cases where the different shapes are being used.

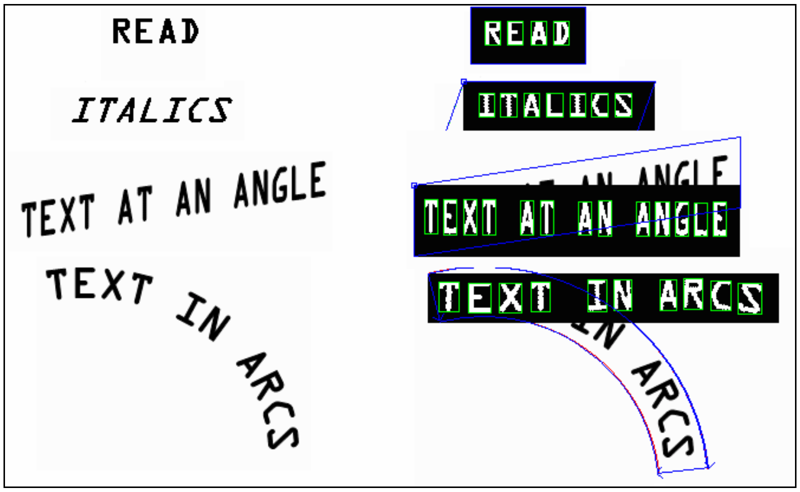


Figure 48: Uses of the different shapes of OCR SoftSensor. The top line uses a rectangle, the second and third lines use a parallelogram, and the last line uses an arc.

As the image shows, besides reading the letters in many positions, the SoftSensor offers image marking that straightens the text so the user can read it as well.

After the SoftSensor is drawn and the threshold is set, the user needs to isolate every character. That is, every character must have a red bounding box around it. When that is not the case, the user can change the extraction parameters (which limit the size of the characters in width and height(, filter the image to eliminate noise, eliminate stray pixels, etc. The extraction section of the SoftSensor is a combination of very powerful processing options. As the user adds options the processing time for the SoftSensor will increase, so a general rule is to improve the image as much as possible to make the inspection more reliable and to minimize the processing time. Figure 49 shows an image containing non-conventional characters. This type of character is very difficult to read because of the deformation that it contains. The first image shows the text itself, and the second one the first attempt to isolate the characters.

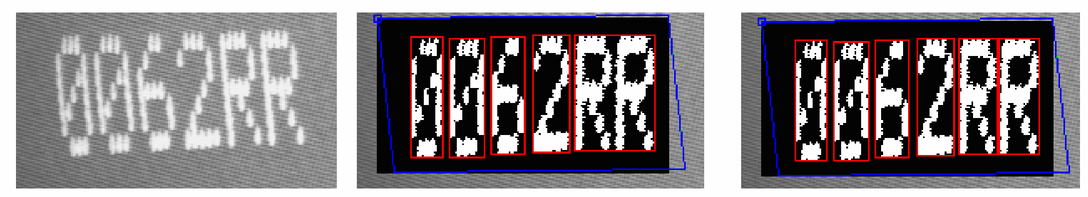


Figure 49: Use of the extraction tools to separate individual characters.

As the image shows, the last two letters share the same red bounding box. That means that the algorithm failed to correctly isolate every character. When the extraction parameters are being used (in this case maximum character width and advanced maximum character width) the SoftSensor makes a better decision as to where the characters end and they are correctly isolated.

The next step in the use is the training phase. The SoftSensor needs to be told what the extracted characters represent. This option is very powerful because anything can be trained, from user's handwriting to stroke characters. The only characters that are not trained are the spaces because the SoftSensor ignores them. Also, for applications where the speed of the process causes the characters to vary, several instances of each character could be trained to maximize reliability. As characters are trained, a font list is created and kept in the SoftSensor. This font list can be backed up to a PC for security purposes. If another system is to be reading the same labels, the font list could be loaded into the new system without the need of training all the characters again. Figure 50 shows a screen capture of the training tab for the OCR SoftSensor of the example in Figure 49. It shows the characters being trained (the string typed by the user) and images of the characters being learned with some information about them. This is the font list that can be backed up to the PC and shared with other SmartImage Sensors.

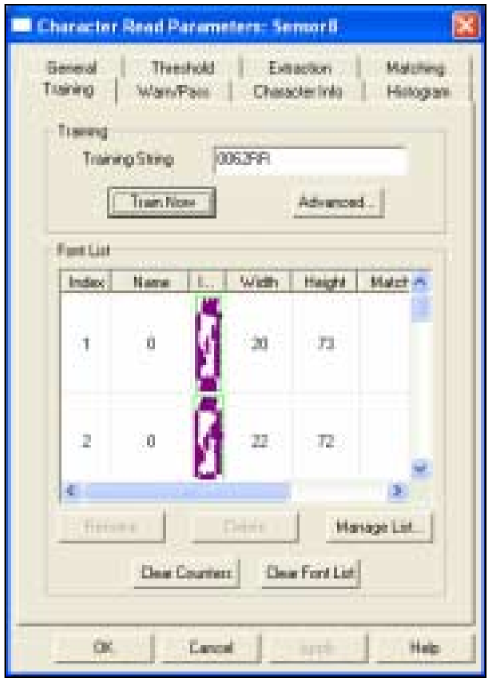


Figure 50: Training tab of the OCR SoftSensor. The list of characters is kept in memory as a font list.

After training the characters, the user needs to establish the acceptance threshold that the characters being read need to match in order to be identified as one of the learned characters. When the SoftSensor has successfully learned a character or matched it to a learned one, the bounding box around the character changes from red to green. The user needs to specify a matching threshold that dictates the percentage of similarity that the characters need to have to be considered the same character. For sharp images with even illumination and very distinct characters, a high percentage could be set. As the image quality decreases, or variation between similar characters increases due to a poor printing process, the matching threshold must be reduced. In cases where some characters tend to be deformed but still need to be read, the ones that do not match the learned model should be trained as new characters. In such cases, the font list will increase for the SoftSensor will learn many instances of every character (for example 4 different uppercase A's). As the font list gets longer, the processing time for this SoftSensor increases due to the increased number of comparisons that it has to make. In order to overcome that increase in processing time, the SoftSensor offers two options: changes in resolution and auto fielding.

The SoftSensor has three different resolution levels: high, medium and low. They differ from each other in the number of features they extract from a character. High resolution mode extracts about 1600 features from the character, medium extracts about 800 and low extracts 200. The selection of medium or low resolution reduces the processing time at the expense of reliability in the identification of the characters. It should be mentioned that the different modes apply only to comparisons. That is, the characters are always learned in high resolution mode to make the comparisons more reliable. The second option to reduce the processing time, increases reliability as well and is called character fielding. Character fielding allows the user to specify the type of character to look for in the different positions of the string to be read. When the application allows for it, the user can specify what type of character to look for in every position of the string (code) to be read. For example, if a particular code consists of two upper case letters and two numbers, the user could specify that in the SoftSensor parameters. This will tell the algorithm to check the first two characters against all 26 upper case letters and the last two characters against all 10 numerical digits. This reduces the processing time by reducing the number of comparisons and makes it more reliable (ex: number 8 will not be confused with letter B because it is not being compared against it).

Another powerful feature of this SoftSensor is that it can search for characters. Once the SoftSensor learns some characters, it can be set to find one or all of the characters. This could be used for position reference for other SoftSensors (even for a second OCR SoftSensor). Before starting a search, the SoftSensor scans the list of trained objects to determine the maximum and minimum height and widths. From this, it creates a search window that is moved inside of the region of interest. When a "potential" edge of data is found, the SoftSensor moves through the font list, one object at a time, placing a "window" the size of that trained object at the location of the "potential" edge. The edge is assumed to be the lower left hand corner of the window. As each window is placed, a read is made attempting to match whatever is in the window with any trained object. If more than one object matches, the object with the highest score is returned. If no object matches, the search window continues from where it was last, progressing bottom to top, left to right.

Searching has significant advantages and some significant disadvantages. The major advantage is the ability to find multiple targets from multiple trained objects. The major disadvantage is speed. Careful consideration must be given when evaluating the benefits of using the OCR search versus other search and translation options. Although the OCR SoftSensor does return its location, it does not return a sub-pixel value and does not provide any rotational information. Depending upon the quality and variability of the images, location values may be only "coarse" positions. DO NOT USE THE POSITION VALUES FOR ANY PURPOSE OTHER THAN LOCATING ANOTHER SOFTSENSOR. Since the search algorithm creates a box the exact size of the trained objects, searching will not find objects larger than those trained. It can find objects that are smaller, however. When searching for text, the acceptance level is generally set higher than normal. This will result in additional training. When searching for patterns, the acceptance level is generally set lower than normal. This will allow for a wider variability. When training patterns, place the ROI as tight to the pattern as possible before training to minimize the effects of noise.

This SoftSensor can be set to pass or fail based on a string to be matched, one or more characters not being read, and a character being read but with a low score. The score represents how well the character matches the learned character. Scores range from 0 to 1000 to show how well the character is matched, a score of 1000 represents a perfect match.

Blob Tools

Blobs in FrameWork are defined as areas of connected pixels of similar intensity. Blob tools are SoftSensors designed to find, count, or track blobs. There are two Blob Tools SoftSensors: the Blob Generator and the Blob Selector.

The Blob Generator analyzes the image and isolates the blobs based on user options (threshold level and type of blob to locate). There are two types of threshold available for blob generation: intensity based and gradient based. There are several intensity based threshold options and one gradient based (edge detection). For an explanation of these options, refer to the section on thresholds. Once the blobs are located, the image processing options (Erode, Dilate, Close and Open) can be performed to eliminate noise in the image. If the image requires more dedicated processing options, the user can make those available from the FrameWork SoftSensor settings (advanced options under the "Edit" menu). These advanced options include image processing options in only one direction and filters to extract blob edges instead of the entire blobs. When the image contains noise, the number of blobs reported by the SoftSensor may be affected by it. When that happens, the SoftSensor becomes less reliable and the processing time for the SoftSensor increases due to the extra blobs being processed. In order to overcome this type of situation, the user can limit the blob size. By doing this, blobs caused by noise (usually very small) can be filtered out of the SoftSensor. Using this size feature of the blob generator saves a lot of processing time because the blobs are filtered out before any heavy computation is performed on the blobs found.

This SoftSensor has a very specific image marking. Green corresponds to background. Gray corresponds to blobs that are being counted. Red corresponds to boundary blobs (blobs that touch the edges of the SoftSensor). In the case of the boundary blobs, the user selects whether those blobs should be counted or discarded. If the option is to count them, they will be marked in gray just like regular blobs. Finally, black and white corresponds to blobs that have been discarded because of their size: black blobs are undersized blobs, and white blobs are oversized blobs.

The Blob Selector is used to isolate blobs created by the blob generator. The main use of this SoftSensor is to help the user find the blob(s) that the application requires by calculating a number of parameters about the blobs(s). If the objective is to track a part for rotation and translation, a particular blob must be isolated and its position must be calculated. When this is done, other SoftSensors can reference the blob selector for position reference. If the objective is simply to count the blobs or to verify their correct size/shape, the SoftSensor can do that automatically. This SoftSensor does not have to be drawn; it simply references the blob generator and uses the area delimited by the boundaries of it. As a consequence, it does not have a threshold to set; it simply extracts the blobs from the blob generator and performs calculations based on them. The parameters that the SoftSensor can extract from every blob are: angle, position, area, bounding box size, eccentricity, compactness, perimeter, average intensity, and radius. For an explanation of these parameters see Table 3 at the end of this section. The user can choose to calculate some or all of these parameters to help discriminate between the desired blobs and other blobs in the image. The SoftSensor can display the value of the selected parameters for every blob in the image.

The image marking for this SoftSensor is different from the blob generator. The background is black, and blobs are highlighted in gray. The blob used for position reference is highlighted in white. Discarded blobs are shown in a darker shade of gray.

Note: the user must set the SoftSensor parameters to select only one blob when doing position reference. That is, if a part needs to be tracked, the position of that part only has to be passed to other SoftSensors.

Figure 51 shows an example where the blob selector is being used for position reference. The blob generator passed 4 blobs to the blob selector and this one selected only the blob that represents the part to be inspected. This particular selection was based on compactness and eccentricity of the blobs.

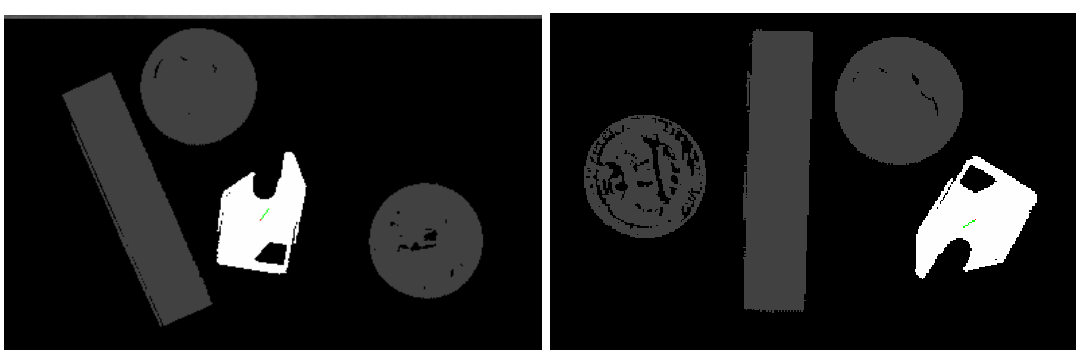


Figure 51: Use of the blob selector for position reference.

As the figure shows only one blob is being selected and used for position reference (highlighted in white). The image also shows the marking for the orientation of the part, the mark at the center of it. This mark consists of a green line (which indicates the angle) starting from a red dot (which marks the center of the blob).

Figure 52 shows an example where blob tools are used for flaw detection and part counting. The first image shows four pills. The second image shows the image marking for the blob selector when all four pills are being found. In this case the SoftSensor can be set to pass based on pill count.

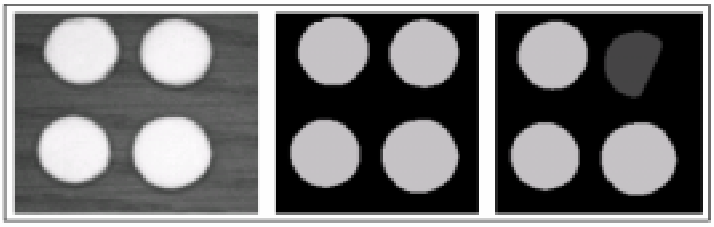


Figure 52: Example of use of blob tools for flaw detection/part counting.

The last image shows the marking for a case where one of the pills is defective. The damaged pill is discarded so the SoftSensor only finds 3 pills. In this case the SoftSensor should fail based on the blob count. Notice that in this case, since there is no positioning reference being used, the SoftSensor does not mark the center and orientation of the blobs.

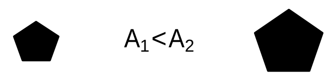
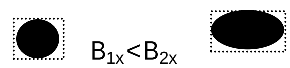
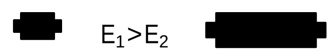
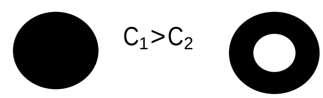
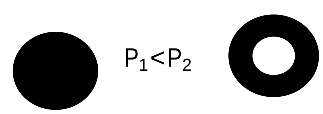
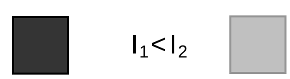
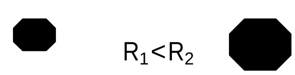
Parameter	Description	Example
Angle	Measured from clockwise from 3 o'clock position. Must check "Calculate Blob Angles" to then select blobs in range.	$A_1<A_2$ 
Position	Position of blob centroid as measured from image origin (upper left corner). Must check "Calculate Blob Positions" to then select blobs in a certain range.	Blob 1 (142, 83); Blob 2 (140, 96) $X_1>X_2$; $Y_1<Y_2$
Area	Measured in pixels. Selecting blobs based on area is a very common way to sort shapes.	 $A_1<A_2$
Bounding Box	BB is the width and height of the rectangle that encloses a blob.	 $B_{1x}<B_{2x}$
Eccentricity	Measures rotational symmetry of a blob. 0-1 scale with most eccentric shapes near 0 (very asymmetrical) and least eccentric near 1 (very symmetrical). Independent of scale.	 $E_1>E_2$
Compactness	A measure of blob density. 0-1 scale with least compact shapes near 0 (less dense) and most compact near 1 (more dense). Independent of scale.	 $C_1>C_2$
Perimeter	Measures the total length of all edges around the blob, including interior edges.	 $P_1<P_2$
Avg. Intensity	Measures average intensity value of the connected pixels.	 $I_1<I_2$
Radius	The radius in pixels of a shape. For non-circular blobs, this value represents the distance from the center of the blob to the point that is farthest away from it.	 $R_1<R_2$

Table 3: Meaning of the parameters associated with the blob selector.

Template Match SoftSensor

Template Match SoftSensors are used to check pixel patterns against a learned template. The main function of this SoftSensor is defect detection. There are two Template Match SoftSensors from which to choose: Template Match: Fixed SoftSensor and Template Match: Searching SoftSensor.

The Fixed Template SoftSensor assumes a consistently placed part or a position reference from another SoftSensor. This SoftSensor learns the pixel pattern contained in the rectangle (this is the only available shape for this SoftSensor) and compares the subsequent images to the learned one. This SoftSensor can use any intensity based type of threshold to determine which pixels are light and which are dark. The user then selects whether the light, dark or both groups of pixels are used for the comparison. The selection of one of the groups of pixels decreases the processing time, but it is recommended to use both groups of pixels for reliability. Figure 53 shows an image sequence where the part that is present in the first image is being inspected for errors. If only the dark pixels are selected for the inspection, the SoftSensor will verify that all the original dark pixels are present in subsequent images.

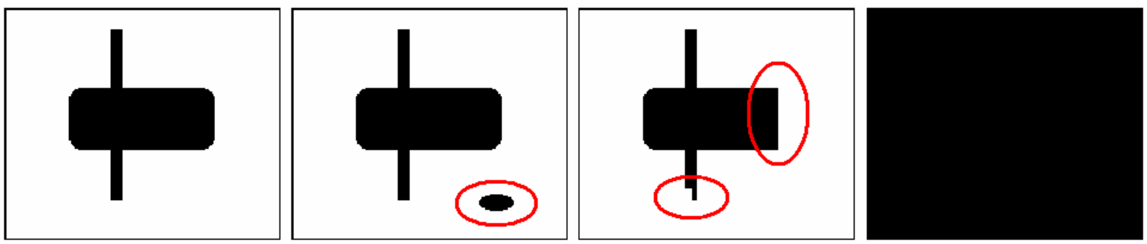


Figure 53: Sample images that illustrate the use of only dark pixels for the inspection.

As the SoftSensor inspects the second image, it finds no errors because all the dark pixels of the part are present. The background noise is not picked up because it means error in the originally light pixels. When it inspects the third image, it detects both errors in the part. When it inspects the last image, it reports no errors because basically all the dark pixels from the original image are still dark. This image sequence illustrates why both pixel groups should be used.

A powerful option that this SoftSensor offers is the option to discard pixels near edges. There are minor differences between images, which can be observed by zooming into the image and observing the pixel intensities. This SoftSensor is powerful enough to detect those changes as well, and the areas that are most sensitive to those changes are the part edges. Therefore, the SoftSensor is likely to report some error near the edges even if the part is flawless. To overcome this issue, the user can select a number of pixels near the edge to be disregarded in the inspections. This operation is performed before checking for errors in the template in order to minimize processing time. If the object contains fine details, the user should select to keep the skeleton when disregarding pixels (available from the “Advanced Model Parameters” window). This option will ensure that the geometry of the object is preserved. Figure 54 shows an example where the option of keeping the skeleton of the part is used to keep the overall geometry while still disregarding some pixels.

