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 *Dynamic Engineering PCI5IP* at our website: *Click HERE*

DYNAMIC ENGINEERING
150 DuBois, Suite 3
Santa Cruz, CA 95060
(831) 457-8891 **Fax** (831) 457-4793
http://www.dyneng.com
sales@dyneng.com
Est. 1988

Pci5Ip

Driver Documentation

Win32 Driver Model

Revision D
Corresponding Hardware: Revision D
10-2002-0304
Corresponding Firmware: Revision E

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

Pci5Ip
WDM Device Driver for the Pci5Ip
PCI based IP Carrier

Dynamic Engineering
150 DuBois, Suite 3
Santa Cruz, CA 95060
(831) 457-8891
FAX: (831) 457-4793

©2008 by Dynamic Engineering.

Trademarks and registered trademarks are owned by their respective manufactures.
Manual Revision D. Revised March 11, 2008.

This document contains information of proprietary interest to Dynamic Engineering. It has been supplied in confidence and the recipient, by accepting this material, agrees that the subject matter will not be copied or reproduced, in whole or in part, nor its contents revealed in any manner or to any person except to meet the purpose for which it was delivered.

Dynamic Engineering has made every effort to ensure that this manual is accurate and complete. Still, the company reserves the right to make improvements or changes in the product described in this document at any time and without notice. Furthermore, Dynamic Engineering assumes no liability arising out of the application or use of the device described herein.

The electronic equipment described herein generates, uses, and can radiate radio frequency energy. Operation of this equipment in a residential area is likely to cause radio interference, in which case the user, at his own expense, will be required to take whatever measures may be required to correct the interference.

Dynamic Engineering's products are not authorized for use as critical components in life support devices or systems without the express written approval of the president of Dynamic Engineering.

This product has been designed to operate with IP Module carriers and compatible user-provided equipment. Connection of incompatible hardware is likely to cause serious damage.

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

Table of Contents

Introduction 4

Note 4

Driver Installation 4

Windows 2000 Installation 5

Windows XP Installation 5

Driver Startup 6

IO Controls 9
IOCTL_PCI5IP_GET_INFO 9
IOCTL_PCI5IP_GET_SW_ID 9
IOCTL_PCI5IP_SET_BASE_CONFIG 9
IOCTL_PCI5IP_GET_BASE_CONFIG 10
IOCTL_PCI5IP_GET_INT_STATUS 10
IOCTL_PCI5IP_REGISTER_EVENT 10
IOCTL_PCI5IP_FORCE_INTERRUPT 10
IOCTL_PCI5IP_READ_ID_PROM 11
IOCTL_PCI5IP_RESET_ALL_IPS 11
IOCTL_PCI5IP_IDENTIFY 11
IOCTL_PCI5IP_REINIT_IPS 11

WARRANTY AND REPAIR 12

Service Policy 12
Out of Warranty Repairs 12

For Service Contact: 12

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

Introduction

The Pci5Ip driver is a Win32 driver model (WDM) device driver for the Pci5Ip Industry Pack (IP) carrier from Dynamic Engineering. Each Pci5Ip board can hold up to five IP modules. When the Pci5Ip is recognized by the PCI bus configuration utility it will load this driver and enumerate the Pci5Ip's IP bus by reading the ID proms of installed IPs. If the IP device has been previously installed, its device driver will be loaded and a Device Object will be created for each installed IP. A separate handle to the Pci5Ip and to each IP device can be obtained using CreateFile() calls (see below). IO Control calls (IOCTLs) are used to configure the Pci5Ip and read status, although this is not necessary to operate the IP modules. The Pci5Ip is responsible for reading its user switch setting, operating the onboard LEDs, and a few other operations.

The Pci5Ip's main function is to act as a PCI<->IP Bus bridge device providing resources for the installed IP modules, which independently operate through their own file handles. See the appropriate IP driver documentation for information on the capabilities of a particular IP module.

Note

This documentation will provide information about all calls made to the driver, and how the driver interacts with the device for each of these calls. For more detailed information on the hardware implementation, refer to the Pci5Ip device user manual.

Driver Installation

There are several files provided in each driver package. These files include Pci5Ip.sys, IpCarrier.inf, DDPci5Ip.h, Pci5IpGUID.h, Pci5IpTest.exe, and Pci5IpTest source files.

DDPci5Ip.h is a C header file that defines the Application Program Interface (API) to the driver. Pci5IpGUID.h is a C header file that defines the device interface identifier for the Pci5Ip. These files are required at compile time by any application that wishes to interface with the Pci5Ip driver, but are not needed for driver installation.

