# Serial Communication via COM-Port

of

## Optris PIX Connect Software

(Rel. 3.1.3015.0)

A brief Overview

## Introduction

One of the features of the PIX Connect software is the ability to communicate via a serial COM-Port interface. This can be a physical COM-Port or a virtual COM-Port (VCP). It must be available on the computer where the PIX Connect software is installed.

## Setup of the interface

To enable the software for the serial communication open the Configuration dialog and enter the tab “External Communication”. Choose the mode “COM-Port” and select a port. Also select the baud rate that matches the baud rate of the other communication device. The other interface parameters are 8 data bits, no parity and one stop bit (8N1). This is mostly used on other communication devices too. The other station must support 8 bit data.
Now connect the computer with the other communication device. If this is a computer too, use a null modem cable.

## Command list

The following commands are implemented. Any command must end with a CR/LF (0x0D, 0x0A).

| Read Commands | Description | Sample (Cmd./Answer) | Remarks |
|---|---|---|---|
| **?T** | Read temperature of main measure area | ?T<br>!T=24.9°C | |
| **?T(i)** | Read temperature of measure area with index i | ?T(1)<br>!T(1)=27.7°C | To get the count of indices use the command *?AreaCount* first |
| **?C** | Read chip temperature | ?C<br>!C=40.0°C | |
| **?F** | Read flag temperature | ?F<br>!C=32.0°C | |
| **?I** | Read internal temperature | ?I<br>!C=32.0°C | |
| **?E** | Read fixed emissivity value | ?E<br>!E=0.950 | |

| | | | |
|---|---|---|---|
| **?XG** | Read fixed transmissivity value | ?XG<br>!XG=1.000 | |
| **?A** | Read fixed ambent temperature value | ?A<br>A=23.0°C | |
| **?Pix(x, y)** | Read temperature of a single pixel at position x/y | ?Pix(80,60)<br>!Pix(80,60)=24.9°C | First use the *!ImgTemp* command, otherwise the answer is “NoImage !” *) |
| **?SN** | Read serial number of imager | ?SN<br>!SN=8050012 | |
| **?CC** | Check if the configuration has changed since last check | ?CC<br>!CC=1 | 0=conf. has not changed<br>1=conf. has changed |
| **?AreaCount** | Read count of measure areas | ?AreaCount<br>!AreaCount=3 | |
| **?AreaConf(i)** | Read configuration of measure area with index i | ?AreaConf(0)<br>!AreaConf(0)=(120,18,150,24,Min) | left, top, right, and bottom coordinate and measure mode of the area |
| **?AreaLoc(i)** | Read location of measure area with index i | ?AreaLoc(0)<br>!AreaLoc(0)=88,42 | x, y coordinate of the area |
| **?AreaShape(i)** | Read shape ID of area i | ?AreaShape(0)<br>!AreaShape(0)=1 | ID: 0(off),<br>1(Point1x1),<br>2(Point3x3),<br>3(Point5x5),<br>4(UserRect),<br>5(Ellipse),<br>6(Polygon),<br>7(Spline) |
| **?AreaMode(i)** | Read mode ID of area i | ?AreaMode(0)<br>!AreaMode(0)=2 | Id: 0(Min), 1(Max),<br>2(Average),<br>3(Distribution) |
| **?AreaBindProfile(i)** | Read if area i is bound to profile | ?AreaBindProfile(0)<br>!AreaBindProfile(0)=1 | 0=not bounded<br>1=bounded |
| **?AreaEmissivity(i)** | Read custom emissivity x of area i | ?AreaEmissivity(0)<br>!AreaEmissivity(0)=0.953 | x = [0.000 … 1.000] |
| **?AreaUseEmissivity(i)** | Read if area i is using custom emissivity | ?AreaUseEmissivity(0)<br>!AreaUseEmissivity(0)=0 | 0=not used<br>1=used |
| **?AreaShowInDigitalGroup(i)** | Read if area I is shown in digital group | ?AreaShowInDigitalGroup(0)<br>!AreaShowInDigitalGroup(0) = 1 | 0=not shown<br>1=shown |
| **?AreaDistributionModeRange(i)** | Read the distribution range x1,x2 of area i | ?AreaDistributionModeRange(0)<br>?AreaDistributionModeRange(0)<br>=20.0,50.0 | Temperature range [x1 … x2] for distribution mode<br>x1 < x2 |
| **?AreaSize(i)** | Read size w,h of area i | ?AreaSize(0)<br>!AreaSize(0)=75,30 | w … width<br>h … height |
| **?AreaIsHotSpot(i)** | Read if area i is hot spot | ?AreaIsHotSpot(0)<br>!AreaIsHotSpot(0)=1 | 0=not hotspot<br>1=hotspot |
| **?AreaIsColdSpot(i)** | Read if area I is cold spot | ?AreaIsColdSpot(0)<br>!AreaIsColdSpot(0)=0 | 0=not coldspot<br>1=coldpot |
| **?AreaName(i)** | Read name of area i | ?AreaName(0)<br>!AreaName=Area01 | |
| **?OpticsCount** | Read count of existing optics | ?OpticsCount<br>!OpticsCount=2 | |
| **?RangeCount** | Read count of existing temperature | ? RangeCount<br>! RangeCount=3 | |

