



# **Talon Ultra**

## **Data Acquisition Software**

**User's Manual**  
**Version 1.51**

**Talon Ultra User's Guide**

# Talon Ultra Version 5.23 Data Analysis Software User's Guide

## Table of Contents

Talon Ultra User's Guide .....	1
Talon Ultra Version 5.23 Data Analysis Software User's Guide .....	2
Table of Contents .....	2
Talon Ultra Data Analysis Software User's Guide.....	3
Features of the Talon Ultra Data Analysis Software (V. 5.23) .....	3
Installation and Setup .....	3
Hardware installation.....	3
Software installation.....	3
External Connections.....	4
Quick Check Out.....	4
Interactive Operation Using the Talon Ultra Menus .....	6
ImagePro Plus Documentation .....	6
Additional Commands on the Acquire Menu .....	6
Camera Preview.....	6
Image Acquire .....	6
Memory Sequence .....	6
Additional Commands on the TalonUtils Menu .....	6
Average Sequence .....	6
Array Stats .....	7
Array Stats w/ Mask .....	7
Additional Command on the Help Menu .....	7
About Driver Versions .....	7
Data Acquisition and Manipulation with Macros.....	8
ImagePro Plus Documentation .....	8
Macro Recording .....	8
Additional Talon Ultra Macro Functions .....	8
IpAcqPreAllocate .....	8
IpAcqPreview .....	8
IpAcqSequenceMem .....	9
IpAcqSnapMem.....	9
IpArrayStats.....	10
IpGetPixelValue .....	11
IpMask.....	12
IpSeqAverageFloat.....	13
Appendix A: FITS File Format .....	14
ABSTRACT .....	14
SCOPE.....	14
OTHER REFERENCES .....	14
DETAILS .....	14
FITS FORMAT SUMMARY:.....	15
Appendix B: Talon Ultra Version 1.01 changes.....	16
Current File Versions.....	16
Functional changes from version 1.00 to 1.01.....	16
Functional changes from version 1.01 to 5.23.....	16

# Talon Ultra Data Analysis Software User's Guide

## Features of the Talon Ultra Data Analysis Software (V. 5.23)

The Talon Ultra Data Analysis Software package:

- ◆ supports all Indigo Merlin<sup>®</sup>, Phoenix<sup>™</sup>, and Alpha<sup>™</sup> infrared cameras;
- ◆ acquires image data of consecutive frames at full camera speed – no missed frames;
- ◆ uses computer system memory for data acquisition – acquisition capacity is easily and inexpensively increased;
- ◆ provides full-featured data display, reduction, analysis capability;
- ◆ sports a powerful Visual Basic-compatible macro language for automating data collection, analysis, and reporting;
- ◆ can call and be called by Visual Basic or Visual C++ programs;
- ◆ runs under Microsoft Windows NT 4.0 or Windows 2000 Professional.

## Installation and Setup

Your Talon Ultra system will be delivered with the hardware and software components already integrated into the computer, so you should not have to install any hardware or software or do any additional configuration of the system. Upon unpacking your system, you can proceed to the section “External Connections,” below. However, the next two sections explain instructions for system installation and are included here for reference, in case it ever becomes necessary to re-install all or part of the system.

### Hardware installation

The Talon Ultra system includes a PCI-bus frame grabber card that resides inside the system computer. In a normal Talon system no modifications or setting changes are required on the frame-grabber card as it comes from the factory. Install this card into an available PCI-bus slot as you would any other PC plug-in card – taking the same precautions to power-down and unplug the computer, and use good anti-static handling procedures. As with any other PC peripheral board, be sure to fasten the card into its slot with the bracket screw.

### Software installation

This installation procedure assumes you are starting with a “clean” machine with no Talon software installed. If this is not the case, please contact Indigo customer support for assistance. **You must be logged into an account with “Administrator” privileges during this installation procedure as well as to run and use the Talon software.** It is also **strongly** recommended that you have no other applications running during the install process.

