PP004
Parallel Port-Based
DSPTÔ 6001 / 6002 CAMAC CONTROLLER
REFERENCE MANUAL

Revised 03/18/07

Computer Methods
4424 Technology Drive
Fremont, CA 94538
510 824 0252
www.computer-methods.com

TABLE OF CONTENTS

Introduction

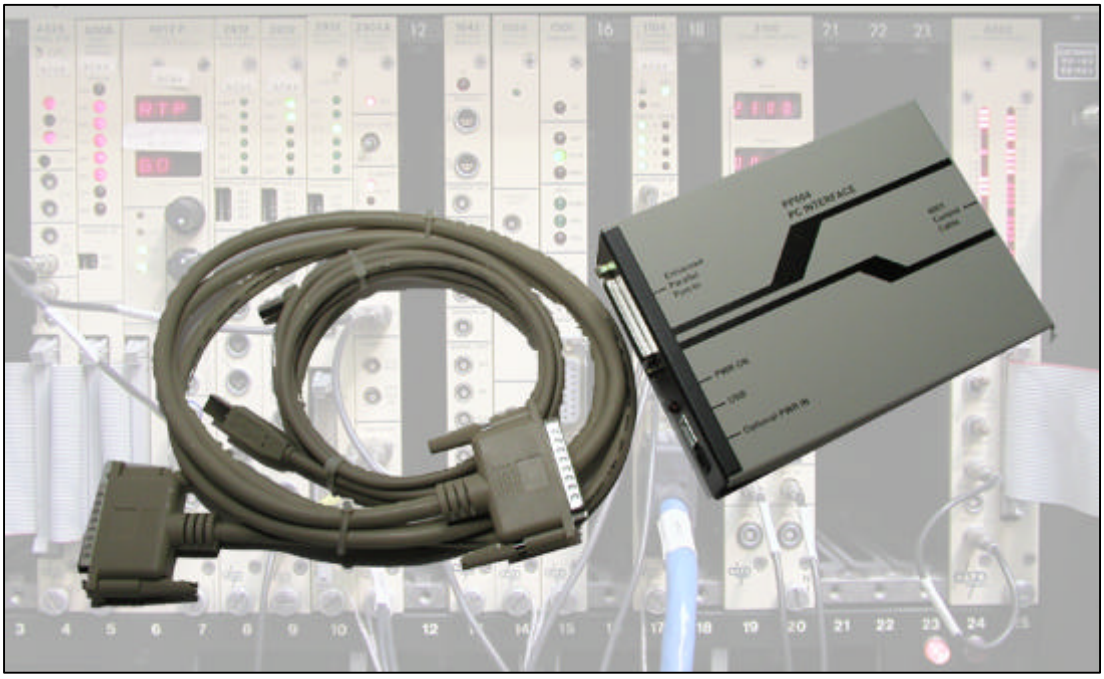

Figure 1: PP004 Interface and Cables

The PP004 interfaces an IBM®-compatible PC with a DSPT™ 6001 / 6002 CAMAC Controller via the PC Parallel Port. This product is designed to replace the functionality of the PC-004 CAMAC interface, which required a 16-bit ISA slot. As newer and faster computers become available, it is increasing difficult to find PCI-based motherboards that support an ISA slot.

The PP004 communicates with the PC through a standard parallel (printer) port operating in EPP mode. Most PCI-based motherboards support EPP mode (enabled via the BIOS). Thus, it is now feasible to upgrade older ISA-based PCs to newer and higher-performance models and continue to utilize existing CAMAC devices. The non-invasive aspect of this interface makes it ideal for use with laptops in the field.

Hardware Setup

The PP004 is shipped with the following items:

- PP004 Interface
- IEEE1284 Extension Cable 1.8M (DB25 to DB25)
- USB A-B Cable 2M
- CD with Drivers and Diagnostics

Before using the PP004, you must decide how to supply power. The PP004 accepts 5 to 15 volts @ 150ma maximum current. There are two options for power:

1. Via USB Connector
2. Via 2.1mm Power Connector

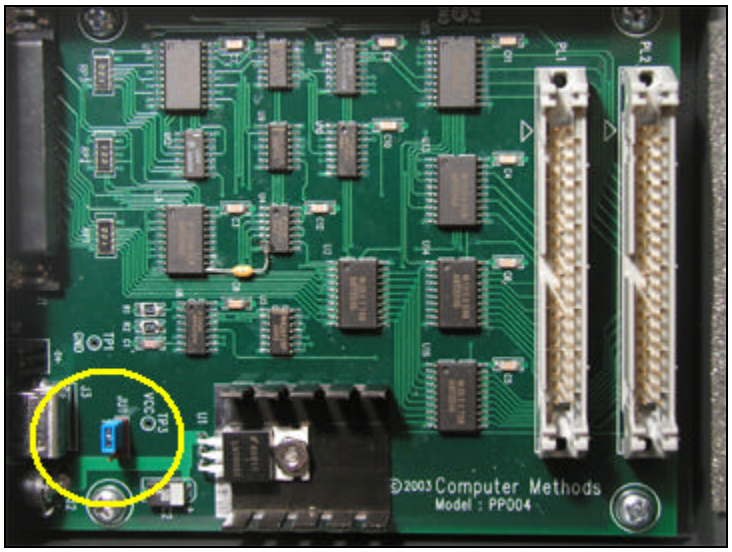

Figure 2: Location of Power Select Jumper JU1

Figure 2 illustrates the location of the blue jumper JU1. When shorting pins 1 & 2, the USB connector is selected for power input. When shorting pins 2 & 3, the power jack is selected for input. The power jack can be used with any DC source between 5 and 15 volts such as a 9v 500ma power-adapter (Digikey Part Number T405-ND).

The green LED on the PP004 front panel indicates that power is connected to the board. The LED will flash orange when a DATASTROBE signal is issued (pin 14 of the parallel port connector). The orange indicator is a useful debugging tool for verifying that communication is established between the PC parallel port and the PP004.

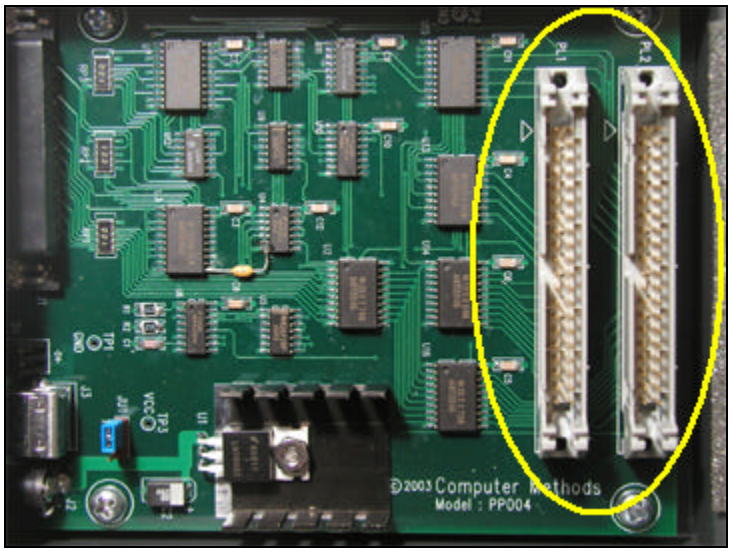

Figure 3: Ribbon cable connectors for two controllers

The PP004 is capable of interfacing with two 6001 / 6002 CAMAC controllers. Figure 3 illustrates the location of connectors PL1 and PL2. A controller connected (via the 40-pin ribbon cable) at PL1 will be addressed as Crate 1. Similarly, a controller connected at PL2 will be addressed as Crate 2. Controllers are selected using software commands written to the parallel port's EPP Address register (see **PP004 Parallel Port Commands**).

The PP004 interfaces with the PC bus via the parallel (printer) port using the DB25 extension cable. The parallel port must be configured in **EPP** mode. This is usually one of the available options for printer settings in the BIOS. Other modes (AT, PS2, SPP, ECP) are not supported. EPP mode is considered to be very reliable and offers ample bandwidth for CAMAC data transfer rates.

PP004 Parallel Port Commands

The I/O registers for a PC parallel port operating in EPP mode are as follows:

Register	I/O Port Offset
SPP Mode Data	0
SPP Mode Status	1
SPP Mode Control	2
EPP Address	3
EPP Data	4

Table 1: Port Offsets for EPP Mode Registers

Communication with the PP004 is handled through EPP Address and Data operations. Many parallel ports require that the SPP Control register be configured with a 0x4 hex to enable EPP transfers. This corresponds to setting control bits **C0**, **C1** and **C3** (active low signals). This is recommended as a one-time initialization step before reading or writing to EPP registers.

