

# Lanner Bypass

## Implementation Manual

---

Julye 05, 2018

Version:2.0.3

Lanner

## Revision History

| Version | Date       | Changes   |
|---------|------------|---|
| 2.0.0   | 09/18/2015 | Initial release   |
| 2.0.1   | 10/06/2015 | Add the new description of fiber-related commands   |
| 2.0.2   | 01/15/2018 | <ul style="list-style-type: none"><li>Remove “4.4 Querying and Setting WDT Count Graduation”</li><li>Remove 0x15 command summary from 4.1</li><li>Modify command description as follows:<ul style="list-style-type: none"><li>4.2.7 0x07</li><li>4.2.8 0x08</li><li>4.2.9 0x09</li><li>4.4.3 0x22</li><li>4.4.4 0x23</li><li>4.5.3 0x32</li><li>4.5.4 0x33</li><li>4.6.3 0x42</li></ul></li></ul> |
| 2.0.3   | 07/05/2018 | Removed 5.2 sample code   |

This document contains proprietary information of Lanner Electronics Inc. and is not to be disclosed or used except in accordance with applicable agreements.

**Copyright © 2018. All Rights Reserved.**

Copyright© 2018 Lanner Electronics Inc. All rights reserved. The information in this document is proprietary and confidential to Lanner Electronics Inc. No part of this document may be reproduced in any form or by any means or used to make any derivative work (such as translation, transformation, or adaptation) without the express written consent of Lanner Electronics Inc. Lanner Electronics Inc. reserves the right to revise this document and to make changes in content from time to time without obligation on the part of Lanner Electronics Inc. to provide notification of such revision or change.

The information in this document is furnished for informational use only, is subject to change without notice, and should not be construed as a commitment by Lanner Electronics Inc. Lanner Electronics Inc. assumes no responsibility or liability for any errors or inaccuracies that may appear in this document or any software that may be provided in association with this document.

# ***Table of Contents***

---

|  |     |
|--|-----|
| Revision History .....   | ii  |
| TABLE OF CONTENTS.....   | III |
| PURPOSE OF THIS DOCUMENT.....  | VI  |
| INTENDED AUDIENCE OF THIS DOCUMENT .....   | VI  |
| CONVENTIONS/TYPOGRAPHY USED IN THIS DOCUMENT .....   | VI  |
| CHAPTER 1. INTRODUCTION TO GENERATION 3 BYPASS.....  | 2   |
| 1.1 FEATURES OF LANNER BYPASS GENERATION 3 MODULE WITH WATCHDOG CONTROL.....                               | 2   |
| 1.2 PRINCIPLES OF BYPASS.....  | 3   |
| 1.3 THE EMPLOYMENT OF UCONTROLLER IN GENERATION 3 BYPASS .....   | 6   |
| CHAPTER 2. SUPPORTING OS .....   | 7   |
| 2.1 LINUX.....   | 7   |
| 2.2 OPENBSD.....   | 7   |
| 2.3 FREEBSD.....   | 7   |
| 2.4 WINDOWS.....   | 7   |
| CHAPTER 3. BUILDING AND INSTALLING THE PROGRAM .....   | 8   |
| 3.1 LINUX.....   | 8   |
| 3.2 OPENBSD.....   | 10  |
| 3.3 FREEBSD.....   | 11  |
| 3.4 WINDOWS.....   | 11  |
| 3.5 EXECUTING THE PROGRAM (LINUX, OPENBSD, FREEBSD).....   | 12  |
| CHAPTER 4. COMMAND DESCRIPTION.....  | 13  |
| 4.1 COMMAND SUMMARY.....   | 13  |
| 4.2 GLOBAL CONFIGURATION COMMAND.....  | 16  |
| 4.2.1 Querying CPLD Major Version.....   | 16  |
| 4.2.2 Querying CPLD Minor Version.....   | 16  |
| 4.2.3 Querying Module Capability .....   | 17  |
| 4.2.4 Querying Bypass Capability (i.e. total number of bypass pairs equipped) in<br>System-off State ..... | 18  |
| 4.2.5 Querying Bypass Capability (i.e. total number of bypass pairs equipped) in<br>Just-on State .....    | 18  |
| 4.2.6 Querying Bypass Capability (i.e. total number of bypass pairs equipped) in<br>Run-time State .....   | 19  |
| 4.2.7 Querying Maximum Level of Timer Interval for Watchdog1 (in Run-time State)<br>.....                  | 20  |
| 4.2.8 Querying Maximum Level of Timer Interval for Watchdog2 (in Run-time State)<br>.....                  | 20  |
| 4.2.9 Querying Maximum Level of Timer Interval for watchdog3 in Just-on State                              |     |

|   |    |
|---|----|
| (Just-on is the brief moment when the internal power supply turns on and booting process starts) .....  | 21 |
| 4.2.10 Setting Modules Back to Factory Default .....  | 22 |
| 4.2.11 Saving Current Settings to Flash .....   | 22 |
| 4.2.12 Querying Board ID .....  | 23 |
| 4.2.13 Enabling Just-on Bypass by Simulating Just-on Instance .....   | 24 |
| 4.3 BYPASS/DISCONNECTED SETTING .....   | 27 |
| 4.3.1 Querying and Setting Bypass Status in System-off State .....  | 27 |
| 4.3.2 Querying and Setting Bypass Status in Just-on State (Just-on is the brief moment when the internal power supply turns on and booting process starts) .....        | 28 |
| 4.3.3 Querying and Setting Bypass Status in Run-time State .....  | 29 |
| 4.3.4 Querying and Setting Disconnection Status in Just-on State (Just-on is the brief moment when the internal power supply turns on and booting process starts) ..... | 30 |
| 4.3.5 Querying and Setting Disconnection Status in Run-Time State .....   | 31 |
| 4.4 BYPASS/DISCONNECTED SETTING WITH TIMER CONTROL OF WATCHDOG1 (IN RUN-TIME STATE) .....   | 32 |
| 4.4.1 Querying Watchdog1 Running Status .....   | 32 |
| 4.4.2 Setting Bypass Pairs When Watchdog1 Timer Expires .....   | 32 |
| 4.4.3 Querying and Setting Watchdog1 Timer .....  | 34 |
| 4.4.4 Querying Time to Expiration of Watchdog1 Timer .....  | 35 |
| 4.4.5 Starting Watchdog1 Timer .....  | 35 |
| 4.4.6 Stopping Watchdog1 Timer in Run-time State .....  | 36 |
| 4.4.7 Setting Watchdog1 Timeout Mode .....  | 36 |
| 4.4.8 Setting Disconnected Pairs When Watchdog1 Timer Expires .....   | 37 |
| 4.5 BYPASS/DISCONNECTED SETTING WITH TIMER CONTROL OF WATCHDOG2 (IN RUN-TIME STATE) .....   | 38 |
| 4.5.1 Querying Watchdog2 Running Status .....   | 38 |
| 4.5.2 Setting Bypass Pairs When Watchdog2 Timer Expires .....   | 38 |
| 4.5.3 Querying and Setting Watchdog2 Timer .....  | 39 |
| 4.5.4 Querying Time to Expiration of Watchdog2 Timer .....  | 40 |
| 4.5.5 Starting Watchdog2 Timer .....  | 41 |
| 4.5.6 Stopping Watchdog2 Timer in Run-time State .....  | 41 |
| 4.5.7 Setting Watchdog2 Timeout Mode .....  | 42 |
| 4.5.8 Setting Disconnected Pairs When Watchdog2 Timer Expires .....   | 42 |
| 4.6 BYPASS/DISCONNECTED SETTING WITH TIMER CONTROL OF WATCHDOG3 (IN JUST-ON STATE) .....  | 44 |
| 4.6.1 Querying Watchdog3 Running Status .....   | 44 |
| 4.6.2 Setting Bypass Pairs When Watchdog3 Timer Expires .....   | 45 |
| 4.6.3 Querying and Setting Watchdog3 Timer .....  | 45 |
| 4.6.4 Querying Time to Expiration of Watchdog3 Timer .....  | 46 |
| 4.6.5 Stopping Watchdog3 Timer .....  | 46 |
| 4.6.6 Setting Watchdog3 Timeout Mode .....  | 47 |
| 4.6.7 Setting Disconnected Pairs When Watchdog3 Timer Expires .....   | 48 |

CHAPTER 5. USING GENERATION 1 AND GENERATION 2 BYPASS ..... 50

5.1 GENERATION 1 BYPASS.....50

5.2 GENERATION 2 BYPASS.....50

Lanner

## ***Purpose of this Document***



The purpose of this document is to provide implementation information for Lanner Bypass and Watchdog functionalities.

## ***Intended Audience of this Document***

This document is expressly provided for individuals who install and configure networking appliances with the aforementioned functionality.

## ***Conventions/Typography Used in this Document***

Pay attention to the following symbols and special characters used throughout this manual.

| <b>Convention/Typography</b>  | <b>Meaning</b>  |
|---|---|
| Press Enter   | Means press the <b><i>Enter</i></b> or <b><i>Return</i></b> key or its equivalent on your computer.                           |
|  | Tips and additional information that is important in installation/operation.  |
|  | Cautions warn you that a failure to follow the recommended procedure could result in loss of data or damage to the equipment. |
| <b>Bold Text</b>  | Used to highlight an item/expression.   |
| <i>Italics</i>  | Used to lay emphasis on a filename.   |

# About this manual

In this implementation manual, it will not only clearly tell you how to build Lanner's Generation 3 Bypass in different OSs but provide many commands as well as examples to do the bypass control.

## Organization of the Manual

---

- Chapter 1 "Introduction to Generation 3 Bypass" describes the functions/features
- Chapter 2 "Supporting OS"
- Chapter 3 "Building and Installing the Program"
- Chapter 4 "Command Description"
- Chapter 5 "Using Generation 1 and Generation 2 Bypass" mentions that how these 2 kinds of Bypass work and the way of operation.

# 1

## Introduction to Generation 3 Bypass

---

### 1.1 Features of Lanner Bypass Generation 3 Module with Watchdog Control

Lanner's Bypass Modules include WDT (Watchdog Timer) controller and Bypass switch. Our Bypass Modules also include a software development kit that enables system designer to efficiently design systems to support bypass functionality. Lanner Bypass Modules with watchdog control have the following features:

- Communication through SMBUS (I2C)
- Independent bypass status control for each pair up to a total of 4 pairs
- Lanner Bypass Modules can bypass systems Ethernet ports on a host system during three instances: Just-on (Just-on is the brief moment when the internal power supply turns on and booting process starts), system off, or upon software request (during run-time).
- Software programmable bypass or normal mode
- Software programmable timer interval:
  - Just-on watchdog timer, used during JUST-ON, has setting of 5~1275 seconds of timer interval.
  - Run-time watchdog timer, used during run-time, has setting of 1~255 seconds of timer interval.
- Multiple Watchdog Timers:
  - Two for run-time: It is designed to give you a more variety of controls of the bypass on port basis. By using dedicated watchdogs for different pairs of bypass, you have the flexibility to manage the bypass status for them differently.
  - One for Just-on: It is designed to give you the precise control of the bypass during this phase. You can use this timer to delay enabling the bypass in Just-on state.