1. Install ImagePro Plus 4.5.1 from the CD-ROM. Follow the instructions in the section titled *Installing Image-Pro Plus* in chapter 1 of the manual *Image-Pro Plus Start-Up Guide*. **Do not** install any video capture drivers from the *Image Capture Drivers* CD provided with Image-Pro Plus. Use the default directory name “C:\IpWin4” for installation of the software. If you have an earlier version of ImagePro, go to [www.mediacy.com](http://www.mediacy.com) to obtain the required update files to get to Version 4.5.1. Be sure you have the software key installed on the parallel port.
2. Install the latest product upgrades, either provided on a separate disk with your Talon Ultra installation, or else available from the Media Cybernetics website at <http://www.mediacy.com/tech/update.htm>. Either way, the upgrades will come in a single file (named **IPP4\_5\_1\_27.EXE** at this writing); execute this file and direct the software to unzip the files to your **C:\IpWin4** directory.
3. Insert the CD marked *Talon Ultra Add-On Install Files* into your CD drive, and run the **BitFlowSDK300.EXE** program. You can do this by browsing to the CD using Windows Explorer, or on your Windows start menu, select “Run” and use the browse button to select and execute this file. Accept the default location for this install, which should be **C:\BitFlow SDK 3.00** (approximately 6 MB required). When prompted, use a serial number of “0” (zero, no quotes) which will install only the binary drivers and not the developer’s source code.
4. After the BitFlow installation program has finished, restart your Windows operating system. This step is necessary for the frame grabber firmware to be correctly initialized. The following steps will not succeed without doing this.
5. When the computer reboots, you may get the “New Hardware Wizard”. Proceed through the wizard, accepting the defaults to find the drivers. When it ask where to look for the drivers, browse to the **C:\BitFlow SDK 3.00\Windows 2000** folder and select the Bitflow.inf file. Finish the wizard and you should not see it again.

6. On the same CD, run the **SelmarIPWinTalon523.EXE** program. The installation program will present instructions for extracting the files to your hard drive. These Talon Add-On files must be installed in the same directory as Image-Pro Plus. By default, this will be the **C:\IpWin4** directory, so you won't have to change this unless you have performed step #1 with a different directory than the default. The program will ask you whether to overwrite some files, and you should respond "Always".
7. Again from the same CD, run the **SelmarCamConfigTalon523.EXE** program. Extract the files to the **C:\BitFlow SDK 3.00\Config** folder.
8. From the **C:\BitFlow SDK 3.00\Bin** directory, run the program **SysReg.exe**.
9. In **SysReg.exe**, accept any defaults that the program offers you. Check to ensure that the Configuration file install directory (e.g. **C:\BitFlow SDK 3.00\Config**) is correctly noted in the box labeled "Camera configuration file path." Now click on the "Board Details" button. If there are any files listed in the "Attached Cameras" box, remove them using the "Remove" button. Then use the "Add" button to add the appropriate file in the **C:\BitFlow SDK 3.00\Config** directory for your camera by referring to Table 1:

Camera Type	File Name
Merlin Mid, Near, or QWIP	ISCMerlin320x256_12low12.cam
Merlin Uncooled	ISCMerlin320x240_12low12.cam
Phoenix™ 320x256	ISCPHXRTIE320x256_16low14.cam
Phoenix™ 640x512, imaging	ISCPHOENIXOEM640x512IImod.cam
Phoenix™ 640x512, nonimaging	ISCPHOENIX640x512.cam
Alpha™ 160x120	ISCALpha160X119.cam
Alpha™ 160x128	ISCALpha160X127.cam
Alpha™ Near 320x256	ISCALphaNIR320x256_12low12.cam

**TABLE 1.**

10. Now click on the **Firmware Details** button. In the MUX AS dropdown box on the Firmware Details dialog box, select "Special 16-bit IR Camera Modes". Click the "Apply" button.
11. Click the "OK" button twice to exit the SysReg.exe program.
12. Now browse to the **C:\IPWin4** folder and use Notepad to open the file "IPWin32.ini". Go to the [Indigo] section at the end of the file and check the "Bypass NUC" setting. It should be set equal to 1 for all cameras except Phoenix 640x512 in imaging mode, which should be set to 0.
13. Now you should do the following to configure the Windows user rights which will optimize the Talon system operation:
  - a) Go to "Start Menu | Settings | Control Panel | Administrative Tools | Local Security Policy | Local Policies | User Rights Assignment".
  - (c) In the Policy panel, scroll down the list to "Increase scheduling priority". If "Administrators" is not listed in the "Local Setting" column, then double click it, then click on "Add" and scroll down to "Administrators", then click OK, and "Administrators" should appear in the "Local Setting" box on the "User Rights Assignment" dialog.
  - (d) Likewise, find "Lock pages in memory", and if "Administrators" is not listed in the "Local Setting" box, then add it as above.
  - (e) Repeat for the "Increase Quotas" right.
  - (f) Repeat for the "Act as part of the operating system" right.
12. Reboot your computer again; it is not necessary to power down again.

### **External Connections**

The frame grabber card has a high-density, 62-pin "D"-style connector for connection to the Camera digital output port. Use the appropriate cable (available from Indigo) to connect between the camera digital output port and this 62-pin "D" connector on the frame grabber board.

### **Quick Check Out**

Your camera should be connected to the frame grabber as explained above, and should be running and imaging normally. Run the Talon Ultra software by clicking on the Image-Pro Plus 4.0 icon on your Windows desktop. From the application main menu, select **Acquire | Live Preview**. You should get a "live mode" window with the IR Image from your camera displayed and updating constantly. If the image is all black, all white, or gray with no contrast, select the **Display Range | Best Fit** menu item and experiment by changing the limits of the display range,

in order to improve the contrast. In addition, you should check in your camera control functions to ensure that the camera is in a 12-bit (14-bit for Phoenix<sup>TM</sup>) output mode (either uncorrected or corrected.) Refer to the camera operator's manual for information on checking and changing these settings.

