

# Table of Contents

<b>BS2_SetOsdpStandardActionConfig</b> .....	1
Declaration .....	1
Parameter .....	1
Return Value .....	1
See Also .....	1
Sample Code(C++) .....	1
Sample Code (C#) .....	4

[Configuration API](#) > [BS2\\_SetOsdpStandardActionConfig](#)

---

## BS2\_SetOsdpStandardActionConfig

[+ 2.9.1] Store the LED/buzzer setting for each action of the OSDP device.

### Declaration

```
#include "BS_API.h"

int BS2_SetOsdpStandardActionConfig(void* context, uint32_t deviceId, const
BS2OsdpStandardActionConfig* config);
```

[See BS2OsdpStandardActionConfig Structure](#)

### Parameter

- [In] *context* : Context
- [In] *deviceId* : Device Identifier
- [Out] *config* : Pointer to get LED/buzzer information of OSDP device

### Return Value

If successfully done, BS\_SDK\_SUCCESS will be returned. If there is an error, the corresponding error code will be returned.

### See Also

[BS2\\_GetOsdpStandardActionConfig](#)

### Sample Code(C++)

[sample\\_setosdpstandardactionconfig.cpp](#)

```
BS2OsdpStandardActionConfig config = { , };

BS2_DEVICE_ID id = Utility::selectDeviceID(deviceList, false, false);

bool activate = false;
```

```

for (int idx = ; idx < BS2_OSDP_STANDARD_ACTION_TYPE_COUNT; idx++)
{
    string msg;
    switch (config.actions[idx].actionType)
    {
        case BS2_OSDP_STANDARD_ACTION_TYPE_SUCCESS:
            msg = "SUCCESS";
            break;
        case BS2_OSDP_STANDARD_ACTION_TYPE_FAIL:
            msg = "FAIL";
            break;
        case BS2_OSDP_STANDARD_ACTION_TYPE_WAIT_INPUT:
            msg = "WAIT_INPUT";
            break;
        case BS2_OSDP_STANDARD_ACTION_TYPE_NONE:
        default:
            msg = "NONE";
            break;
    }
    if (Utility::isYes("Do you want to modify the %s action type?",
msg.c_str()))
    {
        for (int ledidx = ; ledidx < BS2_OSDP_STANDARD_ACTION_MAX_LED;
ledidx++)
        {
            string msg;
            activate = Utility::isYes("Do you want to activate for
LED#%d action?", ledidx);
            config.actions[idx].led[ledidx].use = activate;

            if (activate)
            {
                msg = " Please enter your reader number. Default(0).";
                config.actions[idx].led[ledidx].readerNumber =
(uint8_t)Utility::getInput<uint32_t>(msg);
                msg = " Please enter a led number of the reader.
Default(0).";
                config.actions[idx].led[ledidx].ledNumber =
(uint8_t)Utility::getInput<uint32_t>(msg);

                msg = " Please enter a temporary command (0: NOP, 1:
Cancel, 2: Set)";
                config.actions[idx].led[ledidx].tempCommand =
(uint8_t)Utility::getInput<uint32_t>(msg);
                msg = " Please enter the temporary command on time in
ms.";
                config.actions[idx].led[ledidx].tempOnTime =
(uint8_t)Utility::getInput<uint32_t>(msg);
                msg = " Please enter the temporary command off time in
ms.";
                config.actions[idx].led[ledidx].tempOffTime =

