# BH3
# TECHNICAL NOTE

(Last updated on 2020/04/27)
(Version：1.6)

BiZright Technology Inc.

## 1. About this product

Thank you for purchasing BH series / BH3.　This product is a set embedded the following items.
・Metal case with heat radiation pad
・Raspberry Pi3 Model B
・BiZright original circuit board
・AC adaptor（7.5V3A）

※*microSD card sold separately

## 2. Overview of BH3

BH3 is general-purpose embedded STB using Raspberry Pi3 Model B which has up-graded its functions as changing our reputable device (BH2) releasing assigned GPIO to the general-purpose switch.

Raspberry Pi has been especially adopted in IoT field in recent; however there are lots of problems that need to be cleared to run in the actual environment after finishing prototype as normal shut down in power off, power breaker interlock, system time retention, stabilization of supper supply, countermeasures of noise / electrostatic, watch dog timer.

BH3 was especially developed focusing on robustness STB that can be used in the field.

## 3. Simple UPS function by electric double layer capacitors

As RaspberryPi works under Linux base OS, it needs to be operated with adequate shutdown-process.

BH3 detects a loss of power-supply and generates the interrupt signal to Raspberry Pi's CPU.

BH3 has a simple UPS function using electric double layer capacitors running under no power operation about 30 sec. (depending on the environment) , once the shutdown occurs that makes safely shutdown process possible and drastically reduce the possibility of OS crush / Micro SD card crush. BH3 realizes adopting RaspberryPi for practical situations where industrial single board computers have been used formerly.

## 4. Real Time Clock

Most of the applications as data-logging require the accurate date and time information. BH3 is equipped with battery-backed RTC.

## 5. Stabilization of Power Supply

Raspberry Pi normally supplies the power from micro USB, yet the power supply may become unstable due to the problem of internal resistance in connector when its current consumption increases. However, this problem does not occur since BH3 supplies the power from the I/O bus port. Moreover, BH3 secures the stable operating voltage because it has a power regulator inside.

## 6. Heat Radiation measures of CPU

BH3 is equipped with the heat conduction pad on the case and adheres it to the CPU to promote heat radiation more effectively. Since it has passed the operation test under the external environment of 50℃, it can be used with confidence in the field.

## 7. Noise/Electrostatic measures

Countermeasures such as static electricity, lightning surge and GPIO line noise are taken by using the metal case and the signal grounding.

## 8. Power off support

High-speed shutdown is possible by receiving the signal of power-off detection (GPIO23) at power off. It is possible to execute any commands (such as original programs) at the time of high-speed shutdown.

## 9. Activation of Watch Dog Timer

Enables the Watch Dog Timer mounted on Raspberry Pi to reboot the CPU by the hardware layer even if the CPU goes out of control. It is possible to reduce the possibility of the system hung up.

## 10. Hardware specification

| | |
|---|---|
| OS | Raspbian Jessie or later.<br>（BH set up tools：supplied separately) |
| CPU | Raspberry Pi3 Model B<br>1.2GHz / 64-bit quad-core ARMv8 CPU |
| GPU | 250 MHz / Broadcom VideoCore IV |
| Memory | LPDDR2 SDRAM 1GB |
| RTC | MAXIM DS3231 I2C connection（pulled up at 10KΩ）<br>Used GPIO2, 3 |
| UPS | Electric double layer capacitors |
| Video output | HDMI (1.3 / 1.4) |
| Audio output | 3.5mm / HDMI |
| USB port | USB2.0×4 |
| I/O connector | 40 pin (Pin assignment is the same as original) |
| Network | 10 / 100Mbps Ethernet<br>802.11n Wireless LAN |
| Bluetooth | Bluetooth 4.1<br>Bluetooth Low Energy (BLE) |
| Card slot | Micro SD Card slot |
| Storage | Depends on MicroSD card |
| Power Supply | AC100-240V、50/60Hz（When using dedicated AC adaptor）DC 7.5V 3A |
| Dimensions | (H)32mm x (L) 115mm×(W)115mm |
| Weight | 420g（Without AC adaptor） |
| Operation guarantee temperature | 0～40 degree（No-conclusion -4 dews） |

| | |
|---|---|
| Operation guarantee humidity | 10%～80% |
| Power off detection | GPIO23　L level |
| Reset control | GPIO18 |
| General-purpose switch | Equipped with a center, up, down, left and right momentary switches. |

## 11. Test result

Testing Location：Local Independent Administrative Institution
Hokkaido Research Institution Industrial Testing Site
11-chone 19 Jyo Nishi, Kita-ku, Kita, Sapporo, Hokkaido

Lighting Surge Tolerance Test ： Testing Equipment ： Noise Research Laboratory LSS-15AX-C3

| | |
|---|---|
| Test Contents | 1KV of lightning surge one time positive and negative is applied from AC line in 5 times at 20 sec. intervals; total 10 times. |
| Test Result | Normal |

| | |
|---|---|
| Test Contents | 2KV of lightning surge one time positive and negative is applied from AC line in 5 times at 20 sec. intervals; total 10 times. |
| Test Result | Normal |