## Interactive Operation Using the Talon Ultra Menus

### ***ImagePro Plus Documentation***

Refer to the *Image-Pro Plus Start-Up Guide*, which gives a good general overview of the software, as well as some detail about the most often used functions. The *Image-Pro Plus Reference Guide* goes into more detail and gives an item-by-item explanation of most of the features and functions in the software.

Most of the items in the **Acquire** menu mentioned in the *Start-Up Guide* and the *Reference Guide* have been changed and customized in the Talon Ultra software package, so refer to the next section, *Additional Features*, for information on data acquisition.

### ***Additional Commands on the Acquire Menu***

#### Camera Preview

The **Acquire | Camera Preview** displays a “live” image from the camera in a window. To Use: After you select this menu item, Talon displays the **Preview Camera Data** dialog box. The only thing to do here is click on the **Start Preview** button. The button will then change to **Stop Preview**. Result: an image workspace will be displayed, with the live image in it. There is no Macro Language equivalent for this function.

If the contrast in the displayed image is not adequate, select the **Enhance | Display Range** menu item and experiment by changing the limits of the display range, in order to improve the contrast.

#### Image Acquire

The **Acquire | Image Acquire** menu item displays the dialog box for performing single-frame acquisitions, or “snaps.” To Use: each click of the **Acquire** button in this dialog box will cause one frame to be acquired and displayed in a workspace. Macro Language equivalent: **IpAcqSnapMem( )**

#### Memory Sequence

The **Acquire | Memory Sequence** menu item displays the dialog box for performing multiple-frame (sequence) acquisitions. To Use: set the number of frames to be acquired in the “Frames:” field. Click the **Acquire** button, and a sequence will be acquired and displayed in a workspace.

Macro Language equivalent: **IpAcqSequenceMem( )**.

### ***Additional Commands on the TalonUtils Menu***

#### Average Sequence

The **TalonUtils | Average Sequence** menu item performs a mathematical average and standard deviation calculation on a sequence. To Use: The sequence to be averaged should be the active sequence; to activate a sequence, click one time on it. Then select the **TalonUtils | Average Sequence** menu item. Result: The software will calculate and display two new images, one titled “MEAN” and the other titled “SDEV”, for the mean and standard deviation, respectively. These resulting images are in the floating point class.

Macro language equivalent: **IpSeqAverageFloat( )**

This command differs from the existing **Acquire | Sequence Tools | Average** menu item in two ways: (1) it calculates both the mean and the standard deviation for each pixel (the other function calculates only the mean), and (2) it creates results in the floating-point image class (the other function gives an integer class result), minimizing round-off error in the calculations. These enhancements are important for analysis of both temporal and spatial noise.

This command is much faster than performing such an average frame-by-frame in a macro. On a 450 MHz Pentium III machine, this function averages a 700-frame sequence in around 9 seconds.

## Array Stats

The **TalonUtils | Array Stats** menu item performs a statistical analysis on the population of pixels in an image (it cannot be performed on a sequence.) To Use: The image to be analyzed should be the active image; to activate an image, click one time on it. Then select the **TalonUtils | Array Stats** menu item. Result: The software will display a message box with the following statistics listed:

<b>Mean:</b>	the statistical average of all pixel intensity values in the population
<b>Sdev:</b>	the statistical standard deviation of all pixel intensity values in the population
<b>Min:</b>	the lowest pixel intensity value in the population
<b>Max:</b>	the highest pixel intensity value in the population
<b>Incl:</b>	the number of pixels in the image, which are included in the population
<b>Excl:</b>	the number of pixels in the image, which are excluded from the population (due to masking)

This command displays two columns of numbers: the first column uses the population of all pixels in the image, the second column uses only those pixels which are not “masked”. Since there is no provision for masking with this version of the command, no pixels will be masked, and the two columns will always be identical. See the next command to perform masking. Macro language equivalent: **IpArrayStats ( )**

## Array Stats w/ Mask

The **TalonUtils | Array Stats w/Mask** menu item performs the same statistical analysis as above, but with the additional step that it queries the operator for a “mask” image (of the GrayScale8 image class) to be used to exclude “bad” pixels from the second column statistical calculations. To Use: The image to be analyzed should be the active image; to activate an image, click one time on it. Then select the **TalonUtils | Array Stats w/ Mask** menu item. The program will display a message indicating that the operator needs to click on the mask image. Result: The software will display a message box with the same statistics listed as described above, except that the second column will now most likely be different, since only “good” pixels will be included in its population. Macro language equivalent: **IpArrayStats ( )**

## ***Additional Command on the Help Menu***

### About Driver Versions

The **Help | About Driver Versions** menu item displays a message indicating the version level of several of the installed utility files. This information should be reported to Indigo customer support for assistance in troubleshooting any problems with the Talon software.