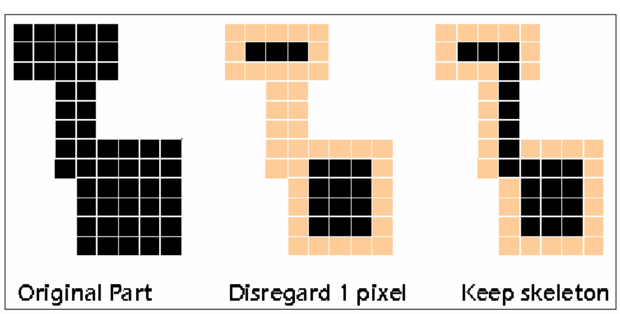


Figure 54: Use of the keep skeleton feature to overcome drastic geometry changes when using the option to disregard pixels near the edge.

Unlike most SoftSensors, the Template Match SoftSensor does not offer threshold based image marking. That is, the user should not expect to see any image marking until inspections are being taken. At that time, areas of the image that are different from the original image will be highlighted in red to indicate areas where the part is potentially defective. The only marking drawn with the SoftSensor is the segmentation of it. The area inside the SoftSensor is automatically divided into segments (rectangles). The reason for this is that this SoftSensor can be set to pass or fail based on the amount of error: either the number of wrong pixels or this as a percentage of the total pixels. When the option being used is the percent of total, a small error may be a very small percentage of the entire SoftSensor. In those cases, the user can limit the per-segment error, which will always be a higher percentage than it is for the entire SoftSensor.

NOTE: Although this SoftSensor can be attached to a positioning SoftSensor to inspect the correct location, it can only translate. That is, the SoftSensor will not inspect the part correctly if it is rotated.

The Template Search SoftSensor performs the same inspection as the fixed type, except that first it searches for the template in a predefined area. The SoftSensor consists of two rectangles: the inner rectangle is the template to be learned and inspected, and the outer rectangle consists of the search region, or where the user expects the template to be located. In this case, the division into segments of the template region has a different meaning. When the SoftSensor is drawn, the template region is divided in segments. After that, the SoftSensor determines how rich in features the segments are, assigning a certain importance to each one of them. There are two ways to select the important segments: monotony based (gives high importance to corner points) and variance based (gives more importance to image details). This process extracts the two most important segments and marks them different from the others. The most important segment is marked with a green plus sign inside a red bounding box at the center of the segment. The second one is marked with a green plus sign inside a green bounding box at its center. The remaining segments are marked with a green or red plus sign at their centers. Green indicates that the segment is very rich in details/corners; red indicates that it does not contain as many details/corners. Figure 55 shows a Template Search SoftSensor. The inner rectangle was automatically divided into 4 segments. The third segment from the left is the most interesting one, while the second one is the second most interesting segment. In this case the SoftSensor allows for some translation in the X and Y directions. If the DVT logo changes the position but stays within the outer rectangle, the SoftSensor should find it and inspect it at that location. This SoftSensor does not track rotations, so if the part rotates a different SoftSensor should be used.

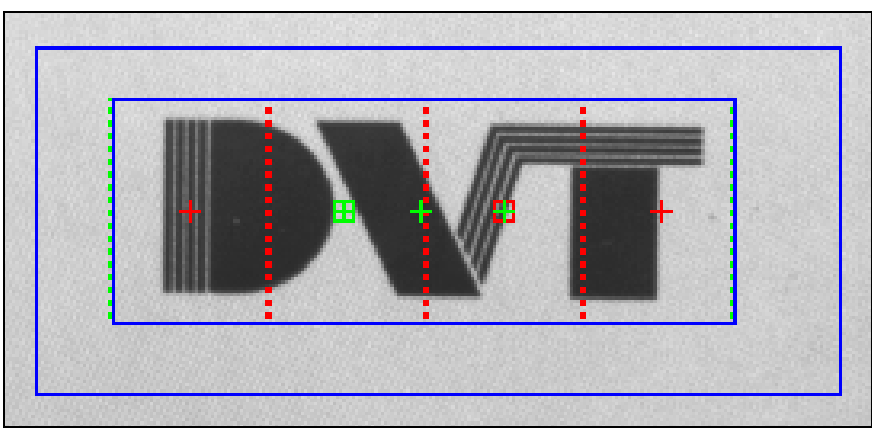


Figure 55: Template Search SoftSensor finding the DVT logo and verifying that no errors are present.

For this specific SoftSensor, the division of the area to be inspected into segments serves yet another purpose. When the SoftSensor searches for the template, it does not search for the entire pixel pattern because that would require a lot of processing time. It searches for the most interesting segments instead. Once it finds a few potential locations for the template it then compares the remaining segments based on the location they should have with respect to the potential locations of the most interesting ones. The user has a few options to speed up the process such as skipping some pixels in the original search and fine tuning the position found by moving the template a number of pixels in all directions starting from the potential location generated by the first search until the best match is found.

Like the fixed template SoftSensor, this SoftSensor can be set to pass or fail based on the amount of error detected inside the SoftSensor or inside the segment that contains the most error.

ObjectFind SoftSensor

ObjectFind SoftSensors use advanced algorithms to learn the characteristics of an object and find occurrences of that object in subsequent images. ObjectFind SoftSensors can be applied in a variety of SmartImage Sensor applications, including:

- **Flexible feeding, motion control, robotics:** Parts could be presented in any orientation and may overlap.
- **Short product runs:** The key to these applications is quick setup because of the variety of parts manufactured.
- **Position finding/referencing:** ObjectFind's learning ability means quicker setup when compared to using Translation, Rotation, or even Blob Tools.

When the ObjectFindSoftSensor is drawn it binarizes the image based on the threshold selected. Then, it analyzes the dark or bright object present (according to user selection) and extracts a number of features that describe the geometry of the object. The list of features is kept in memory and the SoftSensor uses that list as the learned model. In subsequent images, the algorithm searches for the features contained in the model to determine the presence of the learned object. There are two types of ObjectFind SoftSensors: single area and double area. The single area type must be drawn around the object. After the SoftSensor learns the features of the object, it can be resized to cover the area where the object could potentially be located. The double area SoftSensor uses two areas. The inner area is for learning the object and should be drawn just outside the object. The outer area indicates the region where to search for the object. Both SoftSensors perform the same task, but the double area SoftSensor is preferred in cases where an object that changes position over the entire area needs to be relearned using the digital relearn signal. In order to do this, the new object to be learned must be placed within the inner rectangle.

ObjectFind SoftSensors have a number of parameters that the user can set to better find the objects. First, it can be set to find a single object (when used for position reference) or to find multiple objects (when used for part counting). In order to minimize processing time, the user can limit the movement that the objects are to have. That is, when the application allows for it, the user could limit the movement in the X and Y directions as well as rotation, so the search for the object does not take much processing time. The distances and angles indicated by the user are relative to the position that the object had when its model was learned. Likewise, when the SoftSensor is set to look for multiple objects, the user can specify a minimum object separation in pixels (translation) and in degrees (rotation). When this option is used and objects are found within the minimum allowed, only the object that better matches the learned model is reported. If the distance between the objects and the lens tends to vary, objects will seem to vary in size. For these cases, the SoftSensor offers a scaling tolerance parameter which the user can select to identify objects when they are proportionally bigger or smaller than the learned model.

Another important parameter that ObjectFind SoftSensors have is flexibility. The user can set the SoftSensor to be more or less flexible. The SoftSensor should be less flexible when the object to be located does not vary in size or geometry. If the parts are neither defective nor overlapping, the SoftSensor should only recognize the presence of an object when many features of the learned model are present in the area. This option makes the SoftSensor less susceptible to noise. For the opposite case, when objects tend to overlap or even change in geometry, the SoftSensor should be made more flexible. This will allow the SoftSensor to find objects where there are only a few characteristics of the learned model. The user should keep in mind that the more flexible the SoftSensor is, the more susceptible to noise it becomes. That is, if background noise or even other parts in the image present some of the features that the learned model contains, those regions will be identified as a new object. The parameters that allow the SoftSensor to be more or less flexible are:

- **Scale Tolerance:** Allows the SoftSensor to find bigger and smaller objects
- **Object Matching Parameters**: These indicate a minimum percent of features and perimeter that must match the learned model for the object to be indicated as a match
- **Feature Matching Parameters:** Lowering these permits some deformation in the part being analyzed while still considering it a match
- **Polyline Parameters:** Determines how the polygon that follows the contour of an object is drawn, as well as the minimum perimeter that an object feature must have to avoid being discarded (to filter out noise)

All these categories consist of a number of different parameters, for explanations about each one and how to change them please refer to the FrameWork help files. To understand how much to change the values, please refer to the SoftSensor properties: under "Object Info", a number of parameters are displayed corresponding to every object that is found in the image.

Image marking for this SoftSensor is different from the others. The user can customize it by choosing among four options: intensity levels (black and white), edges found in the image (highlighted in red), edges of objects found (highlighted in green), and model reference point (center and orientation of the object). Figure 56 shows the image marking for the SoftSensor with all four options enabled. First, all the edges determined by changes in intensity level (above or below threshold) are marked.

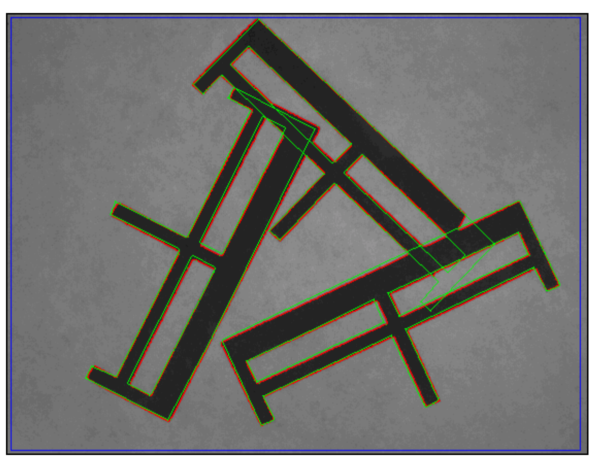


Figure 56: Image marking for the ObjectFind SoftSensor.

This marking is red; it corresponds to any transition from bright to dark. A second marking (in green) can be observed over the edges of the objects even when they are incomplete of overlapping. This marking corresponds to the learned model and it indicates that an object was found in this position. Even though the object in the image does not match the model point by point, most geometric characteristics of the model are found in this object, so it is being identified as a match. These two types of marking are unique to this SoftSensor. It can also mark the intensity levels (white for areas above threshold and black for areas below threshold) and the center of the object. There is another type of marking available from the edges of the objects. Based on user defined parameters, the algorithm sets a number of corner points along edges which are part of the object model. Those corner points are highlighted in blue and they are on top of the red marking that separates the object from the background.

This SoftSensor is mostly used for position feedback, for another SoftSensor or to a robot. However, if the application consists of counting the parts, the SoftSensor can be set to pass or fail based on the number of objects found.

Intro to Color SoftSensors: The RGB Color Space

In order to better understand the functionality of the Color SoftSensors the color space should be explained. As was mentioned before, color systems report four different values from each pixel. Those values refer to the color content and the intensity level. The color content is broken into three different coefficients to indicate the content of each basic color: Red, Green, and Blue. Any color can be created using the correct amount of Red, Green, and Blue, so by having the system report the contents of all three basic colors, any color can be represented in the system memory. The fourth parameter that is reported is the intensity level. This parameter would be the equivalent to the intensity as discussed for grayscale systems, and is computed from the values obtained for the basic colors according to certain weight associated to each one (based on the wavelength of the color). All four parameters (R for Red, G for Green, B for Blue and I for Intensity) range from 0 to 100, so color identification occurs in a three-dimensional space of R, G, and B coordinates whereas the intensity is an independent value. Figure 57 shows this color space. The X, Y, and Z axis are the R, G, and B components. A certain color could be defined by three coordinates; however, that will give a very specific color. In practice, colors are defined by ranges of R, G, and B. This defines a volume instead of a point as shown in the figure. Points inside that volume are of that color, points outside that volume are not of that color.

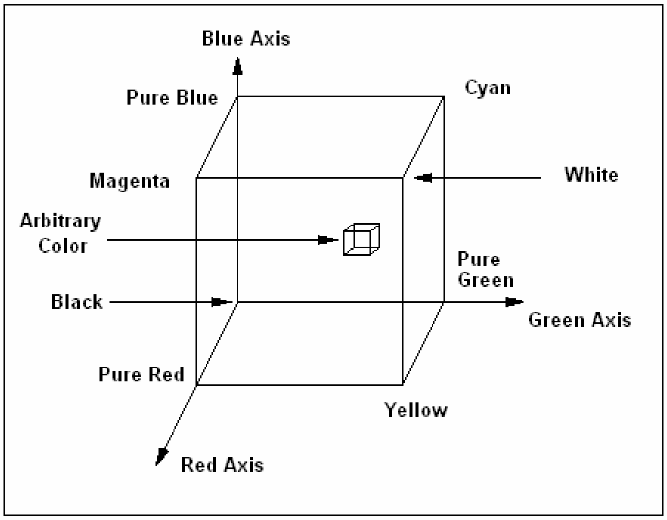


Figure 57: Three-dimensional color space defined by the basic colors Red, Green, and Blue.

This definition of the color space defines a new method for thresholding the images. The boundaries of the volume that defines the arbitrary color are the new threshold. Colors inside that volume are defined to be above threshold. Likewise, colors outside that volume are defined to be below threshold. As an example, if an image contains a number of objects of different colors, the user needs to learn the desired color. Once that happens, the user can use the Dominant Color type of threshold to connect the threshold to the learned color. By doing this, the object of the learned color will be identified as above threshold (highlighted in white). Likewise, the other objects will remain below threshold (outside the volume) and will be marked in black.

Pixel Counting SoftSensor (Color SoftSensor)

The Pixel Counting SoftSensor is the basic Color SoftSensor. It is called Pixel Counting to emphasize what it does and to highlight the fact that its use is not limited to color systems. This SoftSensor learns a color (or many colors) and determines the number of pixels of that (those) color(s) that are found on subsequent images. If a grayscale system is used, the SoftSensor learns a shade of gray and uses it as if it was a color. There is not a concept of threshold in this SoftSensor. The SoftSensor is drawn and based on three parameters it learns about a color. Those parameters are:

- Sigma Factor: Changes the size of the volume that represents the color in the 3D color space. Increasing it will include more (similar) colors in the volume. Decreasing it will make the color selection more specific.
- Matte Areas: Help the user select darker and lighter shades of the same color. In the 3D color space the matte line starts at coordinates 0, 0, 0 and passes through the selected color. The darker shades of the color are located between the color and black, the lighter shades are located on the other side of the matte line.
- Highlight Areas: Are used to include areas of the color that contain direct reflection of light. The areas affected by reflections tend to be white, so the highlight line goes from the selected color to white (coordinates 100, 100, 100).

Figure 58 illustrates the effects of the use of these parameters; the first diagram shows a learned color. The color occupies a limited volume in the RGB space. The image on the upper right corner shows the effect of increasing the sigma factor. The volume increases in all directions bringing similar colors together. At this time the color SoftSensor recognizes any color inside the new volume as the learned color. This makes the color selection more general. If the user wanted to make it more specific, the sigma factor should be decreased to reduce the size of the original volume.

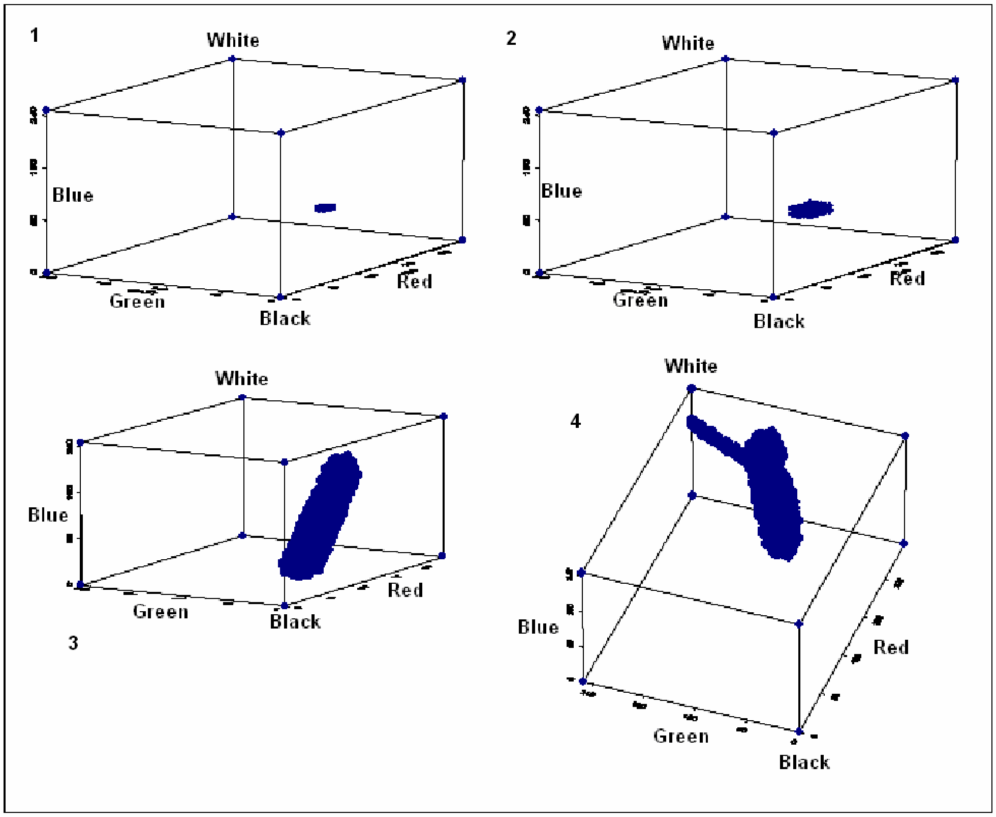


Figure 58: Three dimensional illustration of the effect of the color SoftSensor parameters. The first image shows the learned color, the second one the effect of increasing the sigma factor. The third one shows the effect of increasing the matte areas, both for darker and brighter shades of the color. The last image shows the effect of increasing the highlight areas.

The third diagram (lower left corner) shows the effect of using the matte areas. The volume is expanded towards black (coordinates 0, 0, 0) to pick up darker shades of the color or away from black to pick up brighter shades of the color. The user can select how much to extend the volume and in which direction (darker, brighter or both). Finally, the last diagram illustrates the use of highlight areas. This option is intended to include shades of the color that are caused by reflections. Those shades are between the original color and white (which would be a full reflection of the light). Increasing this factor expands the original volume towards white.

Figure 59 shows an example of the use of the sigma factor. The color SoftSensor learns only the light orange on the left end. It is then expanded over all the colors and image marking shows where the color is identified. For the example on the top row, it happens to be only over the learned color. This example uses a sigma factor of two. For the second example, the sigma factor was doubled. For a sigma factor of four, it identifies other colors as well. The last two examples represent sigma factors of seven and ten respectively. The SoftSensor is able to pick up different colors, colors that do not reside inside the original matte line, but as the sigma factor is expanded, those other colors are included in the volume that identifies the learned color.

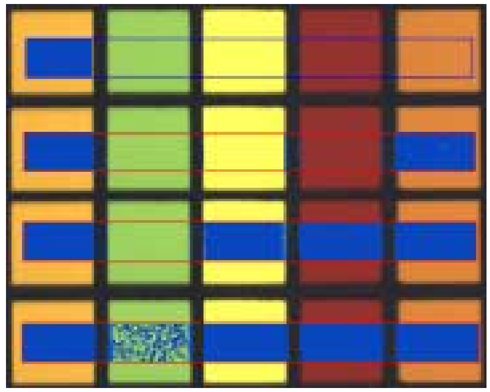


Figure 59: Illustration of the use of sigma factor to include other colors in the selection.

This SoftSensor can learn colors one by one or many at a time. When the SoftSensor is used to learn colors one by one, it must be resized to cover only the area where the desired color is present. This procedure should be repeated to learn as many colors as necessary. The SoftSensor will keep a list of all the learned colors. For complex objects where individual colors are not well defined, the user should use one of two techniques to learn several colors at a time. In FrameWork 2.6 a new tab was added to the Pixel Counting SoftSensor: the “Advanced Training” tab. This tab includes the previously existing clustering mode plus a new set of options for multi-color learning.

When using clustering, the user specifies the minimum cluster size, which indicates the minimum number of pixels that a color must have in order to be added to the list. In this case a fixed standard deviation is used. When the user learns clusters, the SoftSensor learns all the colors in the specified area and makes a list of them. The user can then manage the color list by merging some or all colors into one, or by deleting colors that are no longer needed.