Data is written to the crate controller by setting the internal address of the desired CAMAC register using an EPP Address Write (bits D3-D0), followed by an EPP Data Write. Similarly, data is read from the crate by first setting the desired internal address with an EPP Address Write followed by an EPP Data Read. The high order bits (D7-D5) of an EPP Address Write select the desired controller. Table 2 summarizes PP004 commands.

PP004 Operation	EPP Register	R/W
Point To CAMAC Crate Controller and Internal Address. Bits **D3-D0** select address. Bits **D7-D5** select crate (0=Crate1, 1=Crate2). (**D4** is unused)	Address	Write
Read Crate LAM Signals	Address	Read
Write Data at CAMAC Internal Address*	Data	Write
Read Data at CAMAC Internal Address*	Data	Read
**EPP Data transfer auto-increments pointer to next CAMAC Internal Address.*		

Table 2: PP004 Commands

EPP Data operations increment the pointer to the next CAMAC Internal Address. This reduces the overhead of reading or writing data to successive CAMAC registers. The following table presents a sequence of PP004 operations used to read the module ID (**F3 A0** command) from a CAMAC device assuming the following conditions:

1. PP004 Connected to LPT1 (I/O Port Base Address **0x378**)
2. CAMAC 6001 / 6002 Controller ribbon cable connected to **PL1**
3. CAMAC Module is located at Station **1**

Desired Operation	I/O Port Address	Data	R/W
Initialize SPP Control Register for EPP Transfers	0x37A	0x4	Write
Set pointer to Internal CAMAC **A** Register	0x37B	0x3	Write
Write **0** to **A** Register	0x37C	0x0	Write
Write **3** to **F** Register	0x37C	0x3	Write
Write **1** to **N** Register	0x37C	0x1	Write
Set pointer to CAMAC **Cycle** Register	0x37B	0x7	Write
Initiate CAMAC Cycle	0x37C	0x0	Write
Read **QX** and Encoded **LAM** Register	0x37C	QX and LAM	Read
Read CAMAC data **high** byte	0x37C	RH	Read
Read CAMAC data **middle** byte	0x37C	RM	Read
Read CAMAC data **low** byte	0x37C	RL	Read
All values in hexidecimal unless otherwise noted			

*Table 3: Sample CAMAC Command **F3 A0** (Read Module ID)*

PP004 Diagnostic Program "EPPDIAG.EXE"

The 32-bit application **EPPDIAG** provides low-level access to parallel port I/O registers. A setup file for installing **EPPDIAG** can be found on the ***PP004 Applications CD*** distributed with the product. In addition, the VC++ project used to build **EPPDIAG** is also distributed on the application CD. **EPPDIAG** can be used to check a system for EPP compatibility and verify proper operation of the PP004. Figure 4 illustrates the front panel of this diagnostic program.

The **Initialize Control Register** button performs the operation of writing 0x4 to the SPP Control register. This is highly recommended before attempting any EPP operations. The **Test for EPP Mode** button performs a series of operations that detect the presence of EPP Mode support. If this test is not successful than the port is not operating in EPP mode. Check the BIOS or driver settings for the parallel port. Most PCI-based systems support all of the parallel port modes. Some systems support only SPP and ECP modes. ECP is not the same as EPP and is not supported by the PP004.

Assuming that EPP mode operation is possible, the operator first selects a port register to be accessed. Figure 5 illustrates the registers that are available in EPP mode (SPP Data and SPP Status are generally not used for EPP transfers).

If doing a write operation, the hexadecimal value is typed into the value box and the **Write** button is clicked. If doing a read operation, the **Read** button is clicked and the result is returned in the Value box. The **Loop** buttons continuously repeat an operation until the **Stop** button is clicked. Looping is generally used as a debugging tool with an oscilloscope to trace signals on the PP004 board.