Static electricity tolerance test ： Testing Equipment ： Noise Research Laboratory ESS-2000 & Discharge gun TC=815R

| | |
|---|---|
| Test Contents | Indirect Discharge (Discharge 10 cm away from the devise) |
| Test | 8KV * 20 times: Normal |

| | |
|---|---|
| Result | 15KV * 20 times: Normal |

| | |
|---|---|
| Test Contents | Direct Discharge(Discharge by contacting the discharge gun directly to the devise) |
| Test Result | 4KV * 20 times: Normal<br>8KV * 20 times: Normal |

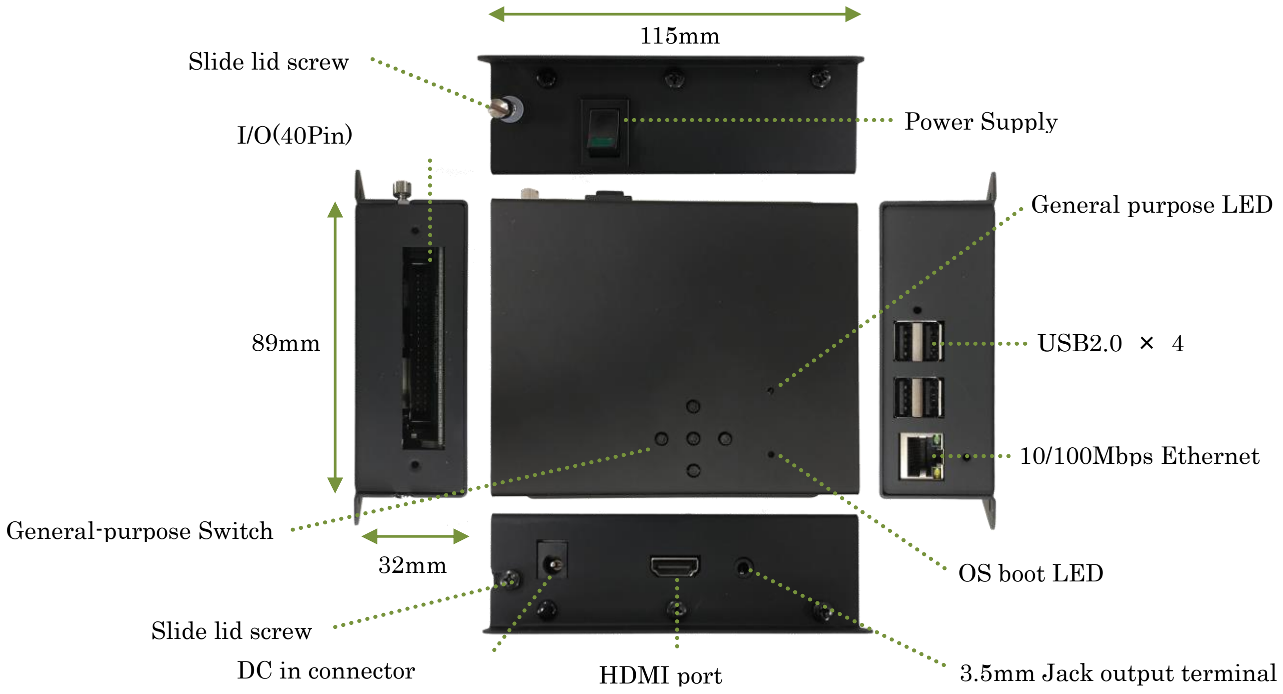
However, the phenomenon has happened that the surface of the monitor was disturbed for a moment and got it back to normal within one second in both patterns during the discharge.　It is presumed as the cause that the synchronization of HDMI is lost for a moment due to the noise caused by the high voltage pulse.

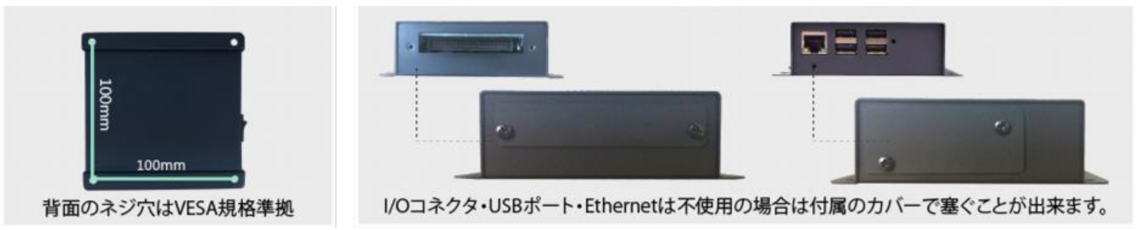
Low temperature constant temperature and humidity test for electronic appliances : Test Equipment : Espec PL-4KP

| | |
|---|---|
| Test Contents | Operate the devise at 0℃ for 24 hours. |
| Test Result | Normal |

| | |
|---|---|
| Test Contents | Operate the device at 50℃ for 24 hours. |
| Test Result | Normal |