There is also a new set of options for learning multiple colors at a time. These are on the “Multiple Colors” section in the “Advanced Training” tab. There are three choices: ReLearn, Remove and Add. Through the first option the user can learn a group of colors and merge them together all in one operation. The multiple remove and add operations will act on a group of colors previously learned and merged. The remove process will delete from the aforementioned group the colors that are in the region of interest of the SoftSensor at the time this action is executed. The add operation will do the opposite: it will add colors that were not present in the previously learned and merged group of colors. After executing any of these options, the user will end up with multiple colors grouped together under a single entry on the “Manager” tab.

One important consideration about this SoftSensor is that whereas the single color learning adds colors to an existing list, clustering mode replaces the existing list with a new list of the colors extracted from the part. The multiple color relearn option will also substitute any previous color(s) learned, this time with a single entry in the “Manager” tab that will include all colors learned and merged together.

Another feature that these SoftSensors present is the digital relearn. They can be programmed to expect an external signal (digital relearn), and when the signal is received, the SoftSensor executes one of several operations. These options go from a single color learning operation to automatic multi-color learn and merge processes. The digital relearn signal can be used to change the match color as well. The match color is the color against which every image is being compared to determine a PASS/FAIL result.

The uses for this SoftSensor are:

- To detect the correct color: The SoftSensor can be set to pass or fail based on color variation.

- To detect the right amount of color: The SoftSensor can be set to pass or fail based on changes in the amount of a certain color.
- To identify parts: The SoftSensor can be used to determine which part is in front of the camera based on the "best match" feature, which tells the user which color from the list best matches its original size in pixels.
- To export colors: The SoftSensor can learn, merge and export the colors to other SoftSensors to be used as a threshold.

When the pixel counting SoftSensor is being used just to export the colors it learned, it might not be necessary for the system to keep processing the SoftSensor in every inspection. The user has two choices: delete the SoftSensor from the product or keep it around in case new colors have to be learned. A new feature allows the user to disable the SoftSensor so that it can stay around but not consume processing time. This is not to be confused with the "Active" option available under the "General" tab, which does not prevent the SoftSensor from being processed (its function is to disregard the result of the inactive SoftSensor at the time the overall result of the inspection is computed). When the option "Disable Sensor" (under the "Advanced Training" tab) is activated, the SoftSensor is only processed and inspected when a digital relearn signal is received and new colors are to be learned.

Color Monitoring

The Color Monitoring SoftSensor is another type of Color SoftSensor. It is a much more precise SoftSensor than the Pixel Counting SoftSensor. This SoftSensor can differentiate approximately 16.7 million colors, whereas the Pixel Counting SoftSensor can only discern about 65000 colors. As a result, the SoftSensor is used mostly to identify different colors when they are very similar. This SoftSensor reports the difference between colors. If many colors have been learned, the SoftSensor reports the color difference with respect to the closest color. The SoftSensor can be setup to work in the following color spaces:

- RGB is the default color space. Color difference is calculated as the Euclidian distance between two points. If Matte Line is selected, distances are measured to the matte line instead of the saved average color.
- Optical Density is the second option; in this case the color difference will be computed on a base 10 logarithmic scale. This color space is mostly used in the printing industry. RGB shows the reflectance of the measured sample, but Optical Density shows the absorption. The average Cyan, Magenta, and Yellow density is displayed. The reported color deviations represent the change in optical density of CMY components compared to the Closest Color. Color Difference is the maximum of DC, DM, and DY
- La*b*ΔE, this is CIE LAB. Please note that La*b* values are approximate. Standard formulas are used to compute these values.
- LC*h°ΔE_{cmc}, L is the Luminance or Intensity component of the sample. C* is the Croma or Saturation component and h° is the Hue angle. Color difference is ΔE_{cmc}.

During Inspection the average color of pixels under the sensor shape is found and displayed in the result panel. Depending on the color space used, the distance between this point and every color in the list is computed. The "Closest Color" is the one with the smallest distance. This distance is displayed in the results panel as Color Difference. Figure 60 illustrates how the distance is reported in the RGB space. The SoftSensor computes the Euclidian distance from the color found in the image to all the learned colors to determine the closest color. Once this color is found, the SoftSensor reports the color difference in each channel (R, G, and B) as well as the overall distance shown in the image.

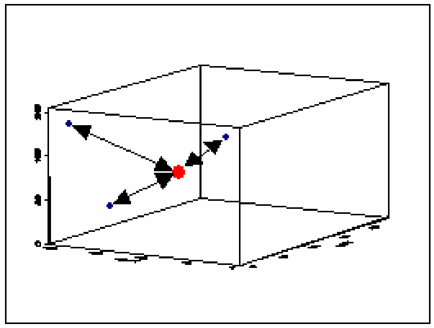


Figure 60: Illustration of the output (color difference) as the Euclidian distance in the RGB space. The SoftSensor reports the distance to the closest of the three colors that in this case have been learned.

An important consideration about this SoftSensor is that it does not mark the image. When the SoftSensor is drawn, all the pixels inside it are taken into account for the computation of the average color. The user is supposed to utilize this SoftSensor only over uniform areas of the color. For that reason image marking is not used (because it would mark every single pixel inside the SoftSensor every time).

This SoftSensor can be set to pass or fail based on five different criteria:

- Maximum Change in Color Data 1: Depending on the color space used it can be the maximum allowable change in Red reflectance, Cyan density, or Luminance.
- Maximum Change in Color Data 2: Depending on the color space used it can be the maximum allowable change in Green reflectance, Magenta density, a* or C*.
- Maximum Change in Color Data 3: Depending on the color space used it can be the maximum allowable change in Blue reflectance, Yellow density, b*, or h°.
- Maximum Color Difference: Is the maximum allowable distance between the sample color and the ClosestColor. The Color Difference is computed depending of the color space as explained above.
- Match Color Name: This allows the user to select one of the colors in the learned list. The name of the ClosestColor must match the selected color or the SoftSensor will fail.

Segmentation SoftSensor

Segmentation is the process of grouping pixels into distinct sets based on intensity changes. Each set has its own characteristics including color, pixel size, and position. Unlike blob tools, this SoftSensor does not threshold the image; it uses fuzzy logic to detect gradient changes between pixels, which determine segment edges. That is, the SoftSensor analyzes the area and determines gradient levels; if those are above a certain level, the SoftSensor marks them as a separator between different sets of pixels. These edges become segment boundaries. During the segmentation process, the pixel on the upper left corner of each segment is used as a seed. The seed pixel is then grown in four directions for area sensors and along the line for line sensors based on color/intensity tolerances. Neighboring pixels can merge with the growing segment based upon pre-defined conditions of color tolerance. In many applications, Edge Contrast is enough to separate different segments from each other. If there is a missing edge pixel, the growing segment can "leak" into the neighboring segment. To prevent a leak, a coarser seed type can be used. There are five different seed types for area sensors the finest one would get through the smallest opening that a boundary could present, while the coarsest one will "seal" the small openings in the boundaries. Figure 61 shows the use of the Segmentation SoftSensor. In this case, the SoftSensor has to count the objects in the FOV. When the SoftSensor is drawn, it determines some limits based on gradient rules. That is, the SoftSensor specifies areas where the gradient exceeds the maximum specified by the user.

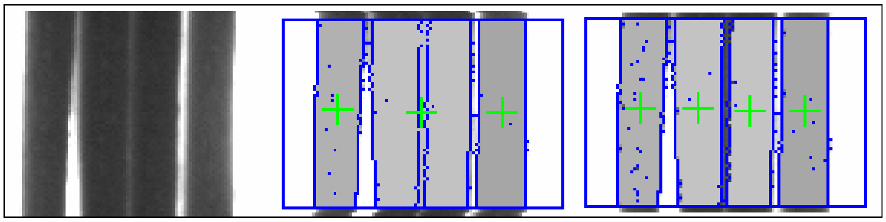


Figure 61: Use of the Segmentation SoftSensor to separate and count objects.

In some cases that delimitation is not enough as shown in the second image. In this case there are openings in the boundaries of the segments causing two of the segments to be merged into one (second image). In order to overcome this issue, the user can select a different gradient threshold or increase the size of the seed. A coarser type of seed cannot get through the smallest openings so the segments are correctly identified as in the third image. In some applications sharp edges do not exist between objects. To segment this type of image, an additional parameter called Color Tolerance can be enabled and configured, which takes color into account in the segment growing process. Based on the Color Tolerance setting, if a particular pixel varies in color from the seed color (the color of the first detected pixel) by a certain amount but still belongs to the same segment, the algorithm will set a limit to exclude it from the segment. If the boundary openings of Figure 61 were too big for the coarsest type of seed, color tolerance could be used to establish the limits between different segments.

This SoftSensor has two different modes: relative and absolute mode.

The example shown above uses absolute mode. The SoftSensor only finds and counts the segments. It has the same functionality as the blob tools, but since it uses gradient based calculations instead of a regular threshold, it is more precise when there is poor contrast (like in the example above). In such cases, the use of a threshold is very difficult: as the image shows, the separation between the two center segments is hard to identify.

The second mode is the relative mode. When working in relative mode the SoftSensor learns a set of segments and detects their presence in subsequent images. The SoftSensor can verify that the segments are present, and determine how much they have changed in position, size and color. In this case the SoftSensor adds a manager tab to its parameters and it shows a list of the learned SoftSensors in it. The user can then select some rules to determine whether the segment is acceptable or not. The available rules are maximum movement in the X direction, maximum movement in the Y direction, maximum change in size, and maximum change in color. This SoftSensor is often used for printing processes where the print register needs to be correct or one of the colors will be printed out of position. Figure 62 shows two images of a print register. The image on the left illustrates the correct position of all the segments. The image on the right shows that some of the segments have changed position. This would cause a problem with the printouts since some of the colors would be out of position.

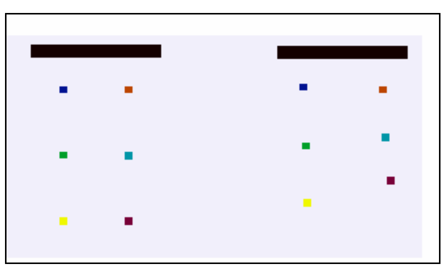


Figure 62: Common application for Segmentation SoftSensors: print registers.

For these applications, the SoftSensor is used in relative mode, so all the segments are learned (size, color, and position of each) and the SoftSensor simply verifies that the learned parameters remain within a user-defined range of acceptability.

This SoftSensor can be set to pass or fail based on the number of segments found. For the absolute mode, this relates to the segments found when the image was segmented. In that case there is no further processing. When working in relative mode there is another option. The segments must follow specific user-defined rules in order to be considered segments. In that process some segments are discarded. Those discarded segments make up for the fail count, or the number of segments that failed to follow those rules. In this case, the pass/fail criteria could be based on both segment count and segment fail count.

Foreground Scripts (Script SoftSensors)

A foreground Script is a programmable tool that allows the user to write a short program with access to data from other SoftSensors. This program, which uses a very common syntax (similar to Java, C, and C++), is always part of a product and is executed when that product is used to inspect an image, which is why they are called Script SoftSensors. Script SoftSensors can:

- Be set to pass or fail based on certain conditions, except that the user defines those conditions. There is not a predefined set of rules because of the versatility of this type of SoftSensor.
- Access data from other SoftSensors and draw a conclusion (pass/fail) based on data from several SoftSensors.
- Perform mathematical calculations using built in functions or by letting the user program new functionality.
- Access registers in memory to interchange data between different products, with background scripts or even with external devices.

For more information, documentation about all the specific functions, and examples about Script SoftSensors please refer to the script documentation and FrameWork help files.

Spectrograph SoftSensor

The Spectrograph SoftSensor is the latest SoftSensor addition to FrameWork. This SoftSensor is designed to work with the newest member of the Legend Series of SmartImage sensors: the SpectroCam. Figure 63 shows the functionality of this Hardware device. It consists of two stages of image acquisition. Stage 1 is a regular lens that has to be used to correctly focus on the part. Stage 2 consists of another optical device that takes a vertical line from the image focused by the lens and spreads the spectrum of that line over the CCD. When this image is projected onto the CCD, it consists of a grayscale image with no defined shapes in it. That is because the spectrum consists of different levels of intensity at different locations in the image. The X axis of the image becomes the Spectral axis, that is, if the external optical device is correctly placed, blue tones should be on the low X coordinates and red tones should be on the high X coordinates. Likewise, the Y axis represents the Spatial axis. This axis is the same as for a regular image but applied to a very narrow opening. In other words, a change in X coordinates would represent a change in wavelength, whereas a change in Y coordinates would represent a change in position.

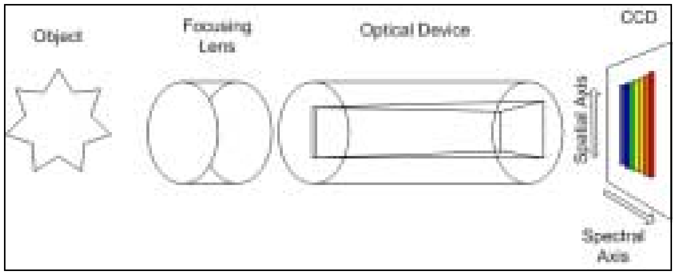


Figure 63: Functionality of the SpectroCam.

Figure 64 shows an image obtained with the SpectroCam and the pixel graph that the Spectrograph SoftSensor obtains from it. As we analyze the image, we can see that it consists of different levels of intensity. The Spectrograph SoftSensor is an area SoftSensor that is only available as a rectangle.

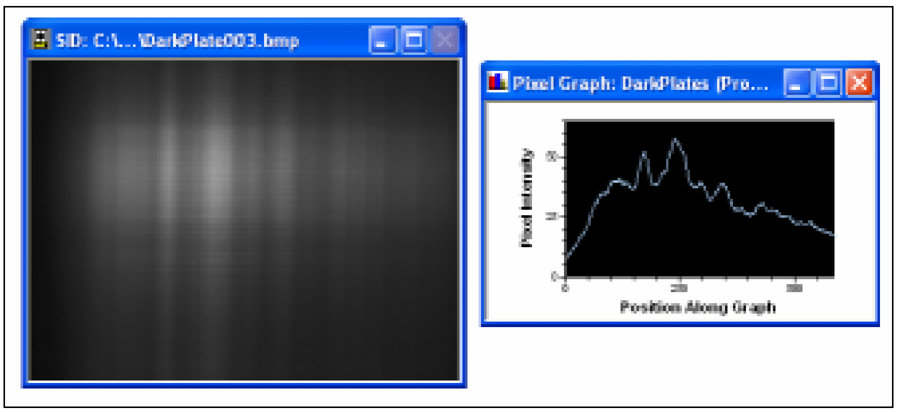


Figure 64: Image obtained with a SpectroCam and pixel graph of the Spectrograph SoftSensor.

The functionality of the SoftSensor is shown in Figure 65. The squares represent the pixels and the text inside them represents the intensity level.

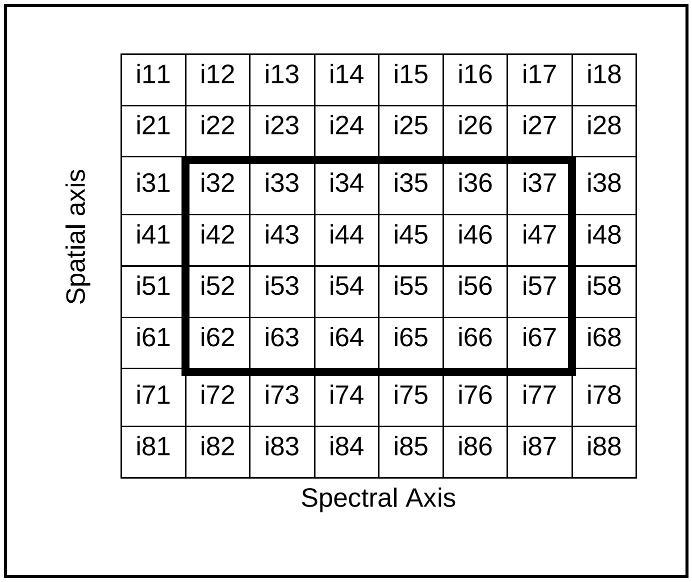


Figure 65: Pixel grid with intensity levels and Spectrograph SoftSensor interpreting the data.

Pixels are arranged in matrix style so we use matrix naming convention. The intensity level at the pixel with coordinates (1,1) is given by i11, the one for the pixel with coordinates (3,5) is given by i35, and so on. The SoftSensor is represented by the dark rectangle inside the pixel grid. In this case the SoftSensor extends from pixel (3,2) to pixel (6,7). From the data inside this rectangle the SoftSensor must compute a spectrum. In order to compute a spectrum the SoftSensor must use only one data point for every wavelength, and since the X axis is the spectral axis, it must use one single data point from every column of pixels inside the SoftSensor. There are five different ways to obtain one data point per column of pixels; the user must specify which one to use. The options are:

- Average: the SoftSensor averages the pixels in each column to obtain a single data point
- Peak: the SoftSensor creates a histogram for the column and reports the intensity level where that histogram peaks
- Minimum: the SoftSensor uses the minimum value found in the column
- Maximum: the SoftSensor uses the maximum value found in the column
- Median: the SoftSensor determines the median and uses that value to represent the column

The SpectroCam will come calibrated in nanometers and the user can recalibrate at any time. This means that peak reflection locations will be specified in this unit rather than just image coordinates. Also new in FrameWork 2.6, the user will be able to perform binning operations in both the spectral and spatial axes. Binning can be used in applications where the brightness has to be increased without increasing the exposure time. Since the SpectroCam will be calibrated, it will be possible to compute absolute color data for different color spaces. After a Dark Image Calibration is done, the user can select the option to subtract the black image from the spectrum being analyzed by the Spectrograph SoftSensor.

This SoftSensor lets the user set up a white reference. White reference refers to another Spectrograph SoftSensor looking at a white region. This white region would show what the spectrum of the light being used looks like and allow the SoftSensor to subtract that from the calculated spectrum. This dynamic operation (calculated for every image) will make the

SoftSensor concentrate only in the color content of the part being inspected. The functionality of this SoftSensor is similar to that of color SoftSensors. The user can learn different spectrums and compare the observed spectrum with the learned ones. In order to maximize accuracy in the comparison, the user can scale the spectrums to the peak values.

This SoftSensor can be set to pass or fail based on maximum spectrum difference, maximum change in location of peak wavelength, and maximum change in peak reflection using a learned spectrum for comparison.

Chapter 4 – Inspection Application Guidelines

This chapter provides major guidelines to approach different inspections. Users should be familiar with the different types of SoftSensors available in order to understand the concepts and issues explained here.

Dealing with Colors

The main uses of the Pixel Counting SoftSensor (Color SoftSensor) are to detect defects and to verify that correct parts are used in an application. Defect detection can be performed by learning the “good colors” of a part and detecting any color not in the “good colors” list in subsequent parts. If the type of defect is known, the “bad colors” can be learned instead and the images can be analyzed by checking for the presence of those colors only. Part presence can be detected by learning the colors of a number of parts and verifying that the correct part is being used by inspecting subsequent images and comparing the colors present to the desired color. Another way in which colors are used is to export them to other SoftSensors. When that is done and another SoftSensor references the color, areas of the imported color become light areas (referred to as above threshold for grayscale systems), and areas of other colors become dark areas (referred to as below threshold for grayscale systems). There are two ways to learn colors using the Pixel Counting SoftSensor: single color learning and multi color learning. The process for single color learning is illustrated in Figure 66.