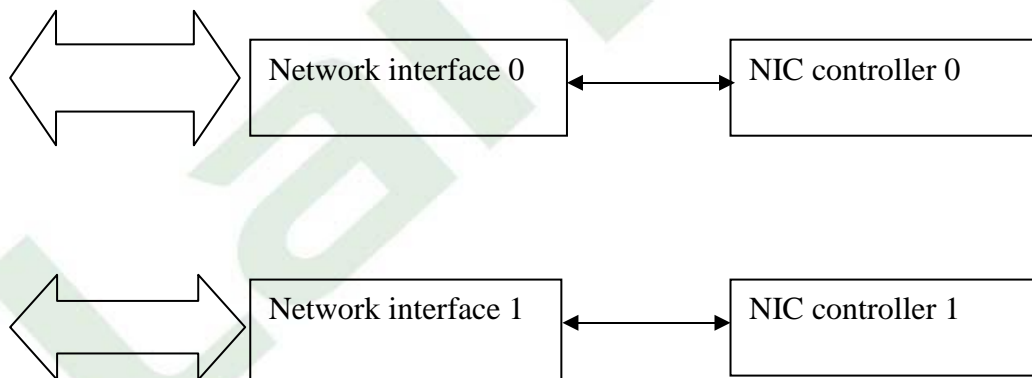


## 1.2 Principles of Bypass

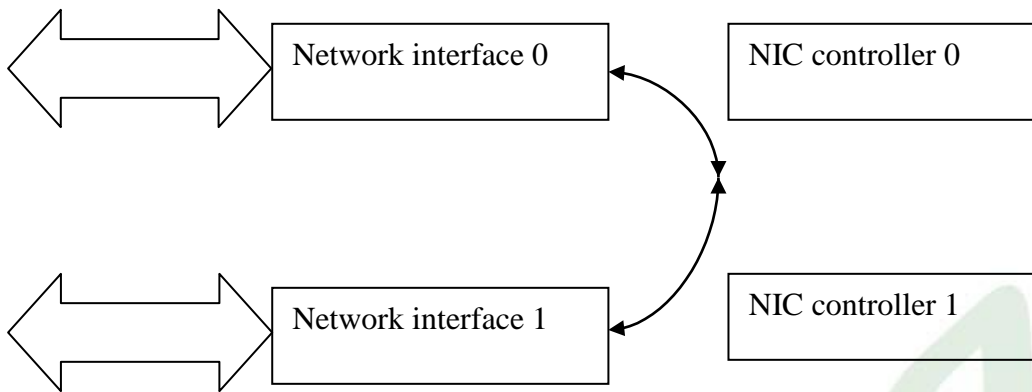
Lanner Bypass and Watchdog module have come into a new generation – Generation 3 Bypass. It now integrates with a dedicated uController as an improvement over the previous generations in implementing bypass and watchdog functionalities via software programming. There are typically two communication statuses for the bypass function: “normal” status and “bypass” status. In bypass status, the connections of the Ethernet network ports are disconnected from the system interfaces and switched over to the other port to create a crossed connection loop-back between the Ethernet ports. Hence, in bypass, all packets received from one port are transmitted to other port and vice versa. Furthermore, bypass status can be controlled under any of the following instances: System-off, Just-on (Just-on is the brief moment when the internal power supply turns on and booting process starts) and run-time. There are also three watchdog timers – Watchdog3 for being used during Just-on (Just-on is the brief moment when the internal power supply turns on and booting process starts) and Watchdog1 and Watchdog2 for being used during Run-Time – and they can work in conjunction with bypass function to trigger the system to be in bypass status when fault conditions are detected. By using the sample driver and software provided by Lanner, users could implement this functionality smoothly.

The following illustration demonstrates the difference between bypass and non-bypass status.

**Non-bypass Status:** NIC controller is functioning normally.

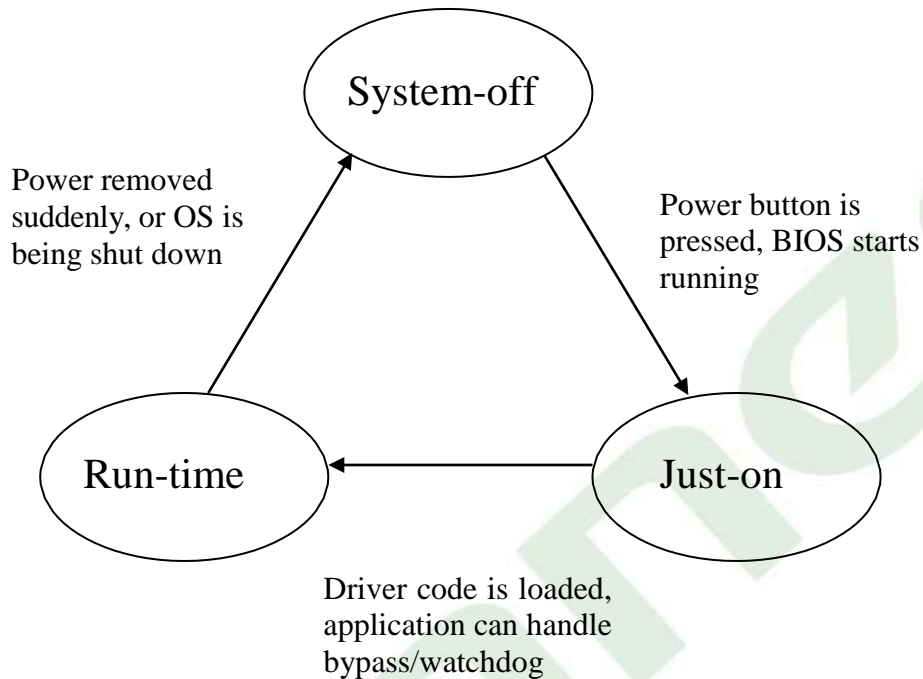


**Bypass Status:** Ethernet network ports are disconnected from the system interfaces. A crossed connection loop-back between the Ethernet ports is established.



LearnMore

**3 States to Manage Bypass Status:** The following diagram shows the 3 instances in which the bypass status can be controlled:

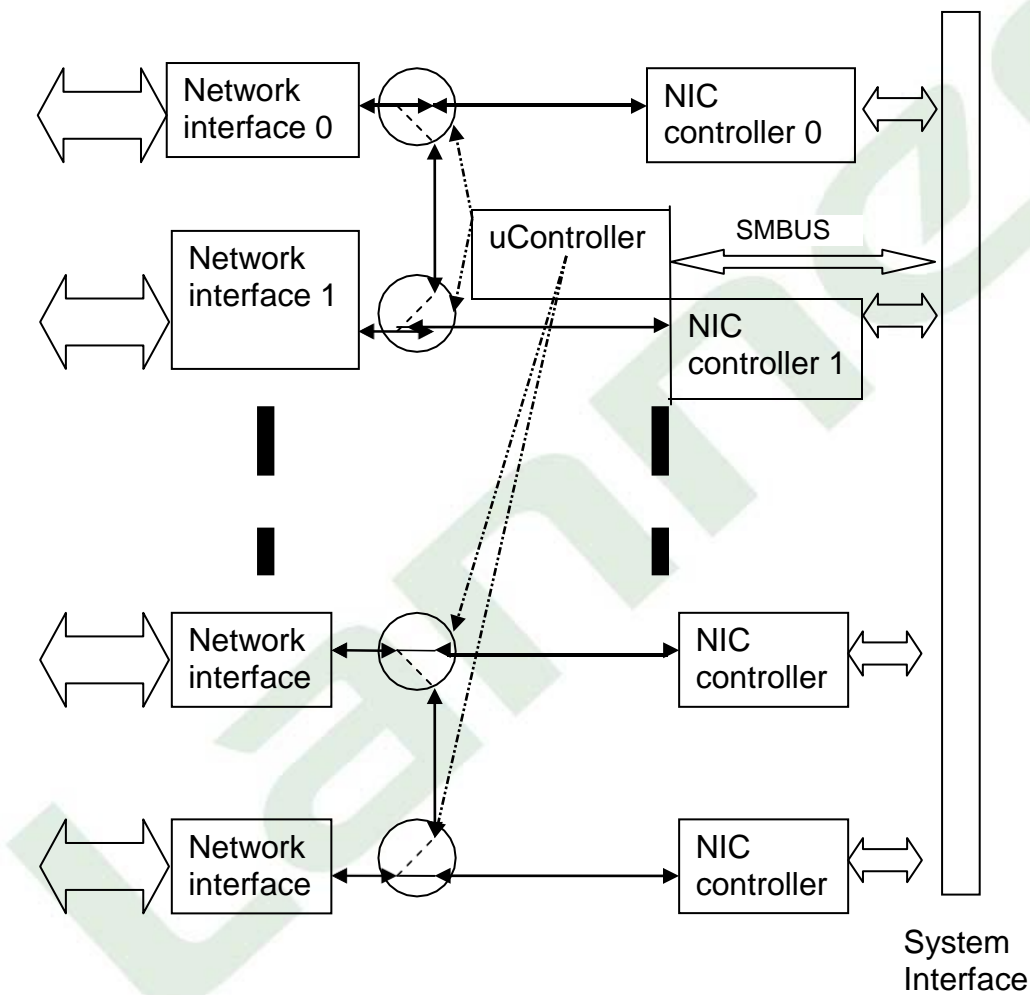


**Note:**

1. Soft reboot, i.e. restarting a computer under software control without shutdown or triggering a reset line, will not activate the bypass settings in JUST-ON. Instead, it will carry on the bypass and watchdog status from the preceding run-time instance. Therefore, it is required to perform any shut-down procedure before starting the system in order to go through JUST-ON.
2. System-off is when the power is turned off, or when the OS is being shut down.
3. Run-time is when the driver code is loaded, and the program can take over the bypass/watchdog functions.
4. Just-on is the brief moment when the internal power supply turns on and booting process starts.

## 1.3 The Employment of uController in Generation 3 Bypass

The uController takes the role as the main controller of the bypass function. The interface between the system and uController is SMBUS (I2C). The uController can control up to 4\* pairs of bypass. The following diagram depicts the utilization of uController in Generation 3 Bypass. By using Lanner Bypass driver and software, system designers can write commands to the onboard ucontroller and bypass circuitry to manage the bypass status.



Note: The number of bypass pairs varies depending on the models of the system.

# Supporting OS

---

## 2.1 Linux

Linux refers to the family of Unix-like computer operating systems using the Linux kernel. Linux can be installed on a wide variety of computer hardware, ranging from mobile phones, tablet computers and video game consoles, to mainframes and supercomputers. Linux is a leading server operating system, and runs the 10 fastest supercomputers in the world. Lanner's bypass/watchdog driver has been tested on the Linux kernel 2.4, Linux kernel 2.6 and above. [http://en.wikipedia.org/wiki/Linux - cite\\_note-10](http://en.wikipedia.org/wiki/Linux_-_cite_note-10)

## 2.2 OpenBSD

OpenBSD is a Unix-like computer operating system descended from Berkeley Software Distribution (BSD), a UNIX derivative developed at the University of California, Berkeley. It was forked from NetBSD by project leader Theo de Raadt in late 1995. The project is widely known for the developers' insistence on open source code and quality documentation, uncompromising position on software licensing, and focus on security and code correctness. Lanner's bypass/watchdog program has been tested on the OpenBSD-4.7.

## 2.3 FreeBSD

FreeBSD is a complete operating system; the kernel, device drivers, and all of the userland utilities, such as the shell, are held in the same source code revision tracking tree. (This is in contrast to Linux distributions, for which the kernel, userland utilities, and applications are developed separately, and then packaged together in various ways by others.) Third-party application software may be installed using various software installation systems, the two most common being source installation and package installation, both of which use the FreeBSD Ports system.

## 2.4 Windows

Microsoft Windows is a series of software operating systems and graphical user interfaces produced by Microsoft. Microsoft first introduced an operating environment named Windows on November 20, 1985 as an add-on to MS-DOS in response to the growing interest in graphical user interfaces (GUIs). Microsoft Windows came to dominate the world's personal computer market, overtaking Mac OS, which had been introduced in 1984. As of October 2009, Windows had approximately 91% of the market share of the client operating systems for usage on the Internet. [http://en.wikipedia.org/wiki/Microsoft\\_Windows - cite\\_note-2](http://en.wikipedia.org/wiki/Microsoft_Windows_-_cite_note-2)

# 3

## Building and Installing the Program

### 3.1 Linux

#### Build

To build program source code on Linux platform, use the following steps as a guideline:

1. Copy the proper makefile from the Driver and Manual CD to your system:  
Makefile.linux
2. Set the access mode with these two parameters by editing the Makefile.linux directly: `DIRECT_IO_ACCESS= [0|1]` (enter either 1 or 0) and `LANNER_DRIVER= [0|1]` (enter either 1 or 0). Refer to the next section for more details on the access mode.
3. Type make to build source code:  
make Makefile (**Note:** omit the file extensions)  
After compiling, the executable program (bpwd\_tst) and the driver (bpwd\_drv.ko) will be in the *bin* subdirectory.



#### Note:

The OS supported by Lanner Bypass function includes platforms based on Linux Kernel series 2.4.x and Linux Kernel series 2.6.x and above and DOS.

#### Install

The installation procedures depend on the access mode that you have set by using the method mentioned above.

If you have set `DIRECT_IO_ACCESS=1`, driver installation is not necessary. Proceed to the next section on executing the Lanner bypass/watchdog program.

If you have set `DIRECT_IO_ACCESS=0` and `LANNER_DRIVER=1`, Lanner bypass driver needs to be installed. Install the driver and create a node in the `/dev` directory as shown in the following example:

## ***Building and Installing the Program***

---

```
#insmod bpwd_drv.[k]o  
#mknod /dev/bpwd_drv c 247 0
```

If you have set `DIRECT_IO_ACCESS=0` and `LANNER_DRIVER=0`, Linux I2C driver which is provided with the respective Linux Kernel version has to be installed on the system.

And i2c subsystem, i2c-core, i2c-dev and buses/i2c-i801 all need to be loaded before accessing the Lanner bypass/watchdog module; so a dev node needs to be created as shown below:

```
#mknod /dev/i2c-0 c 89 0
```



### Note:

1. For descriptions of the command, refer to the Readme file contained within the program.
2. The major number needed with the mknod command varies with different software versions; look up the Readme file for this value.

