ARTISAN®
TECHNOLOGY GROUP

Your **definitive** source for quality pre-owned equipment.

Artisan Technology Group
(217) 352-9330 | sales@artisantg.com | artisantg.com

Full-service, independent repair center
with experienced engineers and technicians on staff.

We buy your excess, underutilized, and idle equipment
along with credit for buybacks and trade-ins.

Custom engineering
so your equipment works exactly as you specify.

- Critical and expedited services
- In stock / Ready-to-ship
- Leasing / Rentals / Demos
- ITAR-certified secure asset solutions

Expert team | Trust guarantee | 100% satisfaction

All trademarks, brand names, and brands appearing herein are the property of their respective owners.

Find the *Teledyne Dalsa / Coreco Imaging IC-PCI* at our website: *Click HERE*

IC-PCI™

IC-PCI™

Software Manual

47-S60002-05

February 2001

this version equals revision 04b found on the CD-ROM

CORECO IMAGING

Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com

IC-PCI Software Manual

Document Number 47-S60002-05

Revision 05; February 2001

Released March 1995

Revised November 1996, February 1997, November 1997,
November 1998, January 2000.

Copyright © Coreco Imaging, Inc 2001
Copyright © transferred to Coreco Imaging, Inc 2000
Copyright © Imaging Technology Incorporated 2000–1994
All rights reserved.
Printed in the United States of America.

All copyrights in this manual, and the hardware and software described in it, are the exclusive property of Coreco Imaging, Inc and its licensors. Claim of copyright does not imply waiver of Coreco Imaging, Inc's or its licensor's other rights in the work. See the following Notice of Proprietary Rights.

NOTICE OF PROPRIETARY RIGHTS

This manual and the related hardware and software are confidential trade secrets and the property of Imaging Technology and its licensors. Use, examination, reproduction, copying, transfer and/or disclosure to others of all or any part of this manual and the related documentation are prohibited except with the express written consent of Coreco Imaging, Inc.

Portions of the software library use the Halo Image File Format Library, which is a copyrighted product of Media Cybernetics.

The information in this document is subject to change without notice. Coreco Imaging, Inc makes no representations or warranties with respect to the contents of this manual and specifically disclaims any implied warranties of merchantability or fitness for a particular purpose. Coreco Imaging, Inc assumes no responsibility for errors or omissions in this document.

IFC-SDK, PC-DIG, PC-RGB, PCVision, Sherlock, SMART Search, and the Coreco Imaging logo are trademarks of Coreco Imaging, Inc.

SherlockPro, Prophecy, MVTools, ITEX, Camera Configurator, and PCVision*plus* are Registered Trademarks of Coreco Imaging, Inc.

All other trademarks are the property of their respective owners.

Coreco Imaging, Inc
(formerly known as Imaging Technology Incorporated
55 Middlesex Turnpike
Bedford, MA 01730–1421
Phone: +1.781.275.2700
FAX: +1.781.275.9590
Sales Email: info@corecoimaging.com
http://www.corecoimaging.com

Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com

PREFACE

ITEX is the library of control functions for image processing hardware from Imaging Technology Incorporated. This manual describes the IC-PCI functions; including calling sequences, functions performed, and examples of code containing the calls.

- Chapter 1, “Introduction” contains general information about the IC-PCI, including software conventions and restrictions.
- Chapter 2, “IC-PCI Functions” describes all the IC-PCI functions in alphabetical order.
- Appendix A, “Registers” summarizes the IC-PCI control registers.
- Appendix B, “Configuration Files” discusses the contents of configuration files.

NOTE *Chapter 1 of this manual contains tables listing IC-PCI routines by* functional groups, *and gives page references. Chapter 2 presents the routines in* alphabetical order. *If you are more used to functional groupings, please consult Chapter 1.*

Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com

DOCUMENT CONVENTIONS

In this manual, the following conventions describe each function:

Returned Values

This paragraph describes the value returned with the symbol INQUIRE or with normal operation. This paragraph also describes the values returned because of error conditions, most often incorrect parameter values.

Hardware

This paragraph lists the registers or bit-fields changed by a function. If a bit-field is changed, the register that contains the bit-field may be listed in parentheses. The hardware manual contains more information about registers and bits.

Constants defined in a header file are shown in uppercase; for example, ON. The header file names are enclosed in quotes; for example, “amclr.h.”

Examples

Program examples throughout this book use an ellipsis (...) to indicate that insignificant code has been removed from the example or the code fragment, for example:

```
...
BYTE pixarray[512];
...
  pci_rhbline(ADDRESS,200,1,pixarray);
...
```

Definitions

AOI	A hardware timing format, optimized for digital data processing. AOI timing is faster than camera or display timing. Functions like *x*_**snap**, *x*_**grab** usually use camera timing. Functions like *x*_**aoi_snap**, *x*_**aoi_grab** use AOI timing, usually for board to board processing or transfer.
ROI	A software data structure, used as a location (source or destination) for image data. An ROI can be located in any useable system resource: camera, display monitor, image memory, system memory, or system disk.

Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com

Table of Contents

Document Conventions iv

Chapter 1. Introduction

Hardware Overview 1-1
Image Memory 1-2
Linear Memory 1-2
“IC-VL Compatible Mode” 1-3
Acquisition Control 1-3
Color Storage 1-4
Bus Master Operation 1-4
Software Introduction 1-4
Argument Types 1-5
IC-PCI Selection 1-5
Function Levels 1-5
Returned Values 1-6
Interrupt Events 1-7
Initialization Functions 1-8
Acquisition Functions 1-9
High Level Acquisition Functions 1-10
Read and Write Functions 1-10
Image Read and Write Functions 1-11
Area Functions 1-11
Module Structure Functions 1-13
DOS Display Functions 1-15

Chapter 2. IC-PCI Functions

icp_acq_addr_status Return the Current Acquisition IC-PCI Memory Block 2-2
icp_acqbits Set or Return the Acquisition Command Bits 2-2
icp_acq_fstart– Return the Starting Field Status 2-3
icp_acq_in_progress Test for Acquisition in Progress 2-3
icp_acq_pending Test for Acquistion Command Pending 2-4
icp_amtrig_enable Enable the Trigger Input from the Acquisition Module 2-4
icp_bitblt Transfer one Frame to VGA (DOS only) 2-5
icp_bitblt_end Stop Direct Transfer to VGA (DOS only) 2-6
icp_bitblt_start Set Up for Direct Transfer from IC-PCI to VGA (DOS only) 2-6
icp_bm_count Set or Return the Bus-Master-Mode Transfer Count 2-7

Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com

icp_bm_done Test for Bus–Master–Mode Transfer Done 2-8
icp_bm_fifo_4P Test FIFO Contains 4 or More DWORDS Ready for Transfer 2-8
icp_bm_fifo_empty Test for FIFO Empty 2-8
icp_bm_fifo_full Test for Bus-Master-Mode FIFO Full 2-9
icp_bm_ilace Set or Return the Bus-Master-Mode Interlace Mode 2-9
icp_bm_lock Lock Regions for Bus Master 2-10
icp_bm_mode Set or Return the Bus Master Enable Flag 2-11
icp_bm_read Read into Locked Region 2-12
icp_bm_read_subregion Read Area into a Locked Region 2-12
icp_bm_unlock Unlock Regions for Bus Master 2-13
icp_bus_request Set or Return the PCI-bus Request 2-14
icp_chbline Clear a Horizontal Line of BYTE Values 2-15
icp_chdwline Clear a Horizontal Line of DWORD Values 2-15
icp_chwline Clear a Horizontal Line of WORD Values 2-16
icp_clr_area Clear an Area in IC-PCI Memory 2-17
icp_clr_frame Clear a Frame in IC-PCI Memory 2-18
icp_clr_roi Clear a ROI 2-18
icp_cp_area Copy an Area in IC-PCI Memory 2-19
icp_cp_roi Copy an ROI to another ROI 2-20
icp_create_frame Create a Frame Buffer 2-21
icp_create_roi Create a ROI Structure 2-22
icp_create_seq_frames Create a Sequential Frames Buffer 2-23
icp_cvbline Clear a Vertical Line of BYTE Values 2-24
icp_cvdwline Clear a Vertical Line of DWORD Values 2-25
icp_cvwline Clear a Vertical Line of WORD Values 2-25
icp_delete_all_frame_rois Delete All ROIs in a Frame Buffer 2-26
icp_delete_all_frames Delete All Frame Buffers and their ROIs 2-26
icp_delete_frame Delete a Frame Buffer 2-27
icp_delete_roi Delete a ROI Structure 2-27
icp_delete_seq_frame Delete a Sequential Frames Buffer 2-28
icp_display_area Define an Area for Display 2-28
icp_display_frame Define a Frame for Display 2-29
icp_display_roi Define a ROI for Display 2-30
icp_display_roi_area Define a ROI Area for Display 2-30
icp_dregs Display the IC-PCI Register Contents 2-31
icp_dump_mem Display (dump) an Area of Memory 2-31
icp_dump_roi_mem Display (dump) a Memory ROI 2-32
icp_fcount Set or Return the Acquisition Frame Count 2-32

Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com

icp_field_status Return the Camera Field Status 2-33
icp_fifo_reset Reset (Clear) the FIFO 2-34
icp_frame_attrib Read Attributes of a Frame Buffer 2-34
icp_frame_exist Test if a Frame Buffer Already Exists 2-35
icp_freeze Stop Acquisition 2-35
icp_get_acq_dim Return Dimensions of Camera Frame 2-36
icp_get_acq_start_addr Return the Camera Acquire Start Address 2-36
icp_get_active_ubm_frame Return a Pointer to the Active “Unattended Bus-Mastered” Frame . . 2-37
icp_get_bm_aoix Return the Bus-Master-Mode AOI Horizontal Size 2-38
icp_get_bm_dst_addr Return the Bus-Master-Mode Destination Address 2-38
icp_get_bm_firstf_start Return the First Field Starting Address of Bus-Master Transfer 2-39
icp_get_bm_secondf_start Return the Second Field Starting Address of Bus-Master Transfer . . . 2-39
icp_get_hw_revision Get the Hardware Revision ID 2-40
icp_get_idle_ubm_frame Return a Pointer to the Idle “Unattended Bus-Mastered” Frame 2-40
icp_get_slot0 Return Pointer to the AM Module Structure 2-41
icp_grab Acquire Images Continuously 2-41
icp_host_large_format_acquire Acquire Large Format Image(s) to Host Memory 2-42
icp_host_large_format_wacquire Acquire a Large Format Image(s) to Host Memory, with Wait . 2-43
icp_image_pitch Set or Return the Image Pitch for Bus-Master Transfers 2-44
icp_iregs Initialize the IC-PCI Registers 2-44
icp_large_format_snap Acquire Single Large- Format Image into Host Memory 2-45
icp_large_format_trig_snap Trigger-Driven Large- Format Image Acquisition into Host Memory 2-46
icp_latency_timer Set or Return the Bus-Master Latency Timer 2-46
icp_line_reverse_read Read an Area in Reverse Line Order 2-47
icp_mailbox_read Read from a PCI Mailbox 2-48
icp_mailbox_write Write to a PCI Mailbox 2-48
icp_mem_size Return the IC-PCI Memory Size 2-49
icp_pixsz Set or Return the Camera Pixel Size 2-49
icp_put_acq_start_addr Set the Camera Acquire Start Address 2-50
icp_put_bm_aoix Set the Bus-Master-Mode AOI Horizontal Size 2-50
icp_put_bm_dst_addr Set the Bus-Master-Mode Destination Address 2-51
icp_put_bm_firstf_start Set the First Field Starting Address of Bus-Master Transfer 2-52
icp_put_bm_secondf_start Set the Second Field Starting Address of Bus-Master Transfer 2-52
icp_read_area Read an Area in IC-PCI Memory to a Buffer 2-53
icp_read_dest_pitch_area Read an Area Adding Destination Pitch 2-54
icp_read_harea Read Large Area in IC-PCI Memory to an Array (Win16 3.x only) 2-55
icp_read_plane_area Read a Color Plane from a Planar Frame 2-56
icp_read_roi_area Read Area from ROI to an Array 2-57

Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com

icp_read_roi_harea Read Large Area from ROI to an Array (Win16 3.x only) 2-58
icp_rhbline Read a Horizontal Line of BYTE Values 2-59
icp_rhdwline Read a Horizontal Line of DWORD Values 2-60
icp_rhwline Read a Horizontal Line of WORD Values 2-60
icp_roi_attrib Read Attributes of a ROI 2-61
icp_roi_rpix Read a Pixel from a ROI 2-62
icp_roi_set_lineio Set Up the IC-PCI for Access into ROIs 2-63
icp_roi_wpix Write a Pixel to a ROI 2-64
icp_rpix Read a Pixel Value in IC-PCI Memory 2-64
icp_rvbline Read a Vertical Line of BYTE Values 2-65
icp_rvdwline Read a Vertical Line of DWORD Values 2-65
icp_rvwline Read a Vertical Line of WORD Values 2-66
icp_scanmd_status Return the Camera Scan Mode Status 2-67
icp_seq_snap Acquire Sequential Images 2-67
icp_seq_snap_async Acquire Sequential Images Asynchronously 2-68
icp_set_cur_dim Set Current Dimensions for an IC-PCI Buffer Frame 2-69
icp_set_frame_ilace Set the Interlace Mode for a Frame 2-69
icp_set_lineio Set Up the IC-PCI for Host Access to Frames 2-70
icp_set_xform Set Up for Bus Master Transform 2-71
icp_snap Acquire a Single Image 2-71
icp_snap_async Acquire a Single Image Asynchronously 2-72
icp_soft_reset Set or Return the Software Reset 2-73
icp_start_field Set or Return the Starting Field 2-74
icp_start_ping_pong Start a “Ping Pong” Acquistion and Transfer into a Host Memory Frame . . 2-74
icp_start_VCR_record Start a VCR Recording 2-75
icp_start_VCR_record_area Start a VCR Recording of an Area 2-76
icp_stop_ping_pong Stop a “Ping Pong” Transfer to Host Memory 2-77
icp_stop_VCR_record Stop a VCR Recording 2-77
icp_sync_mode Set or Return the Sync Mode Flag 2-78
icp_system_colors_off Disable DOS System Colors (DOS only) 2-79
icp_system_colors_on Enable DOS System Colors (DOS only) 2-79
icp_system_colors_restore Restore DOS System Colors (DOS only) 2-79
icp_trig_snap Perform a Triggered Snap 2-80
icp_vblank_status Return the Camera Vertical Blank Status 2-81
icp_VCR_locate Locate a Pixel in a Host Frame 2-81
icp_VCR_play Display a Recorded VCR Sequence 2-82
icp_wait_acq Wait for Completion of the Current Acquisition 2-83
icp_wait_bmdone Wait for Completion of Bus Master Transfer 2-83

Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com

icp_wait_notvb Wait until not in a Vertical Blank 2-84
icp_wait_start_vb Wait for the Start of Next Camera Vertical Blank 2-85
icp_wait_trig Wait for Triggered Acquisition to Complete 2-85
icp_wait_vb Wait for the Next Vertical Blank 2-86
icp_whbline Write a Horizontal Line of BYTE Values 2-87
icp_whdwline Write a Horizontal Line of DWORD Values 2-88
icp_whwline Write a Horizontal Line of WORD Values 2-89
icp_wpix Write a Pixel Value in IC-PCI Memory 2-89
icp_write_area Write an Area in IC-PCI Memory from an Array 2-90
icp_write_harea Write Large Area in IC-PCI Memory from an Array (Win16 3.x only) 2-91
icp_write_roi_area Write an Area in a ROI from an Array 2-92
icp_write_roi_harea Write a Large Area in a ROI from an Array (Windows 3.1 only) 2-93
icp_wvbline Write a Vertical Line of BYTE Values 2-94
icp_wvdwline Write a Vertical Line of DWORD Values 2-94
icp_wvwline Write a Vertical Line of WORD Values 2-95
icp_wxzoom Set or Return the Horizontal Write Zoom Factor 2-95
icp_wyzoom Set or Return the Vertical Write Zoom Factor 2-96

Appendix A. IC-PCI Registers

Appendix B. Configuration Files

Sample Configuration Files B–1
Creating the Binary Configuration File B–2
Required Parameters for IC-PCI Configuration B–3
Optional Parameters for IC-PCI Configuration B–3
ASCII Configuration File Conventions B–5
Comment Fields B–5
White Space B–5
Number Fields B–5
Order of Parameters B–5
Group Names B–5

Index

Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com

List of Tables

Number	Title	Page
Table 1-1.	Color Storage	1-4
Table 1-2.	Error Messages	1-6
Table 1-3.	Initialization and Set-up Functions	1-8
Table 1-4.	Acquisition and Synchronization Functions	1-9
Table 1-5.	High Level Acquisition	1-10
Table 1-6.	Read and Write Functions	1-10
Table 1-7.	Area Functions	1-12
Table 1-8.	Module Structure Functions	1-13
Table 1-9.	DOS Display Functions	1-15
Table A-1.	Interface Control Registers	A-2
Table A-2.	Control Registers	A-3
Table A-2 Continued.	Control Registers	A-4
Table A-2 Continued.	Control Registers	A-5

List of Figures

Number	Title	Page
Figure 1-1.	IC-PCI Simplified Block Diagram	1-2
Figure A-1.	PCI Interface Control Registers	A-6
Figure A-2.	Control Registers	A-7

Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com

CHAPTER 1
INTRODUCTION

The IC-PCI software is a library of functions that control and operate Imaging Technology's IC-PCI (image capture for PCI bus) mother board. The IC-PCI software is distributed as an object code library that can be linked to application programs written in C.

NOTE *Please read the "Software Release Notes" provided before attempting any set-up or use of this package. For complete software installation information, see the "IC-PCI Installation Guide," but examine the "Software Release Notes" for any exceptions to the installation guide.*

HARDWARE OVERVIEW

The IC-PCI is an image capture board with a very fast 32-bit PCI-bus host interface. The IC-PCI contains frame memory, and supports one AM (acquisition module) plug-in camera interface. All acquisition, including camera set-up and selection, is done through the AM.

The IC-PCI is designed for fast acquisition of camera data into its frame memory, and fast transfer from frame memory out to the PCI bus (into host or VGA memory). The IC-PCI also supports PCI bus write access to its memory, and control registers.

The IC-PCI can operate as both Bus Master and Target (Slave). In Bus Master mode the IC-PCI supplies both addresses and data on the bus with the image frame memory being the data source. The IC-PCI Bus Master mode only supports write accesses (IC-PCI data output to the host PCI bus). As a Target, the IC-PCI can be accessed by other Bus Masters in the host system. The IC-PCI supports Target access to IC-PCI control registers, AM address space, and image memory. Both read and write operations are supported in Target mode.

Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com

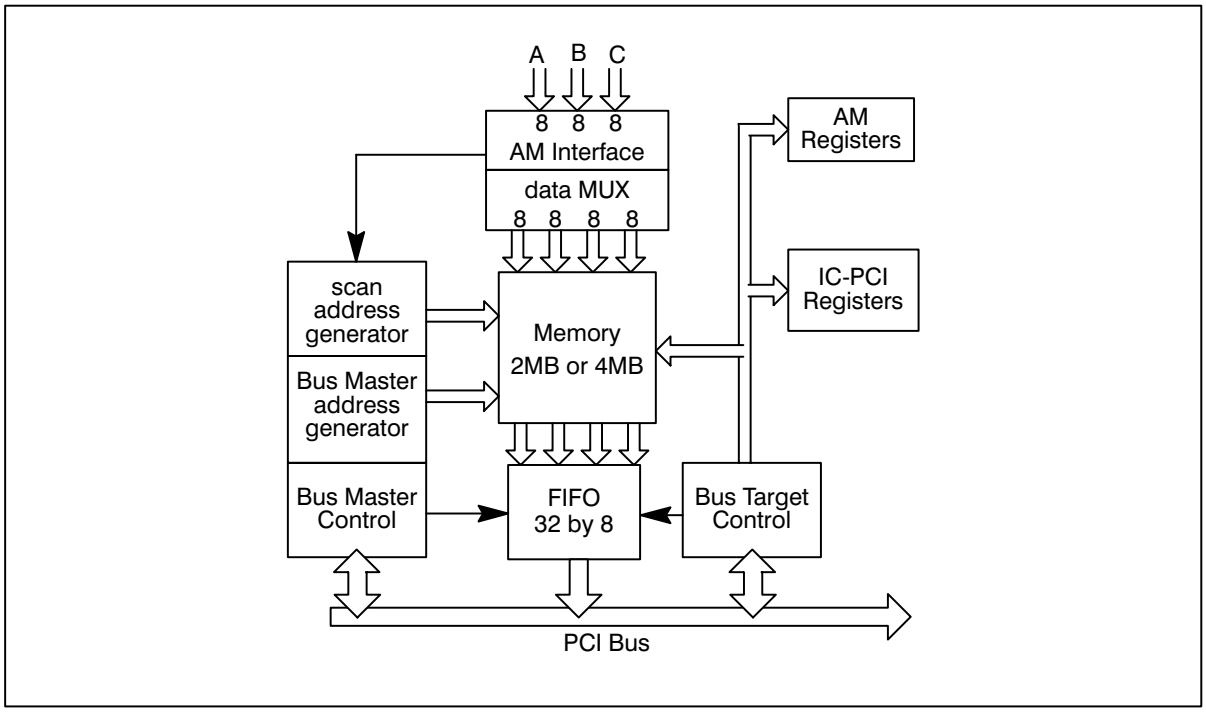


Figure 1-1. IC-PCI Simplified Block Diagram

IMAGE MEMORY

The IC-PCI contains 4MB (four Megabytes) or 2MB of memory. The memory is linear, not rectangular. This allows more flexibility with only a small amount of management. The ITEX software can organize and manage the memory as rectangular frames.

Linear Memory

The linear format provides flexibility by having no rectangular boundary limits on the size of acquired data (up to the 2MB or 4MB limit of installed memory).

The acquisition module accesses IC-PCI memory on 4KB boundaries or blocks. Multiple images can be stored by supplying an acquisition starting address.

CAUTION *There is no internal hardware protection to keep one image from overwriting another. The software depends completely on logical frame definitions. The function icp_create_frame allows you to create frames of any size to buffer the acquired data.*

Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com

Host access to the IC-PCI memory is always on DWORD boundaries. Bytes or WORD's are accessed individually by reading/writing on 32-bit address boundaries and masking or disregarding the extra bytes within the DWORD. 24-bit full color data is stored as 32-bit data with the upper byte zeroed.

The linear memory does not combine the two fields of interlaced video in IC-PCI memory. Each field is stored sequentially in memory. For bus master transfers, the ITEX software assigns a "first field start address" and a "second field start address". The two fields are combined as they are read out to host or VGA memory.

The IC-PCI linear memory is a departure from the earlier members of the MVC family. The IMA, IMS, IM-VL (IML) and IC-VL all used rectangular memory. To date only the CM-PA used linear image memory.

"IC-VL Compatible Mode"

To simplify migrating applications from an IC-VL or an IM-VL, an "IC-VL compatible" mode has been implemented in the IC-PCI software, invoked by the configuration file. This mode applies to Win16 only; this mode is not supported in NT, 95 or 98. This mode allows any (or all) of the following frames to be defined in the configuration file:

ICP_FRAMEA0 – 8-bit frame A0, maximum size: 1024 pixels wide by 1024 lines.

ICP_FRAMEA1 – 8-bit frame A1, maximum size: 1024 pixels wide by 1024 lines.

ICP_FRAMEB0 – 8-bit frame B0, maximum size: 1024 pixels wide by 1024 lines.

ICP_FRAMEB1 – 8-bit frame B1, maximum size: 1024 pixels wide by 1024 lines.

ICP_FRAMEA – 16-bit frame A, maximum size: 1024 pixels wide by 1024 lines. Unique frame; not logically related to A0 and A1.

ICP_FRAMEB – 16-bit frame B, maximum size: 1024 pixels wide by 1024 lines. Unique frame; not logically related to B0 and B1.

ICP_FRAMERGB – 24-bit frame, maximum size: 1024 pixels wide by 512 lines. Unique frame; not logically related to A0, A1, B0, B1, A, or B.

NOTE *Not all 7 frames can be defined at their maximum sizes.*

The image stored does not have to fill the frame buffer; a 512 by 480 image can be stored in a 1024 by 1024 frame buffer without the need to change the frame buffer size.

In "IC-VL compatible" the frames pre-exist, before running the application program, and the application program does not have to define its frames. Frame definitions do not have to be added to the application program (they are added to the configuration file).

Acquisition Control

The IC-PCI provides a camera interface for the AM family. The AM contains line FIFOs which are loaded at the camera rate and unloaded at the 40 MHz operating rate of the IC-PCI. The interface supports 24 bits of data, external triggering and interrupts.

Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com

An image is acquired relative to the camera timing. The acquisition begins at the trailing edge of vertical blank and ends at the leading edge of vertical blank. Alternatively, this operation begins upon receiving a trigger pulse from the AM.

The IC-PCI receives interlaced or non-interlaced camera data from an AM Module. Pixel format must be programmed prior to image acquisition. The IC-PCI accepts data from the AM modules up to 40 MHz per 8-bit bus. Image sizes from the AM module are only limited to occur on 8-byte boundaries, and the size of frame memory (2MB or 4MB). Images can be acquired beginning on any 4KB boundary in memory.

Interlaced image acquires can be programmed to start on either the next field, even field, or odd field.

Color Storage

Table 1-1 lists the output channels from the AM-CLR, the associated frame buffer and the color component stored in that frame buffer. The AM-CLR supports more than the RGB and HSI color models. *AM-CLR Software Manual* contains more information about color acquisition and the color models.

Table 1-1. Color Storage

	ICP_RED	**ICP_GREEN**	**ICP_BLUE**
Color format	*AM output A*	*AM output B*	*AM output C*
RGB	R	G	B
HSI	H	I	S
R-Y,Y, B-Y	R-Y	Y	B-Y

Bus Master Operation

In bus master operation, the IC-PCI is granted control of the PCI bus and becomes the source of both addresses and data to some other target location within the system. The size and start location of the data transferred is programmable, allowing AOIs (Areas of Interest) to be sent to a PCI bus Target. Interlaced images in IC-PCI memory are stored sequentially when acquired (first field-second field). In bus master mode, the IC-PCI provides capability to transfer an interlaced image to the host in its original interlaced format (even line, odd line). This feature greatly reduces software overhead when transferring interlaced images.

SOFTWARE INTRODUCTION

All the functions that operate the IC-PCI begin with the prefix “icp_.” To use these functions in an application, include the header files “itxcore.h” and “icp.h.” These header files pull in the appropriate system and IC-PCI data structures and symbol definitions.

Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com

Argument Types

The argument types used in ITEX IC-PCI functions are:

- Signed integers (int)
- Characters (char)
- Long integer (long)
- Floating-point integers (float)
- Double precision floating-point integers (double)
- 16-bit value (short)

On a DOS system, an integer value is 16 bits. Whenever possible, short and long argument types are used. To ensure that the proper size data element is used, use these additional data types:

- Eight-bit unsigned (BYTE)
- 16-bit unsigned (WORD)
- 32-bit unsigned (DWORD)

For example, a value declared as type WORD is an unsigned 16-bit value. Arrays of these values and pointers to these values are also used.

The IC-PCI software uses these data structures:

- MODCNF defines the general board configuration for the IC-PCI. It contains all the necessary information about the IC-PCI.
- ICPCNF defines the IC-PCI-specific configuration for the IC-PCI.

IC-PCI Selection

In an application, after configuring the system, pointers to the IC-PCI on the system should be set up. The application calls **itx_get_slotmodcnf** which returns a pointer to a MODCNF structure. Any calls to ITEX IC-PCI functions use this MODCNF structure to indicate the IC-PCI to perform on. Each IC-PCI board and acquisition module on a IC-PCI has a pointer. Because the functions in ITEX receive a pointer to MODCNF, the functions have access to the necessary information about the board.

The IC-PCI software supplies the functions to return a board pointer for a daughter board attached to an IC-PCI. The function **icp_get_slot0** retrieves a pointer to the AM.

Function Levels

The functions are classified in three levels: high, medium, and low. A high level function requires no in-depth knowledge of how the IC-PCI hardware and software function. A medium level function suggests familiarity with the IC-PCI hardware and software functional design. A low level function requires detailed knowledge of how the IC-PCI hardware and software function.

Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com

Returned Values

In many cases, functions return a short value. This value is either a return parameter or an error code defined in the header file “itxerr.h” or “icperr.h.” If no error has occurred, the function returns the value ICP_NO_ERROR; otherwise, it returns a negative number that specifies the error. These error codes are noted in the function descriptions as “Returned Values.” Table 1-2 lists the function errors.

Table 1-2. Error Messages

Error	*Description*
ICP_NO_ERROR	Operation completed successfully.
ICP_BAD_ARG	Bad argument.
ICP_BAD_FRAME	Invalid frame, pointer returns NULL, frame not defined.
ICP_NOROIID	Frame ROI child limit of 16 exceeded.
ICP_NOFRAMEID	Out of frame IDs, limit of 16 exceeded.
ICP_FRAMEXISTS	Same frame created twice.
ICP_BAD_ROI	Invalid ROI, pointer returns NULL, ROI not defined.
ICP_BAD_MOD	Invalid mod pointer.
ICP_BAD_AMMOD	Invalid AM mod pointer.
ICP_BAD_VB	Vertical Blank status bit stuck.
ICP_ACQ_TOO_BIG	ACQ dimension exceeds frame.

The IC-PCI can also return system errors; these error start with “ITX_” and are described in the *System-Level Software Manual*.

Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com

Interrupt Events

For mother boards and software platforms that support the ITEX Interrupt API software, the IC-PCI presents the following interrupt events:

ICP_INTR_VB – Interupt on vertical blank.

ICP_INTR_ODD_VB – Interupt on every ODD vertical blank.

ICP_INTR_EVEN_VB – Interupt on every EVEN vertical blank.

ICP_INTR_AM – Interrupt on signal from acquisition module.

ICP_INTR_TRIG – End-of-Trigger interrupt

ICP_INTR_ACQ_ADRS – Interrupt when specified acquisition address accessed

ICP_INTR_ACQ – End-of-Acquisition interrupt

ICP_INTR_BMDONE – Interrupt when bus mastering completed.

ICP_INTR_INPORT – Parallel port interrupt

The correct interrupt mode must be enabled by the configuration file or through API calls.

Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com

Initialization Functions

The functions in Table 1-3 initialize and set-up the IC-PCI and acquire images. The acquisition module (AM) on the IC-PCI performs camera set-up and selection. See the software manuals for the AM-VS, AM-FA, AM-DIG, or AM-CLR for more information.

Table 1-3. Initialization and Set-up Functions

Function	*Description*	*Page*
Pointers		
icp_bm_mode	Set or return the bus master enable flag	2-11
icp_get_slot0	Return the AM module structure	2-41
icp_sync_mode	Set or return the sync mode flag	2-78
Registers		
icp_iregs	Initialize the IC-PCI registers	2-44
icp_dregs	Display the IC-PCI register contents	2-31
Frames		
icp_create_frame	Create a frame buffer	2-21
icp_create_roi	Create a ROI structure	2-22
icp_create_seq_frames	Create a sequential frames buffer	2-23
icp_delete_all_frame_rois	Delete all ROIs in a frame buffer	2-26
icp_delete_all_frames	Delete all frame buffers and ROIs	2-26
icp_delete_frame	Delete a frame buffer	2-27
icp_delete_roi	Delete a ROI structure	2-27
icp_delete_seq_frame	Delete a sequential frames buffer	2-28
icp_frame_attrib	Read attributes of a frame buffer	2-34
icp_frame_exist	Test if a frame buffer already exists	2-35
icp_get_acq_dim	Return dimensions of camera frame	2-36
icp_roi_attrib	Read attributes of a ROI	2-61
icp_set_cur_dim	Set the acquisition frame dimensions	2-69
icp_set_frame_ilace	Set the acquisition frame interlace mode	2-69
icp_set_xform	Set up for bus master transform	2-71

Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com

Acquisition Functions

The functions in Table 1-4 synchronize the IC-PCI and acquire images. Synchronization functions coordinate acquisitions with the camera frame timing to ensure that a complete image is acquired.