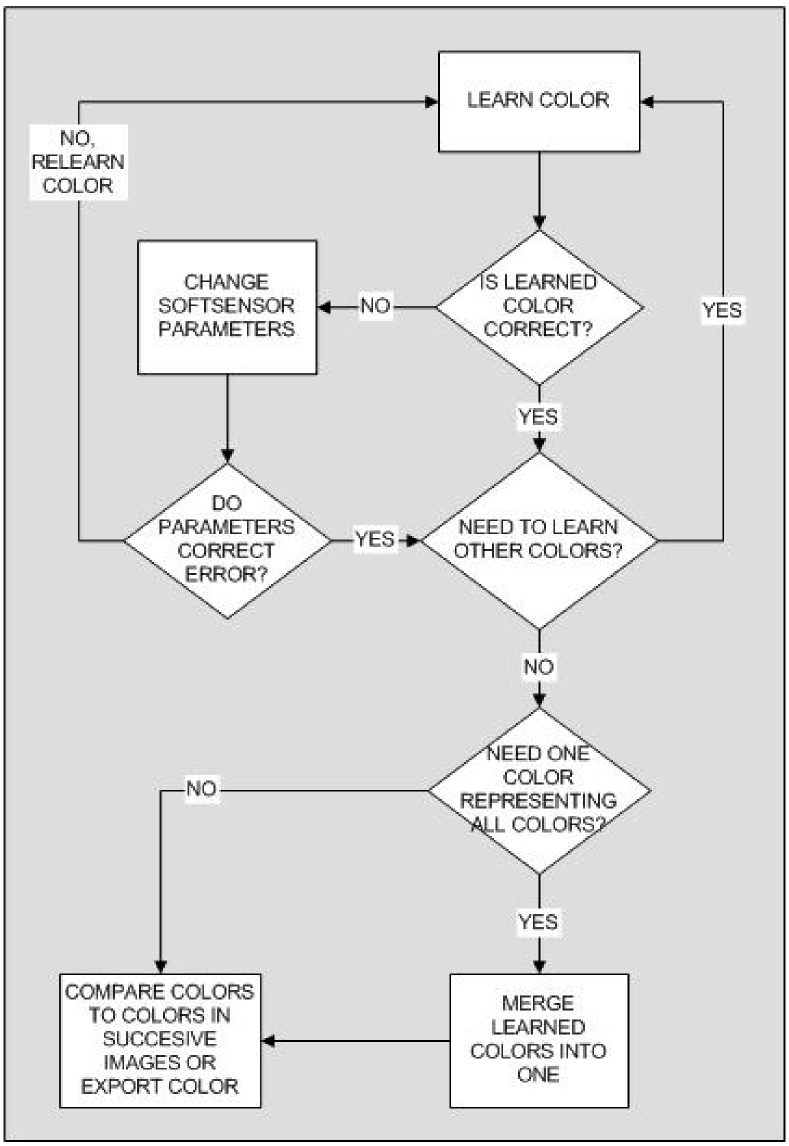


Figure 66: Process for single color learning.

As the process in the figure illustrates, single color learning does not mean the use of a single color but the learning of one color at a time. By following this process, the user will end up with a list of colors (or a single color representing the list) that describe a part or a set of parts and can be used for future inspections.

For this process, only one color can be seen by the SoftSensor when the colors are being learned. Any attempt to learn a color when two or more colors are seen by (are inside of) the SoftSensor will result in incorrect colors being learned.

The second use of the SoftSensor is the multi color learning. This process involves learning multiple colors at the same time. The process is illustrated in Figure 67.

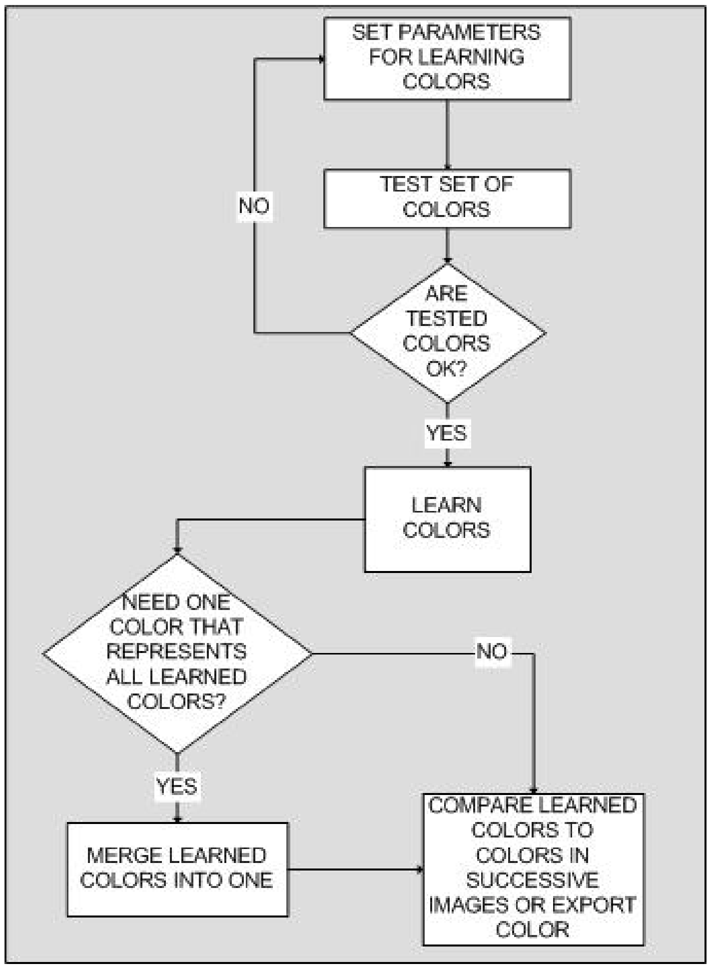


Figure 67: Process for multi color learning.

This process is used to learn the colors of a complex part- one that is not of a well-defined color. The SoftSensor is drawn over the entire area (so all the colors are inside the SoftSensor) and the process begins by setting the parameters. The parameters include the minimum number of pixels that each color must have to be included in the list (Cluster Size) and the maximum number of colors to learn from the part (Maximum Cluster Count). Based on those parameters the image is tested to get image marking showing which colors are learned. Only after the desired colors are marked should the user learn the colors. Since this method learns multiple colors at the same time, the list of learned colors replaces any colors previously learned. For more information about color parameters see Color SoftSensors.

Dealing with Part Movement

In real world applications, parts to be inspected are not consistently placed in the same position for every inspection. There are many parameters that affect this such as the use of proximity switches to indicate presence/absence of the part. Those switches will only tell whether the part is present or not. For more precise information about the location of the part, proximity switches are not very reliable. FrameWork offers a number of tools to help the user locate the part in the image and to create a reference that the inspection SoftSensors can use to track the movement of the part. This set of tools consists of the Translation, Rotation, ObjectFind, Blob, Measurement, Math, Segmentation, and even OCR SoftSensors. These SoftSensors are used to calculate part movement. The user needs to create a reference from the inspection SoftSensors to the positioning SoftSensors so every SoftSensor inspects the correct area of the image every time. Figure 68 illustrates how to track the part. The positioning SoftSensor locates the part and calculates the shift in position (with respect to the original position).

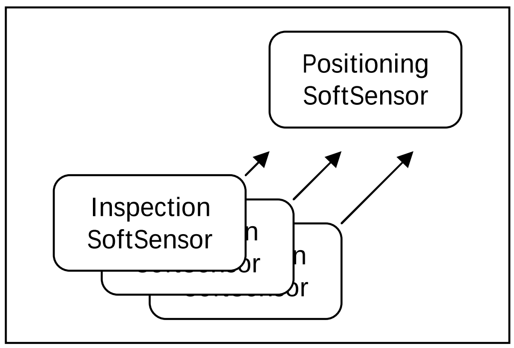


Figure 68: position reference mechanism.

The inspection SoftSensors then reference the positioning SoftSensor to find out where the inspection is to take place. If the application allows for the use of the most powerful positioning SoftSensors (Blob and ObjectFind SoftSensors) just one SoftSensor can produce all the necessary position references. However, some of the fastest positioning SoftSensors can only track one type of movement. Unfortunately, in most cases a part translates in both horizontal and vertical directions and rotates. When that happens and the SoftSensors used do not provide complete position feedback, the user needs to create a chain of references. SoftSensors may need to find the changes in the location of the part in both X and Y directions and then determine the rotation of it. In those cases, the chain of references is extended to multiple levels. Figure 69 shows consecutive images taken from an actual inspection. As the images show, the part moved. If the SoftSensors are to inspect it, positioning SoftSensors need to find it in the image. In this case, the part shifted up, left, and rotated counterclockwise.

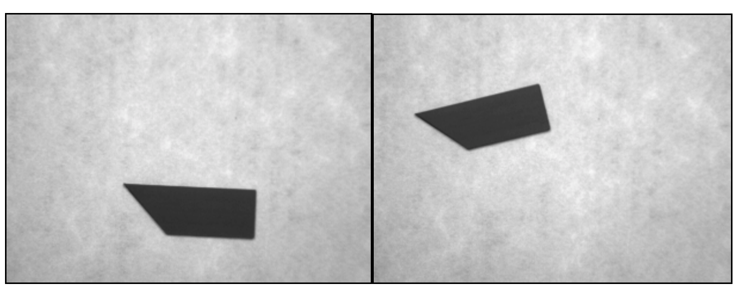


Figure 69: Consecutive images of parts to be inspected. The image on the left shows a part in the original position whereas the one on the right shows why positioning SoftSensors are needed.

The procedure in this case would be to find the vertical displacement (the one that changed the most) first. Then the horizontal displacement, and only then determine the rotation. The diagram in Figure 70 illustrates the process.

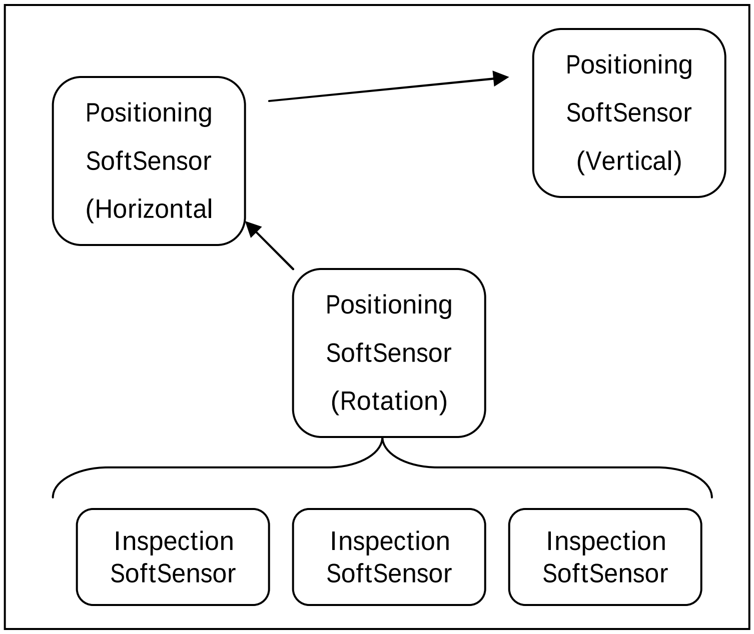


Figure 70: Method for passing references.

The SoftSensor detecting vertical movement finds the change in position in the "Y" direction. Then, the SoftSensor tracking changes in the horizontal position uses that information to look for the part in the upper region of the image. Once it finds the part it automatically records the change it detected plus the change obtained from the other SoftSensor and passes both as a position reference to the SoftSensor detecting rotation. This SoftSensor shifts up and to the left to analyze the rotation in the right location and adds it to the chain of position references. When the inspection SoftSensors request information from this one, they obtain the changes in position generated by all three positioning SoftSensors. This method simplifies the setup process because position references are automatically concatenated and the inspection SoftSensors need only reference one of the positioning SoftSensors.

As mentioned before, just one Blob or ObjectFind SoftSensors could have been used, but they would have consumed more processing time. Blob and ObjectFind SoftSensors are area tools, that is, they scan an entire area looking for a shape. This process takes more processing time than a line scan like the one performed by line Translation SoftSensors.

As mentioned before, OCR SoftSensors can be used for positioning. This is a very powerful method of doing positioning because the SoftSensor limits its search to trained characters only, making it less sensitive to noise.

Note: If any of the SoftSensors in a chain of reference is altered (changes size, position, threshold, etc.) all the SoftSensors that come after it in the chain of references will be altered. This will require that the reference be reset by placing the SoftSensor in the desired location and disabling and re-enabling the reference to the SoftSensor that changed.

Presence/Absence Inspections

Presence/Absence inspections involve inspecting a part for specific features to make sure the feature is present or absent. The locations of the features of interest in a Presence/Absence inspection are generally known. Many of the SoftSensors within FrameWork can be used to inspect for the presence or absence of features of an object. Intensity, EdgeCount, FeatureCount, Blob, ObjectFind, Template Match, and Color (Pixel Counting) SoftSensors can all be used to detect for the presence or absence of a part feature. Each one of these SoftSensors has different advantages and disadvantages. Intensity, FeatureCount, and EdgeCount SoftSensors are the ones that take the least processing time, but they are more sensitive to noise. Blob, ObjectFind, and Template Match SoftSensors take more processing time but they offer a number of features to filter out noise. In every case, image contrast plays a very important role. The objective of the physical setup of the application should be to maximize contrast between features to be inspected and the background. Figure 71 shows an example where the use of a different lens, correction of working distance, and use of appropriate light source increases the contrast between the part and the background.

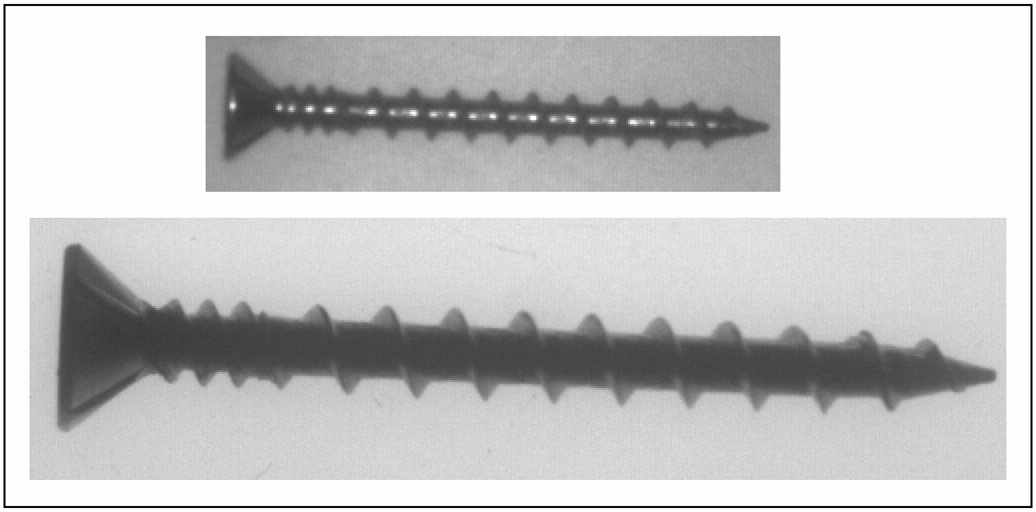


Figure 71: Importance of physical setup of the application to maximize contrast between part and background.

The image on the bottom is a better image, due to the higher contrast between part and background, the absence of reflections (no pixels are saturated), and sharper.

Flaw Detection Inspections

Flaw Detection inspections typically involve looking for a general flaw on the surface of a part where the location of the flaw is unknown. SoftSensors that can be used for this type of inspection are Intensity, Blob, Template, Measurement, Segmentation, Color Pixel Counting or FeatureCount SoftSensors. The choice of SoftSensor depends on the application. Template Match SoftSensors can be used if the part contains a certain pattern that can be learned and searched for in subsequent images. This SoftSensor takes a lot of processing time and system memory, since the SoftSensor needs to keep the learned model in memory. Segmentation SoftSensors compute gradients instead of using a threshold, so when the flaw cannot be isolated using a threshold, this tool should be used. It also takes a lot of processing time. When the flaw can be isolated using a threshold level, the other tools can be used to find it. Intensity and FeatureCount SoftSensors are the ones that take the least processing time.

Counting Inspections

Counting applications require parts or features of a part to be counted. Several SoftSensors can be used for counting applications including EdgeCount, FeatureCount, Blob, Segmentation and ObjectFind SoftSensors. EdgeCount and FeatureCount should be used to identify features along a line, arc, or another predefined shape. These SoftSensors scan along that line counting the edges or features found. They take little processing time, but they require that the pattern in which the features are arranged be known. Blob SoftSensors should be used when counting parts that change the position and that do not touch each other. The way that Blob SoftSensors work will count two parts as one if they touch. If the parts are separated, this type of SoftSensor offers a very reliable solution at a relatively high speed. When parts can be touching or even overlapping, the ObjectFind SoftSensor should be used since it learns the shape of the part and will find it even if the entire shape is not visible. In this case the SoftSensor will report that there is a part when a certain number of features learned from the part are found in the image. The user can select the acceptance level to discard parts that are barely visible or severely damaged.

Measurement Inspections

Measurement applications involve measuring diameters, widths, height, and other key measurements of an object. Usually, a combination of Math and Measurement SoftSensors solves the problems. When more operations are needed, scripts can be used. For instance, scripts offer the flexibility needed to maintain running averages between inspections.

Techniques for Better SubPixel Measurements

Lighting

Backlighting should be used for measurement applications whenever possible. This technique consists of placing the part over a plane of light to highlight its contour. The image produced by such arrangement simplifies the job for measurement tools, and makes them more precise too. When backlight is used on objects with curved surfaces, the light should be masked or collimated to obtain better results. Figure 72 shows examples of the use of masks and collimators. The image on the left uses standard backlight. Since the object being inspected has curved sides, the light produced at the far ends of the light source bounces on the upper regions of the part and is directed towards the SmartImage Sensor. This causes the SmartImage Sensor to see fuzzy edges making the measurements less reliable. When this happens, edges tend to be defined by transitions of as many as 10-20 pixels. The second example shows the effect of masking the backlight. Masking simply means covering the areas not used, or the areas farther away from the edges of the part. This will make the edges of the part appear sharper (transitions of 3-4 pixels) thus making the measurements more precise. When there is part movement and the light cannot be masked, a collimated light should be used; this is shown in the third image. The collimator filters the light so that only rays perpendicular to the surface of the light source get to the SmartImage Sensor causing a similar effect as masking the light.

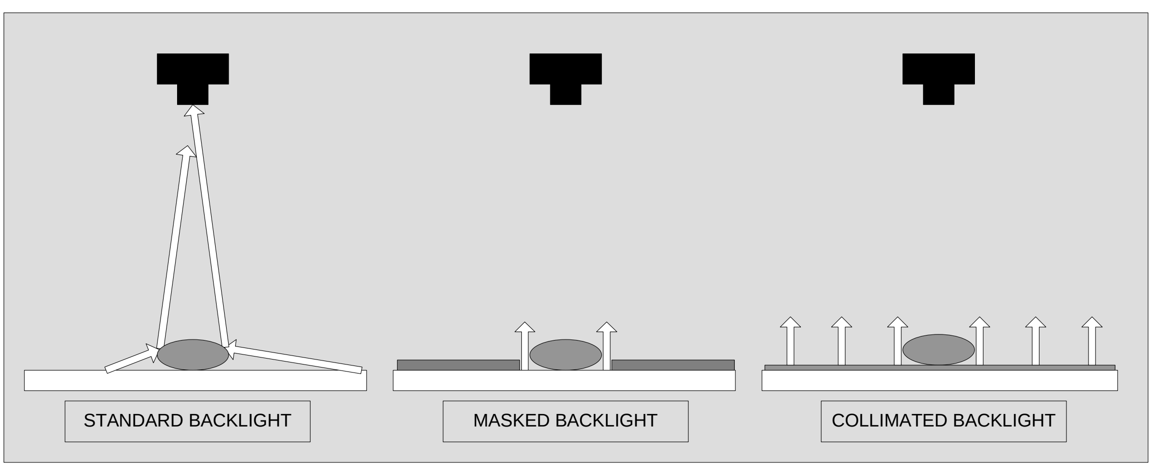


Figure 72: Examples of the use of masks or collimators to enhance part edges when using backlight.

Another important consideration for obtaining precise measurements is to avoid altering the intensity levels in the image. There should be no saturated pixels (pixels reporting 100% intensity level) and AntiBlooming should not be used for it adds noise to the image.