## 12. Dimensions and part of each name



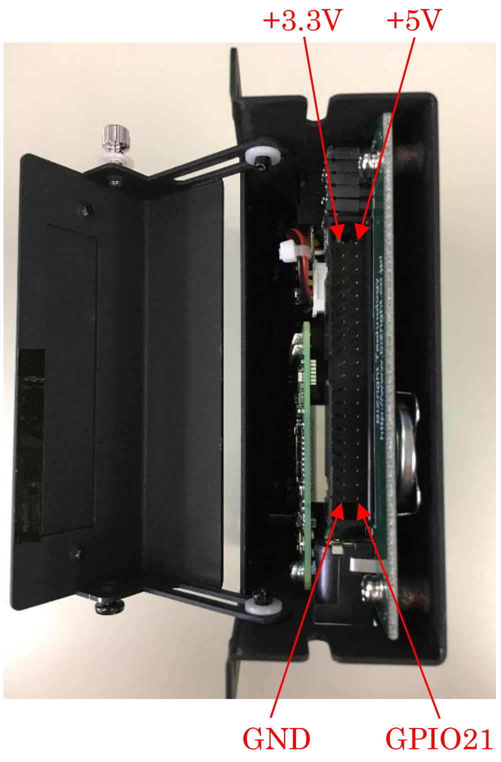

100mm
100mm
背面のネジ穴はVESA規格準拠
I/Oコネクタ・USBポート・Ethernetは不使用の場合は付属のカバーで塞ぐことが出来ます。

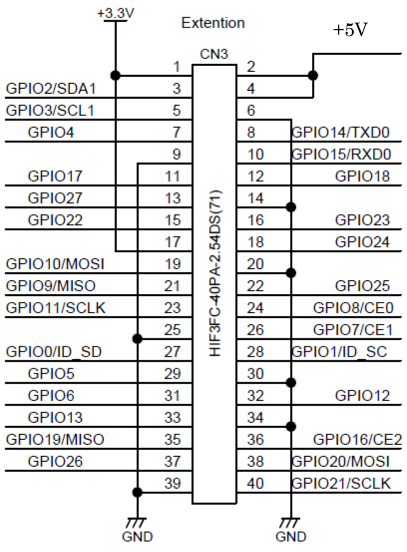
Screw hole of back side conforms to VESA standard.
I/O connector/USB port/Ethernet can be closed with the attached lid.

Open the slide lid and insert the Micro SD card directly into the Raspberry Pi.

## 13. Access to I/O connector and GPIO port

Can access to the I/O connector by removing the slide lid or the lid of the I/O connector.



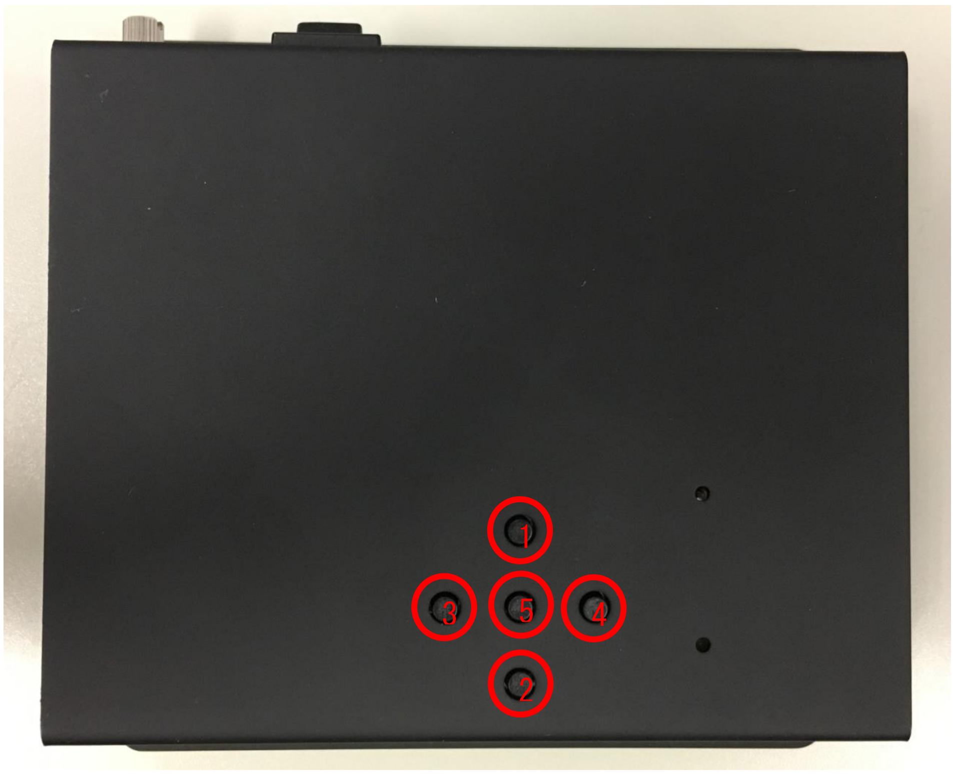


5V of I/O connector turns “ON” when the OS starts.　5V can be turned ON/OFF by the value state of GPIO16.