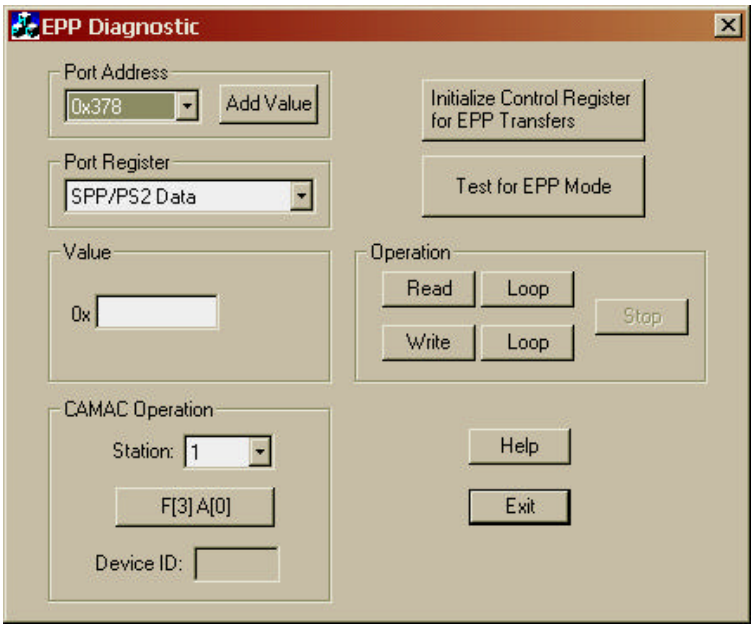

Figure 4: Front Panel of EPP Mode Diagnostic

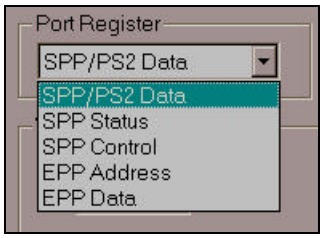

Figure 5: Port Register Selection

This diagnostic could be used to enter the **F3 A0** CAMAC Command sequence described table 3 and retrieve the module ID. This is a very good method of verifying the proper connections and operation of the PP004. The step-by-step instructions for executing this sequence are as follows:

Desired Operation	Port Register	Value	Button
Initialize SPP Control Register for EPP Transfers	SPP Control	0x4	Write
Set pointer to Internal CAMAC **A** Register	EPP Address	0x3	Write
Write **0** to **A** Register	EPP Data	0x0	Write
Write **3** to **F** Register	EPP Data	0x3	Write
Write **1** to **N** Register	EPP Data	0x1	Write
Set pointer to CAMAC **Cycle** Register	EPP Address	0x7	Write
Initiate CAMAC Cycle	EPP Data	0x0	Write
Read **QX** and Encoded **LAM** Register	EPP Data	QX and LAM	Read
Read Module ID	EPP Data	Hi Byte	Read
Read Module ID	EPP Data	Middle Byte	Read
Read Module ID	EPP Data	Low Byte	Read
All values in hexadecimal unless otherwise noted			

*Table 4: **F3 A0** (Read Module ID) Command Sequence Using EPPDIAG Application*

The **CAMAC Operation** command group conveniently performs the sequence described in table 4. After selecting the desired CAMAC station, clicking the **F[3] A[0]** button will set up the command, issue a CAMAC cycle and read back a 16 bit value (formed by reading and combining

the CAMAC RM and RL bytes). The result is displayed (in decimal) as the ***Device ID***. The front panel LED will flash orange to indicate a data transfer.

Accessing the I/O Port in Win98, WinNT, Win2000 and WinXP

In DOS and Windows 98, applications are allowed to execute "_outp" and "_inp" instructions that directly access the PC's I/O port. However, on WinNT, Win2000 and WinXP platforms, these instructions are considered "privileged" and must run under ring 0 in protected mode.

The WinIo library allows 32-bit Windows applications to directly access I/O ports and physical memory. It bypasses Windows protection mechanisms by using a combination of a kernel-mode device driver and several low-level programming techniques.

The WinIO DLL and driver set is included with the EPPDIAG project on the PP004 Applications CD.