Lensing

The type of lens used for measurement applications should minimize lens (or barrel) distortion. The best choice is to use telecentric lenses. These lenses minimize distortion and perspective error. Using a telecentric lens the part size and shape should not vary if it moves within the FOV or even farther and closer to the lens. This type of lens can get very expensive as the part size increases. An inexpensive alternative consists of zero distortion lenses. Most lens manufacturers carry a number of this type of lenses.

Positioning Robotics Applications

Applications requiring the position of an object to be obtained involve SoftSensors that can generate the location of a part feature and conversion from Image coordinates to Robot/Real World coordinates. SoftSensors used for these applications are generally Measurement/Math SoftSensors with a Coordinate Transformation SoftSensor for converting to real world coordinates. ObjectFind and Blob SoftSensors can also be used if general position location is desired. These two SoftSensors require more processing time because they scan entire areas searching for objects. The Coordinate Transformation can be used both to change pixel coordinates to robot coordinates and to correct for lens distortion (both require previous calibration). The second correction can be minimized by using a better type of lens.

Setting up a Coordinate Transformation SoftSensor

The steps to create and calibrate a Coordinate Transformation are explained below:

- Select specific points (minimum 5 for simple coordinate transformation recommended over 10 points if lens distortion is also being corrected) for which the robot coordinates are known. The user should make sure that these points vary in distance from the center of the image, and are not all in a line.
- Create a SoftSensor that generates a point as its output (usually the FindCircle SoftSensor is used) for each one of those points to get the pixel coordinates of them.
- Create a Coordinate System SoftSensor (under Math Tools) and use the SoftSensors created in the previous step as the calibration points. To add the calibration point one by one the user only needs to specify the robot coordinates of each point. By clicking the button to update the coordinates, the pixel coordinates will be automatically entered. This process will create the calibration data necessary for the transformation of any point in the image to robot coordinates. Starting in FrameWork 2.6, the user will be able to create a global calibration that can be used by other Coordinate System SoftSensors across Products in the SmartImage Sensor.
- Create a SoftSensor to track the part being inspected. Usually a Blob or ObjectFind SoftSensor is used.
- Create a New Coordinate Transformation SoftSensor to transform the SoftSensor tracking the part based on calculations performed by the Coordinate System previously calibrated. This SoftSensor should output the position of the part using robot coordinates.

Color Sorting Applications

Color Sorting applications generally involve monitoring the color(s) of an object to determine which object it is. The best match / worst match features in the Pixel Counting or Color Monitoring SoftSensors must be used. For colors which are very close, the Legend SpectroCam SmartImage Sensor should be used in combination with the Spectrograph SoftSensor. Basically, for gross differences the Pixel Counting SoftSensor should be used. For parts that offer very little color difference and when the Pixel Counting SoftSensor does not offer the reliability needed, a Color Monitoring SoftSensor should be used. This SoftSensor requires that the area to be analyzed be of a uniform color, no color variations are allowed. In cases where the color difference is visually hard to detect, the SpectroCam should be used to compare the entire spectrum of the parts and make a more reliable decision. The use of the SpectroCam requires that the part to be analyzed be consistently placed because the positioning SoftSensors cannot not used by the SpectroCam.

Part Identification

Identifying a part is a very broad type of application that generally entails defining differences between the parts and identifying them using SoftSensors. There are several different techniques for achieving part identification and they depend mostly on the part being inspected. They can be implemented using one or many different SoftSensors to detect the differences between parts. Some SoftSensors that can be used by themselves to identify parts are Blob, ObjectFind, Color (Pixel Counting), Reader, Segmentation, and Template Match SoftSensors. Blob SoftSensors should be used when the parts can be identified based on general geometric features (Area, Radius, Eccentricity, Compactness, etc.). When a more precise identification needs to be done, ObjectFind should be used at the expense of some extra processing time. Segmentation and Template Match SoftSensors should be used when the features of interest cannot be separated from the background based on a threshold level. The extra precision comes with extra processing time. Pixel Counting SoftSensors should be used when the difference between parts is based on color. Finally, Reader SoftSensors should be used when the difference is in a barcode, DataMatrix, or a label with characters. In those cases, the user needs a SoftSensor to read the data from the label or part in order to determine which part it is.

Lot Code and Data Code Applications

Lot Code and Data Code Reading can be accomplished with any one of the Reader SoftSensors. The OCR (Optical Character Recognition) SoftSensor is used to read human readable characters. The 2-D and 1-D Readers are used to read information symbols printed on the product.

OCR

The OCR SoftSensor can be trained to read any character (not just letters or numbers). Usually this type of application requires positioning. That is, the label is not located always in the same location. This problem requires more SoftSensors to provide a position reference. As mentioned before, the OCR SoftSensor itself can be used for position reference. The user simply needs to create an OCR SoftSensor to locate the first character and another SoftSensor to read the code. This option of finding the first character makes the positioning very independent of noise, since the SoftSensor will be looking for a specific learned character, not an edge or a feature.

1D Reader

The 1D Reader SoftSensor consists of two line-based tools (arc and straight) and an area based SoftSensor. The linebased versions of the SoftSensor scan along a line trying to read a certain bar code. The user has a few options to speed up the process or to make it very thorough. When the application consists of the same type of code every time, the user can set the SoftSensor to look for that specific code. This will reduce the processing time because the SoftSensor does not need to attempt to decode the data using other methods. When the application determines that different types of barcode are to be used, the user must select the option to remain in auto detect. In this case the SoftSensor not only reads the code but it also determines what type of code it is. This option requires more processing time.

The parallelogram version of the 1D Reader SoftSensor will help in cases where the position of the code changes from image to image, handling translation and some amount of rotation.

2D Reader

There are two types of DataMatrix readers: fixed and searching. The fixed SoftSensor takes less processing time, but it requires that the code be consistently located in the same area every time. A minor change in position will cause it to fail. In some cases, this SoftSensor can be linked to a position reference so the SoftSensor inspects the area where the data is. In cases where images present too much noise, this option should be avoided because this tool requires very precise positioning. The other type of SoftSensor available for DataMatrix searches for the code in a predefined area. Even if the code rotates, the SoftSensor will find it and read it. This option takes more processing time because the SoftSensor needs to search for the code in what usually is a noisy image (typically the code is surrounded by text). In order to speed up the process, the user should provide the SoftSensor with as many parameters as possible about the type of code to search for, so the searching process is faster.

Starting in FrameWork, two versions of SnowFlake readers are provided: a rectangular shape and an elliptical shape reader. They are both searching readers.

Chapter 5 – SmartImage Sensor Integration

This chapter discusses the different ways in which the user can send data to the SmartImage Sensor or get data out of it. The most common method is with the digital I/O lines, which were described under System Parameters. This chapter explains how to implement Modbus transfers, how to set up DataLink, how to set up the built-in drivers, and how to enable/disable different operational modes. It also mentions the types of communications that SmartImage Sensors support.

Transferring Data from a SmartImage Sensor

The most important factor to consider before setting up a data transfer is the timing. Information about an inspection needs to arrive in a certain way for the receiving system to interpret it correctly. There are two ways to transfer data: synchronous and asynchronous.

Synchronous transfers of data happen after every inspection. The user sets up the data to be transferred and that data is sent out after every inspection. This ensures the receiving system that new data is transferred as soon as it becomes available. The methods of communication that support synchronous transfers are the digital I/O (discussed under system parameters), and DataLink.

Asynchronous transfers of data happen at a certain rate. In most cases, the user selects the rate and blocks of data are continuously transferred whether they contain new data or the same data that was already transferred. An example of this method of data transfer is the Modbus master transfer available from FrameWork.

It should be mentioned that because of their flexibility, scripts allow for both types of data transfers, but in this case the user needs to implement the transfers. For more information about scripts see the script documentation and help files.

DataLink

DataLink is a built-in tool used to send data out of the system and even receive a limited number of commands from another device. This tool is Product-specific, that is, every product has its own DataLink that can be configured depending on the inspection and the SoftSensors being used. DataLink consists of a number of ASCII strings that are created based on information from the SoftSensors. The user selects a set of rules under which those strings are to be sent out. Those rules are created using logical operators (and, or) acting on SoftSensor results. For example, if the positioning SoftSensor fails a certain string can be sent out, but if this one passes DataLink could look at the result from another SoftSensor to determine which string (if any) to send out. Figure 73 shows a screen capture of DataLink (a dialog box within FrameWork). In this case the user selects three different messages. The first one ("String 1") will be sent out when the SoftSensors that measure the inner and the outer radius of a part indicate a successful result (PASS) and the SoftSensor reading a lot code fails.

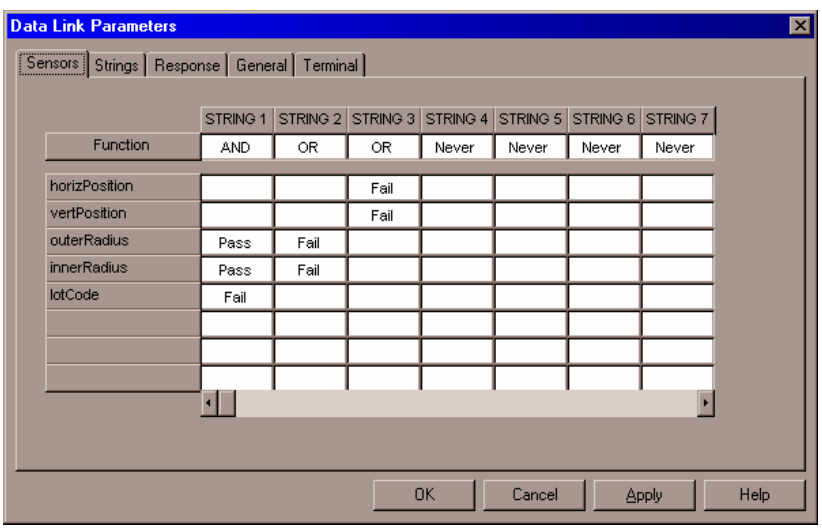


Figure 73: Screen capture of DataLink. In this case three different messages are requested from the system based on the inspection results.

The second one (“String 2”) will be sent out when any of the SoftSensors measuring the radii fail. Notice the “OR” operator at the top of the column. The last one (“String 3”) will be sent out when either or both positioning SoftSensors fail, which could indicate that the part has moved too far away from the original position or that it is not present in the image.

Once the conditions are indicated, the strings need to be created. This is a simple point-and-click process that gives the user access to data from every SoftSensor in the Product plus the possibility of typing some extra characters. Figure 74 shows a new screen capture of DataLink with the correct setup for the message “String 1”. This message was supposed to be sent only when the lot code reader indicates a failure (it failed to read anything); therefore, the message should indicate the code being read. The string was setup using text (“The lot code read was:”) then a reference to the SoftSensor that contains the information to be sent out (in this case lotCode is the SoftSensor and its string is what contains the data, so “lotCode.String” will reference the desired data).

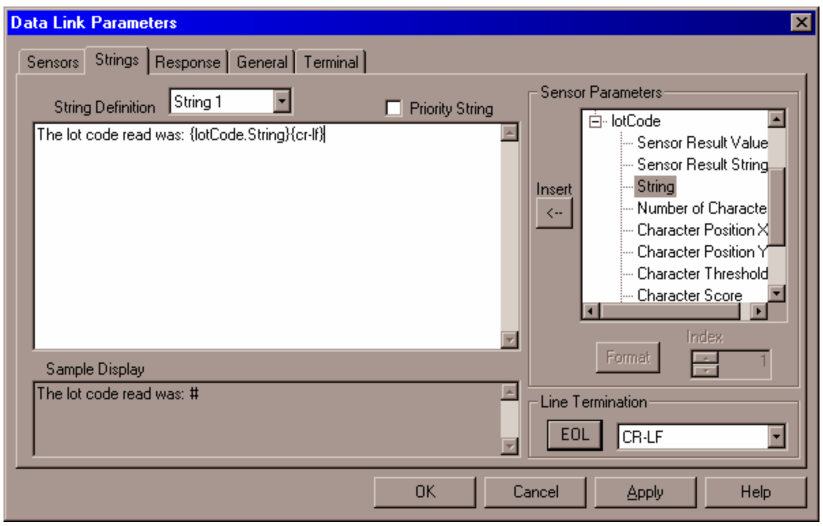


Figure 74: String setup for DataLink.

The last section of the string is the “end of line” character, which is available from the menu shown in the lower right corner. In this case {cr-lf} was selected which gives a carriage return followed by a line feed.

Depending on the output from every SoftSensor, one of the selected messages (Strings) will be sent out or, in some cases, no string will be sent out (ex: if all the SoftSensors pass). This will give the user the tools to determine where the flaw is in the process and allow him to take immediate action. The fact that every string is fully configurable gives the user the flexibility needed to implement an effective inspection. In order to read the data sent out from DataLink, the user needs to establish an Ethernet connection using TCP protocol to port 3247 (port 5001 for older systems).

Starting in FrameWork 2.6, DataLink output can be sent in XML format. XML is becoming the standard for specifying the actual meaning of data. In this case the names of the tags and overall format are hardcoded. Additional meta-data is provided beyond what DataLink’s regular format provides so as to make parsing easier. For example, the parameter name and format (int, float, etc.) is given, as well as the associated Product name. To enable this new output format the user should select the “Enable XML Output” option from the “General” tab.

Modbus Transfers

A very efficient way to share data among SmartImage Sensors or between a SmartImage Sensor and an external device is to perform a Modbus transfer. Modbus transfers, as implemented in the

FrameWork user interface, take a number of registers from one system and copy those registers into another system. Using this simple procedure, one SmartImage Sensor can make a decision based on the result from other SmartImage Sensors or gather data from other SmartImage Sensors and send it out to an external device. In a Modbus network we have master devices and slave devices. SmartImage Sensors can be enabled to be a Modbus slave, which will make the system expect a connection on port 502 (Modbus standard). The Modbus Master initiates the connection and decides the type of operation to perform (write data to the slave or read data from it). Figure 75 shows two possible configurations for sharing data using Modbus transfers. The first configuration uses Modbus transfers to exchange data between SmartImage Sensors. There are four SmartImage Sensors in the DVT network: three of them are being used as Modbus slaves and the fourth one acts as the master. This SmartImage Sensor queries the slave SmartImage Sensors for data and, based on the data from all three slaves and its own data, draws a conclusion. The final output of the inspection is made available to an external device using either Modbus or some other type of communication. It could even be a single I/O line indicating a PASS or FAIL result. The second configuration shows a slightly different setup. In this case, an external device is acting as a master and the SmartImage Sensors performing inspections are acting as slaves. The external device has direct access to the internal memory of the SmartImage Sensors so it can easily query the individual inspection results from each.

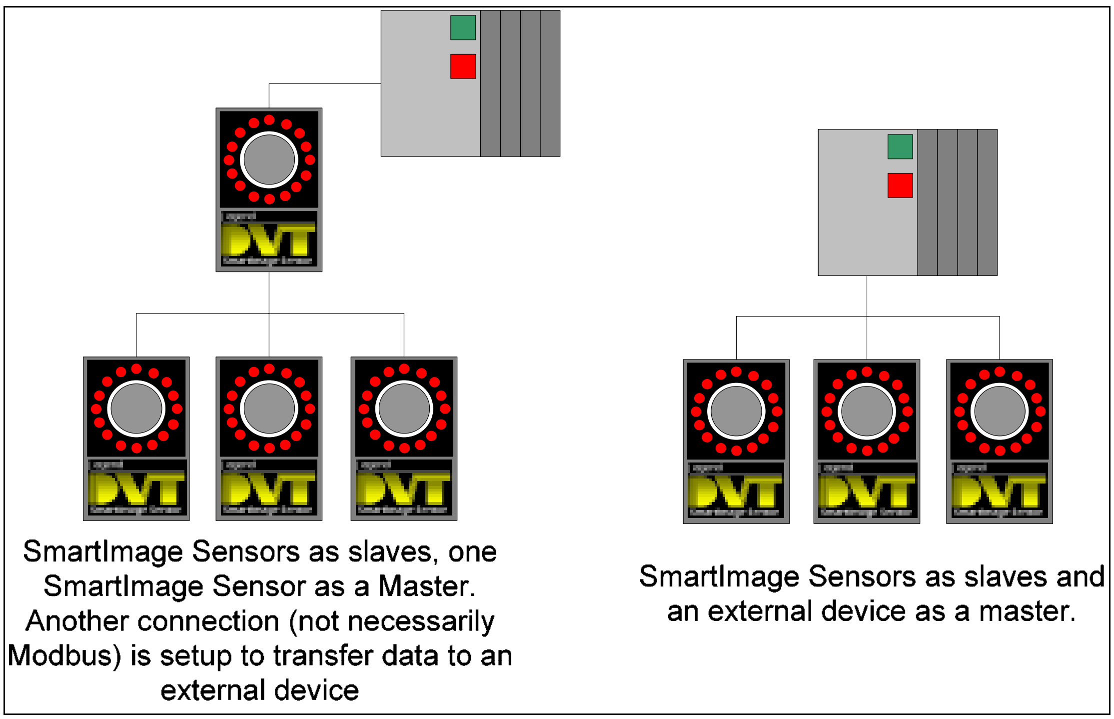


Figure 75: Two different configurations to share data among multiple devices.

In order to make a SmartImage Sensor behave as a master, a Modbus master transfer must be created from FrameWork. This transfer will indicate the slave IP Address, the operation to be performed (read/write), the block of registers to use, and the polling rate for the transfer. In order to specify the block of registers to transfer, the user must be aware of the difference between DVT registers and Modbus registers as well as the size of the different data types used by the SmartImage Sensor.

DVT registers are 8-bit registers whereas Modbus registers are 16-bit registers. This difference poses a problem that the user must solve in order to implement a successful transfer. Since the rules for the transfer must be expressed in Modbus registers, the user needs to convert the register numbers to Modbus register numbers. Fortunately, this happens to be a very simple procedure which will be illustrated by an example. Let's say the user needs to transfer the data in registers 112 to 117 of the slave to registers 30 to 35 in the master. It is a total of 6 DVT registers which means 3 Modbus registers. Instead of starting at DVT register 112 of the slave system, the user has to specify the starting register as 56. Also, the master register to write to is DVT register 30, so the user needs to specify Modbus register 15. In order to simplify the process, FrameWork provides feedback of the register selection to ensure that the correct registers were selected. Figure 76 shows the actual FrameWork dialog box used to set the register numbers. The user selects the Modbus registers using the editable text box on the left.

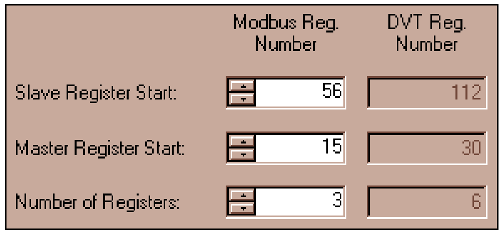


Figure 76: Sample register setup for a Modbus transfer.

Once the numbers are set, FrameWork shows the equivalent DVT registers in the non-editable text field on the right. The particular setup used in the image refers to the previous example.

The second important issue to take into account is the size of the different DVT data types. This becomes a very important issue when trying to transfer data. Different data types occupy different numbers of registers in memory. If a certain value is saved to memory using a script tool and that particular piece of data needs to be transferred to another system, the user has to know the exact number of registers that the piece of data occupies in memory. Table 2 shows the size that every different data type occupies in memory. For example, if a measurement tool outputs a float, that number will occupy 32 bits or 4 DVT registers. If that number needs to be transferred to another system using a Modbus master transfer, the user has to specify that 4 DVT registers need to be transferred. According to the size of the Modbus registers (explained above) this requires the transfer of 2 Modbus registers.

Name	Size (bits)	Size (DVT registers)
byte	8	1
short	16	2
integer	32	4
long	64	8
float	32	4
double	64	8
String	8 per char + 1	1 per char + 1

Table 4: Size of the different data types in bits and DVT registers.

For more information about Modbus protocol and Modbus transfers see the DVT integration notes section of the DVT website.

If the transfer of data needs a more explicit process, Modbus transfers might not be the best approach. In those cases a background script should be used. From a background script, the user can create a Modbus transfer object which can perform a number of standard Modbus functions as specified in the open Modbus protocol. For more information about this see the script documentation.