| | | | |
|---|---|---|---|
| | ranges | | |
| **?VideoCount** | Read count of existing video formats | ?VideoCount<br>!VideoCount=3 | |
| **?OpticsIndex** | Read the index of the current optics | ?OpticsIndex<br>!OpticsIndex=0 | |
| **?RangeIndex** | Read the index of the current temperature range | ?RangeIndex<br>!RangeIndex=1 | |
| **?VideoIndex** | Read the index of the current video format | ?VideoIndex<br>!VideoIndex=1 | |
| **?OpticsFOV(i)** | Read the field of view of the optics with index i | ?OpticsFOV(1)<br>!OpticsFOV(1)=30 | To get the count of indices use the command *?OpticsCount* first |
| **?RangeMin(i)** | Read the low temperature of the temperature range with index i | ?RangeMin(0)<br>!RangeMin(0)=-20.0°C | To get the count of indices use the command *?RangeCount* first |
| **?RangeMax(i)** | Read the high temperature of the temperature range with index i | ?RangeMax(0)<br>!RangeMin(0)=100.0°C | To get the count of indices use the command *?RangeCount* first |
| **?VideoFormat(i)** | Read the video format settings of video format with index i | ?VideoFormat(0)<br>!VideoFormat(0)=382x288@80 | To get the count of indices use the command *?VideoCount* first |
| **?VAppl** | Read the version of the imager.exe | ?VAppl<br>!VAppl=1.2.1129.0 | |
| **?Img($x_0$,$y_0$,$x_1$,$y_1$)** | Binary read the temperatures of the pixels in rectangle with upper left corner: $x_0$,$y_0$ lower right corner: $x_1$,$y_1$ | ?Img(0,0,159,3){ answer is binary: 640 bytes, 2 Bytes per pixel } | First use the *!ImgTemp* command, otherwise the answer is "NoImage !" **) Maximum number of transferrable pixels is 20000 (40000 Bytes) |
| **?ImgHex($x_0$,$y_0$,$x_1$,$y_1$)** | Hexadecimal read the temperatures of the pixels in rectangle with upper left corner: $x_0$,$y_0$ lower right corner: $x_1$,$y_1$ | ?ImgHex(0,0,159,3) answer is binary: 1280 bytes, 4 Bytes per Pixel, $05B1_{Hex} \rightarrow 1457_{Dec} \rightarrow 45.7°C$ | First use the *!ImgTemp* command, otherwise the answer is "NoImage !" **) Maximum number of transferrable pixels is 10000 (40000 Bytes) |
| **?RangeDec_Cali** | Read the number of decimal places according to calibration data | ?RangeDec_Cali<br>!RangeDec_Cali=1 | Return value depends from calibration data |
| **?RangeDec_Eff** | Read the effective number of decimal places | ?RangeDec_Eff<br>!RangeDec_Eff=1 | Return value depends from calibration data and from the checkbox "High resolution temp." on tab "Device" in the config. dialog****) |
| **?Flag** | Read flage state | ?Flag<br>!Flag=0 | 0 = flag is open<br>1 = flag is closed |
| **?InitCounter** | Read the "Initialization Conuter" | ?InitCounter<br>!InitCounter=1200 | Answer is the estimated time in msec. until the init |