## 3.2 OpenBSD

### Build

Put all files in the same folder and follow these procedures:

1. bpwd\_drv.o (Kernel mode driver module)  
`#cc -D_KERNEL -l/sys -c bpwd_drv.c`
2. bpwd\_tst (User mode driver testing program)  
`#cc -o bpwd_tst bpwd_test.c`

### Install

Use the following procedures to install the program:

1. Insert module  
`#modload -e bpwd bpwd_drv.o`
2. Check device mode major number and module ID.  
`#modstat`  
(e.g. Type ID Off Loadaddr...)  
( Dev 0 29 addr...)  
Here, major nume-29, ID=0
3. Make device node  
`#mknod -m 644 /dev/bpwd c 29 0`
4. Test it  
`#!/bpwd_tst`
5. Remove module  
`#modunload -i 0`



#### Note:

For descriptions of the command, refer to the Readme file contained within the program.



## 3.3 FreeBSD

### Build

Use the script file *compiler.sh* to build and compiler the source code. After compiling, you can get the driver file *bpwd\_drv.ko* and the test file *bpwd\_tst* in the same folder.

```
#sh compiler.sh
```

### Install

To load the driver, type:

```
#kldload -v ./bpwd_drv.ko
```

To remove the driver:

1. Use the following command to check the ID of bpwd driver:

```
#kldstat  
(e.g. Id Refs Address Size Name )  
( 1 8 0xc0400000 9fab28 kernel )  
( 2 1 0xc0dfb0009 6a45c acpi.ko )  
( 3 1 0xc5a0f0009 22000 linux.ko )  
( 4 1 0xc5c4d0009 2000 bpwd_drv.ko )
```

2. Use the following command to remove the bpwd driver:

```
#kldunload -i 4
```



#### Note:

For descriptions of the command, refer to the [Readme](#) file contained within the program.

## 3.4 Windows

To build and install, follow the [User Guide.txt] to set up the generation 3 bypass.



#### Note:

For descriptions of the command, refer to the [Readme](#) file contained within the program.

## 3.5 Executing the Program (Linux, OpenBSD, FreeBSD)

This section contains instructions on the execution of the bypass function. Note that the installation needs to be completed before proceeding with the execution.

Scan known device. After scanning the device, a message “response with Read byte command” will be displayed, indicating the address could be used. Specific to Lanner bypass/watchdog, the acceptable I2C address is from **0x30** to **0x37**.

```
#!/bpwd_tst -S -M [Model Name]
```

SMBUS Read data protocol by word– for [Read]

Refer to the next chapter for the command usage and the details of parameters.

```
#!/bpwd_tst -r -d [I2C_address] -c [Command] -M [Model Name]
```

SMBUS write data protocol by byte– for [Write]

Refer to the next chapter for the command usage and the details of parameters.

```
Usage: #./bpwd_tst -w -d [I2C_address] -c [Command] -o [Byte_data] -M [Model Name]
```

Get Lanner bypass module information

```
#!/bpwd_tst -l -d [I2C_address] -M [Model Name]
```

Get help with execution

```
#!/bpwd_tst -h
```



### Note:

1. For the use of the `-M` option, refer to the Readme file contained within the program.
2. The I2C address has rules based on the connection method; refer to the Readme file for more details.

# Command Description

## 4.1 Command Summary

| Command                     | Function Description   | Access Mode |
|-----------------------------|--|-------------|
| <i>Global Configuration</i> |  |             |
| <a href="#"><u>0x01</u></a> | Querying CPLD Major Version  | Read Only   |
| <a href="#"><u>0x02</u></a> | Querying CPLD Minor Version  | Read Only   |
| <a href="#"><u>0x03</u></a> | Querying Module Capability   | Read Only   |
| <a href="#"><u>0x04</u></a> | Querying Bypass Capability (i.e. total number of bypass pairs equipped) in System-off State  | Read Only   |
| <a href="#"><u>0x05</u></a> | Querying Bypass Capability (i.e. total number of bypass pairs equipped) in Just-on State   | Read Only   |
| <a href="#"><u>0x06</u></a> | Querying Bypass Capability (i.e. total number of bypass pairs equipped) in Run-time State  | Read Only   |
| <a href="#"><u>0x07</u></a> | Querying Maximum Level of Timer Interval for Watchdog1 (in Run-time state)   | Read Only   |
| <a href="#"><u>0x08</u></a> | Querying Maximum Level of Timer Interval for Watchdog2 (in Run-time state)   | Read Only   |
| <a href="#"><u>0x09</u></a> | Querying Maximum Level of Timer Interval for watchdog3 in Just-on State (Just-on is the brief moment when the internal power supply turns on and booting process starts) | Read Only   |
| <a href="#"><u>0x0A</u></a> | Setting Modules Back to Factory Default  | Write Only  |

|                                    |  |            |
|------------------------------------|--|------------|
| <a href="#"><u>0x0B</u></a>        | Saving Current Settings to Flash   | Write Only |
| <a href="#"><u>0x0C</u></a>        | Querying Board ID  | Read/Write |
| <a href="#"><u>0x0F</u></a>        | Enabling Just-on Bypass by Simulating Just-on instance   | Read/Write |
| <b><i>Bypass Setting</i></b>       |  |            |
| <a href="#"><u>0x10</u></a>        | Querying and Setting Bypass Status in System-off State   | Read/Write |
| <a href="#"><u>0x11</u></a>        | Querying and Setting Bypass Status in Just-on State (Just-on is the brief moment when the internal power supply turns on and booting process starts)   | Read/Write |
| <a href="#"><u>0x12</u></a>        | Querying and Setting Bypass Status in Run-time State   | Read/Write |
| <b><i>Disconnected Setting</i></b> |  |            |
| <a href="#"><u>0x13</u></a>        | Querying and Setting Disconnection Status in Just-on State (Just-on is the brief moment when the internal power supply turns on and booting process starts)<br><br>(Provided for the Fiber Media Only) | Read/Write |
| <a href="#"><u>0x14</u></a>        | Querying and Setting Disconnection Status in Run-Time State<br><br>(Provided for the Fiber Media Only)   | Read/Write |
| <b><i>Watchdog1 Setting</i></b>    |  |            |
| <a href="#"><u>0x20</u></a>        | Querying Watchdog1 Running Status  | Read Only  |
| <a href="#"><u>0x21</u></a>        | Setting Bypass Pairs When Watchdog1 Timer Expires  | Read/Write |
| <a href="#"><u>0x22</u></a>        | Querying and Setting Watchdog1 Timer   | Read/Write |
| <a href="#"><u>0x23</u></a>        | Querying Time to Expiration of Watchdog1 Timer   | Read Only  |
| <a href="#"><u>0x24</u></a>        | Starting Watchdog1 Timer   | Write Only |
| <a href="#"><u>0x25</u></a>        | Stopping Watchdog1 Timer in Run-time State   | Write Only |
| <a href="#"><u>0x26</u></a>        | Setting Watchdog1 Timeout Mode<br>(Provided for the Fiber Media Only)  | Write Only |

## Command Description

---

|                             |  |            |
|-----------------------------|--|------------|
| <a href="#"><u>0x27</u></a> | Setting Disconnected Pairs When Watchdog1 Timer Expires<br>(Provided for the Fiber Media Only) | Read/Write |
| <b>Watchdog2 Setting</b>    |  |            |
| <a href="#"><u>0x30</u></a> | Querying Watchdog2 Running Status  | Read Only  |
| <a href="#"><u>0x31</u></a> | Setting Bypass Pairs When Watchdog2 Expires  | Read/Write |
| <a href="#"><u>0x32</u></a> | Querying and Setting Watchdog2 Timer   | Read/Write |
| <a href="#"><u>0x33</u></a> | Querying Time to Expiration of Watchdog2 Timer   | Read Only  |
| <a href="#"><u>0x34</u></a> | Starting Watchdog2 Timer   | Write Only |
| <a href="#"><u>0x35</u></a> | Stopping Watchdog2 Timer in Run-time State   | Write Only |
| <a href="#"><u>0x36</u></a> | Setting Watchdog2 Timeout Mode<br>(Provided for the Fiber Media Only)                          | Write Only |
| <a href="#"><u>0x37</u></a> | Setting Disconnected Pairs When Watchdog2 Timer Expires<br>(Provided for the Fiber Media Only) | Read/Write |
| <b>Watchdog3 Setting</b>    |  |            |
| <a href="#"><u>0x40</u></a> | Querying Watchdog3 Running Status  | Read Only  |
| <a href="#"><u>0x41</u></a> | Setting Bypass Pairs When Watchdog3 Expires  | Read/Write |
| <a href="#"><u>0x42</u></a> | Querying and Setting Watchdog3 Timer   | Read/Write |
| <a href="#"><u>0x43</u></a> | Querying Time to Expiration of Watchdog3 Timer   | Read Only  |
| <a href="#"><u>0x45</u></a> | Stopping Watchdog3 Timer   | Write Only |
| <a href="#"><u>0x46</u></a> | Setting Watchdog3 Timeout Mode<br>(Provided for the Fiber Media Only)                          | Write Only |
| <a href="#"><u>0x47</u></a> | Setting Disconnected Pairs When Watchdog3 Timer Expires<br>(Provided for the Fiber Media Only) | Read/Write |

## 4.2 Global Configuration Command

The commands of the global configuration will let you query the capabilities of your modules such as the number of bypass pairs equipped, which watchdog timers has been enabled, and the maximum level of timer interval for each watchdog timers. The following sections give details about each executable command.

### 4.2.1 Querying CPLD Major Version

**Command:** 0x01

**Description:** Get CPLD major version.

**Access mode:** Read

**Return value:**

LSB byte = Command | 0x80

MSB byte = CPLD major version (The returned byte consists of LSB and MSB byte. The LSB byte signals whether uController responds with the command and it equals to 0x80 plus the value of command; the MSB byte is the value returned by the command)

**<example>** In this example, let I2C address=0x30 and Model Name=MB-887X

```
./bpwd_tst -r -d 0x30 -c 0x01 -M MB-887X [enter]
```

```
READ ONLY ADDRESS:0x30 Command:0x01 ...OK, DATA = 0x0081
```

**Interpretation:** The 0x00 byte (MSB byte) of DATA “0x0081” indicates that the version is 0.

### 4.2.2 Querying CPLD Minor Version

**Command:** 0x02

**Description:** Get CPLD minor version.

**Access mode:** Read

**Return value:**

LSB byte = Command | 0x80

MSB byte = CPLD minor version

**<example>** In this example, let I2C address=0x30 and Model Name= MB-887X

```
./bpwd_tst -r -d 0x30 -c 0x02 -M MB-887X [enter]
```

```
READ ONLY ADDRESS:0x30 Command:0x02 ...OK, DATA = 0x0082
```

**Interpretation:** The 0x00 byte (MSB byte) of DATA “0x0082” indicates that the version is 0.

## 4.2.3 Querying Module Capability

**Command:** 0x03

**Description:** Get module capability (1 means capable whereas 0 means incapable.).

**Access mode:** Read

**Return value:**

LSB byte = Command | 0x80

MSB byte = return value of module capability (The returned byte consists of LSB and MSB byte. The LSB byte signals whether uController responds with the command and it equals to 0x80 plus the value of command; the MSB byte is the value returned by the command)

The return value has the following interpretations:

| Return value in hex | Return value in binary | Interpretation                        |
|---------------------|------------------------|---------------------------------------|
| 0x01                | 0000001b               | Capable of System-off bypass function |
| 0x02                | 0000010b               | Capable of Just-on bypass function    |
| 0x04                | 0000100b               | Capable of Run-time bypass function   |
| 0x08                | 0001000b               | Capable of Watchdog1 bypass function  |
| 0x10                | 0010000b               | Capable of Watchdog2 bypass function  |
| 0x20                | 00100000b              | Capable of Watchdog3 bypass function  |

|   |
|---|
| <p><b>&lt;example&gt;</b> In this example, let I2C address=0x30 and Model Name= MB-887X</p> <pre>./bpwd_tst -r -d 0x30 -c 0x03 -M MB-887X [enter]</pre> <p>READ ONLY ADDRESS:0x30 Command:0x03 ...OK, DATA = 0x3f83</p> <p><b>Interpretation:</b> The above result shows that all capabilities are enabled in the module.</p> |
|---|



## 4.2.4 Querying Bypass Capability (i.e. total number of bypass pairs equipped) in System-off State

**Command:** 0x04

**Description:** Get system-off bypass capability.

| Hex  | Binary    | Description                      |
|------|-----------|----------------------------------|
| 0x00 | 00000000b | Means no bypass equipped         |
| 0x01 | 00000001b | Means 1 pair of bypass equipped  |
| 0x03 | 00000011b | Means 2 pairs of bypass equipped |
| 0x07 | 00000111b | Means 3 pairs of bypass equipped |
| 0x0f | 00001111b | Means 4 pairs of bypass equipped |

**Access mode:** Read

**Return value:**

LSB byte = Command | 0x80

MSB byte = System-off bypass pairs enabled (The returned byte consists of LSB and MSB byte. The LSB byte signals whether uController responds with the command and it equals to 0x80 plus the value of command; the MSB byte is the value returned by the command)

**<example>** In this example, let I2C address=0x30 and Model Name= MB-887X  
 ./bpwd\_tst -r -d 0x30 -c 0x04 -M MB-887X [enter]  
 READ ONLY ADDRESS:0x30 Command:0x04 ...OK, DATA = 0x0f84  
**Interpretation:** The above result shows 4 pairs of bypass that can be used in System-off state are equipped.