Table 1-4. Acquisition and Synchronization Functions

Function	*Description*	*Page*
Acquisition to IC-PCI memory		
icp_acqbits	Set or return acquisition command bits	2-2
icp_freeze	Stop acquisition	2-35
icp_get_acq_dim	Return dimensions of camera frame	2-36
icp_grab	Acquire images continuously	2-41
icp_seq_snap	Acquire sequential images	2-67
icp_seq_snap_async	Acquire sequential images asynchronously	2-68
icp_set_cur_dim	Set the acquisition frame dimensions	2-69
icp_set_frame_ilace	Set the interlace mode for a frame	2-69
icp_snap	Acquire a single image	2-71
icp_snap_async	Acquire a single image asynchronously	2-72
icp_trig_snap	Perform a triggered snap	2-80
icp_wxzoom	Set or return horizontal write zoom	2-95
icp_wyzoom	Set oir return vertical write zoom	2-96
Acquisition to host memory		
icp_host_large_format_ _acquire	Aquire large format image to host memory	2-42
icp_host_large_format_ _wacquire	Acquire large format image to host with wait	2-43
icp_large_format_snap	Acquire single large-format image	2-45
icp_large_format_trig_snap	Triggered large-format snap	2-46
Synchronization		
icp_wait_acq	Wait for completion of current acquisition	2-83
icp_wait_bmdone	Wait for completion of bus master transfer	2-83
icp_wait_notvb	Wait until not in a vertical blank	2-84
icp_wait_start_vb	Wait for start of next camera vertical blank	2-85
icp_wait_trig	Wait for triggered acquisition to complete	2-85
icp_wait_vb	Wait for the next vertical blank	2-86

NOTE *Many of the functions in Table 1-4 are also available as system-level (itx_) functions. Using the system-level calls makes your applications more portable.*

Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com

High Level Acquisition Functions

The functions in Table 1-5 perform higher-level acquisition functions. All of these functions acquire to host frames, created by itx_create_host_frame or itx_create_large_format_hframe Refer to the High-Level Acquisition functions in the *System-Level Software Manual* (part number 47-S60000).

Table 1-5. High Level Acquisition

Function	*Description*	*Page*
icp_get_active_ubm_frame	Return a pointer to the active frame	2-37
icp_idle_ubm_frame	Return a pointer to the idle frame	2-40
icp_host_large_format_acquire	Acquire large-format images	2-42
icp_host_large_format_wacquire	Acquire large format image with wait	
icp_start_ping_pong	Start ping-pong acquire/transfer	2-74
icp_start_VCR_record	Start a VCR recording	2-75
icp_start_VCR_record_area	Start a VCR recording of an area	2-76
icp_stop_ping_pong	Stop ping-pong acquire/transfer	2-77
icp_stop_VCR_record	Stop a VCR recording	2-77
icp_VCR_locate	Locate a pixel in a host VCR frame	2-81
icp_VCR_play	Display a VCR sequence	2-82

NOTE *All the functions in Table 1-5 are also available as system–Level (itx_) functions. Using the system-level calls makes your applications more portable.*

Read and Write Functions

The functions in Table 1-6 read or write pixel values to or from frame buffers. You must call **icp_set_lineio** or **icp_roi_set_lineio** before every call to a BYTE, WORD, or DWORD function listed in Table 1-6. The configuration file sets the memory size and pixel depth. The functions in Table 1-6 use these values. For applications that need to change configuration values, use the Module Structure functions in Table 1-8.

Table 1-6. Read and Write Functions

Function	*Description*	*Page*
Single Pixel Functions		
icp_roi_rpix	Read pixel from a ROI	2-62
icp_roi_wpix	Write pixel to a ROI	2-64
icp_rpix	Read pixel value in IC-PCI memory	2-64
icp_wpix	Write pixel value in IC-PCI memory	2-89
Set-Up for BYTE, WORD, and DWORD functions		
icp_bm_lock	Lock regions for bus master	2-10
icp_bm_read	Read into locked region	2-12

Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com

icp_bm_read_subregion	Read area into locked region	2-12
icp_bm_unlock	Unlock regions for bus master	2-13
icp_roi_set_lineio	Set up the IC-PCI for access into ROIs	2-63
icp_set_lineio	Set up IC-PCI for host access to frames	2-70
icp_wait_bmdone	Wait for bus master transfer completion	2-83
BYTE Value		
icp_chbline	Clear horizontal line of BYTE Values	2-15
icp_cvbline	Clear vertical line of BYTE values	2-24
icp_rhbline	Read horizontal line of BYTE values	2-59
icp_rvbline	Read vertical line of BYTE values	2-65
icp_whbline	Write horizontal line of BYTE values	2-87
icp_wvbline	Write vertical line of BYTE values	2-94
WORD Value		
icp_chwline	Clear horizontal line of WORD values	2-16
icp_cvwline	Clear vertical line of WORD values	2-25
icp_rhwline	Read horizontal line of WORD values	2-60
icp_rvwline	Read vertical line of WORD values	2-66
icp_whwline	Write horizontal line of WORD values	2-89
icp_wvwline	Write vertical line of WORD values	2-95
DWORD Value		
icp_chdwline	Clear horizontal line of DWORD values	2-15
icp_cvdwline	Clear vertical line of DWORD values	2-25
icp_rhdwline	Read horizontal line of DWORD values	2-60
icp_rvdwline	Read vertical line of DWORD values	2-65
icp_whdwline	Write horizontal line of DWORD values	2-88
icp_wvdwline	Write vertical line of DWORD values	2-94
Dump		
icp_dump_mem	Display (dump) an area of memory	2-31
icp_dump_roi_mem	Display (dump) memory ROI	2-32

Image Read and Write Functions

To read or write images to and from files, use system-level functions itx_read_image**,** itx_read_imagearea**,** itx_write_image, itx_host_save_frame, and itx_host_save_movie. The *System-Level Software Manual* (part number 47-S60000) describes system-level functions.

Area Functions

The functions in Table 1-7 operate on areas and frames: copy, clear, and display.

NOTE *For icp_cp_area, frame1 and frame2 must have identical depth and color attributes. For icp_cp_roi, roi1 and roi2 must have identical depth, dx, and dy attributes.*

Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com

Table 1-7. Area Functions

Function	*Description*	*Page*
icp_clr_area	Clear an area in IC-PCI memory	2-17
icp_clr_frame	Clear frame in IC-PCI memory	2-18
icp_clr_roi	Clear ROI	2-18
icp_cp_area	Copy an area in IC-PCI memory	2-19
icp_cp_roi	Copy ROI	2-20
icp_display_area	Define an area for display	2-28
icp_display_frame	Define frame for display	2-29
icp_display_roi	Define ROI for display	2-30
icp_display_roi_area	Define ROI area for display	2-30
icp_line_reverse_read	Read an area in reverse line order	2-47
icp_read_area	Read IC-PCI memory area to an array	2-53
icp_read_plane_area	Read a color plane form a planar frame	2-56
icp_read_roi_area	Read an area in ROI to an array	2-57
icp_read_dest_pitch_area	Read an area adding destination pitch	2-54
icp_write_area	Write IC-PCI memory area from array	2-90
icp_write_roi_area	Write an area in ROI from an array	2-92
Win16 only		
icp_read_harea	Read large area to an array	2-55
icp_read_roi_harea	Read large area from ROI to an array	2-58
icp_write_harea	Write large area from an array	2-91
icp_write_roi_harea	Write large area in ROI from an array	2-93

Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com

Module Structure Functions

The functions in Table 1-8 return and change values in the IC-PCI data structure MODCNF. They also write the values to the appropriate registers. MODCNF is created at system initialization with configuration file default values.

Table 1-8. Module Structure Functions

Function	*Description*	*Page*
icp_acq_addr_status	Return the current acquisition IC-PCI memory block	2-2
icp_acq_fstart	Return the starting field status	2-3
icp_acq_in_progress	Test for acquisition in progress	2-3
icp_acq_pending	Test for acquisition command pending	2-4
icp_amtrig_enable	Enable trigger input from the acquisition module	2-4
icp_bm_count	Set or return the bus-master-mode transfer count	2-7
icp_bm_done	Test for bus-master-mode transfer done	2-8
icp_bm_fifo_4P	Test FIFO contains 4 or more DWORDs ready for transfer	2-8
icp_bm_fifo_empty	Test for FIFO Empty	2-8
icp_bm_fifo_full	Test for Bus-Master-Mode FIFO Full	2-9
icp_bm_ilace	Set or return the bus-master-mode interlace mode	2-9
icp_bus_request	Set or return the PCI-bus request	2-14
icp_fcount	Return acquisition frame count	2-32
icp_field_status	Return the camera field status	2-33
icp_fifo_reset	Reset (clear) the FIFO	2-34
icp_get_acq_start_addr	Return the camera acquire start address	2-36
icp_get_bm_aoix	Return the bus-master-mode AOI horizontal size	2-38
icp_get_bm_dst_addr	Return the bus-master-mode destination address	2-38
icp_get_bm_firstf_start	Return the first field starting address of bus-master transfer	2-39
icp_get_bm_secondf_start	Return the second field starting address of bus-master transfer	2-39

(continued)

Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com

Table 1-8 continued. Module Structure Functions

Function	*Description*	*Page*
icp_get_hw_revision	Get the hardware revision ID	2-40
icp_image_pitch	Set bus-master transfer image pitch	2-44
icp_latency_timer	Set or return the bus-master latency timer	2-46
icp_mailbox_read	Read from a PCI mailbox	2-48
icp_mailbox_write	Write to a PCI mailbox	2-48
icp_mem_size	Return the IC-PCI memory size	2-49
icp_pixsz	Set or return the camera pixel size	2-49
icp_put_acq_start_addr	Set the camera acquire start address	2-50
icp_put_bm_aoix	Set the bus-master-mode AOI horizontal size	2-50
icp_put_bm_dst_addr	Set the bus-master-mode destination address	2-51
icp_put_bm_firstf_start	Set the first field starting address of bus-master transfer	2-52
icp_put_bm_secondf_start	Set the second field starting address of bus-master transfer	2-52
icp_scanmd_status	Return camera scan mode status	2-67
icp_set_xform	Set up for bus master transform	2-71
icp_soft_reset	Set or return the software reset	2-73
icp_start_field	Set or return the starting field	2-74
icp_vblank_status	Return camera vertical blank status	2-81

Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com

DOS Display Functions

The functions in Table 1-9 are used for display under DOS (not under Windows). These functions are used with the ITEX-IC System-level DOS display functions.

Table 1-9. DOS Display Functions

Function	*Description*	*Page*
icp_bitblt	Transfer one frame to VGA	2-5
icp_bitblt_end	Stop direct transfer to VGA	2-6
icp_bitblt_start	Set up for direct transfer from IC-PCI to VGA	2-6
icp_system_colors_off	Disable DOS system colors	2-79
icp_system_colors_on	Enable DOS system colors	2-79
icp_system_colors_restore	Restore DOS system colors	2-79

The useful order of these display functions in an application is:

icp_system_colors_off
itx_vga_save_DACs
itx_vga_start
itx_vga_setcolor
itx_vga_setwindow
itx_vga_setpage
icp_bitblt_start
icp_bitblt
icp_bitblt_end
itx_vga_end
itx_vga_load_DACs
icp_system_colors_on

Refer to the display functions and the Image Abstractions API functions in the ITEX-IC System-Level Software Manual

Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com

Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com

CHAPTER 2
IC-PCI FUNCTIONS

The IC-PCI functions are organized in alphabetical order in this chapter. Chapter 1 discusses their functional relationships, and contains tables of functional groupings for: initialization, synchronization, acquisition, pixel read and write, line read and write, image read and write, area operations, and module structure functions.

All IC-PCI function prototypes, and global defines specific to the IC-PCI, are found in the file “icp.h” and all register names are defined in the file “icpreg.h”.

NOTE *Chapter 1 of this manual contains tables listing IC-PCI routines by* functional groups*, and gives page references. Chapter 2 presents the functions in* alphabetical order*. If you prefer functional groupings, please consult Chapter 1.*

Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com

icp_acq_addr_status – Return the Current Acquisition IC-PCI Memory Block

Syntax #include <icp.h>

WORD **icp_acq_addr_status**(MODCNF **icpmod*);

icpmod Pointer to the IC-PCI module structure.

Description

The function **icp_acq_addr_status** returns the current acquisition block. The image memory is partitioned into 4K blocks. The 2MB version has 512 blocks (numbered 0 to 511) and the 4MB version has 1024 blocks (numbered 0 to 1023).

This is a low level function. The programmer should have detailed knowledge of how the IC-PCI hardware and software function.

Returned Values

Acquisition block number: 0 to 1023 (0 to 511 for 2MB version)

Hardware Modified

None

icp_acqbits – Set or Return the Acquisition Command Bits

Syntax #include <icp.h>

short **icp_acqbits**(MODCNF **icpmod*, short *frame*, short *mode*);

icpmod Pointer to the IC-PCI module structure.

frame The frame to write to: a frame ID returned by icp_create_frame, or defined by the configuration file.

mode Acquisition command to write:
ICP_FREEZE – Stop continuous acquire.
ICP_SNAP – Acquire a single frame.
ICP_GRAB – Acquire continuously.
INQUIRE – Return the value of the acquisition bits.

Description

The function **icp_acqbits** writes to the acquisition command bits for *frame*. This is a low–level function, and does *not* wait for acquisition complete.

A new acquisition command is accepted at the leading edge of a vertical blank. Acquisition begins at the trailing edge of the vertical blank, and terminates at the leading edge of a later vertical blank. The command bits change state after the new acquisition begins. Returning the acquisition command bits shows the next command pending, not the current acquisition. The Acquisition Status bits show the current acquisition.

This is a medium level function. The programmer should be familiar with the IC-PCI hardware and software functional design.

Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com

Returned Values

If mode = INQUIRE, the pending command in the acquisition command bits for *frame*: snap, grab, freeze is returned, else an error value is returned.

ICP_NO_ERROR – Operation completed successfully. (continued)

ICP_BAD_ARG invalid frame – frame pointer returned NULL; frame not defined.

ICP_BAD_ARG can't acquire seq frame by itself – frame points to sequential frame definition; use icp_seq_snap.

ICP_BAD_DMODCNF invalid AM mod pointer – pointer returns NULL; AM not defined.

ITX_TIMEOUT – could not set acquisition bits.

Hardware Modified

ICP_ACQMD

icp_acq_fstart– Return the Starting Field Status

Syntax #include <icp.h>

short **icp_acq_fstart**(MODCNF **icpmod*);

icpmod Pointer to the IC-PCI module structure.

Description

The function **icp_acq_fstart** returns which field began the interlaced acquisition. This function reads the status bit ICP_FSTRT.

This is a low level function. The programmer should have detailed knowledge of how the IC-PCI hardware and software function.

Returned Values

ICP_EVEN – frame acquire began on even field.

ICP_ODD – frame acquire began on odd field.

Hardware Modified

ICP_FSTART

icp_acq_in_progress – Test for Acquisition in Progress

Syntax #include <icp.h>

BOOL **icp_acq_in_process**(MODCNF **icpmod*);

icpmod Pointer to the IC-PCI module structure.

Description

The function **icp_acq_in_process** reads the status field ICP_GSTAT to determine if a grab is in progress.

Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com

This is a low level function. The programmer should have detailed knowledge of how the IC-PCI hardware and software function.

Returned Values

TRUE – acquisition in progress.

FALSE – no acquisition in progress.

Hardware Modified

ICP_GSTAT

icp_acq_pending – Test for Acquistion Command Pending

Syntax #include <icp.h>

BOOL **icp_acq_pending**(MODCNF **icpmod*);

icpmod Pointer to the IC-PCI module structure.

Description

The function **icp_acq_pending** reads the status field ICP_NEWAQ to determine if an acquisition command is pending (waiting to begin at the end of the current acquisition).

This is a low level function. The programmer should have detailed knowledge of how the IC-PCI hardware and software function.

Returned Values

TRUE – new acquisition pending (waiting to begin).

FALSE – no command pending.

Hardware Modified

ICP_NEWACQ

icp_amtrig_enable – Enable the Trigger Input from the Acquisition Module

Syntax #include <icp.h>

short **icp_amtrig_enable**(MODCNF **icpmod*, short *mode*);

icpmod Pointer to the IC-PCI module structure.

mode AM trigger mode:
ICP_ENABLED – enable the trigger input
ICP_DISABLED – disable the trigger input
INQUIRE – Return the current setting for *mode*.

Description

The function **icp_amtrig_enable** enables or disables the trigger input from the Acquisition Module or returns the current state of the register. This function modifies the IC-PCI module structure.

Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com

This is a low level function. The programmer should have detailed knowledge of how the IC-PCI hardware and software function.

Returned Values

The current setting for *mode* ICP_ENABLED, ICP_DISABLED.

ICP_NO_ERROR – Operation completed successfully.

Hardware Modified

ICP_TMODE

icp_bitblt – Transfer one Frame to VGA (DOS only)

Syntax #include <icpnovga.h>
void **icp_bitblt**();

Description

The function **icp_bitblt** copies a single image frame located in the IC-PCI frame memory to a block on the VGA display, using the setup and coordinates established by the function icp_bitblt_start. The function icp_bitblt_start must have been previously called to initialize these conditions.

This display function operates in DOS 32-bit mode only. This function is not Windows compatible.

The function icp_bitblt internally calls itx_vga_setpage and icp_read_area. If bus master mode is disabled, this function may change the memory offset pointer by calling icp_set_lineio.

The function icp_bitblt_start must be called before icp_bitblt is called. The function icp_bitblt_start sets up the IC-PCI and PC VGA memory for transfers, but does not initiate a transfer. This function icp_bitblt causes a single transfer. icp_bitblt must be called once for every transfer. The function icp_bitblt_end releases the PC VGA memory from transfer setup.

NOTE *This function operates with VESA compatible Super VGA, in 256 colors resolution. This function works for D32X (Watcom compiler) only.*

This is a medium level function. The programmer should be familiar with the IC-PCI hardware and software functional design.

Returned Values

None

Hardware Modified

None

Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com

icp_bitblt_end – Stop Direct Transfer to VGA (DOS only)

Syntax #include <icpnovga.h>

void **icp_bitblt_end**();

Description

The function **icp_bitblt_end** freezes acquisition, terminates direct video transfer from the IC-PCI to the VGA, disables super VGA graphic mode, and restores the VGA DACs to their previous values before the icp_bitblt_start was called.

This display function operates in DOS 32-bit mode only. This function is not Windows compatible.

The function icp_bitblt_start sets up the IC-PCI and PC VGA memory for transfers, but does not initiate a transfer. The function icp_bitblt causes a single transfer. icp_bitblt must be called once for every transfer. The function icp_bitblt_end releases the PC VGA memory from transfer setup.

The function icp_bitblt_end internally calls icp_freeze, icp_bm_mode, and itx_vga_load_DACs.

This is a medium level function. The programmer should be familiar with the IC-PCI hardware and software functional design.

Returned Values

None

Hardware Modified

ICP_ACQMD

icp_bitblt_start – Set Up for Direct Transfer from IC-PCI to VGA (DOS only)

Syntax #include <icpnovga.h>

void **icp_bitblt_start**(short *x1*, short *x2*, short *y1*, short *y2*, MODCNF **icpmod*, short *icp_x*, short *icp_y*, short *disp_frame*);

x1	Left pixel location of region on VGA screen (left boundary pixel).
x2	Right pixel location of region on VGA screen (right boundary pixel).
y1	Top pixel location of region on VGA screen (top boundary line).
y2	Bottom pixel location of region on VGA screen (bottom boundary line).
icpmod	Pointer to the IC-PCI module structure.
icp_x	Horizontal starting coordinate of image in IC-PCI memory
icp_y	Vertical starting coordinate in IC-PCI memory
disp_frame	The frame to read for display: a frame ID returned by icp_create_frame, or defined in the configuration file.

Description

The function **icp_bitblt_start** sets up the IC-PCI for direct transfer of image frames located in the IC-PCI frame memory, to a block in the PC VGA memory. This function enables VGA graphic mode. This function checks the

Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com

system colors variable, and forces icp_system_colors_on if no call was made to either function icp_system_colors_on or icp_system_colors_off. This function does not cause an image transfer. These initial conditions are canceled by the functions icp_bitblt_end.

This display function operates in DOS 32-bit mode only. This function is not Windows compatible.

This is a medium level function. The programmer should be familiar with the IC-PCI hardware and software functional design.

The function icp_bitblt_start sets up the IC-PCI and PC VGA memory for transfers, but does not initiate a transfer. The function icp_bitblt causes a single transfer. icp_bitblt must be called once for every transfer. The function icp_bitblt_end releases the PC VGA memory from transfer setup.

The image in the IC-PCI is located at pixel coordinates *icp_x* and *icp_y*. The block is defined by pixel coordinates (*x1*,*y1*) and (*x2*,*y2*) on the VGA screen. The horizontal boundaries *x1* and (*x2*+1) must be multiples of 4.

The range of values for *x1, x2, y1, y2* depend on the VGA display mode. The defined block can be the whole VGA screen, or a small sub-region. *The values are clipped to fit within the VGA window.* A 512 by 480 rectangular region with the upper–left corner in the origin (0,0) is defined as *x1* =0, *x2* =511, *y1* =0, *y2* =479.

Returned Values

None

Hardware Modified

None

icp_bm_count – Set or Return the Bus-Master-Mode Transfer Count

Syntax

#include <icp.h>

DWORD **icp_bm_count**(MODCNF **icpmod*, long *count*);

icpmod	Pointer to the IC-PCI module structure.
count	Number of bytes to transfer
	INQUIRE – return the current count

Description

The function **icp_bm_count** programs the number of bytes to transfer from the IC-PCI to another system resource using bus–master mode. This is a low–level function for specifically programming transfers.

This is a low level function. The programmer should have detailed knowledge of how the IC-PCI hardware and software function.

Returned Values

Current transfer count.

ICP_NO_ERROR – Operation completed successfully.

Hardware Modified

ICP_BMXC_32

Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com

icp_bm_done – Test for Bus–Master–Mode Transfer Done

Syntax #include <icp.h>

BOOL **icp_bm_done**(MODCNF **icpmod*);

icpmod Pointer to the IC-PCI module structure.

Description

The function **icp_bm_done** checks the transfer done bit ICP_BMDONE_32 to determine when the bus master transfer is completed. This is a low–level function for specifically programming transfers.

This is a low level function. The programmer should have detailed knowledge of how the IC-PCI hardware and software function.

Returned Values

TRUE – transfer complete

FALSE – transfer in progress

Hardware Modified

None

icp_bm_fifo_4P – Test FIFO Contains 4 or More DWORDS Ready for Transfer

Syntax #include <icp.h>

BOOL **icp_bm_fifo_4P**(MODCNF **icpmod*);

icpmod Pointer to the IC-PCI module structure.

Description

The function **icp_bm_fifo_4P** reads a FIFO status bit ICP_FIFO4P_32 which flags that four or more DWORDs are loaded in the FIFO. This is a low–level function for specifically programming transfers.

This is a low level function. The programmer should have detailed knowledge of how the IC-PCI hardware and software function.

Returned Values

TRUE – FIFO contains 4 or more DWORDS

FALSE – FIFO contains less than 4 DWORDS

Hardware Modified

None

icp_bm_fifo_empty – Test for FIFO Empty

Syntax #include <icp.h>

BOOL **icp_bm_fifo_empty**(MODCNF **icpmod*);

icpmod Pointer to the IC-PCI module structure.

Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com

Description

The function **icp_bm_fifo_empty** reads the FIFO status bit ICP_FIFOEM_32 to verify there is no data in the FIFO. This is a low–level function for specifically programming transfers.

This is a low level function. The programmer should have detailed knowledge of how the IC-PCI hardware and software function.

Returned Values

TRUE – FIFO empty

FALSE – Data present in FIFO

Hardware Modified

None

icp_bm_fifo_full – Test for Bus-Master-Mode FIFO Full

Syntax #include <icp.h>

BOOL **icp_bm_fifo_full**(MODCNF **icpmod*);

icpmod Pointer to the IC-PCI module structure.

Description

The function **icp_bm_fifo_full** reads the FIFO status bit ICP_FIFOFL_32 to verify the FIFO is full of data. This is a low–level function for specifically programming transfers.

This is a low level function. The programmer should have detailed knowledge of how the IC-PCI hardware and software function.

Returned Values

TRUE – FIFO is full

FALSE – FIFO is not full

Hardware Modified

None

icp_bm_ilace – Set or Return the Bus-Master-Mode Interlace Mode

Syntax #include <icp.h>

short **icp_bm_ilace**(MODCNF **icpmod*, short *mode*);

icpmod Pointer to the IC-PCI module structure.

mode Interlace mode:
ICP_DISABLED – data interlacing mode disabled, read one field.
ICP_ENABLED – data interlacing mode enabled, read both fields.
INQUIRE – Return the current mode

Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com

Description

The function **icp_bm_ilace** programs the IC-PCI for retrieving interlaced data using bus master transfers. The IC-PCI does not combine the two fields in frame memory, but stores one field after the other, saving the starting address of each field. When interlace mode is enabled, the IC-PCI reads from both fields, and combines them in its bus master data stream. This is a low–level function for specifically programming transfers. Other data transfer functions call this function. This function modifies the IC-PCI module structure.

This is a low level function. The programmer should have detailed knowledge of how the IC-PCI hardware and software function.

Returned Values

The current interlace mode

Hardware Modified

ICP_BMILACE

icp_bm_lock – Lock Regions for Bus Master

Syntax

#include <icp.h>

DWORD **icp_bm_lock**(MODCNF **icpmod*, char **lin_buf*, DWORD *lin_size*);

icpmod	Pointer to the IC-PCI module structure.
lin_buf	Pointer to the host linear buffer (returned on allocation)
lin_size	Size of a linear host buffer used for transfer

Description

The function **icp_bm_lock** locks a range of linear host memory and prepares the host buffer for a bus master transfer by gathering all physical regions. The function icp_bm_read() does the bus master read. For repetitive bus master reading into the same buffer, call icp_bm_lock() only once. When finished with the host buffer, call icp_bm_unlock() to unlock the linear host memory. IC-PCI software supports having multiple host buffers locked open to be read into. If you use GlobalAlloc() to allocate this buffer, you must also use the GMEM_FIXED flag.

This is a low level function. The programmer should have detailed knowledge of how the IC-PCI hardware and software function.

Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com

Example

```
char  buf [maxsize];
DWORD region;
region = icp_bm_lock(icpmod,buf,maxsize)
if (region==0) error ...
for (....){
/*next reads 512X480 image from frameA0-> buf*/
icp_bm_read(icpmod,region,imp_frameA0,0,0,512, 480)
/* insert code here to process image */
}
icp_bm_unlock(impmod,region)
```

Returned Values

A DWORD pointer to the locked *region*, returns NULL on failure.

ICP_NO_ERROR – Operation completed successfully.

ITX_GEN_ERROR – Error gathering locked region.

Hardware Modified

ICP_BMSTREN, ICP_BMREQ_32

icp_bm_mode – Set or Return the Bus Master Enable Flag

Syntax

#include <icp.h>

short **icp_bm_mode**(MODCNF **icpmod*, short *mode*);

icpmod	Pointer to the IC-PCI module structure.
mode	The bus master enable flag: ICP_ENABLED – The IC-PCI may become a bus master ICP_DISABLED – The IC-PCI can not become a bus master INQUIRE – Return the current status of the flag

Description

The function **icp_bm_mode** sets a flag in the module structure to determine if the IC-PCI can become a bus master. The default is to disable the bus master flag. If the bus master flag is disabled, the IC-PCI can not become a bus master. This function modifies the IC-PCI module structure.

This is a medium level function. The programmer should be familiar with the IC-PCI hardware and software functional design.

Returned Values

The current flag value, if *mode* = INQUIRE

ICP_NO_ERROR – Operation completed successfully (flag changed).

Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com

Hardware Modified

None

icp_bm_read – Read into Locked Region

Syntax #include <icp.h>

short **icp_bm_read**(MODCNF **icpmod*, DWORD *region*);

icpmod	Pointer to the IC-PCI module structure.
region	Pointer to locked region (returned by icp_bm_lock)

Description

The function **icp_bm_read** resets the FIFO and writes data into a region locked for bus master transfer by the function icp_bm_lock. This is a low level function for specifically programming transfers.

NOTE *Allocate a GMEM_FIXED buffer for reading IC-PCI memory to a host buffer. Do not use DISCARDABLE or MOVEABLE memory buffers.*

This is a medium level function. The programmer should be familiar with the IC-PCI hardware and software functional design.

Returned Values

ICP_NO_ERROR

Hardware Modified

ICP_RESET, ICP_BMSTREN, ICP_BMDST_32, ICP_BMXC_32, ICP_BBMCTL_32, ICP_BMREQ_32

icp_bm_read_subregion – Read Area into a Locked Region

Syntax #include <icp.h>

short **icp_bm_read_subregion**(MODCNF **icpmod*, DWORD *region*, DWORD *offset*, DWORD *numbytes*);

icpmod	Pointer to the IC-PCI module structure.
region	Pointer to the locked region; retuned by icp_bm_lock.
offset	byte offset into the locked region where the read begins.
numbytes	the number of bytes to be read.

Description

The function **icp_bm_read_subregion** allows a bus master read of any portion of a locked region. In the application you can call icp_bm_lock once to lock a region; for example 80 Megabytes, and then call icp_bm_read_subregion inside the loop for each subregion.

Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com

NOTE *New function added in ITEX revision 2.8.0 for Windows NT and Windows 95. Not applicable to Win16, DOS32x or Windows 3.11.*

Returned Values

ICP_NO_ERROR

Hardware Modified

ICP_RESET, ICP_BMSTREN, ICP_BMDST_32, ICP_BMXC_32, ICP_BBMCTL_32, ICP_BMREQ_32

Example

In general terms icp_bm_read_subregion can be used as follows:

```
. . .
 host_image_buf = (DWORD *)GlobalAllocPtr(GMEM_FIXED,host_buf_size);

 lock_region = icp_bm_lock(icpmod,(pBYTE)host_image_buf,host_buf_size);

 len_remain = host_buf_size;
 offset = 0;
 while (len_remain > 0) {
   icp_bm_ilace(icpmod,0); /* Non-Interlaced!*/
   icp_put_bm_firstf_start(icpmod,0); /* Image First Field is at
address 0 */
   icp_put_bm_secondf_start(icpmod, BUF_SIZE/2 ); /*If interlaced,
Image Second Field is at Mid*/
   icp_image_pitch(icpmod,DX,ICP_PIX8);
/* Image Size and Depth */
   icp_put_bm_aoix(icpmod,DX,ICP_PIX8); /*AOI is the Complete Image */
   icp_bm_read_subregion(icpmod,lock_region,offset,min(BUF_SIZE,len_rem
ain));
   offset += BUF_SIZE;
   len_remain -= BUF_SIZE;
 }
 icp_bm_unlock(icpmod,lock_region);
. . .
```

icp_bm_unlock – Unlock Regions for Bus Master

Syntax #include <icp.h>

short **icp_bm_unlock**(MODCNF **icpmod*, DWORD *region*);

icpmod	Pointer to the IC-PCI module structure.
region	Pointer to the locked region (returned by icp_bm_lock)

Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com

Description

The function **icp_bm_unlock** unlocks (frees) the region of host memory locked by icp_bm_lock. This is a low level function for specifically programming transfers.

This is a medium level function. The programmer should be familiar with the IC-PCI hardware and software functional design.

Returned Values

ICP_NO_ERROR – Operation completed successfully.

ITX_GEN_ERROR – Error attempting to unlock region.

Hardware Modified

None

icp_bus_request – Set or Return the PCI-bus Request

Syntax #include <icp.h>

short **icp_bus_request**(MODCNF **icpmod*, short *mode*);

icpmod	Pointer to the IC-PCI module structure.
mode	PCI bus mode: ICP_HALT – Request PCI bus control ICP_START – Release PCI bus INQUIRE – Return current status

Description

The function **icp_bus_request** requests control of the PCI bus, allowing the IC-PCI to enter bus master mode. The IC-PCI can become a bus master and write IC-PCI image data to the host memory or the VGA memory. The IC-PCI cannot read data or perform any other PCI bus master functions (beyond transfer handshake for writing data). This is a low level function for specifically programming transfers. This function modifies the IC-PCI module structure.

This is a low level function. The programmer should have detailed knowledge of how the IC-PCI hardware and software function.

Returned Values

Current request status, if *mode* = INQUIRE.

ICP_NO_ERROR – Operation completed successfully.