| | | | |
|---|---|---|---|
| | | | process will be ready |
| **?Embedded** | Read the Embedded state | ?Embedded<br>!Embedded=0 | 0 = Appl. Isn't in embedded state ***)<br>1 = Appl. is in embedded state |
| **?WindowPos** | Read the applications location | ?WindowPos<br>!WindowPos(0, 0, 80, 80) | left, top, right, and bottom coordinate |
| **?AIi** | PIF: Read analogue input | ?AI1<br>!AI1=3.5 | i = input index |
| **?DIi** | PIF: read digital input | ?DI1<br>!DI1=1 | i = input index |
| **?DICount** | PIF: number of digital inputs | ?DICount<br>!DICount=1 | |
| **?AICount** | PIF: number of analogue inputs | ?AICount<br>!AICount=1 | |
| **?AOCount** | PIF: number of analogue outputs | ?AOCount<br>!AOCount=1 | |
| **?FocusmotorMinPos** | Get minimum position of Xi's focus motor | ?FocusmotorMinPos<br>!FocusmotorMinPos=1500 | |
| **?FocusmotorMaxPos** | Get maximum position of Xi's focus motor | ?FocusmotorMaxPos<br>!FocusmotorMaxPos=2500 | |
| **?FocusmotorPos** | Get actual position of Xi's focus motor | ?FocusmotorPos<br>!FocusmotorPos=1700 | |

| Set Commands | Description | Sample (Cmd./Answer) | Remarks |
|---|---|---|---|
| **!ImgTemp** | Get an image and return the image dimensions | !ImgTemp<br>!ImgTemp(160,120,2) | Must be called before *?Pix()* and *?Img()* commands |
| **!OpticsIndex=i** | Set the index of the current optics to i | !OpticsIndex=0<br>!OpticsIndex=0 | To get the count of indices use the command *?OpticsCount* first |
| **!RangeIndex=i** | Set the index of the current temperature range to i | !RangeIndex=1<br>!RangeIndex=1 | To get the count of indices use the command *?RangeCount* first |
| **!VideoIndex=i** | Set the index of the current video format to i | !VideoIndex=0<br>!VideoIndex=0 | To get the count of indices use the command *?VideoCount* first |
| **!Flag=0** | Flag open/close | !Flag=0<br>!Flag=0 | 0 = open flag<br>1 = close flag |
| **!E=x** | Set fixed emissivity value | !E=0.950<br>!E=0.950 | Valid range: 0.1 ... 1.1 |
| **!XG=x** | Set fixed transmissivity value | !XG=1.000<br>!XG=1.000 | Valid range: 0.1 ... 1.1 |
| **!A=x** | Set fixed ambent temperature value | !A=23.0<br>!A=23.0°C | |
| **!AreaLoc(i)=x,y** | Set location of measure area with index i | !AreaLoc(0)=88,42<br>!AreaLoc(0)=88,42 | x, y coordinate of the area |
| **!AreaShape(i)=n** | Set shape ID of area i | !AreaShape(0)=1<br>!AreaShape(0)=1 | ID: 0(off),<br>1(Point1x1),<br>2(Point3x3),<br>3(Point5x5),<br>4(UserRect),<br>5(Ellipse),<br>6(Polygon),<br>7(Spline) |
| **!AreaMode(i)=n** | Set mode ID of | !AreaMode(0)=2 | Id: 0(Min), 1(Max), |