(Example：Turn ON/OFF 5V of I/O connector by using gpio command of Wiring Pi.)

```
gpio -g mode 16 out
gpio -g write 16 1    (I/O connector 5V ON)
gpio -g write 16 0    (I/O connector 5V OFF)
```

## 14. GPIO Assignment of General-purpose buttons



| | |
|---|---|
| ① UP button : | GPIO 17 |
| ② DOWN button : | GPIO 27 |
| ③ LEFT button : | GPIO 24 |
| ④ RIGHT button : | GPIO 22 |
| ⑤ ENTER button : | GPIO 25 |

◆PULLUP mode：enable Raspberry Pi biilt-in internal pull-up resister.

◆GPIO pin and GD connected to General-purpose buttons.
(External pull-up resister is not available.)

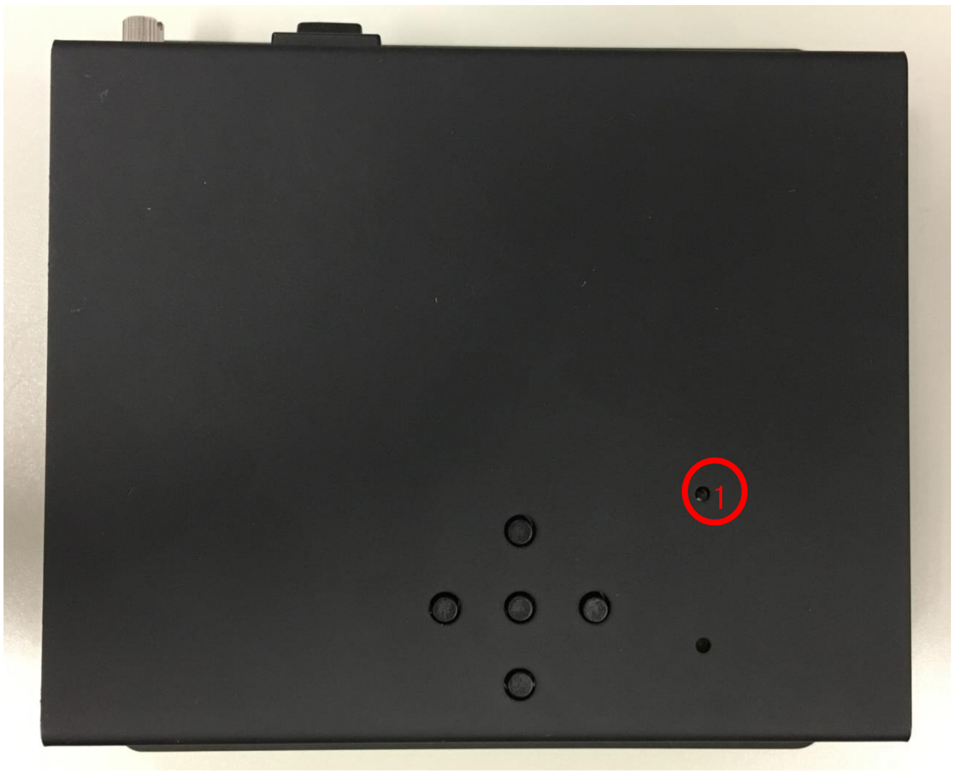
◆Show the value of H level(1) when not pressed, and L level(0) when pressed.

◆Example of getting the value state from the command line.

◆Obtaining the value state of the up button by using gpio command of Wiring Pi.)

```
gpio -g mode 17 input
gpio -g mode 17 up
gpio -g read 17
```

## 15. GPIO Assignment of General-purpose LED



① General-purpose LED: GPIO 12

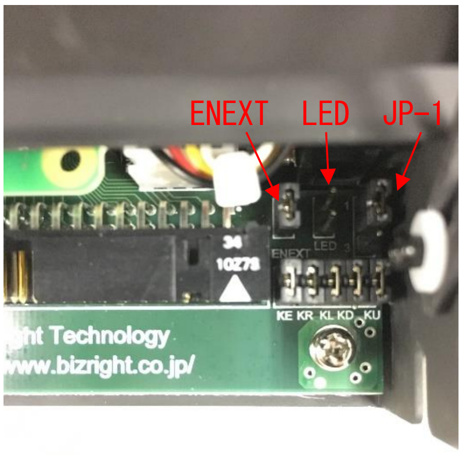
◆When using a General-purpose LED, open the slide lid and confirm the LED jumper pin is loaded.
(See [Disable General-purpose switch and General-purpose LED, reset control terminal ] for details.)