Ethernet Terminal Controller (ETC)

The Ethernet Terminal Controller (ETC) allows the user to set up different Ethernet configurations for communication using the TCP protocol. This converts the SmartImage Sensor into a TCP server or client. Depending on the application, the user can select the SmartImage Sensor to be a client (connect to another device) or a Server (expect a connection from other devices). In both cases, the connection will be established using a user-defined port. Using this connection the following drivers can be used to communicate with different devices:

- System Driver: Redirects the data exchange at the system level that normally happens on port 3246 (port 5000 for older systems).
- DataLink Driver: Redirects the data exchange from DataLink that normally happens on port 3247 (port 5001 for older systems).
- Robotic Drivers including Fanuc, Motoman, ABB, Reis, and Kawasaki Robots.

The user can set the system up as either a TCP/IP server or client and specify the preferred port. The user can also set the Ethernet port for communication as well as the type of communications driver to use on the port from this selection. Figure 77 shows two different configurations for the ETC. The first one makes the SmartImage Sensor a TCP server, expecting a connection on port 5003. In this port it exposes the System Driver, which is the driver that receives most Ethernet commands to control the system. The second configuration is as a TCP client. In this case the system connects to a server with IP address 192.168.0.242 on port 5004 and sends DataLink strings to that server. This is indicated by the selection of the DataLink Driver. This example demonstrates the two main types of configuration (server and client).

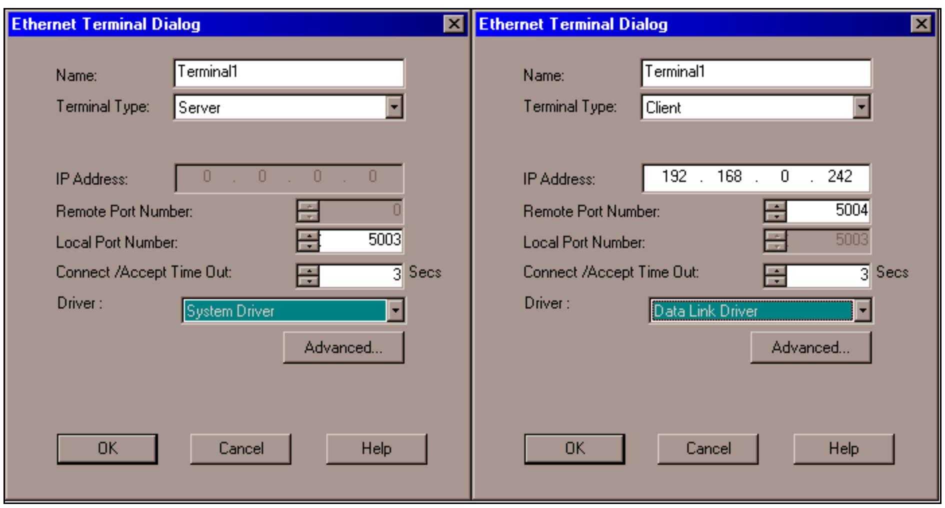


Figure 77: Different configurations for FrameWork's ETC.

After this is properly setup the communications should be established when the Ethernet Terminal Controller is started or on power-up.

Industrial Protocols

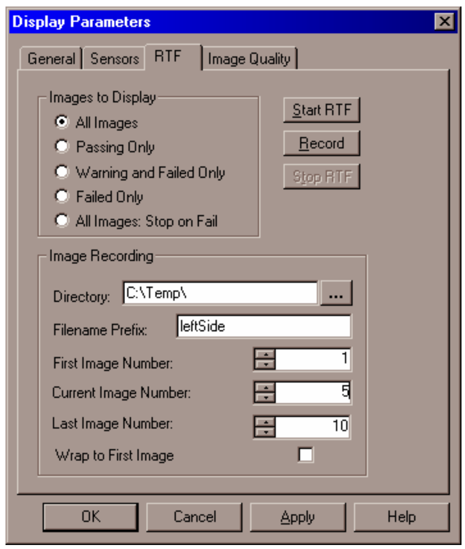
DVT systems support other methods for communications such as standard protocols EtherNet/IP, ProfiBus and DeviceNet. EtherNet/IP support must be enabled from the I/O menu in FrameWork and the power must be cycled for the changes to take effect. It is disabled by default to avoid loading the processor with operations not used. Using scripts, the user can interchange data with other devices that support the protocol. For more information about this type of connectivity please refer to the Script documentation and DVT integration notes. SmartLink, a different DVT product that can be connected to a DVT SmartImage Sensor can communicate via DeviceNet or ProfiBus using expansion cards designed for that purpose. For more information about SmartLink, please refer to the SmartLink documentation.

Appendix A – Emulator tutorial

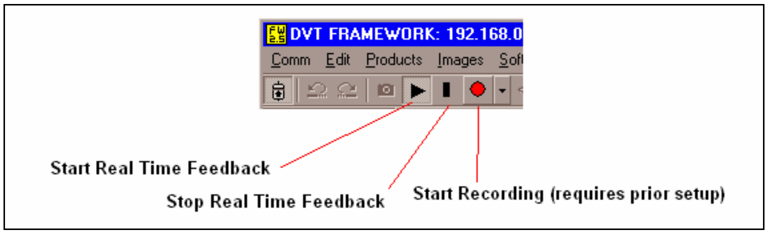
Chapter 2 – A Closer Look at the System explains the basics about the use of the Emulator. This appendix walks the user through a full example on how to work with it. Every step is explained and the FrameWork dialog boxes are included for reference. If the user already has a set of prerecorded images and a product or system file the first 2 steps should be skipped.

Step 1: Connect FrameWork to the SmartImage Sensor and create a backup copy of the System or Product files. In order to do this, the user must select the desired option ("Backup Product to PC" or "Backup System to PC") from the "Products" menu. This will prompt the user for a file name just like in any Windows® application. A file will be saved to the computer hard drive with the name specified by the user and an extension indicating if it is a system file (extension "sysdvt") or a product file (extension "dvt").

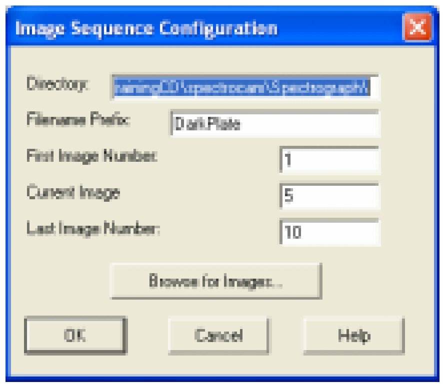
Step 2: Record images from the SmartImage Sensor. In order to do this the user must select "Display Parameters" under the "Images" menu and select the "RTF" tab (RTF stands for Real Time Feedback). The dialog box is shown below. This dialog box lets the user select the directory where the images are to be saved, the name for the images and the number of images to record.



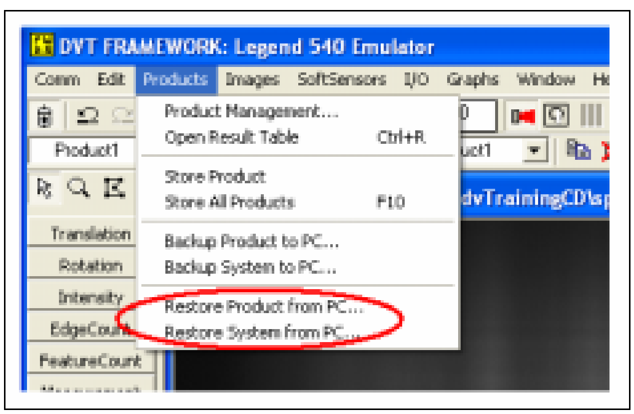
The user must specify the number of the first image in the sequence, the number of the current image (the next image to be recorded) and the last image to record. The last option "Wrap to First Image" indicates whether to stop on the last image or start overwriting from the first one. This will create a number of bitmap images in the selected directory with the name and 3 digits indicating the image number. The setup shown in the dialog box above would save the following images: leftSide005.bmp, leftSide006.bmp, leftSide007.bmp, leftSide008.bmp, leftSide009.bmp, and leftSide010.bmp in the "C:\Temp" directory. This procedure saves every image that FrameWork brings from the SmartImage Sensor. In the top portion of the dialog box, the user can select which images to bring to the user interface. If only the failed images are to be saved, for example, the user should select to display only those. Once this setup is completed, the user should exit it using the "OK" button, which will cause the changes to take place. The Real Time Feedback should be started as usual from FrameWork. When the user wants to start recording, the "Record" button should be pressed from the expanded toolbar as shown below. If this toolbar is not available it should be selected from the "Window" menu.



Step 3: Once the Product or System file and the images are available from the hard drive, the user can start FrameWork and connect to an Emulator. The Emulator should be selected to match the SmartImage Sensor from which the images or Product/System files originated. Upon connection to the Emulator, FrameWork will show the SID and make the standard editing options available. At this point the user can load the image sequence and the Product or System file. In order to load the image sequence, the user needs to select the option "Configure Image Sequence" under "Images". The following dialog box will appear:



Here the user has two options: type everything manually or click on the "Browse for Images…" button to find the desired images. In order to load the Product or System file, the appropriate action should be selected from the main "Products" menu. To load a product file the user needs to select "Restore Product from PC." Likewise, for a System file, the user should select "Restore System from PC." The menu is shown below and both options are circled.



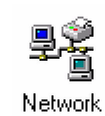
With the images loaded, the user can make any change (that does not depend on physical setup) to the system just like if FrameWork was connected to a SmartImage Sensor. When the user is done, step 1 should be repeated but in this case for the Emulator. A system or product file can be saved from the Emulator in the same way that it can be saved from the camera. This procedure allows the user to set up the inspection parameters in the Emulator, and then load those as a System or Product file into the SmartImage Sensor.

Note: If users want to run the Emulator but they have no images, a standard graphics program could be used to create some images. The images need to be 256-color bitmaps of the same resolution as the SmartImage Sensor to be used (640 by 480 or 1280 by 1024). The naming convention should be followed as well.

Appendix B - Basic TCP/IP Setup

In this section, we will step through setting up the TCP/IP protocol on a PC. TCP/IP stands for Transmission Control Protocol / Internet Protocol and is based on identifying elements on a network by unique numbers, called IP numbers.

We will assign our PC with one IP number and any SmartImage Sensor with another. There are several methods of connecting PCs and SmartImage Sensor systems.



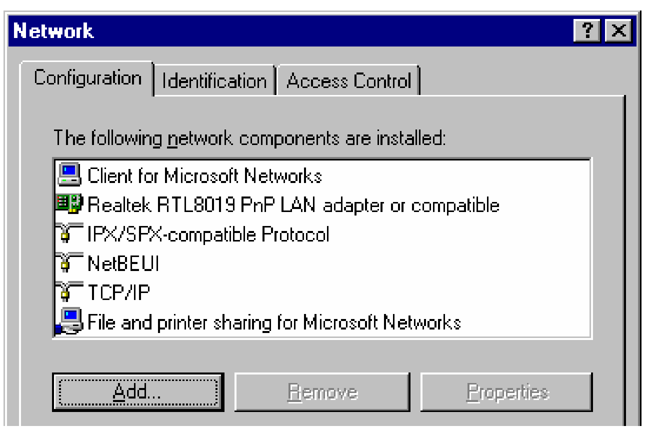
For Windows 98®:

Click on the **Start** menu in Windows®, select **Settings**, then **Control Panel**.

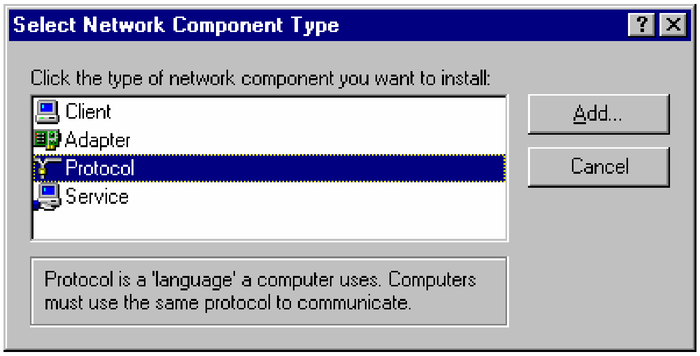
Double click on the **Network** Control Panel icon (shown above right).

Check the list of installed network components for TCP/IP (as shown below). If TCP/IP is on the list, click **Cancel** and skip to the Communicating with the SmartImage Sensor section.

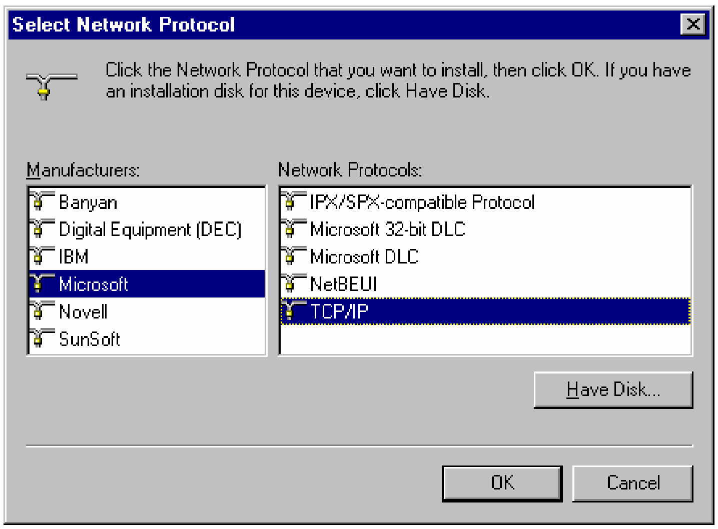
If TCP/IP is not on the list, click the **Add** button.



Click on **Protocol**, then **Add...** (shown below).



Under Manufacturer, click Microsoft and under Protocol, click TCP/IP then click **Ok** (see below).



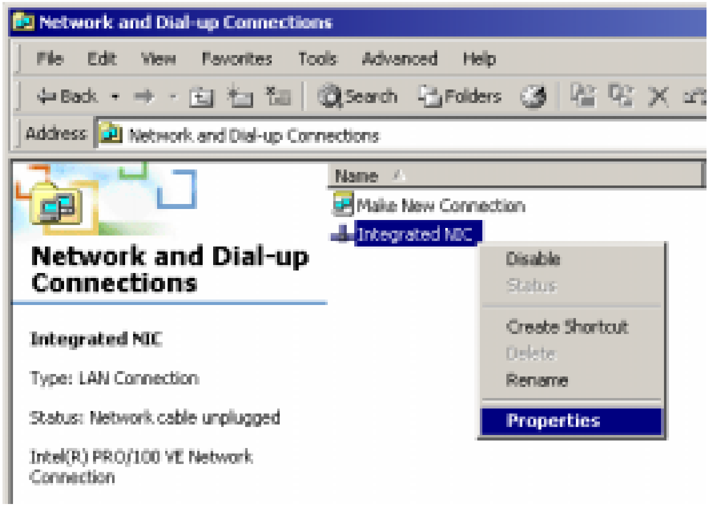
Windows® will then install the protocol and return you to the Network control panel. During this installation process, the PC may ask for the Windows 98® CD-ROM and will copy relevant files to the PC's hard drive. After this is done, restart your PC and continue to the Communicating with the SmartImage Sensor section.

For Windows 2000® (Windows XP® will be similar):

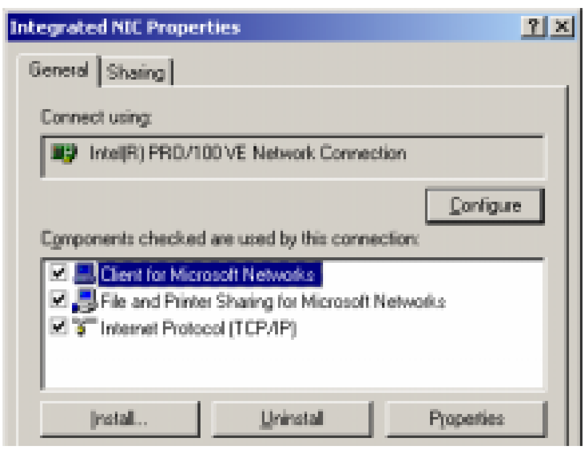
Click on the **Start** menu in Windows®, select **Settings**, then **Control Panel**.

Double click on the **Network and Dial-Up Connections** Control Panel icon (shown above right).

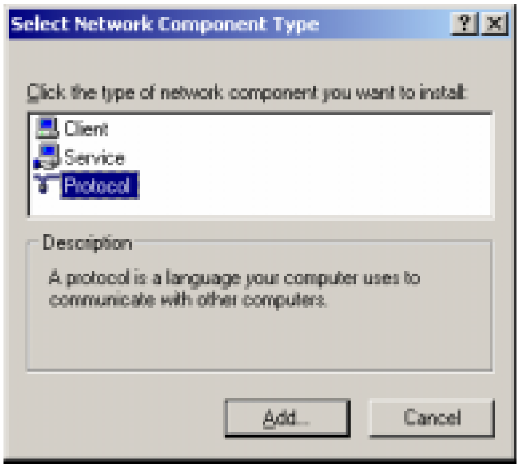
To check for the list of installed network components for your network card, right-click on it and select **Properties** (as shown below). If TCP/IP is on the list in the corresponding dialog box, click **Cancel** and skip to the Communicating with the SmartImage Sensor section.



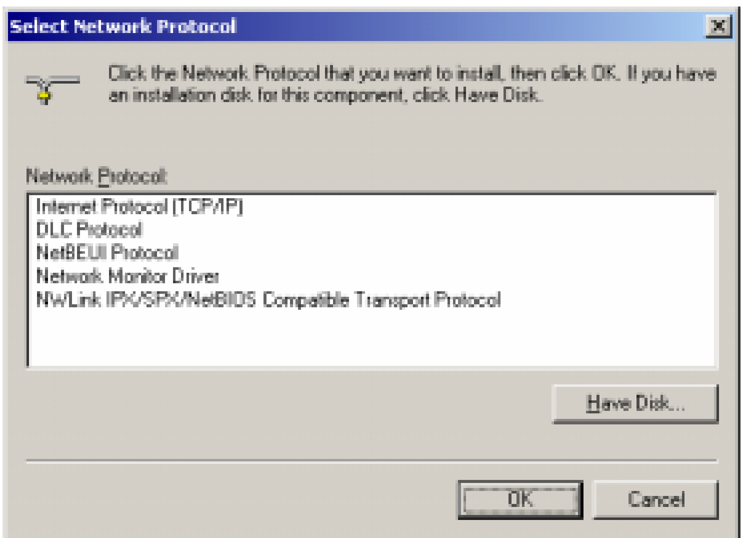
If TCP/IP is not on the list, click the **Install...** button.



Click on **Protocol**, then **Add...** (shown below).



Select **Internet Protocol (TCP/IP)** then click **Ok** (see below).



Windows® will then install the protocol and return you to the Network control panel. During this installation process, the PC may ask for the Windows 2000® CD-ROM and will copy relevant files to the PC's hard drive. After this is done, restart your PC and continue to the Communicating with the SmartImage Sensor section.

Appendix C - Advanced TCP/IP

The SmartImage Sensor runs TCP/IP over an Ethernet (RJ-45 only) connection. This Appendix provides some advanced information about this setup that may be useful to Network Administrators.

TCP/IP Installation Tips

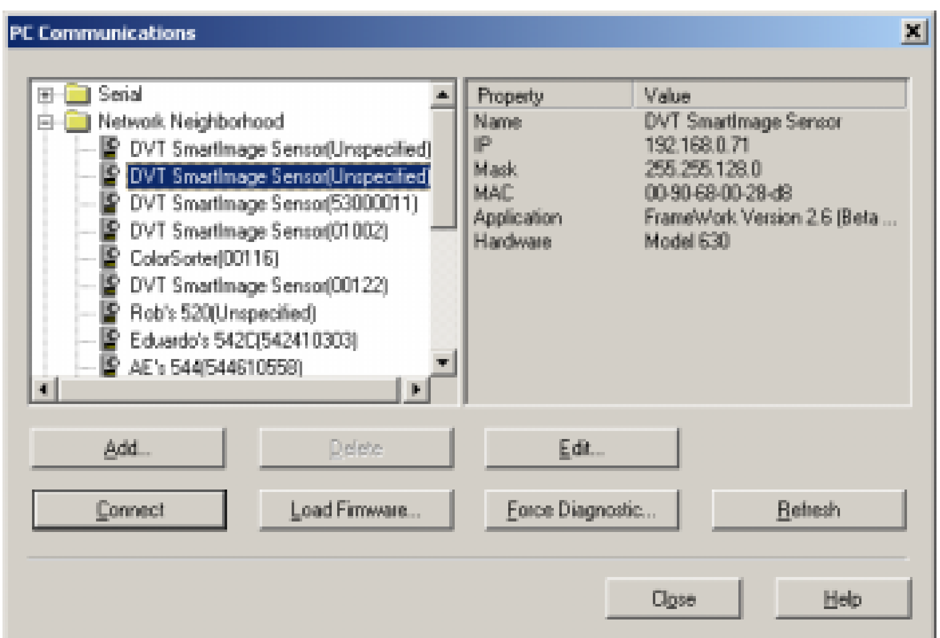
SmartImage Sensors are shipped with a random IP address, which must be changed according to your network. When assigning an IP address to the SmartImage Sensor, you need to make sure that it is one that has not already been assigned to another device on the network.