Hardware Modified

ICP_BMREQ_32

Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com

icp_chbline – Clear a Horizontal Line of BYTE Values

Syntax #include <icp.h>

void **icp_chbline**(DWORD *offadr*, short *nwrites*, short *inc*, BYTE *val*);

offadr	The memory address returned by icp_set_lineio or icp_roi_set_lineio.
nwrites	Number of BYTE locations to clear.
inc	Number of memory elements (BYTEs) to increment between writes, from 0 to 32767 0 – Clear the same location. 1 – Clear every BYTE location (increment one byte between clears). 2 – Clear every other BYTE location (increment 2 bytes between clears). 3 – Clear every third BYTE location. . . . 32767 – Clear every 32,767th location.
val	Value to clear pixels to, or BYTE value to write at each location: 0 to 0xFF.

Description

The function **icp_chbline** clears a horizontal line in IC-PCI frame memory to a BYTE value *val*. The function writes *val* every *inc* pixel addresses *nwrites* times in a horizontal line. **You must select values for *nwrites* and *inc* that do not exceed the size of IC-PCI frame memory.**

This is a medium level function. The programmer should be familiar with the IC-PCI hardware and software functional design.

Returned Values

None

Hardware Modified

None

icp_chdwline – Clear a Horizontal Line of DWORD Values

Syntax #include <icp.h>

void **icp_chdwline**(DWORD *offadr*, short *nwrites*, short *inc*, DWORD *val*);

offadr	The memory address returned by icp_set_lineio or icp_roi_set_lineio**.**
nwrites	Number of DWORD locations to clear.
inc	Number of memory elements (DWORDs) to increment between writes, from 0 to 32767: 0 – Clear the same location. 1 – Clear every location (increment 1 DWORD). 2 – Clear every other location (increment 2 DWORDs). 3 – Clear every third location. . . . 32767 – Clear every 32,767th location.
val	Value to clear pixels to, or DWORD value to write at each location: 0 to 0xFFFFFFFF.

Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com

Description

The function **icp_chdwline** clears a horizontal line in IC-PCI frame memory to value *val*. **You must select values for *nwrites* and *inc* that do not exceed the size of IC-PCI frame memory.** If the pixel size is 24 or 32 bits (ICP_PIX24 or ICP_PIX32) each DWORD clears one 32–bit pixel. If the pixel size is 16 bits (ICP_PIX16) each DWORD clears two horizontally adjacent 16–bit pixels. If the pixel size is 8 bits (ICP_PIX8) each DWORD clears four horizontally adjacent 8–bit pixels.

This is a medium level function. The programmer should be familiar with the IC-PCI hardware and software functional design.

Returned Values

None

Hardware Modified

None

icp_chwline – Clear a Horizontal Line of WORD Values

Syntax

#include <icp.h>

void **icp_chwline**(DWORD *offadr*, short *nwrites*, short *inc*, WORD *val*);

offadr	The memory address returned by icp_set_lineio or icp_roi_set_lineio.
nwrites	Number of WORD locations to clear.
inc	Frequency with which to clear, or number of memory elements (WORDS) to increment between writes. from 0 to 32767: 0 – Clear the same location. 1 – Clear every location (increment 1 WORD between writes). 2 – Clear every other location (increment 2 WORDs between writes). 3 – Clear every third location. . . . 32767 – Clear every 32,767th location.
val	Value to clear pixels to, or WORD value to write, from 0 to 0xFFFF.

Description

The function **icp_chwline** clears a horizontal line in IC-PCI frame memory to a WORD value *val*. **You must select values for *nwrite* and *inc* that do not exceed the size of IC-PCI frame memory.** If the pixel size is 16 bits (ICP_PIX16) each WORD clears one 16–bit pixel. If the pixel size is 8 bits (ICP_PIX8) each WORD clears two horizontally adjacent 8–bit pixels.

This is a medium level function. The programmer should be familiar with the IC-PCI hardware and software functional design.

Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com

Returned Values

None

Hardware Modified

None

icp_clr_area – Clear an Area in IC-PCI Memory

Syntax #include <icp.h>

short **icp_clr_area**(MODCNF **icpmod*, short *frame*, short *x*, short *y*, short *dx*, short *dy*, DWORD *value*);

icpmod	Pointer to the IC-PCI module structure.
frame	The frame to clear: a frame ID returned by icp_create_frame, or defined in the configuration file.
x	Starting X (horizontal) coordinate.
y	Starting Y (vertical) coordinate.
dx	Horizontal dimension (width) of area to clear.
dy	Vertical dimension (height) of area to clear.
value	Value to write to the selected frame, in the range of 0 to 0xFF for 8–bit frames, 0 to 0xFFFF for 16–bit frames, and 0 to 0x00FFFFFF for 24–bit frames.

Description

The function **icp_clr_area** sets all locations in the defined area to *value*. The function checks x+dx against the memory width, and y+dy against the memory height, and clips dx and dy if necessary. This function calls icp_set_lineio and writes horizontal lines to frame memory.

For 8–bit frames, only the least significant byte is used to clear. For 16–bit frames use the least significant word (does *not* replicate the bottom byte). 24–bit frames use a complete DWORD (most significant byte forced to zero).

This is a high level function. The programmer does not need in-depth knowledge of how the IC-PCI hardware and software function.

Returned Values

ICP_NO_ERROR – Operation completed successfully.

ICP_BAD_ARG invalid frame – frame pointer returns NULL, *frame* not defined.

ICP_BAD_ARG invalid *x* and/or *y* – *x* > frame width, or, *y* > frame height, or both.

Hardware Modified

None

Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com

icp_clr_frame – Clear a Frame in IC-PCI Memory

Syntax #include <icp.h>

short **icp_clr_frame**(MODCNF *_icpmod_, short _frame_, DWORD _value_);

icpmod	Pointer to the IC-PCI module structure.
frame	The frame to clear: a frame ID returned by icp_create_frame, or defined in the configuration file.
value	Value to write to the selected frame, in the range of 0 to 0xFF for 8-bit frames, 0 to 0xFFFF for 16-bit frames, and 0 to 0x00FFFFFF for 24-bit frames.

Description

The function **icp_clr_frame** sets all locations for the entire *frame* to *value*. To speed up the frame clear, the function creates a 32–bit DWORD from *value* (4 times an 8–bit BYTE value for 8–bit pixels, twice a 16–bit WORD value for 16–bit pixels, or no modification for the DWORD for 32–bit pixels) calls icp_set_lineio, and icp_chdwline. The number of calls is adjusted by the DWORD and size of *value* to include a complete horizontal line.

This is a high level function. The programmer does not need in-depth knowledge of how the IC-PCI hardware and software function.

Returned Values

ICP_NO_ERROR – Operation completed successfully.

ICP_BAD_ARG invalid frame – frame pointer returned NULL, *frame* not defined.

Hardware Modified

None

Example

```
    . . .
*/clear A1 to value 0xAA */
    icp_clr_frame(icpmod, ICP_FRAMEA1, 0xAA);
    . . .
```

icp_clr_roi – Clear a ROI

Syntax #include <icp.h>

short **icp_clr_roi**(MODCNF *_icpmod_, ROIID _roi_, DWORD _value_);

icpmod	Pointer to the IC-PCI module structure.
roi	ROI ID returned by icp_create_roi.
value	Value to write to the selected ROI, in the range of 0 to 0xFF for 8-bit frames, 0 to 0xFFFF for 16-bit frames, and 0 to 0x00FFFFFF for 24-bit frames.

Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com

Description

The function **icp_clr_roi** sets all locations in the defined ROI to *value*. This function calls icp_roi_set_lineio and writes horizontal lines to frame memory.

For 8–bit frames, only the least significant byte is used to clear. For 16–bit frames use the least significant word (does *not* replicate the bottom byte). 24–bit frames use a complete DWORD (most significant byte forced to zero).

This is a high level function. The programmer does not need in-depth knowledge of how the IC-PCI hardware and software function.

Returned Values

ICP_NO_ERROR – Operation completed successfully.

ICP_BAD_ARG invalid ROI – pointer returned NULL, *roi* not defined.

Hardware Modified

None

icp_cp_area – Copy an Area in IC-PCI Memory

Syntax #include <icp.h>

short **icp_cp_area**(MODCNF **icpmod*, short *frame1*, short *x1*, short *y1*, short *dx*, short *dy*, short *frame2*, short *x2*, short *y2*);

icpmod	Pointer to the IC-PCI module structure.
frame1	The frame to read from; “source frame”: frame ID return- ed by icp_create_frame or defined in the configuration file.
frame2	The frame to write to; “destination frame”: frame ID return- ed by icp_create_frame or defined in the configuration file. *frame1* and *frame2* must have identical *depth* and *color.*
x1	Horizontal starting coordinate in source.
y1	Vertical starting coordinate in source.
dx	Width of area in source.
dy	Height of area in source.
x2	Horizontal starting coordinate in destination.
y2	Vertical starting coordinate in destination.

Description

The function **icp_cp_area** copies an area from *frame1* to *frame2*. The original area in *frame* at location *x1*,*y1* and of dimensions *dx* by *dy* is copied to location *x2*,*y2* in *frame2*. This function calls icp_read_area and icp_write_area.

The arguments are checked against the memory size, and cropped to fit. First, *x1+dx* and *y1+dy* are checked against the width and height, and if necessary *dx* or *dy* is cropped to fit. Then *x2+dx* and *y2+dx* (new cropped *dx* and *dy*) is checked against the size, and if necessary *dx* or *dy* is cropped (again) to fit.

Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com

NOTE frame1 *and* frame2 *must have identical depth and color.*

This is a high level function. The programmer does not need in-depth knowledge of how the IC-PCI hardware and software function.

Returned Values

ICP_NO_ERROR – Operation successfully completed.

ICP_BAD_ARG invalid frame – *frame1* or *frame2* undefined, pointer returns NULL.

ICP_BAD_ARG frame depths not equal – *frame1* and *frame2* must have identical depth.

ICP_BAD_ARG frame colors not equal – *frame1* and *frame2* must have identical color.

Hardware Modified

None

icp_cp_roi – Copy an ROI to another ROI

Syntax

#include <icp.h>

short **icp_cp_roi**(MODCNF **icpmod*, ROIID *roi1*, ROIID *roi2*);

icpmod	Pointer to the IC-PCI module structure.
roi1	ROI ID returned by icp_create_roi, the “source” ROI where data is read.
roi1	ROI ID returned by icp_create_roi, the “destination” ROI where data is written. *roi1* and *roi2* must have identical depth, dx, and dy.

Description

The function **icp_cp_roi** copies *roi1* to *roi2*. *roi1* and *roi2* must have identical depth, dx, and dy. dx and dx are defined in icp_create_roi.

This function internally calls icp_read_roi_area and icp_write_roi_area.

This is a high level function. The programmer does not need in-depth knowledge of how the IC-PCI hardware and software function.

Returned Values

ICP_NO_ERROR – Operation completed successfully.

ICP_BAD_ARG invalid ROI – *roi1* or *roi2* not defined, pointer returns NULL.

ICP_BAD_ARG ROI depths not equal – *roi1* and *roi2* must have identical depth.

ICP_BAD_ARG ROI dx not equal – *roi1* and *roi2* must have identical dx (width).

ICP_BAD_ARG ROI dy not equal – *roi1* and *roi2* must have identical dy (height).

Hardware Modified

None

Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com

icp_create_frame – Create a Frame Buffer

Syntax #include <icp.h>

short **icp_create_frame**(MODCNF **icpmod*, WORD *dx*, WORD *dy*, ICP_DEPTH *depth*, ICP_COLOR *color*);

icpmod	Pointer to the IC-PCI module structure.
dx	Maximum width (in pixels) of an acquire
dy	Maximum height (in pixels) of acquire
depth	Number of bits per pixel: ICP_PIX8 – 8 bits per pixel ICP_PIX16 – 16 bits per pixel ICP_PIX24 – 24 bits per pixel ICP_PIX32 – 32 bits per pixel
color	The color format of the frame pixels; ICP_MONO – 8-bit monochrome ICP_RED – 8-bit red from RGB ICP_GREEN – 8-bit green from RGB ICP_BLUE – 8-bit blue from RGB ICP_RGB – 24 bit RGB ICP_YCRCB – 16-bit YCrCb ICP_YCRCBMONO – 16-bit, containing 8-bit monochrome ICP_RGB_PLANAR – 8-bit “planar” image ICP_RGB_PACK24 – 24 bit pixel–packed RGB (requires IC-PCI boards of Revision B or later).

Description

The function **icp_create_frame** creates an IC-PCI frame with the specified attributes, and returns the frame ID. Up to sixteen frames can be created. This function is an alternative way of creating fames. The configuration file can also define frames.

This is a high level function. The programmer does not need in-depth knowledge of how the IC-PCI hardware and software function.

Returned Values

The frame ID.

ICP_BAD_ARG same frame created twice – this frame is already defined.

ICP_BAD_ARG out of frame memory – all frame memory pages already allocated to other frame definitions.

ICP_BAD_ARG out of frame IDs – all frame IDs already used, maximum number of allowed frames (16) exceeded.

Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com

Hardware Modified

None

Example

```
frame1 = icp_create_frame(icpmod, 512, 512, ICP_PIX8, ICP_RED)
```

icp_create_roi – Create a ROI Structure

Syntax #include <icp.h>

ROIID **icp_create_roi**(MODCNF **icpmod*, short *parent_frame*, WORD *x*, WORD *y*, WORD *dx*, WORD *dy*, ICP_DEPTH *depth*, ICP_COLOR *color*);

icpmod	Pointer to the IC-PCI module structure.
parent_frame	The frame that contains this ROI, (a frame ID returned by icp_create_frame).
x	The horizontal coordinate of the ROI within the parent frame.
y	The vertical coordinate of the ROI within the parent frame.
dx	The width (in pixels) of the ROI.
dy	The height (in pixels) of the ROI.
depth	Number of bits per pixel: ICP_PIX8 – 8 bits per pixel. ICP_PIX16 – 16 bits per pixel. ICP_PIX24 – 24 bits per pixel. ICP_PIX32 – 32 bits per pixel.
color	The color format of the frame pixels. ICP_MONO – 8-bit in 8-bit parent. ICP_RED – 8-bit red in 24-bit RGB parent. ICP_GREEN – 8-bit green in 24-bit RGB parent. ICP_BLUE – 8-bit blue in 24-bit RGB parent. ICP_RGB – 24 bit in 24-bit RGB parent.

Description

The function **icp_create_roi** creates a ROI (region of interest) in a frame, and returns an ROIID. The frame parent_frame must already exist, defined by the configuration file or by icp_create_frame. Up to sixteen ROIs can be defined (created) in a frame.

This is a high level function. The programmer does not need in-depth knowledge of how the IC-PCI hardware and software function.

Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com

Returned Values

The ROI ID

NULL – if operation fails.

ICP_NO_ERROR – Operation completed successfully.

ICP_BAD_ARG invalid frame – frame pointer returned NULL; *parent_frame* not defined.

ICP_BAD_ARG ROI does not fit in frame – x > frame width, or y > frame height, or bad depth.

ICP_BAD_ARG frame ROI child limit exceeded – maximum number of allowed ROIs in a frame (16) exceeded.

ICP_BAD_ARG ROI depth at odds with parent – depth and color of ROI must be identical to parent frame, except when the parent frame is 24-bit RGB, then the ROI may have a depth of ICP_PIX8 and point to one color.

Hardware Modified

None

Example

```
roi1 = icp_create_roi(icpmod,frame1,64,32,128,128,
  ICP_PIX8,ICP_MONO)
```

Example

```
roi9 = icp_create_roi(icpmod,ICP_FRAMERGB,
  128,128,255,255,ICP_PIX24,ICP_RGB)
```

icp_create_seq_frames – Create a Sequential Frames Buffer

Syntax #include <icp.h>

short **icp_create_seq_frames**(MODCNF **icpmod*, WORD *framecnt*, WORD **frames*, WORD *dx*, WORD *dy*, ICP_DEPTH *depth*, ICP_COLOR *color*);

icpmod	Pointer to the IC-PCI module structure.
framecnt	Number of sequential frames in this set: range 1 to 8.
frames	Pointer to where to put frame IDs, one word per frame.
dx	Maximum width (in pixels) 0 to 4K.
dy	Maximum height (in pixels) 0 to 4K.
depth	Number of bits per pixel: ICP_PIX8 – 8 bits per pixel ICP_PIX16 – 16 bits per pixel ICP_PIX24 – 24 bits per pixel ICP_PIX32 – 32 bits per pixel

Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com

color The color format of the frame pixels:
ICP_MONO – 8-bit monochrome
ICP_RED – 8-bit red
ICP_GREEN – 8-bit green
ICP_BLUE – 8-bit blue
ICP_RGB – 24 bit RGB

Description

The function **icp_create_seq_frames** creates a sequential set of frames with the specified attributes. icp_seq_snap is the only acquire function used with a sequential frames buffer. You can not delete one frame in the sequential set, the whole set is deleted, with icp_delete_seq_frames.

This is a high level function. The programmer does not need in-depth knowledge of how the IC-PCI hardware and software function.

Returned Values

ICP_NO_ERROR – Operation completed successfully.

ICP_BAD_ARG out of frame memory – all frame memory pages already allocated to other frame definitions.

ICP_BAD_ARG out of frame IDs – maximum number of allowed frames exceeded.

Hardware Modified

None

icp_cvbline – Clear a Vertical Line of BYTE Values

Syntax

#include <icp.h>

void **icp_cvbline**(DWORD *offadr*, short *nwrites*, short *linesz*, DWORD *ilacedata*, BYTE *val*);

offadr The memory address, returned from icp_set_lineio or icp_roi_set_lineio.
nwrites Number of locations to clear.
linesz Horizontal size (in bytes) of the parent frame, returned as *xsz* by icp_set_lineio or icp_roi_set_lineio.
ilacedata Interlace status of the parent frame, returned from icp_set_lineio or icp_roi_set_lineio.
val Value to clear pixels to, or BYTE value to write at each location: 0 to 0xFF.

Description

The function **icp_cvbline** clears a vertical line memory to a BYTE value *val*. You must select a value for *nwrites* that does not exceed the bounds of the frame containing the line.

This is a medium level function. The programmer should be familiar with the IC-PCI hardware and software functional design.

Returned Values

None

Hardware Modified

None

Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com

icp_cvdwline – Clear a Vertical Line of DWORD Values

Syntax #include <icp.h>

void **icp_cvdwline**(DWORD *offadr*, short *nwrites*, short *linesz*, DWORD *ilacedata*, DWORD *val*);

offadr	The memory address returned from icp_set_lineio or icp_roi_set__lineio.
nwrites	Number of locations to clear.
linesz	Horizontal size (in bytes) of the parent frame, returned as *xsz* by icp_set_lineio or icp_roi_set_lineio.
ilacedata	Interlace status of the parent frame, returned from icp_set_lineio or icp_roi_set_lineio.
val	Value to clear pixels to, or DWORD value to write at each location: 0 to oxFFFFFFFF.

Description

The function **icp_cvdwline** clears a vertical line in image memory to a DWORD value *val*. You must select a value for *nwrites* that does not exceed the bounds of the frame containing the line. If the pixel size is 24–bits (ICP_PIX24) each DWORD clears one pixel. If the pixel size is 16-bits (ICP_PIX16) each DWORD clears two horizontally adjacent 16-bit pixels. If the pixel size is 8-bits (ICP_PIX8) each DWORD clears four horizontally adjacent 8-bit pixels.

This is a medium level function. The programmer should be familiar with the IC-PCI hardware and software functional design.

Returned Values

None

Hardware Modified

None

icp_cvwline – Clear a Vertical Line of WORD Values

Syntax #include <icp.h>

void **icp_cvwline**(DWORD *offadr*, short *nwrites*, short *linesz*, DWORD *ilacedata*, WORD *val*);

offadr	The memory address returned from icp_set_lineio or icp_roi_set__lineio.
nwrites	Number of locations to clear.
linesz	Horizontal size (in bytes) of the parent frame, returned as *xsz* by icp_set_lineio or icp_roi_set_lineio.
ilacedata	Interlace status of the parent frame, returned from icp_set_lineio or icp_roi_set_lineio.
val	Value to clear pixels to, or WORD value to write at each location: 0 to 0xFFFF.

Description

The function **icp_cvwline** clears a vertical line in image memory to a WORD value. You must select a value for *nwrites* that does not exceed the bounds of the frame containing the line. If the pixel size is 16–bits (ICP_PIXSZ16) each WORD clears one 16–bit pixel. If the pixel size is 8–bits (ICP_PIXSZ8) each WORD clears two horizontally adjacent 8–bit pixels.

Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com

This is a medium level function. The programmer should be familiar with the IC-PCI hardware and software functional design.

Returned Values

None

Hardware Modified

None

icp_delete_all_frame_rois – Delete All ROIs in a Frame Buffer

Syntax #include <icp.h>

WORD **icp_delete_all_frame_rois**(MODCNF **icpmod*, short *frame*);

icpmod	Pointer to the IC-PCI module structure.
frame	The frame that contains the ROIs: a frame ID returned by icp_create_frame or defined by the configuration file.

Description

The function deletes all defined ROIs in a specified parent frame.

This is a high level function. The programmer does not need in-depth knowledge of how the IC-PCI hardware and software function.

Returned Values

ICP_NO_ERROR – Operation completed successfully.

ICP_BAD_ARG invalid frame – frame pointer returned NULL; frame not defined.

Hardware Modified

None

icp_delete_all_frames – Delete All Frame Buffers and their ROIs

Syntax #include <icp.h>

WORD **icp_delete_all_frames**(MODCNF **icpmod*);

icpmod	Pointer to the IC-PCI module structure.

Description

The function **icp_delete_all_frames** deletes all frames and child ROIs for the IC-PCI specified by *icpmod*.

This is a high level function. The programmer does not need in-depth knowledge of how the IC-PCI hardware and software function.

Returned Values

ICP_NO_ERROR – Operation completed successfully.

Hardware Modified

None

Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com

icp_delete_frame – Delete a Frame Buffer

Syntax #include <icp.h>

WORD **icp_delete_frame**(MODCNF **icpmod*, short *frame*);

icpmod	Pointer to the IC-PCI module structure.
frame	The frame to delete: a frame ID returned by icp_create_frame or defined by the configuration file.

Description

The function **icp_delete_frame** deletes a frame and all child ROIs. This routine can not delete sequential frames.

This is a high level function. The programmer does not need in-depth knowledge of how the IC-PCI hardware and software function.

Returned Values

ICP_NO_ERROR – Operation completed successfully.

ICP_BAD_ARG invalid frame – *frame* returned a NULL pointer.

ICP_BAD_ARG can't delete seq frame by itself – *frame* indicated a sequential frame, you must use icp_delete_seq_frames to delete entire sequence.

Hardware Modified

None

icp_delete_roi – Delete a ROI Structure

Syntax #include <icp.h>

WORD **icp_delete_roi**(MODCNF **icpmod*, ROIID *roi*);

icpmod	Pointer to the IC-PCI module structure.
roi	The ROI ID (returned by icp_create_roi)

Description

The function **icp_delete_roi** deletes a specified ROI, and frees the ROI ID. This function deletes only one ROI.

This is a high level function. The programmer does not need in-depth knowledge of how the IC-PCI hardware and software function.

Returned Values

ICP_NO_ERROR – Operation completed successfully.

Hardware Modified

None

Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com

icp_delete_seq_frame – Delete a Sequential Frames Buffer

Syntax #include <icp.h>

WORD **icp_delete_seq_frame**(MODCNF **icpmod*, WORD *framecnt*, WORD **frames*);

icpmod	Pointer to the IC-PCI module structure.
framecnt	Number of frames in sequence, listed in *frames*
frames	Pointer to list of sequential frames.

Description

The function **icp_delete_seq_frame** deletes sequential frames and all child ROIs. This routine must be used to delete sequential frames created by icp_create_seq_frames

This is a high level function. The programmer does not need in-depth knowledge of how the IC-PCI hardware and software function.

Returned Values

ICP_NO_ERROR – Operation completed successfully.

ICP_BAD_ARG invalid frame – *frame* returned a NULL pointer.

ICP_BAD_ARG can't delete seq frame by itself – *frame* indicated a sequential frame, you must use icp_delete_seq_frames to delete entire sequence.

Hardware Modified

None

icp_display_area – Define an Area for Display

Syntax #include <icp.h>

short **icp_display_area**(MODCNF **icpmod*, short *frame*, short *x*, short *y*, short *dx*, short *dy*, ICP_ACQTYP *dispmode*);

icpmod	Pointer to the IC-PCI module structure.
frame	The frame to use for display: a frame ID returned by icp_create_frame or defined in the configuration file.
x	The horizontal coordinate of the area within *frame*
y	The vertical coordinate of the area within *frame*.
dx	The width (in pixels) of the area.
dy	The height (in pixels) of the area.
dispmode	Type of display transfer: ICP_SNAP – display a single frame. ICP_GRAB – display continuously. ICP_FREEZE – terminate continuous display.

Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com

Description

The function **icp_display_area** sends parameters of an area in a defined *frame* to an IC-PCI display utility (included on the ICP distribution diskettes). The area is a subsection of a *frame*. The size is clipped to fit the frame memory.

This is a high level function. The programmer does not need in-depth knowledge of how the IC-PCI hardware and software function.

Returned Values

ICP_NO_ERROR – Operation completed successfully.

ICP_BAD_ARG invalid frame – *frame* not defined, pointer returns NULL.

ICP_BAD_ARG invalid x and/or y – either x > frame width, or, y > frame height, or both.

Hardware Modified

ICP_MB_HSHK_32, ICP_MB_FXSZ_32, ICP_BM_ILACE_32, ICP_BM_FYSZ_32, ICP_BM_PIX_32, ICP_BM_AOIXST_32, ICP_BM_AOIYST_32, ICP_BM_AOIXSZ_32, ICP_BM_AOIYSZ_32, ICP_BM_STFLD_32, ICP_BM_IMGADR_32, ICP_BM_CONT_32

icp_display_frame – Define a Frame for Display

Syntax

#include <icp.h>

short **icp_display_frame**(MODCNF **icpmod*, short *frame*, ICP_ACQTYP *dispmode*);

icpmod	Pointer to the IC-PCI module structure.
frame	The frame to use for display: a frame ID returned by icp_create_frame, or defined in the configuration file.
dispmode	Type of display transfer: ICP_SNAP – display a single frame. ICP_GRAB – display continuously. ICP_FREEZE – terminate continuous display.

Description

The function **icp_display_frame** sends the parameters of a defined *frame* to an IC-PCI display routine (included on the distribution diskettes).

This is a high level function. The programmer does not need in-depth knowledge of how the IC-PCI hardware and software function.

Returned Values

ICP_NO_ERROR – Operation completed successfully.

ICP_BAD_ARG invalid frame – *frame* not defined, pointer returns NULL.

Hardware Modified

ICP_MB_HSHK_32, ICP_MB_FXSZ_32, ICP_BM_ILACE_32, ICP_BM_FYSZ_32, ICP_BM_PIX_32, ICP_BM_AOIXST_32, ICP_BM_AOIYST_32, ICP_BM_AOIXSZ_32, ICP_BM_AOIYSZ_32, ICP_BM_STFLD_32, ICP_BM_IMGADR_32, ICP_BM_CONT_32

Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com

icp_display_roi – Define a ROI for Display

Syntax #include <icp.h>

short **icp_display_roi**(MODCNF **icpmod*, ROIID *roi*, ICP_ACQTYP *dispmode*);

icpmod	Pointer to the IC-PCI module structure.
roi	ROI ID returned by icp_create_roi.
dispmode	Type of display transfer: ICP_SNAP – display a single frame. ICP_GRAB – display continuously. ICP_FREEZE – terminate continuous display.

Description

The function **icp_display_roi** sends the parameters of a defined ROI to an IC-PCI display routine (included on the distribution diskettes).

This is a high level function. The programmer does not need in-depth knowledge of how the IC-PCI hardware and software function.

Returned Values

ICP_NO_ERROR – Operation completed successfully.

ICP_BAD_ARG invalid ROI – *roi* not defined, pointer returns NULL.

Hardware Modified

ICP_MB_HSHK_32, ICP_MB_FXSZ_32, ICP_BM_ILACE_32, ICP_BM_FYSZ_32, ICP_BM_PIX_32, ICP_BM_AOIXST_32, ICP_BM_AOIYST_32, ICP_BM_AOIXSZ_32, ICP_BM_AOIYSZ_32, ICP_BM_STFLD_32, ICP_BM_IMGADR_32, ICP_BM_CONT_32

icp_display_roi_area – Define a ROI Area for Display

Syntax #include <icp.h>

short **icp_display_roi_area**(MODCNF **icpmod*, ROIID *roi*, short *x*, short *y*, short *dx*, short *dy*, ICP_ACQTYP *dispmode*);

icpmod	Pointer to the IC-PCI module structure.
roi	ROI ID returned by icp_create_roi.
x	The horizontal coordinate of the area within *frame*
y	The vertical coordinate of the area within *frame*.
dx	The width (in pixels) of the area.
dy	The height (in pixels) of the area.
dispmode	Type of display transfer: ICP_SNAP – display a single frame. ICP_GRAB – display continuously. ICP_FREEZE – terminate continuous display.

Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com

Description

The function **icp_display_roi_area** sends parameters of an area in a defined ROI to an IC-PCI display utility (included on the ICP distribution diskettes). The area is a subsection of a *frame*. The size is clipped to fit the frame memory if necessary.

This is a high level function. The programmer does not need in-depth knowledge of how the IC-PCI hardware and software function.

Returned Values

ICP_NO_ERROR – Operation completed successfully.

ICP_BAD_ARG invalid ROI – *roi* not defined, pointer returns NULL.

ICP_BAD_ARG invalid x and/or y – either
x > ROI width, or, y > ROI height, or both.

Hardware Modified

ICP_MB_HSHK_32, ICP_MB_FXSZ_32, ICP_BM_ILACE_32, ICP_BM_FYSZ_32, ICP_BM_PIX_32, ICP_BM_AOIXST_32, ICP_BM_AOIYST_32, ICP_BM_AOIXSZ_32, ICP_BM_AOIYSZ_32, ICP_BM_STFLD_32, ICP_BM_IMGADR_32, ICP_BM_CONT_32

icp_dregs – Display the IC-PCI Register Contents

Syntax

#include <icp.h>

void **icp_dregs**(MODCNF **icpmod*);

icpmod Pointer to the IC-PCI module structure.

Description

The function **icp_dregs** displays the contents of all hardware registers in the IC-PCI module.

This is a high level function. The programmer does not need in-depth knowledge of how the IC-PCI hardware and software function.

Returned Values

None

Hardware Modified

None

icp_dump_mem – Display (dump) an Area of Memory

Syntax

#include <icp.h>

short **icp_dump_mem**(MODCNF **icpmod*, short *frame*, short *x*, short *y*, short *dx*, short *dy*);

icpmod Pointer to the IC-PCI module structure.

frame The frame to report on: a frame ID returned by icp_create_frame, or defined by the configuration file.

Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com

x	Horizontal starting location in *frame*.
y	Vertical starting location in *frame.*
dx	Horizontal width of area to dump.
dy	Vertical height of area to dump.

Description

The function **icp_dump_mem** displays a block of frame memory in hexadecimal format. Frame memory is displayed in the pixel depth for *frame*. This function calls icp_set_lineio and icp_rhdwline.

This is a high level function. The programmer does not need in-depth knowledge of how the IC-PCI hardware and software function.

Returned Values

NULL if *frame* is not defined, or *x* is larger than xsz of *frame*, or *y* is larger than ysz of *frame*.

ICP_NO_ERROR – Operation successfully completed.

Hardware Modified

None

icp_dump_roi_mem – Display (dump) a Memory ROI

Syntax #include <icp.h>

short **icp_dump_roi_mem**(MODCNF **icpmod*, ROIID *roi*);

icpmod	Pointer to the IC-PCI module structure.
roi	ROI ID (returned by icp_create_roi).

Description

The function **icp_dump_roi_mem** displays a ROI of frame memory in hexadecimal format.

This is a high level function. The programmer does not need in-depth knowledge of how the IC-PCI hardware and software function.

Returned Values

ICP_NO_ERROR – Operation successfully completed.

ICP_BAD_ARG invalid ROI – *roi* not defined, or NULL

Hardware Modified

None

icp_fcount – Set or Return the Acquisition Frame Count

Syntax #include <icp.h>

short **icp_fcount**(MODCNF **icpmod*, short *count*);

icpmod	Pointer to the IC-PCI module structure.

Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com

count Number of frames to acquire in a sequence,
range 1 to 8
INQUIRE – Return the current value: ICP_FCNT+1

Description

The function **icp_fcount** programs the number of frames in a sequential acquisition. The sequential frames must be defined by icp_create_seq_frames. The number *count* can be less than *framecnt* specified in icp_create_seq_frames but cannot be greater. This function modifies the IC-PCI module structure.