## Data Acquisition and Manipulation with Macros

### *ImagePro Plus Documentation*

Refer to the *Auto-Pro Guide*, which gives a general overview of writing and using macros, as well as a detailed listing of most of the function calls available in the macro language.

All of the **IpAcq...** functions listed in the *Auto-Pro Guide* have been disabled and replaced with customized versions for the Talon Ultra data analysis software package. The new versions, as well as additional features to further extend Talon's capabilities, are explained in the following section.

### **Macro Recording**

The additional macro functions listed here are all integrated into the Talon Ultra software package so that if you are recording a macro and you perform one of these functions interactively from the user interface, the appropriate function syntax will be inserted in the macro you are recording. Sometimes recording a macro can be a fast and efficient way to set down the basic structure and syntax of a procedure you are trying to automate – then you can edit and “finish” the macro in the macro editor to complete its function. For detailed information on recording macros, see the chapter entitled “*The Macro Menu*” in the *Image-Pro Plus Reference Guide*.

### **Additional Talon Ultra Macro Functions**

#### **IpAcqPreAllocate**

**Syntax:** IpAcqPreAllocate(NumFrames, Cmd) As Integer

**Description:** Reserves or releases memory for a sequence of frames

**Parameters:** NumFrames Integer size of sequence, in frames, to preallocate  
 Cmd Integer ALLOCATE : allocate memory  
 DEALLOCATE: release allocated memory

**Return Value:** On success, returns 0. On failure, returns a non-zero error code:

- 1 = internal error getting application object
- 2 = error initializing frame grabber hardware
- 3 = error allocating sequence memory (on ALLOCATE)
- 4 = error releasing sequence memory (on DEALLOCATE)
- 5 = unknown “Cmd” value

**Example:**

```
ret = IpAcqPreAllocate(32,ALLOCATE)
      \ allocates memory for 32 frame sequence
ret = IpAcqPreAllocate(32,DEALLOCATE)
      \ releases that memory
```

**Comments:** This function can be used to determine whether sufficient system memory exists to acquire a given number of frames, without having to actually attempt the acquisition. It is not necessary to use this function; performing an IpAcqSequenceMem() or IpAcqSnapMem() directly will automatically allocate sufficient memory for the commanded acquisition.

However, this function might be useful in critical situations where it is important to know for sure that sufficient memory is available for a large acquisition, before committing to the acquisition.

#### **IpAcqPreview**

**Syntax:** IpAcqPreview(Cmd) As Integer

**Description:** Starts or stops the live-mode preview under macro control

**Parameters:** Cmd Integer PREV\_START (=1): start a live-mode preview  
 PREV\_STOP (=0): stop a live-mode window

**Return Value:** Returns 0

**Comments:** This function can be used to determine whether sufficient system memory exists to acquire a given number of frames, without having to actually attempt the acquisition. It is not necessary to use this function; performing an IpAcqSequenceMem() or IpAcqSnapMem() directly will automatically allocate sufficient memory for the commanded acquisition.

However, this function might be useful in critical situations where it is important to know for sure that sufficient memory is available for a large acquisition, before committing to the acquisition.

## IpAcqSequenceMem

**Syntax:** IpAcqSequenceMem (NumFrames, Holdoff, Skip, TriggerType) As Integer

**Description:** Acquires a sequence of frames

**Parameters:**

NumFrames	Integer	size of sequence to acquire, in frames
Holdoff	Integer	delay before start of acquisition, in frames
Skip	Integer	delay between each acquired frame, in frames
TriggerType	Integer	TRIG_NONE (=0) – acquisition starts immediately TRIG_ONCE (=1) – acquisition starts after one trigger TRIG_EACH (=2) – each trigger acquires one frame

**Return Value:** On success, returns the Document ID of the acquired sequence (a non-negative number.) On failure, returns a negative value as an error code:

- 1 = internal error getting application object
- 2 = error initializing frame grabber hardware or acquiring sequence
- 3 = error determining Document ID of acquired sequence

**Example:**

```
Dim MySequence As Integer
MySequence = IpAcqSequenceMem(32, 0, 0, TRIG_NONE)
  ` acquires 32 consecutive frames with no triggering
```

**Comments:**

## IpAcqSnapMem

**Syntax:** IpAcqSnapMem(Dest) As Integer

**Description:** Acquires a single frame

**Parameters:**

Dest	Integer	ACQ_NEW (=1) – create a new image with acquired data ACQ_CURRENT (=2) – not supported; do not use
------	---------	--

**Return Value:** On success, returns the Document ID of the acquired image (a non-negative number.) On failure, returns a negative value as an error code:

- 1 = internal error getting application object
- 2 = error initializing frame grabber hardware
- 3 = general error acquiring image
- 4 = error determining Document ID of acquired image

**Example:**

```
Dim MyImage As Integer
MyImage = IpAcqSnapMem(ACQ_NEW)  ` snaps one image
```

**Comments:** At this time, the ACQ\_CURRENT option for the Dest parameter is not available, so Dest must be set to "ACQ\_NEW."