PSP9100 Driver Set Compatibility

Drivers to support PSP9100 are located on the application CD in the folder **PSP9100_PP004**. The common elements of PSP9100 drivers are encapsulated in the file **CAM6001.ASM**. By setting the appropriate compiler directives, this component generates an object module (OBJ) that supports a variety of language platforms. The file **CAM6001.ASM** has been modified to support the PP004 interface. Descriptions of the key files provided on the application CD are:

FILENAME	DESCRIPTION
CAM6001.ASM	Assembly language driver exposing a set of functions to read and write to the 6001 / 6002 Controller
CAMTURBO.OBJ	Object module compiled for Turbo Pascal 4
CAMTUNIT.OBJ	Object module compiled for Turbo Pascal Unit encapsulating CAM6001 functions
CAMBASIC.OBJ	Object module compiled for Basic
CAMACPRO.OBJ	Object module compiled for Professional Fortran
CAMACMS.OBJ	Object module compiled for Microsoft Pascal and C
CAMAC_C.OBJ	Object module compiled for C without Pascal attribute
CAMTUNIT.TPU	Turbo Pascal Unit encapsulating CAM6001 functions
MAKE9100.BAT	Batch file that compiles object modules for the various language platforms
NAFTEST.PAS	Turbo Pascal program source for sending and receiving CAMAC commands to/from 6001 / 6002 controller
NAFTEST.EXE	NAFTEST executable compiled for PP004

Table 5: PSP9100 Driver Files

It should be noted that the PP004 does not support DMA transfers analogous to the PC004. Instead, the **DMAI** and **DMAO** functions in CAM6001.ASM are modified to issue repeated calls to **CAMI** and **CAMO** respectively. Data rates in the vicinity of 225K bytes per second have been achieved with the PP004 implementation of this driver.

When specifying a transfer length of three bytes in the **DMASET** function, the PP004 reads/writes the three CAMAC data bytes to a **DWORD** location. This implies that data is read (or stored) in the PC on **DWORD** (32 bit) boundaries. This adheres to the same convention implemented by the Jorway SCSI controller drivers. To move 24- bit wide CAMAC data on boundaries other than 32-bit, simply modify the addressing in the **DMAI** and **DMAO** functions in the CAM6001.ASM driver as needed and recompile the object module using the appropriate compiler directive applicable to the platform in use.

The object modules supplied on the application CD were compiled using Microsoft MASM Version 6.0 and Linker 5.13. The NAFTEST utility was compiled using Borland Turbo Pascal 5.0. Development was performed on a Win98 platform using a DOS Shell.

PPCAM32.DLL AND CAMTEST

The **PPCAM32** folder contains a Visual C++ 5.0 DLL project for PPCAM32.DLL. This DLL contains a set of C functions that communicate with a CAMAC crate controller via the PP004 connected to the parallel port. The **CAMTEST** folder contains a simple dialog-based MFC application that uses the DLL to send simple commands and retrieve data to/from a CAMAC crate. These projects are provided as samples of tested code to communicate successfully with a CAMAC crate over a range of Windows platforms.

The following table documents the functions exported in PPCAM32.DLL. The DLL and corresponding header and library files are available in the "CAMTEST" project folder.

PPCAM32.DLL Function Declarations	
Function	**Description**
`void Crate_set(int *p_C);` `  p_C = pointer to Crate Number`	Selects a CAMAC Crate. Subsequent calls to **cami** and **camo** are directed to this crate.

`Void cami  (int *p_N,int *p_F,int *p_A, int *p_DRW,int *p_Q,int *p_X);`
`  p_N = pointer to station`
`  p_F = pointer to function`
`  p_A = pointer to address`
`  p_DRW = pointer to data`
`  p_Q = pointer to Q flag`
`  p_Q = pointer to X flag`
	Reads a 16-bit value from the CAMAC crate. The state of the "Q" and "X" flags are returned.

`Void camo  (int *p_N,int *p_F,int *p_A, int *p_DRW,int *p_Q,int *p_X);`
`  p_N = pointer to station`
`  p_F = pointer to function`
`  p_A = pointer to address`
`  p_DRW = pointer to data`
`  p_Q = pointer to Q flag`
`  p_Q = pointer to X flag`
	Writes a 16-bit value to the CAMAC crate. The state of the "Q" and "X" flags are returned.

`cami24(int *p_N,int *p_F,int *p_A,unsigned long *p_DRW,int *p_Q,int *p_X);` `  (see **cami**)`	24-bit version of **cami**. Reads a 24-bit value from the CAMAC crate.
`camo24(int *p_N,int *p_F,int *p_A,unsigned long *p_DRW,int *p_Q,int *p_X);` `  (see **camo**)`	24-bit version of **camo**. Writes a 24-bit value to the CAMAC crate.
`unsigned int caml();`	Returns the **LAM** of the highest priority station (or 0 for no LAM).