Currently, SmartImage Sensors do not support DHCP.

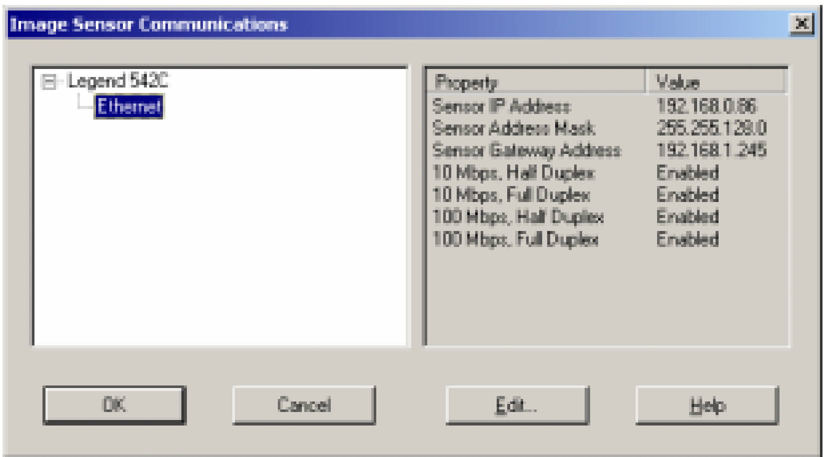
SmartImage Sensors do not require a Default Gateway to be set. For more information on what a Default Gateway is, please refer to the respective section later in this Appendix.

Typically, when connecting your PC to the SmartImage Sensor, the network masks must match.

The IP address can be changed from FrameWork software (**Comm** menu, **Image Sensor Communications** selection), or from the Network Neighborhood Browse in FrameWork 2.3 or greater (see below).



Highlighting **Ethernet** in left pane of FrameWork's **PC Communications** dialog box gives the PC (host) IP address (see below).



TCP/IP Troubleshooting Tips

IP addresses for each PC (and for each network card installed on the PC) and each device that uses the TCP/IP protocol (such as the SmartImage Sensor) is required to have a unique IP address.

IP addresses should not use 0 or 255 as the last value (e.g. 192.168.1.0 and 192.168.1.255, respectively). These are reserved and have special meaning:

Addresses that end in 0 specify all addresses for a particular subnet.

Addresses that end in 255 specify a broadcast packet that is sent to all hosts on the network.

With both PC and SmartImage Sensor using the Subnet (or Address) mask of 255.255.255.0, the IP addresses of PC and 600 must be identical for the first three values and unique in the fourth value. **Example**: SmartImage Sensor (IP: 192.168.0.242) & PC (IP: 192.168.0.240). Please refer to the section on subnetting later in this Appendix.

Query the SmartImage Sensor for its IP address in two places. 1) Run FrameWork, click **Comm**, **Image Sensor Communications**, select "Ethernet – Port A" in the left pane and view the IP address in the right pane. 2) From a terminal command line (serial communications, HyperTerminal @ 38400 baud, 8-N-1, no flow control), the '#Cn' command queries the SmartImage Sensor for its IP address and address mask.

Type ‘ping ###.###.###.###’ (replacing # with numbers) to check if an IP address is connected to the network. This command can be useful in determining if a node is powered up, operational, and physically connected to the network.

Type ‘arp –a’ from a DOS prompt to view the ARP (Address Resolution Protocol) table. The response should look something like this:

```
C:\WINDOWS>arp -a
Interface: 192.0.0.111
  Internet Address      Physical Address      Type
  192.0.0.211           00-90-68-00-00-42     dynamic
  192.0.0.212           00-90-68-00-00-42     dynamic
  192.0.0.253           00-60-97-4f-c0-45     dynamic
```

The “Internet Address” column refers to the IP addresses you’ve recently communicated with. The “Physical Address” column contains the corresponding MAC Addresses for the IP’s in column 1. DVT SmartImage Sensors will have Physical Addresses starting with “00-90-68”. In the case above, 192.0.0.253 is not a SmartImage Sensor, but another PC on the same network as the named Interface. The ARP table is cleared approximately every two minutes. The ‘arp’ command is useful in determining if an IP address belongs to a SmartImage Sensor system or another node on the network.

TCP/IP Subnetting

Please note that the creation of custom subnets is beyond the scope of this appendix and so will not be discussed. This section will only discuss default subnets (i.e. 255.0.0.0, 255.255.0.0, and 255.255.255.0), some key terms relating to subnetting, and some basic rules on setting up IP addresses with the default subnet masks.

Subnetting is a common technique among network administrators to segregate a network so that network traffic remains in certain areas (or subnets). If a device in one subnet needs to communicate to another device in another subnet and both devices reside on the same physical network, a router will be required (see the definition of a Default Gateway) to provide that link between the two areas. However if the two devices reside in the same subnet, then no router is required and the two devices can communicate with each other. As a rule of thumb, if the subnet mask is 255.255.255.0 on all devices, then the first three numbers of an IP address, which is the Network ID (e.g. **192.168.203**.1), will need to be the same and the last number, which is the Host ID (e.g. 192.168.203.**1**, 192.168.203.**2**, 192.168.203.**3**, etc.), will need to be unique in order for those devices to communicate with each other. Similarly, if the subnet mask is 255.255.0.0 on all devices, then the first two numbers of an IP address will need to be the same and the last two can be different.

Below are some key terms related to subnetting:

Physical Segment – A Physical Segment (or subnet) is a portion of the network that can receive a broadcast packet. All computers within a specified Physical Segment will share a common Network ID. Physical Segments are separated by routers because routers cannot forward broadcast packets; they can, however, traverse hubs, switches, and bridges.

Network ID – The Network ID is the part of an IP address that is common to all computers on a specified physical segment. The Network ID is equivalent to the area code of a phone number.

Host ID – The Host ID is the part of an IP address that uniquely identifies the device on a specified physical segment. The Host ID is equivalent to the phone number.

Default Gateway – A Default Gateway is typically a router that provides a route between two Physical Segments. It does not forward broadcast packets. So, if your SmartImage Sensor resides in one Physical Segment and your PC resides in another Physical Segment, then you will need to setup a Default Gateway to provide a route between the two Physical Segments. You would then need to setup up your SmartImage Sensor and your PC to point to that Default Gateway. Specifying a Default Gateway is not required, especially if both your SmartImage Sensor and your PC reside on the same subnet.

Subnet Mask (SNM) – A Subnet Mask is the second element required in TCP/IP communications, along with the IP address. Without a subnet mask, communication will not be able to take place between two network devices. The Subnet Mask also helps you to identify what is the Network ID and what is the Host ID in an IP address.

Appendix D - Upgrading or Reinstalling Legend Series SmartImage Sensor's Firmware

Before installing FrameWork on your PC and upgrading the firmware file in the SmartImage Sensor, take the time to backup the existing system to the PC.

To back up a SmartImage Sensor system to a PC:

- Run the older version of FrameWork and connect to the SmartImage Sensor.
- Click on the **Products** menu.
- Choose Backup System to PC.
- When prompted, provide a filename.
- Exit FrameWork.

Now that the old system has been fully backed up, FrameWork may be installed on the PC.

To install FrameWork on a PC:

- Insert the FrameWork CD. The installation should automatically start.
- If the CD does not automatically start, do the following:
    - From the **Start** menu, choose **Run**.
    - Type **D:\Setup.exe** (substitute the appropriate drive letter of your CD-ROM drive for **D**) and press **Enter**.

OR

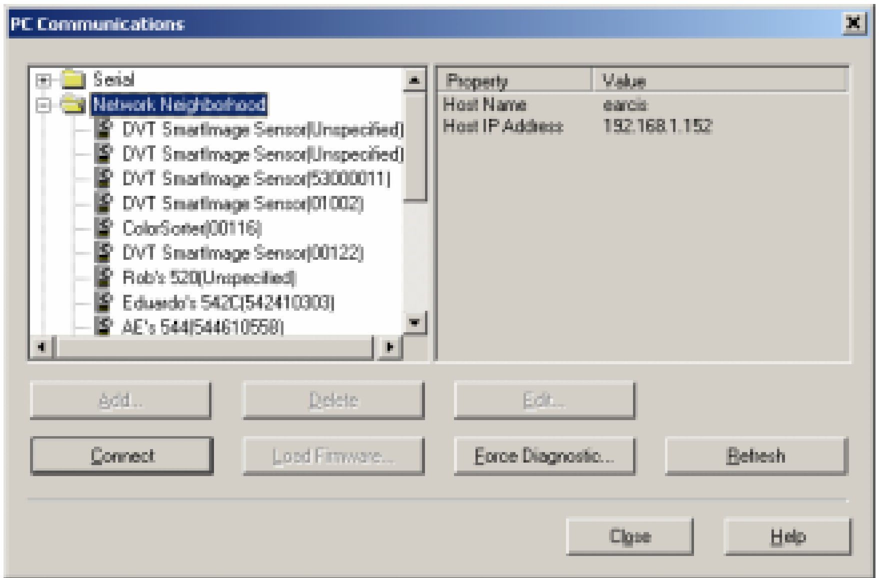
- Run the installation file downloaded from the DVT website
- Follow the installation instructions on screen.
- After FrameWork has been installed on the PC, the firmware file on the SmartImage Sensor must be upgraded.

Upgrading FrameWork firmware on a Legend Series SmartImage Sensor

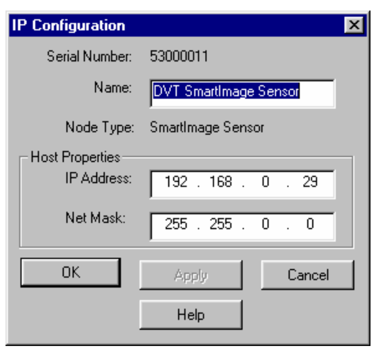
Note: The SmartImage Sensor is a stand-alone unit that runs the firmware. All SmartImage Sensors arrive from the factory with FrameWork firmware installed. This step should not be necessary unless reinstalling or upgrading firmware.

Before upgrading, start FrameWork and click Help|About to verify the version of the software being used.

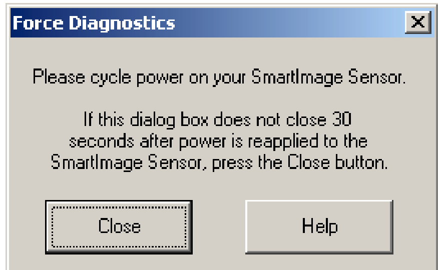
1. Start FrameWork. The "PC Communications" dialog box will come up.
2. Select "Network Neighborhood". This will search your local network for DVT systems.



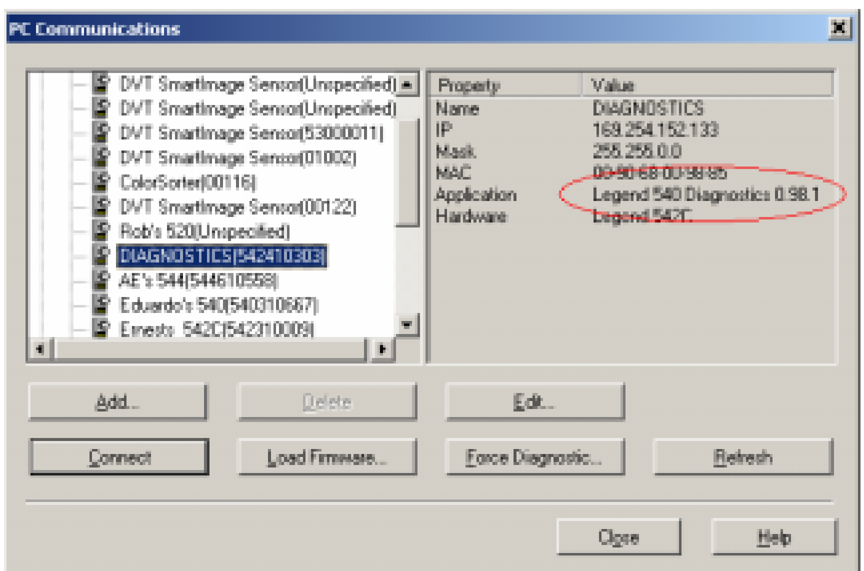
3. Highlight the appropriate sensor by choosing the system with the matching serial number and make sure the IP address and subnet mask are compatible with your computer's IP address and subnet mask.
4. If you need to change the IP address and/or the subnet mask of the SmartImage Sensor, click on the edit button. The following dialog box will come up and you will be able to change these parameters and the name with which this particular SmartImage Sensor is identified.



5. It is recommended that before loading firmware into the system, the user erases the content of flash memory. This will avoid potential problems that could arise when loading different firmware versions on top of one another. The first step to do this is forcing the system into diagnostics mode. Select the appropriate SmartImage Sensor under "Network Neighborhood" and click on the "Force Diagnostics" button in the "PC Communications" dialog box. FrameWork will instruct you to cycle power on the system you are trying to upgrade.

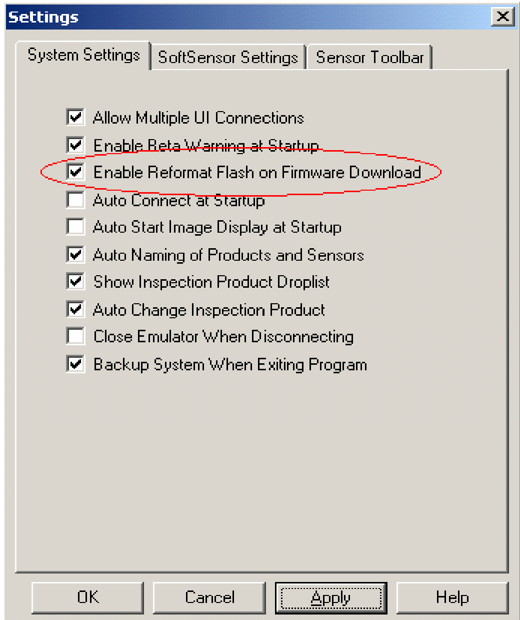


6. After the system boots up it should be in diagnostics mode. You can check this by verifying that the "Status" LED on the back of the SmartImage Sensor is blinking red. Also under "Network Neighborhood", the system should appear now as "DIAGNOSTICS".
7. When the system is forced into diagnostics mode, it resets its IP address. This means that at this point you would need to select the system, click on the "Edit…" button and reassign the appropriate IP address and net mask.
8. Now that the system is in diagnostics mode you can proceed to erase flash memory. While the SmartImage Sensor is in diagnostics, select it under "Network Neighborhood" and look at the right hand pane of the "PC Communications" dialog box to identify the diagnostics version of the system.

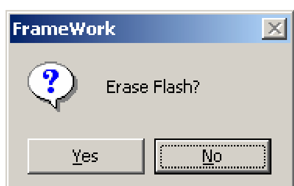


9. If this version is 0.28 or a more recent release (like in the case above) you would be able to erase flash memory and load the new firmware version from FrameWork. If your system has an older diagnostics version go to the section "Erasing flash using Telnet" at the end of this appendix and then continue on step 11. The mentioned section explains a fail-safe method to erase flash memory; so if your system has a recent diagnostics version but the procedure described below to accomplish this task does not work, you can still refer to the technique covered at the end of the appendix to achieve your objective of clearing the contents of permanent memory.
10. Before proceeding, make sure that the option "Enable Reformat Flash on Firmware Download" is enabled in FrameWork. You could verify this by closing the "PC

Communications" dialog box and clicking on "Options..." under the "Edit" menu in FrameWork. The option is indicated in the next figure.

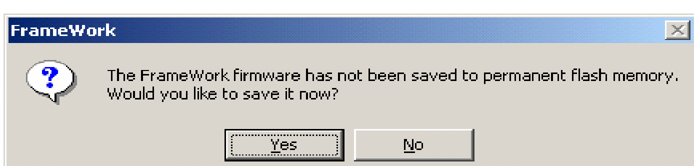


11. On the "PC Communications" dialog box accessed through the "Comm" menu, click on "Network Neighborhood", select the SmartImage Sensor on the list and click on the "Load Firmware" button. Select the firmware version that you want to load based on your model. The firmware files for all the systems are included in the FrameWork installation and their extensions identify the sensors for which they were designed. At this point, if you are erasing flash from FrameWork, you should select "Yes" when the message shown below appears.



12. Wait for approximately 30 seconds. If the progress bar closes before that, make sure the correct SmartImage Sensor is highlighted in the left hand pane and click the "Connect" button.

13. Before getting into FrameWork, the program will ask you if you would like to store the firmware to Flash Memory. Choose Yes. It will take a few seconds to store the file to flash memory. Then the Sampled Image Display will appear in the User Interface of Framework.



Erasing flash using Telnet

This is a fail-safe method to erase flash memory. You should read this section and follow the instructions explained if the diagnostics version of the system you are trying to upgrade is older than version 0.28 or if the procedure covered in steps 10-11 above does not work.

After forcing the system into diagnostics mode and reassigning the appropriate IP address and net mask, you can proceed to clear the contents of flash memory. You should use telnet or any other terminal emulator to connect to the SmartImage Sensor through Ethernet on port 23 (which is the default port for telnet.) Telnet is included as an accessory in all versions of Windows®.

- Click on "Start" then "Run…"
- Type "telnet" and press Enter.
    - In Windows 2000®, XP and NT telnet is included as a DOS-based program. In this case you will see a DOS console appear. At the prompt, type "open" followed by a space and then the SmartImage Sensor's IP address.
    - In Windows 98® and ME telnet is a Windows®-based application. When the program comes up, click on the "Connect" menu, then select "Remote System…" In the dialog box that appears type the IP address of your system under "Host Name", select "telnet" as the "Port" and for "Term Type" choose "VT100".
- After you are connected, you will see some information about the system displayed. At the question mark prompt, type the following command without the quotes: "*F1bdc" (You might not see what you are typing if you do not have local echo enabled in the telnet program.)
- You should see the message "Erasing Flash". After the process is completed, you will receive the question mark prompt. At this time, permanent memory has been erased and you should close the telnet program.

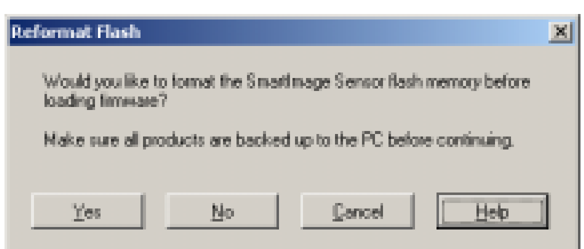
Appendix E - Upgrading or Reinstalling Series 600 SmartImage Sensor's Firmware

Refer to the first section of Appendix B to learn how to install FrameWork and back up a SmartImage Sensor system to your PC. The process of backing up your system is very important since you will probably be erasing flash memory before loading a new version of FrameWork firmware to the sensor. You should also backup the system file when installing a newer version of FrameWork.

