

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.

Visit our website - *Click HERE*

HPCI-PPD553A series

Attached software manual

Hivertec Co., Ltd.
http://www.hivertec.co.jp/

in this manual

The HPCI-PPD553A series board is
HPCI-PPD552
HPCI-PPD553 and

HPCI-PPD552A
It is HPCI-PPD553A.

The Windows version device driver type is

Windows 95, 98	Then hippd553. hippd553.vxd.
Windows NT	sys in hp553w2k. sys
Windows 2000	hp553wxp. sys Windows Vista,
Windows XP	7 (32-bit) hp553wdm. sys
Windows Vista, 7 (64-bit) hp553x64. sys.	

As a Windows version driver I/F library
hippd553. dll is used in common for the above OS.

Unauthorized reproduction or copying of all or part of this document and
program is prohibited. The contents of this series of products are subject to change without prior notice due
to improvements, etc. If you have any questions about the content of this series of products, please feel free to contact us.

Windows 7, Windows Vista, Windows XP Home Edition, Windows XP Professional, Windows 2000 Professional, Windows 98, Windows 95, Visual C++, Visual Basic, Visual Basic.NET are registered trademarks of Microsoft Corporation in the United States and other countries. Other company names and product names mentioned are trademarks or registered trademarks of their respective companies.

HIVER TECH Co., Ltd.
Mitsui Life Shinohashi Bldg., 1-8-11
Shinohashi, Koto-ku, Tokyo
TEL 03-3846-3801
FAX 03-3846-3773
salesÿhivertec.co.jp

Edition 2.7 Issued on February 12, 2015
All rights reserved.

Warranty range

1. The warranty period for this series of products is three years from the date of purchase. If a defect is found at our discretion during the warranty period, we will take back the product from this series and repair or replace it.

2. Regardless of the warranty period, for direct, indirect, secondary damages or loss of profits suffered by the customer or a third party due to the use, supply (delivery date) or failure of our products, Please note that our company will not be held responsible for anything beyond the selling price of this series product.

Disclaimer

1. Please note that we are not responsible for any damage caused by installation, connection, setting, or operation of the product that does not comply with the contents described in this manual.

2. This series of products uses semiconductor products manufactured for use in general electronic equipment (machine tools, measuring equipment, FA/OA equipment, communication equipment, etc.), so any malfunction or failure could directly lead to life-threatening consequences. It is not designed, intended, approved, or guaranteed to be applicable to equipment (medical equipment, transportation equipment, combustion equipment, safety equipment, etc.) that may threaten or cause physical or property harm. not. Therefore, please note that we do not guarantee the safety, quality and performance of this series of products, either explicitly or implicitly, other than what is stated in this manual (or catalog).

3. Regardless of the warranty period, we are not responsible for damages caused by unapproved modification or repair of the product by the customer. Please note that we will not take any responsibility for this.

4. The contents of this manual are not subject to patent rights, copyrights, trademark rights, or other intellectual property rights of our company or third parties. No ownership rights are guaranteed or any license is granted. Also, please note that our company will not be held responsible for any problems related to the intellectual property rights of third parties due to the use of the information described in this manual.

For safe use

Thank you for choosing our NC board series.

Please be sure to read this document as it describes handling and points to note when using the attached software.

Please read it carefully before using it.

In addition, always keep this manual in an easy-to-understand place near the permanent installation location of the NC board to which this manual is attached.

Please refer to and check as appropriate.

Safety notice

Before using this product, be sure to read this user's manual and attached documents carefully and understand the contents.

Please use it correctly. Learn all the knowledge, safety information and precautions of this series product.

Please use it after ripening.

In this user's manual, the safety precautions are ranked as "WARNING" and "CAUTION".

increase.

⚠ warning	Ignoring this indication and handling incorrectly may result in death or serious injury. Indicates content that may possibly occur.
⚠ Note	Ignoring this indication and handling the product incorrectly may result in personal injury. Indicates content that is assumed to cause physical damage.

1. Target user

⚠ Note

⚠ This series of products and manuals are intended for the following types of users.

•Those who have basic knowledge about adding expansion boards and wiring.

•Those who have basic knowledge of control electronic equipment and personal computers.•Those who have basic knowledge of OS operation and software development environment.

2. Compatible OS

⚠ Note

⚠ This product controls boards under Windows 7, Windows Vista, Windows XP Professional, Windows XP HomeEdition, Windows NT4.0, Windows2000 Professional, Windows95, and Windows98.

It is software for Please contact our sales office for use with OS other than the above.

3. Hardware setup, installation, and connection

warning

Please read the user's manual carefully when installing and wiring the board, and follow the instructions in the user's manual.

4. Run the "try it" program

warning

The "Try to run" program in this bundled software is to check whether the board is set and installed correctly, whether the operating environment is set correctly, and to help you understand the functions and operations of the board. . Therefore, do not use the "try to run" program for equipment operation on a regular basis, as it does not include safety measures processing specific to each equipment used. When operating by connecting a motor or device, set operating conditions that take into consideration the characteristics of the motor or device. In particular, it is recommended to use sufficiently safe values during trial operation and gradually change them to the specified values.

5. Target development tool

Note

The sample program in this attached software targets the following development tools.•Microsoft Visual C++ (6.0)

- Microsoft Visual Basic (6.0 / .NET)

Please contact our sales department for use with development tools other than the above.

6. Run the sample program

Note

The sample program in this attached software is for you to understand the procedure for controlling the board and how to create a control program. Therefore, do not use the sample program for constant operation of equipment, as it does not include safety measures processing specific to each equipment used. When operating by connecting a motor or device, set operating conditions that take into consideration the characteristics of the motor or device. In particular, it is recommended to use sufficiently safe values during trial operation and gradually change them to the specified values.

7. Creating a user program



warning



When creating a user program, consider the characteristics of the motor and equipment, and make sure that the necessary interlocks and safety measures are sufficiently included in the design. A slight difference in the program code or data may cause unexpected operation and damage the device or human body. Please be very careful when creating programs and performing test runs.

table of contents

1. Introduction ..... 1 1.1 Structure of attached software ..... 1 1.1.1 32-bit version OS ..... 1 1.1.2 64-bit OS ..... 2 1.2 Contents of the attached disk ..... 3 2. Device driver installation ..... 4 2.1 Installation on Windows 9X..... 4 2.2 Windows NT 4.0 Installation on ..... 4 2.3 Installation on Windows 2000..... 4 2.4 Installation on Windows XP .. .. 5 2.5 Installation on Windows Vista (32-bit), Windows 7 (32-bit) ..... 5 2.6 Installation on Windows Vista (64-bit) and Windows 7 (64-bit) ..... 5 2.7 Uninstallation ..... 5 2.7.1 For Windows 9x, Windows NT4.0, Windows 2000, Windows XP..... 5 2.7.2 For Windows Vista and Windows 7..... 5 3. When using multiple boards ..... 6 3 .1 board slot number confirmation ..... 6 3.2 How to check the slot number..... 6 3.3 Use of Board ID (Board Address) ..... 6 4 How to use the DLL for the driver I/F ..... 7 4.1 Overview..... 7 4.2 Functions List..... 7 4.3 Preparations..... 8 4.4 Control concept..... 9 4.4.1 Data Structure for Board (Device) Recognition ..... 9 4.4 .2 Preparation procedure for board access ..... 10 4.4.3 Abnormal reporting of device driver ..... 11 5. Function details ..... 12 6. Sample program ..... 33 6.1 Operation of the sample program ..... 33 6.1.1 Various operations of the board (device) ..... 34 6.1.2 Operation of each axis on the board ..... 34 6.1.3 Setting conditions for axis operation ..... 37 6.2 Changes to the sample program ..... 38 6.2.1 Project file ..... 38 6.2 .2 Functions that make up the sample program ..... 39 7. "Try It Out" Program ..... 40 7.1 Operation confirmation screen of "Try to move" ..... 40 7.1.1 Displaying device information ..... 40 7.1.2 Individual axis display and motion command ..... 40 7.2 Setting screen for "Try moving" ..... 44 7.2.1 Board selection and device information ..... 44 7.2.2 Changeable Axis Operating Conditions ..... . 45 Appendix A PCL5014 Register Excerpts ..... 47 1. Classification of use of PCL5014 registers ..... 47 2. Individual Registers ..... 48 2.1 Environment register 1 (R6) . ..... 48 2.2 Environment register 2 (R7) ..... 49 2.3 Environment Register 3 (R8) ..... 50 2.4 Counter monitor (R12) ..... 50 2.5 Command monitor 1 (R13) ..... 50 2.6 Command monitor 2 (R14) .....50 Appendix B Major Differences between Attached Software Version 2.0 or Later and Earlier Versions..... 51

Figure Table Contents

Table 1.1-1 32-bit version OS bundled software related chart 1
Table 1.1-2 64-bit version OS bundled software related chart ....
2 Fig. 3.1-1 When using multiple boards .... 6 Fig. 3.3 ÿ1
HPCI-PPD553 board ID setting .... 6 Table 4.2-1 List of
functions .... 7 Table 4.4-1 Abnormal
details List .... 11 Table 5.1-1 Command data
without using write register: sw = 0 .... 18 Table 5.1-2 Write register and
command data: sw = 1... ..... 18 Table 5.1-3 Operation mode..... .....
19 Table 5.1-4 Control mode data 20
Table 5.1-5 Status 1 . ... 21 Table 5.1-6 Status
2 .... 22 Table 5.1-7 Interrupt status
status .... 23 Refer to
Table 5.1-8 Register read command ..... 25 Table 6.1-1 Axis Control Registers
Initial setting 37

1. Introduction

This document is included in the software attached to the HPCI-PPD553A series boards (PPD552, PPD553, PPD552A, PPD553A).

I will explain about it. (Hereafter, the above boards are collectively referred to as PPD55x.)

This bundled software works with Windows 7, Windows Vista, Windows XP Professional/Home Edition (hereafter referred to as XP), Windows 2000 (hereafter referred to as 2K), Windows NT 4.0, and Windows 95/98 (hereafter referred to as 9X) on the PPD55x board. It is used to control the

For explanations of board control, related items are covered in each function and in "Appendix A", but if you need a detailed explanation, please refer to the user's manual attached to the board.

1.1 Configuration of attached software

1.1.1 32-bit OS

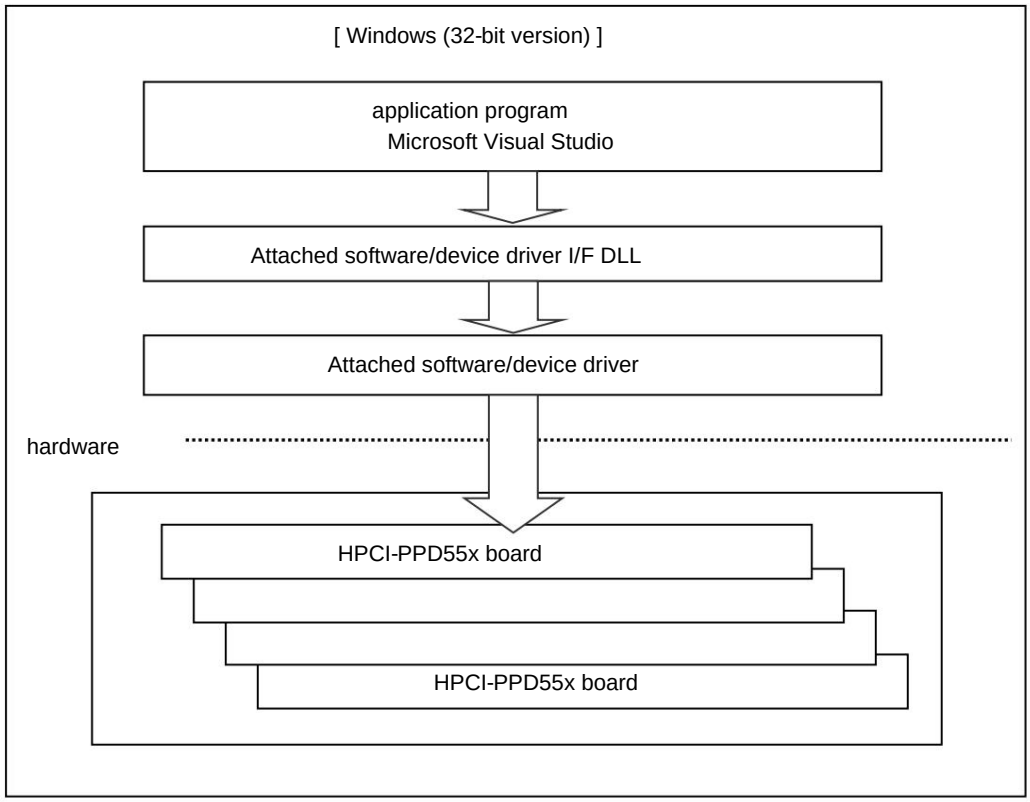


Table 1.1-1 32-bit version OS attached software related chart

1.1.2 64-bit OS

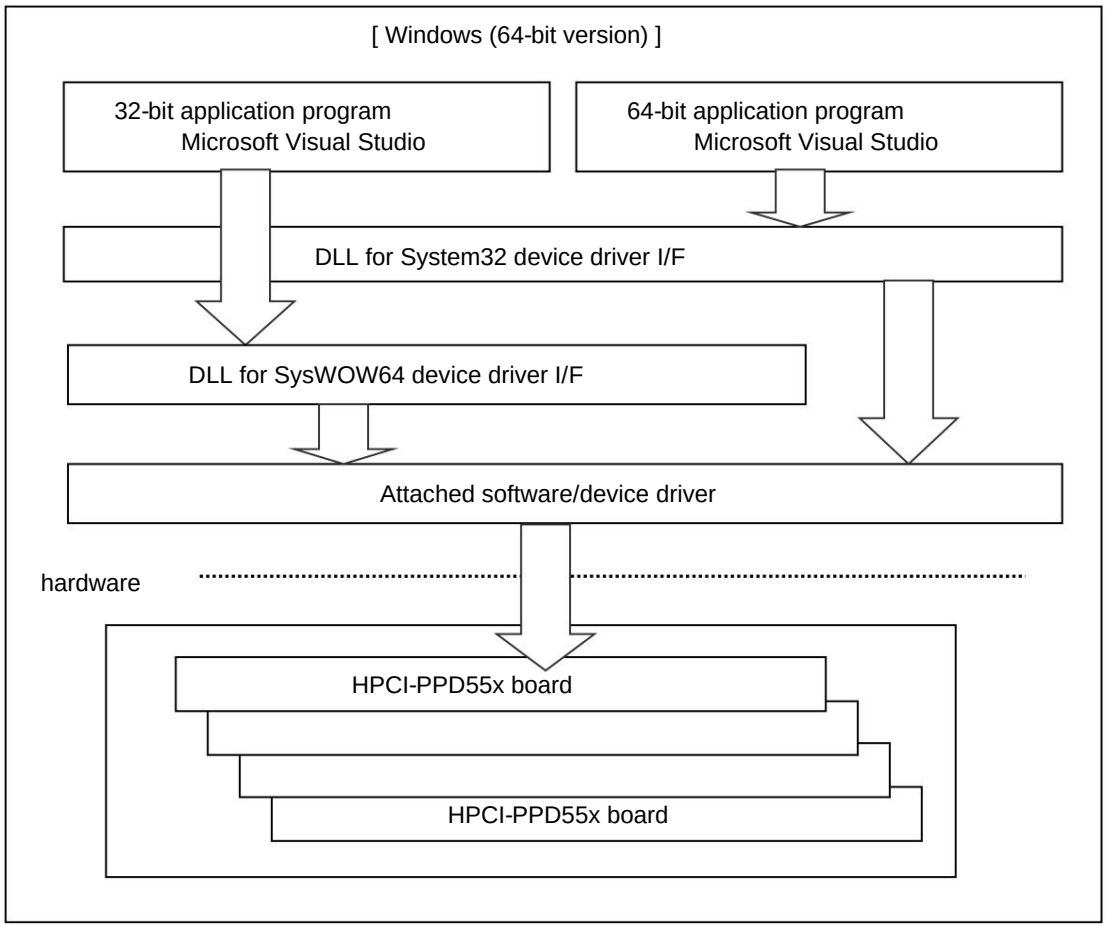


Table 1.1-2 64-bit version OS attached software related chart

1.2 Contents of attached disk

ÿÿhippd553.dll ÿÿÿÿ DLL for driver I/F

ÿÿp553uins.exe ÿÿÿÿ Executable program for uninstallation (common to Win9X, WinNT, Win2K)
ÿÿHistory.txt ÿÿÿÿ Update history of attached disk
ÿÿWin9xÿÿÿhippd553.vxd ÿÿÿÿ Windows9X virtual device driver
ÿ ÿÿhippd553.inf ÿÿÿÿ Installation information for Windows9X (for VxD installation)
ÿÿWinNtÿÿÿhippd553.sys ÿÿÿÿ WindowsNT4.0 device driver
ÿ ÿÿp553inst.inf ÿÿÿÿ Installation information file
ÿ ÿÿp553inst.bat ÿÿÿÿ Batch file for installation
ÿÿWin2kÿÿÿhp553w2k.sys ÿÿÿÿ Windows2000 device driver
ÿ ÿÿhp553w2k.inf ÿÿÿÿ Installation information file
ÿÿWinXPÿÿÿhp553wxp.sys ÿÿÿÿ WindowsXP device driver
ÿ ÿÿhp553wxp.inf ÿÿÿÿ Installation information file
ÿÿWin_x86ÿÿ dpinst.exe ÿÿÿÿ Driver package installer
ÿ ÿÿdpinst.xml ÿÿÿÿ xml file for driver package installer
ÿ ÿÿhp553wdm.cat ÿÿÿÿ security catalog file
ÿ ÿÿhp553wdm.sys ÿÿÿÿ Windows Vista, 7 (32-bit version) device driver
ÿ ÿÿhp553wdm.inf ÿÿÿÿ Installation information file
ÿ ÿÿhippd553.dll ÿÿÿÿ DLL for driver I/F
ÿÿWin_x64ÿÿ dpinst.exe ÿÿÿÿ Driver package installer
ÿ ÿÿdpinst.xml ÿÿÿÿ xml file for driver package installer
ÿ ÿÿhp553x64.cat ÿÿÿÿ security catalog file
ÿ ÿÿhp553x64.sys ÿÿÿÿ Windows Vista, 7 (64-bit version) device driver
ÿ ÿÿhp553x64.inf ÿÿÿÿ Installation information file
ÿ ÿÿhippd553.dll ÿÿÿÿ DLL for driver I/F (32-bit version)
ÿ ÿÿhippd553.x64 ÿÿÿÿ Driver I/F DLL (64-bit version)
ÿÿdocumentÿPPD553_U_2_10E.pdf ÿÿÿÿ User's Manual
ÿ ÿÿPPD553_W_2_60E.pdf ÿÿÿÿ Software Manual
ÿÿInclude ÿVc_x86 ÿhippd553.lib ÿÿÿÿ (32-bit version) import library (for building C applications)
ÿ ÿ ÿÿÿhippd553.h ÿÿÿÿ Header file (for building C applications)
ÿ ÿÿVc_x64 ÿhippd553.lib ÿÿÿÿ (64-bit version) import library (for building C applications)
ÿ ÿ ÿÿÿhippd553.h ÿÿÿÿ Header file (for building C applications)
ÿ ÿÿVb6 ÿÿÿhippd553.bas ÿÿÿÿ DLL function definition standard module (for building VB 6.0 applications)
ÿ ÿÿVb_netÿhippd553.vb ÿÿÿÿ DLL function definition file (for building VB .NET applications)
ÿÿSampleÿÿVc ÿÿÿspp55300.exe ÿÿÿÿ Sample execution file (Visual C)
ÿ ÿ ÿÿspp55300.dsw ÿÿÿÿ Sample project workspace for Visual C
ÿ ÿ ÿÿspp55300.dsp Sample project file
ÿ ÿ ÿÿspp55300.opt sample opt file
ÿ ÿ ÿÿspp55300.c ÿÿÿÿ Sample source program for Visual C
ÿ ÿ ÿÿspp55300.h sample header file
ÿ ÿ ÿÿspp55300.rc sample resource file
ÿ ÿ ÿÿresource.h Sample resource header file
ÿ ÿ ÿÿbgreen.bmp sample bitmap (green)
ÿ ÿ ÿÿbred.bmp sample bitmap (red)
ÿ ÿ ÿÿbwhite.bmp Sample bitmap (white)
ÿ ÿ ÿÿhippd553.xxx ÿ ÿÿVcpp ÿÿÿÿ 3 types of driver attachment files for C language (xxx = lib,h,dll)
ÿÿspp55301.exe (ÿÿC++ Sample execution file (Visual C++ version)
ÿ ÿ ÿÿspp55301.dsw ÿÿÿÿ Sample project workspace for VC++(C++)
ÿ ÿ ÿÿspp55301.dsp Sample project file
ÿ ÿ ÿÿspp55301.opt sample opt file
ÿ ÿ ÿÿreadme.txt Explanation of the sample configuration file
ÿ ÿ ÿÿspp55301dlg.cpp ÿÿÿÿ Sample source program for VC++(C++)
ÿ ÿ ÿÿspp55301dlg.h ÿÿÿÿ Sample header file
ÿ ÿ ÿÿhippd553.xxx ÿÿÿÿ C language driver attachment files (xxx = lib,h,dll)
ÿ ÿ ÿÿOther files ÿÿÿÿ See the "readme.txt" file.
ÿ ÿ ÿÿres ÿÿbgreen.bmp ÿÿÿÿ Sample bitmap file (green)
ÿ ÿ ÿbred.bmp Bitmap file (red)
ÿ ÿ ÿbwhite.bmp ÿÿÿÿ Bitmap file (white)
ÿ ÿ ÿOther files ÿÿÿÿ See the "readme.txt" file.
ÿ ÿÿVb6ÿÿÿspp55302.exe ÿÿÿÿ Sample executable file (Visual Basic 6.0)
ÿ ÿ ÿÿspp55302.vbp ÿÿÿÿ Sample project file for VB
ÿ ÿ ÿÿspp55302.frm ÿÿÿÿ Sample form module file
ÿ ÿ ÿÿspp55302Module.bas ÿÿÿÿ Sample standard module file
ÿ ÿ ÿÿhippd553.bas ÿÿÿÿ DLL function definition standard module file for driver I/F
ÿ ÿ ÿÿhippd553.dll ÿÿÿÿ DLL for driver I/F
ÿ ÿÿVb_net ÿspp55303.sln ÿÿÿÿ solution file
ÿ ÿÿspp55303.suo ÿÿÿÿ Solution user option
ÿ ÿÿspp55303.vbproj ÿÿÿÿ Project file
ÿ ÿÿspp55303.vbproj.user ÿÿÿÿ Project user option
ÿ ÿÿspp55303.resx ÿÿÿÿ Form module resource template
ÿ ÿÿspp55303.vb ÿÿÿÿ Form module resource file
ÿ ÿÿspp55303Module.vb ÿÿÿÿ Standard module file
ÿ ÿÿAssemblyInfo.vb ÿÿÿÿ Assembly information file
ÿ ÿÿhippd553.vb ÿÿÿÿ DLL function definition file for Driver/F
ÿ ÿÿobj ÿRelease ÿspp55303.exe ÿÿÿÿ Sample executable file (Visual Basic .NET)
ÿÿTest ÿÿhippd553.dll ÿÿÿÿ DLL for driver I/F
ÿÿtpp55300.exe ÿÿÿÿ Executable file to run