## 4.2.5 Querying Bypass Capability (i.e. total number of bypass pairs equipped) in Just-on State

**Command:** 0x05

**Description:** Get Just-on (Just-on is the brief moment when the internal power supply turns on and booting process starts) bypass capability.

| Hex  | Binary    | Description                      |
|------|-----------|----------------------------------|
| 0x00 | 00000000b | Means no bypass equipped         |
| 0x01 | 00000001b | Means 1 pair of bypass equipped  |
| 0x03 | 00000011b | Means 2 pairs of bypass equipped |
| 0x07 | 00000111b | Means 3 pairs of bypass equipped |
| 0x0f | 00001111b | Means 4 pairs of bypass equipped |



**Access mode:** Read

**Return value:**

LSB byte = Command | 0x80

MSB byte = Just-on bypass pairs enabled (The returned byte consists of LSB and MSB byte. The LSB byte signals whether uController responds with the command and it equals to 0x80 plus the value of command; the MSB byte is the value returned by the command)

**<example>** In this example, let I2C address=0x30 and Model Name= MB-887X

```
./bpwd_tst -r -d 0x30 -c 0x05 -M MB-887X [enter]
```

```
READ ONLY ADDRESS:0x30 Command:0x05 ...OK, DATA = 0x0f85
```

**Interpretation:** The above result shows 4 pairs of bypass that can be used in JUST-ON state are equipped.

### 4.2.6 Querying Bypass Capability (i.e. total number of bypass pairs equipped) in Run-time State

**Command:** 0x06

**Description:** Get run-time (Just-on is the brief moment when the internal power supply turns on and booting process starts) bypass equipped.

| Hex  | Binary    | Description                      |
|------|-----------|----------------------------------|
| 0x00 | 0000000b  | Means no bypass equipped         |
| 0x01 | 0000001b  | Means 1 pair of bypass equipped  |
| 0x03 | 0000011b  | Means 2 pairs of bypass equipped |
| 0x07 | 0000111b  | Means 3 pairs of bypass equipped |
| 0x0f | 00001111b | Means 4 pairs of bypass equipped |

**Access mode:** Read

**Return value:**

LSB byte = Command | 0x80

MSB byte = run-time bypass pairs enabled (The returned byte consists of LSB and MSB byte. The LSB byte signals whether uController responds with the command and it equals to 0x80 plus the value of command; the MSB byte is the value returned by the command)

**<example>** In this example, let I2C address=0x30 and Model Name= MB-887X

```
./bpwd_tst -r -d 0x30 -c 0x06 -M MB-887X [enter]
```

```
READ ONLY ADDRESS:0x30 Command:0x06 ...OK, DATA = 0x0f86
```

**Interpretation:** The above result shows 4 pairs of bypass that can be used in run-time state are equipped.

## 4.2.7 Querying Maximum Level of Timer Interval for Watchdog1 (in Run-time State)



**Note:**

The maximum level of timer interval indicates the maximum value at which the watchdog can start the countdown.

**Command: 0x07**

**Description:** Get the maximum level of timer interval of watchdog1 in run-time state. Range: 0~255 levels (1 level = 1 second).

**Access mode:** Read

**Return value:**

LSB byte = Command | 0x80

MSB byte = Maximum timer interval for watchdog1 (The returned byte consists of LSB and MSB byte. The LSB byte signals whether uController responds with the command and it equals to 0x80 plus the value of command; the MSB byte is the value returned by the command)

**<example>** In this example, let I2C address=0x30 and Model Name= MB-887X

```
./bpwd_tst -r -d 0x30 -c 0x07 -M MB-887X [enter]
```

```
READ ONLY ADDRESS:0x30 Command:0x07 ...OK, DATA = 0x ff87
```

**Interpretation:** The return value shows the maximum timer interval of watchdog1 is 255 seconds.

## 4.2.8 Querying Maximum Level of Timer Interval for Watchdog2 (in Run-time State)

**Note:**



The maximum level of timer interval indicates the maximum value at which the watchdog can start the countdown.

**Command: 0x08**

**Description:** Get the maximum level of timer interval of watchdog2 during run-time. Range: 0~255 levels (1 level = 1 second).

**Access mode:** Read

**Return value:**

LSB byte = Command | 0x80

MSB byte = Maximum timer interval for watchdog2 (The returned byte consists of LSB and MSB byte. The LSB byte signals whether uController responds with the command and it equals to 0x80 plus the value of command; the MSB byte is the value returned by the command)

|   |
|---|
| <p><b>&lt;example&gt;</b> In this example, let I2C address=0x30 and Model Name= MB-887X</p>                       |
| <pre>./bpwd_tst -r -d 0x30 -c 0x08 -M MB-887X [enter]</pre>   |
| <pre>READ ONLY ADDRESS:0x30 Command:0x08 ...OK, DATA = 0xff88</pre>   |
| <p><b>Interpretation:</b> The return value shows that the maximum timer interval of watchdog2 is 255 seconds.</p> |

### 4.2.9 Querying Maximum Level of Timer Interval for watchdog3 in Just-on State (Just-on is the brief moment when the internal power supply turns on and booting process starts)



**Note:**

The maximum level of timer interval indicates the maximum value at which the watchdog can start the countdown.

**Command: 0x09**

**Description:** Get the maximum level of timer interval of watchdog3 in Just-on state(Just-on is the brief moment when the internal power supply turns on and booting process starts) mode. Range: 0~255 scale (1 scale = 5 second).

**Access mode:** Read

**Return value:**

LSB byte = Command | 0x80

MSB byte = Maximum timer interval for watchdog3 (The returned byte consists of LSB and MSB byte. The LSB byte signals whether uController responds with the command and it equals to 0x80 plus the value of command; the MSB byte is the value returned by the command)

|  |
|--|
| <p><b>&lt;example&gt;</b> In this example, let I2C address=0x30 and Model Name= MB-887X</p>                                |
| <pre>./bpwd_tst -r -d 0x30 -c 0x09 -M MB-887X [enter]</pre>  |
| <pre>READ ONLY ADDRESS:0x30 Command:0x09 ...OK, DATA = 0xff89</pre>  |
| <p><b>Interpretation:</b> The return value shows that the maximum timer interval of watchdog3 is 255x5 (1275) seconds.</p> |

## 4.2.10 Setting Modules Back to Factory Default

**Command:** 0x0A

**Byte\_data:** x (insignificant)

**Description:** Reset module to the factory default values as shown below:

1. All pairs set of system-off bypass: enable
2. All pairs set of just-on bypass: disable
3. All pairs set of run-time bypass: disable
4. **watchdog1** timer (run-time) stop  
**watchdog2** timer(run-time) stop  
**watchdog3** timer(Just-on) stop
5. **watchdog1** (run-time) timer interval=0 (value which can be accessed in read/write mode by 0x22 command, refer to the section of 4.5.3 Querying and Setting Watchdog1 Timer).  
**watchdog2** (run-time) timer interval=0 (value which can be accessed in read/write mode by 0x32 command, refer to the section of 4.6.3 Querying and Setting Watchdog2 Timer).  
**watchdog3** (Just-on) timer interval=0 (value which can be accessed in read/write mode by 0x42 command, refer to the section of 4.7.3 Querying and Setting Watchdog3 Timer).

**Access mode:** Write

**<example>** In this example, let I2C address=0x30 and Model Name=FW-9655  
 ./bpwd\_tst -w -d 0x30 -c 0x0A -o 0x00 -M MB-887X [enter] ( enter any value between 0x00 and 0xff, it is not significant)  
 WRITE ADDRESS:0x30 Command:0x0A DATA:0x00...OK



**Note:**

The 0x0A command will only reset the settings back to its factory defaults; it will not rewrite the settings to the flash. In order to save the current running configurations to the flash permanently, always use the 0x0B command.

## 4.2.11 Saving Current Settings to Flash

**Command:** 0x0B

**Byte\_data:** x (insignificant)

**Description:** Save the current value to NVRAM. It includes the following settings:

1. watchdog3 (Just-on) timer counter setting
2. Just-on disconnect setting
3. Enable the disconnection function of the port setting while watchdog1(run-time) timer is timed out.
4. watchdog1(run-time) mode setting
5. Enable the disconnection function of the port setting while watchdog2(run-time) timer is timed out.
6. watchdog2(run-time) mode setting
7. Enable the disconnection function of the port setting while watchdog3(Just-on) timer is timed out.
8. watchdog3(Just-on) mode setting

**Access mode:** Write

**<example>** In this example, let I2C address=0x30 and Model Name= MB-887X  
 ./bpwd\_tst -w -d 0x30 -c 0x0B -o 0xff -M MB-887X [enter]( enter any value between 0x00 and 0xff, it is not significant)  
 WRITE ADDRESS:0x30 Command:0x0B DATA:0x00...OK



### Caution:

There is a limitation of 10,000 writes to the NVRAM via the 0x0B command.

## 4.2.12 Querying Board ID

To get the board id, you need to do 1 time of [Write] and then 3 times of [Read] command consecutively to get the full board ID.

### Command: 0x0C

**Byte\_data:** x (insignificant)

**Description:** Write this command to get the board ID from CPLD.

After executing this command, you must execute the Read command below for Board ID 3 times consecutively.

**Access Mode:** Write

### Command: 0x0C

**Description:** Read the Board ID (MAC address). When reading the Board ID, you need to execute this command to communicate with the register (execution instruction) three times to get the full Board ID. Before reading the Board ID, make sure you write first by using the command in write mode (see above).

**Return Value:**

MSB byte = 0xXXXX (XXXX means the Board ID)

**Access Mode:** Read

|  |
|--|
| <p><b>&lt;example&gt;</b> In this example, let I2C address=0x30 and Model Name= MB-887X</p> <pre>./bpwd_tst -w -d 0x30 -c 0x0C -o 0x00 -M MB-887X [enter] ( enter any value between 0x00 and 0xff, it is not significant) WRITE ADDRESS:0x30 Command:0x0C DATA:0x00...OK ./bpwd_tst -r -d 0x30 -c 0x0C -M MB-887X [enter] READ ONLY ADDRESS:0x30 Command:0x0C ...OK, DATA = 0x<u>90 00</u> ./bpwd_tst -r -d 0x30 -c 0x0C -M MB-887X [enter] READ ONLY ADDRESS:0x30 Command:0x0C ...OK, DATA = 0x<u>1a 0b</u> ./bpwd_tst -r -d 0x30 -c 0x0C -M MB-887X [enter] READ ONLY ADDRESS:0x30 Command:0x0C ...OK, DATA = 0x<u>ee 72</u> <b>Interpretation:</b> It indicates that the Board ID is 00 90 0b 1a 72 ee.</pre> |
|--|

## 4.2.13 Enabling Just-on Bypass by Simulating Just-on Instance

This command will forcibly enable the bypass pair during run-time by simulating the just-on instance.

This command is provided as a means to let the system restarted by software without the complete shutdown procedures to force to enable just-on bypass by simulating just-on instance (Soft reboot is restarting a computer under software control without shutdown or triggering a reset line). This command will also force to enable the setting in watchdog3. For more information, please see the sections of 4.3.2 Querying and Setting Bypass Status in Just-on State (Just-on is the brief moment when the internal power supply turns on and booting process starts) and 4.7 Bypass/Disconnected Setting with Timer Control of Watchdog3 (in Just-on state).

### Command: 0x0F

**Description:** Get Just-on (Just-on is the brief moment when the internal power supply turns on and booting process starts) bypass status in either enable or disable.

**Access mode:** Read

**Return value:**

LSB byte = Command | 0x80

MSB byte = Bypass pairs that are forced to enable during Just-on

(The returned byte consists of LSB and MSB byte. The LSB byte signals whether uController responds with the command and it equals to 0x80 plus the value of command; the MSB byte is the value returned by the command), and the meaning of return value is listed as **Table 1**.