`camcl(unsigned int *ZCI_bits);`
`  ZCI = pointer to ZCI bit configuration`
`  Bit 1 - Z CYCLE`
`  Bit 2 - C CYCLE`
`  Bit 3 - SET I,CAMAC INHIBIT`
`  Bit 4 - SET BUS INHIBIT REG`
`  Bit 5 - RESET BUS INHIBIT REG`
`  Bit 6 - RESET ACL DETECT REG`
`  Bit 7 - RESET 6002,ACL & BUS INHIBIT REG`
	Writes the CAMAC control register with the desired bit configuration

`dmaset(int *p_C, int *NOB, int *QBL,unsigned int *NTR);`
`  p_C = pointer to Crate Number`
`  NOB = pointer to Number of Bytes per transfer`
	Initializes the DLL for a "dma" transfer. The PP004 emulates DMA transfer by performing repeated calls to **cami** (or **camo**)

QBL = (not used) NTR = Number of transfers	that are optimized for speed.
int dmai(int *p_N,int *p_F,int *p_A,unsigned char *p_DRW,int *p_E); p_N = pointer to station p_F = pointer to function p_A = pointer to address p_DRW = pointer to data p_E = (not used)	Performs a “dma” transfer from the device to the data buffer based on the parameters passed in **dmaset**.
int dmao(int *p_N,int *p_F,int *p_A,unsigned char *p_DRW,int *p_E); p_N = pointer to station p_F = pointer to function p_A = pointer to address p_DRW = pointer to data p_E = (not used)	Performs a “dma” transfer from the data buffer to the device based on the parameters passed in **dmaset**.
unsigned int camcyc(unsigned int *NTR); NTR = (not used)	Returns the number of transfers completed in the last **dmai** or **dmao** call.

*Table 6: Descriptions of Functions Exported in **PPCAM32.DLL***

Figure 6 illustrates the dialog presented by the CAMTEST application. The operation of this dialog is self-explanatory and should be familiar to users with CAMAC experience. The dialog allows the operator to select a crate, station, function and address. Clicking the **EXECUTE** button will issue a **CAMI** or **CAMO** call (depending on the function selected). Radio buttons reflect the status of the **Q** and **X** flags. DMAI calls can be tested by entering the number of transfers, selecting the byte transfer length and clicking the **READ** button. The results of the transfer are logged in the list box.

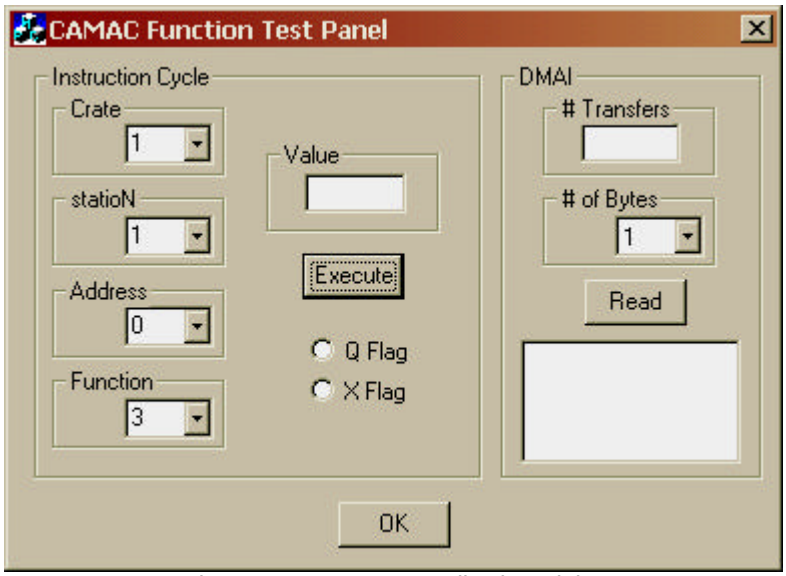

Figure 6: CAMTEST Application Dialog

PPCAM32.INI (PPCAM32.DLL CONFIGURATION)

PPCAM32.DLL assumes that CAMAC crates 1 & 2 will be cabled to connectors PL1 and PL2 (respectively) of a PP004 interfaced to the "primary" parallel port at I/O address 0x378. Similarly, crates 3 & 4 are assumed to be connected to a PP004 at the "secondary" parallel port at I/O address 0x278.