2. Device driver installation

2.1 Installation on Windows 9X

(1) After confirming that the personal computer is powered off, connect the HPCI-PPD55x board to the PCI bus port of the personal computer. Attach to the lot. Turn on the computer and start Windows.

(2) When Windows 9X starts, HPCI-PPD55x is detected by the system and the necessary device drivers are automatically installed. The driver installation screen will be displayed.

(3) When the system requests the installation source directory, insert the attached CD into the CD drive. ÿ Be sure to check the check box of ÿSpecify search location.
(In Windows95, click the Specify Location button.)

ÿ CD drive: Specify \WIN9X.
Follow the system instructions to complete the installation.

2.2 Installation on Windows NT4.0

(1) Installation of device driver

ÿ Insert the attached CD into the CD drive. Start
NT Explorer and select CD drive:\WinNT\p553inst.inf. ÿ Next, click the right mouse button.
Select "Install" from the pop-up menu that appears.
This operation will start the installation of the device driver. Follow the system instructions to complete the installation.

Installation can also be started by executing CD drive:\WinNT\p553inst.bat from the command prompt.

(2) Starting and stopping the
device After the installation is completed, the device driver is set to "automatic start", and the service for the HPCI-PPD55x board is started when Windows NT is started. If you need to change to stop for some reason, do the following.

(1) From the control panel, double-click the "Devices" icon, and select from the list of devices.
Select "Hivertec HPCI-PPD553(PCI)". (2) Click the
"Startup" button to display the "Startup type" dialog.
Select "Stop" and press "OK".
When restarting the HPCI-PPD55x device, perform the "Device" operation on the control panel.
Select "Hivertec HPCI-PPD553(PCI)" and press the "Start" button.

2.3 Installation on Windows 2000

(1) After confirming that the personal computer is powered off, connect the HPCI-PPD55x board to the PCI bus port of the personal computer. Attach to the lot. Turn on the computer and start Windows.

(2) When Windows 2000 starts, the HPCI-PPD55x is detected by the system and the necessary devices are automatically installed. The driver installation screen will be displayed. (3)
When the system requests the installation source directory, insert the attached CD into the CD drive. ÿ Be sure to check the check box of ÿSpecify search location. ÿ If the screen "Some files on the Hivertec HPCI-PPD553 (Win2k) DriverDisk are required." is displayed, click Browse and specify the CD drive:\WIN2k\hp553w2k.sys. Follow the system instructions to complete the installation.

2.4 Installation on Windows XP

*1. When the device driver for Windows2000 was used on WindowsXP
Uninstall the device driver (run P530uins.exe) before following steps
please.

(1) After confirming that the personal computer is powered off, connect the HPCI-PPD55x board to the PCI bus port of the personal computer.
Attach to the lot. Turn on the computer and start Windows.
(2) When WindowsXP starts, HPCI-PPD55x is detected by the system and the necessary device drivers are automatically installed.
The driver installation screen will be displayed. Insert the attached CD into the CD drive.
ÿ Check Install software automatically (recommended).
ÿ Select Hivertec HPCI-PPD553 (WinXP).
(5) A warning "Windows logo test not passed" may be displayed, but select "Continue" to continue the installation.
Please continue the rule. Follow the system instructions to complete the installation.

2.5 Installation on Windows Vista (32-bit) and Windows 7 (32-bit)

ÿ Before inserting the PPD55x board into the PCI bus slot of the personal computer, turn on the power of the personal computer and start Windows.
start the .
ÿ CD drive: \win7_x86\dpinst. Start the exe.
ÿ When “dpinst.exe” is started, click “Next” to continue. ÿ After completing the installer,
turn off the power of the personal computer and insert the HPCI-PPD55x into the PCI bus slot of the personal computer.
Put it on.
ÿ Turn on the computer and start Windows.
ÿ The device will be installed automatically, and you will be prompted to reboot, so reboot to complete the installation.

2.6 Installation on Windows Vista (64-bit) and Windows 7 (64-bit)

(1) Before inserting the PPD55x board into the PC's PCI bus slot, turn on the power of the PC and
Start windows.
ÿ CD drive: \win7_x64\dpinst. Start the exe.
ÿ When “dpinst.exe” is started, click “Next” to continue. ÿ After completing the installer,
turn off the power of the personal computer and insert the HPCI-PPD55x into the PCI bus slot of the personal computer.
Put it on.
ÿ Turn on the computer and start Windows.
ÿ The device will be installed automatically, and you will be prompted to reboot, so reboot to complete the installation.

2.7 Uninstall

2.7.1 For Windows 9x, Windows NT4.0, Windows 2000, Windows XP

Insert the attached CD into the CD drive.
Start Explorer and run CD drive:\P553uins.exe.

2.7.2 For Windows Vista and Windows 7

Windows "Start" ÿ "Control Panel" ÿ "Uninstall a program"
ÿ Right-click on "Windows Driver Package Hivertec HPCI-PPD55x"
Uninstall it.

3 When using multiple boards

I want to install multiple PPD55x boards in the same computer and have each board correspond to the external connection on a one-to-one basis.
I will explain the case.

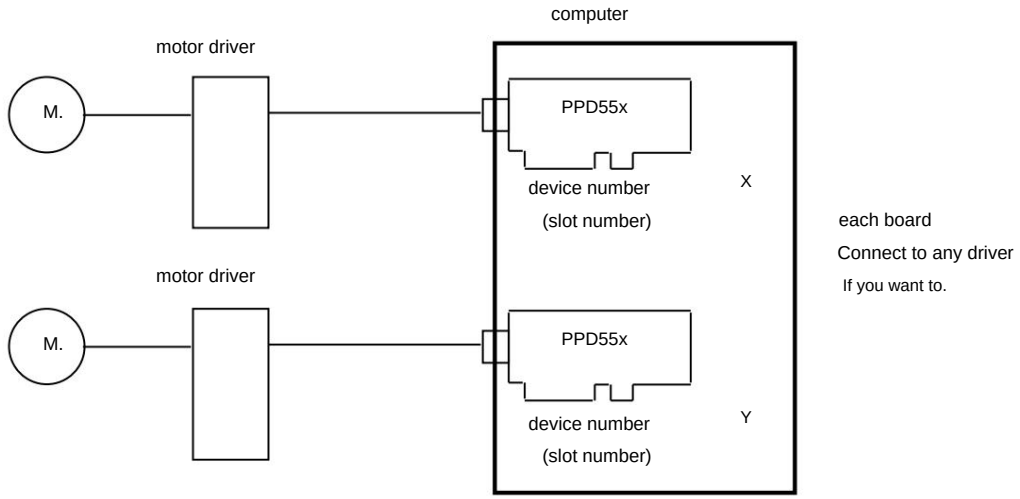


Figure 3.1-1 When using multiple boards

3.1 Confirmation of board slot number

Each personal computer assigns a fixed number to the slot in which a PCI bus card is installed.
By knowing the number, it is possible to connect the board in the slot to be installed with any motor driver.

3.2 How to check the slot number

By starting the following executable file, the device number (slot number) currently installed in the slot can be easily obtained.
You can check.

\test\tpp55300.exe ••• Common to 2-axis and 3-axis.

This program acquires device information from the DLL.
This allows you to get the device number (slot number).
(For details, see "5. Function details (2) Acquisition of device information".)

3.3 Use of board ID (board address)

The board ID (0-15) set by the jumper on the board can be used for the following boards in the PPD55x board.
This usage will be explained later.

(1) Board ID available boards: PPD552A to PPD553A
(2) Fixed board ID (15) board PPD552 to PPD553

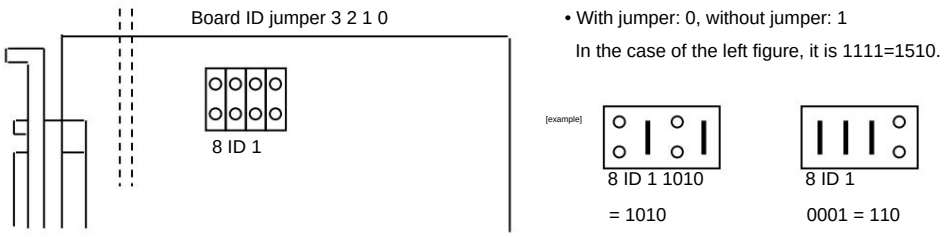


Figure 3.3-1 HPCI-PPD553 board ID setting

4 How to use DLL for driver I/F

4.1 Overview

Windows DLL for Windows Vista, Windows XP, Windows 2K, Windows NT, Windows 9X
This is a group of functions for controlling the HPCI-PPD55x board.

Each function is "Microsoft Visual C++ (Ver6 or later)" (hereinafter VC++),
"Microsoft Visual Basic(Ver6.0)" (hereinafter VB6)
"Microsoft Visual Basic .NET" (hereafter VB.NET)
is invoked as an external function from
The application program calls the DLL function, and the DLL is the HPCI-PPD55x device driver.
access the bar.

The device driver is hippd553.vxd for Windows9X, hippd553.sys for WindowsNT, and for Windows2K
hp553w2k.sys, hp553wxp.sys for Windows XP, and hp553wdm.sys for Windows Vista are used.

4.2 List of functions

The driver I/F DLL contains the following 18 types of functions.

No	ÿ ÿ ÿ ÿ	Noh	Description page
1	hppd553_GetDeviceCount()	Machine Get number	11
2	hppd553_GetDeviceInfo() 3	of boards Get device	12
	hppd553_OpenDeviceID() 4	information Open device (see board ID) Open	13
	hppd553_OpenDevice() 5	device (ignore board ID) Close device Write	14
	hppd553_CloseDevice() 6	command buffer Write operation mode buffer	15
	hppd553_CmdWrite() 7	Write control mode buffer Read status 1 Read	16
	hppd553_DousaWrite() 8	status 2 Interrupt Status read Register read	18
	hppd553_SeigyoWrite() 9	General-purpose I/O number switching port read	19
	hppd553_status1Read() 10	General-purpose I/O number switching port	20
	hppd553_status2Read() 11	write General-purpose input port byte read	21
	hppd553_intstatusRead() 12	General-purpose input port word read General-	22
	hppd553_regRead( ) 13	purpose output port byte read General-purpose	23
	hppd553_ioswRead()	output port byte write General-purpose output	25
	hppd553_ioswWrite() 14	port word read General-purpose output port	25
	hppd553_inpReadB()	word write ELS polarity selection port read ELS	26
	hppd553_inpReadW() 15	polarity selection port write Comparator	26
	hppd553_outpReadB()	simultaneous start read Comparator simultaneous	27
	hppd553_outpWriteB()	start write Simultaneous start/stop read by	27
	hppd553_outpReadW()	general-purpose input Simultaneous start/stop	27
	hppd553_outpWriteW()	write by general-purpose input	27
16	hppd553_ElstRead()		29
	hppd553_ElstWrite() 17		29
	hppd553_CmpcRead()		30
	hppd553_CmpcWrite() 18		30
	hppd553_diRead()		31
	hppd553_diWrite()		31

Table 4.2-1 List of functions

4.3 Preparation

The procedure for using this DLL will be explained.

[Building applications with VC++]
Add the following files to your project.
•hippd553.lib ••• Import library •hippd553.h
••• Header file for building C application

(Note) 1. DLL functions are written in C language.
2. The DLL function prototype declaration in the header file is described as follows.
is.

```
//-------------------------------------
// DLL function pro and type
declaration // -------------------------------------
#ifdef __cplusplus extern "C" { #endif

    DLL function prototype declaration

#ifdef __cplusplus }
#endif
```

This is the application in C++ coding (file extension = cpp)
This is a process in preparation for
creation. 3. The definition of "#ifdef __cplusplus" is for "VC++".
It can be changed to an explicit declaration when used in other languages.

```
(example)
#define CPLUS 1
"""""""""
#if          CPLUS
"""""""""
#endif
```

"1" for C++ (file extension: cpp) "0"
for C (file extension: c)

[Building an application with VB6] Add
the standard module "hippd553.bas" to the project. This file contains
external function declarations (Declare declarations) and user-defined type declarations.

[VB. NET application construction]
Add the standard module "hippd553.vb" to the project. This file contains
external function declarations (Declare declarations) and user-defined type declarations.

ÿ Notes

(1) The attached software does not support platforms other than machines equipped with Intel-compatible CPUs.
(2) Since the board ID is used, 16 PPD55x boards are controlled.

4.4 Control concept

DLL and device driver can control multiple HPCI-PPD55x boards.
To access one HPCI-PPD55x board, first open this device and enter the
It is necessary to acquire the device ID value that serves as a foothold.

In order to open a device, information about what kind of hardware resources the device is to be opened
is required. (Hardware resources, such as I/O port addresses and IRQ numbers, are provided by the Windows system
is determined by)

4.4.1 Data structure for board (device) recognition

Prepare the following HPCDEVICEINFO type structure for board recognition .

[C language: Windows: VC++]

```
typedef struct _HPCDEVICEINFO {
    DWORD       nBusNumber;          /* bus number */
    DWORD       nDeviceNumber;       /* Device number (PCI slot number) */
    DWORD       dwIoPortAddress;     /* I/O port address */
    DWORD       dwIrqNo; dwNumber;   /* IRQ number */
    DWORD                            /* Control number */
    DWORD dwBoardID;                 /* Board ID (0-15) */
} HPCDEVICEINFO, *PHPCDEVICEINFO
```

[Windows: VB6]

```
Public Type HPCDEVICEINFO
    nBusNumber          As Long      'bus number
    nDeviceNumber As Long            'device number
    dwIoPortAddress As Long          'I/O port address
    dwIrqNo             As Long      'IRQ number
    dwNumber            As Long      'Control number
    dwBoardID           As Long      'Board ID (0-15)
End Type
```

[Windows: VB.NET]

```
Public Structure HPCDEVICEINFO
    Dim nBusNumber          As Integer 'Bus number
    Dim nDeviceNumber As Integer 'Device number
    Dim dwIoPortAddress As Integer       'I/O port address
    Dim dwIrqNo             As Integer   'IRQ number
    Dim dwNumber            As Integer 'Management number
    Dim dwBoardID           As Integer 'Board ID
End Structure
```

(Note) 1. Management numbers are not used in Windows 9X.
"INVALID_HPC_NUMBER(-1)" is always stored.
2. There are boards that can use board IDs and boards that cannot (fixed at 15).
be careful.

4.4.2 Preparation procedure for board access

(1) Acquisition of device information for all boards used

First, get the device information of all HPCI-PPD55x in the "HPCDEVICEINFO" type structure area (array of). • hppd553_GetDeviceCount()•••Check the number of boards • hppd553_GetDeviceInfo() •••Get device information of all boards

(2) Device open for each board Pass the

device information of one HPCI-PPD55x to the device open function. As a result, the HPCI-PPD55x is opened, and the device open function returns the device ID value for accessing this board. If the number of boards is two or more, this process is performed for each board. • hppd553_OpenDeviceID() … Board open processing [Refer to board ID] • hppd553_OpenDevice() … Board open processing [Ignore board ID]

(3) Initial setting of board signal processing method

After that, you will be able to access the HPCI-PPD55x using this “device ID as a handle”. Before initializing the board, use the following function to set the device usage conditions.

• hppd553_ElstWrite() ••• ELS polarity setting (extreme limit sensor) • hppd553_CmpcWrite() ••• Comparator simultaneous start setting • hppd553_diWrite() ••• Simultaneous start/stop setting by general-purpose input

(4) Initialization of each board and each

axis After the above setting, initialize each axis of all boards to be used. This enables pulse output operation for each axis as normal operation. When using general-purpose input/output, set the number of inputs/outputs and perform initial output to the output port.

ÿNotesÿ •Check the following when each axis is connected to the motor operable state.
•Activation test of ±ELS signal (only the sensor is activated when the motor is stopped.)
•Signal input state when the servo alarm signal is connected.•Origin signal (OLS/Z phase) input status. If the above signals are not input correctly, normal operation is not guaranteed. •If the command pulse output to the motor does not work properly, the following checks are required. - Does the command pulse output setting match the "servo driver" input?•Is there any factor to stop the motor in the "servo driver" input signal?

(5) When all processing is completed and the application is to be terminated, perform the “close processing” of the opened device. Please.

• hppd553_CloseDevice() ••• Close processing of board (for one board)

4.4.3 Device driver error report

When using various functions of the device driver, if the return value of the function is an abnormal value (0), this abnormal content is displayed.

Performs processing corresponding to reading and abnormal contents.

(1) Read error details : Start GetLastError() function

[C language: VC++]

```
DWORD dErrorCode;
dErrorCode = GetLastError();
```

[VB6]

```
dErrorCode As Long;
dErrorCode = GetLastError()
```

[VB.NET]

```
dErrorCode As Integer
dErrorCode = GetLastError()
```

(2) List of error contents

No	Read value (value: C language hexadecimal notation)	Error content
1	NO_ERROR (0x00000000)	No error (no device)
2	ERROR_FILE_NOT_FOUND (0x00000002)	Device driver does not exist
3	ERROR_NOT_ENOUGH_MEMORY (0x00000008)	Insufficient memory for storing device information
4	ERROR_HPC_ALREADY_OPENED (0x20000001)	Opening an already opened device
5	ERROR_HPC_ILLEGAL_DEVICE (0x20001000)	Not Recognized as HPCI Board
6	ERROR_INVALID_PARAMETER (0x00000087)	No board corresponding to the specified device information
7	ERROR_HPC_INVALID_HANDLE(0x20000002)	Invalid device ID specified
8	ERROR_NOT_READY (0x00000021) Table 4.4-1	Device I/O port not available

List of error contents

(3) Check items when an error occurs

ÿ NO_ERROR ••••••••ÿNo error occurred.