**<example>** In this example, let I2C address=0x30 and Model Name= MB-887X

```
./bpwd_tst -r -d 0x30 -c 0x0f -M MB-887X [Enter]
```

```
READ ADDRESS:0x30 Command:0x0f ...OK, DATA = 0x038f
```

**Interpretation:** The above return value means that pair 1 and pair 2 will be set to enable just-on bypass forcibly.

### Command: 0x0F

**Byte\_data:** 0x00~0x0f

**Description:** Force to enable or disable the just-on bypass by simulating just-on instance. Bit 0~7 indicate to force to enable or disable just-on bypass for that pair; 1 means to enable whereas 0 means to disable.

**Access mode:** Write

**<example>**In this example, let I2C address=0x30 and Model Name=MB-887X

```
./bpwd_tst -w -d 0x30 -c 0x0f -o 0x03 -M MB-887X [Enter]
```

```
WRITE ADDRESS:0x30 Command:0x03 Data:0x03...OK
```

**Explanation:** The above command will force to enable bypass pair 1 and 2 as well as just-on *watchdog timer 3* during just-on instance.





### Note:

The 0x0f command will enable the bypass settings specified here as well as the settings specified by Watchdog 3 command set (0x41, 0x42). When both 0x11 and 0x0f commands are used, the later issued command will take precedence in system reset condition.

For example a system with the following settings:

#### 1. Using 0x41

```
./bpwd_tst -w -d 0x30 -c 0x41 -o 0x04 -M MB-887X [enter]
```

#### 2. Using 0x42

```
./bpwd_tst -w -d 0x30 -c 0x42 -o 0x01 -M MB-887X [enter]
```

#### 3. Using 0x11:

```
./bpwd_tst -w -d 0x30 -c 0x11 -o 0x01 -M MB-887X [enter]
```

#### 4. Using 0x0f:

```
./bpwd_tst -w -d 0x30 -c 0x0f -o 0x02 -M MB-887X [enter]
```

### VS.

#### 1. Using 0x41

```
./bpwd_tst -w -d 0x30 -c 0x41 -o 0x04 -M MB-887X [enter]
```

#### 2. Using 0x42

```
./bpwd_tst -w -d 0x30 -c 0x42 -o 0x01 -M MB-887X [enter]
```

#### 3. Using 0x0f:

```
./bpwd_tst -w -d 0x30 -c 0x0f -o 0x02 -M MB-887X [enter]
```

#### 4. Using 0x11:

```
./bpwd_tst -w -d 0x30 -c 0x11 -o 0x01 -M MB-887X [enter]
```

The system will behave like this:

| Condition  |  | Result                                      |
|--|--|---|
| System Reset or restart from a complete shutdown | When the system powers on:<br>Pair 2 bypass is enabled | Over 5 seconds:<br>Pair 3 bypass is enabled |
| Soft Reboot                                      | When the system powers on:<br>Pair 2 bypass is enabled | Over 5 seconds:<br>Pair 3 bypass is enabled |

vs.

| Condition  |  | Result                                      |
|--|--|---|
| System Reset or restart from a complete shutdown | When the system powers on:<br>Pair 1 bypass is enabled | Over 5 seconds:<br>Pair 3 bypass is enabled |
| Soft Reboot                                      | When the system powers on:<br>Pair 2 bypass is enabled | Over 5 seconds:<br>Pair 3 bypass is enabled |



## 4.3 Bypass/Disconnected Setting

The commands of bypass setting will let you query and configure the number of bypass pairs in each of the 3 instances, namely, System-off, Run-time, and Just-on (Just-on is the brief moment when the internal power supply turns on and booting process starts).

In the run-time and Just-on (Just-on is the brief moment when the internal power supply turns on and booting process starts) instances, the commands of disconnected setting are also offered for the fiber media to allow you querying and configuring the number of disconnected pairs.

### 4.3.1 Querying and Setting Bypass Status in System-off State



Note:

System-off is the state in which the system is powered off.

#### Command: 0x10

**Description:** Get system-off bypass status in either enable or disable. Bit 0~7 indicate that system-off bypass is enabled or disabled for that pair; 1 means enabled and 0 means disabled.

**Access mode:** Read

**Return value:**

LSB byte = Command | 0x80

MSB byte = System-off bypass status (The returned byte consists of LSB and MSB byte. The LSB byte signals whether uController responds with the command and it equals to 0x80 plus the value of command; the MSB byte is the value returned by the command). You can get bypass pairs setting by command 0x10 in system-off state, and the meaning of return value is listed as **Table 1**.

**<example>** In this example, let I2C address=0x30 and Model Name= MB-887X

```
./bpwd_tst -r -d 0x30 -c 0x10 -M MB-887X [enter]
```

```
READ ONLY ADDRESS:0x30 Command:0x10 ...OK, DATA = 0x0f90
```

**Interpretation:** The above result shows pairs 1, 2, 3 and 4 of bypass are enabled.

#### Command: 0x10

**Byte\_data:** 0x00~0x0f

**Description:** Set to enable or disable the system-off bypass. Bit 0~7 indicate to disable or enable system-off bypass for each pair; 1 means to enable whereas 0 means to disable.

**Access mode:** Write

**<example>** In this example, let I2C address=0x30 and Model Name= MB-887X

```
./bpwd_tst -w -d 0x30 -c 0x10 -o 0x0f -M MB-887X [enter]
```

```
WRITE ADDRESS:0x30 Command:0x10 DATA:0x0f...OK
```

**Explanation:** The above command demonstrates to enable pairs 1, 2, 3 and 4 of LAN bypass.

## 4.3.2 Querying and Setting Bypass Status in Just-on State (Just-on is the brief moment when the internal power supply turns on and booting process starts)

Just-on (Just-on is the brief moment when the internal power supply turns on and booting process starts) is the state in which the system is just powering on and CPU starts to run BIOS code until an OS is fully loaded.



### Note:

1. Soft reboot, i.e. restarting a computer under software control without shutdown or triggering a reset line, will not activate the bypass settings in JUST-ON. Instead, it will carry on the bypass and watchdog status from the preceding run-time instance. Therefore, it is required to perform any shut-down procedure before starting the system in order to go through JUST-ON.
2. The command in this section will only enable just-on bypass which satisfies the above condition; to enable the just-on bypass forcibly, see the section 4.2.13 Enabling Just-on Bypass by Simulating Just-on instance.

### Command: 0x11

**Description:** Get Just-on (Just-on is the brief moment when the internal power supply turns on and booting process starts) bypass status in either enable or disable, Bit 0~7 indicate Just-on (Just-on is the brief moment when the internal power supply turns on and booting process starts) bypass status is enabled or disabled for that pair; 1 means enabled whereas 0 means disabled.

**Access mode:** Read

**Return value:**

LSB byte = Command | 0x80

MSB byte = Just-on bypass status (The returned byte consists of LSB and MSB byte. The LSB byte signals whether uController responds with the command and it equals to 0x80 plus the value of command; the MSB byte is the value returned by the command). You can get bypass pairs setting by the command 0x11 in Just-on state, and the meaning of return value is listed as **Table 1**.

**<example>** In this example, let I2C address=0x30 and Model Name= MB-887X

```
./bpwd_tst -r -d 0x30 -c 0x11-M MB-887X [enter]
```

```
READ ONLY ADDRESS:0x30 Command:0x11 ...OK, DATA = 0x0791
```

**Explanation:** The above command shows that pairs 1, 2, and 3 of LAN bypass are enabled.

### Command: 0x11

**Byte\_data:** 0x00~0x0f

**Description:** Set to enable or disable the Just-on (Just-on is the brief moment when the internal power supply turns on and booting process starts) bypass, Bit 0~7 indicate to enable or disable bypass in Just-on state (Just-on is the brief moment when the internal power supply turns on and booting process starts) for each pair; 1 means to enable whereas 0 means to disable.

**Access mode:** Write

**<example>** In this example, let I2C address=0x30 and Model Name= MB-887X

```
./bpwd_tst -w -d 0x30 -c 0x11 -o 0x06 -M MB-887X [enter]
```

```
WRITE ADDRESS:0x30 Command:0x11 DATA:0x06...OK
```

**Explanation:** The above command demonstrates to enable pairs 2 and 3 of bypass.

### 4.3.3 Querying and Setting Bypass Status in Run-time State

Run-time is the state in which the system is fully up and is running an Operating System. And the system designer can use Lanner software to control bypass and watchdog functions during this time.

#### Command: 0x12

**Description:** Get run-time bypass status setting in either enabled or disabled, Bit 0~7 indicate run-time bypass status is enabled or disabled for that pair; 1 means enabled and 0 means disabled.

**Access mode:** Read

**Return value:**

LSB byte = Command | 0x80

MSB byte = Run-time bypass pairs enabled (The returned byte consists of LSB and MSB byte. The LSB byte signals whether uController responds with the command and it equals to 0x80 plus the value of command; the MSB byte is the value returned by the command)

You can get bypass pairs setting by the command 0x12 in run-time state, and the meaning of return value is listed as **Table 1**.

**<example>** In this example, let I2C address=0x30 and Model Name= MB-887X

```
./bpwd_tst -r -d 0x30 -c 0x12 -M MB-887X [enter]
```

```
READ ONLY ADDRESS:0x30 Command:0x12 ...OK, DATA = 0x0092
```

**Explanation:** The above command shows that none of the bypass pairs are enabled.

#### Command: 0x12

**Byte\_data:** 0x00~0x0f

**Description:** Set to enable or disable run-time bypass, Bit 0~7 indicate to enable or disable the run-time bypass for each pair; 1 means to enable whereas 0 means to disable.

**Access mode:** Write

**<example>** In this example, let I2C address=0x30 and Model Name= MB-887X

```
./bpwd_tst -w -d 0x30 -c 0x12 -o 0x00 -M MB-887X [enter]
```

```
WRITE ADDRESS:0x30 Command:0x12 DATA:0x00...OK
```

**Explanation:** The above command demonstrates not to enable any bypass pairs.

### 4.3.4 Querying and Setting Disconnection Status in Just-on State (Just-on is the brief moment when the internal power supply turns on and booting process starts)



**Note:**  
Only the fiber media supports this function.

Just-on (Just-on is the brief moment when the internal power supply turns on and booting process starts) is the state in which the system is just powering on and CPU starts to run BIOS code until an OS is fully loaded.

#### Command: 0x13

**Description:** Get Just-on (Just-on is the brief moment when the internal power supply turns on and booting process starts) disconnection status in either enable or disable, Bit 0~7 indicate Just-on (Just-on is the brief moment when the internal power supply turns on and booting process starts) disconnection status is enabled or disabled for that pair; 1 means enabled whereas 0 means disabled.

**Access mode:** Read

**Return value:**

LSB byte = Command | 0x80

MSB byte = Just-on disconnection status (The returned byte consists of LSB and MSB byte. The LSB byte signals whether uController responds with the command and it equals to 0x80 plus the value of command; the MSB byte is the value returned by the command). You can get disconnected pairs setting by the command 0x13 in Just-on state, and the meaning of return value is listed as

**Table 1.**

|  |
|--|
| <p><b>&lt;example&gt;</b> In this example, let I2C address=0x30 and Model Name= MB-887X</p> <pre>./bpwd_tst -r -d 0x30 -c 0x13 -M MB-887X [enter]</pre> <p>READ ONLY ADDRESS:0x30 Command:0x13 ...OK, DATA = 0x0793</p> <p><b>Explanation:</b> The above command shows that pairs 1, 2, and 3 of LAN disconnection function are enabled.</p> |
|--|

#### Command: 0x13

**Byte\_data:** 0x00~0x0f

**Description:** Set to enable or disable the Just-on (Just-on is the brief moment when the internal power supply turns on and booting process starts) disconnection function, Bit 0~7 indicate to enable or disable the disconnection function in Just-on state (Just-on is the brief moment when the internal power supply turns on and booting process starts) for each pair; 1 means to enable whereas 0 means to disable.

**Access mode:** Write

|  |
|--|
| <p><b>&lt;example&gt;</b> In this example, let I2C address=0x30 and Model Name= MB-887X</p> <pre>./bpwd_tst -w -d 0x30 -c 0x13 -o 0x06 -M MB-887X [enter]</pre> <p>WRITE ADDRESS:0x30 Command:0x13 DATA:0x06...OK</p> <p><b>Explanation:</b> The above command demonstrates to enable pairs 2 and 3 of LAN disconnection function.</p> |
|--|

## 4.3.5 Querying and Setting Disconnection Status in Run-Time State



**Note:**  
Only the fiber media supports this function.

Run-time is the state in which the system is fully up and is running an Operating System. And the system designer can use Lanner software to control disconnection and watchdog functions during this time.