| | | | |
|---|---|---|---|
| | area i | !AreaMode(0)=2 | 2(Average), 3(Distribution) |
| **!AreaBindProfile(i)=x** | Bind area to profile | !AreaBindProfile(0)=1<br>!AreaBindProfile(0)=1 | 0=not bounded<br>1=bounded |
| **!AreaEmissivity(i)=x** | Set custom emissivity x of area i | !AreaEmissivity(0)=0.953<br>!AreaEmissivity(0)=0.953 | x = [0.000 … 1.000] |
| **!AreaUseEmissivity(i)=x** | Set custom emissivity x of area i | !AreaUseEmissivity(0)=0<br>!AreaUseEmissivity(0)=0 | 0=not used<br>1=used |
| **!AreaShowInDigitalGroup(i)=x** | Show area i in digital group | !AreaShowInDigitalGroup(0) = 1<br>!AreaShowInDigitalGroup(0) = 1 | 0=not shown<br>1=shown |
| **!AreaDistributionModeRange(i)=x1,x2** | Set the range x1,x2 of area I for distribution mode | !AreaDistributionModeRange(0) =20.0,50.0<br>!AreaDistributionModeRange(0) =20.0,50.0 | Temperature range [x1 … x2] for distribution mode x1 < x2 |
| **!AreaSize(i)=w,h** | Set size(w,h) of area i | !AreaSize(0)=75,30<br>!AreaSize(0)=75,30 | w … width<br>h … height |
| **!AreaIsHotSpot(i)=x** | Set area i for hot spot | !AreaIsHotSpot(0)=1<br>!AreaIsHotSpot(0)=1 | 0=not hotspot<br>1=hotspot |
| **!AreaIsColdSpot(i)=x** | Set area i for cold spot | !AreaIsColdSpot(0)=0<br>!AreaIsColdSpot(0)=0 | 0=not coldspot<br>1=coldpot |
| **!AreaName(i)=name** | Set name of area i | !AreaName(0)=Area01<br>!AreaName(0)=Area01 | |
| **!AOi=x** | PIF: Set AO1, AO2, AO3 | !AO1=5.43<br>!AO1=5.43 | Valid range:<br>0.00V ... 10.00V |
| **!FocusmotorPos** | Set actual position of Xi's focus motor | !FocusmotorPos=1500<br>!FocusmotorPos=1500 | |
| **!Snapshot** | Make snapshot | !Snapshot | |
| **!RecordStart** | Start record | !RecordStart | |
| **!RecordStop** | Stop record | !RecordStop | |

| Other Commands | Description | Sample (Cmd./Answer) | Remarks |
|---|---|---|---|
| **!Close** | Close the imager application | !Close<br>!Closed | The answer of course is generated before the application is closed. |
| **!Reinit** | Re-initializes the imager | !Reinit<br>!Reinit started | |
| **!Layout=x** | Loads an existing layout | !Layout=Multiple Areas | Layout must exist in the layout directory |
| **!Embedded** | Set the Embedded state | !Embedded=0<br>!Embedded=0 | 0 = Appl. Isn't in embedded state<br>1 = Appl. is in embedded state |
| **!WindowPos** | Set the applications location | !WindowPos(0, 0, 80, 80)<br>!WindowPos(0, 0, 80, 80) | left, top, right, and bottom coordinate |

*) see: Getting data from a single pixel
**) see: Getting data from a region
***) Embedded means: The video window can be embedded into another application. The user sees just the video window, no tool windows, no tool bars, and no frames. The window is always on top.
****) This information is needed to determine in which format the temperature values of the matrix will be send:

- Decimal places: 1 → temperature value of pixel is unsigned short → T[°C] = (value-1000)/10
- Decimal places: 2 → temperature value of pixel is signed short → T[°C] = value/100

| Issue | Response |
|---|---|
| Unknown command | “Unknown Command! xxxx”. |
| Known command with wrong syntax | “Bad Syntax!” |
| Known command with wrong index | “Wrong Index!” |
| Known commands with wrong parameter | “Wrong Parameter!” |
| Known commands is inappropriate for setting | “Inappropriate command!” |
| No Image available | “No Image!” |
| Known command, argument is out of valid range | “Out of range!” |

## Multiple Instances of PIX Connect - Addressing

As with the software release 2.7.2134.0 the software supports multiple devices. If a bus address is defined in the configuration dialog of PIX Connect, the software will listen to commands starting with this address only. The heading address must always have three digits. Addresses within the range of 1 ... 999 are valid. Any PIX Connect instance connected to the bus must be configured with a different bus address. No instance should be configured without a bus address. The answer also contains the address. If no device with the given address exists there will be no answer.

Examples:

| | | |
|---|---|---|
| **005?T** | Read the temperature of the main measure area of device with address 005 | Answer: 005!T=25.7°C |
| **010!Flag=1** | Close the flag of device with address 10 | Answer: 010!Flag=1 |

## Getting data from a single pixel

To enable the user to have more than one pixel information from the same video frame, make a copy of one video frame first before receiving the information:

| | |
|---|---|
| First use command: **!ImgTemp** | Stores an internal copy of one video frame |
| Answer: **!ImgTemp(160,120,2)** | The image has a size of 160x120 with 2 bytes per pixel. |
| Second request a pixel: **?Pix($x_0$,$y_0$)** | Ask for the pixel at position $x_0$, $y_0$ |
| Answer: **!Pix($x_0$,$y_0$)=25.0°C** | |
| **?Pix($x_n$,$y_n$)** | Ask for other pixels (all from the same frame) |
| … | |
| **!ImgTemp** | Preparing a new video frame |

## Getting data from a region

Likewise as getting a single pixel, at first use the `!ImgTemp` command.

| | |
|---|---|
| First use command: **!ImgTemp** | Stores an internal copy of one video frame |
| Answer: **!ImgTemp(160,120,2)** | The image has a size of 160x120 with 2 bytes per pixel. |
| Second request a rectangular region: **?Img($x_0$,$y_0$,$x_1$,$y_1$)** | Read the temperatures of the pixels in rectangle with upper left corner: $x_0$,$y_0$ and lower right corner: $x_1$,$y_1$ |
| Answer: binary data | |
| **?Img($x_2$,$y_2$,$x_3$,$y_3$)** | Ask for other regions (all from the same frame) |
| … | |
| **!ImgTemp** | Preparing a new video frame |

The answer to the `?Img(x0,y0,x1,y1)` command is binary. The user gets twice as much byte as the size of the rectangle – example:

`?Img(0,0,9,9)` → Size = 10 x 10 = 100 → answer length is 200

Do not transfer the whole image in one surge, as with `?Img(0,0,159,119)`. Consider two things: Not any hardware supports transmit or receive buffers of a size of 38400 bytes. Second: Sending of 38400 Bytes with 115200 Baud takes 3 seconds. This means the application has to wait this time for the answer. It is recommended to split the whole image in bites of one kilobyte. The order of image dimensions is not relevant (lines, rows or tiles).
Don't forget the `!ImgTemp` command before starting over with the next regions.

Data format of the image buffer:
Any pixel uses a word in the buffer. The first pixel is the upper left pixel in the rectangle. The last is the lower right. For the `?Img(0,0,9,9)` sample the user will get the pixels $P_{x,y}$ as follow:

$$P_{0,0}, P_{1,0}, P_{2,0}, \ldots P_{9,0}, P_{0,1}, P_{1,1}, P_{2,1}, \ldots P_{9,1}, \ldots \ldots \ldots P_{0,9}, P_{1,9}, P_{2,9}, \ldots P_{9,9}$$

Any word contains a temperature value. The format is used as in other Optris applications: Temperature = (Value -1000)/10

Berlin, 08.05.2018