ÿ ERROR_FILE_NOT_FOUND •••ÿThe device driver is not installed.

•ÿThe device driver is not stored in the specified folder.

(3) ERROR_NOT_ENOUGH_MEMORY ÿThere is not enough memory for the application.

ÿInsufficient main memory of personal computer.

ÿ System resources (memory for OS) are insufficient.

ÿLaunching many applications.

ÿ Multiple windows were opened at once.

ÿ ERROR_HPC_ALREADY_OPENED• ÿAn open command was issued to an already opened device.

ÿThe opened device can be used until it is closed.

(Do not open multiple times)

ÿDevices to open when using two or more boards

Please check for updated information.

ÿ ERROR_HPC_ILLEGAL_DEVICE •ÿThe device information is normal, but the board functions do not match.

ÿ ERROR_INVALID_PARAMETER ÿCheck the contents of the device information specified by the device open function.

ÿ ERROR_HPC_INVALID_HANDLE •ÿThis is not the "device ID" obtained by opening the device.

ÿThis device has already been closed.

ÿ ERROR_NOT_READY ••••• ÿThere is no input/output port inside the device (board).

(The OS may be unstable.)

5. Function details

Hexadecimal notation in the function description is written in C language.

When using VB, read the C language hexadecimal notation as follows.

C language	VB6.0, VB. NET
0x00000000	&H0
0x2000 1000	&H20001000

(1) Acquiring the number of devices Function	hppd553_GetDeviceCount() Acquire the number of HPCI-PPD55x boards currently installed in the personal computer. PPD552 to PPD553, PPD552A to PPD553A This is the total number of
format	boards. [C language: VC++] DWORD WINAPI hppd553_GetDeviceCount(WORD dummy); [VB6] Declare Function hppd553_GetDeviceCount Lib "hippd553.DLL" _ (ByVal dummy As Integer) As Long [VB. NET] Declare Function hppd553_GetDeviceCount Lib "hippd553.DLL" _ (ByVal dummy As short) As Integer Argument • DWORD dummy ••• Not actually
used. (Set an arbitrary value.) Return value Number of HPCI-PPD55x boards The value of the number of boards is 1 or more. When the value of the number of sheets is 0 --- See "4.4.3 Device driver error report (P10)". The result is as follows. ÿ ERROR_FILE_NOT_FOUND (0x00000002) ÿ The device driver does not exist. ÿ NO_ERROR (0x00000000) ÿ The device driver of HPCI-PPD55x is installed.	
Calling example [C language: VC++]	DWORD count; count = hppd553_GetDeviceCount(1); [VB6] Dim count As Long count = hppd553_GetDeviceCount(1) [VB. NET] Dim count As Integer count = hppd553_GetDeviceCount(1)

(2) hppd553_GetDeviceInfo() Function	Get device information
Acquires the	device information of the HPCI-PPD55x board currently installed in the personal computer. As a result, device information is stored in an array of type HPCDEVICEINFO. This value is used when opening the device.
format	[C language: VC++] DWORD WINAPI hppd553_GetDeviceInfo (DWORD* pcnDeviceNumber, PHPCDEVICEINFO pHpcDeviceInfo); [VB6] Declare Function hppd553_GetDeviceInfo Lib "hippd553.DLL" _ (ByRef pcnDeviceNumber As Long, pHpcDeviceInfo As HPCDEVICEINFO) As Long [VB. NET] Declare Function hppd553_GetDeviceInfo Lib "hippd553.DLL" _ (ByRef pcnDeviceNumber As Integer,pHpcDeviceInfo As HPCDEVICEINFO) As Integer
Arguments	• DWORD* pcnDeviceNumber Pass the address of the DWORD type area that stores the maximum number of boards for which information is to be obtained. After calling the function, the number of boards for which information was actually obtained is stored. • PHPCDEVICEINFO pHpcDeviceInfo The address of the area where the device information of each board should be set, that is, the HPCDEVICEINFO type Pass the starting address of the array.
Return value	Processing result 1: Successful --- Check the number of sheets for command and return. 0: Failed --- See "4.4.3 Device driver error report (P10)".

Calling example	It is assumed that two HPCI-PPD55x are installed in the personal computer.
	
	[C language: VC++]
	DWORD ret; // function return value
	DWORD count = 2; //max number is 2
	HPCDEVICEINFO HpcDeviceInfo[2]; // 2 HPCI-PPD55x device information
	// area to be set
	ret = hppd553_GetDeviceInfo(
	&count, //pass the address of count.
	&HpcDeviceInfo[0]); //Pass the top address of the array.
	
	[VB6]
	Dim ret As Long
	Dim count As Long
	Dim HpcDeviceInfo(2) As HPCDEVICEINFO
	count = 2
	ret = hppd553_GetDeviceInfo(
	count, 'Pass the address of count.
	HpcDeviceInfo(0)) ' Pass the top address of the array.
	
	[VB. NET]
	Dim ret As Integer
	Dim count As Integer
	Dim HpcDeviceInfo(2) As HPCDEVICEINFO
	count = 2
	ret = hppd553_GetDeviceInfo(
	count, 'Pass the address of count.
	HpcDeviceInfo(0)) ' Pass the top address of the array.

Opens the HPCI-PPD55x board with the specified information and obtains the device ID to distinguish it from others.	HPCI-PPD55x board with the specified information and obtains the device ID to distinguish it from others. vinegar. From now on, this device ID will be the handle for accessing this HPCI-PPD55x.
format	[C language: VC++] DWORD WINAPI hppd553_OpenDeviceID(PHPCDEVICEINFO pHpcDeviceInfo); [VB6] Declare Function hppd553_OpenDeviceID Lib "hippd553.DLL" _ (pHpcDeviceInfo As HPCDEVICEINFO) As Long [VB. NET] Declare Function hppd553_OpenDeviceID Lib "hippd553.DLL" _ (pHpcDeviceInfo As HPCDEVICEINFO) As Integer
Arguments	• PHPCDEVICEINFO pHpcDeviceInfo Pass the address of the area where the information of the device to be opened is set.
Return value	Device ID value INVALID_HANDLE_VALUE (=-1): Open failure See "4.4.3 Device driver error report (P10)". stomach. Other than the above: Successful opening •Higher word: Board ID (Jumper setting value on the board: 0 to 15) •Lower word: Device ID (Number in order of opening: 1 to) [Hint] The upper word "Board ID" can be used as the device ID. In this case, the lower word is forcibly set to "0". VC++ [hDeviceID &= 0xffff0000;] VB [hDeviceID = hDeviceID And &HFFFF0000] Calling
example	It is assumed that two HPCI-PPD55x are installed in the personal computer. Prepare the HPCDEVICEINFO type array HpcDeviceInfo[2] as the device information storage area, and assume that the device information of all boards has already been entered in this by the hppd553_GetDeviceInfo function. [C language: VC++] DWORD hDeviceID[2]; //Device ID acquisition area hDeviceID[0] = hppd553_OpenDeviceID(&HpcDeviceInfo[0]); //First device information hDeviceID[1] = hppd553_OpenDeviceID(&HpcDeviceInfo[1]); //Second device information [VB6] Dim hDeviceID(2) As Long hDeviceID(0) = hppd553_OpenDeviceID(HpcDeviceInfo(0)) 'First device information hDeviceID(1) = hppd553_OpenDeviceID(HpcDeviceInfo(1)) 'Second device information [VB. NET] Dim hDeviceID(2) As Integer hDeviceID(0) = hppd553_OpenDeviceID(HpcDeviceInfo(0)) 'First device information hDeviceID(1) = hppd553_OpenDeviceID(HpcDeviceInfo(1)) 'Second device information

Opens the HPCI-PPD55x specified with the passed information and acquires the device ID to distinguish it from others.	hppd553_OpenDevice() Open device (generating handle) (4) Function vinegar. From now on, this device ID will be the handle for accessing this HPCI-PPD55x .
format	[C language: VC++] DWORD WINAPI hppd553_OpenDevice(PHPCDEVICEINFO pHpcDeviceInfo); [VB6] Declare Function hppd553_OpenDevice Lib "hippd553.DLL" _ (pHpcDeviceInfo As HPCDEVICEINFO) As Long [VB. NET] Declare Function hppd553_OpenDevice Lib "hippd553.DLL" _ (pHpcDeviceInfo As HPCDEVICEINFO) As Integer
Arguments	• PHPCDEVICEINFO pHpcDeviceInfo Pass the address of the area where the information of the device to be opened is set.
Return value	Device ID value INVALID_HANDLE_VALUE (=-1): Open failure See "4.4.3 Device driver error report (P10)". stomach. Other than the above: Successful opening - Upper word: always 0 - Lower word: Device ID (number in order of opening) : 1~)

Calling example	It is assumed that two HPCI-PPD55x are installed in the personal computer.
	Prepare the HPCDEVICEINFO type array HpcDeviceInfo[2] as the device information storage area, and assume that the device information of all boards has already been entered in this by the hppd553_GetDeviceInfo function.
	
	[C language: VC++]
	DWORD hDeviceID[2]; //Device ID acquisition area
	
	hDeviceID[0] = hppd553_OpenDevice(
	&HpcDeviceInfo[0]); //First device information
	hDeviceID[1] = hppd553_OpenDevice(
	&HpcDeviceInfo[1]); //Second device information
	
	[VB6]
	Dim hDeviceID(2) As Long
	
	hDeviceID(0) = hppd553_OpenDevice(
	HpcDeviceInfo(0)) 'First device information hDeviceID(1)
	= hppd553_OpenDevice(
	HpcDeviceInfo(1)) 'Second device information
	
	[VB. NET]
	Dim hDeviceID(2) As Integer
	
	hDeviceID(0) = hppd553_OpenDevice(
	HpcDeviceInfo(0)) 'First device information hDeviceID(1)
	= hppd553_OpenDevice(
	HpcDeviceInfo(1)) 'Second device information

Function	hppd553_CloseDevice() Closed by the passed
Closes the board specified by the passed device ID.	
	After that, this device ID becomes invalid. [C
format	language: VC++] DWORD WINAPI hppd553_CloseDevice(DWORD hDevID); [VB6] Declare Function hppd553_CloseDevice Lib "hippd553.DLL" _ (ByVal hDevID As Long) As Long [VB. NET] Declare Function hppd553_CloseDevice Lib "hippd553.DLL _ "(ByVal hDevID As Long) As Integer
Argument	DWORD hDevID ... Device ID of the board to be closed. Device driver
	Processing result. 0: Success Other than 0: Failed to close. See "4.1.3 Device driver error report (P10)". Calling example [C language: VC++]

	DWORD ret; // function return value
	ret = hppd553_CloseDevice(hDeviceID); // Device ID
	
	[VB6]
	Dim ret As Long
	ret = hppd553_CloseDevice(hDeviceID) 'Device ID
	
	[VB. NET]
	Dim ret As Integer
	ret = hppd553_CloseDevice(hDeviceID) 'Device ID

(6)	hppd553_CmdWrite() Write command buffer
Function	1) Send a command to the command buffer of the axis specified by AxisNo of the HPCI-PPD55x specified by the device ID. data. (2) Write to each register of PPD55x. For details on the register write command, see User's Please see the manual.
format	[C language: VC++] DWORD WINAPI hppd553_CmdWrite(DWORD hDevID, WORD AxisNo, BYTE byData, WORD sw, DWORD dwReg); [VB6] Declare Function hppd553_CmdWrite Lib "hippd553.DLL" _ (ByVal hDevID As Long, ByVal AxisNo As Integer, _ ByVal byData As Byte, ByVal sw As Integer, _ ByVal dw Reg As Long) As Long [VB. NET] Declare Function hppd553_CmdWrite Lib "hippd553.DLL" _ (ByVal hDevID As Integer, ByVal AxisNo As short, _ ByVal byData As Byte, ByVal sw As short, _ ByVal dwReg As Integer) As Integer
Arguments	• DWORD hDevID•••Device ID of the target device. • WORD AxisNo: Axis specification [0: X axis, 1: Y axis, 2: Z axis] • BYTE byData•••Command data "Table 5.1-1 Command data not using write registers: sw = 0", See "Table 5.1-2 Write Register and Command Data: sw = 1" • WORD sw =0: No register writing. ÿ0: Register writing is performed. • DWORD dwReg ••• Write data to register
Return value	Processing result 1: Success 0: Failed --- See "4.4.3 Device driver error report (P10)".

Calling example	[C language: VC++]
	DWORD ret; // function return value
	ret = hppd553_CmdWrite(hDeviceID, //Device ID
	//Specify the X axis
	0, 0x60, // (this is a "soft reset" command)
	// Not a register write.
	0, 0);
	
	[VB6]
	Dim ret As Long 'function return value
	ret = hppd553_CmdWrite(hDeviceID, 'Device ID
	'Specify the X axis
	&H60, (this is a "soft reset" command)
	0, 'Not a register write.
	0)
	
	[VB. NET]
	Dim ret As Integer 'function return value
	ret = hppd553_CmdWrite(hDeviceID, 'Device ID
	0, 'Specify the X axis
	&h60, ' (this is a "soft reset" command)
	'Not a register write.
	0, 0)

Command	Command contents	Command	Command contents
data 0x58	SVON ON	data 0x00	Instantly change to R1 speed
0x48 0x59	SVON OFF	0x01 0x02	Instantly change to R2 speed
	SV RESET ON		Decelerate and change to R1 speed
0x49	SV RESET OFF	0x03	Accelerate and change to R2 speed
0x10 FL constant speed start FL constant speed start pending		0x20	
0x11 FH constant speed start FH constant speed start pending		0x21	
0x13 Fast Start Fast Start Pending		0x23	
0x14 Remaining amount FL start Remaining amount FL start pending		0x24	
0x15 Remaining amount FH start Remaining amount FH start pending		0x25	
0x17 Remaining fast start Remaining fast start pending		0x27	
0x30 Simultaneous start Simultaneous stop		0x28	
0x09 Immediate stop Deceleration stop		0x0a	
0x60	Software reset Emergency stop	0x63	
0x61	UP/DOWN counter reset Pre-register fixed set	0x66	
0x62	Step-out detection deviation counter reset Pre-register fixed reset	0x67	

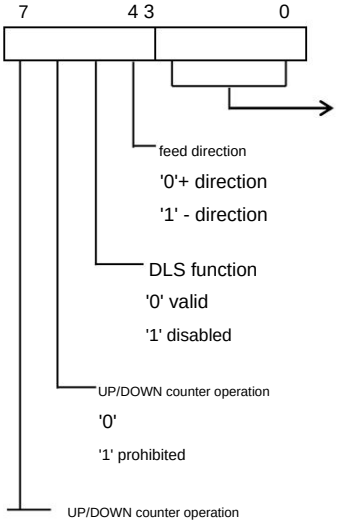
Table 5.1-1 Command data without using write registers: sw = 0

Write register	Command data	Register contents	Number of register bytes
PR0	0xc0 Feed amount		4
PR1	0xc1 FL (PR1 x magnification (PR4)) speed 0xc2 FH (PR2		2
PR2	x magnification (PR4)) speed 0xc3 Acceleration/deceleration		2
PR3	rate 0xc4 Magnification 0xc5 Deceleration point 0xc6		2
PR4	Environment register 1 0xc7 Environment register 2 0xc8		2
PR5	Environment register 3 0xc9 Up/down count value 0xca		3
R6	Comparator 1 data 0xcb Comparator 2 data 0xd1 FL (R1		4
R7	x magnification) speed change during operation 0xd2 FH		4
R8	(R2 x magnification) speed change during operation 0xd3		2
R9	Acceleration/deceleration rate change during operation		4
R10	0xd4 Slow-down point change during operation 0xd8		4
R11	Deceleration rate 0xd9 S-curve section 0xda Deceleration		4
R1	rate change during operation 0xdb S-curve section change		2
R2	during operation Table 5.1-2 Write register and command		2
R3	data:		2
R5			3
PR15			2
PR16			2
R15			2
R16			2

sw = 1

(7) hppd553_DousaWrite() Function	Write operation mode buffer
Writes an operation mode to the operation mode buffer of the axis specified by AxisNo of the HPCI-PPD55x specified by the device ID. Write code data.	
format	[C language: VC++] DWORD WINAPI hppd553_DousaWrite(DWORD hDevID, WORD AxisNo, BYTE byData); [VB6] Declare Function hppd553_DousaWrite Lib "hippd553.DLL" _ (ByVal hDevID As Long, ByVal AxisNo As Integer, ByVal byData As Byte) As Long [VB. NET] Declare Function hppd553_DousaWrite Lib "hippd553.DLL" _ (ByVal hDevID As Integer, ByVal AxisNo As short, ByVal byData As Byte) As Integer
Arguments	• DWORD hDevID•••Device ID of the target device • WORD AxisNo: Axis specification [0: X axis, 1: Y axis, 2: Z axis] • BYTE byData•••Operating mode data (See “Table 5.1-3 Operating mode data” on the next page.) see)
Return value	Processing result 1: Success 0: Failed --- See "4.4.3 Device driver error report (P10)".

Calling example	[C language: VC++]
	ret; // function return value
	DWORD ret = hppd553_DousaWrite(hDeviceID, // Device ID
	0, //Specify the X axis
	0x20); // (Refer to operation mode data)
	[VB6]
	Dim ret As Long ret 'function return value
	= hppd553_DousaWrite(hDeviceID, 'Device ID
	0, 'Specify the X axis
	&H20) ' (Refer to operation mode data)
	[VB. NET]
	Dim ret As Integer ret = 'function return value
	hppd553_DousaWrite(hDeviceID, 'Device ID
	0, 'Specify the X axis
	&H20) ' (Refer to operation mode data)



b3 - b0	Operation mode
0000 Continuous mode 1	(manual feed)
0001 ÿ 2 0010 Origin mode 1	(handle manual feed)
0011 2	ÿ
0100 Origin search mode	
0101 Origin extraction mode	
0110 1 pulse output mode	
0111 Timer mode (dwell)	
1000 Preset mode 1 (direction and feed amount)	
1001	ÿ 2 (Relative feed)
1010	ÿ 3 (absolute value feed)
1011 Floating origin return mode	
1100 Preset mode 4 (handle synchronous feed)	

Table 5.1-3 Operation mode data

(8) hppd553_SeigyoWrite() Function	Control mode buffer write
Writes a control mode to the control mode buffer of the axis specified by AxisNo of the HPCI-PPD55x specified by the device ID. Write code data.	
format	[C language: VC++] DWORD WINAPI hppd553_SeigyoWrite(DWORD hDevID, WORD AxisNo, BYTE byData); [VB6] Declare Function hppd553_SeigyoWrite Lib "hippd553.DLL" _ (ByVal hDevID As Long, ByVal AxisNo As Integer, ByVal byData As Byte) As Long [VB. NET] Declare Function hppd553_SeigyoWrite Lib "hippd553.DLL" _ (ByVal hDevID As Integer, ByVal AxisNo As Short, ByVal byData As Byte) As Integer
Arguments	• DWORD hDevID•••Device ID of the target device. • WORD AxisNo: Axis specification [0: X axis, 1: Y axis, 2: Z axis] • BYTE byData•••Control mode data (Refer to "Table 5.1-4 Control mode data")
Return value Processing result	1: Success 0: Failed --- See "4.4.3 Device driver error report (P10)".