### Command: 0x14

**Description:** Get run-time disconnection status setting in either enabled or disabled, Bit 0~7 indicate run-time disconnection status is enabled or disabled for that pair; 1 means enabled and 0 means disabled.

**Access mode:** Read

**Return value:**

LSB byte = Command | 0x80

MSB byte = Run-time disconnected pairs enabled (The returned byte consists of LSB and MSB byte. The LSB byte signals whether uController responds with the command and it equals to 0x80 plus the value of command; the MSB byte is the value returned by the command)

You can get disconnected pairs setting by the command 0x14 in run-time state, and the meaning of return value is listed as **Table 1**.

**<example>** In this example, let I2C address=0x30 and Model Name= MB-887X

```
./bpwd_tst -r -d 0x30 -c 0x14 -M MB-887X [enter]
```

```
READ ONLY ADDRESS:0x30 Command:0x14 ...OK, DATA = 0x0094
```

**Explanation:** The above command shows that none of the disconnected pairs are enabled.

### Command: 0x14

**Byte\_data:** 0x00~0x0f

**Description:** Set to enable or disable run-time disconnection function, Bit 0~7 indicate to enable or disable the run-time disconnection function for each pair; 1 means to enable whereas 0 means to disable.

**Access mode:** Write

**<example>** In this example, let I2C address=0x30 and Model Name= MB-887X

```
./bpwd_tst -w -d 0x30 -c 0x14 -o 0x00 -M MB-887X [enter]
```

```
WRITE ADDRESS:0x30 Command:0x14 DATA:0x00...OK
```

**Explanation:** The above command demonstrates not to enable any disconnected pairs.



## 4.4 Bypass/Disconnected Setting with Timer Control of Watchdog1 (in Run-time State)

The commands of watchdog1 timer will let you query the operating status of watchdog1, configure the bypass/disconnected pairs when it expires, and configure and query the timer interval as well as the time to expiration. This set of commands also offers a method to start and stop the watchdog1 timer.

The bypass function with watchdog1 is meant to provide a means to control the bypass status when system failures occur or requested by software in run-time state. These commands are meant to be embedded in the software which would monitor the system's operating status.

### 4.4.1 Querying Watchdog1 Running Status

**Command:** 0x20

**Description:** Get watchdog1 timer status.

**Access mode:** Read

**Return value:**

LSB byte = Command | 0x80

MSB byte = 0x00 (The returned byte consists of LSB and MSB byte. The LSB byte signals whether uController responds with the command and it equals to 0x80 plus the value of command; the MSB byte is the value returned by the command)

The return value has the following interpretation:

0= watchdog1 timer stopped.

1= watchdog1 timer is running, but not expired yet.

2= watchdog1 timer expired.

**<example>** In this example, let I2C address=0x30 and Model Name= MB-887X

```
./bpwd_tst -r -d 0x30 -c 0x20 -M MB-887X [enter]
```

```
READ ONLY ADDRESS:0x30 Command:0x20 ...OK, DATA = 0x02A0
```

**Interpretation:** The above example shows that watchdog1 is expired.

### 4.4.2 Setting Bypass Pairs When Watchdog1 Timer Expires

**Command:** 0x21

**Description:** Get the bypass pair that will be enabled when watchdog1 timer expires, Bit 0~7: pair bit mask.

**Access mode:** Read

**Return value:**

LSB byte = Command | 0x80

MSB byte = Bypass pairs that are enabled (The returned byte consists of LSB and MSB byte. The LSB byte signals whether uController responds with the command and it equals to 0x80 plus the value of command; the MSB byte is the value returned by the command).

**<example>** In this example, let I2C address=0x30 and Model Name= MB-887X

```
./bpwd_tst -r -d 0x30 -c 0x21 -M MB-887X [enter]
```

```
READ ONLY ADDRESS:0x30 Command:0x21 ...OK, DATA = 0x0fA1
```

**Interpretation:** The result shows that bypass pairs 1, 2, 3, and 4 will be enabled when watchdog1 expires.

**Command: 0x21****Byte\_data:** <pair bit mask>**Description:** Set which bypass pairs will be enabled when watchdog1 timer expires. The acceptable <pair bit mask> values are listed as **Table 1**.**Access mode:** Write

**<example>** In this example, let I2C address=0x30 and Model Name= MB-887X

```
./bpwd_tst -w -d 0x30 -c 0x21 -o 0x0f -M MB-887X [enter]
```

```
WRITE ADDRESS:0x30 Command:0x21 DATA:0x0f...OK
```

**Interpretation:** The example demonstrates to enable bypass pairs 1, 2, 3 and 4 when watchdog1 expires.

### 4.4.3 Querying and Setting Watchdog1 Timer



**Note:**

The timer interval indicates the value (in the metrics of level) at which the watchdog will start to count down, i.e. the time elapsed before the timer expires.

**Command: 0x22**

**Description:** Get timer setting of watchdog1.

**Access mode:** Read

**Return value:**

LSB byte = Command | 0x80

MSB byte = timer setting of watchdog1 (The returned byte consists of LSB and MSB byte. The LSB byte signals whether uController responds with the command and it equals to 0x80 plus the value of command; the MSB byte is the value returned by the command)

**<example>** In this example, let I2C address=0x30 and Model Name= MB-887X

```
./bpwd_tst -r -d 0x30 -c 0x22 -M MB-887X [enter]
```

```
READ ONLY ADDRESS:0x30 Command:0x22 ...OK, DATA = 0xffA2
```

**Interpretation:** The return value (MSB byte) shows the timer interval of watchdog 1 is 255 which is the value of 0xff in hexadecimal calculation. And it indicates 255 seconds (1 level = 1 second), which depends on your time unit setting.

For more details on this setting. Note the LSB=22+80=A0

**Command: 0x22**

**Byte\_data:** 0x00~0xff

**Description:** Set timer interval of watchdog1 timer, the data value is 00~ff (1 level = 1 second), range=0~255; 0 means to disable watchdog1 timer.

**Access mode:** Write

**<example>** In this example, let I2C address=0x30 and Model Name= MB-887X

```
./bpwd_tst -w -d 0x30 -c 0x22 -o 0x0f -M MB-887X [enter]
```

```
WRITE ADDRESS:0x30 Command:0x22 DATA:0x0f...OK
```

**Explanation:** The entered value is 0X0f which equals to 15 in hexadecimal calculation and it indicates 15 seconds.



## 4.4.4 Querying Time to Expiration of Watchdog1 Timer



**Note:**

The time to expiration indicates the amount of time (in the metrics of level) left before the watchdog expires.

**Command: 0x23**

**Description:** Get the time to expiration on watchdog1 timer, the return value is 0~255 (1 level = 1 second), range=0~255, 0 means disabling watchdog1 timer.

**Access mode:** Read

**Return value:**

LSB byte = Command | 0x80

MSB byte = time to expiration of watchdog1 (The returned byte consists of LSB and MSB byte. The LSB byte signals whether uController responds with the command and it equals to 0x80 plus the value of command; the MSB byte is the value returned by the command)

If the return is 0, it indicates watchdog1 timer is disabled (value obtained from 0x22 is 0), stopped (not counting down actively), or expired (after a successful completion of countdown)

**<example>** In this example, let I2C address=0x30 and Model Name= MB-887X

```
./bpwd_tst -r -d 0x30 -c 0x23 -M MB-887X [enter]
```

```
READ ONLY ADDRESS:0x30 Command:0x23 ...OK, DATA = 0x00A3
```

**Interpretation:** The above result shows that watchdog1 timer is not enabled.

## 4.4.5 Starting Watchdog1 Timer

**Command: 0x24**

**Byte\_data:** x (insignificant)

**Description:** Starting watchdog1 timer. When watchdog1 starts, 4 events will be triggered as listed below:

1. CPLD will stop watchdog3 (in Just-on state) upon receiving this command.
2. Reset watchdog1 timer's status to be stopped
3. Reset bypass pair back to be disabled
4. Start to count down

However, watchdog1 will not count down if any of the following situations is true:

- Watchdog1 timer setting = 0 (value obtained from command 0x22).
- The number of bypass pairs = 0 (value obtained from command 0x21).

**Access mode:** Write

**<example>** In this example, let I2C address=0x30 and Model Name= MB-887X

```
./bpwd_tst -w -d 0x30 -c 0x24 -o 0x00 -M MB-887X [enter] ( enter any value between 0x00 and 0xff, it is not significant)
```

```
WRITE ADDRESS:0x30 Command:0x24 DATA:0x00...OK
```



### Note:

If the computer turns on from a previous system-off state with a completion of shut-down procedure, the watchdog timer setting will reset to 0. Therefore, it is necessary to set the timer interval first (with the command 0x22) to enable the timer before executing the starting watchdog timer command. On the other hand, if the computer restarts by soft reboot procedure, i.e. restarting a computer under software control without removing power or (directly) triggering a reset line, the watchdog timer setting will remain as the previously set value.

## 4.4.6 Stopping Watchdog1 Timer in Run-time State

**Command:** 0x25

**Byte\_data:** x (insignificant)

**Description:** The CPLD will stop watchdog1 used in run-time state and watchdog3 used in Just-on state (Just-on is the brief moment when the internal power supply turns on and booting process starts) upon receiving this command.

**Access mode:** Write

|   |
|---|
| <p><b>&lt;example&gt;</b> In this example, let I2C address=0x30 and Model Name= MB-887X<br/>./bpwd_tst -w -d 0x30 -c 0x25 -o 0x00 [enter] -M MB-887X ( enter any value<br/>between 0x00 and 0xff, it is not significant)<br/>WRITE ADDRESS:0x30 Command:0x25 DATA:0x00...OK</p> |
|---|

## 4.4.7 Setting Watchdog1 Timeout Mode



### Note:

Only the fiber media supports this function.

**Command:** 0x26

**Byte\_data:** 0x00~0x01

**Description:** Change the timeout mode for the Watchdog1 timer between “bypass” and “disconnect” (0 = bypass, 1 = disconnect).

**Access mode:** Write

|  |
|--|
| <p><b>&lt;example&gt;</b> In this example, let I2C address=0x30 and Model Name= MB-887X<br/>./bpwd_tst -w -d 0x30 -c 0x26 -o 0x01 [enter] -M MB-887X<br/>WRITE ADDRESS:0x30 Command:0x26 DATA:0x01...OK<br/><b>Interpretation:</b> The result shows that the timeout mode of Watchdog1 is set as<br/>“disconnect”.</p> |
|--|

## 4.4.8 Setting Disconnected Pairs When Watchdog1 Timer Expires



**Note:**  
Only the fiber media supports this function.

### Command: 0x27

**Description:** Get the disconnected pair that will be enabled when watchdog1 timer expires, Bit 0~7: pair bit mask.

**Access mode:** Read

**Return value:**

LSB byte = Command | 0x80

MSB byte = Disconnected pairs that are enabled (The returned byte consists of LSB and MSB byte. The LSB byte signals whether uController responds with the command and it equals to 0x80 plus the value of command; the MSB byte is the value returned by the command)

**< example >** In this example, let I2C address=0x30 and Model Name= MB-887X  
 ./bpwd\_tst -r -d 0x30 -c 0x27 -M MB-887X [enter]  
 READ ONLY ADDRESS:0x30 Command:0x27 ...OK, DATA = 0x0fA7  
**Interpretation:** The result shows that disconnected pairs 1, 2, 3, and 4 will be enabled when watchdog1 expires.

### Command: 0x27

**Byte\_data:** <pair bit mask>

**Description:** Set which disconnected pairs will be enabled when watchdog1 timer expires. The acceptable <pair bit mask> values are listed as **Table 1**.

**Access mode:** Write

**< example >** In this example, let I2C address=0x30 and Model Name= MB-887X  
 ./bpwd\_tst -w -d 0x30 -c 0x27 -o 0x0f -M MB-887X [enter]  
 WRITE ADDRESS:0x30 Command:0x27 DATA:0x0f...OK  
**Interpretation:** The example demonstrates to enable disconnected pairs 1, 2, 3 and 4 when watchdog1 expires.

## 4.5 Bypass/Disconnected Setting with Timer Control of Watchdog2 (in Run-time State)

The commands of watchdog2 timer will let you query the operating status of watchdog2, configure the bypass/disconnected pairs when it expires, and configure and query the timer interval as well as the time to expiration. This set of commands also offers a method to start and stop the watchdog2 timer.

The bypass function with watchdog2 provides a means to control the bypass status when system failures occur or requested by software in run-time state. These commands are meant to be embedded in the software which would monitor the system's operating status.