Pci5IpTest.exe is a sample Win32 console application that makes calls into the Pci5Ip driver to test the driver calls without actually writing any application code. It is not required during the driver installation. Open a command prompt console window and type ***Pci5IpTest –d0 -?*** to display a list of commands (the Pci5IpTest.exe file must be in the directory that the window is referencing). The commands are all of the form ***Pci5IpTest –dn –im*** where ***n*** and ***m*** are the device number and driver ioctl number respectively.

This application is only intended to test the proper functioning of the driver calls and should not be used for normal operation since it will result in diminished performance.

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

Windows 2000 Installation

Copy IpCarrier.inf and Pci5Ip.sys to a floppy disk, or CD if preferred.

With the Pci5Ip installed, power-on the PCI host computer and wait for the ***Found New Hardware Wizard*** dialogue window to appear.

- Select ***Next***.
- Select ***Search for a suitable driver for my device.***
- Select ***Next***.
- Insert the disk prepared above in the desired drive.
- Select the appropriate drive e.g. ***Floppy disk drives***.
- Select ***Next***.
- The wizard should find the ipcarrier.inf file.
- Select ***Next***.
- Select ***Finish*** to close the ***Found New Hardware Wizard***.

Windows XP Installation

Copy IpCarrier.inf and Pci5Ip.sys to a floppy disk, or CD if preferred.

With the Pci5Ip installed, power-on the PCI host computer and wait for the ***Found New Hardware Wizard*** dialogue window to appear.

- Insert the disk prepared above in the desired drive.
- Select ***No when asked to connect to Windows Update***.
- Select ***Next***.
- Select ***Install the software automatically.***
- Select ***Next***.
- Select ***Finish*** to close the ***Found New Hardware Wizard***.

At this point, if there are IP modules installed on the Pci5Ip, the wizard will reactivate. The process for installing the IP drivers is similar, and details can be found in the appropriate IP driver manual.

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

Driver Startup

Once the driver has been installed it will start automatically when the system recognizes the hardware.

A handle can be opened to a specific board by using the CreateFile() function call and passing in the device name obtained from the system. The interface to the device is identified using a globally unique identifier (GUID), which is defined in Pci5IpGUID.h.

Below is example code for opening a handle for device devNum.

```
// The maximum length of the device name for a given instance of an interface
#define MAX_DEVICE_NAME 256
// Handle to the device object
HANDLE                              hPci5Ip = INVALID_HANDLE_VALUE;
// Index for device interface search loop
Int                                 i;
// Return status from command
LONG                                status;
// Handle to device interface information structure
HDEVINFO                            hDeviceInfo;
// The actual symbolic link name to use in the createfile
CHAR                                deviceName[MAX_DEVICE_NAME];
// Size of buffer required to get the symbolic link name
DWORD                               requiredSize;
// Interface data structures for this device
SP_DEVICE_INTERFACE_DATA            interfaceData;
PSP_DEVICE_INTERFACE_DETAIL_DATA pDeviceDetail;

hDeviceInfo = SetupDiGetClassDevs(
               (LPGUID)&GUID_DEVINTERFACE_PCI5IP,
                       NULL,
                       NULL,
                       DIGCF_PRESENT | DIGCF_DEVICEINTERFACE);

if(hDeviceInfo == INVALID_HANDLE_VALUE)
{
   printf("**Error: couldn't get class info, (%d)\n", GetLastError());
   exit(-1);
}

interfaceData.cbSize = sizeof(interfaceData);

for(i = 0; i <= devNum; i++)
{// Find the interface for device devNum
   if(!SetupDiEnumDeviceInterfaces(hDeviceInfo,
                                   NULL,
                           (LPGUID)&GUID_DEVINTERFACE_PCI5IP,
                                   i,
                                   &interfaceData))
   {
      status = GetLastError();
      if(status == ERROR_NO_MORE_ITEMS)
      {
         printf("**Error: couldn't find device(no more items), (%d)\n", i);
         SetupDiDestroyDeviceInfoList(hDeviceInfo);
         exit(-1);
      }
```

```
      else
      {
         printf("**Error: couldn't enum device, (%d)\n", status);
         SetupDiDestroyDeviceInfoList(hDeviceInfo);
         exit(-1);
      }
   }
}

// Get the details data to obtain the symbolic link name
if(!SetupDiGetDeviceInterfaceDetail(hDeviceInfo,
                                    &interfaceData,
                                    NULL,
                                    0,
                                    &requiredSize,
                                    NULL))
{
   if(GetLastError() != ERROR_INSUFFICIENT_BUFFER)
   {
      printf("**Error: couldn't get interface detail, (%d)\n",
             GetLastError());

      SetupDiDestroyDeviceInfoList(hDeviceInfo);
      exit(-1);
   }
}

// Allocate a buffer to get detail
pDeviceDetail = (PSP_DEVICE_INTERFACE_DETAIL_DATA)malloc(requiredSize);
if(pDeviceDetail == NULL)
{
   printf("**Error: couldn't allocate interface detail\n");
   SetupDiDestroyDeviceInfoList(hDeviceInfo);
   exit(-1);
}

pDeviceDetail->cbSize = sizeof(SP_DEVICE_INTERFACE_DETAIL_DATA);

// Get the detail info
if(!SetupDiGetDeviceInterfaceDetail(hDeviceInfo,
                                    &interfaceData,
                                    pDeviceDetail,
                                    requiredSize,
                                    NULL,
                                    NULL))
{
   printf("**Error: couldn't get interface detail(2), (%d)\n",
          GetLastError());

   SetupDiDestroyDeviceInfoList(hDeviceInfo);
   free(pDeviceDetail);
   exit(-1);
}

// Save the name
lstrcpyn(deviceName, pDeviceDetail->DevicePath, MAX_DEVICE_NAME);

// Cleanup search
free(pDeviceDetail);
SetupDiDestroyDeviceInfoList(hDeviceInfo);
```