Calling example [C language: VC++]	DWORD ret; // function return value
	ret = hppd553_SeigyoWrite(hDeviceID, // device ID
	//Specify the Y-axis
	1, 0x04); // (Refer to control mode data)
	[VB6]
	Dim ret As Long 'function return value
	ret = hppd553_SeigyoWrite(hDeviceID, 'Device ID
	1, 'Specify the Y-axis
	&H4) ' (see control mode data)
	[VB. NET]
	Dim ret As Integer 'function return value
	ret = hppd553_SeigyoWrite(hDeviceID, 'Device ID
	1, 'Specify the Y-axis
	&H4) ' (see control mode data)

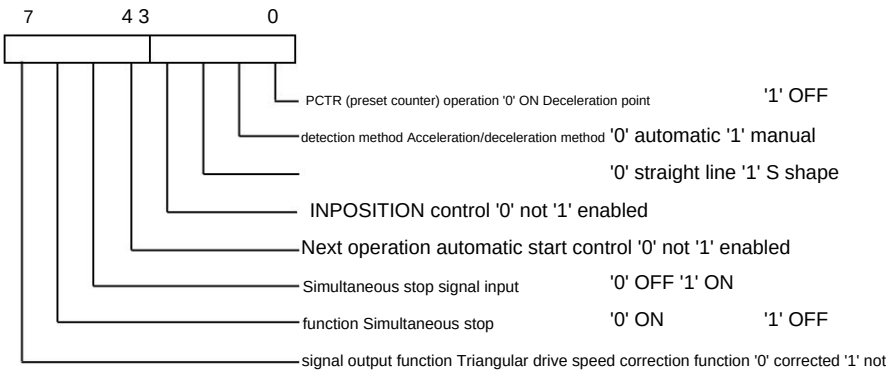


Table 5.1-4 Control mode data

hppd553_status1Read() Function Reads the Read Status 1	axis specified by AxisNo and stores it in the area specified by the device ID and specifies stored in the 1-byte area. [C language Text
format	VC++] DWORD WINAPI hppd553_status1Read(DWORD hDevID, WORD AxisNo, BYTE* bySts1); [VB6] Declare Function hppd553_status1Read Lib "hippd553.DLL" _ (ByVal hDevID As Long, ByVal AxisNo As Integer, ByRef bySts1 As Byte) As Long [VB. NET] Declare Function hppd553_status1Read Lib "hippd553.DLL"_ (ByVal hDevID As Integer, ByVal AxisNo As Short, ByRef bySts1 As Byte) As Integer
Arguments	• DWORD hDevID•••Device ID of the target device. • WORD AxisNo•••Axis specification [0: X axis, 1: Y axis, 2: Z axis] • BYTE* bySts1•••Storage location of read status 1 (Refer to "Table 5.1-5 Status 1")
Return value Processing	result 1: Success 0: Failed --- See "4.4.3 Device driver error report (P10)". Calling example [C language: VC++] DWORD ret; // function return value BYTE sts1; ret = hppd553_status1Read(hDeviceID, //device ID 1, // specify Y axis storage, // destination address [VB6] Dim ret As Long 'function return value Dim sts1 As Byte ret = hppd553_status1Read(hDeviceID, 'Device ID 'Specify sts1) 'Address of sts1, destination [VB. NET] Dim ret As Integer 'function return value Dim sts1 As Byte ret = hppd553_status1Read(hDeviceID, 'Device ID 'Specify sts1) 'Address of sts1, destination

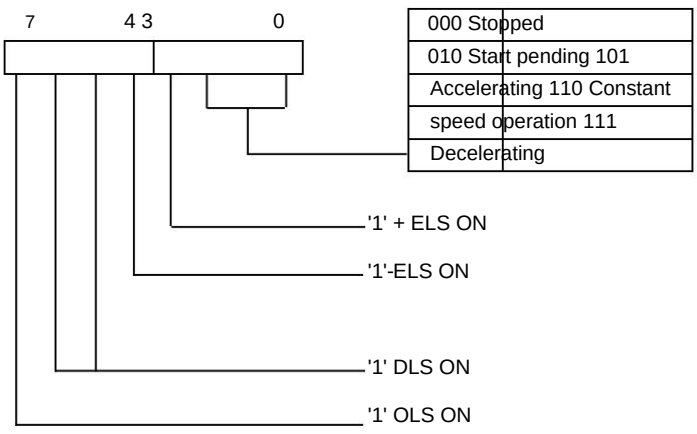


Table 5.1-5 Status 1

(10) hppd553_status2Read() Function	Read status 2
Reads the status 2 of the axis specified by AxisNo of the HPCI-PPD55x specified by the device ID. stored in the 1-byte area.	
format	[C language: VC++] DWORD WINAPI hppd553_status2Read(DWORD hDevID, WORD AxisNo, BYTE* bySts2); [VB6] Declare Function hppd553_status2Read Lib "hippd553.DLL"_ (ByVal hDevID As Long, ByVal AxisNo As Integer, ByRef bySts2 As Byte) As Long [VB. NET] Declare Function hppd553_status2Read Lib "hippd553.DLL"_ (ByVal hDevID As Intege, ByVal AxisNo As Short, ByRef bySts2 As Byte) As Integer
Arguments	• DWORD hDevID•••Device ID of the target device. • WORD AxisNo: Axis specification [0: X axis, 1: Y axis, 2: Z axis] • BYTE* bySts2•••Storage destination of read status 1 (Refer to "Table 5.1-6 Status 2")
Return value	Processing result 1: Success 0: Failed --- See "4.4.3 Device driver error report (P10)".

Calling example	[C language: VC++]
	DWORD ret; // function return value
	BYTE sts2;
	ret = hppd553_status1Read(hDeviceID, //Device ID
	1, //Specify the Y axis
	&sts2); //Store address
	
	[VB6]
	Dim ret As Long 'function return value
	Dim sts2 As Byte
	ret = hppd553_status1Read(hDeviceID, 'Device ID
	'Specify the Y-axis
	1, sts2) 'Address of storage destination
	
	[VB. NET]
	Dim ret As Integer 'function return value
	Dim sts2 As Byte
	ret = hppd553_status1Read(hDeviceID, 'Device ID
	'Specify the Y-axis
	1, sts2) 'Address of storage destination

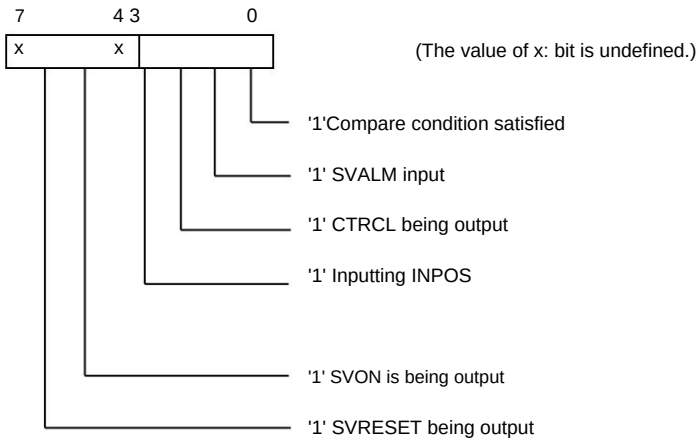


Table 5.1-6 Status 2

(11)	hppd553_intstatusRead() Read interrupt status
Function	Reads the interrupt status of the axis specified by AxisNo of the HPCI-PPD55x specified by the device ID. stored in the specified 1-byte area.
format	[C language: VC++] DWORD WINAPI hppd553_intstatusRead(DWORD hDevID, WORD AxisNo, BYTE* byIntsts); [VB6] Declare Function hppd553_intstatusRead Lib "hippd553.DLL"_ (ByVal hDevID As Long, ByVal AxisNo As Integer, ByRef byIntsts As Byte) As Long [VB. NET] Declare Function hppd553_intstatusRead Lib "hippd553.DLL"_ (ByVal hDevID As Integer, ByVal AxisNo As Short, ByRef byIntsts As Byte) As Integer
Arguments	• DWORD hDevID•••Device ID of the target device. • WORD AxisNo•••Axis specification [0: X axis, 1: Y axis, 2: Z axis] • BYTE* byIntsts•••The address of the 1-byte area where the read interrupt status data is stored. (Refer to "Table 5.1-7 Interrupt Status")
Return value	Processing result 1: Success 0: Failed --- See "4.4.3 Device driver error report (P10)".

Calling example	[C language: VC++]
	DWORD ret; // function return value
	BYTE intsts;
	ret = hppd553_intstatusRead(hDeviceID, //Device ID
	1, //Specify the Y axis
	&intsts); //Store address
	[VB6]
	Dim ret As Long 'function return value
	Dim intsts As Byte
	ret = hppd553_intstatusRead(hDeviceID, 'Device ID
	1, 'Specify the Y-axis
	intsts) 'Address of storage destination
	[VB. NET]
	Dim ret As Integer 'function return value
	Dim intsts As Byte
	ret = hppd553_intstatusRead(hDeviceID, 'Device ID
	1, 'Specify the Y-axis
	intsts) 'Address of storage destination

Status 0x00	cause	Enable interrupt (mask bit)
No interrupt factor		
0x01 Stop by writing deceleration stop command (0Ah) R8 b0='1'		
0x02 Stop due to completion of positioning operation R8 b1='1'		
0x03 Stop due to completion of home return (origin search) operation R8 b2 = '1'		
0x04 Stop due to completion of home exit operation R8 b2='1'		
0x05 Stop by writing immediate stop command (09h) R8 b0 = '1'		
0x06 Stop by simultaneous stop command Control mode b5 = '1'		
0x07 - stop by ELS signal ON		
0x08 +Stop by ELS signal ON		
0x09 - Deceleration stop by DLS signal ON 0x0a		R6 b29='1'
+ Deceleration stop by DLS signal ON 0x0b Stop		R6 b29='1'
by SVALM signal ON		
0x0c Stop due to step-out detection		R8 b12-b8 0 ▪
0x11 Current position encoder input error 0x12		R6 b27='1'
Manual pulse encoder input error 0x13 Step-out		Continuous mode 2, preset mode 4
detection encoder input error 0x14 Next operation		R8 b12-b8 0 ▪
start (pre-register changeable) 0x15 Deceleration start		R8 b14='1'
0x16 Compare -Changed condition from false to true		R8 b3='1'
		R8 b13='1'

Table 5.1-7 Interrupt status

(12)	hppd553_regRead() register read
Function	Reads the register contents of the axis specified by AxisNo of the HPCI-PPD55x specified by the device ID, and stored in the specified 4-byte area. Set "value of register read command" to byData. See the user's manual for details of register contents.
format	[C language: VC++] DWORD WINAPI hppd553_regRead(DWORD hDevID, WORD AxisNo, BYTE byData, DWORD* dwReg); [VB6] Declare Function hppd553_regRead Lib "hippd553.DLL" _ (ByVal hDevID As Long, ByVal AxisNo As Integer, ByVal byData As Byte, ByRef dwReg As Long) As Long [VB. NET] Declare Function hppd553_regRead Lib "hippd553.DLL" _ (ByVal hDevID As Integer, ByVal AxisNo As Short, ByVal byData As Byte, ByRef dwReg As Integer) As Integer
Arguments	• DWORD hDevID•••Device ID of the target device. • WORD AxisNo: Axis specification [0: X axis, 1: Y axis, 2: Z axis] • BYTE byData --- Register read command value (see "Table 5.1-8 Register read command") see) • DWORD* dwReg ••• Address of 4-byte area where read data is stored
Return value	Processing result 1: Success 0: Failed --- See "4.4.3 Device driver error report (P10)".

Calling example	[C language: VC++]
	DWORD ret; // function return value
	DWORD regstr;
	ret = hppd553_regRead(hDeviceID, //Device ID
	//Specify the X axis
	0, 0x80, // read the contents of R0
	&regstr); //Store destination address
	
	[VB6]
	Dim ret As Long 'function return value
	Dim regstr As Long
	ret = hppd553_regRead(hDeviceID, _ 0, _ 'Device ID
	&h80, _ regstr) 'Specify the X axis
	Read the contents of 'R0
	'Address of storage destination
	
	[VB. NET]
	Dim ret As Integer 'function return value
	Dim regstr As Integer
	ret = hppd553_regRead(hDeviceID, _ 0, _ 'Device ID
	&h80, _ regstr) 'Specify the X axis
	Read the contents of 'R0
	'Address of storage destination

Read register	Register read command	Register contents
R0	0x80 Feed amount	
R1	0x81 FL (R1 x magnification) speed	
R2	0x82 FH (R2 x magnification) speed	
R3	0x83 Acceleration/deceleration rate	
R4	0x84 Magnification	
R5	0x85 deceleration point	
R6	0x86 environment register 1	
R7	0x87 environment register 2	
R8	0x88 environment register 3	
R9	0x89 up/down counter	
R10	0x8a Comparator 1 data	
R11	0x8b Comparator 2 data	
R12	0x8c counter monitor	
R13	0x8d Command monitor 1	
R14	0x8e Command monitor 2	
PR0	0x90 Preset amount for next operation	
PR1	0x91 FL for next operation (PR1 x magnification) speed	
PR2	0x92 FH (PR2 x magnification) speed for next operation	
PR3	0x93 Acceleration/deceleration rate for next operation	
PR4	0x94 Magnification for next operation	
PCTR	0x95 Preset counter count value	
SCTR	0x96 Deceleration point counter count value	
PR5	0x97 Slow down pointer for next operation	
PR15	0x98 Deceleration rate for next operation	
PR16	0x99 S-shaped section for next operation	
R15	0x9a deceleration rate	
R16	0x9b S-shaped section	

Refer to Table 5.1-8 Register read command

(13)

port read hppd553_ioswRead() hppd553_ioswWrite() General-purpose I/O number switching

number switching port write function Reads the general-purpose I/O number

switching port of the HPCI-PPD55x specified by the device ID... Reads the general-purpose I/O number switching port , is stored in the specified area. General-purpose input/output number switching port write: Writes the specified data to the general-purpose input/output number switching port. [C language: VC++]

format

```
DWORD WINAPI hppd553_ioswRead ( DWORD hDevID, BYTE* byIoSw );
DWORD WINAPI hppd553_ioswWrite( DWORD hDevID, BYTE byData );
[ VB6 ]
Declare Function hppd553_ioswRead Lib "hippd553.DLL" _
    (ByVal hDevID As Long, ByRef byIoSw As Byte) As Long Declare
Function hppd553_ioswWrite Lib "hippd553.DLL" _ (ByVal hDevID
    As Long, ByVal byData As Byte) As Long [VB. NET]

Declare Function hppd553_ioswRead Lib "hippd553.DLL" _
    ( ByVal hDevID As Integer, ByRef byIoSw As Byte ) As Integer
Declare Function hppd553_ioswWrite Lib "hippd553.DLL" _
    ( ByVal hDevID As Integer, ByVal byData As Byte ) As Integer Argument
```

• DWORD hDevID •••Device ID of the target device. • BYTE* byIoSw •••Address of 1-byte area where read data is stored • BYTE byData •••Write data to general-purpose I/O switching port

7	4 3	0
x x x x x x		
0 0 0 0 0 0	0	

Read (x: bit value is undefined.) Write

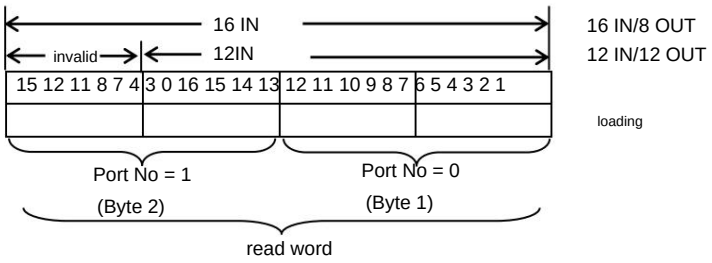
0: 16 IN/8 OUT, 1: 12 IN/12 OUT

Return value Processing

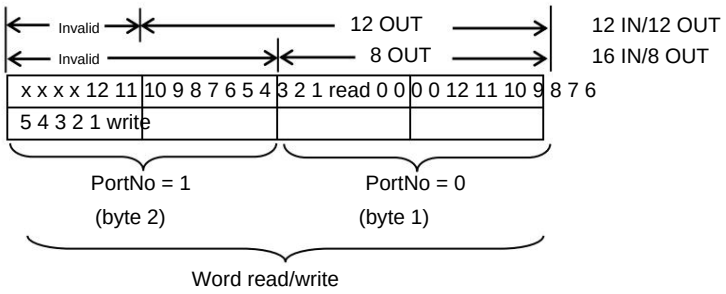
result 1:

Success 0: Failed --- See "4.4.3 Device driver error report (P10)". Calling example [C language: VC++]

```
DWORD ret; // function return value
BYTE IoSw;
    ret = hppd553_ioswRead( hDeviceID, // Device ID
                                        &IoSw ); // Storage destination for the read value of the general-purpose I/O
    number switching port ret = hppd553_ioswWrite( hDeviceID, // Device ID 0x80 );
                                                    //12IN/12OUT
[VB6]
Dim ret As Long
Dim IoSw As Byte
    ret = hppd553_ioswRead ( hDeviceID, 'Device ID
                                        IoSw ) 'Read value of general-purpose I/O switching
    port ret = hppd553_ioswWrite( hDeviceID, 'Device ID
                                        &H80 )          '12IN/12OUT
[VB. NET]
Dim ret As Integer
Dim IoSw As Byte
    ret = hppd553_ioswRead ( hDeviceID, 'Device ID
                                        IoSw ) 'Read value of general-purpose I/O switching
    port ret = hppd553_ioswWrite( hDeviceID, 'Device ID
                                        &H80 )          '12IN/12OUT
```

(14)	hppd553_inpReadB() General purpose input port byte read hppd553_inpReadW() General purpose input port word read
Function	HPCI-PPD55x specified by device ID General-purpose input port byte reading: Reads the specified input port and stores it in the specified area. General-purpose input port word reading: Reads input ports 1 and 2 and stores them in the specified area.
format	[C language: VC++] DWORD WINAPI hppd553_inpReadB(DWORD hDevID, WORD PortNo, BYTE* byInp); DWORD WINAPI hppd553_inpReadW(DWORD hDevID, WORD* wInp); [VB6] Declare Function hppd553_inpReadB Lib "hippd553.DLL" (ByVal hDevID As Long, ByVal PortNo As Integer, ByRef byInp As Byte) As Long Declare Function hppd553_inpReadW Lib "hippd553.DLL" (ByVal hDevID As Long, ByRef wInp As Integer) As Long [VB. NET] Declare Function hppd553_inpReadB Lib "hippd553.DLL" (ByVal hDevID As Integer, ByVal PortNo As Short, ByRef byInp As Byte) As Integer Declare Function hppd553_inpReadW Lib "hippd553.DLL" (ByVal hDevID As Integer, ByRef wInp As Short) As Integer
Arguments	• DWORD hDevID•••Device ID of the target device. • WORD PortNo: Specify port number [0: input port 1, 1: input port 2] • BYTE* byInp ••• Address of 1-byte area where read data is stored • WORD* wInp ••• Address of 2-byte area where read data is stored  (Note) (1) The numbers in the frames are general-purpose input signals (DI) 1 to 16. (2) The input value of invalid bits is undefined.
Return value	Processing result 1: Success 0: Failed --- See "4.4.3 Device driver error report (P10)".

Calling example	[C language: VC++]
	DWORD ret; // function return value
	BYTE by Inp;
	WORD wInp;
	ret = hppd553_inpReadB(hDeviceID, //Device ID
	0, // input port 1
	&byInp); //Store destination address
	ret = hppd553_inpReadW(hDeviceID, //Device ID
	&wInp); //Store address
	[VB6]
	Dim ret As Long 'function return value
	Dim by Inp As Byte
	Dim wInp As Integer
	ret = hppd553_inpReadB(hDeviceID, _ 0, _ 'Device ID
	byInp) ret = 'Input port 1
	'Address of storage destination
	hppd553_inpReadW(hDeviceID, _ wInp) 'Device ID
	'Address of storage destination
	[VB, NET]
	Dim ret As Integer 'function return value
	Dim by Inp As Byte
	Dim wInp As short
	ret = hppd553_inpReadB(hDeviceID, _ 0, _ 'Device ID
	byInp) ret = 'Input port 1
	'Address of storage destination
	hppd553_inpReadW(hDeviceID, _ wInp) 'Device ID
	'Address of storage destination

(15)	port byte read hppd553_outpReadB() hppd553_outpWriteB() General-purpose output port byte write General-purpose hppd553_outpReadW() hppd553_outpWriteW() General-purpose output port word write Function
	General-purpose output port byte reading: Reads the specified output port and stores it in the specified area. General-purpose output port byte write: Writes the specified data to the specified output port. General-purpose output port word reading: Reads output ports 1 and 2 and stores them in the specified area. General-purpose output port word write: Writes specified data to output ports 1 and 2. [C language: VC++]
format	DWORD WINAPI hppd553_outpReadB (DWORD hDevID, WORD PortNo, BYTE* byOutp); DWORD WINAPI hppd553_outpWriteB(DWORD hDevID, WORD PortNo, BYTE byData); DWORD WINAPI hppd553_outpReadW (DWORD hDevID, WORD* wOutp WORD);); [VB6] Declare Function hppd553_outpReadB Lib "hippd553.DLL" (ByVal hDevID As Long, ByVal PortNo As Integer, ByRef byOutp As Byte) As Long Declare Function hppd553_outpWriteB Lib "hippd553.DLL" (ByVal hDevID As Long, ByVal PortNo As Integer, ByVal byData As Byte) As Long Declare Function hppd553_outpReadW Lib "hippd553.DLL" (ByVal hDevID As Long, ByRef wOutp As Integer) As Long Declare Function hppd553_outpWriteW Lib "hippd553.DLL" (ByVal hDevID As Long, ByVal wData As Integer) As Long [VB. NET] Declare Function hppd553_outpReadB Lib "hippd553.DLL" (ByVal hDevID As Integer, ByVal PortNo As Short, ByRef byOutp As Byte) As Integer Declare Function hppd553_outpWriteB Lib "hippd553.DLL" (ByVal hDevID As Integer, ByVal PortNo As Short, ByVal byData As Byte) As Integer Declare Function hppd553_outpReadW Lib "hippd553.DLL" (ByVal hDevID As Integer, ByRef wOutp As Short) As Integer Declare Function hppd553_outpWriteW Lib "hippd553.DLL" (ByVal hDevID As Integer, ByVal wData As Short) As Integer
Argument	• DWORD hDevID•••Device ID of the target device. • WORD PortNo: Port number specification [0: Output port 1, 1: Output port 2] • BYTE* byOutp: 1-byte area address where read data is stored • WORD* wOutp: Read data • BYTE byData•••Write data to output port • WORD wData •••Write data to output port  (Note) (1) The numbers in the frames are the general-purpose output signals (DO) 1 to 12. (2) The bit value of an invalid output port is set to 0, and is undefined for input.
Return value	Processing result 1: Success 0: Failed --- See "4.4.3 Device driver error report (P10)".