The primary and secondary parallel port assignments can be altered by editing the

file **PPCAM32.INI** and modifying the port address specifications. This file will be located in the same folder as PPCAM32.DLL.

Suppose the parallel port of the target system does not support EPP mode. If a second (EPP-compatible parallel port) is installed, it will be assigned an I/O port address by the Windows Plug and Play driver (assume this new address is 0x278). To configure the DLL for this I/O address, the primary port for the PP004 must be set to 0x278 in the **PPCAM32.INI** file. Changes to the INI file configuration will take effect the next time the DLL is loaded.

On fast motherboards, there may be timing issues when initiating a CAMAC cycle. The DLL must wait at least 1 usec for a CAMAC cycle to be completed. The INI parameter **Delay Cycles** specifies a delay period (in machine cycles) after a CAMAC cycle is initiated to allow the cycle to complete. This period may be increased as needed on faster machines.

ACAP COMPATIBILITY (CMPP004 Kernel Mode Driver)

The PP004 supports operation with **ACAP** – an application for engine combustion analysis. ACAP is a 16-bit Windows® application that communicates with the CAMAC 6001 (or 6002) controller using an ISA-based PC004 interface card. Most contemporary mother boards no longer support the ISA bus making it difficult to migrate the **ACAP** application to newer computers running Windows 2000 or XP.

The **CMPP004_SETUP.EXE** file (distributed on the PP004 Applications CD) installs a special set of drivers and DLLs that interface the PP004 with the ACAP application. At present, the only version of **ACAP** tested with the PP004 interface is **6.0B**. However, previous versions of ACAP are expected to be compatible with the PP004.

It is required that the ACAP application already be installed on the target system before running the **CMPP004_SETUP.EXE** setup file. During the setup process, the user will specify the location of the **ACAP** program folder. The setup file will then perform the following actions:

1. Copy the existing WINCAMAC.DLL and WINCAM32.DLL files in the ACAP program folder to a subfolder named “DLL”
2. Overwrite the WINCAMAC.DLL and WINCAM32.DLL files in the ACAP program folder with versions that support the PP004
3. Install the CAMTEST.EXE utility in the ACAP program folder.
4. Install the CMPP004.SYS driver in the Windows System folder
5. Update the Windows registry with configuration information for the CMPP004.SYS driver
6. Prompt the user to reboot

It is assumed that the PP004 is connected to the primary printer port (LPT1) located at port address 0x378. If this is not the case, it will be necessary to modify the registry to point to the correct port address. To determine the port address of the printer port, use the Device Manager as follows:

1. Right-click on *My Computer* and select *Properties*
2. Select the *Hardware* tab
3. Click the *Device Manager* button
4. Expand the *Ports* section (see figure 7)
5. Right-click on the printer port being used for the PP004 and select *Properties*
6. Select the *Resources* tab (see figure 8)