---

## IpArrayStats

**Syntax:** IpArrayStats(docId, maskId, statsArray(0)) As Integer

**Description:** Calculates population statistics for an image

**Parameters:**

docId	Integer	Document ID of image to be analyzed
maskId	Integer	Document ID of Grayscale8 mask image denoting bad pixels; this value may be negative, if so, masking is not performed
statsArray	Single(12)	array of 12 Singles to return results; you must pass this array as a reference to its first element; see comments for array layout

**Return Value:** On success, returns 0. On failure, returns a negative value as an error code:

- 1 = internal error getting image VRI
- 2 = internal error getting mask image VRI
- 3 = internal error checking number of frames parameter
- 4 = number of frames is greater than 1; sequence statistics not supported
- 5 = internal error checking size of image
- 6 = internal error checking size of mask image
- 7 = image and mask image sizes do not match
- 8 = internal error checking class of image
- 9 = class of image is not supported; must be GrayScale8, 12, 16, or Float
- 10 = internal error checking class of mask image
- 11 = class of mask image not supported; must be GrayScale8
- 12 = internal error getting data pointer for image
- 13 = internal error getting data pointer for mask image

**Example:**

```
Dim stats(12) As Single
ret = IpArrayStats(MyImage, MyMask, stats(0) )
Debug.Print "Mean value of all pixels is: ";stats(0)
Debug.Print "Mean value of good pixels only is: "; stats(1)
```

**Comments:**

The “even” indices from 0 to 10 in the statsArray represent the Mean, Standard Deviation, Minimum, Maximum, and number of good and bad pixels in the entire image without a mask applied. The “bad” pixels in this case (index 10) is always zero. The “odd” indices from 1 to 11 in the stats array represent the same statistics for the image with the mask applied, i.e. for just the population of “good” pixels. The values in statsArray are as follows:

- StatsArray(0)** = Mean value, all pixels in image
- StatsArray(1)** = Mean value, good pixels only, as denoted in mask image
- StatsArray(2)** = Standard Deviation, all pixels in image
- StatsArray(3)** = Standard Deviation, good pixels only
- StatsArray(4)** = Minimum pixel intensity value, all pixels in image
- StatsArray(5)** = Minimum pixel intensity value, good pixels only
- StatsArray(6)** = Maximum pixel intensity value, all pixels in image
- StatsArray(7)** = Maximum pixel intensity value, good pixels only
- StatsArray(8)** = Number of pixels in image
- StatsArray(9)** = Number of good pixels as denoted in mask
- StatsArray(10)** = always zero (i.e. “bad” pixels in an unmasked image is zero)
- StatsArray(11)** = Number of bad pixels as denoted in mask

---

## IpGetPixelValue

**Syntax:** IpGetPixelValue (docId, XLoc, YLoc, ByRef OutValInt, ByRef OutValFloat) As Integer

**Description:** Returns the pixel value at a specified (x,y) location in a specified image.

**Parameters:**

docId	Integer	Document ID of image to be analyzed; can be DOCSEL_ACTIVE to analyze the active image
XLoc	Integer	x-coordinate of pixel to be returned; count from zero at left edge
YLoc	Integer	y-coordinate of pixel to be returned; count from zero at top edge
OutValInt	Integer ref	variable which will receive the pixel intensity as a rounded integer
OutValFloat	Single ref	variable which will receive the pixel intensity as floating point

**Return Value:** On success, returns 0. On failure, returns a non-zero value as an error code:

- 1 = internal error getting image VRI
- 5 = internal error checking size of image
- 8 = internal error checking class of image
- 9 = class of image is not supported; must be GrayScale8, 12, 16, or Float
- 12 = internal error getting data pointer for image

**Example:**

```
Dim MyInt As Integer, MyFloat As Single
` get intensity of upper left corner pixel
ret = IpGetPixelValue (DOCSEL_ACTIVE, 0, 0, OutValInt, OutValFloat)
```

**Comments:** This function provides a quick way to pull a single pixel value from a specific location in an image.

If the image is an integer type, (8, 12, or 16-bit) the value returned in OutValInt will be an exact representation of the pixel intensity. This value will also be converted to floating point representation and placed in the OutValFloat variable.

If the image is a floating point type, the OutValFloat will be the exact representation of the pixel intensity stored in the image, and additionally, this value will be rounded to an integer and stored in the OutValInt variable.