Calling example [C language: VC++]

```
DWORD     ret;          // function return value
BYTE by Outp;
WORD wOutp;
   ret = hppd553_outpReadB ( hDeviceID, //Device ID
                             0, // output port 1
                             &byOutp );
   ret = hppd553_outpWriteB( hDeviceID, //Device ID
                             0, // output port 1
                             0x01 );          //Turn ON only OUT1
   ret = hppd553_outpReadW ( hDeviceID, //Device ID
                             &wOutp );
   ret = hppd553_outpWriteW( hDeviceID, //Device ID
                             0x0001 ); //Turn ON only OUT1 of output port 1
```

[VB6]

```
Dim ret       As Long              'function return value
Dim by Outp   As bytes
Dim w Outp    As Integer
   ret = hppd553_outpReadB ( hDeviceID, _ 'Device ID
                             0, _ 'Output port 1
                             byOutp )          'Address of storage destination
   ret = hppd553_outpWriteB( hDeviceID, _ 'Device ID
                             0, _ 'Output port 1
                             &H1 )             'Turn ON only OUT1
   ret = hppd553_outpReadW ( hDeviceID, _ 'Device ID
                             wOutp ) 'Address of storage destination
   ret = hppd553_outpWriteW( hDeviceID,_ 'Device ID
                             &H1 )             'Turn ON only OUT1 of output port 1
```

[VB. NET]

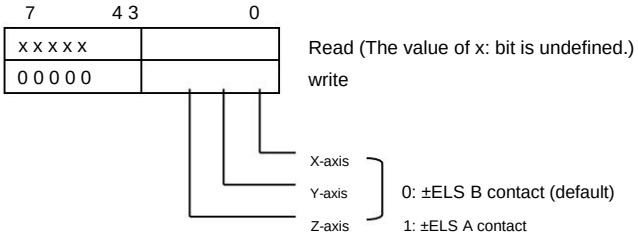
```
Dim ret       As Integer           'function return value
Dim by Outp   As bytes
Dim w Outp    As Short
   ret = hppd553_outpReadB ( hDeviceID, _ 'Device ID
                             0, _ 'Output port 1
                             byOutp ) 'Address of storage destination
   ret = hppd553_outpWriteB( hDeviceID, _ 'Device ID
                             0, _ 'Output port 1
                             &H1 )             'Turn ON only OUT1
   ret = hppd553_outpReadW ( hDeviceID, _ 'Device ID
                             wOutp ) 'Address of storage destination
   ret = hppd553_outpWriteW( hDeviceID,_ 'Device ID
                             &H1 )             'Turn ON only OUT1 of output port 1
```

(16) hppd553_ElstRead() Read ELS polarity selection port
hppd553_ElstWrite() ELS polarity select port write

Function HPCI-PPD55x specified by device ID
Read ELS polarity selection port: Reads the ELS polarity selection port and stores it in the specified area.
Write to ELS polarity selection port: Writes specified data to the ELS polarity selection port.

format
[C language: VC++]
DWORD WINAPI hppd553_ElstRead (DWORD hDevID, BYTE* byElsData);
DWORD WINAPI hppd553_ElstWrite(DWORD hDevID, BYTE byData);
[VB6]
Declare Function hppd553_ElstRead Lib "hippd553.DLL"
(ByVal hDevID As Long, ByRef byElsData As Byte) As Long
Declare Function hppd553_ElstWrite Lib "hippd553.DLL"
(ByVal hDevID As Long, ByVal byData As Byte) As Long
[VB. NET]
Declare Function hppd553_ElstRead Lib "hippd553.DLL"
(ByVal hDevID As Integer, ByRef byElsData As Byte) As Integer
Declare Function hppd553_ElstWrite Lib "hippd553.DLL"
(ByVal hDevID As Integer, ByVal byData As Byte) As Integer

Arguments • DWORD hDevID •••Device ID of the target device.
• BYTE* byElsData•••The address of the 1-byte area where the read data is stored.
• BYTE byData •••Write data to ELS polarity selection port



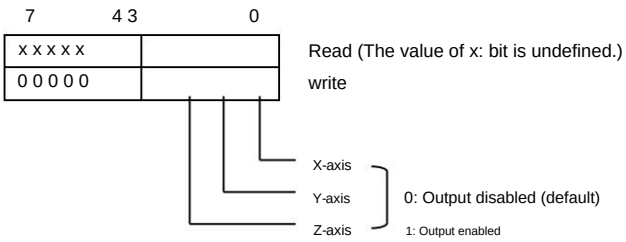
Return value Processing result
1: Success
0: Failed --- See "4.4.3 Device driver error report (P10)".

Calling example [C language: VC++]

```
DWORD       ret;                // function return value
BYTE        ElsData;
   ret = hppd553_ElstRead( hDeviceID, //Device ID
                           &ElsData ); //storage destination address
   ret = hppd553_ElstWrite( hDeviceID, //Device ID
                           0x01 ); // X-axis contact A, Y/Z-axis contact B

[VB6]
 Dim ret          As Long          'function return value
 Dim Els Data As Byte
   ret = hppd553_ElstWrite( hDeviceID, _          'Device ID
                           &H1 )                  'X axis A contact, Y/Z axis B contact
   ret = hppd553_ElstRead ( hDeviceID, _          'Device ID
                           ElsData)               'Address of storage destination

[VB. NET]
 Dim ret          As Integer       'function return value
 Dim Els Data As Byte
   ret = hppd553_ElstWrite( hDeviceID, _          'Device ID
                           &H1 )                  'X axis A contact, Y/Z axis B contact
   ret = hppd553_ElstRead ( hDeviceID, _          'Device ID
                           ElsData)               'Address of storage destination
```

(17)	hppd553_CmpcRead() Comparator simultaneous start reading hppd553_CmpcWrite() Comparator simultaneous start write
Function	HPCI-PPD55x specified by device ID Read Comparator Simultaneous Start: Reads the comparator simultaneous start and stores it in the specified area. vinegar. Comparator simultaneous start write: Writes the specified data to the comparator simultaneous start.
format	[C language: VC++] DWORD WINAPI hppd553_CmpcRead (DWORD hDevID, BYTE* byCmpData); DWORD WINAPI hppd553_CmpcWrite(DWORD hDevID, BYTE byData); [VB6] Declare Function hppd553_CmpcRead Lib "hippd553.DLL" (ByVal hDevID As Long, ByRef byCmpData As Byte) As Long Declare Function hppd553_CmpcWrite Lib "hippd553.DLL" (ByVal hDevID As Long, ByVal byData As Byte) As Long [VB. NET] Declare Function hppd553_CmpcRead Lib "hippd553.DLL" (ByVal hDevID As Integer, ByRef byCmpData As Byte) As Integer Declare Function hppd553_CmpcWrite Lib "hippd553.DLL" (ByVal hDevID As Integer, ByVal byData As Byte) As Integer
argument	• DWORD hDevID •••Device ID of the target device. • BYTE* byCmpData•••The address of the 1-byte area where the read data is stored. • BYTE byData •••Write data to comparator simultaneous start 
Return value	Processing result 1: Success 0: Failed --- See "4.4.3 Device driver error report (P10)".

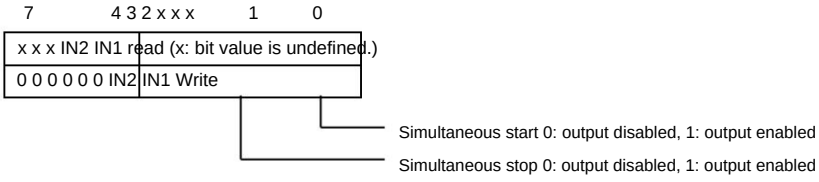
Calling example	[C language: VC++]
	DWORD ret; // function return value
	BYTE CmpData;
	ret = hppd553_CmpcRead(hDeviceID, //Device ID
	&CmpData); //Store destination address
	ret = hppd553_CmpcWrite(hDeviceID, //Device ID
	0x01); //Only the X axis can be output
	
	[VB6]
	Dim ret As Long 'function return value
	Dim CmpData As bytes
	ret = hppd553_CmpcRead(hDeviceID, _ 'Device ID
	CmpData) 'Address of storage destination
	ret = hppd553_CmpcWrite(hDeviceID, _ 'Device ID
	&H1) 'Only the X axis can be output
	
	[VB. NET]
	Dim ret As Integer 'function return value
	Dim CmpData As bytes
	ret = hppd553_CmpcRead(hDeviceID, _ 'Device ID
	CmpData) 'Address of storage destination
	ret = hppd553_CmpcWrite(hDeviceID, _ 'Device ID
	&H1) 'Only the X axis can be output

(18) hppd553_diRead() Simultaneous start/stop reading by general-purpose input
hppd553_diWrite() Simultaneous start/stop writing by general-purpose input

Function HPCI-PPD55x specified by device ID
Simultaneous start by general-purpose input IN1 and simultaneous stop reading by IN2
…Reads the simultaneous start/stop by general-purpose input and stores it in the specified area.
vinegar.
Simultaneous start by general-purpose input IN1 and simultaneous stop writing by IN2
… Writes specified data to simultaneous start/stop by general-purpose input.

format
[C language: VC++]
DWORD WINAPI hppd553_diRead (DWORD hDevID, BYTE* byDi);
DWORD WINAPI hppd553_diWrite(DWORD hDevID, BYTE byData);
[VB6]
Declare Function hppd553_diRead Lib "hippd553.DLL"
(ByVal hDevID As Long, ByRef byDi As Byte) As Long
Declare Function hppd553_diWrite Lib "hippd553.DLL"
(ByVal hDevID As Long, ByVal byData As Byte) As Long
[VB. NET]
Declare Function hppd553_diRead Lib "hippd553.DLL"
(ByVal hDevID As Integer, ByRef byDi As Byte) As Integer
Declare Function hppd553_diWrite Lib "hippd553.DLL"
(ByVal hDevID As Integer, ByVal byData As Byte) As Integer

argument
• DWORD hDevID •••Device ID of the target device.
• BYTE* byDi ••• Address of 1-byte area where read data is stored
• BYTE byData ••• Write data to simultaneous start/stop by general-purpose input



Return value Processing result
1: Success
0: Failed --- See "4.4.3 Device driver error report (P10)".

Calling example [C language: VC++]

```
DWORD ret;                          // function return value
BYTE by Di;
   ret = hppd553_diRead( hDeviceID, //Device ID
                                    &byDi ); //Store destination address
   ret = hppd553_diWrite( hDeviceID, //Device ID
                                    0x01 );          //Simultaneous start setting by IN1
```

[VB6]

```
Dim ret As Long                     'function return value
Dim by Di As Byte
   ret = hppd553_diRead ( hDeviceID, 'Device ID
                                    byDi ) 'Address of storage destination
   ret = hppd553_diWrite( hDeviceID, 'Device ID
                                    &H1 )            ' Simultaneous start setting by IN1
```

[VB. NET]

```
Dim ret As Integer                  'function return value
Dim by Di As Byte
   ret = hppd553_diRead ( hDeviceID, 'Device ID
                                    byDi ) 'Address of storage destination
   ret = hppd553_diWrite( hDeviceID, 'Device ID
                                    &H1 )            ' Simultaneous start setting by IN1
```

6. sample program

A sample program is attached for the purpose of explaining how to use each function of the DLL for the driver I/F. There are three types of sample programs, and they have almost the same screen display and operation. In the explanation of the sample program below, we will use ÿ “C coding”.

ÿ VC++... C coding [spp55300.exe] ÿ VC++... C++ coding [spp55301.exe] ÿ VB 6.0 [spp55302.exe] ÿ VB. NET [spp55303.exe]

When using the sample program, copy it to your hard disk and use it. Each sample executable file (spp55300.exe) can be executed by performing a "double-click" operation.

ÿCautionÿ (1)
The VB sample requires "Microsoft Visual Basic 6.0" to be installed as a development tool.

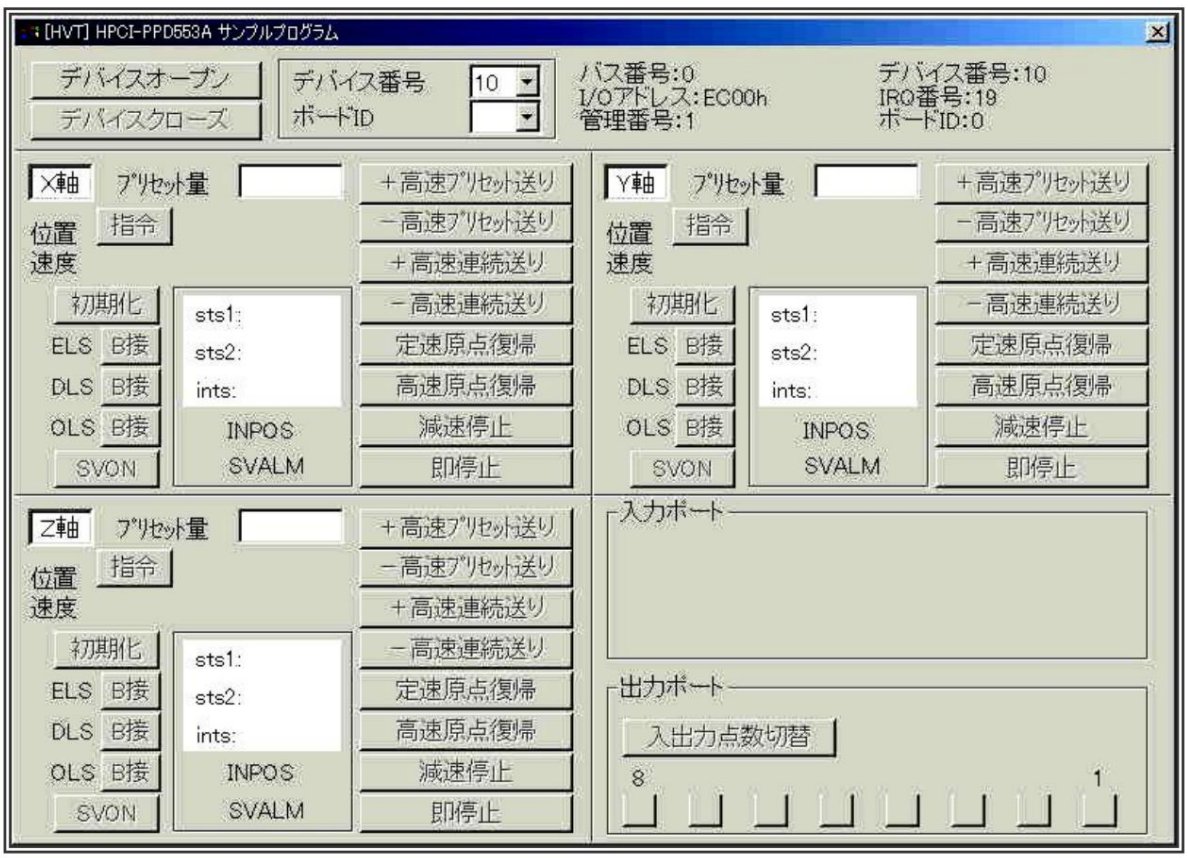
(2) If the following error message is displayed at the start of execution, the sample program will not run.



[Error message display] * The HPCI-PPD55x board is not installed. * The device driver is not installed.

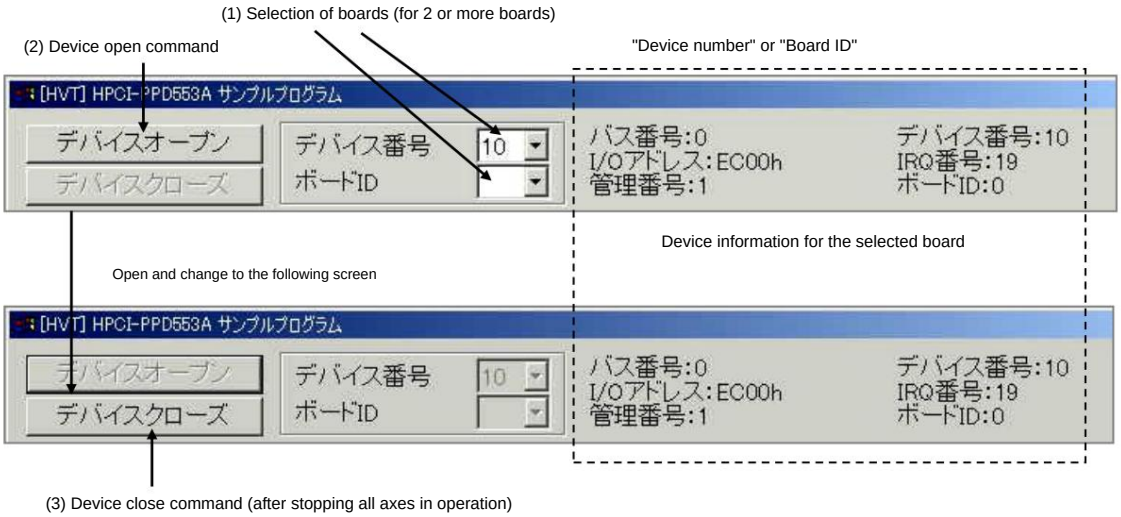
6.1 Operation of the sample program

When the sample program is started and one or more boards (devices) are recognized normally, the following screen will be displayed.



6.1.1 Various operations of the board (device)

In the sample program, the following procedure is used to start and end the motion axis operation on the board.

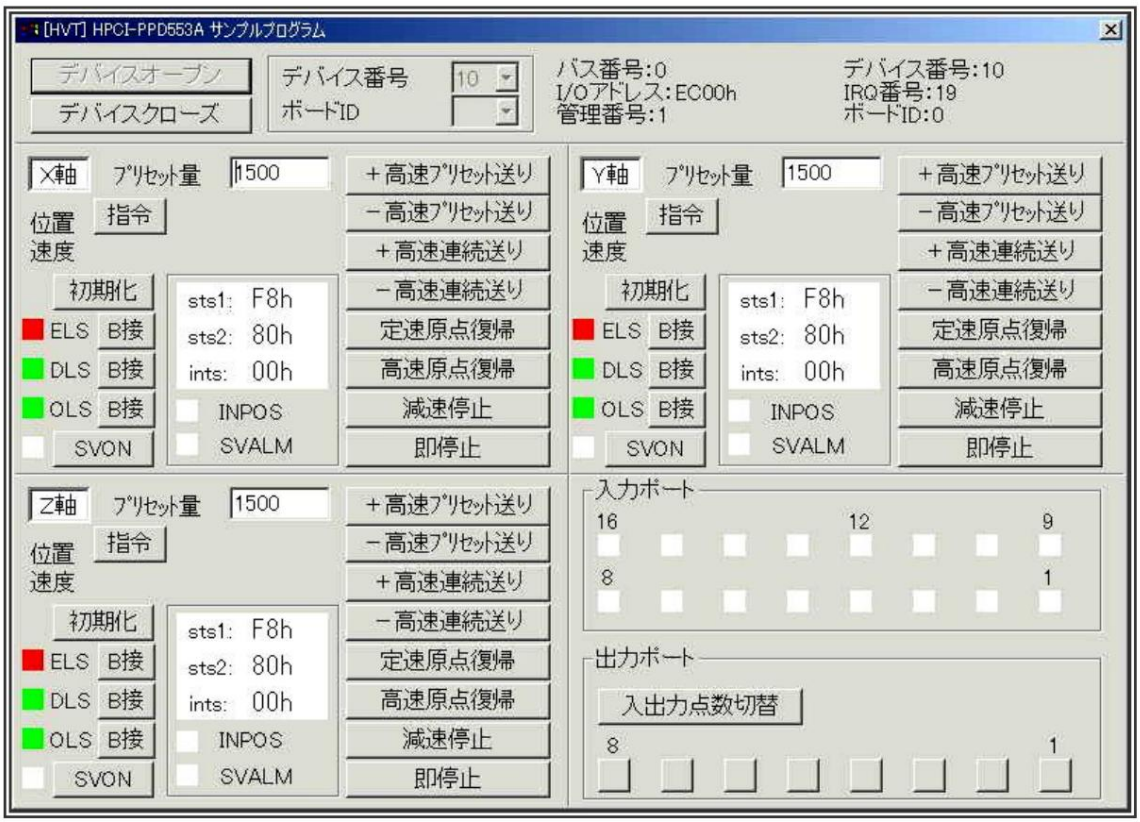


(4) Ending the sample program (after closing the device)



6.1.2 Operation of each axis on the board

When you open the device, the following screen appears.