```

```

(uint8_t)Utility::getInput<uint32_t>(msg);
    msg = " Please enter the temporary command on color.
(0: Black, 1: Red, 2: Green, 3: Amber, 4: Blue, 5: Magenta, 6: Cyan, 7:
White)";
    config.actions[idx].led[ledidx].tempOnColor =
(uint8_t)Utility::getInput<uint32_t>(msg);
    msg = " Please enter the temporary command off color.
(0: Black, 1: Red, 2: Green, 3: Amber, 4: Blue, 5: Magenta, 6: Cyan, 7:
White)";
    config.actions[idx].led[ledidx].tempOffColor =
(uint8_t)Utility::getInput<uint32_t>(msg);
    msg = " Please enter the temporary run time in ms.";
    config.actions[idx].led[ledidx].tempRunTime =
(uint16_t)Utility::getInput<uint32_t>(msg);

    msg = " Please enter a permanent command (0: NOP, 1:
Cancel, 2: Set)";
    config.actions[idx].led[ledidx].permCommand =
(uint8_t)Utility::getInput<uint32_t>(msg);
    msg = " Please enter the permanent on time in ms.";
    config.actions[idx].led[ledidx].permOnTime =
(uint8_t)Utility::getInput<uint32_t>(msg);
    msg = " Please enter the permanent off time in ms.";
    config.actions[idx].led[ledidx].permOffTime =
(uint8_t)Utility::getInput<uint32_t>(msg);
    msg = " Please enter the permanent on color. (0:
Black, 1: Red, 2: Green, 3: Amber, 4: Blue, 5: Magenta, 6: Cyan, 7:
White)";
    config.actions[idx].led[ledidx].permOnColor =
(uint8_t)Utility::getInput<uint32_t>(msg);
    msg = " Please enter the permanent off color. (0:
Black, 1: Red, 2: Green, 3: Amber, 4: Blue, 5: Magenta, 6: Cyan, 7:
White)";
    config.actions[idx].led[ledidx].permOffColor =
(uint8_t)Utility::getInput<uint32_t>(msg);
    }
}

activate = Utility::isYes("Do you want to activate for buzzer
action?");
config.actions[idx].buzzer.use = activate;
if (activate)
{
    msg = " Please enter your reader number. Default(0).";
    config.actions[idx].buzzer.readerNumber =
(uint8_t)Utility::getInput<uint32_t>(msg);
    msg = " Please enter a tone type (0: None, 1: Off, 2:
On)";
    config.actions[idx].buzzer.tone =
(uint8_t)Utility::getInput<uint32_t>(msg);
    msg = " Please enter the buzzer turn-on time in ms.";

```

```

        config.actions[idx].buzzer.onTime =
(uint8_t)Utility::getInput<uint32_t>(msg);
        msg = " Please enter the buzzer turn-off time in ms.";
        config.actions[idx].buzzer.offTime =
(uint8_t)Utility::getInput<uint32_t>(msg);
        msg = " Please enter the number of cycle the buzzer on and
off.";
        config.actions[idx].buzzer.numOfCycle =
(uint8_t)Utility::getInput<uint32_t>(msg);
    }
}
}

int sdkResult = BS2_SetOsdpStandardActionConfig(context_, id, &config);
if (BS_SDK_SUCCESS != sdkResult)
    printf("BS2_SetOsdpStandardActionConfig call failed: %d",
sdkResult);

return sdkResult;

```

## Sample Code (C#)

[sample\\_setosdpstandardactionconfig.cs](#)

```

BS20sdpStandardActionConfig config;
bool activate = false;
for (int idx = ; idx < (int)BS20sdpStandardActionType.COUNT; idx++)
{
    Console.WriteLine("Do you want to modify the {0} action type?
[Y/n]", (BS20sdpStandardActionType)config.actions[idx].actionType);
    Console.Write(">> ");
    if (Util.IsYes())
    {
        for (int ledidx = ; ledidx <
BS2Environment.BS2_OSDP_STANDARD_ACTION_MAX_LED; ledidx++)
        {
            string tempStr = String.Format("LED#{0}", ledidx);
            string msgStr = String.Format("Do you want to activate for
{0} action? [Y/n]", tempStr);
            Util.HighlightLine(msgStr, tempStr);
            Console.Write(">> ");
            activate = Util.IsYes();
            config.actions[idx].led[ledidx].use =
Convert.ToByte(activate);

            if (activate)
            {
                Util.HighlightLine(" Please enter your reader number.