---

## IpMask

**Syntax:** IpMask(docId, maskId, cmd, limitA, limitB) As Integer

**Description:** Adds "bad" pixels to a mask array

<b>Parameters:</b>	docId	Integer	Document ID of image to be analyzed; do not use DOCSEL_ACTIVE; image may be GrayScale, GrayScale12, GaryScale 16 or Floating Point class
	maskId	Integer	Document ID of mask image to add bad pixels to. If maskId is negative, this function will create a new mask
	cmd	Integer	MASK_LESS_THAN – pixels less than limitA are marked bad MASK_GREATER_THAN – pixels greater than limitA are marked bad MASK_INSIDE_RANGE – pixels within the non-inclusive range between limitA and limitB are marked bad MASK_OUTSIDE_RANGE – pixels less than limitA and pixels greater than limitB are all marked bad MASK_NO_ACTION – no pixels are marked bad (used to create an "empty" mask of appropriate size to match a given image)
	limitA	Single	limit value, used as noted above
	limitB	Single	limit value, used only with inside and outside masking, as above

**Return Value:** On success, returns the Document ID of the mask (a non-negative number.) On failure, returns a negative value as an error code:

- 1 = internal error getting image VRI
- 2 = internal error getting mask image VRI
- 3 = internal error checking number of frames parameter
- 4 = number of frames is greater than 1; cannot mask a sequence
- 5 = internal error checking size of image
- 6 = internal error checking size of mask image
- 7 = image and mask image sizes do not match
- 8 = internal error checking class of image
- 9 = class of image is not supported; must be GrayScale8, 12, 16, or Float
- 10 = internal error checking class of mask image
- 11 = class of mask image not supported; must be GrayScale8
- 12 = internal error getting data pointer for image
- 13 = internal error getting data pointer for mask image

**Example:**

```
MyMask1 = IpMask(MyImage1, -1, MASK_NO_ACTION, 0.0, 0.0)
  ` creates a new empty mask
MyMask2 = IpMask(MyImage1, -1, MASK_LESS_THAN, 0.001, 0.0)
  ` creates new mask and marks all pixels less than 0.001 as "bad"
  ` note that this guarantees pixels equal to 0.000 will be marked
ret = IpMask(MyImage2, MyMask2, MASK_GREATER_THAN, 10.0, 0.0)
  ` adds bad pixels (greater than 10.0) to the existing mask
```

**Comments:** The limitB value is only used for the MASK\_INSIDE\_RANGE and MASK\_OUTSIDE\_RANGE operations. For the MASK\_LESS\_THAN and MASK\_GREATER\_THAN operations, only the limitA value is used, and for the MASK\_NO\_ACTION operation, both limit values are ignored. All comparisons are done with "less-than" and "greater-than" operators. Pixels exactly equal to a limit value will not be marked bad. For example, in the second mask operation in the Example section above, if you want to mask out pixels less than or equal to zero, you should set the limit to a value incrementally larger than zero to ensure that "zero" pixels will be marked bad.

---

**IpSeqAverageFloat****Syntax:** IpSeqAverageFloat (docId, avgId, sdevId) As Integer**Description:** Calculates pixel averages and standard deviations across a sequence.

**Parameters:**

docId	Integer	Document ID of sequence to be analyzed; can be DOCSEL_ACTIVE to analyze the active sequence
avgId	Integer ref	variable which will receive the DocumentID of the Mean image, which will be Floating Point class
sdevId	Integer ref	variable which will receive the DocumentID of the Standard Deviation image, which will be Floating Point class

**Return Value:** On success, returns 0. On failure, returns a non-zero value as an error code:

- 1 = internal error getting image VRI
- 2 = internal error checking number of frames parameter
- 3 = number of frames is 1; cannot process a single image
- 5 = internal error checking size of image
- 6 = internal error checking class of image
- 7 = class of image is not supported; must be GrayScale8, 12, or 16
- 8 = internal error getting data pointer for image
- 9 = memory allocation problem for "sum" array
- 10 = memory allocation problem for "sum of squares" array
- 20 = internal error getting data pointer for Floating Point "Mean" image
- 21 = internal error getting data pointer for Floating Point "Sdev" image

**Example:**

```
Dim MySeq As Integer, AvgImg As Integer, SdevImg As Integer
MySeq = IpAcqSequenceMem(32, 0, 0, TRIG_NONE)
ret = IpSeqAverageFloat(MySeq, AvgImg, SdevImg)
ret = IpAppSelectDoc(AvgImg)  ` make "mean" result the active image
```

**Comments:** This function extends the capabilities of the **IpSeqAverage** macro documented in the *Auto-Pro Guide*. That function only calculates the mean for each pixel (not the standard deviation) and produces an integer result.