### 4.5.1 Querying Watchdog2 Running Status

**Command:** 0x30

**Description:** Get watchdog2 timer status.

**Access mode:** Read

**Return value:**

LSB byte = Command | 0x80

MSB byte = Watchdog2 running status (The returned byte consists of LSB and MSB byte. The LSB byte signals whether uController responds with the command and it equals to 0x80 plus the value of command; the MSB byte is the value returned by the command).

The return value has the following interpretations:

0= watchdog2 timer stopped.

1= watchdog2 timer running, but not expired yet.

2= watchdog2 timer expired.

**<example>** In this example, let I2C address=0x30 and Model Name= MB-887X

```
./bpwd_tst -r -d 0x30 -c 0x30 -M MB-887X [enter]
```

```
READ ONLY ADDRESS:0x30 Command:0x30 ...OK, DATA = 0x00B0
```

**Interpretation:** The above result shows that the watchdog2 timer is stopped.

### 4.5.2 Setting Bypass Pairs When Watchdog2 Timer Expires

**Command:** 0x31

**Description:** Get the bypass pair that will be enabled when watchdog2 timer expires, Bit 0~7: pair bit mask.

**Access mode:** Read

**Return value:**

LSB byte = Command | 0x80

MSB byte = Bypass pairs that are enabled (The returned byte consists of LSB and MSB byte. The LSB byte signals whether uController responds with the command and it equals to 0x80 plus the value of command; the MSB byte is the value returned by the command).

**<example>** In this example, let I2C address=0x30 and Model Name= MB-887X

```
./bpwd_tst -r -d 0x30 -c 0x31 -M MB-887X [enter]
```

```
READ ONLY ADDRESS:0x30 Command:0x31 ...OK, DATA = 0x0fB1
```

**Interpretation:** The result shows that bypass pairs 1, 2, 3, and 4 will be enabled when watchdog2 expires.

**Command: 0x31****Byte\_data:** <pair bit mask>**Description:** Set which bypass pairs will be enabled when watchdog2 timer expires. The acceptable <pair bit mask> values are listed as **Table 1**.**Access mode:** Write

**<example>** In this example, let I2C address=0x30 and Model Name= MB-887X  
 ./bpwd\_tst -w -d 0x30 -c 0x31 -o 0x0f -M MB-887X [enter]  
 WRITE ADDRESS:0x30 Command:0x31 DATA:0x0f...OK  
**Explanation:** The above example demonstrates to enable bypass pairs 1, 2, 3, and 4 when watchdog 2 expires.

### 4.5.3 Querying and Setting Watchdog2

**Note:**

The timer interval indicates the value (in the metrics of level) at which the watchdog will start to count down, i.e. the time elapsed before the timer expires.

**Command: 0x32****Description:** Get timer setting of watchdog 2.**Access mode:** Read

Return value:

LSB byte = Command | 0x80

MSB byte = Timer settings in seconds (The returned byte consists of LSB and MSB byte. The LSB byte signals whether uController responds with the command and it equals to 0x80 plus the value of command; the MSB byte is the value returned by the command)

**<example>** In this example, let I2C address=0x30 and Model Name= MB-887X  
 ./bpwd\_tst -r -d 0x30 -c 0x32 -M MB-887X [enter]  
 READ ONLY ADDRESS:0x30 Command:0x32 ...OK, DATA = 0x0fb2  
**Interpretation:** The above result shows the timer interval of watchdog2 is 15 which is the value of 0x0f in hexadecimal calculation. And it indicates 15 seconds (1 level = 1 second), which depends on your time unit setting.

### Command: 0x32

**Byte\_data:** 0x00~0xff

**Description:** Set the timer interval of watchdog2 timer, the data value is 00~ff (1 level = 1 second), range=0~255, 0 means to disable watchdog2 timer.

**Access mode:** Write

**<example>** In this example, let I2C address=0x30 and Model Name= MB-887X

```
./bpwd_tst -w -d 0x30 -c 0x32 -o 0x00 -M MB-887X [enter]
```

```
WRITE ADDRESS:0x30 Command:0x32 DATA:0x00...OK
```

**Explanation:** The above example shows to disable the Watchdog2 timer.

## 4.5.4 Querying Time to Expiration of Watchdog2 Timer



**Note:**

The time to expiration indicates the amount of time (in the metrics of level) left before the watchdog expires.

### Command: 0x33

**Description:** Get the time to expiration on watchdog2 timer, the return value is 0~255 (1 scale = 1 second), range=0~255, 0 means that watchdog2 timer is disabled.

**Access mode:** Read

**Return value:**

LSB byte = Command | 0x80

MSB byte = Time to expiration (The returned byte consists of LSB and MSB byte. The LSB byte signals whether uController responds with the command and it equals to 0x80 plus the value of command; the MSB byte is the value returned by the command)

If the return value=0, it means watchdog2 timer is stopped (not counting down actively), disabled (value obtained from 0x32 is 0), or expired (after successful completion of countdown activity).

**<example>** In this example, let I2C address=0x30 and Model Name= MB-887X

```
./bpwd_tst -r -d 0x30 -c 0x33 -M MB-887X [enter]
```

```
READ ONLY ADDRESS:0x30 Command:0x33 ...OK, DATA = 0xffB3
```

**Interpretation:** The return value shows that the time to expiration is 255 seconds.



## 4.5.5 Starting Watchdog2 Timer

**Command:** 0x34

**Byte\_data:** x (insignificant)

**Description:** Starting watchdog2 timer. When watchdog2 starts, 4 events will be triggered:

1. CPLD will stop watchdog3 used in Just-on state (Just-on is the brief moment when the internal power supply turns on and booting process starts) upon receiving this command.
2. Reset watchdog2 timer
3. Reset bypass pair (can be set by command 0x31) back to be disabled
4. Start to count down

However, watchdog2 will not count down if any of the following situations is true:

- Watchdog2 timer setting = 0 (value obtained from command 0x32).
- The number of bypass pairs = 0 (value obtained from command 0x31)

**Access mode:** Write

**<example>** In this example, let I2C address=0x30 and Model Name= MB-887X  
 ./bpwd\_tst -w -d 0x30 -c 0x34 -o 0x00 -M MB-887X [enter] ( enter any value  
 between 0x00 and 0xff, it is not significant)  
 WRITE ADDRESS:0x30 Command:0x34 DATA:0x00...OK



### Note:

If the computer turns on from a previous system-off state with a completion of shut-down procedure, the watchdog timer setting will reset to 0. Therefore, it is necessary to set the timer interval first (with the command 0x32) to enable the timer before executing the starting watchdog timer command. On the other hand, if the computer restarts by soft reboot procedure, i.e. restarting a computer under software control without removing power or (directly) triggering a reset line, the watchdog timer setting will remain as the previously set value.

## 4.5.6 Stopping Watchdog2 Timer in Run-time State

**Command:** 0x35

**Byte\_data:** x (insignificant)

**Description:** The CPLD will stop watchdog2 used in run-time state and watchdog3 used in Just-on state (Just-on is the brief moment when the internal power supply turns on and booting process starts) upon receiving this command.

**Access mode:** Write

**<example>** In this example, let I2C address=0x30 and Model Name= MB-887X  
 ./bpwd\_tst -w -d 0x30 -c 0x35 -o 0x00 -M MB-887X [enter] ( enter any value  
 between 0x00 and 0xff, it is not significant)  
 WRITE ADDRESS:0x30 Command:0x35 DATA:0x00...OK



## 4.5.7 Setting Watchdog2 Timeout Mode



**Note:**  
Only the fiber media supports this function.

**Command:** 0x36

**Byte\_data:** 0x00~0x01

**Description:** Change the timeout mode for the Watchdog2 timer between “bypass” and “disconnect” (0 = bypass, 1 = disconnect).

**Access mode:** Write

**< example >** In this example, let I2C address=0x30 and Model Name= MB-887X

```
./bpwd_tst -w -d 0x30 -c 0x36 -o 0x01 [enter] -M MB-887X
```

```
WRITE ADDRESS:0x30 Command:0x36 DATA:0x01...OK
```

**Interpretation:** The result shows that the timeout mode of Watchdog 2 is set as “disconnect”.

## 4.5.8 Setting Disconnected Pairs When Watchdog2 Timer Expires



**Note:**  
Only the fiber media supports this function.

**Command:** 0x37

**Description:** Get the disconnected pair that will be enabled when watchdog2 timer expires, Bit 0~7: pair bit mask.

**Access mode:** Read

**Return value:**

LSB byte = Command | 0x80

MSB byte = Disconnected pairs that are enabled (The returned byte consists of LSB and MSB byte. The LSB byte signals whether uController responds with the command and it equals to 0x80 plus the value of command; the MSB byte is the value returned by the command)

**< example >** In this example, let I2C address=0x30 and Model Name= MB-887X

```
./bpwd_tst -r -d 0x30 -c 0x37 -M MB-887X [enter]
```

```
READ ONLY ADDRESS:0x30 Command:0x37 ...OK, DATA = 0x0fB7
```

**Interpretation:** The result shows that disconnected pairs 1, 2, 3, and 4 will be enabled when watchdog2 expires.

**Command:** 0x37

**Byte\_data:** <pair bit mask>

**Description:** Set which disconnected pairs will be enabled when watchdog2 timer expires. The acceptable <pair bit mask> values are listed as **Table 1**.

**Access mode:** Write

< **example**> In this example, let I2C address=0x30 and Model Name= MB-887X

```
./bpwd_tst -w -d 0x30 -c 0x37 -o 0x0f -M MB-887X [enter]
```

```
WRITE ADDRESS:0x30 Command:0x37 DATA:0x0f...OK
```

**Interpretation:** The example demonstrates to enable disconnected pairs 1, 2, 3 and 4 when watchdog2 expires.

Learnnet

## 4.6 Bypass/Disconnected Setting with Timer Control of Watchdog3 (in Just-on State)

The commands of watchdog3 timer category will let you query the operating status of watchdog3, configure the bypass/disconnected pairs when it expires, and configure and query the timer interval as well as the time to expiration. The commands also offer a method to start and stop the watchdog3 timer.

Unlike watchdog1 and watchdog2 which are used to detect system anomalies during run-time, watchdog 3 becomes active since just-on and will start countdown as soon as the system is powered up. These following commands in this section are designed to be embedded in the program which would reset the system when certain conditions are met.



**Note:**

1. Soft reboot, i.e. restarting a computer under software control without shutdown or triggering a reset line, will not enable the bypass settings in JUST-ON. Instead, it will carry on the bypass and watchdog status from the preceding run-time instance. Therefore, it is required to perform any shut-down procedure before starting the system in order to go through Just-on state.
2. The command in this section will only enable just-on bypass which satisfies the above condition; to force to enable the Watchdog3 in run-time state, see Section 4.2.13 Enabling Just-on Bypass by Simulating Just-on Instance.

### 4.6.1 Querying Watchdog3 Running Status

**Command:** 0x40

**Description:** Get watchdog3 timer status.

**Access mode:** Read

**Return value:**

LSB byte = Command | 0x80

MSB byte = Watchdog3 running status (The returned byte consists of LSB and MSB byte. The LSB byte signals whether uController responds with the command and it equals to 0x80 plus the value of command; the MSB byte is the value returned by the command).

The return value has the following interpretations:

**0=** watchdog3 timer stopped.

**1=** watchdog3 timer running, but not expired yet.

**2=** watchdog3 timer expired.

**<example>** In this example, let I2C address=0x30 and Model Name= MB-887X

```
./bpwd_tst -r -d 0x30 -c 0x40 -M MB-887X [enter]
```

```
READ ONLY ADDRESS:0x30 Command:0x40 ...OK, DATA = 0x00C0
```

**Interpretation:** The above result shows that watchdog3 timer has been stopped.

## 4.6.2 Setting Bypass Pairs When Watchdog3 Timer Expires

### Command: 0x41

**Description:** Get the bypass pairs that will be enabled when watchdog3 timer expires, Bit 0~7: pair bit mask.

**Access mode:** Read

**Return value:**

LSB byte = Command | 0x80

MSB byte = Bypass pairs that are enabled (The returned byte consists of LSB and MSB byte. The LSB byte signals whether uController responds with the command and it equals to 0x80 plus the value of command; the MSB byte is the value returned by the command).

**<example>** In this example, let I2C address=0x30 and Model Name= MB-887X

```
./bpwd_tst -r -d 0x30 -c 0x41 -M MB-887X [enter]
```

```
READ ONLY ADDRESS:0x30 Command:0x41 ...OK, DATA = 0x0fC1
```

**Interpretation:** The above result shows that bypass pairs 1, 2, 3, and 4 will be enabled when watchdog3 expires.

### Command: 0x41

**Byte\_data:** <pair bit mask>

**Description:** Set which bypass pairs will be enabled when watchdog3 timer expires. The acceptable <pair bit mask> values are listed as **Table 1**.