This is a low level function. The programmer should have detailed knowledge of how the IC-PCI hardware and software function.

Returned Values

The number of frames acquired by the last acquisition operation.

ICP_NO_ERROR – Operation completed successfully.

Hardware Modified

ICP_FCNT

icp_field_status – Return the Camera Field Status

Syntax

#include <icp.h>
short **icp_field_status**(MODCNF **icpmod*);

icpmod Pointer to the IC-PCI module structure.

Description

The function **icp_field_status** reads the camera field status in ICP_FLDSTAT.

This is a low level function. The programmer should have detailed knowledge of how the IC-PCI hardware and software function.

Returned Values

ICP_EVEN – camera is in even field.

ICP_ODD – camera is in odd field.

Hardware Modified

None

Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com

icp_fifo_reset – Reset (Clear) the FIFO

Syntax #include <icp.h>
short **icp_fifo_reset**(MODCNF **icpmod*, short *mode*);

icpmod	Pointer to the IC-PCI module structure.
mode	The reset status to write: ICP_RESET – Perform a FIFO reset ICP_NO_RESET – Do not perform a reset

Description

The function **icp_fifo_reset** writes to the FIFO reset bit.

This is a low level function. The programmer should have detailed knowledge of how the IC-PCI hardware and software function.

Returned Values

ICP_NO_ERROR – Operation completed successfully.

Hardware Modified

ICP_FIFOCLR_32

icp_frame_attrib – Read Attributes of a Frame Buffer

Syntax #include <icp.h>
WORD **icp_frame_attrib**(MODCNF **icpmod*, short *frame*, WORD **maxdx*, WORD **maxdy*, WORD **curdx*, WORD **curdy*, WORD **pxdepth*, WORD **color*);

icpmod	Pointer to the IC-PCI module structure.
frame	The frame to write to: a frame ID returned by icp_create_frame, or defined by the configuration file.
maxdx	Maximum horizontal size for acquisition into frame, a returned value.
maxdy	Maximum vertical size for acquisition into frame, a returned value.
curdx	Current width of the image in frame, a returned value.
curdy	Current height of the image in frame, a returned value.
pxdepth	Number of bits per pixel defined for frame, a returned value: ICP_PIX8, ICP_PIX16, ICP_PIX24.
color	Color mode for *frame*, a returned value: ICP_MONO, ICP_RED, ICP_GREEN, ICP_BLUE, ICP_RGB.

Description

Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com

The function **icp_frame_attrib** reads the current attributes of a frame. The values for *maxdx*, *maxdy*, *pxdepth*, and *color* are programmed by icp_create_frame.

This is a high level function. The programmer does not need in-depth knowledge of how the IC-PCI hardware and software function.

Returned Values

maxdx, *maxdy*, *curdx*, *curdy*, *pxdepth*, *color*

ICP_NO_ERROR – Operation completed successfully.

ICP_BAD_ARG invalid frame – *frame* not defined, pointer returned NULL.

Hardware Modified

None

icp_frame_exist – Test if a Frame Buffer Already Exists

Syntax

#include <icp.h>

WORD **icp_frame_exist**(MODCNF **icpmod*, short *frame*);

icpmod	Pointer to the IC-PCI module structure.
frame	The frame ID to validate or report on.

Description

The function **icp_frame_exist** verifies that *frame* is already defined.

This is a high level function. The programmer does not need in-depth knowledge of how the IC-PCI hardware and software function.

Returned Values

TRUE – frame already exists

FALSE – frame does not exist.

Hardware Modified

None

icp_freeze – Stop Acquisition

Syntax

#include <icp.h>

short **icp_freeze**(MODCNF **icpmod*, short *frame*);

icpmod	Pointer to the IC-PCI module structure.
frame	The frame ID: a frame ID returned by icp_create_frame, or defined by the configuration file.

Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com

Description

The function **icp_freeze** stops acquisition into *frame* at the beginning of the next vertical blank that completes a full image frame (not the next field in an interlaced image).

This is a high level function. The programmer does not need in-depth knowledge of how the IC-PCI hardware and software function.

Returned Values

ICP_NO_ERROR – Operation completed successfully.

ICP_BAD_ARG invalid frame – *frame* not defined.

ICP_TIMEOUT can't freeze ACQ bits – Could not freeze *frame*.

Hardware Modified

ICP_ACQMD

NOTE *This function is also available as a system-level function. Using the system-level call makes your application more portable.*

icp_get_acq_dim – Return Dimensions of Camera Frame

Syntax

#include <icp.h>

short **icp_get_acq_dim**(MODCNF **icpmod*, WORD **camdx*, WORD **camdy*);

icpmod Pointer to the IC-PCI module structure.

camdx Pointer to the camera frame width, a returned value.

camdy Pointer to the camera frame height, a returned value.

Description

The function **icp_get_acq_dim** returns the camera frame dimensions for the current or "active" camera port from the AM module's data structure. This function is used to get the dimensions of the camera frame, and use them as input to the function icp_set_cur_dim.

Returned Values

ICP_BAD_ARG invalid mod pointer – *icpmod* not defined.

ICP_BAD_DMODCNF invalid AM mod pointer – *ammod* not defined.

Hardware Modified

None

NOTE *Function added in version 2.6.2 for DOS32x and Win16, and version 2.6.1 for Win32.*

icp_get_acq_start_addr – Return the Camera Acquire Start Address

Syntax

#include <icp.h>

DWORD **icp_get_acq_start_addr**(MODCNF **icpmod*);

icpmod Pointer to the IC-PCI module structure.

Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com

Description

The function **icp_get_acq_start_addr** returns the starting address used for acquisition from a camera. The memory is addressable on 4K boundaries (4096). The acquisition can begin writing at any one of the 4K boundaries, as set by icp_put_start_addr.

This is a low level function. The programmer should have detailed knowledge of how the IC-PCI hardware and software function.

Returned Values

The starting address in ICP_AQSTART

Hardware Modified

None

icp_get_active_ubm_frame – Return a Pointer to the Active "Unattended Bus-Mastered" Frame

Syntax #include <icp.h>

short **icp_get_active_ubm_frame**(MODCNF *_icpmod_, DWORD _context_, short *_cur_acq_frame_);

icpmod	Pointer to the IC-PCI module structure.
context	The context, returned by itx_start_ping_pong.
cur_acq_frame	Pointer to the total number of frames acquired on the current pass.

Description

The function **icp_get_active_ubm_frame** returns the host frame ID of the host frame currently being acquired into, in an "Unattended Bus-Mastered" operation.

This is a high level function. The programmer does not need in-depth knowledge of how the hardware and software function.

Returned Values

The Host Frame ID for the active frame, passed as *host_fr1* or *host_fr2* in itx_start_ping_pong. Returns NULL on failure.

ICP_BAD_ARG UBM service failure – error from TSR; Failure in "unattended bus master" transfer.

Hardware Modified

None.

NOTE *ITEX software revision 2.7.1 redefines this function as a system-level (itx_) function. ITEX revision 2.6.3 for DOS and Windows 3.1, defines this function as a board-level (icp_) function.*

Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com

icp_get_bm_aoix – Return the Bus-Master-Mode AOI Horizontal Size

Syntax #include <icp.h>

WORD **icp_get_bm_aoix**(MODCNF **icpmod*, short *pixsz*);

icpmod	Pointer to the IC-PCI module structure.
pixsz	Pixel size to use: ICP_PIX8 – 8-bit pixels. ICP_PIX16 – 16-bit pixels. ICP_PIX24 – 24-bit pixels.

Description

The function **icp_get_bm_aoix** reads the bus master horizontal AOI size in bytes in ICP_AOIX and calculates the number of pixels based on *pixsz*.

This is a low level function. The programmer should have detailed knowledge of how the IC-PCI hardware and software function.

Returned Values

Bus master transfer horizontal size, in *pixsz* pixels

Hardware Modified

None

icp_get_bm_dst_addr – Return the Bus-Master-Mode Destination Address

Syntax #include <icp.h>

DWORD **icp_get_bm_dst_addr**(MODCNF **icpmod*);

icpmod	Pointer to the IC-PCI module structure.

Description

The function **icp_get_bm_dst_addr** reads and returns the destination physical address programmed by icp_put_bm_dst_addr used by the IC-PCI for a bus master transfer (write) to another memory resource, usually host memory or VGA display memory.

This is a low level function. The programmer should have detailed knowledge of how the IC-PCI hardware and software function.

Returned Values

Destination (write) physical address.

Hardware Modified

None

Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com

icp_get_bm_firstf_start – Return the First Field Starting Address of Bus-Master Transfer

Syntax #include <icp.h>
DWORD **icp_get_bm_firstf_start**(MODCNF **icpmod*);

icpmod Pointer to the IC-PCI module structure.

Description

The function **icp_get_bm_firstf_start** reads and returns the first field starting address in IC-PCI memory, programmed by icp_put_bm_firstf_start. The application must provide the starting address of the first and second fields of data for a bus master transfer.

This is a low level function. The programmer should have detailed knowledge of how the IC-PCI hardware and software function.

Returned Values

Starting address in IC-PCI memory of first field acquired

Hardware Modified

None

icp_get_bm_secondf_start – Return the Second Field Starting Address of Bus-Master Transfer

Syntax #include <icp.h>
DWORD **icp_get_bm_secondf_start**(MODCNF **icpmod*);

icpmod Pointer to the IC-PCI module structure.

Description

The function **icp_get_bm_secondf_start** reads and returns the second field starting address in IC-PCI memory, programmed by icp_put_bm_secondf_start. The application must provide the starting address of the first and second fields of data for a bus master transfer.

This is a low level function. The programmer should have detailed knowledge of how the IC-PCI hardware and software function.

Returned Values

Starting address in IC-PCI memory of second field acquired

Hardware Modified

None

Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com

icp_get_hw_revision – Get the Hardware Revision ID

Syntax #include <icp.h>

short **icp_get_hw_revision**(MODCNF **icpmod*);

icpmod Pointer to the IC-PCI module structure.

Description

The function **icp_get_hw_revision** reads the hardware revision ID field, and returns the value.

NOTE *New function added in ITEX revision 2.8.0 for Windows NT and Windows 95. Not applicable to Win16 DOS32x or Windows 3.11.*

Returned Values

Number value of hardware revision

Hardware Modified

None

Example

```
boardrev=icp_get_hw_revision(ipcmod);
```

icp_get_idle_ubm_frame – Return a Pointer to the Idle "Unattended Bus-Mastered" Frame

Syntax #include <icp.h>

short **icp_get_idle_ubm_frame**(MODCNF **icpmod*, DWORD *context*, short **cur_acq_frame*);

icpmod Pointer to the IC-PCI module structure.

context The context, returned by itx_start_ping_pong.

cur_acq_frame Pointer to the total number of frames acquired on the current pass.

Description

The function **icp_get_idle_ubm_frame** returns the host frame id of the host frame currently not being acquired into, in an "Unattended Bus-Mastered" operation. This function calls itx_get_active_ubm_frame.

This is a high level function. The programmer does not need in-depth knowledge of how the hardware and software function.

Returned Values

A pointer to the Frame ID for the idle frame, passed as *host_fr1* or *host_fr2* in itx_start_ping_pong. Returns NULL on failure.

ICP_BAD_ARG UBM service failure – error from TSR; Failure in "unattended bus master" transfer.

Hardware Modified

None.

NOTE *ITEX software revision 2.7.1 redefines this function as a system-level (itx_) function. ITEX revision 2.6.3 for DOS and Windows 3.1, defines this function as a board-level (icp_) function.*

Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com

icp_get_slot0 – Return Pointer to the AM Module Structure

Syntax #include <icp.h>
pMODCNF **icp_get_slot0**(MODCNF **icpmod*);

icpmod Pointer to the IC-PCI module structure.

Description

The function **icp_get_slot0** returns a pointer to the AM on the IC-PCI.

This is a medium level function. The programmer should be familiar with the IC-PCI hardware and software functional design.

Returned Values

Pointer to the AM MODCNF structure. Returns NULL on failure.

ICP_BAD_MODCNF – *icpmod* is not valid.

ICP_BAD_DMODCNF Bad daughter board MODCNF structure pointer – No AM detected in Slot 0, or not defined in Configuration file; pointer returns NULL.

Hardware Modified

None

icp_grab – Acquire Images Continuously

Syntax #include <icp.h>
short **icp_grab**(MODCNF **icpmod*, short *frame*);

icpmod Pointer to the IC-PCI module structure.

frame The frame to write to: a frame ID returned by icp_create_frame, or defined by the configuration file.

Description

The function **icp_grab** sets up *frame* to continuously acquire images. This function waits for a vertical blank, writes to the acquisition command bits. Use icp_freeze to stop the acquisition.

This is a high level function. The programmer does not need in-depth knowledge of how the IC-PCI hardware and software function.

Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com

Returned Values

ICP_NO_ERROR – Operation completed successfully.

ICP_BAD_ARG invalid frame – *frame* not defined, pointer returns NULL.

ICP_BAD_ARG invalid mod pointer – pointer to IC-PCI returns NULL.

ICP_BAD_ARG can't acquire seq frame by itself – frame ID points to sequential frames, use icp_snap_seq to acquire into sequential frames.

ICP_BAD_ARG invalid AM mod ptr – AM not defined, pointer returns NULL.

ICP_OUT_FB_MEM – IC-PCI memory full

ITX_TIMEOUT – stuck in vertical blank

Hardware Modified

ICP_ACQMD

icp_host_large_format_acquire – Acquire Large Format Image(s) to Host Memory

Syntax

#include <icp.h>

short **icp_host_large_format_acquire**(MODCNF **mod*, short *host_frame_id*);

icpmod Pointer to the IC-PCI module structure.

host_frame_id Id for the host frame, a pointer returned by itx_create_large_format_hframe.

Description

The function **icp_host_large_format_acquire** initiates acquisition of an image from a large format camera into host memory. This function can use the entire frame buffer; any frames previously written to the buffer will be overwritten.

This is a high level function. The programmer does not need in-depth knowledge of how the hardware and software function.

A host frame exists for acquiring the large frame which consumes all of the frame buffer. This is necessary since the acquire wrap-around wraps to the beginning of the frame buffer.

A special large frame snap function exists which does not require that the acquisition dimensions fit within the allotted frame buffer dimensions.

Returned Values

ICP_NO_ERROR – Operation completed successfully.

ICP_BAD_ARG invalid host frame – *host_frame_id* is not defined.

ICP_BAD_ARG invalid frame – *host_frame_id* points to an IC-PCI frame that is not defined, or has been deleted.

Hardware Modified

None

NOTE *ITEX software revision 2.7.1 redefines this function as a system-level function. ITEX revision 2.6.3 for DOS and Windows 3.1, defines this function as a board-level (icp_) function.*

Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com

icp_host_large_format_wacquire – Acquire a Large Format Image(s) to Host Memory, with Wait

Syntax #include <itxcore.h>

short **icp_host_large_format_wacquire**(MODCNF **icpmod*, short *host_frame_id*, long *waittim*e);

icpmod Pointer to the IC-PCI module structure.

host_frame_id Id for the host frame, a pointer returned by itx_create_large_format_hframe

waittime Time to wait for acquire to complete; in miliseconds (ms).

Description

The function **icp_host_large_format_wacquire** initiates acquisition of an image from a large format camera into host memory. This function can use the entire frame buffer; any frames previously written to the buffer will be overwritten. The IC-PCI frame should be sized to the maximum available. The host frame size should be sized to the large format camera frame size. The Acquisition Module Horizontal Active and Vertical Active registers may limit the size of the acquired image.

The wait time is the expected time it takes to acquire this large frame. The unit is miliseconds. 2000 equals 2 seconds.

This is a high level function. The programmer does not need in-depth knowledge of how the hardware and software function.

A host frame exists for acquiring the large frame which consumes all of the frame buffer. This is necessary since the acquire wrap-around wraps to the beginning of the frame buffer.

Returned Values

ICP_NO_ERROR – Operation completed successfully.

ICP_BAD_ARG invalid host frame – *host_frame_id* is not defined.

ICP_BAD_ARG invalid frame – *host_frame_id* points to an IC-PCI frame that is not defined, or has been deleted.

Hardware Modified

None

NOTE *New function added in ITEX revision 2.8.0 for Windows NT and Windows 95. Not applicable to Win16 DOS32x or Windows 3.11.*

Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com

icp_image_pitch – Set or Return the Image Pitch for Bus-Master Transfers

Syntax #include <icp.h>

short **icp_image_pitch**(MODCNF **icpmod*, short *pitch*, short *pixsz*);

icpmod	Pointer to the IC-PCI module structure.
pitch	Horizontal size (in pixels): 8 to 32760 for 8-bit images 4 to 16380 for 16-bit images 2 to 8190 for 24-bit images INQUIRE – return the current value.
pixsz	Pixel depth: ICP_PIX8 – 8-bit image ICP_PIX16 – 16-bit image ICP_PIX24 –24-bit image INQUIRE – use the current depth value

Description

The function **icp_image_pitch** programs (or returns) the horizontal dimension and pixel size of image data transferred by the IC-PCI in bus master mode. The *pitch* must match the horizontal size of the acquired image. The pixel size *pixsz* must match the depth of the acquired image. If pitch is a valid number and *pixsz* is INQUIRE, the current pixel depth is used.

Any area transferred in bus master mode must be a multiple of 64 bytes: (pitch × pixel size) must be multiples of 64.

This is a low level function. The programmer should have detailed knowledge of how the IC-PCI hardware and software function.

Returned Values

The current pitch value, if *pitch* = INQUIRE

ICP_NO_ERROR – Operation completed successfully.

Hardware Modified

ICP_PIXSZ, ICP_PITCH

icp_iregs – Initialize the IC-PCI Registers

Syntax #include <icp.h>

void **icp_iregs**(MODCNF **icpmod*);

icpmod	Pointer to the IC-PCI module structure.

Description

Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com

The function **icp_iregs** initializes the IC-PCI registers to a predefined state. Use the function icp_dregs to display the initial register values. This function halts bus master mode, and clears the FIFO.

This is a low level function. The programmer should have detailed knowledge of how the IC-PCI hardware and software function.

Returned Values

None

Hardware Modified

All the IC-PCI registers

icp_large_format_snap – Acquire Single Large- Format Image into Host Memory

Syntax

#include <icp.h>

short **icp_large_format_snap**(MODCNF **icpmod*, short *hframe*);

icpmod Pointer to the IC-PCI module structure.

hframe ID of a large-format host frame.

Description

The function **icp_large_format_snap** writes to the ACQ bits to perform a snap. This snap can fill all the frame buffer and then continue acquiring after wrapping around to the beginning of the frame buffer. This function does not require that the frame fit into the frame buffer.

This is a low level function. The programmer should have detailed knowledge of how the IC-PCI hardware and software function.

Returned Values

ICP_NO_ERROR – Operation completed successfully.

ICP_TIMEOUT GRAB status did not self-clear – Acquisition status bits did not clear, acquisition did not complete.

ICP_BAD_ARG invalid mod pointer – *mod* not defined.

ICP_BAD_ARG invalid frame – *hframe* not defined.

ICP_BAD_DMODCNF invalid AM mod pointer – *ammod* not defined.

ICP_OUT_FB_MEM ACQ dimension exceeds frame – size of acquisition frame is larger than host frame.

Hardware Modified

ICP_ACQMD

Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com

icp_large_format_trig_snap – Trigger-Driven Large- Format Image Acquisition into Host Memory

Syntax #include <icp.h>

short **icp_large_format_trig_sna**p(MODCNF **icpmod*, short *hframe*);

icpmod	Pointer to the IC-PCI module structure.
hframe	ID of a large-format host frame.

Description

The function **icp_large_format_trig_snap** is a triggered version of the function icp_large_format_snap. This snap can fill all the frame buffer and then continue acquiring after wrapping around to the beginning of the frame buffer. This function does not require that the frame fit into the frame buffer.

This is a high level function. The programmer does not need in-depth knowledge of how the hardware and software function.

Returned Values

ICP_NO_ERROR – Operation completed successfully.

ICP_BAD_ARG invalid mod pointer – *mod* not defined.

ICP_BAD_ARG invalid frame – *hframe* not defined.

ICP_BAD_DMODCNF invalid AM mod pointer – *ammod* not defined.

ICP_OUT_FB_MEM ACQ dimension exceeds frame – size of acquisition frame is larger than host frame.

Hardware Modified

ICP_ACQMD

NOTE *New function added in ITEX revision 2.7.1 and 2.7.2, not available in revisions 2.7.0 or 2.6.3.*

icp_latency_timer – Set or Return the Bus-Master Latency Timer

Syntax #include <icp.h>

short **icp_latency_timer**(MODCNF **icpmod*, short *mode*);

icpmod	Pointer to the IC-PCI module structure.
mode	Latency timer value, in PCI clocks: 0 to 248; must be a multiple of 8. INQUIRE – Return the current value.

Description

The function **icp_latency_timer** programs the PCI bus latency timer used by the IC-PCI in bus master mode transfers. The latency timer value can also be specified in the configuration file. Programming a latency time guarantees the IC-PCI will be granted the bus during the bus master operation for the programmed amount of

Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com

time. During the latency time period the IC-PCI will ignore all PCI bus interrupts or requests; the IC-PCI bus master transfer cannot be interrupted during the latency time period. This function modifies the IC-PCI module structure.

This is a low level function. The programmer should have detailed knowledge of how the IC-PCI hardware and software function.

Returned Values

Value of the latency timer; 0 to 248.

ICP_NO_ERROR – Operation successfully completed.

Hardware Modified

None

icp_line_reverse_read – Read an Area in Reverse Line Order

Syntax

#include <icp.h>

short **icp_line_reverse_read**(MODCNF **icpmod*, short *frame*, short *x*, short *y*, short *dx*, short *dy*, short *destlinebytes*, BYTEHUGE **destadr*);

icpmod	Pointer to the IC-PCI module structure.
frame	Frame to read from: a frame ID returned by icp_create_frame, or defined by the configuration file.
x	Horizontal starting location of the area read in *frame*.
y	Vertical starting location of the area read in *frame*.
dx	Width of area read in *frame*; number of pixels.
dy	Height of area read in *frame*; number of lines.
destlinebytes	Number of bytes in the destination line.
destadr	Pointer to a destination physical address to transfer to.

Description

The function **icp_line_reverse_read** reads lines from a frame in reverse order. A horizontal line is still read from left to right; however, the line order is reversed, (read from bottom to top). *destlinebytes* is the “pitch” of the rectangular destination. This function is useful for filling in DIB structures.

Returned Values

ICP_BAD_ARG invalid frame – *frame* is not defined, pointer returned NULL.

ICP_BAD_ARG invalid x and/or y – either x > frame width, or, y > frame height, or both.

ICP_BAD_ARG not multiple of 8 – *x*, *dx*, or resulting *offadr* must be multiples of 8.

Hardware Modified

None

NOTE *New function added in ITEX revision 2.8.0 for Windows NT and Windows 95. Not applicable to Win16 DOS32x or Windows 3.11.*

Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com

icp_mailbox_read – Read from a PCI Mailbox

Syntax #include <icp.h>

DWORD **icp_mailbox_read**(MODCNF **icpmod*, short *mbox*);

icpmod	Pointer to the IC-PCI module structure.
mbox	Which mail box register to read; ICP_MBOX1_32 – mailbox 1 register ICP_MBOX2_32 – mailbox 2 register ICP_MBOX3_32 – mailbox 3 register ICP_MBOX4_32 – mailbox 4 register

Description

The function **icp_mailbox_read** reads a 32-bit value form the specified mailbox *mbox*. The IC-PCI uses the mail boxes for communication between an IC-PCI application and the PCI display utility program.

This is a low level function. The programmer should have detailed knowledge of how the IC-PCI hardware and software function.

Returned Values

32-bit data in a mail box

Hardware Modified

None

icp_mailbox_write – Write to a PCI Mailbox

Syntax #include <icp.h>

void **icp_mailbox_write**(MODCNF **icpmod*, short *mbox*, DWORD *data*);

icpmod	Pointer to the IC-PCI module structure.
mbox	Which mail box register to read; ICP_MBOX1_32 – mailbox 1 register ICP_MBOX2_32 – mailbox 2 register ICP_MBOX3_32 – mailbox 3 register ICP_MBOX4_32 – mailbox 4 register

Description

The function **icp_mailbox_read** writes a 32-bit value form in the specified mailbox register *mbox*. The IC-PCI uses the mail boxes for communication between an IC-PCI application and the PCI display utility program. This function modifies the IC-PCI module structure.

This is a low level function. The programmer should have detailed knowledge of how the IC-PCI hardware and software function.

Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com

Returned Values
None

Hardware Modified
ICP_MBOX1_32, ICP_MBOX2_32, ICP_MBOX3_32, ICP_MBOX4_32

icp_mem_size – Return the IC-PCI Memory Size

Syntax #include <icp.h>
short **icp_mem_size**(MODCNF **icpmod*);

icpmod Pointer to the IC-PCI module structure.

Description
The function **icp_mem_size** reads the IC-PCI memory size from the module structure. At initialization, this value is read from the IC-PCI configuration registers.

This is a medium level function. The programmer should be familiar with the IC-PCI hardware and software functional design.

Returned Values
The memory size:

ICP_2M – 2MB image memory

ICP_4M – 4MB image memory

Hardware Modified
None

icp_pixsz – Set or Return the Camera Pixel Size

Syntax #include <icp.h>
short **icp_pixsz**(MODCNF **icpmod*, short *pixsz*);

icpmod Pointer to the IC-PCI module structure.

pixsz Size used for operation on pixel data:
ICP_PIX8 – 8–bit pixel size.
ICP_PIX16 – 16–bit pixel size.
ICP_PIX24 – 24-bit pixel size
INQUIRE – Return the current setting.

Description
The function **icp_pixsz** configures the AM input multiplexers, to store 8-bit, 16-bit, or 24-bit image data in frame memory. This function also sets the pixel size used for host access to IC-PCI memory. The pixel size must be set

Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com

appropriately before calling any read/write/clear functions for IC-PCI memory for frames of 16-bit and 24-bit sizes. This function changes the value for pixel size in the module structure.

This is a medium level function. The programmer should be familiar with the IC-PCI hardware and software functional design.

Returned Values

Current setting for pixel size, if *pixsz* = INQUIRE .

ICP_NO_ERROR – Operation completed successfully.

Hardware Modified

ICP_PIXSZ

icp_put_acq_start_addr – Set the Camera Acquire Start Address

Syntax #include <icp.h>

void **icp_put_acq_start_addr**(MODCNF **icpmod*, DWORD *addr*);

icpmod	Pointer to the IC-PCI module structure.
addr	Starting memory address for acquisition.

Description

The function **icp_put_acq_start_addr** programs the starting address used for acquisition from a camera. The memory is addressable on 4K boundaries (4096). The acquisition can begin writing at any one of the 4K boundaries. The maximum address is 4KB less than the total amount of image memory: 2MB or 4MB. This function modifies the IC-PCI module structure.

This is a low level function. The programmer should have detailed knowledge of how the IC-PCI hardware and software function.

Returned Values

None

Hardware Modified

ICP_AQSTART

icp_put_bm_aoix – Set the Bus-Master-Mode AOI Horizontal Size

Syntax #include <icp.h>

void **icp_put_bm_aoix**(MODCNF **icpmod*, WORD *x*, short *pixsz*);

icpmod	Pointer to the IC-PCI module structure.
x	Horizontal size of image, in pixels: 6 to 32768: 24 to 32768 for 8-bit pixels, 12 to 16385 for 16-bit pixels, 6 to 8192 for 24-bit pixels

Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com

pixsz Pixel size:
ICP_PIX8 – 8-bit pixels.
ICP_PIX16 – 16-bit pixels.
ICP_PIX24 – 24-bit pixels.
INQUIRE – Use the current pixel size

Description

The function **icp_put_bm_aoix** programs the horizontal image size for bus master transfers. The number of bytes transferred is calculated from *x* and *pixsz*, and stored. The minimum transfer is 24 bytes. This function modifies the IC-PCI module structure. The actual formula is:

$$(ICP_AOIX + 1) \times 8 = \text{number of bytes}$$

This is a low level function. The programmer should have detailed knowledge of how the IC-PCI hardware and software function.

Returned Values

None

Hardware Modified

PCI_AOIX

icp_put_bm_dst_addr – Set the Bus-Master-Mode Destination Address

Syntax

#include <icp.h>
void **icp_put_bm_dst_addr**(MODCNF **icpmod*, DWORD *addr*);

icpmod Pointer to the IC-PCI module structure.
addr PCI destination for a bus master transfer.

Description

The function **icp_put_bm_dst_addr** programs the destination address used by the IC-PCI for a bus master transfer (write) to another memory resource, usually host memory or VGA display memory. The address cannot be a user-allocated buffer. A DMA request function must be called to get an address to use as an argument for this function.

This is a low level function. The programmer should have detailed knowledge of how the IC-PCI hardware and software function.

Returned Values

None

Hardware Modified

ICP_BMDST_32

Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com

icp_put_bm_firstf_start – Set the First Field Starting Address of Bus-Master Transfer

Syntax #include <icp.h>

void **icp_put_bm_firstf_start**(MODCNF *_icpmod_, DWORD _addr_);

icpmod	Pointer to the IC-PCI module structure.
addr	Starting address in IC-PCI memory of the first field

Description

The function **icp_put_bm_firstf_start** programs the starting address in IC-PCI memory of the first field of an image to be transferred. The application must provide the starting address of the first and second fields of data for a bus master transfer of an interlaced image. This function modifies the IC-PCI module structure.

This is a low level function. The programmer should have detailed knowledge of how the IC-PCI hardware and software function.

Returned Values

None

Hardware Modified

ICP_BMSFL, ICP_BMSFH

icp_put_bm_secondf_start – Set the Second Field Starting Address of Bus-Master Transfer

Syntax #include <icp.h>

void **icp_put_bm_secondf_start**(MODCNF *_icpmod_, DWORD _addr_);

icpmod	Pointer to the IC-PCI module structure.
addr	Starting address in IC-PCI memory of the first field

Description

The function **icp_put_bm_secondf_start** programs the starting address in IC-PCI memory of the second field of an image to be transferred. The application must provide the starting address of the first and second fields of data for a bus master transfer of an interlaced image frame. This function modifies the IC-PCI module structure.

This is a low level function. The programmer should have detailed knowledge of how the IC-PCI hardware and software function.

Returned Values

None

Hardware Modified

ICP_BMSSL, ICP_BMSSH

Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com

icp_read_area – Read an Area in IC-PCI Memory to a Buffer

Syntax #include <icp.h>

short **icp_read_area**(MODCNF **icpmod*, short *frame*, short *x*, short *y*, short *dx*, short *dy*, DWORD **darray*);

icpmod	Pointer to the IC-PCI module structure.
frame	The frame to write to: a frame ID returned by icp_create_frame, or defined by the configuration file.
x	Horizontal coordinate of top-left corner of rectangle.
y	Vertical coordinate of top-left corner of rectangle.
dx	Width of rectangle, in pixels.
dy	Height of rectangle, in pixels.
darray	Pointer to array to read data into.

Description

The function **icp_read_area** reads an area of size dx by *dy* into an array *darray* from the selected *frame*, starting at location *x*,*y*. This function reads horizontal lines (BYTE, WORD, or DWORD depending on the offset, boundary, and pixel size). This function attempts to perform a bus master transfer. If bus master mode is not enabled, the function calls icp_set_lineio, and performs host reads to IC-PCI memory.

darray should be a buffer *dx* times *dy* for a monochrome 8–bit image, and *dx* times *dy* times 4 bytes for a 24-bit color image. The IC-PCI stores and reads 24-bit color images as 32-bits.

The size of the area is clipped to fit the size of the image frame. If *x*+*dx* is larger than the width of the frame, *dx* is limited to the frame width minus *x*. If *y*+*dy* is larger than the frame height, *dy* is limited to frame height minus *y*.

This is a high level function. The programmer does not need in-depth knowledge of how the IC-PCI hardware and software function.

NOTE *For the read area to use bus master mode, the following conditions must be met:*

If the pixel depth for *frame* is ICP_PIX8; x, y, dx, and the frame's X dimension must be multiples of 8. If the pixel depth for *frame* is ICP_PIX16; x, y, dx, and the frame's X dimension must be multiples of 4. If the pixel depth for *frame* is ICP_PIX24; x, y, dx, and the frame's X dimension must be multiples of 2. Additionally bus master mode must be enabled, refer to icp_bm_mode.

Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com

Returned Values

ICP_NO_ERROR – Operation completed successfully.