◆Set the L level (0) to turn on the General-purpose LED, and set the H level (1) to turn off.

Example (Turn on/off of general-purpose LED by using gpio command of Wiring Pi.)

```
gpio -g mode 12 out
gpio -g write 12 0     (General-purpose LED turn on)
gpio -g write 12 1     (General-purpose LED turn off)
```

## 16. Invalidation of General-purpose switch and General-purpose LED, Reset control terminal switch



(In this product, 5V ON/OFF switching of general-purpose switch, general-purpose LED and I/O connecter is disabled by jumper pin, and can use assigned GPIO port from the I/O connector.)

Reset control terminal can change the number of the GPIO port.

Enable to access the jumper pin by opening the slide lid.

| | |
|---|---|
| JP-1 | At the time of shipment, reset control and OS start up LED control are set to GPIO 18. (pins 1 and 2 are connected)<br>It can change into GPIO5 by switching jumper pin; pins 2 and 3 to be connected. |
| LED | Enable the general-purpose LED by inserting jumper pin.<br>If you remove the jumper pin, the General-purpose LED will be disabled and GPIO12 will be available. |
| ENEXT | If you remove the jumper pin, the 5V ON/OFF switching of I/O connecter will be disabled and GPIO16 will be available.<br>When the time the jumper pin is removed, the 5V pin(2,4) of I/O connecter will always be ON. |
| KU | If you remove the jumper pin, the UP button will be disabled and GPIO17 will be available. |

| | |
|---|---|
| KD | If you remove the jumper pin, the DOWN button will be disabled and GPIO17 will be available. |
| KL | If you remove the jumper pin, the LEFT button will be disabled and GPIO24 will be available. |
| KR | If you remove the jumper pin, the RIGHT button will be disabled and GPIO22 will be available. |
| KE | If you remove the jumper pin, the ENTER button will be disabled and GPIO25 will be available. |

If you change the setting to JP-1 GPIO5, create the configuration file "shutdownSensor.conf" under "/usr/local/sbin/ as following content, and save it. After save the configuration file, the OS reboot is needed.

/usr/local/sbin/shutdownSensor.conf

```
# Shutdown sensor config file

enrun pin = 5
```

## 17. About OS

Install the setup tools to your standard Raspbian.

With setup tools, the following functions will be appended.

◆High-speed shutdown process in a power down situation.)
Set RTC time to the System time at OS startup

◆Obtain the current time by using ntpdate, and set RTC time
Setting frequency: 30 min./hr when connecting to a network

◆Enable the Watch Dog Timer

Download the set-up tools shown below
http://dl.bizright.jp/bh/bh-tools-latest.tar.gz
(Supported OS: Raspbian Jessie or later)

Run the following command after install Raspbian 3.18 or later.

Install command of set-up tools

```
curl -O http://dl.bizright.jp/bh/bh-tools-latest.tar.gz
tar xfvz bh-tools-latest.tar.gz
cd bh-tools
./build
```

(*Installation of BH tools completed" is displayed when process is completed properly.)

```
"sudo reboot
```

*Caution : Make sure to be connected to the Internet to execute git and apt-get in the build file.

If you already installed the set-up tools, and update your OS version, run this command as well.

List of libraries and packages installed simultaneously is listed below.

- ◆Wiring Pi
- ◆ntpdate
- ◆watchdog
- ◆Install upstart if systemd is not installed
- ◆Install git if git is not installed

If you need uninstall the set-up tools, execute the following command.

Uninstall commands of set-up tools

cd bh-tools (*Move to the directory used during installation.)
./build uninstall
(*Success if "Uninstall of BH tools completed.")
cd wiringPi
./build uninstall (*When not using "Wiring Pi" in the future.)
sudo apt-get -y remove ntpdate (*When not using "ntpdate" in the future.)
sudo apt-get -y remove watchdog (When not using "watchdog" in the future.)
sudo reboot -f

*When if cannot install due to set-up error

If normal installation cannot be finished due to the Error:WDT module is unknown, BH may not have reboot after doing "apt-get update
After executing "sudo reboot", reinstall the set-up tools.

## 18. Processing sequence of High-speed shutdown

Processing sequence of High-speed shutdown on BH3.

1.Change the 23rd(INPUT mode) of GPIO from H level(1)　into L level(0).

2.Resident program” /usr/local/sbin/shutdownSensor” excecutes the following script in response to step1 interrupt signal.

- /usr/local/sbin/fastshutdown
- /usr/local/sbin/__fastdown-function(called from fastshutdown)

3.Output the start time to the log to measure the processing time of High-speed shutdown.