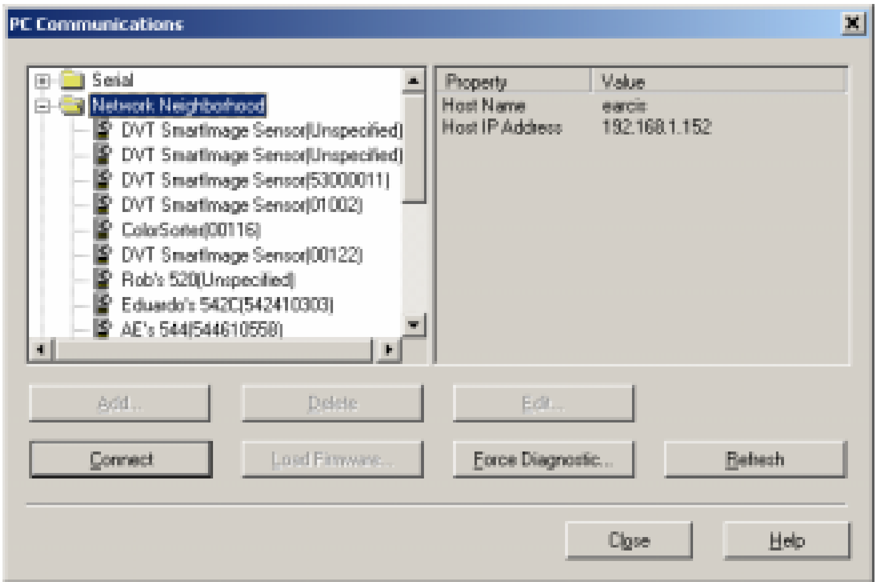
Note: The SmartImage Sensor is a stand-alone unit that runs the firmware. All SmartImage Sensors arrive from the factory with FrameWork firmware installed. This step should not be necessary unless reinstalling or upgrading firmware.

Before upgrading, start FrameWork and click Help|About to verify the version of the software being used.

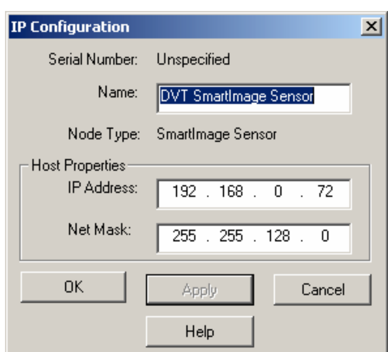
1. Start FrameWork. The "PC Communications" dialog box will come up.
2. It is recommended that before loading firmware into the system, the user erases the content of flash memory. This will avoid potential problems that could arise when loading different firmware versions on top of one another. We will force the system into diagnostics mode in order to erase flash memory. FrameWork provides a way to do all this through the serial port at the click of a button. We will discuss this method next. However if for any reason, this procedure does not work refer to the section "Forcing Diagnostics and Erasing Flash from HyperTerminal" at the end of this appendix. This last section describes a fail-safe method to put the system into diagnostics mode and erase flash memory.
3. To complete this process entirely from FrameWork, click on "Start" on the Windows® taskbar. Then click on "Run" and type "FWK.ini" without the quotes. This will open up the FrameWork initialization file. Once here make sure the line ReformatFlashEnable=1 exists under [Settings]. Add this line if necessary. This will ensure that FrameWork will ask the user whether to erase flash memory or not after the system is in diagnostics.
4. Select the appropriate port under "Serial" and click "Load Firmware". Choose the firmware version that you want to load. The firmware files for all the systems are included in the FrameWork installation and their extensions would identify the sensors for which they were designed. If you used the instructions on the section "Forcing Diagnostics and Erasing Flash from HyperTerminal" skip to step 6.
5. FrameWork will ask whether you want to format flash memory before loading firmware or not. Select "Yes".



6. After the loading process is completed, you can connect to the SmartImage Sensor. You have two options: to establish a serial or an Ethernet connection. To connect using the serial port, select the appropriate port and click “Connect” and continue on step 9. To connect via Ethernet select “Network Neighborhood”.This will browse the local network for DVT systems.



7. Highlight the appropriate sensor by choosing the system with the matching serial number and make sure the IP address and subnet mask are compatible with your computer’s IP address and subnet mask. If you need to change the IP address and/or the subnet mask of the SmartImage Sensor, click on the “Edit” button. The following dialog box will come up and you will be able to change these parameters.



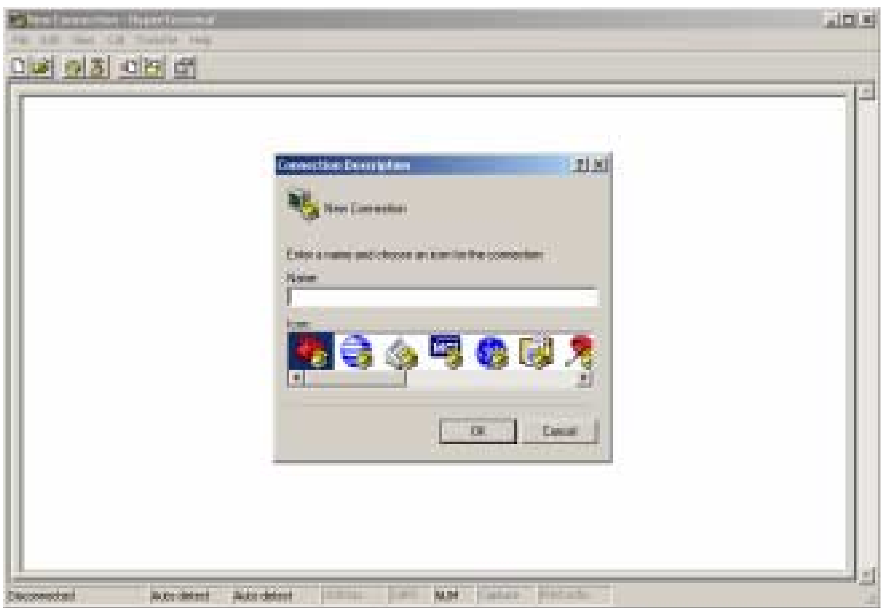
8. Select the SmartImage sensor under “Network Neighborhood” and click “Connect”.

9. Initially the firmware is loaded into RAM memory. Framework will ask you to save the loaded firmware to flash memory every time a connection is established until you confirm this operation. This will complete loading firmware into the SmartImage Sensor.

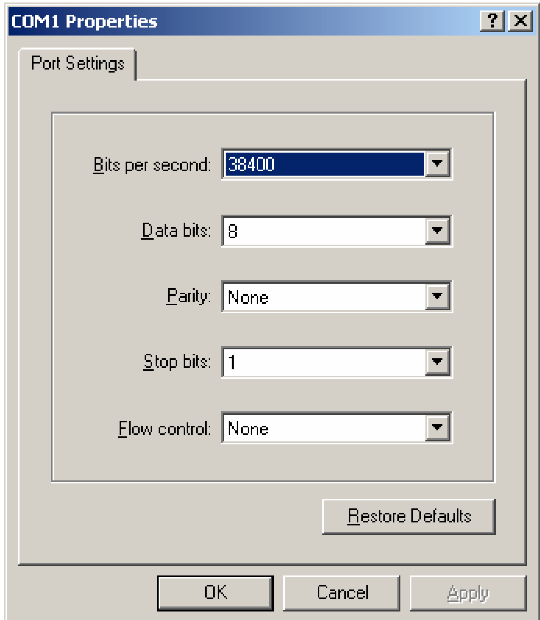
Forcing Diagnostics and Erasing Flash from HyperTerminal

This is a fail-safe procedure to force the system into diagnostics mode and erase the contents of flash memory. This method should be used when troubleshooting communication problem with the Series 600 SmartImage Sensor when all other procedures have failed. To do this you would have to establish a serial connection to the SmartImage Sensor with a terminal emulator. Although there are different programs that could be used to connect to other devices using the serial port on the PC, we will use HyperTerminal, which is included as an accessory in all versions of Windows®.

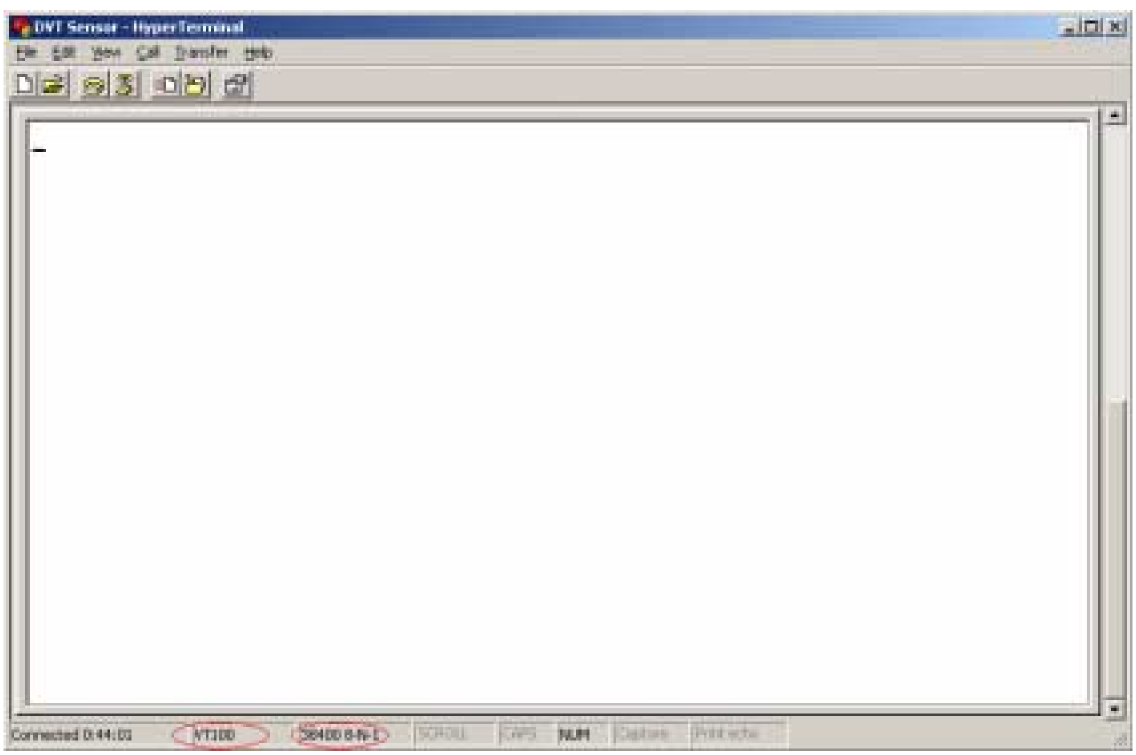
- To start this application click on “Start” -> “Programs” -> “Accessories” -> “Communications” -> “HyperTerminal”. You should get a window similar to the following:



- Now you will prepare a connection to the SmartImage Sensor. The first step is to assign a name for this connection. Next select the appropriate serial port in your PC. Then choose the following communication parameters:

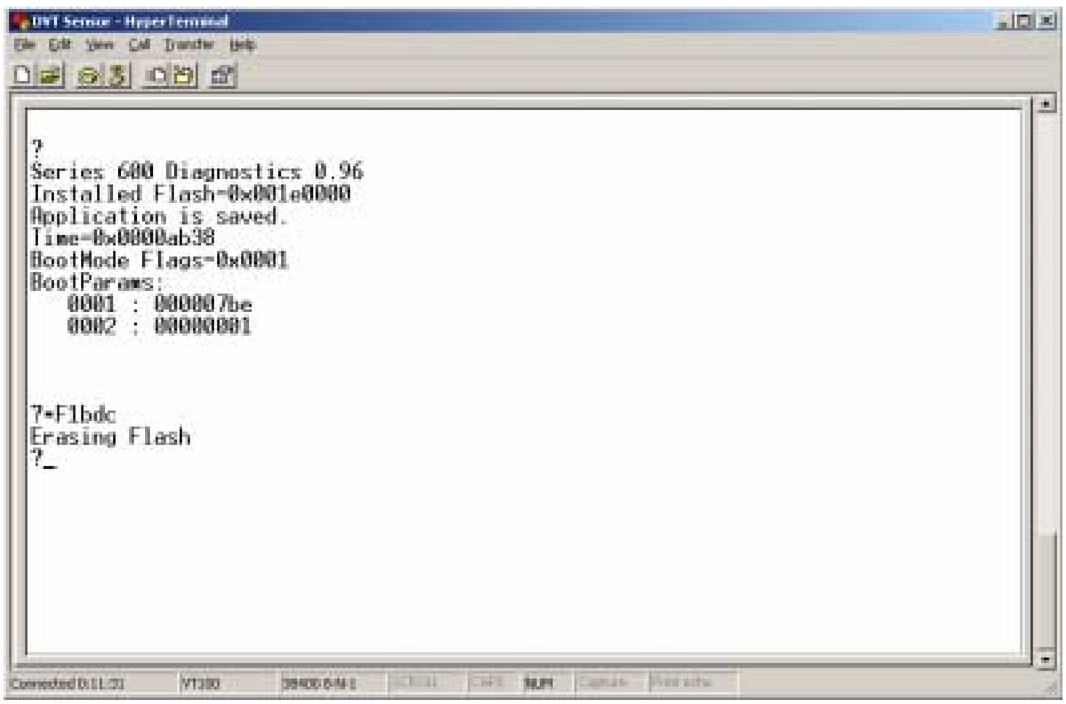


- Once connected, make sure that HyperTerminal displays the correct baud rate for communicating with the SmartImage Sensor as circled below. If that is not the case, disconnect, go to “Properties” under the “File” menu, and click “Configure” on the “Connect To” tab of the dialog box that comes up. Once here, specify again the communications parameters and click “OK”. Connect again.



- While connected, hold down the "Shift" and "+" keys (or hold down "Shift" and continuously tap the "+"key if the character repeat is not enabled in your computer) as you cycle power to the system. This will force the SmartImage Sensor into diagnostics mode. You can make sure that the system is in this state by verifying that the "Ready" LED is flashing red approximately every half a second. You should also receive some system information through the HyperTerminal connection.

- The next step is to erase flash memory. At the question mark prompt type the command "*F1bdc" without the quotes. The message "Erasing Flash" will be displayed. After a few seconds you will get another prompt; this means the process was completed successfully. Close the HyperTerminal connection.

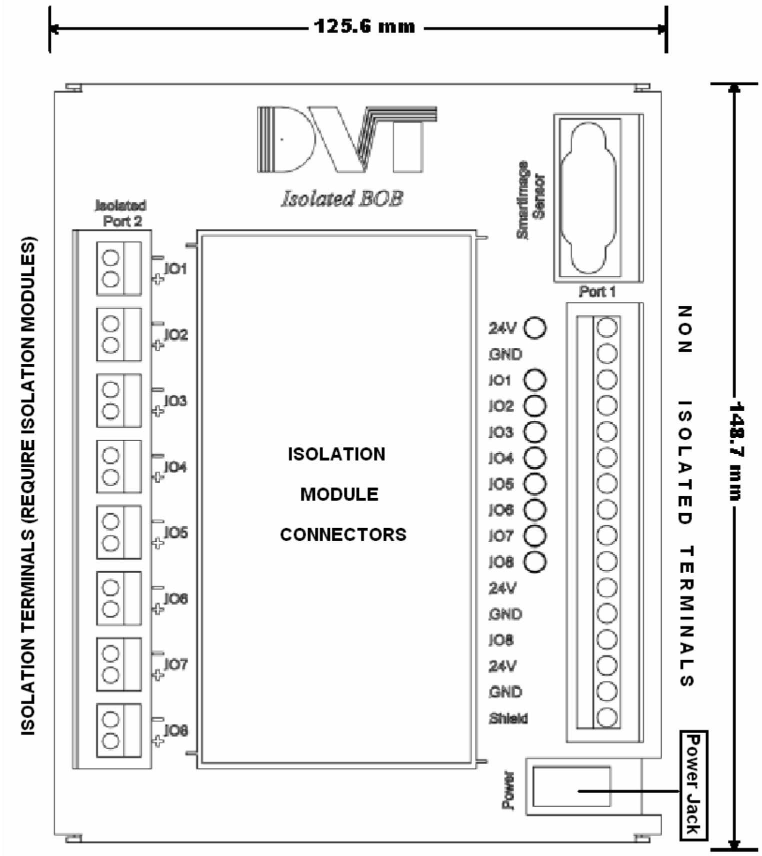


Appendix F – Breakout Boards

Isolation Breakout Board

The isolated breakout board for the Legend Series not only provides a convenient method to connect digital I/O, power, and strobe illumination lines; but it also enables the use of isolation modules. The board is housed in a metal/plastic DIN Rail-mountable case with powder coating on top. The breakout board is 126mm wide by 149mm high by 45mm deep (including all necessary cable clearances.) The board has two main ports. The port on the right includes I/O, strobe and power screw-down terminals. The I/O lines on this port are to be used when no isolation is necessary. The second port, on the left side, has the I/O connections to be used in conjunction with isolation modules. The optional isolation modules can be acquired from third-party companies to provide power isolation when working with devices that are using a separate power supply.

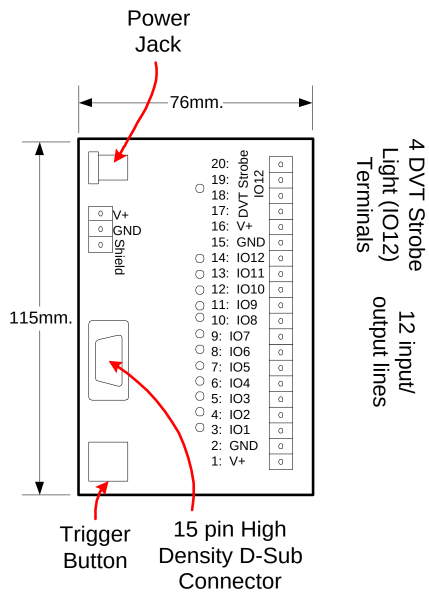
Features	Purpose
Power Jack	Included for the use of a 24V (1.42 A) wall-mount power supply. It should only be used for demonstration and laboratory purposes.
Power Terminals	There are three main power terminals (V+, GND, Shield) providing connections for an industrial power supply. There are two additional pairs of V+ and GND terminals.
Non-isolated I/O Terminals	Located on the right side of the board there are screw-down terminals for the 8 I/O lines available on the Legend Series. There is an extra terminal connected to I/O line 8 intended to provide one more output for a strobe illumination. This output will be activated during image exposure only if it is configured in FrameWork as the Strobe output.
Isolated I/O Terminals	Located on the left side of the board, these are to be used in combination with isolation modules.
Isolation modules slots	There are 8 screw-down slots for input or output isolation modules.
15 pin high density D-sub plug	This provides connection to the I/O and power lines of the Legend Series systems which use a 10-pin keyed RJ-45 port.


125.6 mm
148.7 mm
Isolated BOB
SmartImage Sensor
Isolated Port 2
ISOLATION TERMINALS (REQUIRE ISOLATION MODULES)
IO1
IO2
IO3
IO4
IO5
IO6
IO7
IO8
ISOLATION
MODULE
CONNECTORS
Port 1
24V
GND
IO1
IO2
IO3
IO4
IO5
IO6
IO7
IO8
24V
GND
IO8
24V
GND
Shield
NON ISOLATED TERMINALS
Power
Power Jack

Series 600 Breakout Board

The I/O Breakout Board for the Series 500 and 600 (CON-BRK) provides an easy way to interface digital I/O, power, and strobe illumination lines. The board is housed in a plastic DIN Rail-mountable case. The Breakout Board is 75mm wide by 115mm high by 155mm deep (including all necessary cable clearances).

Feature	Purpose
Power Jack	For demonstration purposes only. Use a 24V @ 1.42 A wall-mount power supply.
Power Terminals	3 terminals on left side of board (V+, GND, Shield) provide connections for an industrial power supply.
15 pin high density D-sub plug	Digital I/O and Power lines. The board breaks these lines out on screw down terminals.
Trigger Button	Connected to IO1. If configured as a Trigger Input in FrameWork, this button will trigger inspections.
I/O Terminals	Individual screw-down terminals for all 12 I/O lines, V+, and GND. (Note: for the Series 500, I/O 1-4 in FrameWork map to IO 1-4 on the breakout board, I/O 5-8 in FrameWork map to IO 9-12 on the breakout board.)
Strobe Terminals	Four screw-down terminals, tied to IO12. These terminals will be activated during image exposure time if IO12 is configured in FrameWork as the Strobe Output,.



Index