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

```
// Create the handle to the device
hPci5Ip = CreateFile(deviceName,
                     GENERIC_READ    | GENERIC_WRITE,
                     FILE_SHARE_READ | FILE_SHARE_WRITE,
                     NULL,
                     OPEN_EXISTING,
                     NULL,
                     NULL);

if(hPci5Ip == INVALID_HANDLE_VALUE)
{
   printf("**Error: couldn't open %s, (%d)\n", deviceName,
           GetLastError());

   exit(-1);
}
```


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

IO Controls

The driver uses IO Control calls (IOCTLs) to configure the device. IOCTLs refer to a single Device Object, which controls a single board. IOCTLs are called using the Win32 function DeviceIoControl() (see below), and passing in the handle to the device opened with CreateFile(). IOCTLs generally have input parameters, output parameters, or both. Often a custom structure is used.

```
BOOL DeviceIoControl(
  HANDLE       hDevice,          // Handle opened with CreateFile()
  DWORD        dwIoControlCode,  // Control code defined in DDPci5Ip.h
  LPVOID       lpInBuffer,       // Pointer to input parameter
  DWORD        nInBufferSize,    // Size of input parameter
  LPVOID       lpOutBuffer,      // Pointer to output parameter
  DWORD        nOutBufferSize,   // Size of output parameter
  LPDWORD      lpBytesReturned,  // Pointer to return length parameter
  LPOVERLAPPED lpOverlapped,     // Optional pointer to overlapped structure
);                               //   used for asynchronous I/O
```

IOCTL_PCI5IP_GET_INFO

Function: Returns the current driver version and instance number.
Input: None
Output: DRIVER_CARRIER_DEVICE_INFO structure
Notes: This call does not access the hardware, only driver parameters. See DDPci5Ip.h for the definition of DRIVER_CARRIER_DEVICE_INFO.

IOCTL_PCI5IP_GET_SW_ID

Function: Reads the eight-position onboard dipswitch.
Input: None
Output: Switch value (unsigned long integer)
Notes: The switch can be used for any purpose that the user wishes. It can uniquely identify the boards installed in a chassis, or be used to distinguish configuration classes to the user's application software.

IOCTL_PCI5IP_SET_BASE_CONFIG

Function: Writes to the base configuration register on the Pci5Ip.
Input: Register configuration value (unsigned long integer)
Output: None
Notes: This call is used to control the eight LEDs on the board, the interrupt enables and the bus error interrupt clear. The bus error and master interrupt enables default to TRUE when the driver initializes, however they can be disabled using this call. Be aware that if this is done it will disable the interrupts of all of the IPs installed on the carrier. If the bus error interrupt clear is written as a one to clear the latched bus error status, it will be automatically cleared and does not need to be re-written as a zero.

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

IOCTL_PCI5IP_GET_BASE_CONFIG

Function: Returns the configuration of the base control register.
Input: None
Output: Register configuration value (unsigned long integer)
Notes: This call is used mainly for testing.