　　Output destination : /var/log/fastshutdown.log

　　　　Start time: (content of /proc/uptime)

4.Turn OFF the signal of the display

5.Execute the script "/usr/local/sbin/system-stop-use-usb"
(Timeout after 10 sec. and forcibly shutdown.)

6.Power OFF of the USB controller. Disable the devices connected to wired LAN /USB　to reduce power consumption during the shutdown process.

7.Execute the script "/usr/local/sbin/system-stop".
(Timeout after 10 sec. and forcibly shutdown.)

8.Save the current time to enable operation even when the RTC is not connected.

9.Terminate all the processes normally

10.Forcibly make the on memory-data flush out into the disk.

11.Forcibly terminate all the processes.

12.Output end time to the log to measure the processing time of High-speed shutdown.

Output Destination : /var/log/fastshutdown.log

End time: (contents of /proc/uptime)

13.(Forcibly make the on memory-data flush out into the disk.)

14.(Remount all mounted file system as the read-only mode.)

15.(When if GPIO 23rd(INPUT mode) is H level(1), system will be rebooted and shutdown will be executed the system if its L level(0).)

(Assuming the user turns off the AC power accidentally, and a case that the AC power restored after the shutdown process stated running.)

If you want to execute any command "programs created independently" during the high-speed shutdown process, add any command to the following script.
/usr/local/sbin/system-stop

If you want to perform the process of net work communication or using USB (as using wired LAN/wireless LAN, and saving file to the memory) in any commands during the high-speed shutdown process, add any commands to the following script without writing 「/usr/local/sbin/system-stop」.
/usr/local/sbin/system-stop-use-usb

Please complete the process of "system-stop" script and "system-stop-use-usb" script within 10 sec.　After 10 sec. passed, it will be timeout and force-quit an application.　The timeout sec. will be changeable with following script.

/usr/local/sbin/__fastdown-function

```
timeout -s 9 number of sec. for timeout /usr/local/sbin/system-stop-use-usb
timeout -s 9 number of sec. for timeout /usr/local/sbin/system-stop
```

When perform the closing process within original programs at high-speed shut down, close the file in response to the interruption of the kill signal.

Make the original programs at high-speed shut down receive the kill signal by adding as killall commands to “system-stop” script or “system-stop-use-usb” script.

Additional example:
/usr/local/sbin/system-stop

```
killall (original program name)
```

Keep the GPIO #18(OUT mode) H level(1) of the resident program at the time of startup(OS startup),　and remain until the shutdown complete to prevent unexpected reset occurred during OS boot.

## 19. RTC setting sequence

At OS boot (/usr/local/sbin/set-rtc):

1.Initializes the I2C connection of RTC
2.Set the RTC time to the system time
3.If the result of step 2 is failed, set the last shutdown time to the system time using “fake-hwclock”

Execute at the network connection and each 30 min./hr(every hour).

1.Check duplicate launches : if running already, stop the process
2.Wait for 20 sec.; “ntpdate” tends to be filed right after connected the network.
3.Execute “/usr/sbin/ntpdate-debian” command to set the system time via network.
4.If the step 3 succeeded, set the system time to RTC time.
5.If the step 3 succeeded, wait for 10 min. to prevent “ntpdate” from running continuously.

The interval of the process for ntpdate is changeable with following file.
/etc/cron.d/ntpdate　each 30 min./hr(every hour).

```
30 * * * * root /etc/network/if-up.d/ntpdate > /dev/null 2>&1
```

If assign NTP server, revise the following file.

/etc/default/ntpdate

```
NTPSERVERS="ntp.nict.jp ntp.jst.mfeed.ad.jp ntp.ring.gr.jp"
```

If manually execute nptdate and set he system time to RTC, execute the following commands.

```
sudo /usr/sbin/ntpdate-debian
sudo hwclock -w
```

Delete the following files if not connected to the network or no need to set up the system time by using ntpdate.

```
/etc/cron.d/ntpdate
/etc/network/if-up.d/ntpdate
```

Execute the following commands to set any time to the RTC.

```
sudo date -s 'time (example : 2015-06-19 00:00:00)'
sudo hwclock -w
```

Execute the following command to show the RTC time.

```
sudo hwclock -r
```

Execute the following command to set the RTC time to the system time.

```
sudo hwclock -s
```

## 20. Set up of Watch Dog Timer

Enables Watch Dog Timer by executing the following commands at the time of using set up tools.