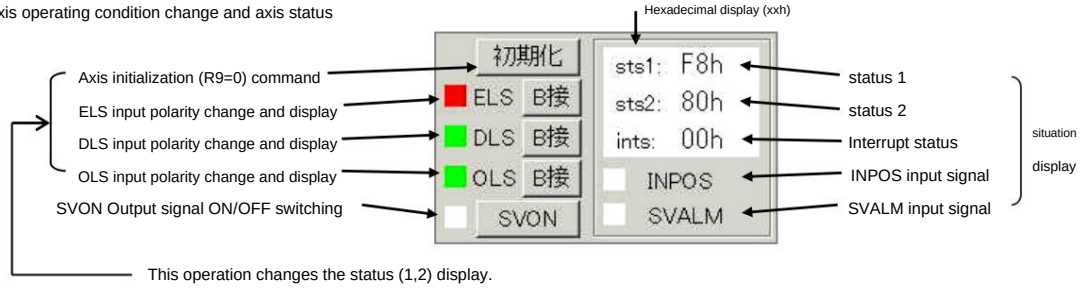
Regardless of how many axes you have on your board, the operations for each individual axis are the same.

In the sample program, the initialization of each axis is partly fixed in the source program.

Therefore, if you want to change the initialization conditions and operate it, you need to change the source program.

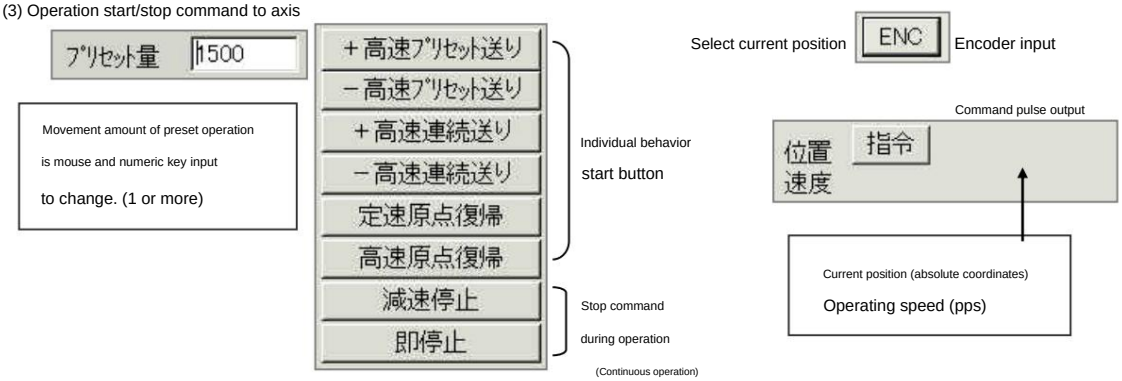
(1) When the number of axes is 2 or less: Buttons for the number of valid axes can be operated.

(2) Axis operating condition change and axis status

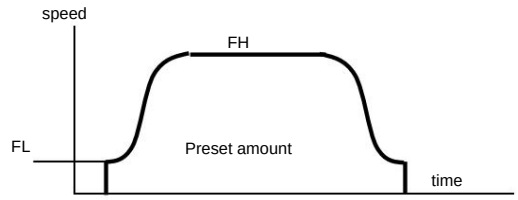


*Signal status display: "White"•OFF, "Red/Green"•ﾊON

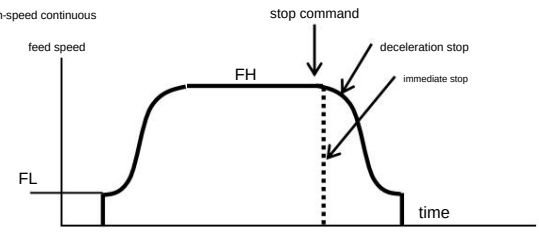
(3) Operation start/stop command to axis



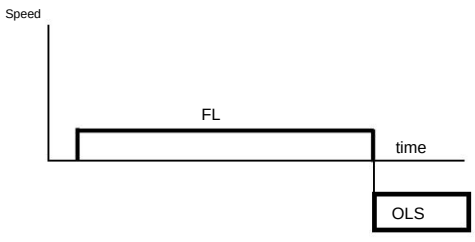
(1) High-speed preset feed



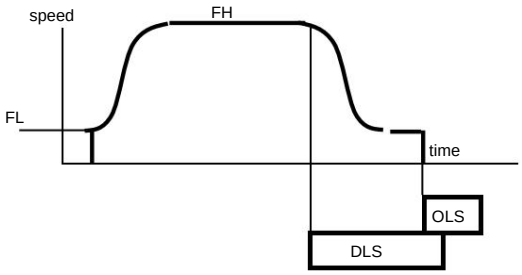
(2) High-speed continuous



(3) Constant speed return to origin (- direction)



(4) High-speed return to origin (- direction)

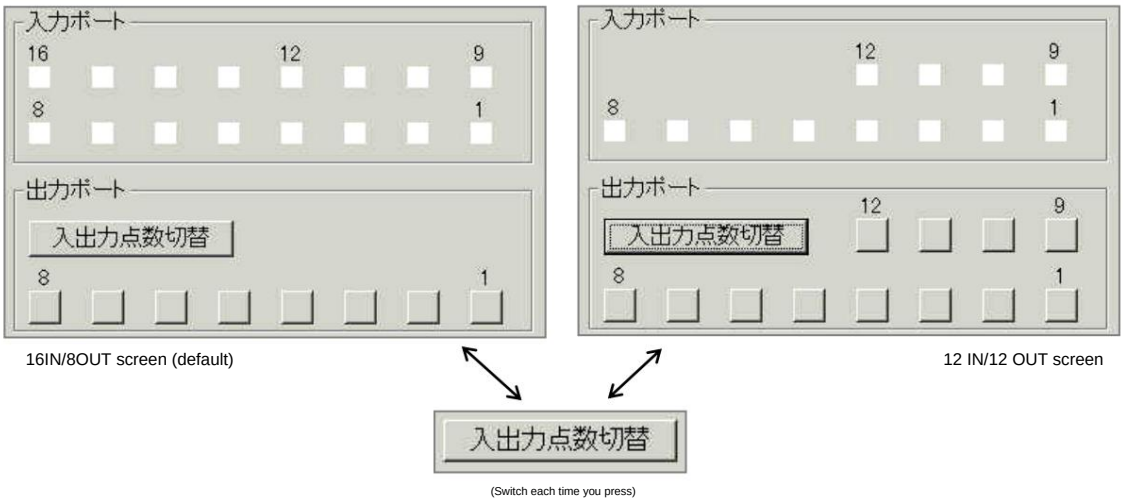


(Note) 1. S-curve is used for acceleration/deceleration.

2. DLS is used for deceleration of high-speed return to origin.

3. The direction of return to origin is fixed to the "-" direction.

(4) Switching between 16IN/8OUT and 12IN/12OUT I/O points



16IN/8OUT screen (default)

12 IN/12 OUT screen

(Switch each time you press)

6.1.3 Setting conditions for axis operation

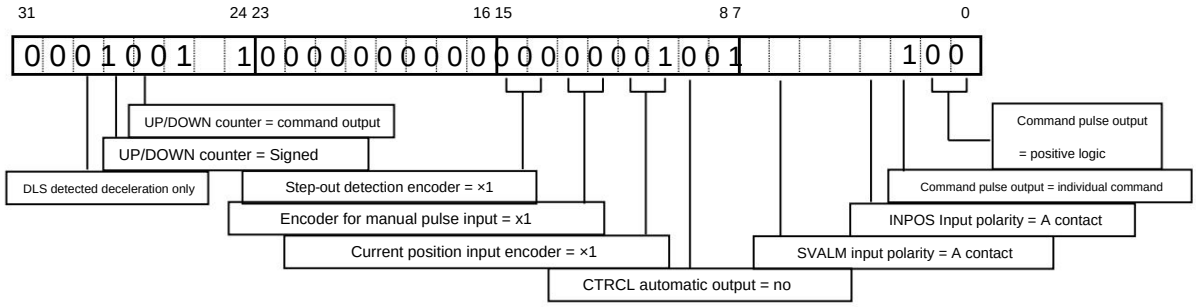
(1) Axis control register initial setting (default value): when the device is open

PCL register setting value consideration		preparation
PR 0	1500	Feed amount (preset movement amount)
PR 1	500	FL speed
PR 2	5000	FH speed
PR 3	545	Acceleration/deceleration rate (1 second) [(2457600 * 1sec)/(FH - FL) - 1]
PR 4	299	Speed multiplier = 1pps [= 300 / (PR4 + 1)]
PR 5	0	Deceleration point (Set "deceleration point detection = automatic" in control mode)
R6	0x1300004c Environment register 1	
R7	0x000f0c40 Environment register 2	
R8	0x0000 Environment register 3	
R9	0	Up/down counter (current position)
R10	0	Comparator 1 (not used)
R11	0	Comparator 2 (not used)
PR15	0	Deceleration rate (Set value = 0, acceleration rate = deceleration rate)
R16	0	S-curve acceleration/deceleration/S-curve section

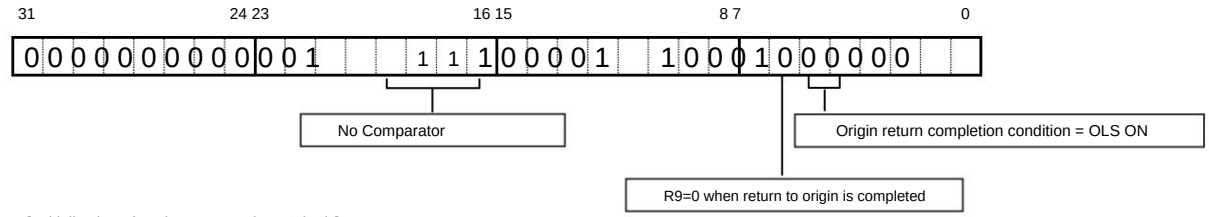
Table 6.1-1 Register initial setting for axis control

Next, the detailed contents of the environment registers 1 to 3 are described.•••The reduced characters "1, 0" are fixed bit data.

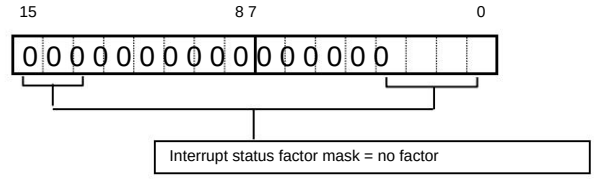
[Initialization of environment register 1 (R6)]



[Initialization of environment register 2 (R7)]

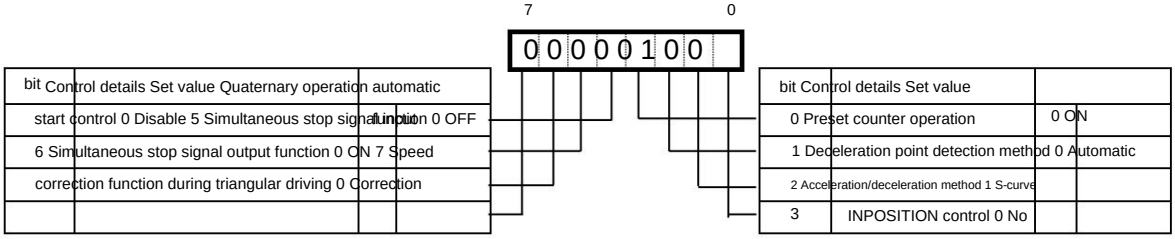


[Initialization of environment register 3 (R8)]



(2) Control mode buffer

Writing to the control mode buffer is performed when the device is opened.



6.2 Changes to the sample program

The four sample programs are
Microsoft Visual C++ 6.0 and Microsoft Visual Basic (6.0/.NET)
using.
When modifying the sample program, make sure the above development environment is installed on your computer.
Being there is a prerequisite.

Each directory (holder) contains an "executable file (.exe)" to create a new executable.
Contains all files for creating function files.

File Type	Development environment/coding category			
	VC++ (C)	VC++(C+	VB6	VB.NET
Executable	spp55300.exe	+) spp55301.exe	spp55302.exe	spp55303.exe
File Project File (Solution File)	spp55300.dsw spp55300.dsp spp55300.opt	spp55301.dsw spp55300.dsp spp55300.opt	spp55302.vbp	spp55303.sln spp55303.suo spp55303.vbproj spp55303.vbproj.user
Driver I/F file	hippd553.dll hippd553.lib hippd553.h	hippd553.dll hippd553.lib hippd553.h	hippd553.dll	hippd553.dll
source program	Other than above	Other than those above	Other than those above	Other than those above

6.2.1 Project file

(1) When changing the VB6 version sample program
Select spp55302.vbp for the project file.
Microsoft Visual Basic 6.0 must be installed.

(2) When changing the VC++(C) version sample program
Select spp55300.dsw for the project workspace.
Microsoft Visual C++ 6.0 must be installed.

(3) When changing the VC++ (C++) version sample program
Select spp55301.dsw for the project workspace.
Microsoft Visual C++ 6.0 must be installed.

(4) When changing the VB.NET version sample program
Select spp55303.sln for the project file.
Microsoft Visual Basic .NET must be installed.

(5) When the version of the development environment is different
If you are using Microsoft Visual Basic 5.0, Microsoft Visual C++ 5.0, etc.,
The project file cannot be used as it is.

In this case, create a new project and copy the sample program to this project.
Add various files to configure.

Note that the functions used in the sample program may not be available depending on the type of development language.
there is.
Please delete the source program before use.

[Example] When building the "VC++(C++)" sample with Microsoft Visual C++ 5.0
stdafx.h ••• Make one line of #include <afxdtctl.h> a comment.

6.2.2 Functions that make up the sample program

The functions that make up the sample program are roughly classified as follows.

Category	No	Function name	Function processing (button correspondence)
	1	WinMain()	Windows main function
	2	WndProc()	Called from Windows to retrieve messages from the message queue be handed over
Screen	3	FormLoad1() Surface	center window, get handle
	4	FormLoad3() Table	5 Processing after obtaining device information
FormUnload()	6	CmbJoho_Click()	close the window
7 BtnOpen2_Click()	Processing	8	Device information selection & display (click combo box)
BtnClose1_Click()	Processing	9	Clicking the device open button
BtnClose3_Click()	Function	10	Device close button click (immediately before closing)
BtnEls2_Click()	Number	11	Clicking the device close button (immediately after closing)
BtnDls2_Click()			Clicking the ELS (Contact A/B) button reverses the ELS input polarity.
			Reversal of DLS input polarity by clicking the DLS (A contact/B contact) button
	12	BtnOls2_Click()	Click the OLS (A contact/B contact) button to invert the OLS input polarity.
	13	BtnEnc2_Click()	Click the encoder input/command output button to read and change the current position
	14	BtnIoSw1_Click()	Switch IN/OUT display by clicking input/output points switching button
	15	ErrMsg()	Error message output (end of sample program)
	16	FormLoad2()	Get device information
	17	BtnOpen1_Click()	Initialize the board by clicking the device open button
	18	Initial()	Axis initialization on the board (for one axis)
	19	BtnClose2_Click()	20 Clicking the device close button stops the axis and closes the device.
BtnPPre_Click() |			ÿPreset operation starts by clicking the high-speed preset feed button
	21	BtnMPre_Click() control	- Predetermined operation starts by clicking the high-speed preset feed button
22 BtnPRen_Click() control		23	ÿPredetermined operation starts by clicking the high-speed continuous feed button
BtnMRen_Click() control		24	- Predetermined operation starts by clicking the high-speed continuous feed button
BtnTGen_Click()			Clicking the constant-speed return-to-origin button starts the specified operation.
•	25	BtnKGen_Click() axis	Predetermined operation starts by clicking the high-speed return-to-origin button
26 BtnGStop_Click() control		27	Issue a deceleration stop command to the axis by clicking the deceleration stop button
BtnSStop_Click() control		28	Issue an immediate stop command to the axis by clicking the immediate stop button
BtnIni_Click()	29	BtnSvOn_Click()	Clear "up-down counter (R9)" by clicking the initialization button
function	30	BtnEls1_Click() number	Clicking the SVON (ON/OFF) button reverses the output signal.
31 BtnDls1_Click()			Input signal reversal command by clicking the ELS (A contact/B contact) button
			Input signal reversal command by clicking the DLS (A contact/B contact) button
	32	BtnOls1_Click()	Input signal reversal command by clicking OLS (A contact/B contact) button
	33	BtnEnc1_Click()	Click the encoder input/command output button to switch current position input
	34	BtnIoSw2_Click()	Switch between 16IN/8OUT and 12IN/12OUT by clicking the number of input/output points switching button
	35	BtnOut_Click()	Output signal reversal command by clicking the output port (ON/OFF) button
	36	TmrSokudo()	Current speed & current position display (moving axis: every 0.1 seconds)
	37	TmrSts()	Read status (during operation: every 0.1 seconds)
	38	TmrInOut()	I/O port display (axis during operation: every 0.1 seconds)

(Note) 1. The table above is for the VC++(C) version.

The "VB version" is almost the same.

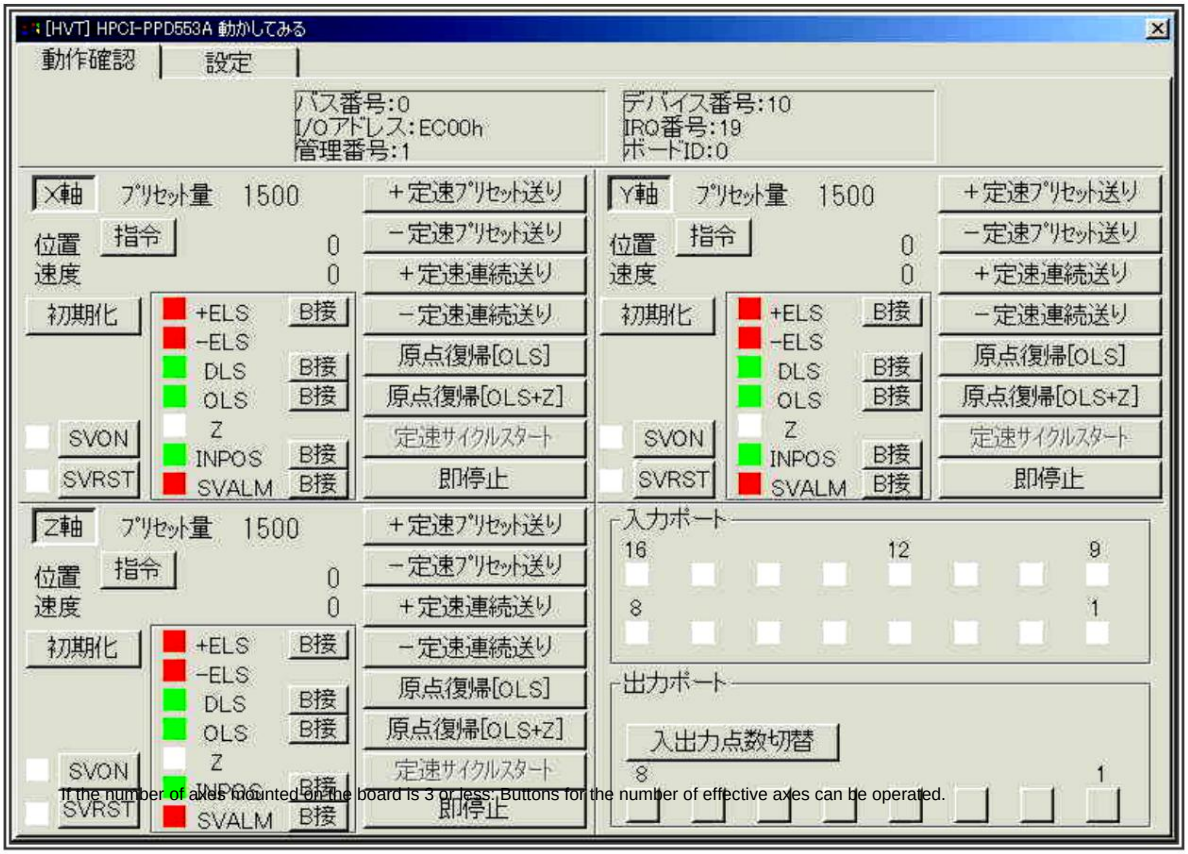
In the case of "VC++ (C++) version", we are trying to make the same function structure as much as possible.