ICP_BAD_ARG invalid frame – *frame* not defined, pointer returns NULL.

ICP_BAD_ARG invalid x and/or y – either x > frame width, or, y > frame height, or both.

ICP_BAD_ARG Arg not multiple of 8 – *x*, *dx*, or resulting *offadr* must be multiples of 8.

ICP_BAD_ARG destination address not DWORD aligned.

ITX_TIMEOUT BMSEMA bit stuck – not able to obtain bus master semaphore.

Hardware Modified

None

icp_read_dest_pitch_area – Read an Area Adding Destination Pitch

Syntax #include <icp.h>

short **icp_read_dest_pitch_area**(MODCNF **icpmod*, short *frame*, short *x*, short *y*, short *dx*, short *dy*, short *destlinebytes*, DWORD *physdest*, WORD *disp_depth*);

icpmod	Pointer to the IC-PCI module structure.
frame	The frame to from: a frame ID returned by icp_create_frame, or defined by the configuration file.
x	Horizontal starting location of the area read in *frame*.
y	Vertical starting location of the area read in *frame*.
dx	Width of area read in *frame*; number of pixels.
dy	Height of area read in *frame*; number of lines.
destlinebytes	Number of bytes in the destination line.
physdest	Physical address to transfer to.
disp_depth	Pixel size of of the destination; number of bits.

Description

The function **icp_read_dest_pitch_area** transfers an area in an IC-PCI *frame* to a memory at physical address *physdest*. The destination is a rectangular memory with a line length (or "pitch") of value *destlinebytes* and a depth of *disp_depth*. The area in IC-PCI *frame* is defined by *x*,*y* and *dx*,*dy*.

Returned Values

ICP_BAD_ARG invalid frame – *frame* is not defined, pointer returned NULL.

ICP_BAD_ARG invalid x and/or y – either x > frame width, or, y > frame height, or both.

ICP_BAD_ARG not multiple of 8 – *x*, *dx*, or resulting *offadr* must be multiples of 8.

Hardware Modified

None

NOTE *New function added in ITEX revision 2.8.0 for Windows NT and Windows 95. Not applicable to Win16 DOS32x or Windows 3.11.*

Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com

icp_read_harea – Read Large Area in IC-PCI Memory to an Array (Win16 3.x only)

Syntax #include <icp.h>

short **icp_read_harea**(MODCNF **icpmod*, short *frame*, short *x*, short *y*, short *dx*, short *dy*, BYTE-HUGE **darray*);

icpmod	Pointer to the IC-PCI module structure.
frame	The frame to write to: a frame ID returned by icp_create_frame, or defined by the configuration file.
x	Horizontal coordinate of top-left corner of rectangle.
y	Vertical coordinate of top-left corner of rectangle.
dx	Width of rectangle.
dy	Height of rectangle.
darray	Pointer to the GMEM_FIXED memory area to read data into, can be larger than 64K. GlobalAlloc and GlobalLock are intended to be used to allocate target memory *darray.*

Description

The function **icp_read_harea** is a function available under Windows only, that reads an area of size *dx* by *dy* into *darray* from the selected *frame*, starting at location *x,y*. This function reads horizontal lines (BYTE, WORD, or DWORD depending on the offset, boundary, and pixel size). This function attempts to perform a bus master transfer. If bus master mode is not enabled, the function calls icp_set_lineio, and performs host reads to IC-PCI memory. icp_read_harea handles reading areas greater than 64KB.

The size of the area is clipped to fit the size of the image frame. If *x+dx* is larger than the width of the frame, *dx* is limited to the frame width minus *x*. If *y+dy* is larger than the frame height, *dy* is limited to frame height minus *y*.

darray should be a GMEM_FIXED buffer *dx* times *dy* for a monochrome 8–bit image, and *dx* times *dy* times 4 bytes for a 24-bit color image. The IC-PCI stores and reads 24-bit color images as 32-bits.

This is a high level function. The programmer does not need in-depth knowledge of how the IC-PCI hardware and software function.

NOTE *For the read area to use bus master mode the following conditions must be met:*

If the pixel depth for *frame* is ICP_PIX8; *x, y, dx,* and the frame's X dimension must be multiples of 8. If the pixel depth for *frame* is ICP_PIX16; *x, y, dx,* and the frame's X dimension must be multiples of 4. If the pixel depth for *frame* is ICP_PIX24; *x, y, dx,* and the frame's X dimension must be multiples of 2. Also bus master mode must be enabled (refer to icp_bm_mode).

Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com

Returned Values

ICP_NO_ERROR – Operation completed successfully.

ICP_BAD_ARG invalid frame – *frame* not defined, pointer returns NULL.

ICP_BAD_ARG invalid x and/or y – x > frame width, y > frame height, or both.

ICP_BAD_ARG Arg not multiple of 8 – *x*, *dx*, or resulting *offadr* must be multiples of 8.

ICP_BAD_ARG destination address not DWORD aligned.

ITX_TIMEOUT BMSEMA bit stuck – not able to obtain bus master semaphore.

Hardware Modified

None

icp_read_plane_area – Read a Color Plane from a Planar Frame

Syntax #include <icp.h>

short **icp_read_plane_area**(MODCNF **icpmod*, short *frame*, short *plane*, short *x*, short *y*, short *dx*, short *dy*, DWORD **darray*);

icpmod	Pointer to the IC-PCI module structure.
frame	Pointer to a frame created by icp_create_frame; must be of depth ICP_PIX24 or ICP_PIX32 and of color ICP_RBG_PLANAR.
plane	color plane to read: ICP_RED – red plane from RGB ICP_GREEN – green plane from RGB ICP_BLUE – blue plane from RGB
x	Horizontal coordinate of top-left corner of rectangle.
y	Vertical coordinate of top-left corner of rectangle.
dx	Width of rectangle, in pixels.
dy	Height of rectangle, in pixels.
darray	Pointer to array to read data into.

NOTE *New function added in ITEX revision 2.8.0 for Windows NT and Windows 95. Not applicable to Win16 DOS32x or Windows 3.11.*

Description

The function **icp_read_plane_area** reads data from a single color plane that was captured by the AM-STD-RGB in Dynamic MUX mode. *frame* must be *color* = ICP_RGB_PLANAR.

x, y, dx, and the frame's X dimension must be multiples of 8. Additionally bus master mode must be enabled; refer to icp_bm_mode.

This function reads an area of size dx by *dy* into an array *darray* from the selected *frame*, starting at location *x*,*y*. This function reads horizontal lines (BYTE, WORD, or DWORD depending on the offset, boundary, and pixel

Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com

size). This function attempts to perform a bus master transfer. If bus master mode is not enabled, the function calls icp_set_lineio, and performs host reads to the IC-PCI memory.

darray should be a buffer *dx* times *dy* for a planar 8–bit image.

The size of the area is clipped to fit the size of the image frame. If *x+dx* is larger than the width of the frame, *dx* is limited to the frame width minus *x*. If *y+dy* is larger than the frame height, *dy* is limited to frame height minus *y*.

This is a high level function. The programmer does not need in-depth knowledge of how the IC-PCI hardware and software function.

NOTE *For the read area to use bus master mode, the following conditions must be met:*

Returned Values

ICP_NO_ERROR – Operation completed successfully.

ICP_BAD_ARG invalid frame – *frame* not defined, pointer returns NULL.

ICP_BAD_ARG invalid x and/or y – either x > frame width, or, y > frame height, or both.

ICP_BAD_ARG – *frame* is not a planar frame.

ICP_BAD_ARG – Planar read only allowed when bus mastering is enabled.

Hardware Modified

None

icp_read_roi_area – Read Area from ROI to an Array

Syntax

#include <icp.h>

short **icp_read_roi_area**(MODCNF **icpmod*, ROIID *roi*, short *x*, short *y*, short *dx*, short *dy*, DWORD **darray*);

icpmod	Pointer to the IC-PCI module structure.
roi	ROI ID returned by icp_create_roi.
x	Horizontal coordinate of top-left corner of rectangle.
y	Vertical coordinate of top-left corner of rectangle.
dx	Width of rectangle.
dy	Height of rectangle.
darray	Pointer to the array to read data into.

Description

The function **icp_read_roi_area** reads an area inside a defined ROI of size *dx* by *dy* into an array *darray* starting at location *x,y*. This function calls icp_set_lineio, and reads horizontal lines (BYTE, WORD, or DWORD depending on the offset, boundary, and pixel size). The depth, color, and interlace format is taken from the parent frame.

This is a high level function. The programmer does not need in-depth knowledge of how the IC-PCI hardware and software function.

Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com

The size of the area is clipped to fit the size of the image frame. If *x+dx* is larger than the width of the ROI, *dx* is limited to the ROI width minus *x*. If *y+dy* is larger than the ROI height, *dy* is limited to the ROI height minus *y*.

darray should be a GMEM_FIXED buffer *dx* times *dy* for a monochrome 8–bit image, and *dx* times *dy* times 4 bytes for a 24-bit color image. The IC-PCI stores and reads 24-bit color images as 32-bits.

NOTE *For bus master mode to be used, the following conditions must be met:*

If the pixel depth is ICP_PIX8; *x*, *y*, and *dx* must be multiples of 8. If the pixel depth is ICP_PIX16; *x*, *y*, and *dx* must be multiples of 4. If the pixel depth is ICP_PIX24; *x*, *y*, and *dx* must be multiples of 2.

Returned Values

ICP_NO_ERROR – Operation completed successfully.
ICP_BAD_ARG invalid roi – *roi* not defined, pointer returns NULL.
ICP_BAD_ARG invalid x and/or y – x > ROI width, or, y > ROI height, or both.

Hardware Modified

None

icp_read_roi_harea – Read Large Area from ROI to an Array (Win16 3.x only)

Syntax

#include <icp.h>

short **icp_read_roi_harea**(MODCNF **icpmod*, ROIID *roi*, short *x*, short *y*, short *dx*, short *dy*, BYTE-HUGE **darray*);

icpmod	Pointer to the IC-PCI module structure.
roi	ROI ID returned by icp_create_roi.
x	Horizontal coordinate of top-left corner of rectangle.
y	Vertical coordinate of top-left corner of rectangle.
dx	Width of rectangle.
dy	Height of rectangle.
darray	Pointer to the array to read data into.

Description

The function **icp_read_roi_harea** is a function available *under Windows only*, that reads an area of size *dx* by *dy* into an array *darray* from the selected *frame*, starting at location *x,y*. This function calls icp_set_lineio, and reads horizontal lines (BYTE, WORD, or DWORD depending on the offset, boundary, and pixel size). icp_read_roi_harea handles crossing 64K boundaries.

This is a high level function. The programmer does not need in-depth knowledge of how the IC-PCI hardware and software function.

The size of the area is clipped to fit the size of the image frame. If *x+dx* is larger than the width of the frame, *dx* is limited to the frame width minus *x*. If *y+dy* is larger than the frame height, *dy* is limited to frame height minus *y*.

Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com

darray should be a GMEM_FIXED buffer *dx* times *dy* for a monochrome 8–bit image, and *dx* times *dy* times 4 bytes for a 24-bit color image. The IC-PCI stores and reads 24-bit color images as 32-bits.

NOTE *Bus master mode requires the following condition:*

If the pixel depth for *frame* is ICP_PIX8; *x, y, dx,* and the frame's X dimension must be multiples of 8. If the pixel depth for *frame* is ICP_PIX16; *x, y, dx,* and the frame's X dimension must be multiples of 4. If the pixel depth for *frame* is ICP_PIX24; *x, y, dx,* and the frame's X dimension must be multiples of 2.

Returned Values

ICP_NO_ERROR – Operation completed successfully.

ICP_BAD_ARG invalid roi – *roi* not defined, pointer returns NULL.

ICP_BAD_ARG invalid x and/or y – x > ROI width, or, y > ROI height.

Hardware Modified

None

icp_rhbline – Read a Horizontal Line of BYTE Values

Syntax

#include <icp.h>

void **icp_rhbline**(DWORD *offadr*, short *nreads*, short *inc*, BYTE **b_ptr*);

offadr The memory address returned from icp_set_lineio or icp_roi_set_lineio.

nreads Number of locations to read.

inc Number of memory elements (BYTEs) to increment between writes, from 0 to 32767. If reading from an ROI, this should be the number of bytes in the parent frame pixel depth.

- 0 Read the same location.
- 1 Increment one byte location between reads
- 2 Increment 2 bytes between reads.
- 3 Increment by 3 bytes between reads.

. . .

32,767 Read every 32,767th BYTE location.

b_ptr Pointer to a buffer, to store the BYTE data read.

Description

The function **icp_rhbline** reads a horizontal line of BYTE values from image frame memory. You must select values for *nreads* and *inc* that do not exceed the size of memory. This function returns immediately (with no error) if *nreads* is less than or equal to zero.

This is a medium level function. The programmer should be familiar with the IC-PCI hardware and software functional design.

Returned Values

None

Hardware Modified

None

Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com

icp_rhdwline – Read a Horizontal Line of DWORD Values

Syntax #include <icp.h>

void **icp_rhdwline**(DWORD *offadr*, short *nreads*, short *inc*, DWORD **d_ptr*);

offadr The memory address returned from icp_set_lineio or icp_roi_set_lineio.

nreads Number of locations to read.

inc Number of memory elements (DWORDs) to increment between writes, from 0 to 32767.

- 0 Read the same location.
- 1 Increment one DWORD between reads.
- 2 Increment two DWORDs between reads.
- 3 Increment three DWORDs between reads.

. . .

32767 Read every 32,767th DWORD location.

d_ptr Pointer to a buffer, to store the DWORD data read.

Description

The function **icp_rhdwline** reads a horizontal line of DWORD values in image memory. You must select values for *nreads* and *inc* that do not exceed the size of memory. Each DWORD read by icp_rhdwline contains four (horizontally adjacent) 8-bit pixels. This function returns immediately (with no error) if *nreads* is less than or equal to zero.

This is a medium level function. The programmer should be familiar with the IC-PCI hardware and software functional design.

Returned Values

None

Hardware Modified

None

icp_rhwline – Read a Horizontal Line of WORD Values

Syntax #include <icp.h>

void **icp_rhwline**(DWORD *offadr*, short *nreads*, short *inc*, WORD **w_ptr*);

offadr The memory address returned from icp_set_lineio.

nreads Number of locations to read.

Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com

inc Number of memory elements (WORDs) to increment between writes, from 0 to 32767.

0 Read the same location.

1 Increment one WORD between reads.

2 Increment two WORDs between reads.

3 Increment three WORDs between reads.

. . .

32767 Read every 32,767th WORD location.

w_ptr Pointer to a buffer, to store the WORD data read.

Description

The function **icp_rhwline** reads a horizontal line of WORD values in image frame memory. You must select values for *nreads* and *inc* that do not exceed the size of memory. Each WORD read by icp_rhwline contains two (horizontally adjacent) 8-bit pixels. This function returns immediately (with no error) if *nreads* is less than or equal to zero.

This is a medium level function. The programmer should be familiar with the IC-PCI hardware and software functional design.

Returned Values

None

Hardware Modified

None

icp_roi_attrib – Read Attributes of a ROI

Syntax

#include <icp.h>

WORD **icp_roi_attrib**(MODCNF **icpmod*, ROIID *roi*, WORD **maxdx*, WORD **maxdy*, WORD **curdx*, WORD **curdy*, WORD **pxdepth*, WORD **color*);

icpmod Pointer to the IC-PCI module structure.

roi Pointer to the ROI define structure

maxdx Maximum dx of ROI, a returned value

maxdy Maximum dy of ROI, a returned value

curdx Current dx of ROI after any clipping, a returned value

curdy Current dy of ROI after any clipping, a returned value

pxdepth Pixel depth of the ROI, a returned value

color Color mode of the ROI, a returned value

Description

The function **icp_roi_attrib** reads the ROI module structure for *roi* and returns the values. *maxdx*, *maxdy* *pxdepth*, and *color* are programmed by icp_create_roi. *curdx* and *curdy* are the results after any clipping to fit the size of the acquisition in the parent frame.

Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com

This is a high level function. The programmer does not need in-depth knowledge of how the IC-PCI hardware and software function.

Returned Values

maxdx, maxdy, curdx, curdy, pxdepth, color.

ICP_NO_ERROR – Operation completed successfully.

ICP_BAD_ARGUMENT invalid ROI – ROIID pointer returns NULL; ROI not defined

Hardware Modified

None

icp_roi_rpix – Read a Pixel from a ROI

Syntax

#include <icp.h>

DWORD **icp_roi_rpix**(MODCNF **icpmod*, ROIID *roi*, short *x*, short *y*);

icpmod	Pointer to the IC-PCI module structure.
roi	ROI ID returned by icp_create_roi
x	Horizontal coordinate relative to ROI origin
y	Vertical coordinate relative to ROI origin

Description

The function **icp_roi_rpix** reads a single pixel value from location x,y in an ROI. The function returns a BYTE, WORD, or DWORD based on the current pixel size set by icp_pixsz. The pixel size should agree with the depth of the parent frame. This function returns a NULL if the *roi* is not defined.

This is a high level function. The programmer does not need in-depth knowledge of how the IC-PCI hardware and software function.

Returned Values

A pixel data value (byte, word, or dword)

NULL if *roi* is not defined (if pointer returns NULL)

Hardware Modified

None

Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com

icp_roi_set_lineio – Set Up the IC-PCI for Access into ROIs

Syntax #include <icp.h>

short **icp_roi_set_lineio**(MODCNF **icpmod*, ROIID *roi*, short *x*, short *y*, short **xsz*, DWORD **offadr*, DWORD **ilacedata*);

icpmod	Pointer to the IC-PCI module structure.
x	Horizontal location to start
y	Vertical location to start
xsz	Pointer to the horizontal size (in bytes) of parent frame, a returned value.
offadr	Pointer to the memory address, a returned value.
ilacedata	Interlace status of the parent frame, a returned value.

Description

The function **icp_roi_set_lineio** computes *offadr* and *ilacedata*, which can be used to access a pixel or pixels starting at location *x,y* in a ROI. This function must be called before any line read, write, or clear functions (refer to Table 1-6, page 1-10).

This is a medium level function. The programmer should be familiar with the IC-PCI hardware and software functional design.

Subsequent line accesses should recall icp_roi_set_lineio. If the ROI parent frame is interlaced you can not add xsz to *offadr* to access the next line.

NOTE *The appropriate pixel size must be set using* icp_pixsz

Returned Values

xsz, offadr, ilacedata

NULL if frame or ROI not defined.

ICP_NO_ERROR – Operation successfully completed.

ICP_BAD_ARG invalid roi – *roi* not defined, pointer returns NULL.

ICP_BAD_ARG invalid x and/or y – either
x > ROI width, or, y > ROI height, or both.

Hardware Modified

None

Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com

icp_roi_wpix – Write a Pixel to a ROI

Syntax #include <icp.h>

void **icp_roi_wpix**(MODCNF **icpmod*, ROIID *roi*, short *x*, short *y*, DWORD *pv*);

icpmod	Pointer to the IC-PCI module structure.
roi	ROI ID returned by icp_create_roi
x	Horizontal coordinate relative to ROI origin
y	Vertical coordinate relative to ROI origin
pv	Pixel value to write.

Description

The function **icp_roi_wpix** writes a single pixel value at location *x*,*y* in a ROI. The function writes a BYTE, WORD, or DWORD according to the pixel size of the ROI.

This is a high level function. The programmer does not need in-depth knowledge of how the IC-PCI hardware and software function.

Returned Values

None

Hardware Modified

None

icp_rpix – Read a Pixel Value in IC-PCI Memory

Syntax #include <icp.h>

DWORD **icp_rpix**(MODCNF **icpmod*, short *frame*, short *x*, short *y*);

icpmod	Pointer to the IC-PCI module structure.
frame	The frame to write to: a frame ID returned by icp_create_frame, or defined by the configuration file.
x	Horizontal (X) coordinate: 0 to 4096, *x* should be within *frame*.
y	Vertical (Y) coordinate: 0 to 4096, *y* should be within *frame*.

Description

The function **icp_rpix** reads a single pixel value from *frame* at location *x*,*y*. The function reads a BYTE, WORD, or DWORD according to the current pixel size set by icp_pixsz. The pixel size should agree with the depth of the frame.

This is a high level function. The programmer does not need in-depth knowledge of how the IC-PCI hardware and software function.

Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com

Returned Values

A pixel value (byte, word, or dword)

NULL if *frame* not defined (if pointer returns NULL)

Hardware Modified

None

icp_rvbline – Read a Vertical Line of BYTE Values

Syntax #include <icp.h>

void **icp_rvbline**(DWORD *offadr*, short *nreads*, short *linesz*, DWORD *ilacedata*, BYTE **b_ptr*);

offadr	The memory address returned by icp_set_lineio or icp_roi_set_lineio.
nreads	Number of locations to read.
linesz	Horizontal size (in bytes) of the parent frame, returned as *xsz* by icp_set_lineio or icp_roi_set_lineio.
ilacedata	Interlace status of the parent frame, returned by icp_set_lineio or icp_roi_set_lineio.
b_ptr	Pointer to a buffer, to store the BYTE data read.

Description

The function **icp_rvbline** reads a vertical line of BYTE values in a frame. You must select a value for *nreads* that does not exceed the size of the frame (or image). This function returns immediately if *nreads* is less than or equal to zero.

This is a medium level function. The programmer should be familiar with the IC-PCI hardware and software functional design.

Returned Values

None

Hardware Modified

None

icp_rvdwline – Read a Vertical Line of DWORD Values

Syntax #include <icp.h>

void **icp_rvdwline**(DWORD *offadr*, short *nreads*, short *linesz*, DWORD *ilacedata*, DWORD **d_ptr*);

offadr	The memory address returned by icp_set_lineio or icp_roi_set_lineio.
nreads	Number of locations to read.
linesz	Horizontal size (in bytes) of the parent frame, returned as *xsz by* icp_set_lineio or icp_roi_set_lineio.

Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com

ilacedata Interlace status of the parent frame, returned by icp_set_lineio or icp_roi_set_lineio.

d_ptr Pointer to a buffer, to store the DWORD data read.

Description

The function **icp_rvdwline** reads a vertical line of DWORD values in a frame. You must select a value for *nreads* that does not exceed the size of the frame (or image). Each DWORD read by icp_rvdwline contains one 24-bit pixel, or two (horizontally adjacent) 16-bit pixels, or four (horizontally adjacent) 8-bit pixels. This function returns immediately if *nreads* is less than or equal to zero.

This is a medium level function. The programmer should be familiar with the IC-PCI hardware and software functional design.

Returned Values

None

Hardware Modified

None

icp_rvwline – Read a Vertical Line of WORD Values

Syntax

#include <icp.h>

void **icp_rvwline**(DWORD *offadr*, short *nreads*, short *linesz*, DWORD *ilacedata*, WORD **w_ptr*);

offadr The memory address returned by icp_set_lineio or icp_roi_set_lineio.

nreads Number of locations to read.

linesz Horizontal size (in bytes) of the parent frame, returned by icp_set_lineio or icp_roi_set_lineio.

ilacedata Interlace status of the parent frame, returned as *xsz* by icp_set_lineio or icp_roi_set_lineio.

w_ptr Pointer to a buffer, to store the WORD data read.

Description

The function **icp_rvwline** reads a vertical line of WORD values in a frame. You must select a value for *nreads* that does not exceed the size of the frame (or image). Each WORD read by icp_rvwline contains one 16-bit pixel, or two (horizontally adjacent) 8-bit pixels. This function returns immediately if *nreads* is less than or equal to zero.

This is a medium level function. The programmer should be familiar with the IC-PCI hardware and software functional design.

Returned Values

None

Hardware Modified

None

Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com

icp_scanmd_status – Return the Camera Scan Mode Status

Syntax #include <icp.h>
short **icp_scanmd_status**(MODCNF **icpmod*);

icpmod Pointer to the IC-PCI module structure.

Description

The function **icp_scanmd_status** reads the scan mode status bit ICP_SMSTAT, showing the scan mode of the camera interface.

This is a low level function. The programmer should have detailed knowledge of how the IC-PCI hardware and software function.

Returned Values

ICP_ILACED – interlaced camera.

ICP_NILACED – non-interlaced camera.

Hardware Modified

None

icp_seq_snap – Acquire Sequential Images

Syntax #include <icp.h>
short **icp_seq_snap**(MODCNF **icpmod*, WORD *framecnt*, WORD **frames*);

icpmod Pointer to the IC-PCI module structure.

framecnt Number of sequential frames to acquire: 1 to 8

frames Pointer to IDs for sequential frames, created by icp_create_seq_frames.

Description

The function **icp_seq_snap** acquires *framecnt* frames. The sequential *frames* ID must be already defined by icp_create_seq_frames.

This function waits for the acquisition to complete before returning.

The maximum number of frames is 8. The function icp_seq_snap can acquire up to 8 frames, and *framecnt* can be less than the maximum number defined by icp_create_seq_frames; however, frames must always point to the first frame in a sequential frame buffer.

This is a high level function. The programmer does not need in-depth knowledge of how the IC-PCI hardware and software function.

Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com

Returned Values

ICP_NO_ERROR – operation completed successfully; or *framecnt* less than 1.

ICP_BAD_ARG invalid frame – *frames* not defined, pointer returns NULL.

ICP_BAD_ARG invalid mod pointer – *icpmod* returns NULL.

ICP_BAD_ARG invalid AM mod pointer – pointer to AM returns NULL.

ICP_OUT_FB_MEM – memory full.

ITX_TIMEOUT – ACQMD bits did not return to freeze at end of acquisition.

Hardware Modified

ICP_ACQMD, ICP_FLDSEL

icp_seq_snap_async – Acquire Sequential Images Asynchronously

Syntax

#include <icp.h>

short **icp_snap_async**(MODCNF **icpmod*, WORD *framecnt*, WORD **frames*);

icpmod	Pointer to the IC-PCI module structure.
framecnt	Number of sequential frames to acquire: 1 to 8
frames	Pointer to IDs for sequential frames, created by icp_create_seq_frames.

Description

The function **icp_seq_snap_async** acquires *framecnt* frames. This function returns immediately, without waiting for the acquisition to complete. The sequential *frames* ID must be already defined by icp_create_seq_frames.

The maximum number of frames is 8. The function icp_seq_snap_async can acquire up to 8 frames, and *framecnt* can be less than the maximum number defined by icp_create_seq_frames; however, frames must always point to the first frame in a sequential frame buffer.

NOTE *icp_seq_snap waits for the acquisition to complete before returning. icp_seq_snap_async returns immediately, without waiting for the acquisition to complete.*

This is a high level function. The programmer does not need in-depth knowledge of how the IC-PCI hardware and software function.

Returned Values

ICP_NO_ERROR – operation completed successfully; or *framecnt* less than 1.

ICP_BAD_ARG invalid frame – *frames* not defined, pointer returns NULL.

ICP_BAD_ARG invalid mod pointer – *icpmod* returns NULL.

ICP_BAD_ARG invalid AM mod pointer – pointer to AM returns NULL.

ICP_OUT_FB_MEM – memory full.

ITX_TIMEOUT – ACQMD bits did not return to freeze at end of acquisition.

Hardware Modified

ICP_ACQMD, ICP_FLDSEL

Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com

icp_set_cur_dim – Set Current Dimensions for an IC-PCI Buffer Frame

Syntax #include <icp.h>

short **icp_set_cur_dim**(MODCNF *_icpmod_, short _frame_, WORD _curdx_, WORD _curdy_, short _reformat_);

icpmod	Pointer to the IC-PCI module structure.
frame	ID for the IC–PCI frame to modify, a pointer returned by icp_create_frame.
curdx	New value for frame width (dx).
curdy	New value for frame height (dy).
reformat	Reserved for possible future use. Program to zero.

Description

The function **icp_set_cur_dim** modifies the attributes for the selected *frame*, with the new dimensions *curdx* and *curdy*. Frames are defined by icp_create_frame, but the acquired image can be smaller than the defined frame.

Returned Values

ICP_NO_ERROR – Operation successfully completed.

ICP_BAD_ARG invalid frame – *frame* not defined

ICP_BAD_ARG ACQ dimension exceeds frame – *curdx* greater than maximum dx or *curdy* greater than maximum dy in *frame*.

Hardware Modified

None

NOTE *Function added in version 2.6.2 for DOS32x and Win16, and version 2.6.1 for Win32.*

icp_set_frame_ilace – Set the Interlace Mode for a Frame

Syntax #include <icp.h>

short **icp_set_frame_ilace**(MODCNF *_icpmod_, short _frame_, ICP_ILACE _interlaced_, ICP_FIELD_STATUS _fld_status_);

icpmod	Pointer to the IC-PCI module structure.
frame	The frame to write to: a frame ID returned by icp_create_frame, or defined by the configuration file.
interlaced	Interlace mode: ICP_ENABLE – interlacing enabled. ICP_DISABLE – interlacing disabled. INQUIRE – return current mode.
fld_status	The starting field: ICP_ODD – odd field first ICP_EVEN – even field first

Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com

Description

The function **icp_set_frame_ilace** sets the current interlace attributes for an IC-PCI frame. This function changes the IC-PCI module structure. This funciton can be used to read an interlaced image in IC-PCI memory as a non-interlaced image; for example, to read the fields individually.

This function should be called after the acquisition, and before the read: icp_grab . . . icp_set_frame_ilace . . . icp_read_area . . . .

Returned Values

ICP_NO_ERROR – Operation completed successfully.

ICP_BAD_ARG invalid frame – *frame* not defined, pointer returned NULL.

Hardware Modified

None

icp_set_lineio – Set Up the IC-PCI for Host Access to Frames

Syntax

#include <icp.h>

short **icp_set_lineio**(MODCNF **icpmod*, short *frame*, short *x*, short *y*, short **xsz*, DWORD **offadr*, DWORD **ilacedata*);

icpmod	Pointer to the IC-PCI module structure.
frame	The frame to write to: a frame ID returned by icp_create_frame, or defined by the configuration file.
x	Horizontal location relative to *frame* origin.
y	Vertical location relative to *frame* origin.
xsz	Pointer to the horizontal size of *frame*, a returned value.
offadr	Pointer to the memory address, a returned value.
ilacedata	Interlace status of *frame*, a returned value.

Description

The function **icp_set_lineio** calculates the *offadr* and *ilacedata* which can be used to access a pixel or pixels starting at location *x*,*y* within frame This function must be called before any line read, write, or clear functions (refer to Table 1-6, page 1-10). This function must also be called when you need to change the location you are accessing.

Subsequent line accesses use *offadr* to specify where data is accessed. Access to pixel at location (*x*,*y*+1) can be achieved by adding *xsz* to *offadr* only if *frame* is not interlaced.

This is a medium level function. The programmer should be familiar with the IC-PCI hardware and software functional design.

Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com

Returned Values

xsz, offadr, ilacedata, Returns NULL on failure.

ICP_NO_ERROR – Operation successfully completed.

ICP_BAD_ARG invalid frame – *frame* not defined, pointer returns NULL.

ICP_BAD_ARG invalid x and/or y – either
x > frame width, or, y > frame height, or both.

Hardware Modified

None

icp_set_xform – Set Up for Bus Master Transform

Syntax #include <icp.h>

void **icp_set_xform**(MODCNF **icpmod*, ICP_XFORM_TYPE *xform*);

icpmod	Pointer to the IC-PCI module structure.
xform	Transform performed during bus master transfer to host: NO_XFORM – transfer image without transform. FLIP_XFORM – flip transform, flip vertical. MIRROR_XFORM – mirror transform, flip horizontal. ROT180_XFORM – rotate image 180 detrees.

Description

The function **icp_set_xform** configures the bus master interface to perform an image transform during the bus master transfer to the host memory.

NOTE *New function added in ITEX revision 4.0.0 for Windows NT and WIndows 95. Not applicable to Win16 DOS32x or Windows 3.11.*

Returned Values

None

Hardware Modified

None

icp_snap – Acquire a Single Image

Syntax #include <icp.h>

short **icp_snap**(MODCNF **icpmod*, short *frame*);

icpmod	Pointer to the IC-PCI module structure.
frame	The frame ID to acquire into: a frame ID returned by icp_create_frame, or defined by the configuration file.

Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com

Description

The function **icp_snap** acquires a single image into *frame*. This function waits for acquisition to complete before returning.

This is a high level function. The programmer does not need in-depth knowledge of how the IC-PCI hardware and software function.

Returned Values

ICP_BAD_ARG invalid frame – *frame* not defined, pointer returns NULL.

ICP_BAD_ARG invalid mod pointer – *icpmod* returns NULL.