The "MEAN" and "SDEV" images produced by this function, **IpSeqAverageFloat**, give the results as floating point values. This is important in noise analysis, as it avoids significant round-off error that occurs when the average calculation is performed in integer math.

## Appendix A: FITS File Format

### ABSTRACT

The file format most commonly used for storage of infrared images and sequences in Talon is the FITS file format. The FITS file format, administered by NASA, is a public specification primarily used by astronomers and scientific users who need flexible, platform-independent data transfer, including the capability to handle images in 8, 16 and 32-bit integer format, 32-bit floating-point format, and multi-frame image "sequences". This document is a brief overview, describing the basics of reading the raw image data from Talon image files.

### SCOPE

This document IS intended to:

- ◆ provide a programmer of modest capabilities sufficient information to read raw image data from FITS files with images from Talon.

This document IS NOT intended to:

- ◆ provide a complete and detailed description of the FITS format, e.g. for writing FITS files;
- ◆ provide algorithms for performing thermographic calibration on infrared images.

### OTHER REFERENCES

The complete document describing the FITS file format is available on the Internet, from the NASA Office of Standards and Technology (NOST), in either PostScript, Latex, raw text, or PDF formats. Start at the page of the NASA FITS Support Office Home Page on the World Wide Web at:

[http://fits.gsfc.nasa.gov/fits\\_home.html](http://fits.gsfc.nasa.gov/fits_home.html)

Although version 2.0 of the FITS standard has been approved and is now current, the Talon Software packages currently implement version 1.1 of the FITS standard. The only significant difference likely to ever affect Talon users is that version 1.1 calls for 2-digit year representation in the DATE-OBS field. The 2.0 standard specifies that 2-digit year representation is allowed through the end of the year 1999, but that data collected after the beginning of the year 2000 should have a 4-digit year in the DATE-OBS field. *At this time, Talon Light and Talon Ultra do not write this optional field to the FITS header anyway, but if future releases do so, they will adhere to the updated FITS version 2.0 specification.*

### DETAILS

The details described here are the minimum required to understand Talon FTS image files.

FITS file headers consist of multiple "card images" of 80 ASCII characters. There are no "end of line" characters or delimiters between card images. A header consists of groups of 36 card images, thus a header will be a multiple of 2880 bytes in length (36 card images \* 80 characters.) To date, Talon FTS file have only one group of 36 card images (i.e. 2880 bytes). (*The FITS standard allows for "n" groups of 36 card images, so a general FTS reader would not count on the header being 2880 bytes long.*)

Each card image consists of a "keyword" of up to 8 characters, usually followed by an equals sign ("=") and a parameter, all as ASCII text. The first card images in a Talon FITS file will always appear in the following order, and might look like this:

```

SIMPLE =          T          / this is the FTS file signature
BITPIX =         16         / will be 16, for Merlin cameras
NAXIS  =          3         / 3-dimensional data - implies a sequence of images
NAXIS1 =         320        / size of each axis in pixels
NAXIS2 =         240
NAXIS3 =          16        / number of frames in sequence

```

One of these card images which might change between different Talon FITS files is the "BITPIX" parameter, which can be 8, 16, or -32 depending upon how the user stored the image data. If the BITPIX parameter is 8, then each pixel in the image is represented as an unsigned 8-bit byte.

With BITPIX = 16, each pixel is represented as a 16-bit signed integer (-32768 to 32767). Since the acquired data is actually created as an unsigned integer (0 to 65535), an offset of -32768 is applied to the acquired data before storing it in the file, so that the entire 16-bit range can still be stored. One easy way to reverse this process upon reading a file is to read each 16-bit pixel from the file into an unsigned 16-bit integer variable, but simply invert the Most-Significant-Bit of each pixel before processing any data. This reverses the -32768 offset, returning the pixels back to the range of 0 to 65535.

If BITPIX is -32, then each pixel is represented as a 4-byte IEEE floating point value. Although the FITS format allows for other data formats, these are the only ones currently used by Talon Ultra.

The binary image data is stored immediately following the last byte of the header. There is no compression applied to the image data. Image data is stored in "row-major" order. In other words, all pixels of the first row are stored first, followed by the pixels of the second row, etc.... Images will always be rectangular, of size specified by the NAXIS1 ("X") and NAXIS2 ("Y") header parameters. There is no delimiter or padding between rows in the binary data; the first pixel of row "n+1" immediately follows the last pixel of row "n".

If the NAXIS parameter is "3", there will be another parameter "NAXIS3" which will indicate that the FITS file contains a sequence consisting of multiple images. All images are of the same size. Binary data for each subsequent image in the sequence follows immediately after the previous one, with no delimiters or padding between images.

If the NAXIS parameter is "2", then there will be only one image in the file, and there will be no NAXIS3 parameter.

According to the FITS standard, the end of the FITS file should be padded with zero data as required to make the length of the entire file a multiple of 2880 bytes. The Talon FITS implementation may not adhere to this requirement, but this should not cause problems for a FITS reader.