7. "Try it" program

The "try to move" program allows you to check the minimum operation on the display by simply attaching the board to your computer.
It's soft. Execute "\test\tpp55300.exe" in the attached software floppy disk.

7.1 Operation confirmation screen of “Try it”

The following screen will be displayed when you run the "Try it" program.



If the number of axes mounted on the board is 3 or less, buttons for the number of effective axes can be operated.

7.1.1 Displaying device information

The device information of the currently selected board is displayed in the following area.



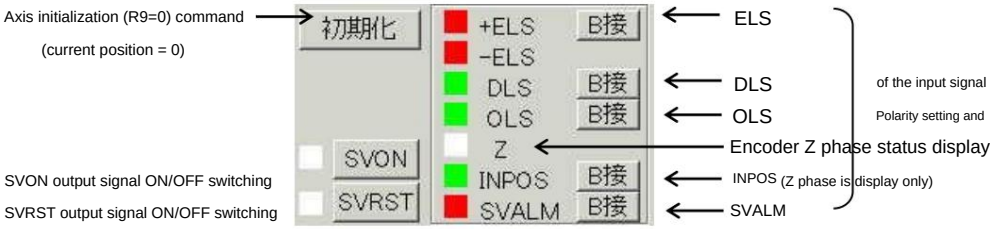
(Note) 1. Control number — It is "-1" in Windows9X.
2. Board ID — This is the jumper setting value on the board. A board without a jumper will be "15".

7.1.2 Individual axis display and motion command

Regardless of how many axes you have on your board, the operations for each individual axis are the same.
In the sample program, the initialization of each axis is partly fixed in the source program.
Therefore, if you want to change the initialization conditions and operate it, you need to change the source program.

(1) When the number of axes is 3 or less: Buttons for the number of valid axes can be operated.

(2) Axis operating condition change and axis status

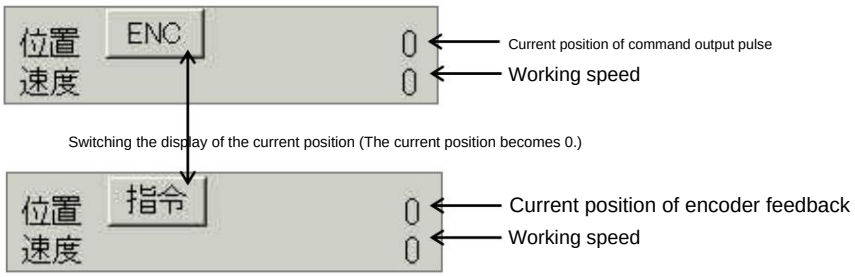


*Signal status display: "White"•OFF, "Red/Green"•ON

(3) Current axis position/operation speed display

The current position and operating speed of each axis are updated every 0.1 seconds.

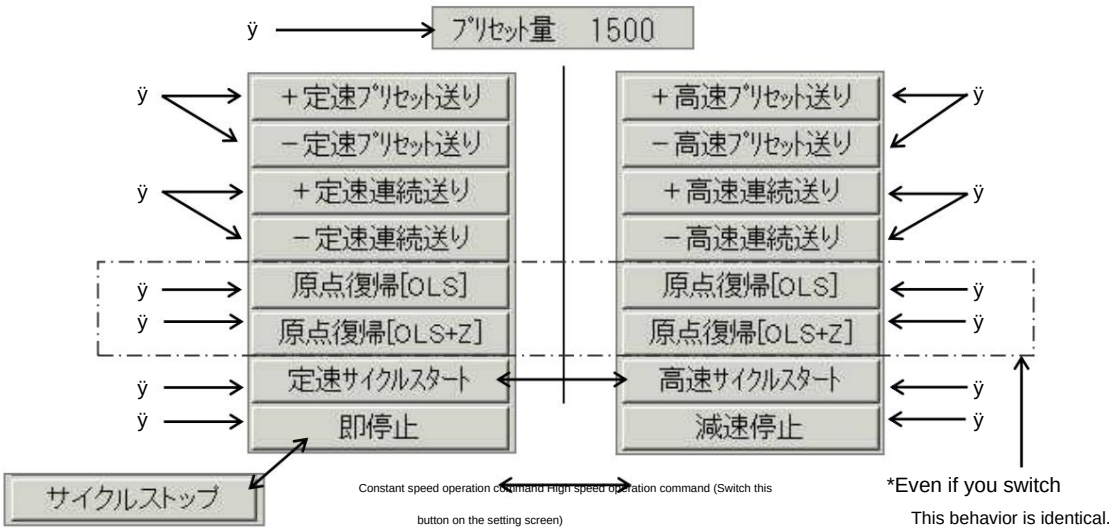
For the current position, the display of "command output pulse" and "encoder feedback" can be selected.



* Encoder feedback (magnification = x1)

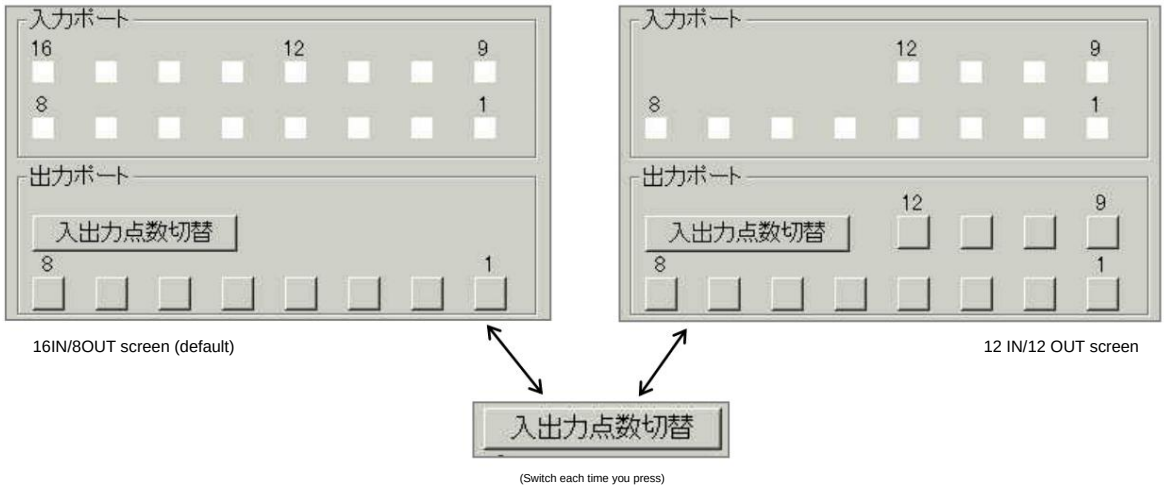
(4) Operation start/stop command to axis

There are 1 to 10 movements for each axis, and 11 preset amounts are 4 types of movement amounts of 1, 2, 7, and 8.



ÿStopping during cycle operation

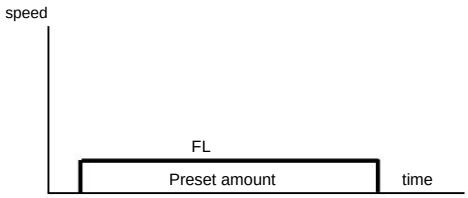
(5) Switching between 16IN/8OUT and 12IN/12OUT I/O points



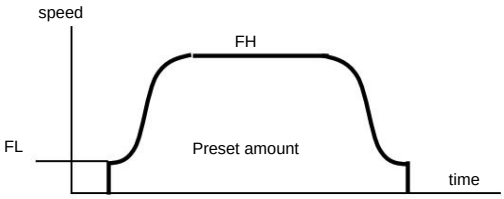
(6) Commands to axes and details of operation

The contents of the operation of the "try to move" program are as shown in the figure below.

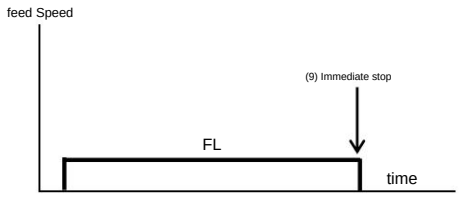
(1) Constant speed preset feed



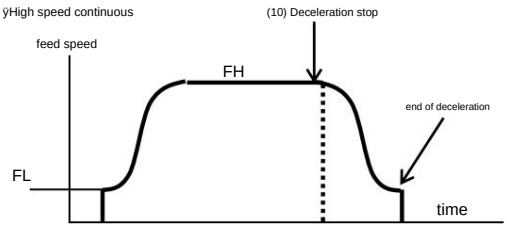
(2) High-speed preset feed



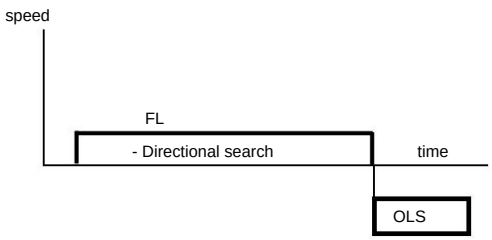
ÿConstant continuous



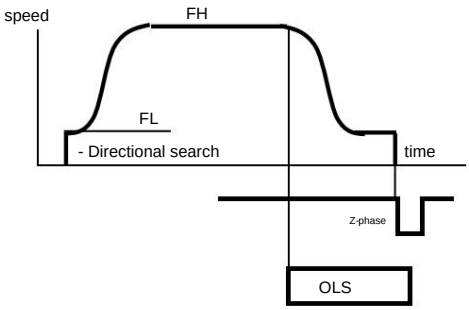
ÿHigh speed continuous



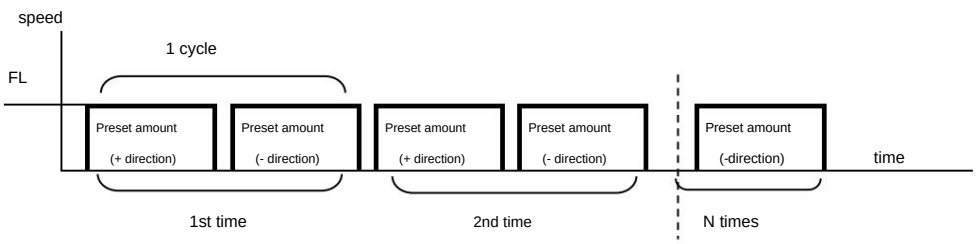
(5) Origin return [OLS]



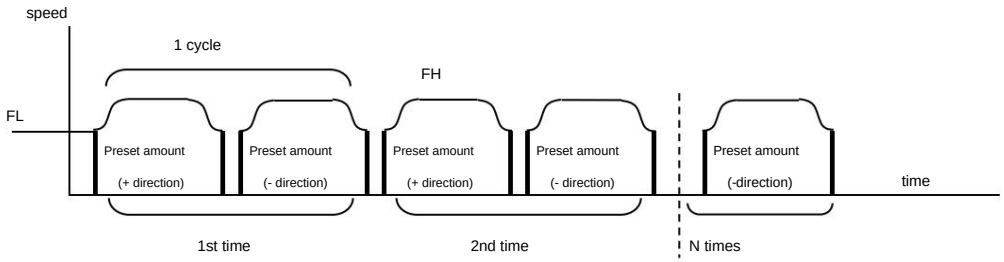
ÿReturn to origin [OLS+Z]



ÿConstant-speed cycle (reciprocating operation of constant-speed preset)



ÿ High-speed cycle (reciprocating motion of high-speed preset)



(Note) 1. S-curve is used for acceleration/deceleration.

2. The direction of return to origin is fixed to the "-" direction.

3. Origin return [OLS] is constant speed and origin signal = OLS.

4. Origin return [OLS + Z] is high speed, and origin signal = OLS deceleration + Z phase once.

5. Elements related to operation (preset amount, speed, etc.) can be changed on the "setting screen".

6. For "cycle" operation, specify the number of times on the "setting screen".

7. In "Cycle" operation, the "Automatic start of next operation" function enables continuous operation.

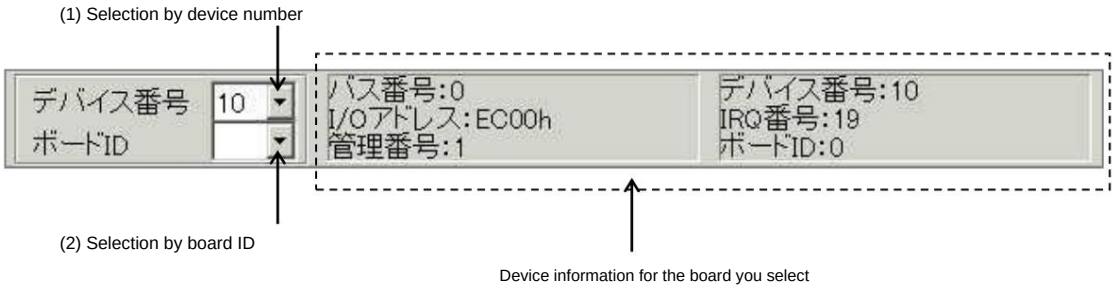
7.2 Setting screen for “Try moving”

Stop all axes on the "Operation check" screen and select "Set" to display the following screen.



7.2.1 Board selection and device information

If two or more boards are installed, select "Move" by "Device number" or "Board ID".
Specify the "Try it" board.



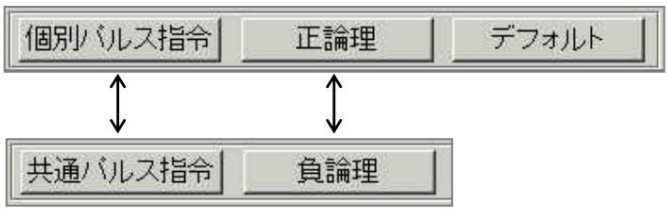
7.2.2 Changeable Axis Operating Conditions

Operating conditions can be set individually for all operable axes.

(1) Setting items and PCL register

The setting items on the setting screen and the relationship between the setting values and the PCL register are as follows.

[Setting the command pulse output] When connecting a motor, select the appropriate connection method for the motor driver. Please.

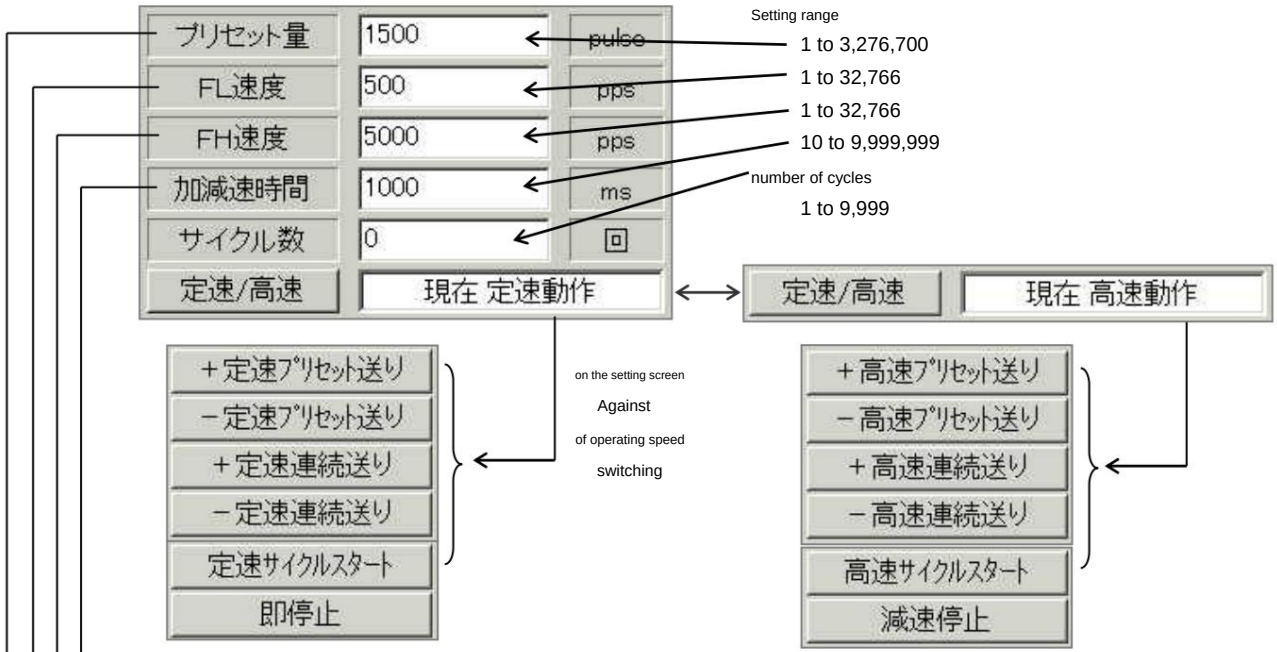


*Default button

"Individual pulse command" and

Return to "positive logic".

[Travel amount, speed, cycle, operation mode]



Setting range

1 to 3,276,700

1 to 32,766

1 to 32,766

10 to 9,999,999

number of cycles

1 to 9,999

on the setting screen

Against

of operating speed

switching

PCL register setting value		preparation thought
PR 0 1500		Feed amount (preset movement amount)
PR 1 500		FL speed
PR 2 5000		FH speed
PR 3	545	Acceleration/deceleration rate (1 second) [(2457600 * 1sec)/(FH - FL) - 1]
PR 4	299	Speed multiplier = 1pps [= 300 / (PR4 + 1)]
PR 5	0	Deceleration point (set "deceleration point detection = automatic" in control mode)
R6	0x13000004 Environment register 1	
R7	0x000f0c40 Environment register 2 See the next page for details of each.	
R8	0x0000 Environment register 3	
R9	0	Up/down counter (current position)
R10	0	Comparator 1 (not used)
R11	0	Comparator 2 (not used)
PR15	0	Deceleration rate (Set value = 0, acceleration rate = deceleration rate)
R16	0	S-curve acceleration/deceleration/S-curve section

* Default values are used except for associated registers and environment register 1, and the device is open.

Initialized at time.

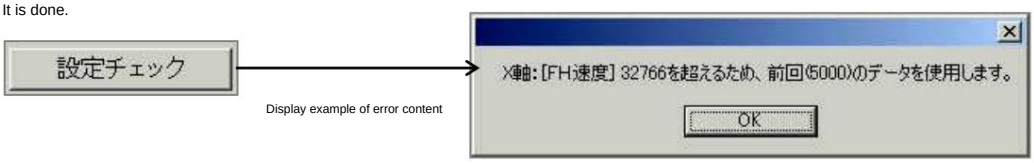
(2) Confirmation operation of set value

The value entered numerically on the screen is confirmed by pressing the "Setting check" button.

If the set value is out of range, an error message is displayed.

If you return to the "Operation check" screen without performing this button operation, the setting check will be automatically performed.

It is done.