IOCTL_PCI5IP_GET_INT_STATUS

Function: Returns the IP interrupt status.
Input: None
Output: Interrupt status register value (unsigned long integer)
Notes: Returns the masked and unmasked interrupt status for all the IP slots as well as the bus error interrupt and combined status. See DDPci5Ip.h for details on all the status bits.

IOCTL_PCI5IP_REGISTER_EVENT

Function: Registers an event to be signaled when an interrupt occurs.
Input: Handle to Event object
Output: None
Notes: The caller creates an event with CreateEvent() and supplies the handle returned from that call as the input to this IOCTL. The driver then obtains a system pointer to the event and signals the event when an interrupt is serviced. The user interrupt service routine waits on this event, allowing it to respond to the interrupt. Please note that the Pci5Ip only handles the bus error interrupt and its own force interrupt, all the IP interrupts are handled by the individual IP drivers and will not cause the event to be signaled.

IOCTL_PCI5IP_FORCE_INTERRUPT

Function: Causes a system interrupt to occur.
Input: None
Output: None
Notes: Causes an interrupt to be asserted on the PCI bus as long as the master interrupt has not been disabled. This IOCTL is used for development, to test interrupt processing.

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

IOCTL_PCI5IP_READ_ID_PROM

Function: Returns the contents of the IP ID prom for a particular slot.
Input: Slot name (WCHAR)
Output: IP ID PROM contents (ID_DATA structure)
Notes: Returns the contents of the requested IP ID prom. The slot A..G (F is 32-bit slot B-C and G is 32-bit slot D-E) is passed into this call as a Unicode character and the ID_DATA structure is returned. This structure contains two Boolean fields that indicate if the IP prom is valid and if it is capable of 32 MHz operation. It also contains a 12-element array of UCHAR that has the ID prom contents, provided the prom was found to be valid. See DDPci5Ip.h for the structure definition.

IOCTL_PCI5IP_RESET_ALL_IPS

Function: Resets all the IP slots.
Input: None
Output: None
Notes: Resets all IP slots by setting and then clearing the BASE_RESET_ALL_IPS bit in the base configuration register. This bit cannot be controlled by the IOCTL_PCI5IP_SET_BASE_CONFIG call.

IOCTL_PCI5IP_IDENTIFY

Function: Flashes all user LEDs three times.
Input: None
Output: None
Notes: This call can be used when more than one device is installed in a chassis and it is desired to identify the physical location of a particular device.

IOCTL_PCI5IP_REINIT_IPS

Function: Remove all child devices and re-enumerate all the IPs on the carrier.
Input: None
Output: None
Notes: All handles referencing any of the IP modules on the carrier must be closed before this call is made in order for the child device object to be removed. This call should be made after the IOCTL_PCI5IP_RESET_ALL_IPS call is made in order to properly initialize the device registers and stored driver values.

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

Warranty and Repair

Dynamic Engineering warrants this product to be free from defects under normal use and service and in its original, unmodified condition, for a period of one year from the time of purchase. If the product is found to be defective within the terms of this warranty, Dynamic Engineering's sole responsibility shall be to repair, or at Dynamic Engineering's sole option to replace, the defective product.

Dynamic Engineering's warranty of and liability for defective products is limited to that set forth herein. Dynamic Engineering disclaims and excludes all other product warranties and product liability, expressed or implied, including but not limited to any implied warranties of merchandisability or fitness for a particular purpose or use, liability for negligence in manufacture or shipment of product, liability for injury to persons or property, or for any incidental or consequential damages.

Dynamic Engineering's products are not authorized for use as critical components in life support devices or systems without the express written approval of the president of Dynamic Engineering.

Service Policy

Before returning a product for repair, verify as well as possible that the driver is at fault. The driver has gone through extensive testing and in most cases it will be a "cockpit error" rather than an error with the driver. When you are sure or at least willing to pay to have someone help then call the Customer Service Department and arrange to speak with an engineer. We will work with you to determine the cause of the issue. If the issue is one of a defective driver we will correct the problem and provide an updated module(s) to you [no cost]. If the issue is of the customer's making [anything that is not the driver] the engineering time will be invoiced to the customer. Pre-approval may be required in some cases depending on the customer's invoicing policy.

Out of Warranty Repairs

Out of warranty support will be billed. The current minimum repair charge is $125. An open PO will be required.

For Service Contact:

Customer Service Department
Dynamic Engineering
150 DuBois, Suite 3
Santa Cruz, CA 95060
(831) 457-8891 Fax (831) 457-4793
support@dyneng.com

All information provided is Copyright Dynamic Engineering

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