ICP_BAD_ARG can't acquire seq frame by itself – *frame* points to sequential frame ID, use icp_seq_snap, or change *frame* argument.

ICP_BAD_ARG invalid AM mod ptr – pointer to AM returns NULL.

ICP_OUT_FB_MEM – x or y not inside frame.

ITX_TIMEOUT – ICP_ACQMD bits did not return to freeze at end of snap.

Hardware Modified

ICP_FLDSEL, ICP_ACQMD

icp_snap_async – Acquire a Single Image Asynchronously

Syntax #include <icp.h>

short **icp_snap_async**(MODCNF **icpmod*, short *frame*);

icpmod	Pointer to the IC-PCI module structure.
frame	The frame ID to acquire into: a frame ID returned by icp_create_frame, or defined by the configuration file.

Description

The function **icp_snap_async** acquires a single image into *frame*. This function returns immediately, without waiting for the acquisition to complete.

NOTE icp_snap *waits for the acquisition to complete.* icp_snap_async *returns immediately without waiting for the acquisition to complete.*

This is a high level function. The programmer does not need in-depth knowledge of how the IC-PCI hardware and software function.

Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com

Returned Values

ICP_BAD_ARG invalid frame – *frame* not defined, pointer returns NULL.

ICP_BAD_ARG invalid mod pointer – *icpmod* returns NULL.

ICP_BAD_ARG can't acquire seq frame by itself – *frame* points to sequential frame ID, use icp_seq_snap, or change *frame* argument.

ICP_BAD_ARG invalid AM mod ptr – pointer to AM returns NULL.

ICP_OUT_FB_MEM – x or y not inside frame.

ITX_TIMEOUT – ICP_ACQMD bits did not return to freeze at end of snap.

Hardware Modified

ICP_FLDSEL, ICP_ACQMD

icp_soft_reset – Set or Return the Software Reset

Syntax #include <icp.h>

short **icp_soft_reset**(MODCNF **icpmod*, short *mode*);

icpmod	Pointer to the IC-PCI module structure.
mode	Reset status to write: ICP_RESET – Perform a software reset ICP_NO_RESET – Do not perform reset INQUIRE – Return current mode

Description

The function **icp_soft_reset** writes to the software reset bit which causes the IC-PCI hardware module to reset.

This is a low level function. The programmer should have detailed knowledge of how the IC-PCI hardware and software function.

Returned Values

Current status if *mode* = INQUIRE.

ICP_NO_ERROR – Operation completed successfully.

Hardware Modified

ICP_RST_32

Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com

icp_start_field – Set or Return the Starting Field

Syntax #include <icp.h>

short **icp_start_field**(MODCNF **icpmod*, short *mode*);

icpmod	Pointer to the IC-PCI module structure.
mode	Starting field: ICP_EVEN – Start acquire on the even field. ICP_ODD – Start acquire on the odd field. ICP_NEXT – Start acquire on the next field. INQUIRE – Return the current mode.

Description

The function **icp_start_field** programs which field the IC-PCI memory begins acquiring (writing) image data from the AM. The start field becomes the "first field" in IC-PCI memory. This function modifies the IC-PCI module structure.

This is a low level function. The programmer should have detailed knowledge of how the IC-PCI hardware and software function.

Returned Values

ICP_NEXT, ICP_EVEN, ICP_ODD.

ICP_NO_ERROR – Operation completed successfully.

Hardware Modified

ICP_FLDSEL

icp_start_ping_pong – Start a "Ping Pong" Acquistion and Transfer into a Host Memory Frame

Syntax #include <icp.h>

DWORD **icp_start_ping_pong**(MODCNF **icpmod*, short *host_fr1*, short *host_fr2*,
ICP_RECORD_MODE *mode*);

icpmod	Pointer to the IC-PCI module structure.
host_fr1	ID for the first host frame, a pointer returned by itx_create_host_frame.
host_fr2	ID for the second host frame, a pointer returned by itx_create_host_frame.
mode	Acquisition mode: ICP_ONCE – Acquire a single frame. ICP_CONT – Acquire continuously.

Description

The function **icp_start_ping_pong** initiates an unattended ping-pong operation. The function returns the context if successful. A ping-pong operation acquires a frame into frame *host_fr1* then acquires the next frame into

Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com

host_fr2. If the mode is ICP_CONT, the function then acquires into host_fr1, then host_fr2, and continues repeating this sequence until halted by icp_stop_ping_pong.

NOTE *During ping-pong operations, application programs must NOT call any other ITEX routine.*

This is a high level function. The programmer does not need in-depth knowledge of how the IC-PCI hardware and software function.

Returned Values

Context

ICP_BAD_ARG invalid host frame – *host_fr1* or *host_fr2* is not defined

Hardware Modified

None

NOTE *ITEX revision 2.7.1 redefines this function as a system-level (itx_) function. ITEX revision 2.6.3 for DOS and Windows 3.1, defines this function as a board-level (icp_) function.*

icp_start_VCR_record – Start a VCR Recording

Syntax

#include <icp.h>

short **icp_start_VCR_record**(MODCNF **icpmod*, short *host_frame_id*,
ICP_RECORD_MODE *mode)*;

icpmod Pointer to the IC-PCI module structure.

host_frame_id Frame ID for a host frame, a pointer returned by itx_create_host_frame.

mode Acquisition mode:
ICP_ONCE – Acquire a single sequence of frame areas.
ICP_CONT – Acquire continuously.

Description

The function **icp_start_VCR_record** records a frame from the frame buffer to the host frame *host_frame_id*. The source frame is specified in itx_create_host_frame. The source frame is in the structure pointed to by *host_frame_id*.

This routine records as many frames as fit in the host frame. Call itx_hframe_attrib to determine the number of frames recorded.

NOTE *During VCR operations, the application program must NOT call any other ITEX routine. If mode is ICP_CONT, the program must call icp_stop_VCR_record before calling any other ITEX routine.*

This is a high level function. The programmer does not need in-depth knowledge of how the IC-PCI hardware and software function.

Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com

Returned Values

ICP_NO_ERROR – Operation completed successfully.

ICP_BAD_ARG invalid host frame – *host_frame_id* not defined.

ICP_BAD_ARG invalid frame – *host_frame_id* points to an IC-PCI frame that is not defined, or has been deleted.

Hardware Modified

None

NOTE *ITEX revision 2.7.1 redefines this function as a system-level (itx_) function. ITEX revision 2.6.3 for DOS and Windows 3.1, defines this function as a board-level (icp_) function.*

icp_start_VCR_record_area – Start a VCR Recording of an Area

Syntax #include <icp.h>

short **icp_start_VCR_record_area**(MODCNF **icpmod*, short *host_frame_id*, short *x*, short *y*, short *dx*, short *dy*, ICP_RECORD_MODE *mode*);

icpmod	Pointer to the IC-PCI module structure.
host_frame_id	Frame ID for a host frame, a pointer returned by itx_create_host_frame.
x	Horizontal start of area in an IC-PCI frame.
y	vertical start of area in an IC-PCI frame.
dx	width of area in an IC-PCI frame.
dy	height of area in an IC-PCI frame.
mode	acquisition mode: ICP_ONCE – Acquire a single sequence of frame areas. ICP_CONT – Acquire continuously.

Description

The function **icp_start_VCR_record_area** records an area in a frame from the MVC module frame to the host frame. Coordinates *x,y* and size *dx,dy* define the MVC module memory area. itx_create_host_frame specifies the source frame, which is in the structure pointed to by *host_frame_id*.

This routine records as many frames as fit in the host frame. Call itx_hframe_attrib to determine the number of frames recorded.

During VCR operations,application programs must NOT call other ITEX routines. In ICP_CONT mode, the program must call icp_stop_VCR_record before any other ITEX routine.

This is a high level function. The programmer does not need in-depth knowledge of how the IC-PCI hardware and software function.

Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com

Returned Values

ICP_NO_ERROR – Operation completed successfully.

ICP_BAD_ARG invalid host frame – *host_frame_id* not defined.

ICP_BAD_ARG invalid frame – *host_frame_id* frame undefined or deleted.

Hardware Modified

None

NOTE *ITEX revision 2.7.1 redefines this function as a system-level (itx_) function. ITEX revision 2.6.3 for DOS and Windows 3.1, defines this function as a board-level (icp_) function.*

icp_stop_ping_pong – Stop a "Ping Pong" Transfer to Host Memory

Syntax #include <icp.h>

short **icp_stop_ping_pong**(MODCNF *_icpmod_);

icpmod Pointer to the IC-PCI module structure.

Description

The function **icp_stop_ping_pong** terminates a continuous ping-pong acquisition.

This is a high level function. The programmer does not need in-depth knowledge of how the IC-PCI hardware and software function.

NOTE *During ping-pong operations, application programs must NOT call any other ITEX routine.*

Returned Values

The number of frames transferred on the current pass.

Hardware Modified

None.

NOTE *ITEX software revision 2.7.1 redefines this function as a system-level function. ITEX revision 2.6.3 for DOS and Windows 3.1, defines this function as a board-level (icp_) function.*

icp_stop_VCR_record – Stop a VCR Recording

Syntax #include <icp.h>

short **icp_stop_VCR_record**(MODCNF *_icpmod_);

icpmod Pointer to the IC-PCI module structure.

Description

The function **icp_stop_VCR_record** stops a VCR recording already in progress. This function returns NULL if no recording is in process.

Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com

NOTE *During VCR operations, the application program must call this routine before any other ITEX routine.*

This is a high level function. The programmer does not need in-depth knowledge of how the hardware and software function.

Returned Values

Last full frame recorded. Range is from 0 to the number of frames in the sequence.

NULL if no recording was in process.

Hardware Modified

None

NOTE *ITEX software revision 2.7.1 redefines this function as a system-level function. ITEX revision 2.6.3 for DOS and Windows 3.1, defines this function as a board-level (icp_) function.*

icp_sync_mode – Set or Return the Sync Mode Flag

Syntax

#include <icp.h>

short **icp_sync_mode**(MODCNF **icpmod*, short *mode*);

icpmod	Pointer to the IC-PCI module structure.
mode	Sync mode: ICP_SYNC_MODE – Run in synchronous mode. ICP_ASYNC_MODE – Run in asynchronous mode. INQUIRE – Return the current mode.

Description

The function **icp_sync_mode** programs the sync mode for the IC-PCI. Application programs can be run in synchronous or asynchronous mode. This function modifies the IC-PCI module structure.

NOTE *This function is for future capabilities, currently the IC-PCI software runs in synchronous mode only.*

Returned Values

The current mode.

ICP_NO_ERROR – Operation completed successfully.

Hardware Modified

None

Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com

icp_system_colors_off – Disable DOS System Colors (DOS only)

Syntax #include <icpnovga.h>
void **icp_system_colors_off**();

Description

The function **icp_system_colors_off** frees the first 16 VGA colors, normally reserved for system use, for video display values. This function also saves the VGA DAC settings in an array and reprograms them for gray-scale image display. The video range becomes 0-255. This function is for DOS operation, not for Windows operation.

Normally the first 16 VGA colors 0-15 are reserved (system_colors_on) for system use, such as color frames, labels, etc. The down-side is that color dots may appear in the very dark areas of the image, as the values 0-15 are re-assigned to system display colors.

This is a low level function. The programmer should have detailed knowledge of how the IC-PCI hardware and software function.

Returned Values

None

Hardware Modified

None

icp_system_colors_on – Enable DOS System Colors (DOS only)

Syntax #include <icpnovga.h>
void **icp_system_colors_on**();

Description

The function **icp_system_colors_on** reserves the first 16 VGA colors 0-15 for system use, for frames, labels, etc. This function is for DOS operation, not for Windows operation.

This is a low level function. The programmer should have detailed knowledge of how the IC-PCI hardware and software function.

Returned Values

None

Hardware Modified

None

icp_system_colors_restore – Restore DOS System Colors (DOS only)

Syntax #include <icpnovga.h>
void **icp_system_colors_restore**();

Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com

Description

The function **icp_system_colors_restore** restores the VGA DAC colors to their original program start–up values previously saved by icp_system_colors_on or icp_system_colors_off. This function does nothing if the system colors were not saved.

This is a low level function. The programmer should have detailed knowledge of how the IC-PCI hardware and software function.

Returned Values

None

Hardware Modified

None

icp_trig_snap – Perform a Triggered Snap

Syntax #include <icp.h>

short **icp_trig_snap**(MODCNF **icpmod*, short *frame*);

icpmod	Pointer to the IC-PCI module structure.
frame	Frame to acquire into: a frame ID returned by icp_create_frame, or defined by the configuration file in “IC-VL compatible” mode.

Description

The function **icp_trig_snap** programs the IC-PCI for a triggered acquisition. The application program must have a subsequent call to icp_wait_trig. The occurrence of a trigger begins the acquisition.

Returned Values

ICP_NO_ERROR – Operation completed successfully.

ICP_BAD_ARG invalid mod pointer – *icpmod* not defined.

ICP_BAD_ARG invalid frame – *frame* not defined.

ICP_BAD_ARG can’t acquire SEQ frame by itself – *frame* is a sequential frame, and is not valid for a snap.

ICP_BAD_DMODCNF invalid AM mod pointer – *ammod* not defined.

ICP_OUT_FB_MEM ACQ dimensions exceed frame – Acquisition dimensions greater than *frame* size; either curdx > maxdx or curdy > maxdy

Hardware Modified

ICP_FLDSEL, ICP_TMODE, ICP_ACQMD, ICP_AQCSTART, ICP_PIXSZ

NOTE *Function added in version 2.6.2 for DOS32x and Win16, and version 2.6.1 for Win32.*

Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com

icp_vblank_status – Return the Camera Vertical Blank Status

Syntax #include <icp.h>
short **icp_vblank_status**(MODCNF **icpmod*);

icpmod Pointer to the IC-PCI module structure.

Description

The function **icp_vblank_status** reads the vertical blank status bit ICP_VBSTAT. This bit indicates the AM or camera vertical blank.

This is a low level function. The programmer should have detailed knowledge of how the IC-PCI hardware and software function.

Returned Values

ICP_IN_VB – camera is in vertical blank

ICP_OUT_VB – camera is not in vertical blank

Hardware Modified

None

icp_VCR_locate – Locate a Pixel in a Host Frame

Syntax #include <icp.h>
short **icp_VCR_locate**(MODCNF **icpmod*, short *host_frame_id*, WORD *pixel_x*, WORD *pixel_y*, DWORD **offset*);

icpmod Pointer to the IC-PCI module structure.
host_frame_id ID for a host frame, pointer returned by itx_create_host_frame.
pixel_x Horizontal coordinate within host frame.
pixel_y Vertical coordinate within host frame.
offset Pointer to the pixel, a returned value.

Description

The function **icp_VCR_locate** locates a particular pixel located at rectangular coordinates (*pixel_x*,*pixel_y*) within a host frame. *offset* returns the address displacement from pixel (0,0) for the specified the pixel.

NOTE *During VCR operations, the application program must NOT call any other ITEX routine.*

This is a high level function. The programmer does not need in-depth knowledge of how the IC-PCI hardware and software function.

Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com

Returned Values

offset = address offset to the pixel in host frame.

ITX_NO_ERROR – Operation completed successfully.

ITX_BAD_ARG invalid host frame – *host_frame_id* is not defined.

ITX_BAD_ARG invalid frame – *host_frame_id* points to an IC-PCI frame that is not defined, or has been deleted.

Hardware Modified

None

NOTE *ITEX software revision 2.7.1 redefines this function as a system-level (itx_) function. ITEX revision 2.6.3 for DOS and Windows 3.1, defines this function as a board-level (icp_) function.*

icp_VCR_play – Display a Recorded VCR Sequence

Syntax #include <icp.h>

short **icp_VCR_play**(MODCNF **icpmod*, short *host_frame_id*, WORD *start_frame*, WORD *n_frames*, DWORD *frames_per_sec*, WORD *direction*, short *x*, short *y*, short *dx*, short *dy*, HSINKIMG *display*);

icpmod	Pointer to the IC-PCI module structure.
host_frame_id	ID for the host frame, a pointer returned by itx_create_host_frame.
start_frame	Sequence number of the frame to start with.
n_frames	Number of frames in playback loop.
frames_per_sec	Speed of playback, in frames per second.
direction	direction to play back: ITX_FORWARD ITX_BACKWARD
x	Horizontal starting coordinate of the host frame area.
y	Vertical starting coordinate of the host frame area.
dx	Width of area in host frame.
dy	Height of area in host frame.
display	Returned by an ITEX create image sink routine.

Description

The function **itx_VCR_play** plays a previously recorded sequence of frames. A forward playback starts at *start_frame* and plays *n_frames*. A backward playback starts at (*start_frame* + *n_frames* -1).

NOTE *During VCR operations, the application program must NOT call any other ITEX routine.*

This is a high level function. The programmer does not need in-depth knowledge of how the IC-PCI hardware and software function.

Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com

Returned Values

ITX_CANNOT_PROFILE Cannot profile playback – Operating system clock error.

CANNOT_KEEP_UP – Requested play rate is faster than rate supported.

Hardware Modified

None

NOTE *ITEX software revision 2.7.1 redefines this function as a system-level function. ITEX revision 2.6.3 for DOS and Windows 3.1, defines this function as a board-level (icp_) function.*

icp_wait_acq – Wait for Completion of the Current Acquisition

Syntax #include <icp.h>

short **icp_wait_acq**(MODCNF **icpmod*, short *frame*);

icpmod Pointer to the IC-PCI module structure.

frame The frame ID:
a frame ID returned by icp_create_frame, or defined by the configuration file in "IC-VL compatible" mode.

Description

The function **icp_wait_acq** waits for the acquisition command bits change to 0, indicating acquisition completed.

NOTE *This function does not check to see if acquisition is continuous (grab) and may therefore time out. Do not program this function if the current acquisition is a grab (continuous).*

This is a medium level function. The programmer should be familiar with the IC-PCI hardware and software functional design.

Returned Values

ICP_NO_ERROR – Operation successfully completed.

ICP_BAD_ARG invalid ACQ buffer – Not a valid *frame* for acquisition.

ITX_TIMEOUT grab status bits did not self-clear *frame* – Acquisition bits did not clear (freeze/complete), continuous acquire or bits are stuck.

Hardware Modified

Monitors ICP_ACQMD without writing or modifying.

icp_wait_bmdone – Wait for Completion of Bus Master Transfer

Syntax #include <icp.h>

short **icp_wait_bmdone**(MODCNF **icpmod*);

icpmod Pointer to the IC-PCI module structure.

Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com

Description

The function **icp_wait_bmdone** waits for the completion of a IC-PCI bus master transfer.

This is a medium level function. The programmer should be familiar with the IC-PCI hardware and software functional design.

Returned Values

ICP_NO_ERROR – Operation completed successfully.

ITX_TIMEOUT BMDONE bit stuck – transfer did not complete, bus hung.

Hardware Modified

None

icp_wait_notvb – Wait until not in a Vertical Blank

Syntax

#include <icp.h>

short **icp_wait_notvb**(MODCNF **icpmod*, short *frame*, short *vbsync*);

icpmod	Pointer to the IC-PCI module structure.
frame	The frame ID: a frame ID returned by icp_create_frame, or defined by the configuration file in “IC-VL compatible” mode.
vbsync	Source of vertical blank to synchronize to ON – Wait for a vertical blank before waiting for not vertical blank. OFF – Do not wait for vertical blank.

Description

The function **icp_wait_notvb** waits for the end of vertical blank before returning. If the IC-PCI is not in vertical blank, the vbsync value determines if the function returns immediately, or waits for a vertical blank to occur.

This is a medium level function. The programmer should be familiar with the IC-PCI hardware and software functional design.

Returned Values

ICP_NO_ERROR – Operation successfully completed.

ITX_TIMEOUT VB status bit stuck – Vertical blank did not occur within timeout period.

Hardware Modified

None

Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com

icp_wait_start_vb – Wait for the Start of Next Camera Vertical Blank

Syntax #include <icp.h>

short **icp_wait_start_vb**(MODCNF **icpmod*, short *frame*);

icpmod	Pointer to the IC-PCI module structure.
frame	The frame ID: a frame ID returned by icp_create_frame, or defined by the configuration file in "IC-VL compatible" mode.

Description

The function **icp_wait_start_vb** waits for the occurrence of the next vertical blank. If the timing is already in vertical blank, the function waits for the current vertical blank to end, then waits for the next vertical blank to begin.

This is a medium level function. The programmer should be familiar with the IC-PCI hardware and software functional design.

Returned Values

ICP_NO_ERROR – Operation successfully completed.

ITX_TIMEOUT VB status bit stuck – Vertical blank did not occur within timeout period.

Hardware Modified

None

icp_wait_trig – Wait for Triggered Acquisition to Complete

Syntax #include <icp.h>

short **icp_wait_trig**(MODCNF **icpmod*, short *frame*, long *waittime*);

icpmod	Pointer to the IC-PCI module structure.
frame	The acquiring frame, a frame ID returned by icp_create_frame, or defined by the configuration file.
waittime	The maximum wait time, in milliseconds.

Description

The function **icp_wait_trig** monitors the status bit ICP_ACQMD. When ICP_ACQMD becomes true (snap completed), the function disables trigger mode and returns the status. Test the return value.

NOTE *Note: This function must be called after icp_trig_snap and before any other acquisition occurs. This function must be invoked following a call to icp_trig_snap, even if another method is used to determine when trigger has occurred.*

Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com

Returned Values

The triggered status:
TRUE = acquisition completed,
FALSE = acquisition did not complete.

ICP_BAD_ARG invalid mod pointer – *icpmod* not defined.

ICP_BAD_ARG invalid frame – *frame* not defined.

ICP_BAD_ARG can't ACQ seq frame by itself – *frame* points to a sequential frame, and is not valid for a snap.

Hardware Modified

ICP_TMODE, ICP_ACQMD

NOTE *Function added in version 2.6.2 for DOS32x and Win16, and version 2.6.1 for Win32.*

icp_wait_vb – Wait for the Next Vertical Blank

Syntax

#include <icp.h>
short **icp_wait_vb**(MODCNF **icpmod*, short *frame*);

icpmod	Pointer to the IC-PCI module structure.
frame	The frame ID: a frame ID returned by icp_create_frame, or defined by the configuration file in "IC-VL compatible" mode.

Description

The function **icp_wait_vb** verifies that the system is not in vertical blank (or waits to exit the current vertical blank) then waits for the occurrence of the next vertical blank that *starts a full frame*, according to the scan mode.

This is a medium level function. The programmer should be familiar with the IC-PCI hardware and software functional design.

Returned Values

ICP_NO_ERROR – Operation successfully completed.

ITX_TIMEOUT VB status bit stuck – Vertical blank did not occur within timeout period.

Hardware Modified

None

Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com

icp_whbline – Write a Horizontal Line of BYTE Values

Syntax #include <icp.h>

void **icp_whbline**(DWORD *offadr*, short *nwrites*, short *inc*, BYTE **b_ptr*);

offadr	The memory address returned from icp_set_lineio or icp_roi_set_lineio.
nwrites	Number of locations to write.
inc	Number of memory elements (BYTEs) to increment between writes, from 0 to 32767. If writing to an ROI this should be the number of bytes in the parent frame pixel depth. 0 – Write the same location. 1 – Increment one byte address between writes. 2 – Increment two byte addresses between writes. 3 – Increment three byte addresses between writes. . . . 32,767 – Write every 32,767th BYTE location.
b_ptr	Pointer to the array containing the BYTE values to write.

Description

The function **icp_whbline** writes a horizontal line of BYTE values to IC-PCI frame memory. You must select values for *nwrites* and *inc* that do not exceed the dimensions of the frame containing *offadr*. This function returns immediately (with no error) if *nwrites* is less than or equal to zero.

This is a medium level function. The programmer should be familiar with the IC-PCI hardware and software functional design.

Returned Values

None

Hardware Modified

None

Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com

icp_whdwline – Write a Horizontal Line of DWORD Values

Syntax #include <icp.h>

void **icp_whdwline**(DWORD *offadr*, short *nwrites*, short *inc*, DWORD **d_ptr*);

offadr	The memory address returned from icp_set_lineio or icp_roi_set_lineio.
nwrites	Number of locations to write.
inc	Number of memory elements (DWORDs) to increment between writes, from 0 to 32767. 0 – Write the same location. 1 – Increment one DWORD between writes. 2 – Increment two DWORDs between writes. 3 – Increment three DWORDS between writes. . . . 32767 – Write every 32,767th DWORD location.
d_ptr	Pointer to the array containing the DWORD values to write.

Description

The function **icp_whdwline** writes a horizontal line of DWORD values to IC-PCI frame memory. You must select values for *nwrites* and *inc* that do not exceed the dimensions of the frame containing *offadr*. Each DWORD written by icp_whdwline contains one 24-bit pixel, or two horizontally adjacent 16-bit pixels, or four horizontally adjacent 8-bit pixels. This function returns immediately (with no error) if *nwrites* is less than or equal to zero.

This is a medium level function. The programmer should be familiar with the IC-PCI hardware and software functional design.

Returned Values

None

Hardware Modified

None

Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com

icp_whwline – Write a Horizontal Line of WORD Values

Syntax #include <icp.h>

void **icp_whwline**(DWORD *offadr*, short *nwrites*, short *inc*, WORD **w_ptr*);

offadr	The memory address returned from icp_set_lineio or icp_roi_set_lineio.
nwrites	Number of locations to write.
inc	Number of memory elements (WORDs) to increment between writes, from 0 to 32767: 0 – Write the same location. 1 – Increment one WORD between writes. 2 – Increment two WORDs between writes. 3 – Increment three WORDs between writes. . . . 32,767 – Write every 32,767th WORD location.
w_ptr	Pointer to the array containing the WORD values to write.

Description

The function **icp_whwline** writes a horizontal line of WORD values to IC-PCI frame memory. You must select values for *nwrites* and *inc* that do not exceed the dimensions of the frame containing *offadr*. Each WORD written by icp_whwline contains one 16-bit pixel or two horizontally adjacent 8-bit pixels. This function returns immediately (with no error) if *nwrites* is less than or equal to zero.

This is a medium level function. The programmer should be familiar with the IC-PCI hardware and software functional design.

Returned Values

None

Hardware Modified

None

icp_wpix – Write a Pixel Value in IC-PCI Memory

Syntax #include <icp.h>

void **icp_wpix**(MODCNF **icpmod*, short *frame*, short *x*, short *y*, DWORD *pv*);

icpmod	Pointer to the IC-PCI module structure.
frame	The frame to write to: a frame ID returned by icp_create_frame, or defined by the configuration file.
x	Horizontal (X) coordinate in the range of 0 to frame width.
y	Vertical (Y) coordinate in the range of 0 to frame height.
pv	Value to write, range based on current pixel size.

Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com

Description

The function **icp_wpix** writes a single value in *frame* at location *x,y*. The function returns immediately (with no error) if *x* or *y* is greater than the frame width or height. This function writes one BYTE if the current pixel size is ICP_PIX8, or one WORD if the current size is ICP_PIX16, or one DWORD if the current pixel size is ICP_PIX24.

This is a high level function. The programmer does not need in-depth knowledge of how the IC-PCI hardware and software function.

Returned Values

ICP_NO_ERROR – Operation successfully completed.

ICP_BAD_ARG invalid frame – *frame* not defined, pointer returns NULL.

Hardware Modified

None

icp_write_area – Write an Area in IC-PCI Memory from an Array

Syntax

#include <icp.h>

short **icp_write_area**(MODCNF **icpmod*, short *frame*, short *x*, short *y*, short *dx*, short *dy*, DWORD **darray*);

icpmod	Pointer to the IC-PCI module structure.
frame	The frame to write to: a frame ID returned by icp_create_frame, or defined by the configuration file.
x	Horizontal starting coordinate.
y	Vertical starting coordinate.
dx	Width of area to write.
dy	Height of area to write.
darray	Pointer to an array containing the data to write.

Description

The function **icp_write_area** writes the contents of an array *darray* to a selected area of the selected image *frame*. The selected area of size *dx* by *dy* begins at location *x,y*. The function calls icp_set_lineio and writes horizontal lines (BYTE, WORD, or DWORD depending on the offset, boundary, and pixel size).

The size of the area is clipped to fit the size of the image frame. If *x+dx* is larger than the width of the frame, *dx* is limited to frame width minus *x*. If *y+dy* is larger than the frame height, *dy* is limited to frame height minus *y*.

This is a high level function. The programmer does not need in-depth knowledge of how the IC-PCI hardware and software function.

Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com

Returned Values

ICP_NO_ERROR – Operation completed successfully.

ICP_BAD_ARG invalid frame – frame not defined, pointer returns NULL.

ICP_BAD_ARG invalid x and/or y – x > frame width, or, y > frame height, or both.

Hardware Modified

None

icp_write_harea – Write Large Area in IC-PCI Memory from an Array (Win16 3.x only)

Syntax #include <icp.h>

short **icp_write_harea**(MODCNF **icpmod*, short *frame*, short *x*, short *y*, short *dx*, short *dy*, BYTE-HUGE **darray*);

icpmod	Pointer to the IC-PCI module structure.
frame	A frame ID for the frame to write to.
x	Horizontal starting coordinate.
y	Vertical starting coordinate.
dx	Width of area to write.
dy	Height of area to write.
darray	Pointer to an array containing the data to write.

Description

The function **icp_write_harea** is a function available under Windows only, that writes the contents of an array *darray* to a selected area of the selected image *frame*. The selected area of size *dx* by *dy* begins at location *x,y*. The function calls icp_set_lineio and writes horizontal lines (BYTE, WORD, or DWORD depending on the offset, boundary, and current pixel size). Bus master mode cannot be used to write to the IC-PCI. The function calls icp_set_lineio, and performs host writes to IC-PCI memory. icp_write_harea handles crossing 64K boundaries.

This is a high level function. The programmer does not need in-depth knowledge of how the IC-PCI hardware and software function.

The size of the area is clipped to fit the size of the image frame. If *x+dx* is larger than the width of the frame, *dx* is limited to frame width minus *x*. If *y+dy* is larger than the frame height, *dy* is limited to frame height minus *y*.

Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com

Returned Values

ICP_NO_ERROR – Operation completed successfully.

ICP_BAD_ARG invalid frame – *frame* not defined, pointer returns NULL.

ICP_BAD_ARG invalid x and/or y – x > frame width, or, y > frame height.

ICP_BAD_ARG Arg not multiple of 8 – *x*, *dx*, or resulting *offadr* must be multiples of 8.

ICP_BAD_ARG destination address not DWORD aligned.

Hardware Modified

None

icp_write_roi_area – Write an Area in a ROI from an Array

Syntax

#include <icp.h>

short **icp_write_roi_area**(MODCNF **icpmod*, ROIID *roi*, short *x*, short *y*, short *dx*, short *dy*, DWORD **darray*);

icpmod	Pointer to the IC-PCI module structure.
roi	ROI ID returned by icp_create_roi.
x	Horizontal starting coordinate (top-left corner).
y	Vertical starting coordinate (top left corner).
dx	Width of area to write.
dy	Height of area to write.
darray	Pointer to an array containing the data to write.

Description

The function **icp_write_roi_area** writes an area inside a defined ROI of size *dx* by *dy* starting at pixel location *x*,*y* with values from *darray*. This function calls icp_roi_set_lineio and writes horizontal lines of BYTE, WORD, or DWORD data, depending on the offset, boundary, and current pixel size. The depth, color, and interlace format is taken from the parent frame of the ROI.

This is a high level function. The programmer does not need in-depth knowledge of how the IC-PCI hardware and software function.

The size of the area is clipped to fit the size of the image frame. If *x*+*dx* is larger than the width of the frame, *dx* is limited to frame width minus *x*. If *y*+*dy* is larger than the frame height, *dy* is limited to frame height minus *y*.

NOTE *darray must have dimensions dx, dy, and pixel depth identical to the ROI.*

Returned Values

ICP_NO_ERROR – Operation completed successfully.

ICP_BAD_ARG invalid ROI – *roi* not defined, pointer returns NULL.

ICP_BAD_ARG invalid x and/or y – x >ROI width, or, y >ROI height, or both.

Hardware Modified

None

Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com

icp_write_roi_harea – Write a Large Area in a ROI from an Array (Windows 3.1 only)

Syntax #include <icp.h>

short **icp_write_roi_harea**(MODCNF **icpmod*, ROIID *roi*, short *x*, short *y*, short *dx*, short *dy*, BYTE-HUGE **darray*);

icpmod	Pointer to the IC-PCI module structure.
x	Horizontal starting coordinate.
y	Vertical starting coordinate.
dx	Width of area to write.
dy	Height of area to write.
darray	Pointer to an array containing the data to write.