Change the setting file by installing watchdog program, then automatically download bcm2708_wdog module(if earlier Raspbian 4.2) or bcm2835_wdt module(If later Raspbian 4.3) at the time of OS startup.

```
sudo apt-get install watchdog
sudo vi /etc/default/watchdog
        Change before:
                watchdog_module="none"
        After change (Raspbian version earlier 4.2)
                watchdog_module="bcm2708_wdog"
        After change (Raspbian version later 4.3)
                watchdog_module=" bcm2835_wdt "
sudo vi /etc/watchdog.conf
        Change before:
                #max-load-1= 24
                #watchdog-device = /dev/watchdog
        After change:
                max-load-1= 24
                watchdog-device = /dev/watchdog
                watchdog-timeout = 10
sudo vi /etc/modules
        Add to last line (Raspbian version earlier 4.2):
                bcm2708_wdog
        Add to last line (Raspbian version later 4.3):
                bcm2835_wdt
sudo vi /lib/systemd/system/watchdog.service
        Add to last line (Add under [Install] in the file)
                WantedBy=multi-user.target
sudo update-rc.d watchdog enable
sudo reboot
```

## 21. Sample source of programs using General-purpose switch

Sample source of programs showing each name of the button when you push the General-purpose switch.

This sample source can be downloaded from below.
http://dl.bizright.jp/bh/bhButton.zip

When if use "Wiring Pi" by C language
bhButton.c

```
#include <stdio.h>
#include <sys/time.h>
#include <wiringPi.h>

#define GPIO_UP 17
#define GPIO_DOWN 27
#define GPIO_RIGHT 22
#define GPIO_LEFT 24
#define GPIO_ENTER 25
#define BUTTON_USLEEP 50000
#define RESET_SEC 5

int setupGpio();
void isrUp();
void isrDown();
void isrRight();
void isrLeft();
void isrEnter();
void isrButton(int pin, char *button);

// Button name currently pressed
static volatile char *currentButton = NULL;

// Time pressed
static volatile long currentButtonTime = 0;
```

```
// Main function
int main(int argc, char *argv[]) {
        struct timeval now;

        // Initialization of GPIO
        if (setupGpio() == -1) return 1;

        while (1) {
                sleep(1);

                // When the general-purpose button is pressed currently.

                // Disable other general-purpose buttons for up to RESET_SEC
                  (5 sec.)

                if (currentButton != NULL) {
                        gettimeofday(&now, NULL);
                        if (RESET_SEC < now.tv_sec - currentButtonTime) {
                                currentButton = NULL;
                        }
                }
        }
        return 0;
}

// Initialization function of GPIO
int setupGpio() {
        // Initialization of WiringPi
        if (wiringPiSetupGpio() == -1) return -1;

        // Set to input mode
        pinMode(GPIO_UP, INPUT);
        pinMode(GPIO_DOWN, INPUT);
        pinMode(GPIO_RIGHT, INPUT);
        pinMode(GPIO_LEFT, INPUT);
        pinMode(GPIO_ENTER, INPUT);

        // Set to PULLUP mode
        pullUpDnControl(GPIO_UP, PUD_UP);
        pullUpDnControl(GPIO_DOWN, PUD_UP);
        pullUpDnControl(GPIO_RIGHT, PUD_UP);
        pullUpDnControl(GPIO_LEFT, PUD_UP);
        pullUpDnControl(GPIO_ENTER, PUD_UP);

        // Detect the status change of GPIO and perform the interrupt process.
        wiringPiISR(GPIO_UP, INT_EDGE_BOTH, &isrUp);
        wiringPiISR(GPIO_DOWN, INT_EDGE_BOTH, &isrDown);
        wiringPiISR(GPIO_RIGHT, INT_EDGE_BOTH, &isrRight);
        wiringPiISR(GPIO_LEFT, INT_EDGE_BOTH, &isrLeft);
        wiringPiISR(GPIO_ENTER, INT_EDGE_BOTH, &isrEnter);
        return 0;
}

// Interrupt function of up button
void isrUp() {
        isrButton(GPIO_UP, "up");
}

// Interrupt function of down button
void isrDown() {
        isrButton(GPIO_DOWN, "down");
}

// Interrupt function of right button
void isrRight() {
        isrButton(GPIO_RIGHT, "right");
}

// Interrupt function of left button
void isrLeft() {
        isrButton(GPIO_LEFT, "left");
```

```
}

// Interrupt function of enter button
void isrEnter() {
	isrButton(GPIO_ENTER, "enter");
}

// Interrupt function of button
void isrButton(int pin, char *button) {
	int read1, read2;
	struct timeval now;

Prevent malfunction from chattering, obtain the value of GPIO 2 times at the
interval of BUTTON_USLEEP(0.05 sec), then show the button names only if
they mach.
	read1 = digitalRead(pin);
	usleep(BUTTON_USLEEP);
	read2 = digitalRead(pin);
	if (read1 == 0 && read2 == 0) {
		if (currentButton == NULL) {
			gettimeofday(&now, NULL);
			currentButtonTime = now.tv_sec;
			currentButton = button;
			printf("%s¥n", button);
		}
	} else if (currentButton == button && read1 == 1 && read2 == 1) {
		currentButton = NULL;
	}
}
```