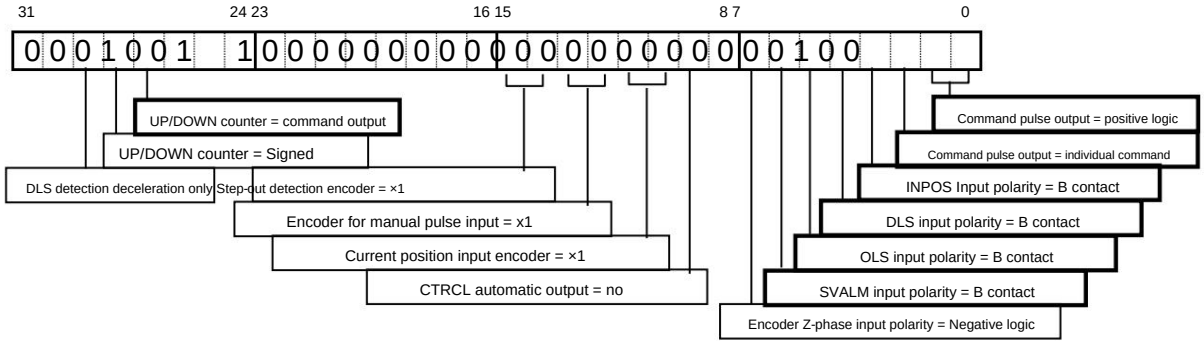


(3) Environment register settings

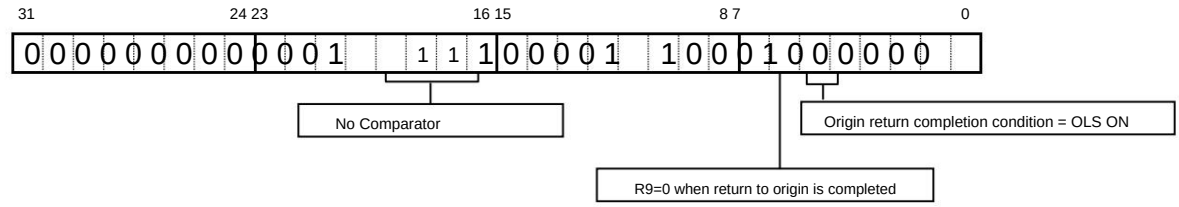
The setting contents of the three types of environment registers are as follows.

on individual items The content indicated by is changed by the operation on the screen.

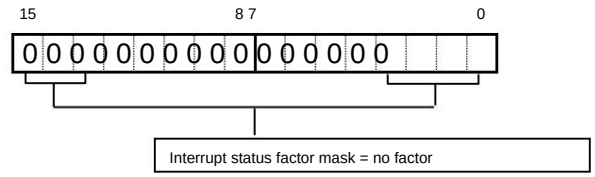
[Set value of environment register 1 (R6)]



[Set value of environment register 2 (R7)]

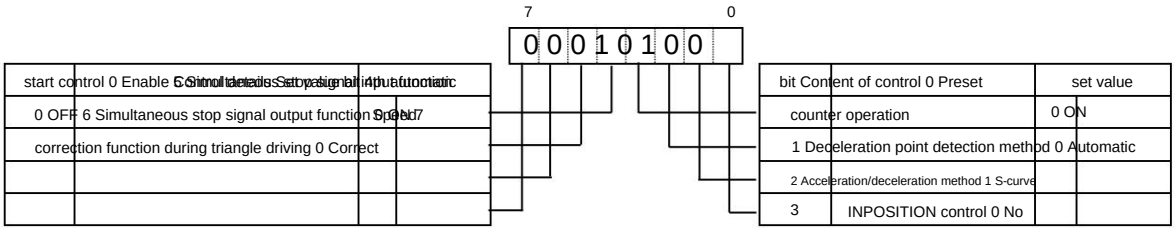


[Set value of environment register 3 (R8)]



(4) Control mode buffer

Writing to the control mode 0 mode buffer is performed when the device is opened.



* "Automatic start of next operation" is a setting for "Cycle operation".

Appendix A Extract of PCL5014 Registers

1. Use classification of PCL5014 registers

Figure 4.3-1 systematically illustrates the register group.

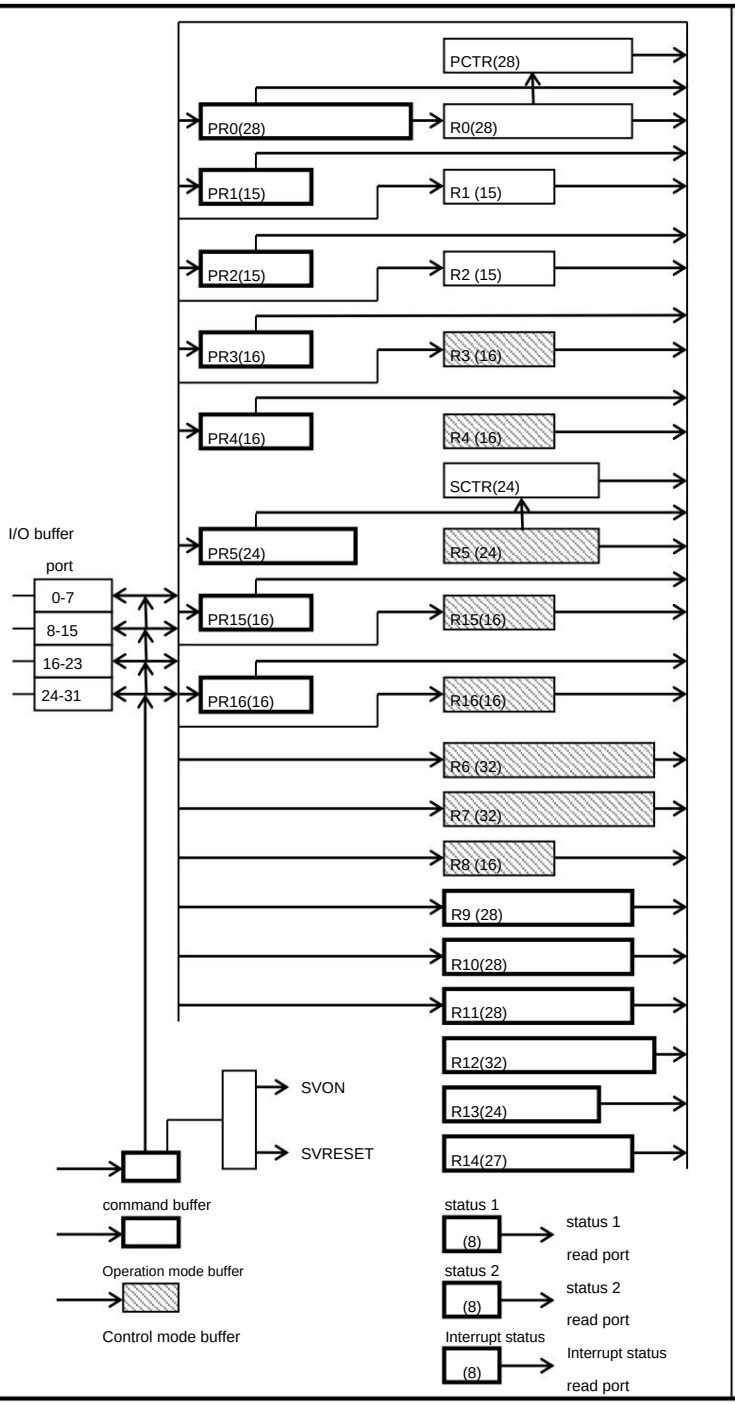
The use of setting and reading registers is roughly classified as follows. (Distinguished by frame lines in the figure)

(1) Registers that, in principle, need only be set once during initialization

(2) Registers that are set during execution or registers that are read and used as necessary

ÿ Other registers





PCTR: Preset counter

PR0: Pre-register R0

R0: Feed amount register

PR1: FL speed pre-register

R1: FL speed register

PR2: FH speed pre-register

R2: FH speed register

PR3: Acceleration/deceleration rate pre-register

R3: Acceleration/deceleration rate register

PR4: Magnification pre-register

R4: Magnification register

SCTR: Deceleration point counter

PR5: Deceleration point pre-register

R5: Deceleration point register

PR15: Deceleration rate pre-register

R15: Deceleration rate register

PR16: S-section pre-register

R16: S-shaped section register

R6: Environment register 1

R7: ÿ 2

R8: ÿ 3

R9: UP/DOWN counter

R10: Comparator value 1 register

R11: ÿ 2 register

R12: Counter monitor

R13: Command monitor 1

R14: Command monitor 2

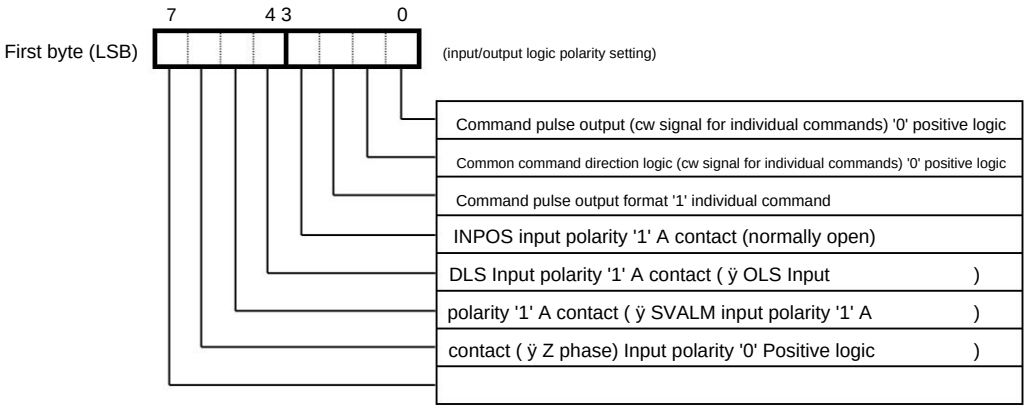
Status 1

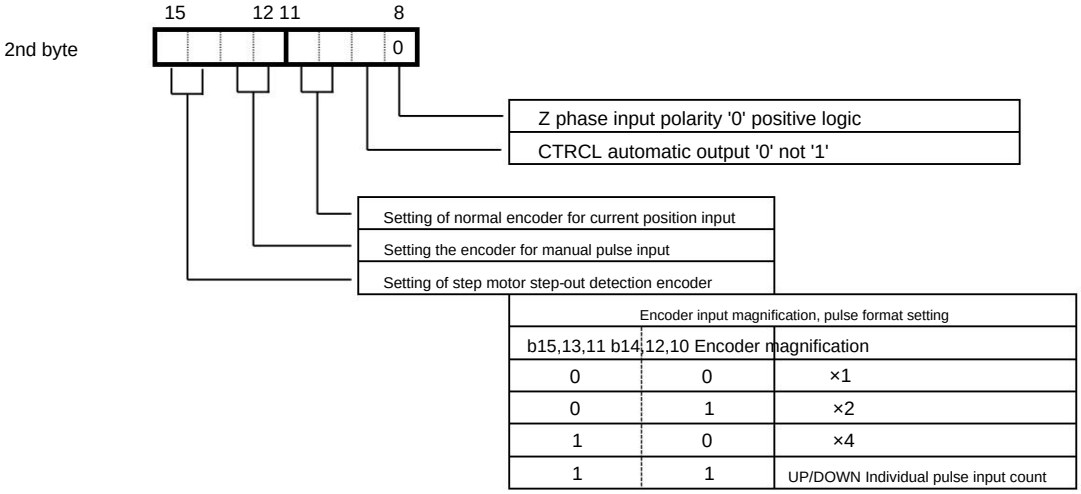
status 2

Interrupt status

2. Individual register

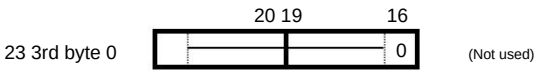
2.1 Environment Register 1 (R6)

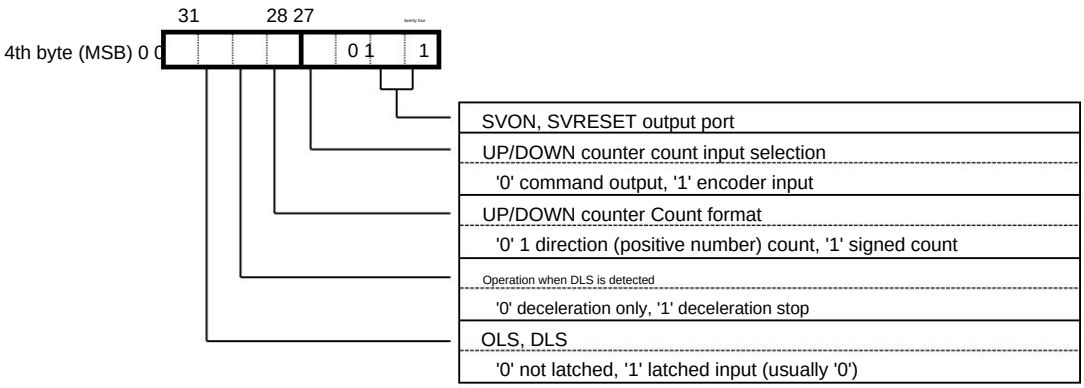




Encoder input magnification, pulse format setting

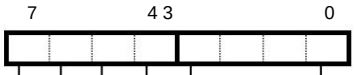
b15,13,11	b14,12,10	Encoder magnification
0	0	×1
0	1	×2
1	0	×4
1	1	UP/DOWN Individual pulse input count





2.2 Environment Register 2 (R7)

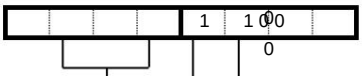
First byte (LSB)



Encoder Z-phase count setting when returning to origin
0 (1st time) to 15 (16th time)

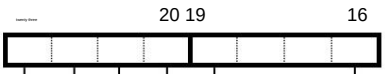
Origin return completion condition
00 Stop with OLS ON
01 Stop at Z-phase count after OLS ON
10 After OLS ON, start deceleration and count Z phase
11 Prohibited
UP/DOWN counter (R9) when origin return is completed
Do not clear '0', clear '1'
Machine lock '1' without '0'

second byte 0



Enable next start command delay after CTRCL output
Enable direction change pulse output delay when using common command output format
Idling pulse number setting 0 to 7

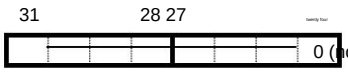
3rd byte 0



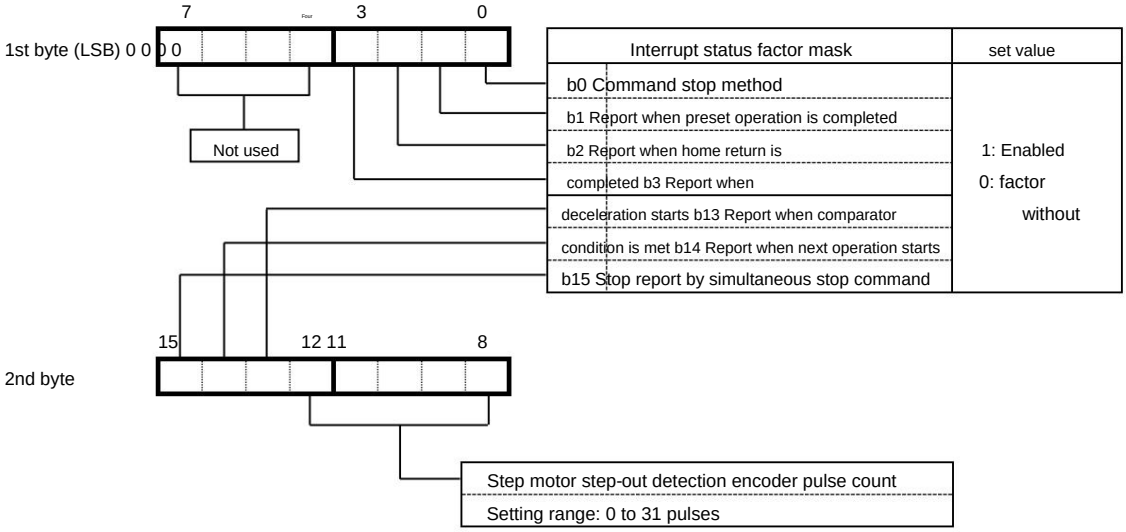
Operation selection for the command pulse when the comparator condition is met
00 do nothing
01 Command pulse output stop
10 Internal operation continues but pulse is not output
11 Set R2 (FH) and R3 (acceleration/deceleration rate) to PR2 and PR3 (speed change)
Comparison counter selection '0' UP/DOWN counter (R9), '1' preset counter (PCTR)
Not used

Comparator condition selection	
condition b19-b16 0000 R10> (counter) Comparison	result
0001 R10=(counter) 0010 R10<(counter) 0100 R11>(counter)	(1) Status 2 b0 = '1' Confirmation of condition fulfillment can.
0101 R11=(counter) 0110 R11<(counter) 1000 R10>(counter) or R11<(counter) 1001	(2) '=' condition matches with a counter change from change immediately. Interrupt in this case use status
R10<(counter) >(counter) 1010 R10=(counter) or R11=(counter) 1011 R10>(counter) or R11>(counter) 1100	
R10<(counter) or R11<(counter) 1101 R10<(counter)<R11	(3) Comparator non-use Set to "1111"
1110 R10>(counter)>R11	

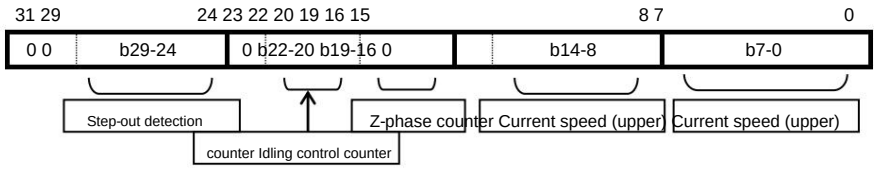
4th byte (MSB) 0



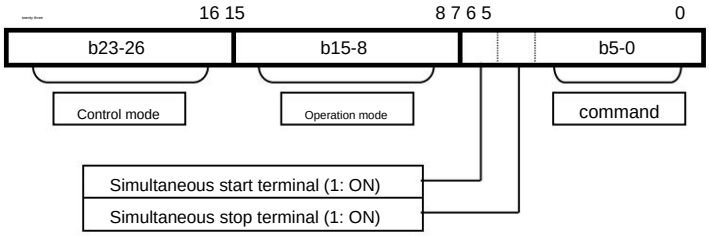
2.3 Environment Register 3 (R8)



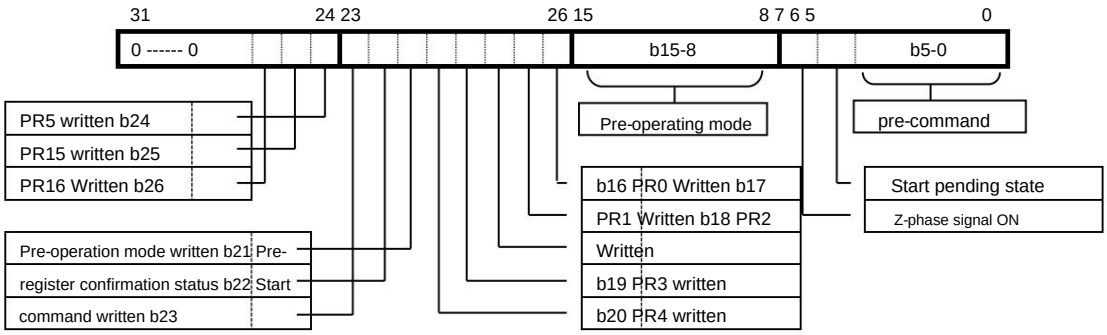
2.4 Counter monitor (R12)



2.5 Command monitor 1 (R13)



2.6 Command monitor 2 (R14)



Appendix B Major Differences between Attached Software Version 2.0 or Later and Earlier Versions

ÿBoard ID (0 to 15) is now supported in "Device Driver".

The board ID is a value determined by a 4-bit DIP switch on the board and can be set by software.

It is now possible to specify a device (board) that specifies this value.

•The item "Board ID" has been added to the device information.

[C language: Windows: VC++]

```
typedef struct _HPCDEVICEINFO {
    DWORD nBusNumber;                /* bus number */
    DWORD nDeviceNumber;             /* Device number (PCI slot number) */
    DWORD dwIoPortAddress;           /* I/O port address */
    DWORD dwIrqNo; dwNumber;         /* IRQ number */
    DWORD                            /* Control number */
→   DWORD dwBoardID;                 /* Board ID (0-15) */
} HPCDEVICEINFO, *PHPCDEVICEINFO
```

[Windows: VB6]

```
Public Type HPCDEVICEINFO
    nBusNumber As Long               'bus number
    nDeviceNumber As Long            'device number
    dwIoPortAddress As Long          'I/O port address
    dwIrqNo As Long                  'IRQ number
    dwNumber As Long                 'Control number
→   dwBoardID As Long                'Board ID (0-15)
End Type
```

[Windows: VB.NET]

```
Public Structure HPCDEVICEINFO
    Dim nBusNumber As Integer 'Bus number
    Dim nDeviceNumber As Integer 'Device number
    Dim dwIoPortAddress As Integer Dim          'I/O port address
    dwIrqNo As Integer 'IRQ number
    Dim dwNumber As Integer 'Management number
→   Dim dwBoardID As Integer 'Board ID
End Structure
```

ÿDriver I/F DLL function arguments and data types of return values

•VC++

``BOOL" and ``INT" are defined as ``DWORD (unsigned long)".

The return value is the same as before.

•VB

``Boolean" is defined as ``long".

The return value is the same as before.

ÿ Unification of header files for C language

•Driver I/F There is only one header file for DLL connection.

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