Description

The function **icp_write_roi_harea** writes the contents of an array *darray* to the selected ROI, under Windows only. The area of size *dx* by *dy* begins at location *x,y*. The function writes horizontal lines (BYTE, WORD, or DWORD depending on the offset, boundary, and current pixel size). The function calls icp_roi_set_lineio, and performs host writes to IC-PCI memory. icp_write_roi_harea handles crossing 64K boundaries.

This is a high level function. The programmer does not need in-depth knowledge of how the IC-PCI hardware and software function.

The size of the area is clipped to fit the size of the ROI. If *x+dx* is larger than the width of the ROI, *dx* is limited to width minus *x*. If *y+dy* is larger than the ROI height, *dy* is limited to height minus *y*.

Returned Values

ICP_NO_ERROR – Operation completed successfully.

ICP_BAD_ARG invalid ROI – *roi* not defined, pointer returns NULL.

ICP_BAD_ARG invalid x and/or y – x >ROI width, or, y >ROI height, or both.

ICP_BAD_ARG destination address not DWORD aligned.

Hardware Modified

None

Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com

icp_wvbline – Write a Vertical Line of BYTE Values

Syntax #include <icp.h>

void **icp_wvbline**(DWORD *offadr*, short *nwrites*, short *linesz*, DWORD *ilacedata*, BYTE **b_ptr*);

offadr	The memory address returned from icp_set_lineio or icp_roi_set_lineio.
nwrites	Number of locations to write.
linesz	Horizontal size (in bytes) of the parent frame, returned as *xsz* by icp_set_lineio or icp_roi_set_lineio.
ilacedata	Interlace status of the parent frame, returned by icp_set_lineio or icp_roi_set_lineio.
b_ptr	Pointer to the array containing the BYTE values to write.

Description

The function **icp_wvbline** writes a vertical line of BYTE values in IC-PCI frame memory. You must select a value for *nwrites* that does not exceed the dimensions of the frame that containing *offadr*. This function returns immediately (with no error) if *nwrites* is less than or equal to zero.

This is a medium level function. The programmer should be familiar with the IC-PCI hardware and software functional design.

Returned Values

None

Hardware Modified

None

icp_wvdwline – Write a Vertical Line of DWORD Values

Syntax #include <icp.h>

void **icp_wvdwline**(DWORD *offadr*, short *nwrites*, short *linesz*, DWORD *ilacedata*, DWORD **d_ptr*);

offadr	The memory address returned by icp_set_lineio or icp_roi_set_lineio.
nwrites	Number of locations to write.
linesz	Horizontal size (in bytes) of the parent frame, returned as *xsz* by icp_set_lineio or icp_roi_set_lineio.
ilacedata	Interlace status of the parent frame, returned by icp_set_lineio or icp_roi_set_lineio.
d_ptr	Pointer to the array containing the DWORD values to write.

Description

The function **icp_wvdwline** writes a vertical line of DWORD values in IC-PCI frame memory. You must select a value for *nwrites* that does not exceed the dimensions of the frame containing *offadr*. This function returns immediately (with no error) if *nwrites* is less than or equal to zero. Each DWORD written by icp_wvdwline contains one 24-bit pixel, or two horizontally adjacent 16-bit pixels, or four horizontally adjacent 8-bit pixels.

This is a medium level function. The programmer should be familiar with the IC-PCI hardware and software functional design.

Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com

Returned Values
None

Hardware Modified
None

icp_wvwline – Write a Vertical Line of WORD Values

Syntax #include <icp.h>

void **icp_wvwline**(DWORD *offadr*, short *nwrites*, short *linesz*, DWORD *ilacedata*, WORD **w_ptr*);

offadr	The memory address returned by icp_set_lineio or icp_roi_set_lineio.
nwrites	Number of locations to write.
linesz	Horizontal size (in bytes) of the parent frame, returned as *xsz* by icp_set_lineio or icp_roi_set_lineio.
ilacedata	Interlace status of the parent frame, returned by icp_set_lineio or icp_roi_set_lineio.
w_ptr	Pointer to the array containing the DWORD values to write.

Description
The function **icp_wvwline** writes a vertical line of WORD values in IC-PCI memory. You must select a value for *nwrites* that does not exceed the dimensions of the frame containing *offadr*. This function returns immediately (with no error) if *nwrites* is less than or equal to zero. Each word written by icp_wvwline contains one 16-bit pixel or two (horizontally adjacent) 8-bit pixels.

This is a medium level function. The programmer should be familiar with the IC-PCI hardware and software functional design.

Returned Values
None

Hardware Modified
None

icp_wxzoom – Set or Return the Horizontal Write Zoom Factor

Syntax #include <icp.h>

short **icp_wxzoom**(MODCNF **icpmod*, short *frame*, short *factor*);

icpmod	Pointer to the IC-PCI module structure.
frame	The frame ID, returned by icp_create_frame.
factor	Decimation factor: ICP_ZOOM_NO – Zoom (decimation) by 1; acquire all pixels. ICP_ZOOM_X2 – Zoom (decimation) by 2; acquire every second even pixel. ICP_ZOOM_X4 – Zoom (decimation) by 4; acquire every fourth even pixel. ICP_ZOOM_X8 – Zoom (decimation) by 8; acquire every eighth even pixel.

Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com

Description

The function **icp_wxzoom** sets the horizontal zoom factor (x decimation factor) for writing video data (acquire) into the IC-PCI memory. The actual line length is divided by 2, 4 or 8. Decimation by 8 is only available with 8-bit images.

If factor = ICP_ZOOM_X2, only every other pixel (0, 2, 4, 6, . . .) is acquired (written into memory) and the stored image is 1/2 the width of the camera frame. If factor is ICP_ZOOM_X4, only every fourth pixel is acquired (0, 4, 8, . . .) and the stored image is 1/4 the width of the camera frame. The acquire always starts with pixel 0.

Returned Values

ICP_NO_ERROR – Operation completed successfully.

ICP_BAD_ARG – Cannot do decimation with Board Rev. < 3 – older board revisions are not capable of write zoom function.

ICP_BAD_ARG – Invalid frame – *frame* not defined.

ICP_BAD_ARG – Invalid argument *factor* – not set to a defined value.

Hardware Modified

ICP_WXZOOM

NOTE *New function added in ITEX revision 2.8.0 for Windows NT and Windows 95. Not applicable to Win16 DOS32x or Windows 3.11.*

Example

```
decimate=icp_wxzoom(icpmod,frame0,ICP_ZOOM_X4);
```

icp_wyzoom – Set or Return the Vertical Write Zoom Factor

Syntax #include <icp.h>

short **icp_wyzoom**(MODCNF *icpmod, short frame, short factor);

icpmod	Pointer to the IC-PCI module structure.
frame	The frame ID, returned by icp_create_frame.
factor	Decimation factor: ICP_ZOOM_NO – Zoom by 1; acquire all lines. ICP_ZOOM_X2 – Zoom by 2; acquire every second line. ICP_ZOOM_X4 – Zoom by 4; acquire every fourth line. ICP_ZOOM_X8 – Zoom by 8; acquire every eighth line.

Description

The function **icp_wyzoom** sets the vertical zoom factor (y decimation factor) for writing video data (acquire) into the IC-PCI memory. The actual image size is divided by 2, 4 or 8. Decimation by 8 is only available with 8-bit images.

If *factor* = ICP_ZOOM_X2, only every other line (0, 2, 4, . . .) is acquired (written into memory) and the stored image is 1/2 the number of lines in the camera frame. For interlaced camera images, only the even field is ac-

Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com

quired. If *factor* is ICP_ZOOM_X4, only every fourth line is acquired (0, 4, 8, . . .) and the stored image is 1/4 the number of lines in the camera frame. The acquire always starts with line 0.

NOTE *The acquired image size must be a multiple of eight pixels.*

Returned Values

ICP_NO_ERROR – Operation completed successfully.

ICP_BAD_ARG – Cannot do decimation with Board Rev. < 3 – older board revisions are not capable of write zoom function.

ICP_BAD_ARG – Invalid frame – *frame* not defined.

ICP_BAD_ARG – Invalid argument *factor* – not set to a defined value.

Hardware Modified

ICP_WYZOOM

NOTE *New function added in ITEX revision 2.8.0 for Windows NT and Windows 95. Not applicable to Win16 DOS32x or Windows 3.11.*

Example

```
decimate=icp_wyzoom(icpmod,frame0,ICP_ZOOM_X4);
```


Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com

Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com

APPENDIX A
IC-PCI REGISTERS

This appendix lists the registers available on the IC-PCI. Some registers only support 32-bit access. Use the system-level functions read_reg32 and write_reg32 to access the registers with "32" in their symbolic names. Use the system–level functions read_reg and write_reg to access the other registers.

PCI Interface Control Registers

Table A-1 gives the symbolic defines for the Interface Control registers. The Interface Control registers are all 32-bit access only. You must use the system-level functions read_reg32 and write_reg32 for these registers and bits. Figure A-1 illustrates how the individual bits are mapped in the Interface Control registers.

IC-PCI Control Registers

Table A-2 (pages A-3 through A-5) gives the symbolic defines for the IC-PCI Control registers, and some value ranges where symbols are not used. The Control registers are all 16-bit access only. You must use the system-level functions read_reg and write_reg to access these registers and bits. Figure A-2 (pages A-7 through A-8) illustrates how the individual bits are mapped in the IC-PCI Control registers.

Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com

Table A-1. Interface Control Registers

Symbolic Name	Description	Symbolic Defines
ICP_MBOX1_32	Mailbox 1 register	
ICP_MBOX2_32	Mailbox 2 register	
ICP_MBOX3_32	Mailbox 3 register	
ICP_MBOX4_32	Mailbox 4 register	
ICP_BMDST_32	Bus Master Destination Address	
ICP_BMXC_32	Bus Master Transfer Size	
ICP_INTCTL_32	Interrupts Control register	
ICP_INTEN_32	bus interrupts enable	ICP_DISABLE ICP_ENABLE
ICP_BINTEN_32	bus interrupt enable	ICP_DISABLE ICP_ENABLE
ICP_INTST_32		*read-only*
ICP_BINTST_32	bus interrupt status	*read-only*
ICP_MAINT_32	master abort interrupt status	*read-only*
ICP_TAINT_32	target abort interrupt status	*read-only*
ICP_BMCTL_32	Bus Master Control register	
ICP_FIFOFL_32	FIFO full status	*read-only*
ICP_FIFO4P_32	FIFO 4+ DWORDs	*read-only*
ICP_FIFOEM_32	FIFO empty status	*read-only*
ICP_BMDONE_32	bus master done status	*read-only*
ICP_BMREQ_32	bus master request	ICP_HALT ICP_START
ICP_RST_32	board reset	ICP_DISABLE ICP_ENABLE
ICP_FIFOCLR_32	FIFO clear	ICP_DISABLE ICP_ENABLE

Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com

Table A-2. Control Registers

Symbolic Name	*Description*	*Symbolic Defines*
ICP_CON	Control register	
ICP_BMSTREN	bus master mode enable	ICP_DISABLE ICP_ENABLE
ICP_BMILACE	bus master interlace control	ICP_DISABLE ICP_ENABLE
ICP_PIXSZ	pixel size	ICP_PIX8 ICP_PIX16 ICP_PIX24
ICP_MEMCFG	memory size *(wrap address)*	ICP_2M ICP_4M
ICP_VBSTAT	vertical blank status	*read-only*
ICP_FLDSTAT	field status	*read-only*
ICP_SMSTAT	scan mode status	*read-only*
ICP_AMP	AM present status	*read-only*
ICP_ACQCON	Acquisition Control register	
ICP_TMODE	trigger mode	ICP_DISABLED ICP_ENABLED
ICP_FCNT	frame count	(1 to 8)
ICP_FLDSEL	starting field select	ICP_NEXT ICP_EVEN ICP_ODD
ICP_ACQMD	acquisition command	ICP_FREEZE ICP_SNAP ICP_GRAB
ICP_NEWAQ	new acquire command status	*read-only*
ICP_GSTAT	grab (in progress) status	*read-only*
ICP_FSTRT	field start status	ICP_EVEN ICP_ODD

(continued)

Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com

Table A-2 Continued. Control Registers

Symbolic Name	*Description*	*Symbolic Defines*
ICP_PITCH	Pitch register	0–4095 *(multiplier changes with pixel size)*
ICP_AOIX	AOI Horizontal Size register	0–4095 *(multiplier changes with pixel size)*
ICP_BMSF	Bus Master First Field Start Address; BMSFL and BMSFH	(address bits 21–3) (BMSF x 8= address)
ICP_BMSFL	Bus Master First Field Start Address Low	0–1023 *(address bits 12–3)*
ICP_BMSFH	Bus Master First Field Start Address High	0–511 *(address bits 21–13)*
ICP_BMSS	Bus Master Second Field Start Address; BMSSL and BMSSH	(address bits 21–3) (BMSS x 8 = address)
ICP_BMSSL	Bus Master Second Field Start Address Low	0–1023 *(address bits 12–3)*
ICP_BMSSH	Bus Master Second Field Start Address High	0–511 *(address bits 21–13)*
ICP_AQSTART	Acquisition Start Address	0–1023 *((AQSTART x 4096) = start address*
ICP_AQADR	Acquisition Address Status register	*read only* 0–1023 *((AQADR+1) x4095) = address*
ICP_INTADR	Acquisition Address Interrupt register	0–63 *(INTADR x 65536) = address*

(continued)

Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com

Table A-2 Continued. Control Registers

Symbolic Name	*Description*	*Symbolic Defines*
ICP_INTCON	Interrupts Control register	
ICP_VBIEN	vertical blank interrupt enable	ICP_DISABLE ICP_ENABLE
ICP_OVBIEN	odd vertical blank interrupt enable	ICP_DISABLE ICP_ENABLE
ICP_EVBIEN	even vertical blank interrupt enable	ICP_DISABLE ICP_ENABLE
ICP_AMIEN	AM interrupt enable	ICP_DISABLE ICP_ENABLE
ICP_TRIGIEN	trigger interrupt enable	ICP_DISABLE ICP_ENABLE
ICP_ADRIEN	address interrupt enable	ICP_DISABLE ICP_ENABLE
ICP_AQIEN	acquisition interrupt enable	ICP_DISABLE ICP_ENABLE
ICP_VBINT	vertical blank interrupt status	read-only
ICP_OVBINT	odd vertical blank interrupt status	read-only
ICP_EVBINT	even vertical blank interrupt status	*read-only*
ICP_AMINT	AM interrupt status	*read-only*
ICP_TRIGINT	trigger interrupt status	*read-only*
ICP_ADRINT	address interrupt status	*read-only*
ICP_AQINT	acquisition interrupt status	*read-only*
ICP_MDATA	MASK register	0xFFFF

Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com

Address offset		*Register*
0x0	32-bit data value	MBOX1_32
0x4	32-bit data value	MBOX2_32
0x8	32-bit data value	MBOX3_32
0xC	32-bit data value	MBOX4_32
0x24	32-bit address	BMDST_32
0x28	Reserved(10) | 23-bit count value	BMXC_32

Address offset									*Register*
0x38	0	0	0	0	0	0	0	0	INTCTL_32
	Reserved	BINTEN_32	0	INTEN_32	1	1	1	1	
	Reserved	0	TAINT_32	MAINT_32	Reserved	BINTST_32	INTST_32	Reserved	
	0	0	0	0	0	0	0	0	
0x3C	BMDONE	Reserved	FIFOEM_32	FIFO4P_32	FIFOFL_32	Reserved	Reserved	Reserved	BMCTL_32
	0	0	0	0	0	BMREQ_32	1	1	
	0	0	0	0	0	0	0	0	
	0	0	0	0	Reserved	FIFOCLR	Reserved	RST_32	

Figure A-1. PCI Interface Control Registers

Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com

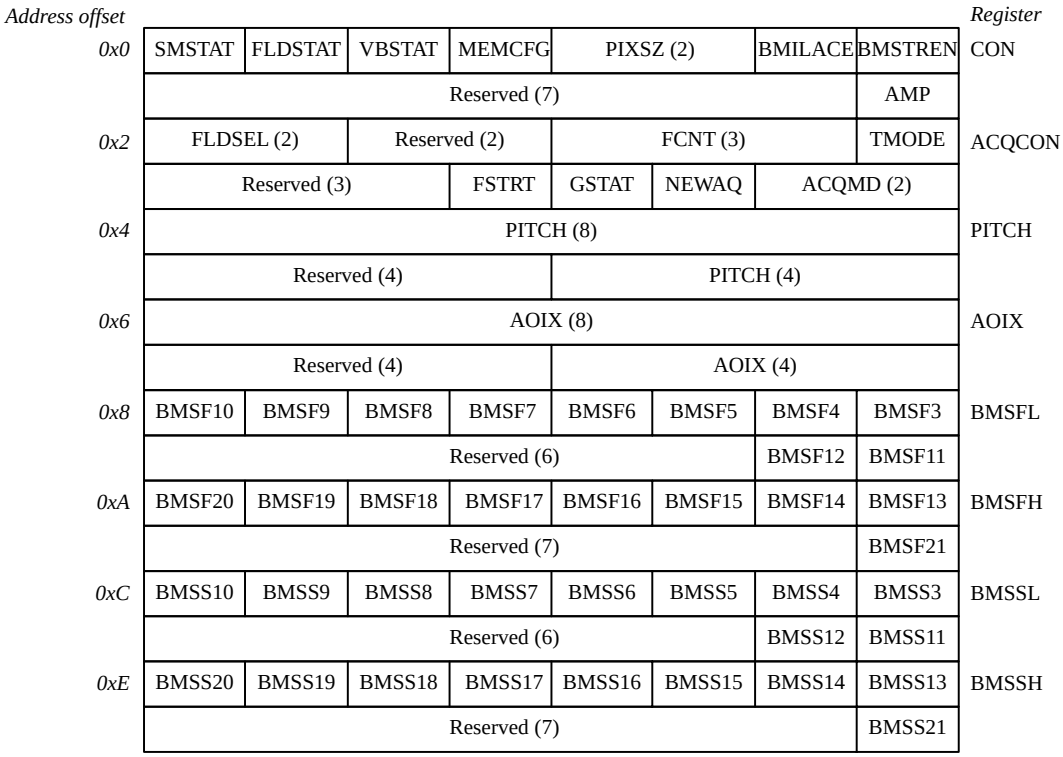


Figure A-2. Control Registers

Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com

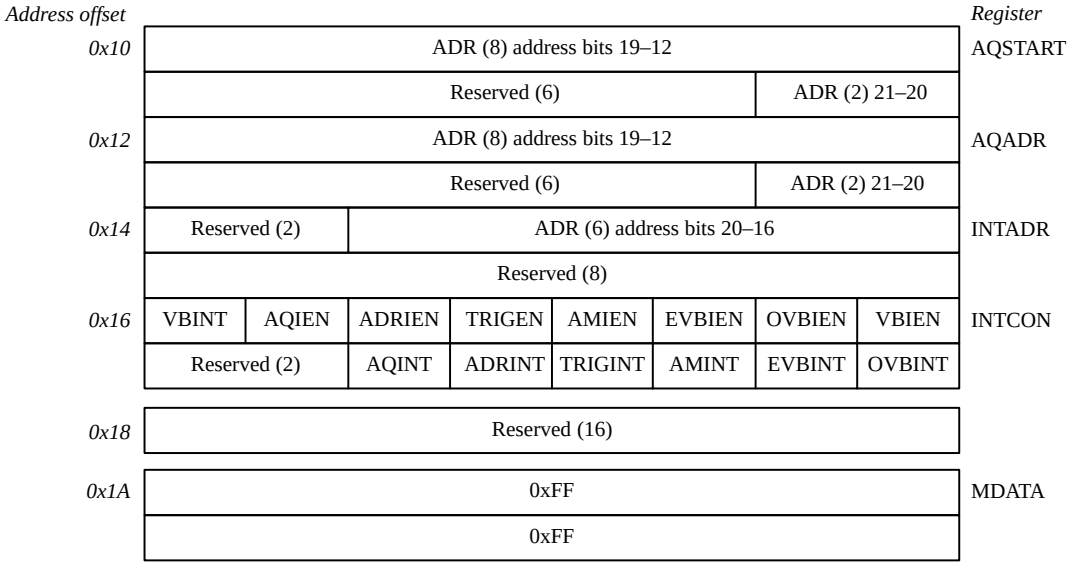


Figure A-2. Control Registers — continued.

Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com

APPENDIX B
CONFIGURATION FILES

The Configurator Utility creates an ASCII file and a Binary file describing the hardware configuration. The ITEX library uses the binary version. If you choose to edit the ASCII version, you must create the binary file, according to the instructions found on page B–2.

The ASCII configuration file lists all the translator boards, MVC mother boards, plug-in modules, and camera ports as “groups” of parameters. Each group is followed by a series of tokens.

Information about Comment fields, Whitespace, Number fields, Order of Parameters, Group numbers, System numbers, and Sequence numbers are covered at the back of this Appendix.

Sample Configuration Files

The software distribution contains sample configuration files, located in the “icpci/config” directory.

icptest.txt	IC-PCI and AM-VS
master.txt	Superset of all MVC boards (VME, VL-bus, and PCI-bus) showing all configuration parameters

The file “icptest.txt” shows both the mandatory and optional parameters for the IC-PCI, and only the mandatory parameters for the AMs. The “token” files “amvs.tok”, “amc1.tok”, “amdig.tok”, and “amfa.tok” give the complete list of parameters for the AMs. If all the “commented out” lines were removed, “icvltest.txt” would look like the following example.

“icptest.txt” uncommented lines

```
[System:0]
[ICP]                   Group name. Must be called ICP
    Latency = 240
[AM-VS]                 Group name
    Slot_Number = 0     AM location on the mother board
[AT-PCI]
```

The file “master.txt” shows the mandatory and optional parameters for all MVC 150/40 boards, modules, and supported translators in their suggested order. The translators are for the VME system. Only the IC-PCI is discussed in this Appendix. Refer to the software manual for your AM for additional information on AM configurator tokens.

Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com

Creating the Binary Configuration File

After you have edited the file to match your hardware system, you must convert it to binary, using the utility “bincnf”. The syntax of this command is:

bincnf *textfilename binaryfilename*

Create the binary configuration file from “icptest.txt” by typing:

> **bincnf icptest**

This example assumes your file is called “icptest.txt”. The output is a binary file “icptest.cnf”. This the default name used by all the IC-PCI example programs on the distribution.

Leave this file “icptest.cnf” in the “icpci/config” directory so the ITEX software can locate it with the environment variable CNF15040.

The extensions “.txt” and “.cnf” are optional. The utility will supply them if you do not.

If you do not supply an output filename, the utility uses the input text file name, deletes the extension “.txt” (if present) and adds the extension “.cnf”. The following example reads a text file named “myown.txt” and outputs a binary file named “myown.cnf”:

bincnf myown

The name “myown.cnf” will *not* be found by any of the example programs, because all examples use the default name “icpci.cnf”. To use a configuration file by any other name, you must supply the file name in your application program; for example:

```
.. . .
itx_load_cnf(“myown.cnf”)
. . .
```


Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com

Required Parameters for IC-PCI Configuration

The necessary parameters for configuring the IC-PCI are:

Minimum Necessary Parameters

[ICP]	*IC-PCI board group name. Must be ICP*

The register and memory base addresses are set by the PCI bus BIOS.

The default values for undefined parameters are: 4MB memory, Bus Master Mode enabled. These values are automatically used if the parameters are not specified in your configuration.

NOTE *When using a Dalsa CA-D4 camera with the AM-MTD and an IC-PCI mother board, you must specify the following configuration file token in the ICP group, not the AM group:*

PIXSZ_Override = 16

Do not use this token for Dalsa CT-E1 cameras or other AMs. For example:

IC-PCI Minimum Parameters for Dalsa CA-D4 with AM-MTD

```
[ICP:0]
    PIXSZ_Override=16
[AMMTD:0]
    Slot_Number=0
[AT-PCI]
```

Optional Parameters for IC-PCI Configuration

The following table shows the optional IC-PCI parameters. You can also change all these parameters in the program with the module structure functions.

Optional Parameters

The Latency Timer is the number of PCI clocks that the IC-PCI is guaranteed to remain the PCI Bus master once it is granted. ITI recommends using a value close to the maximum.

Latency = 240 *0 to 248 clocks*

You may specify a PC interrupt line, IRQ0 through IRQ15.

IRQ_Line = 4 *Uses PC IRQ4 line. Lines 0-15 are availible.*

You can specify initial 32-bit values for the PCI Mailbox registers. The use is defined by the application.

Mailbox1 = 0x0000
Mailbox2 = 0x0000
Mailbox3 = 0x0000
Mailbox4 = 0x0000

(continued)

Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com

Optional IC-PCI Parameters Continued.

You can enable the AM trigger mode for acquisiton. The default is "disabled"
AM_Trigger_enabled

You may set the acquisition frame count from 1 to 8; but, you must define a sequential frame in your application program to use the frame count.
Frame_Count = 1 *1 to 8 frames*

You may specify the first field in an interlaced frame acquire.
Start_Field = Next_Field
Start_Field = Next_even_field
Start_Field = Next_odd_field

You may set up the IC-PCI to emulate the IC-VL (or IML). In this mode you define your frames in the configuraiton file instead of in the application program. The default is disabled.
IML_Compatible_Mode *enables emulation, requires frame deffinition*

If compatible mode is enabled, you MUST define the frames. You must specify horizontal size (DX) vertical size (DY) and pixel depth (one color type) for each frame ID defiened.The possible frame definitions are: FrameA0, FrameA1, FrameB0, FrameB1, FrameRGB. You must specify all frames used in your application program, but you do not have to specify all 5 frames avalible.
FrameA0 *frame ID*
IML_Mode_DX = 512 *horizontal size 8 to 4096 pixels*
IML_Mode_DY = 512 *vertical size 1 to 4096 lines*
Color_Type = Mono *8-bit monochrome frame*
Color_Type = Red *8-bit red frame in RGB*
Color_Type = Green *8-bit green framein RGB*
Color_Type = Blue *8-bit blue frame in RGB*
Color_Type = RGB *24-bit RGB frame*

If the AM-MTD transferred data width is different from the input pixel size, the token PIXSZ_Override must indicate the transfer size:
PIXSZ_Override=16 *# bits in data transfer*

You may specify the time to wait for an acquisiton to complete. This is most useful with large-format frames. The default is 2 seconds.
Acq_Timeout= 20000 *time in miliseconds*

end of IC-PCI parameters

Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com

ASCII Configuration File Conventions

The following sections explain conventions used in formatting configuration files for the IC–PCI.

Comment Fields

Comments may appear anywhere within the configuration file, and can use two different formats:

d. "Remark" format #
Where text is ignored on any line starting with a hash mark or pound sign "#"

e. "assembly style" format ;
Where text is ignored on any line starting with a semicolon ";"

White Space

White space (spaces, tabs) between tokens, numbers, and parameters is ignored. Text strings must be terminated in the same line.

Number Fields

Numbers may be signed or unsigned decimal numbers (123 or –436), hexadecimal numbers (0x5AC or 5ACH), or floating point numbers (1.23).

Order of Parameters

The order of parameters shown in this section should be used. Parameter values occurring later in the file can override parameters occurring earlier. Each module has necessary minimum parameters, and optional parameters. The minimum necessary parameters are mandatory in any configuration file for that board or module.

Group Names

A "group" begins with a name in square brackets along with its sequence number, and is followed by a list of parameters. A subgroup immediately follows a group. All the daughter modules on a mother board come immediately after their mother board, the camera port group(s) follow immediately after the acquisition module group. A system should always have a sequence number. (See Example: Group Names on page B–5).

Example: Group Names

[System:0]	*System Group name, sequence number 0*
[ICP]	*Mother board Group name*
[AM-VS]	*Daughter board Group name*
[CameraPort:0]	*Camera Group names*
[CameraPort:1]	
[CameraPort:2]	
[CameraPort:3]	
[AT-PCI]	*host bus or translator must be the last entry*

Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com

Each module in a hardware configuration is uniquely identified by a combination of its system number, slot number, and sequence number.

Since the IC-PCI and AM form a complete system, no further understanding of slot, sequence, and system numbers is needed.

Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com

INDEX

A

acquire
- continuously, **icp_grab**, 2-41
- large–format
  - **icp_large_format_snap**, 2-45
  - triggered, **icp_large_format_trig_snap**, 2-46
- large–format frames, **icp_host_large_format_acquire**, 2-42
- single image, **icp_snap**, 2-71
- single image asynchronously, **icp_snap_async**, 2-72
- triggered
  - **icp_large_format_trig_snap**, large format frames, 2-46
  - **icp_trig_snap**, single image, 2-80
  - **icp_wait_trig**, wait for end of icp_trig_snap, 2-85

acquisition
- command bits, set or return, **icp_acqbits**, 2-2
- command pending, **icp_acq_pending**, 2-4
- control, 1-3
- frame count, **icp_fcount**, 2-32
- start address, in IC-PCI memory
  - **icp_get_acq_start_addr**, 2-36
  - **icp_put_acq_start_addr**, 2-50
- stop continuous to IC–PCI frame, **icp_freeze**, 2-35
- test for in progress, **icp_acq_in_progress**, 2-3
- wait for acquisition to complete, **icp_wait_acq**, 2-83

acquisition functions, 1-9

acquisition memory block, **icp_acq_addr_status**, 2-2

active, UBM frame, **icp_get_active_ubm_frame**, 2-37

address
- bus master destination address
  - **icp_get_bm_dst_addr**, 2-38
  - **icp_put_bm_dst_addr**, 2-51
- bus master first field
  - **icp_get_bm_firstf_start**, 2-39
  - **icp_put_bm_firstf_start**, 2-52
- bus master second field
  - **icp_get_bm_secondf_start**, 2-39
  - **icp_put_bm_secondf_start**, 2-52

AM
- interface, 1-3
- returning pointer to, **icp_get_slot0**, 2-41

AM trigger, IC-PCI, B–4

AOI
- short definition, iv
- versus ROI. *See* system–level software manual

area
- define area for display utility, **icp_display_area**, 2-28
- read form ROI to buffer, **icp_read_roi_area**, 2-57
- read from IC-PCI frame into buffer, **icp_read_area**, 2-53
- read huge area from IC-PCI frame into buffer, **icp_read_harea**, (Windows 3.1 only), 2-55
- read huge area from ROI to buffer, **icp_read_roi_harea**, 2-58
- read to host adding destination pitch, **icp_read_dest_pitch_area**, 2-54
- start VCR recording to host frame, **icp_start_VCR_record_area**, 2-76
- write area from an array, in IC-PCI ROI, **icp_write_roi_area**, 2-92
- write huge area from array to IC-PCI frame, **icp_write_harea**, 2-91
- write huge area from array to IC-PCI ROI, **icp_write_roi_harea**, 2-93
- write IC-PCI frame area from an array, **icp_write_area**, 2-90

area functions, 1-11

area of IC-PCI memory, displaying, **icp_dump_mem**, 2-31

area of IC-PCI memory ROI, displaying, **icp_dump_roi_mem**, 2-32

argument types, 1-5

asynchronous
- snap image, **icp_snap_async**, 2-72
- snap sequence, **icp_seq_snap_async**, 2-68

B

bus master
- active UBM frame, **icp_get_active_ubm_frame**, 2-37
- AOI horizontal size
  - **icp_get_bm_aoix**, 2-38
  - **icp_put_bm_aoix**, 2-50
- destination address
  - **icp_get_bm_dst_addr**, 2-38
  - **icp_put_bm_dst_addr**, 2-51
- enable flag, **icp_bm_mode**, 2-11

Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com

FIFO contains 4 DWORDs, **icp_bm_fifo_4P**, 2-8
FIFO empty, **icp_bm_fifo_empty**, 2-8
FIFO full, **icp_bm_fifo_full**, 2-9
first field starting address
icp_get_bm_firstf_start, 2-39
icp_put_bm_firstf_start, 2-52
idle UBM frame, **icp_get_idle_ubm_frame**, 2-40
image pitch, **icp_image_pitch**, 2-44
latency timer, **icp_latency_timer**, 2-46
lock memory regions, **icp_bm_lock**, 2-10
read area from ROI to buffer, **icp_read_roi_area**, 2-57
read area in IC-PCI frame to host buffer, **icp_read_area**, 2-53
read huge area from IC-PCI to host buffer, **icp_read_harea**, (Windows 3.1 only), 2-55
read huge area from ROI to buffer, **icp_read_roi_harea**, 2-58
read IC–PCI memory to locked host region, **icp_bm_read**, 2-12
read IC–PCI memory to subregion of locked host region, **icp_bm_read_subregion**, 2-12
request PCI bus, **icp_bus_request**, 2-14
return interlace status, **icp_bm_ilace**, 2-9
second field starting address
icp_get_bm_secondf_start, 2-39
icp_put_bm_secondf_start, 2-52
set or return, transfer count, **icp_bm_count**, 2-7
transfer done, **icp_bm_done**, 2-8
unlock host regions, **icp_bm_unlock**, 2-13
wait for transfer to complete, **icp_wait_bmdone**, 2-83

bus request, **icp_bus_request**, 2-14

BYTEs, clearing
horizontal line, **icp_chbline**, 2-15
vertical line, **icp_cvbline**, 2-24

BYTEs, reading
horizontal line, **icp_rhbline**, 2-59
vertical line, **icp_rvbline**, 2-65

BYTEs, writing
horizontal line, **icp_whbline**, 2-87
vertical line, **icp_wvbline**, 2-94

C

camera
field status, **icp_field_status**, 2-33
frame dimensions, **icp_get_acq_dim**, 2-36
scan mode, **icp_scanmd_status**, 2-67
vertical blank status, **icp_vblank_status**, 2-81
wait for start of camera vertical blank, **icp_wait_start_vb**, 2-85

clear
FIFO, **icp_fifo_reset**, 2-34
frame memory area, **icp_clr_area**, 2-17
ROI, **icp_clr_roi**, 2-18
selected image frame, **icp_clr_frame**, 2-18

clearing horizontal line
bytes, **icp_chbline**, 2-15
DWORD, **icp_chdwline**, 2-15
WORD, **icp_chwline**, 2-16

clearing vertical line
byte, **icp_cvbline**, 2-24
DWORD, **icp_cvdwline**, 2-25
WORD, **icp_cvwline**, 2-25

color, 1-4

command, acquisition command bits, icp_acqbits, 2-2
See also itx_acqbits

complete, wait for current acquisition to complete, **icp_wait_acq**, 2-83

continuous, acquire, **icp_grab**, 2-41

copy
area in IC-PCI frame, **icp_cp_area**, 2-19
ROI in IC-PCI frame, **icp_cp_roi**, 2-20

count
return acquisition frame count, **icp_fcount**, 2-32
set or return bus master transfer count, **icp_bm_count**, 2-7

create
frame in IC-PCI memory, **icp_create_frame**, 2-21
ROI in IC-PCI frame, **icp_create_roi**, 2-22
sequence frame, in IC-PCI memory, **icp_create_seq_frames**, 2-23

D

daughter board, returning pointer to, **icp_get_slot0**, 2-41

decimation, write zoom
icp_wxzoom, horizontal write zoom, 2-95
icp_wyzoom, vertical write zoom, 2-96

delete
a frame definition, in IC–>PCI memory, **icp_delete_frame**, 2-27
all frame definitions, in IC–PCI memory, **icp_delete_all_frames**, 2-26
all ROIs in an IC–PCI frame, **icp_delete_all_frame_rois**, 2-26
ROI definition, in IC–PCI frame, **icp_delete_roi**, 2-27
sequence frame, in IC–PCI memory, **icp_delete_seq_frame**, 2-28

Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com

destination address, bus master
icp_get_bm_dst_addr, 2-38
icp_put_bm_dst_addr, 2-51

dimensions, for IC-PCI frame, **icp_set_cur_dim**, 2-69

disable, system colors, **icp_system_colors_off**, (DOS only), 2-79

display
define area for display utility, **icp_display_area**, 2-28
define area inside a ROI for display utility, **icp_display_roi_area**, 2-30
define frame for display utility, **icp_display_frame**, 2-29
define ROI for display utility, **icp_display_roi**, 2-30
IC-PCI frame to VGA, **icp_bitblt**, (DOS only), 2-5
set up frame to VGA transfer, **icp_bitblt_start**, (DOS only), 2-6
stop transfer to VGA, **icp_bitblt_end**, (DOS only), 2-6

displaying
IC–PCI registers, **icp_dregs**, 2-31
memory IC-PCI, **icp_dump_mem**, 2-31
memory ROI IC-PCI, **icp_dump_roi_mem**, 2-32

dump
IC-PCI memory to display, **icp—dump_mem**. *See* displaying
IC-PCI memory ROI to display, **icp_dump_roi_mem**. *See* displaying

DWORD, clearing
horizontal line, **icp_chdwline**, 2-15
vertical line, **icp_cvdwline**, 2-25

DWORD, reading, horizontal line, **icp_rhdwline**, 2-60

DWORD, writing
horizontal line, **icp_whdwline**, 2-88
vertical line, **icp_wvdwline**, 2-94

E

emulation mode, IC-PCI, B–4

enable
bus master enable flag, **icp_bm_mode**, 2-11
system colors, **icp_system_colors_on**, (DOS only), 2-79
trigger input from AM, **icp_amtrig_enable**, 2-4

exist, test for frame definition, **icp_frame_exist**, 2-35

F

field
camera field status, **icp_field_status**, 2-33
start acquisition, **icp—start_field**, 2-74

FIFO
contains 4 DWORDS, **icp_bm_fifo_4P**, 2-8
empty, **icp_bm_fifo_empty**, 2-8
full, **icp_bm_fifo_full**, 2-9
reset or clear, **icp_fifo_reset**, 2-34

first field
bus master starting address
icp_get_bm_firstf_start, 2-39
icp_put_bm_firstf_start, 2-52
IC-PCI, B–4

flip, icp_set_xform, transform during bus master transfer, 2-71

frame, 1-2
attributes, **icp_frame_attrib**, 2-34
color, 1-4
create frame buffer, in IC-PCI memory, **icp_create_frame**, 2-21
create sequence frame, in IC-PCI memory, **icp_create_seq_frame**, 2-23
define frame for display utility, **icp_display_frame**, 2-29
delete a frame definition, in IC–PCI memory, **icp_delete_frame**, 2-27
delete all frame definitions, in IC–PCI memory, **icp_delete_all_frames**, 2-26
delete sequence frame, in IC–>PCI memory, **icp_delete_seq_frame**, 2-28
set interlace mode, **icp_set_frame_ilace**, 2-69
test for definition or existence, **icp_frame_exist**, 2-35
transfer to VGA display, **icp_bitblt**, (DOS only), 2-5
wait for next vertical blank, **icp_wait_vb**, 2-86

frame count, IC-PCI, B–4

frame ID, IC-PCI, B–4

frame size
IC-PCI, B–4
return camera frame dimensions, **icp_get_acq_dim**, 2-36

freeze, acquisition to IC–PCI frame, **icp_freeze**, 2-35

G

get, pointer to AM, **icp_get_slot0**, 2-41

grab, images, **icp_grab**, 2-41

H

hardware
overview, 1-1
revision, **icp_get_hw_revision**, 2-40

horizontal line, clearing
bytes, **icp_chbline**, 2-15

Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com

DWORD, **icp_chdwline**, 2-15
WORD, **icp_chwline**, 2-16

horizontal line, reading
byte, **icp_rhbline**, 2-59
DWORD, **icp_rhdwline**, 2-60
WORD, **icp_rhwline**, 2-60

horizontal line, writing
byte, **icp_whbline**, 2-87
DWORD, **icp_whdwline**, 2-88
WORD, **icp_whwline**, 2-89

horizontal size, bus master AOI
icp_get_bm_aoix, 2-38
icp_put_bm_aoix, 2-50

host frame
large format acquire
icp_large_format_snap, 2-45
triggered, **icp_large_format_trig_snap**, 2-46
large–format frame acquire, **icp_host_large_format_acquire**, 2-42
play a VCR sequence, **icp_play_VCR**, 2-82
start ping–pong acquire and transfer, **icp_start_ping_pong**, 2-74
start VCR area recording, **icp_start_VCR_record_area**, 2-76
start VCR recording, **icp_start_VCR_record**, 2-75
stop ping–pong acquire and transfer, **icp_stop_ping_pong**, 2-77
stop VCR recording, **icp_stop_VCR_record**, 2-77

host memory, lock regions for bus master transfer, **icp_bm_lock**, 2-10

I

icp_acq_addr_status, return acquisition memory block, 2-2

icp_acq_fstart, starting field, return status, 2-3

icp_acq_in_process, test for acquisition in progress, 2-3

icp_acq_pending, test for acquisition command pending, 2-4

icp_acqbits, acquisition command bits, set or return, 2-2

icp_amtrig_enable, enable trigger input, 2-4

icp_bitblt, transfer frame to VGA, (DOS only), 2-5

icp_bitblt_end, stop transfer to VGA display, (DOS only), 2-6

icp_bitblt_start, set up frame to VGA transfer, (DOS only), 2-6

icp_bm_count, set or return, bus master transfer count, 2-7

icp_bm_done, bus master transfer done, 2-8

icp_bm_fifo_4P, FIFO contains 4 DWORDS, 2-8

icp_bm_fifo_empty, FIFO empty, 2-8

icp_bm_fifo_full, FIFO full, 2-9

icp_bm_ilace, return bus master, interlace status, 2-9

icp_bm_lock, lock memory regions, for bus–master transfers, 2-10

icp_bm_mode, bus master enable flag, 2-11

icp_bm_read, read into locked region, 2-12

icp_bm_read_subregion, read into subregion of locked region, 2-12

icp_bm_unlock, unlock regions for bus master, 2-13

icp_bus_request, request PCI bus, 2-14

icp_chbline, clear horizontal line, to byte values, 2-15

icp_chdwline, clear horizontal line, to DWORD values, 2-15

icp_chwline, clear horizontal line, to WORD values, 2-16

icp_clr_area, clear area of frame memory, 2-17

icp_clr_frame, clear the selected image frame, 2-18

icp_clr_roi, clear a ROI, 2-18

icp_cp_area, copy area in frame memory, 2-19

icp_cp_roi, copy ROI in IC-PCI frame, 2-20

icp_create_frame, create frame buffer, in IC-PCI memory, 2-21

icp_create_roi, create ROI in IC-PCI frame, 2-22

icp_create_seq_frame, create sequence frame buffer, in IC-PCI memory, 2-23

icp_cvbline, clear vertical line, to byte values, 2-24

icp_cvdwline, clear vertical line, to DWORD values, 2-25

icp_cvwline, clear vertical line, to WORD values, 2-25

icp_delete_all_frame_rois, delete all ROIs in an IC–PCI frame, 2-26

icp_delete_all_frames, delete all frame definitions, in IC–PCI memory, 2-26

icp_delete_frame, delete a frame definition, in IC–PCI memory, 2-27

icp_delete_roi, delete a ROI definition, in IC–PCI frame, 2-27

icp_delete_seq_frame, delete sequence frame, in IC–PCI memory, 2-28

Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com

icp_display_area, define area for display utility, 2-28

icp_display_frame, define a frame for display utility, 2-29

icp_display_roi, define ROI for display utility, 2-30

icp_display_roi_area, define area inside a ROI, for display utility, 2-30

icp_dregs, display/dump IC-PCI registers contents, 2-31

icp_dump_mem, display an area of IC-PCI memory, 2-31

icp_dump_roi_mem, display content of ROI in IC-PCI memory, 2-32

icp_fcount, return acquisition frame count, 2-32

icp_field_status, camera field status, 2-33

icp_fifo_reset, reset/clear FIFO, 2-34

icp_frame_attrib, return frame attributes, IC-PCI frame, 2-34

icp_frame_exist, test for definition of a frame, 2-35

icp_freeze, stop continuous acquisition, to IC–PCI frame, 2-35

icp_get_acq_dim, return camera frame dimensions, 2-36

icp_get_acq_start_addr, return acquisition start address, in IC-PCI memory, 2-36

icp_get_active_ubm_frame, get pointer to active UBM frame, 2-37

icp_get_bm_aoix, return bus master AOI horizontal size, 2-38

icp_get_bm_dst_addr, return bus master destination address, 2-38

icp_get_bm_firstf_start, return bus master first field starting address, 2-39

icp_get_bm_secondf_start, return bus master second field starting address, 2-39

icp_get_hw_revision, return hardware revision ID, 2-40

icp_get_idle_ubm_frame, get pointer to idle UBM frame, 2-40

icp_get_slot0, return pointer to AM, 2-41

icp_grab, acquire continuously, 2-41

icp_host_large_format_acquire, acquire–large format frames into host frame, 2-42

icp_image_pitch, bus master, image pitch, 2-44

icp_iregs, initialize IC-PCI registers, 2-44

icp_large_format_snap, acquire single large–format frame, to host frame, 2-45

icp_large_format_trig_snap, triggered large–format acquire, to host frame, 2-46

icp_latency_timer, bus master latency timer, 2-46

icp_line_reverse_read, read area in reverse line order, 2-47

icp_mailbox_read, read from PCI mailbox, 2-48

icp_mailbox_write, write to PCI mailbox, 2-48

icp_mem_size, return IC-PCI memory size, 2-49

icp_pixsz, set or return current pixel size, 2-49

icp_put_acq_start_addr, acquisition start address, in IC-PCI memory, 2-50

icp_put_bm_aoix, return bus master AOI horizontal size, 2-50

icp_put_bm_dst_addr, return bus master destination address, 2-51

icp_put_bm_firstf_start, bus master first field starting address, 2-52

icp_put_bm_secondf_start, bus master second field starting address, 2-52

icp_read_area, read area in IC-PCI frame to buffer, 2-53

icp_read_dest_pitch_area, read area to host adding destination pitch, 2-54

icp_read_harea, read huge area form IC-PCI frame to buffer, (Windows 3.1 only), 2-55

icp_read_roi_area, read area from ROI to buffer, 2-57

icp_read_roi_harea, read huge area from ROI to buffer, 2-58

icp_rhbline, read horizontal line, of byte values, 2-59

icp_rhdwline, read horizontal line, of DWORD values, 2-60

icp_rhwline, read horizontal line, of WORD values, 2-60

icp_roi_attrib, return ROI attributes, 2-61

icp_roi_rpix, read a pixel from a ROI, 2-62

icp_roi_set_lineio, set up for line read write, 2-63

icp_roi_wpix, write a pixel to a ROI, 2-64

icp_rpix, read pixel in IC-PCI memory frame, 2-64

icp_rvbline, read vertical line, of byte values, 2-65

icp_rvdwline, read vertical line, of WORD values, 2-65

icp_rvwline, read vertical line, of WORD values, 2-66

icp_scanmd_status, return camera scan mode status, 2-67

icp_seq_snap, snap sequence, 2-67

Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com

icp_seq_snap_async, snap sequence asynchronously, 2-68

icp_set_cur_dim, set current dimensions, IC-PCI frame, 2-69

icp_set_frame_ilace, set the interlace mode for a frame, 2-69

icp_set_lineio, set up for line read write, 2-70

icp_set_xform, Set up for bus master transform, 2-71

icp_snap, acquire single image, 2-71

icp_snap_async, acquire single image asynchronously, 2-72

icp_soft_reset, set or return soft reset, 2-73

icp_start_field, starting field, set or return, 2-74

icp_start_ping_pong, start ping pong acquisition and transfer to host frame, 2-74

icp_start_VCR_record, start a VCR recording sequence, into host frame, 2-75

icp_start_VCR_record_area, start a VCR recording an area sequence, to host frame, 2-76

icp_stop_ping_pong, stop ping–pong acquire and transfer to host, to host frame, 2-77

icp_stop_VCR_record, stop a VCR recording sequence, 2-77

icp_sync_mode, sync mode flag, set or return, 2-78

icp_system_colors_off, disable system colors, (DOS only), 2-79

icp_system_colors_on, enable system colors, (DOS only), 2-79

icp_system_colors_restore, restore system colors, (DOS only), 2-79

icp_trig_snap, triggered snap, 2-80

icp_vblank_status, return camera vertical blank status, 2-81

icp_VCR_locate, locate a pixel in a VCR sequence, host frame, 2-81

icp_VCR_play, display recorded sequence, 2-82

icp_wait_acq, wait for current acquisition to complete, 2-83

icp_wait_bmdone, wait for bus master transfer to complete, 2-83

icp_wait_notvb, wait for not in vertical blank, 2-84

icp_wait_start_vb, wait for start of camera vertical blank, 2-85

icp_wait_trig, wait for end of triggered acquire, 2-85

icp_wait_vb, wait for next vertical blank, in IC-PCI frame, 2-86

icp_whbline, write horizontal line, of byte values, 2-87

icp_whdwline, write horizontal line, of DWORD values, 2-88

icp_whwline, write horizontal line, of WORD values, 2-89

icp_wpix, write a single pixel, in IC-PCI frame, 2-89

icp_write_area, write IC-PCI frame area from an array, 2-90

icp_write_harea, write huge area from array to IC-PCI frame, 2-91

icp_write_roi_area, write area from an array into IC-PCI ROI, 2-92

icp_write_roi_harea, write huge area from array to IC-PCI ROI, 2-93

icp_wvbline, write vertical line, of byte values, 2-94

icp_wvdwline, write vertical line, of DWORD values, 2-94

icp_wvwline, write vertical line, of WORD values, 2-95

icp_wxzoom, set or return horizontal write zoom, 2-95

icp_wyzoom, set or return vertical write zoom, 2-96

ICVL compatible mode, IC-PCI, B–4

idle, UBM frame, **icp_get_idle_ubm_frame**, 2-40

image memory, 1-2

image read, 1-11

image save, 1-11

information, pointer to AM, **icp_get_slot0**, 2-41

initialization functions, 1-8

initializing, IC-PCI registers, **icp_iregs**, 2-44

interface, module, 1-3

interlace
- bus master status, **icp_bm_ilace**, 2-9
- set mode for a frame, **icp_set_ilace_mode**, 2-69

interlaced, camera scan mode status, **icp_scanmd_status**, 2-67

interrupt events, 1-7

interrupt line, IRQ0–15, IC-PCI, B–3

L

large–format
- acquire, **icp_host_large_format_acquire**, 2-42
- acquire single frame to host, **icp_large_format_snap**, 2-45
- triggered acquire to host frame, **icp_large_format_trig_snap**, 2-46

Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com

latency, timer, IC-PCI, B–3

latency timer, bus master, **icp_latency_timer**, 2-46

linear memory, 1-2

locate pixel, in host frame, VCR recording, **icp_VCR_locate**, 2-81

lock, frame memory for bus master transfers, **icp_bm_lock**, 2-10

M

mailbox
- initial values, IC-PCI, B–3
- read PCI mailbox, **icp_mailbox_read**, 2-48
- write to PCI mailbox, **icp_mailbox_write**, 2-48

memory
- acquisition block, **icp_acq_addr_status**, 2-2
- acquisition boundaries, 1-2
- acquisition start address, in IC-PCI memory, **icp_put_acq_start_addr**, 2-50
- clear area in frame, **icp_clr_area**, 2-17
- clear frame, **icp_clr_frame**, 2-18
- copy area in IC-PCI frame, **icp_cp_area**, 2-19
- create frame in IC-PCI memory, **icp_create_frame**, 2-21
- create ROI in IC-PCI frame, **icp_create_roi**, 2-22
- create sequence frame, in IC-PCI memory, **icp_create_seq_frame**, 2-23
- delete a frame definition, in IC–>PCI memory, **icp_delete_frame**, 2-27
- delete all frame definitions, in IC–PCI memory, **icp_delete_all_frames**, 2-26
- delete all ROIs in an IC–PCI frame, **icp_delete_all_frame_rois**, 2-26
- delete ROI definition, in IC–PCI frame, **icp_delete_roi**, 2-27
- delete sequence frame, in IC–PCI memory, **icp_delete_seq_frame**, 2-28
- display IC-PCI memory content, **icp_dump_mem**, 2-31
- display IC-PCI memory ROI content, **icp_dump_roi_mem**, 2-32
- image, 1-2
- read pixel from IC-PCI frame, **icp_rpix**, 2-64
- return acquisition start address, in IC-PCI memory, **icp_get_acq_start_addr**, 2-36
- set up for line read write
  - **icp_roi_set_lineio**, 2-63
  - **icp_set_lineio**, 2-70
- size, **icp_mem_size**, 2-49
- test for IC-PCI frame defined, **icp_frame_exist**, 2-35
- write area in IC-PCI frame, from an array, **icp_write_area**, 2-90
- write area in IC-PCI ROI, from an array, **icp_write_roi_area**, 2-92
- write huge area in IC-PCI frame, from an array, **icp_write_harea**, 2-91
- write huge area in IC-PCI ROI, from an array, **icp_write_roi_harea**, 2-93
- write single pixel in IC-PCI frame, **icp_wpix**, 2-89

memory sizes, 1-2

mirror, icp_set_xform, transform during bus master transfer, 2-71

mode, bus master enable flag, **icp_bm_mode**, 2-11

module interface, 1-3

module structure functions, 1-13

movie. *See* ,
itx_start_VCR_record, itx_VCR_play

N

non-interlaced, camera scan mode status, **icp_scanmd_status**, 2-67

P

pending, acquisition command, **icp_acq_pending**, 2-4

ping–pong
- start acquisition and transfer to host frame, **icp_start_ping_pong**, 2-74
- stop acquire and transfer, **icp_stop_ping_pong**, to host frame, 2-77

pitch
- bus master image pitch, **icp_image_pitch**, 2-44
- read area to host adding pitch, **icp_read_dest_pitch_area**, 2-54

pixel
- locate in VCR host frame, **icp_VCR_locate**, 2-81
- read from IC–PCI memory frame, **icp_rpix**, 2-64
- read from ROI, **icp_roi_rpix**, 2-62
- size, **icp_pixsz**, 2-49
- write single in IC-PCI frame, **icp_wpix**, 2-89
- write to ROI, **icp_roi_wpix**, 2-64

pixel size
- return IC-PCI frame attributes, **icp_frame_attrib**, 2-34
- return ROI attributes, **icp_roi_attrib**, 2-61

play, display VCR sequence, **icp_VCR_play**, 2-82

pointer
- to active UBM frame, **icp_get_active_ubm_frame**, 2-37

Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com

to idle UBM frame, **icp_get_idle_ubm_frame**, 2-40

R

read
area in IC-PCI frame to buffer, **icp_read_area**, 2-53
area in ROI to buffer, **icp_read_roi_area**, 2-57
area to host adding destination pitch, **icp_read_dest_pitch_area**, 2-54
huge area from IC-PCI frame to buffer, **icp_read_area**, (Windows 3.1 only), 2-55
huge area in ROI to buffer, **icp_read_roi_harea**, 2-58
IC–PCI memory into locked host region, **icp_bm_read**, 2-12
IC–PCI memory into subregion oflocked host region, **icp_bm_read_subregion**, 2-12
image, 1-11
lines in reverse order, **icp_line_reverse_read**, 2-47
PCI mailbox, **icp_mailbox_read**, 2-48
pixel from ROI, **icp_roi_rpix**, 2-62
pixel in IC-PCI memory frame, **icp_rpix**, 2-64

read and write functions, 1-10

reading, bits and registers, A-1
See also system–level software **read_reg**; system–level software **read_reg read_reg32**

reading horizontal line
byte, **icp_rhbline**, 2-59
DWORD, **icp_rhdwline**, 2-60
WORD, **icp_rhwline**, 2-60

reading vertical line
byte, **icp_rbvline**, 2-65
WORD, **icp_rvwline**, 2-65, 2-66

recording, VCR sequence in host frame, **icp_start_VCR_record**, 2-75

register, read/write. *See* system–level software, **read_reg write_reg**

registers
display IC-PCI registers, **icp_dregs**, 2-31
initializing IC-PCI registers, **icp_iregs**, 2-44

request, PCI bus, **icp_bus_request**, 2-14

reset
FIFO, **icp_fifo_reset**, 2-34
soft, **icp_soft_reset**, 2-73

restore
image, 1-11
system colors, **icp_system_colors_restore**, (DOS only), 2-79

return
attributes for IC-PCI frame, **icp_frame_attrib**, 2-34
attributes for ROI, **icp_roi_attrib**, 2-61
camera frame dimensions, **icp_get_acq_dim**, 2-36
pointer to AM, **icp_get_slot0**, 2-41

return values, 1-6

reverse, order of lines, **icp_line_reverse_read**, 2-47

ROI
attributes, **icp_roi_attrib**, 2-61
clear, **icp_clr_roi**, 2-18
copy, **icp_cp_roi**, 2-20
create in IC-PCI frame, **icp_create_roi**, 2-22
define area inside ROI for display utility, **icp_display_roi_area**, 2-30
define ROI for display utility, **icp_display_roi**, 2-30
delete all ROIs in an IC–PCI frame, 2-26
delete ROI definition, in IC–PCI frame, **icp_delete_roi**, 2-27
read area to buffer, **icp_read_roi_area**, 2-57
read huge area to buffer, **icp_read_roi_harea**, 2-58
read pixel, **icp_roi_rpix**, 2-62
set up for line read/write, **icp_roi_set_lineio**, 2-63
short definition, iv
versus AOI. *See* system–level software manual
write a pixel, **icp_roi_wpix**, 2-64

rotate180, icp_set_xform, transform during bus master transfer, 2-71

S

save, image, 1-11

scan mode, camera, **icp_scanmd_status**, 2-67

second field, bus master starting address
icp_get_bm_secondf_start, 2-39
icp_put_bm_secondf_start, 2-52

sequence
snap, **icp_seq_snap**, 2-67
snap asynchronous, **icp_seq_snap_async**, 2-68

set frame dimensions, **icp_set_cur_dim**, 2-69

set up
for line read write
icp_roi_set_lineio, 2-63
icp_set_lineio, 2-70
frame to VGA display, **icp_bitblt_start**, 2-6

size
bus master AOI horizontal size
icp_get_bm_aoix, 2-38

Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com

icp_put_bm_aoix, 2-50
IC-PCI memory option, **icp_mem_size**, 2-49
pixel current size, **icp_pixsz**, 2-49
return IC-PCI frame attributes, **icp_frame_attrib**, 2-34
return ROI attributes, **icp_roi_attrib**, 2-61

snap
acquire single image, **icp_snap**, 2-71
acquire single image asynchronously, **icp_snap_async**, 2-72
large–format, **icp_large_format_snap**, 2-45
sequence, **icp_seq_snap**, 2-67
sequence asynchronous, **icp_seq_snap_async**, 2-68
triggered, **icp_trig_snap**, 2-80

soft reset, **icp_soft_reset**, 2-73

start
ping–pong acquisition, **icp_start_ping_pong**, 2-74
VCR area recording to host frame, **icp_start_VCR_record_area**, 2-76
VCR record, **icp_start_VCR_record**, into host frame, 2-75

start address, in IC-PCI memory
icp_get_acq_start_addr, 2-36
icp_put_acq_start_addr, 2-50

start field
IC-PCI, B–4
icp_start_field, set or return, 2-74

starting field, return status, **icp_acq_fstart**, 2-3

status
acquisition command pending, **icp_acq_pending**, 2-4
acquisition in progress, **icp_acq_in_progress**, 2-3
bus master enable flag, **icp_bm_mode**, 2-11
bus master interlace, **icp_bm_ilace**, 2-9
bus master transfer done, **icp_bm_done**, 2-8
camera field, **icp_field_status**, 2-33
camera scan mode, **icp_scanmd_status**, 2-67
camera vertical blank, **icp_vblank_status**, 2-81
FIFO empty, **icp_bm_fifo_empty**, 2-8
FIFO full, **icp_bm_fifo_full**, 2-9
starting field, **icp_acq_fstart**, 2-3

stop
continuous acquisition to IC–PCI frame, **icp_freeze**, 2-35
frame to VGA transfer, **icp_bitblt_end**, (DOS only), 2-6
ping–pong acquire and transfer, **icp_stop_ping_pong**, 2-77
VCR recording, **icp_stop_VCR_record**, 2-77

sub–sample, write zoom
icp_wxzoom, horizontal write zoom, 2-95
icp_wyzoom, vertical write zoom, 2-96

sync mode flag, set or return, **icp_sync_mode**, 2-78

system colors
disable, **icp_system_colors_off**, (DOS only), 2-79
enable, **icp_system_colors_on**, (DOS only), 2-79
restore, **icp_system_colors_restore**, (DOS only), 2-79

T

test
acquisition command pending, **icp_acq_pending**, 2-4
for frame definition, **icp_frame_exist**, 2-35

timer, bus master latency, **icp_latency_timer**, 2-46

transfer count, bus master, set or return, **icp_bm_count**, 2-7

transfer done, bus master, **icp_bm_done**, 2-8

trigger, enable input from AM, **icp_amtrig_enable**, 2-4

triggered
snap, **icp_trig_snap**, 2-80
wait for end of acquire, **icp_wait_trig**, 2-85

U

UBM, unattended bus mastered frame
icp_get_active_ubm_frame, 2-37
icp_get_idle_ubm_frame, 2-40

unlock, regions for bus master, **icp_bm_unlock**, 2-13

V

VCR
display recorded sequence, **icp_VCR_play**, 2-82
start recording, **icp_start_VCR_record**, 2-75
start recording an area to host frame, **icp_start_VCR_record_area**, 2-76
stop recording, **icp_stop_VCR_record**, 2-77

vertical blank
camera status, **icp_vblank_status**, 2-81
wait, **icp_wait_vb**, 2-86
wait for start of camera vertical blank, **icp_wait_start_vb**, 2-85
wait until not in vertical blank, **icp_wait_notvb**, 2-84

vertical line, clearing
byte, **icp_cvbline**, 2-24
DWORD, **icp_cvdwline**, 2-25
WORD, **icp_cvwline**, 2-25

vertical line, reading
byte, **icp_rvbline**, 2-65

Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com

WORD, **icp_rvwline**, 2-65, 2-66
vertical line, writing
byte, **icp_wvbline**, 2-94
DWORD, **icp_wvdwline**, 2-94
WORD, **icp_wvwline**, 2-95
VGA
set up frame to VGA display, **icp_bitblt_start**, (DOS only), 2-6
stop frame to display transfers, **icp_bitblt_end**, (DOS only), 2-6
transfer frame, **icp_bitblt**, (DOS only), 2-5

W

wait
for bus master transfer to complete, **icp_wait_bmdone**, 2-83
for current acquisition to complete, **icp_wait_acq**, 2-83
for end of triggered acquire, **icp_wait_trig**, 2-85
for next vertical blank, IC-PCI frame, **icp_wait_vb**, 2-86
for start of next camera vertical blank, **icp_wait_start_vb**, 2-85
until not in vertical blank, **icp_wait_notvb**, 2-84
WORD, clearing
horizontal line, **icp_chwline**, 2-16
vertical line, **icp_cvwline**, 2-25
WORD, reading
horizontal line, **icp_rhwline**, 2-60
vertical line, **icp_rvwline**, 2-65, 2-66
WORD, writing
horizontal line, **icp_whwline**, 2-89
vertical line, **icp_wvwline**, 2-95
write
a single pixel in IC-PCI frame, **icp_wpix**, 2-89
area from array in IC-PCI ROI, **icp_write_roi_area**, 2-92
huge area from array to IC-PCI frame, **icp_write_harea**, 2-91
huge area from array to IC-PCI ROI, **icp_write_roi_harea**, 2-93
IC-PCI frame area from array, **icp_write_area**, 2-90
PCI mailbox, **icp_mailbox_write**, 2-48
pixel to ROI, **icp_roi_wpix**, 2-64
writing, bits and registers, A-1
See also system–level software, **write_reg**; system–level software, **write_reg read_reg32**
writing horizontal line
byte, **icp_whbline**, 2-87
DWORD, **icp_whdwline**, 2-88
WORD, **icp_whwline**, 2-89
writing vertical line
byte, **icp_wvbline**, 2-94
DWORD, **icp_wvdwline**, 2-94
WORD, **icp_wvwline**, 2-95

Z

zoom
icp_wxzoom, horizontal write zoom, 2-95
icp_wyzoom, vertical write zoom, 2-96

Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com

Artisan Technology Group is an independent supplier of quality pre-owned equipment

Gold-standard solutions

Extend the life of your critical industrial, commercial, and military systems with our superior service and support.

We buy equipment

Planning to upgrade your current equipment? Have surplus equipment taking up shelf space? We'll give it a new home.

Learn more!

Visit us at **artisantg.com** for more info on price quotes, drivers, technical specifications, manuals, and documentation.

Artisan Scientific Corporation dba Artisan Technology Group is not an affiliate, representative, or authorized distributor for any manufacturer listed herein.

We're here to make your life easier. How can we help you today?

(217) 352-9330 | sales@artisantg.com | artisantg.com

ARTISAN® TECHNOLOGY GROUP