```

```
Default(0).", "reader number");
    Console.WriteLine(" >>>> ");
    config.actions[idx].led[ledidx].readerNumber =
Util.GetInput((byte));
    Util.HighlightLine(" Please enter a led number of the
reader. Default(0).", "led number");
    Console.WriteLine(" >>>> ");
    config.actions[idx].led[ledidx].ledNumber =
Util.GetInput((byte));

    Util.HighlightLine(" Please enter a temporary command
(0: NOP, 1: Cancel, 2: Set)", "temporary command");
    Console.WriteLine(" >>>> ");
    config.actions[idx].led[ledidx].tempCommand =
Util.GetInput((byte)BS20sdpStandardLEDCommand.NOP);
    Util.HighlightLine(" Please enter the temporary
command on time in ms.", "on time");
    Console.WriteLine(" >>>> ");
    config.actions[idx].led[ledidx].tempOnTime =
Util.GetInput((byte)10);
    Util.HighlightLine(" Please enter the temporary
command off time in ms.", "off time");
    Console.WriteLine(" >>>> ");
    config.actions[idx].led[ledidx].tempOffTime =
Util.GetInput((byte));
    Util.HighlightLine(" Please enter the temporary
command on color. (0: Black, 1: Red, 2: Green, 3: Amber, 4: Blue, 5:
Magenta, 6: Cyan, 7: White)", "on color");
    Console.WriteLine(" >>>> ");
    config.actions[idx].led[ledidx].tempOnColor =
Util.GetInput((byte)BS20sdpStandardColor.GREEN);
    Util.HighlightLine(" Please enter the temporary
command off color. (0: Black, 1: Red, 2: Green, 3: Amber, 4: Blue, 5:
Magenta, 6: Cyan, 7: White)", "off color");
    Console.WriteLine(" >>>> ");
    config.actions[idx].led[ledidx].tempOffColor =
Util.GetInput((byte)BS20sdpStandardColor.BLACK);
    Util.HighlightLine(" Please enter the temporary run
time in ms.", "run time");
    Console.WriteLine(" >>>> ");
    config.actions[idx].led[ledidx].tempRunTime =
Util.GetInput((UInt16)10);

    Util.HighlightLine(" Please enter a permanent command
(0: NOP, 1: Cancel, 2: Set)", "permanent command");
    Console.WriteLine(" >>>> ");
    config.actions[idx].led[ledidx].permCommand =
Util.GetInput((byte)BS20sdpStandardLEDCommand.NOP);
    Util.HighlightLine(" Please enter the permanent on
time in ms.", "on time");
    Console.WriteLine(" >>>> ");
```

```

        config.actions[idx].led[ledidx].permOnTime =
Util.GetInput((byte));
        Util.HighlightLine(" Please enter the permanent off
time in ms.", "off time");
        Console.Write(" >>>> ");
        config.actions[idx].led[ledidx].permOffTime =
Util.GetInput((byte));
        Util.HighlightLine(" Please enter the permanent on
color. (0: Black, 1: Red, 2: Green, 3: Amber, 4: Blue, 5: Magenta, 6:
Cyan, 7: White)", "on color");
        Console.Write(" >>>> ");
        config.actions[idx].led[ledidx].permOnColor =
Util.GetInput((byte)BS20sdpStandardColor.BLACK);
        Util.HighlightLine(" Please enter the permanent off
color. (0: Black, 1: Red, 2: Green, 3: Amber, 4: Blue, 5: Magenta, 6:
Cyan, 7: White)", "off color");
        Console.Write(" >>>> ");
        config.actions[idx].led[ledidx].permOffColor =
Util.GetInput((byte)BS20sdpStandardColor.BLACK);
    }
}

    Util.HighlightLine("Do you want to activate for buzzer action?
[Y/n]", "buzzer");
    Console.Write(">> ");
    activate = Util.IsYes();
    config.actions[idx].buzzer.use = Convert.ToByte(activate);
    if (activate)
    {
        Util.HighlightLine(" Please enter your reader number.
Default(0).", "reader number");
        Console.Write(" >>>> ");
        config.actions[idx].buzzer.readerNumber =
Util.GetInput((byte));

        Util.HighlightLine(" Please enter a tone type (0: None, 1:
Off, 2: On)", "tone");
        Console.Write(" >>>> ");
        config.actions[idx].buzzer.tone =
Util.GetInput((byte)BS20sdpStandardTone.ON);
        Util.HighlightLine(" Please enter the buzzer turn-on time
in ms.", "on time");
        Console.Write(" >>>> ");
        config.actions[idx].buzzer.onTime = Util.GetInput((byte)2);
        Util.HighlightLine(" Please enter the buzzer turn-off time
in ms.", "off time");
        Console.Write(" >>>> ");
        config.actions[idx].buzzer.offTime = Util.GetInput((byte));
        Util.HighlightLine(" Please enter the number of cycle the
buzzer on and off.", "number of cycle");
        Console.Write(" >>>> ");
    }
}

```

```
        config.actions[idx].buzzer.numOfCycle =
Util.GetInput((byte)1);
    }
}

Console.WriteLine("Trying to set OsdpStandardActionConfig");
BS2ErrorCode result =
(BS2ErrorCode)API.BS2_SetOsdpStandardActionConfig(sdkContext, deviceID,
ref config);
if (result != BS2ErrorCode.BS_SDK_SUCCESS)
    Console.WriteLine("Got error({0}).", result);
else
    Console.WriteLine("Call success.");

return result;
```

From:

<https://kb.supremainc.com/kbtest/> - **BioStar 2 Device SDK**

Permanent link:

[https://kb.supremainc.com/kbtest/doku.php?id=en:bs2\\_setosdpstandardactionconfig](https://kb.supremainc.com/kbtest/doku.php?id=en:bs2_setosdpstandardactionconfig)

Last update: **2023/02/28 15:01**