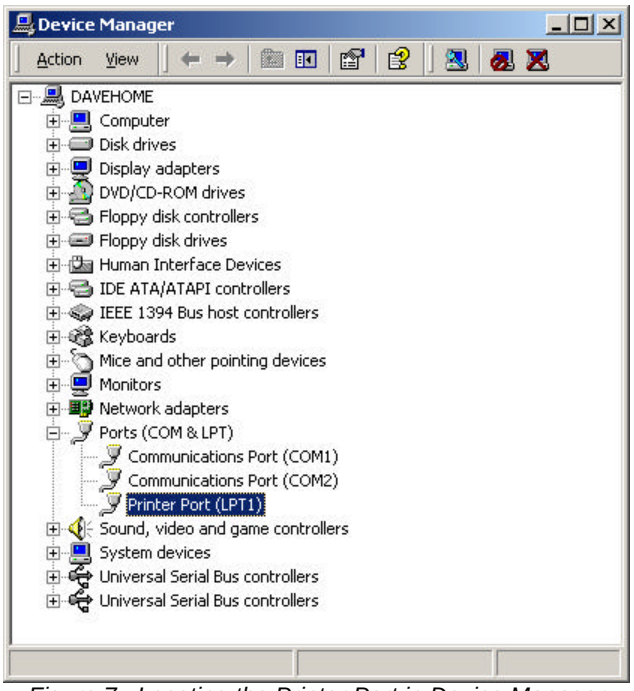

Figure 7: Locating the Printer Port in Device Manager

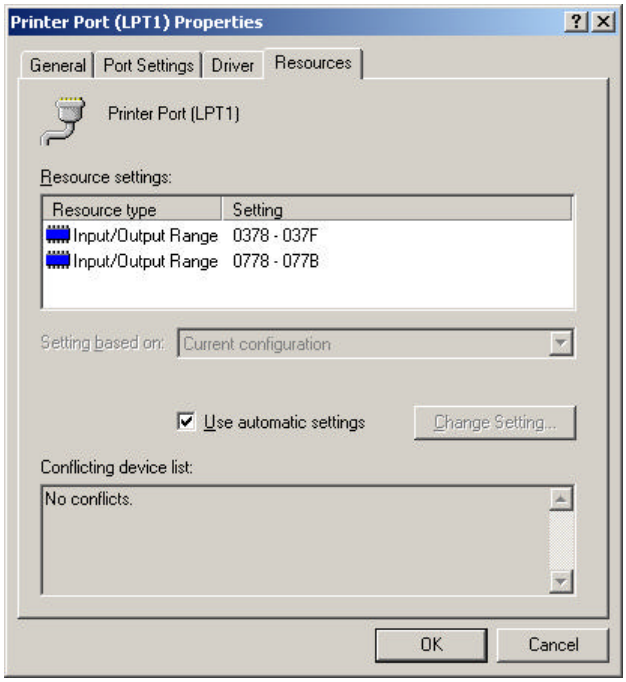

Figure 8: Displaying the Port's Resources

In the example of figure 8, the port address for the LPT1 printer port is 0378 (hex). To edit the appropriate registry key, follow these steps:

1. Click *Start*, *Run*
2. Type the program name "REGEDIT" and click OK
3. Navigate to the following key:
   HKEY_LOCAL_MACHINE\SYSTEM\CurrentControlSet\Services\cmpp004\Parameters (see figure 9)

4. Right-click on **IoPortAddress** and select *Modify*
5. Enter the hexadecimal value of the port address obtained from Device Manager and click OK
6. Reboot the system.

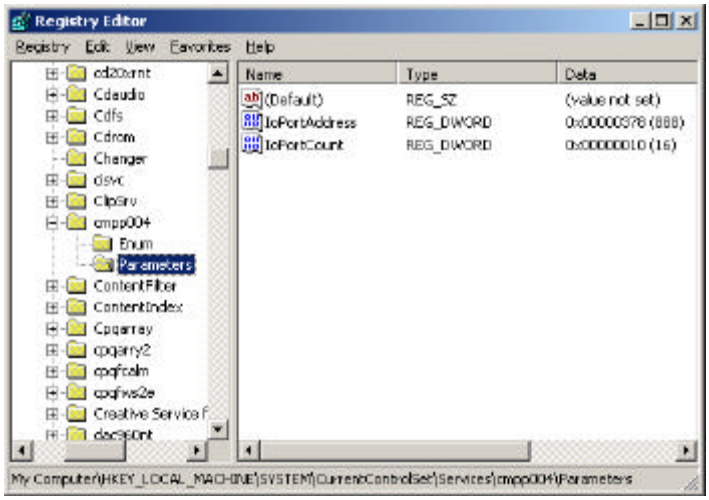

Figure 9: Editing the IoPortAddress registry key

References

Parallel Port Complete: *Programming, Interfacing & Using the PC's Parallel Printer Port* Copyright 2000 Axelson, Jan, Lakeview Research, 5310 Chinook Ln., Madison, WI USA 608 241 5824 mailto:info@lvr.com, URL:http://www.lvr.com.

DSP Technology Inc. Technical Reference Manual: *Model 6001 / 6002 CAMAC Crate Controller*. Copyright 1990 DSP Technology, Inc. Fremont, CA 94538

WinIo v2.0: *Direct Hardware Access Under Windows 9x/NT/2000/XP* Copyright 1998-2002 **Yariv Kaplan http://www.internals.com**