**IMPORTANT NOTE:** Because the FITS standard was intended to be platform independent, it was necessary to choose the ordering of bytes in multiple-byte binary values. The choice made was the "big-endian" format, also sometimes referred to as "Motorola format." This means that the most significant byte of each binary pixel value is stored first in the file, followed by the less significant byte or bytes. Although this does not affect 8-bit files, it is important in the interpretation of 16-bit (integer) and 32-bit (floating point) files, especially if they are being read on a PC Compatible computer, which generally uses the opposite convention. In this case, remember to reverse the byte order of each pixel value after reading the image data.

## FITS FORMAT SUMMARY:

- ◆ FITS is a format used in the astronomical and scientific community for exchange of data
- ◆ FITS is used by Indigo's Talon Light and Talon Ultra software for image data
- ◆ Talon **header length is 2880 bytes**: 36 strings of 80 ASCII characters each, no delimiters
- ◆ Talon FITS files may be NAXIS=2 (single image) or NAXIS=3 (sequence)
- ◆ **NAXIS1 and NAXIS2 are the image size** (x,y)
- ◆ **NAXIS3 is the number of images** in a sequence (if present)
- ◆ **data is stored in raw binary format** with no compression of image data
- ◆ binary image **data is stored in "big-endian" or "Motorola" format** (MSB first)
- ◆ image **data is stored in row-major order** immediately following header
- ◆ no delimiters or padding between rows of image data, or between images in a sequence
- ◆ generally, FITS supports many binary data formats (8, 16, or 32-bit integer, 32 or 64 bit floating point)
- ◆ specifically, **Talon Ultra images will either be BITPIX=8 (8-bit unsigned integer), BITPIX=16 (16-bit signed integer) or BITPIX=-32 (4-byte IEEE floating point).**
- ◆ **BITPIX=16 data is stored with -32768 offset** to fit into "signed" integer range of -32768 to 32767
- ◆ on reading BITPIX=16 data, invert MS Bit of each pixel value to return to "unsigned" range of 0 to 65535

## Appendix B: Talon Ultra Version 1.01 changes

### Current File Versions

This release assumes the use of Image-Pro Plus version 4.1 as the baseline installation, and upgrades from the Media Cybernetics website (as described in the **Installation and Setup** chapter of this manual) to at least Image-Pro Plus version 4.1.0.9. This release continues to use the Bitflow low-level driver version 1.20, as did Talon Ultra 1.00. Version numbers of specific support files, as found on the **Help | About Driver Versions** menu item, are:

◆ IPWin32.EXE	4,1,0,9
◆ STCapt32.DLL	1,0,3,0
◆ CaptRRNT.DLL	4,1,0,0
◆ STIUtils.DLL	1,0,1,0
◆ IPFile32.DLL	4,1,0,4
◆ IFFFTS32.DLL	not available in this release
◆ R2D.DLL	not available in this release
◆ BFD.DLL	1,0,0,1

### Functional changes from version 1.00 to 1.01

- ◆ Cleaned up several memory allocation inefficiencies associated with sequences.
- ◆ Talon Ultra now sets the "dirty" flag on acquired sequences to force a "Save?" confirmation message if you try to close before saving.
- ◆ Added macro function `IpAcqPreview()`.
- ◆ Added macro function `IpGetPixelValue()`.
- ◆ Updated mid-level acquisition driver to fix inconsistency on fast CPU machines when doing non-zero frame skip.
- ◆ Included new camera configuration files (for shifting Merlin<sup>®</sup> and Phoenix<sup>™</sup> data down to lower bits.)
- ◆ Enhanced error messages in some locations.
- ◆ Menu item **Help | About Driver Versions** now gives dynamic version numbers of several specific component files including `IPWin32.Exe`, `STCapt32.dll`, `CaptRRNT.DLL`, `STIUtils.DLL`, `IFFFTS.DLL`, `R2D.DLL`, and `BFD.DLL`. This is a much more useful data set for problem debug and customer support.
- ◆ Changed **Acquire** menu to get rid of vestigial **Video** and **Setup** functions inappropriate to Talon drivers.
- ◆ Changed **STIUtils** menu to **TalonUtils**, and changed other "STI" references to "Talon" for product consistency.
- ◆ Removed **TalonUtils | About**; replaced it with **Help | About Driver Versions** described above.
- ◆ Program name on Caption bar of main window will now read **Talon Ultra 1.01**.

### Functional changes from version 1.01 to 5.23

- ◆ The main change in this version is Windows 2000 Professional compatibility.
- ◆ This version uses new BitFlow drivers which improve functionality and eliminates software crashes caused by no data coming into the board.
- ◆ For other changes, please refer to the Release Notes in the C:\IPWin4\Ultra Docs folder.