Build command

```
gcc -lwiringPi -o bhButton bhButton.c
```

Execution command

```
sudo ./bhButton
```

When using Rpi.GPIO by Python
bhButton.py

```
# coding: UTF-8

import RPi.GPIO as GPIO
import time

GPIO_UP = 17
GPIO_DOWN = 27
GPIO_RIGHT = 22
GPIO_LEFT = 24
GPIO_ENTER = 25
BUTTON_SLEEP = 0.05

# Interrupt function of button
def gpio_callback(channel):

Prevent malfunction from chattering, obtain the value of GPIO 2 times at the
interval of BUTTON_USLEEP(0.05 sec), then show the button names only if
they mach.

        value1 = GPIO.input(channel)
        if value1 == 1:
                return
        time.sleep(BUTTON_SLEEP)
        value2 = GPIO.input(channel)
        if value2 == 1:
                return
        if value1 != value2:
                return

        if channel == GPIO_UP:
                print('up')
        elif channel == GPIO_DOWN:
                print('down')
        elif channel == GPIO_RIGHT:
                print('right')
        elif channel == GPIO_LEFT:
                print('left')
        elif channel == GPIO_ENTER:
                print('enter')

# Set INPUT mode, PULLUP mode
GPIO.setmode(GPIO.BCM)
GPIO.setup(GPIO_UP, GPIO.IN, pull_up_down=GPIO.PUD_UP)
GPIO.setup(GPIO_DOWN, GPIO.IN, pull_up_down=GPIO.PUD_UP)
GPIO.setup(GPIO_RIGHT, GPIO.IN, pull_up_down=GPIO.PUD_UP)
GPIO.setup(GPIO_LEFT, GPIO.IN, pull_up_down=GPIO.PUD_UP)
GPIO.setup(GPIO_ENTER, GPIO.IN, pull_up_down=GPIO.PUD_UP)

# Detect from the change of GPIO value (from 1 to 0), perform interrupt
processing.
GPIO.add_event_detect(GPIO_UP, GPIO.FALLING)
GPIO.add_event_detect(GPIO_DOWN, GPIO.FALLING)
GPIO.add_event_detect(GPIO_RIGHT, GPIO.FALLING)
GPIO.add_event_detect(GPIO_LEFT, GPIO.FALLING)
GPIO.add_event_detect(GPIO_ENTER, GPIO.FALLING)
GPIO.add_event_callback(GPIO_UP, gpio_callback)
GPIO.add_event_callback(GPIO_DOWN, gpio_callback)
GPIO.add_event_callback(GPIO_RIGHT, gpio_callback)
GPIO.add_event_callback(GPIO_LEFT, gpio_callback)
GPIO.add_event_callback(GPIO_ENTER, gpio_callback)

while 1:
        time.sleep(1)
```

Execution command

```
sudo python bhButton.py
```

## 22. Sample of source of program using General-purpose LED

Sample source of grogram blinking general-purpose LED.

This sample source can be downloaded from below
http://dl.bizright.jp/bh/bhLED.zip

When if use “Wiring Pi” by C language
bhLED.c

```
#include <stdio.h>
#include <wiringPi.h>

#define GPIO_LED 12

int setupGpio();

// Main function
int main(int argc, char *argv[]) {

        // Initialization of GPIO
        if (setupGpio() == -1) return 1;

        while (1) {

                // Lights general-purpose LED
                digitalWrite(GPIO_LED, 0);

                // 0.5 sec. sleep
                delay(500);

                // Turn-off general-purpose LED
                digitalWrite(GPIO_LED, 1);

                // 0.5 sec. sleep
                delay(500);
```

```
	}
	return 0;
}

// Initial function of GPIO
int setupGpio() {
	// Initialization of WiringPi
	if (wiringPiSetupGpio() == -1) return -1;

	// Set OUTPUT mode
	pinMode(GPIO_LED, OUTPUT);

	return 0;
}
```

Build command

```
gcc -lwiringPi -o bhLED bhLED.c
```

Execute command

```
sudo ./bhLED
```

When using RPi.GPIO by Python
bhLED.py

```
# coding: UTF-8

import RPi.GPIO as GPIO
import time

GPIO_LED = 12

# Set OUTPUT mode
GPIO.setmode(GPIO.BCM)
GPIO.setup(GPIO_LED, GPIO.OUT)

try:
        while 1:

                # Lights general-purpose LED
                GPIO.output(GPIO_LED, 0)

                # 0.5 sec. sleep
                time.sleep(0.5)

                # Lights general-purpose LED
                GPIO.output(GPIO_LED, 1)

                # 0.5 sec. sleep
                time.sleep(0.5)

finally:
        # Reset GPIO
        GPIO.cleanup()
```

Execute command

```
sudo python bhLED.py
```

## 23. Revision Histories

| Date of update | Version | Contents |
|---|---|---|
| 2020/4/27 | 1.6 | 1st edition |
| | | |
| | | |
| | | |
| | | |
| | | |
| | | |