**Access mode:** Write

**<example>** In this example, let I2C address=0x30 and Model Name= MB-887X

```
./bpwd_tst -w -d 0x30 -c 0x41 -o 0x07 -M MB-887X [enter]
```

```
WRITE ADDRESS:0x30 Command:0x41 DATA:0x07...OK
```

**Explanation:** The above result demonstrates to enable bypass pairs 1, 2, and 3 when watchdog3 timer expires.

## 4.6.3 Querying and Setting Watchdog3 Timer



**Note:**

The timer interval indicates the value (in the metrics of level) at which the watchdog will start to count down, i.e. the time elapsed from the moment the system is powered up until the timer expires.

### Command: 0x42

**Description:** Get timer setting of watchdog 3.

**Access mode:** Read

**Return value:**

LSB byte = Command | 0x80

MSB byte = timer setting of watchdog3 (The returned byte consists of LSB and MSB byte. The LSB byte signals whether uController responds with the command and it equals to 0x80 plus the value of command; the MSB byte is the value returned by the command)

If return value= 0x00, it means that watchdog3 timer is disabled.

|   |
|---|
| <p><b>&lt;example&gt;</b> In this example, let I2C address=0x30 and Model Name= MB-887X</p> <pre>./bpwd_tst -r -d 0x30 -c 0x42 -M MB-887X [enter]</pre> <p>READ ONLY ADDRESS:0x30 Command:0x42 ...OK, DATA = 0x0f C2</p> <p><b>Interpretation:</b> The above result shows that the timer interval of watchdog 3 is 15x5 (75) seconds.</p> |
|---|

### Command: 0x42

**Byte\_data:** 0x00~0xff

**Description:** Set the timer interval of watchdog3 timer, the data value is 00~ff (1 level = 5 second), range=0~255, 0 means to disable watchdog3 timer.

**Access mode:** Write

|   |
|---|
| <p><b>&lt;example&gt;</b> In this example, let I2C address=0x30 and Model Name= MB-887X</p> <pre>./bpwd_tst -w -d 0x30 -c 0x42 -o 0x0f -M MB-887X [enter]</pre> <p>WRITE ADDRESS:0x30 Command:0x42 DATA:0x0f...OK</p> <p><b>Explanation:</b> The above example demonstrates to set the timer interval of watchdog3 to be 15x5 (75) seconds.</p> |
|---|

## 4.6.4 Querying Time to Expiration of Watchdog3 Timer

### Note:



The time to expiration indicates the amount of time (in the metrics of level) left before watchdog expires.

### Command: 0x43

**Description:** Get the time to expiration of watchdog3 timer, the return value is 0~255 (1 level = 5 second), range=0~255, 0 means watchdog3 timer is disabled.

**Access mode:** Read

**Return value:**

LSB byte = Command | 0x80

MSB byte = Time to expiration of watchdog3 timer (The returned byte consists of LSB and MSB byte. The LSB byte signals whether uController responds with the command and it equals to 0x80 plus the value of command; the MSB byte is the value returned by the command)

If return value=0, it means watchdog3 timer is disabled (value obtained from command 0x42), stopped (not counting down actively) or expired (after a successful completion of countdown).

|   |
|---|
| <p><b>&lt;example&gt;</b> In this example, let I2C address=0x30 and Model Name= MB-887X</p> <pre>./bpwd_tst -r -d 0x30 -c 0x43 -M MB-887X [enter]</pre> <p>READ ONLY ADDRESS:0x30 Command:0x43 ...OK, DATA = 0x0fC3</p> <p><b>Interpretation:</b> The above result shows that the time to expiration of watchdog3 is 15x5 (75) seconds.</p> |
|---|

## 4.6.5 Stopping Watchdog3 Timer

**Command:** 0x45

**Byte\_data:** x (insignificant)

**Description:** The CPLD will stop watchdog3 used in Just-on state (Just-on is the brief moment when the internal power supply turns on and booting process starts) upon receiving this command.

**Access mode:** Write

**<example>** In this example, let I2C address=0x30 and Model Name= MB-887X  
 ./bpwd\_tst -w -d 0x30 -c 0x45 -o 0x00 -M MB-887X [enter] ( enter any value  
 between 0x00 and 0xff, it is not significant)  
 WRITE ADDRESS:0x30 Command:0x45 DATA:0x00...OK

## 4.6.6 Setting Watchdog3 Timeout Mode



**Note:**

Only the fiber media supports this function.

**Command:** 0x46

**Byte\_data:** 0x00~0x01

**Description:** Change the timeout mode for the Watchdog3 timer between “bypass” and “disconnect” (0 = bypass, 1 = disconnect).

**Access mode:** Write

**<example>** In this example, let I2C address=0x30 and Model Name= MB-887X  
 ./bpwd\_tst -w -d 0x30 -c 0x46 -o 0x01 [enter] -M MB-887X  
 WRITE ADDRESS:0x30 Command:0x46 DATA:0x01...OK  
**Interpretation:** The result shows that the timeout mode of Watchdog 3 is set as  
 “disconnect”.

## 4.6.7 Setting Disconnected Pairs When Watchdog3 Timer Expires



**Note:**  
Only the fiber media supports this function.

### Command: 0x47

**Description:** Get the disconnected pair that will be enabled when watchdog3 timer expires, Bit 0~7: pair bit mask.

**Access mode:** Read

**Return value:**

LSB byte = Command | 0x80

MSB byte = Disconnected pairs that are enabled (The returned byte consists of LSB and MSB byte. The LSB byte signals whether uController responds with the command and it equals to 0x80 plus the value of command; the MSB byte is the value returned by the command)

**<example>** In this example, let I2C address=0x30 and Model Name= MB-887X

```
./bpwd_tst -r -d 0x30 -c 0x47 -M MB-887X [enter]
```

```
READ ONLY ADDRESS:0x30 Command:0x47 ...OK, DATA = 0x0fC7
```

**Interpretation:** The result shows that disconnected pairs 1, 2, 3, and 4 will be enabled when watchdog3 expires.

### Command: 0x47

**Byte\_data:** <pair bit mask>

**Description:** Set which disconnected pairs will be enabled when watchdog3 timer expires. The acceptable <pair bit mask> values are listed as **Table 1**.

**Access mode:** Write

**<example>** In this example, let I2C address=0x30 and Model Name= MB-887X

```
./bpwd_tst -w -d 0x30 -c 0x47 -o 0x0f -M MB-887X [enter]
```

```
WRITE ADDRESS:0x30 Command:0x47 DATA:0x0f...OK
```

**Interpretation:** The example demonstrates to enable disconnected pairs 1, 2, 3 and 4 when watchdog3 expires.



**Table 1**

| Hex  | Binary    | Description                            |
|------|-----------|--|
| 0x00 | 00000000b | Means none of the pairs is enabled.    |
| 0x01 | 00000001b | Means pair 1 is enabled.               |
| 0x02 | 00000010b | Means pair 2 is enabled.               |
| 0x03 | 00000011b | Means pairs 1 and 2 are enabled.       |
| 0x04 | 00000100b | Means pair 3 is enabled.               |
| 0x05 | 00000101b | Means pairs 1 and 3 are enabled.       |
| 0x06 | 00000110b | Means pairs 2 and 3 are enabled.       |
| 0x07 | 00000111b | Means pairs 1, 2 and 3 are enabled.    |
| 0x08 | 00001000b | Means pair 4 is enabled.               |
| 0x09 | 00001001b | Means pairs 1 and 4 are enabled.       |
| 0x0a | 00001010b | Means pairs 2 and 4 are enabled.       |
| 0x0b | 00001011b | Means pairs 1, 2 and 4 are enabled.    |
| 0x0c | 00001100b | Means pairs 3 and 4 are enabled.       |
| 0x0d | 00001101b | Means pairs 1, 3 and 4 are enabled.    |
| 0x0e | 00001110b | Means pairs 2, 3 and 4 are enabled.    |
| 0x0f | 00001111b | Means pairs 1, 2, 3 and 4 are enabled. |

## Using Generation 1 and Generation 2 Bypass

---

### 5.1 Generation 1 Bypass

In Generation 1, bypass is controlled by jumpers on the motherboard.



**Note:**

The jumper settings vary depending on product models; refer to the product's manual to enable the bypass.

### 5.2 Generation 2 Bypass

In Generation 2, bypass settings can be configured by BIOS or programming the GPIO with the use of watchdog timer (WDT).

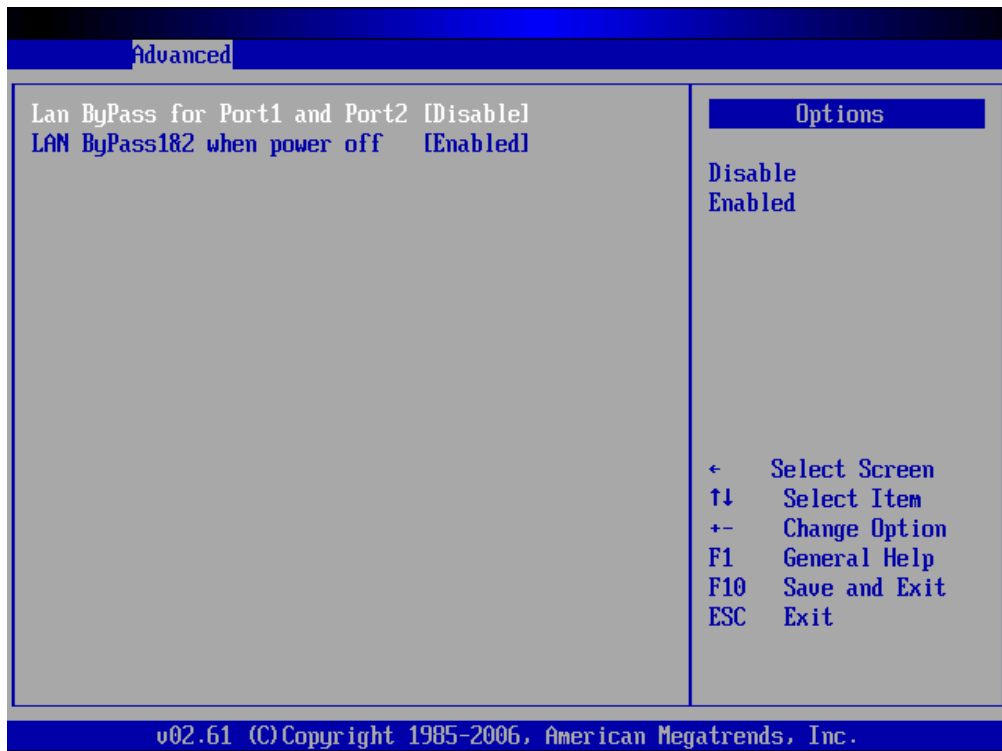
Examples of bypass setting in Generation 2:

The LAN bypass can be turned on or off in two system states, i.e. power on and power off. The following are the BIOS menu and illustration of the possibilities of LAN bypass configuration in each state.



**Note:**

- (1) The BIOS menu may not look the same on all the products; the following just gives the principle of bypass setting.
- (2) A watchdog timer can be used to control the LAN Bypass function dynamically by programming. Lanner also provides sample code for bypass control with WDT via programming. For a reference utility that contains sample code for LAN Bypass function programming, please contact Lanner's technical support.



|                                 |                                 |            |                                      |
|---------------------------------|---------------------------------|------------|--------------------------------------|
| Bypass Settings \ System Status | LAN Bypass for Port1 and Port 2 |            | LAN Bypass 1&2 when <i>power off</i> |
|                                 | Enabled                         | Disabled   |                                      |
| PWR ON                          | Bypass                          | Non-Bypass | Enabled                              |
| PWR OFF                         | Bypass                          | Bypass     |                                      |

|                                 |                                 |            |                                      |
|---------------------------------|---------------------------------|------------|--------------------------------------|
| Bypass Settings \ System Status | LAN Bypass for Port1 and Port 2 |            | LAN Bypass 1&2 when <i>power off</i> |
|                                 | Enabled                         | Disabled   |                                      |
| PWR ON                          | Non-Bypass                      | Non-Bypass | Disabled                             |
| PWR OFF                         | Non-Bypass                      | Non-Bypass |                                      |

**Lanner**

**Lanner Electronics Inc.**  
**7F, No.173, Sec.2, Datong Rd.**  
**XiZhi District, New Taipei City 221, Taiwan**  
**Tel: +886-2-86926060**  
**Fax: +886-2-86926103**  
**[www.lannerinc.com](http://www.lannerinc.com)**