

- Configuration API** 1
- 3
- BS2FactoryConfig 3
- BS2SystemConfig 4
- BS2AuthConfig 6
- BS2StatusConfig 8
- BS2DisplayConfig 9
- BS2IpConfig 12
- BS2IpConfigExt 13
- BS2TNAConfig 14
- BS2CardConfig 15
- BS2FingerprintConfig 18
- BS2Rs485Config 20
- BS2WiegandConfig 22
- BS2WiegandDeviceConfig 24
- BS2InputConfig 26
- BS2WlanConfig 27
- BS2Trigger 28
- BS2Action 30
- BS2TriggerActionConfig 34
- BS2EventConfig 35
- BS2WiegandMultiConfig 35
- BS1CardConfig 36
- BS2SystemConfigExt 37
- BS2VoipConfig 38
- BS2FaceConfig 39
- BS2Rs485ConfigEX 42
- BS2CardConfigEx 44
- BS2DstConfig 45
- BS2Configs 46
- BS2IPV6Config 47
- BS2DesFireCardConfigEx 49
- BS2AuthConfigExt 50
- BS2FaceConfigExt 53
- BS2ThermalCameraConfig 56
- BS2BarcodeConfig 56
- BS2InputConfigEx 58
- BS2RelayActionConfig 59
- BS2VoipConfigExt 61
- BS2RtspConfig 63
- BS2License 64
- BS2LicenseConfig 65
- BS2BarcodeConfig 65
- BS2OsdpStandardConfig 67
- BS2OsdpStandardActionConfig 69
- BS2CustomMifareCard 72
- BS2CustomDesFireCard 73
- BS2CustomCardConfig 74

Configuration API

API

- [BS2_ResetConfig](#):
- [BS2_ResetConfigExceptNetInfo](#):
- [BS2_GetConfig](#): Configuration blob 가
- [BS2_SetConfig](#): Configuration blob
- [BS2_GetFactoryConfig](#): 가
- [BS2_GetSystemConfig](#): 가
- [BS2_SetSystemConfig](#):
- [BS2_GetAuthConfig](#): 가
- [BS2_SetAuthConfig](#):
- [BS2_GetStatusConfig](#): led, buzzer 가
- [BS2_SetStatusConfig](#): led, buzzer
- [BS2_GetDisplayConfig](#): UI 가
- [BS2_SetDisplayConfig](#): UI
- [BS2_GetIPConfig](#): IP 가
- [BS2_GetIPConfigViaUDP](#): IP UDP broadcasting 가
- [BS2_SetIPConfig](#): IP
- [BS2_SetIPConfigViaUDP](#): IP UDP broadcasting
- [BS2_GetIPConfigExt](#): DNS Server URL 가
- [BS2_SetIPConfigExt](#): DNS Server URL
- [BS2_GetTNAConfig](#): TNA 가
- [BS2_SetTNAConfig](#): TNA
- [BS2_GetCardConfig](#): 가
- [BS2_SetCardConfig](#):
- [BS2_GetFingerprintConfig](#): 가
- [BS2_SetFingerprintConfig](#):
- [BS2_GetRS485Config](#): RS485 가
- [BS2_SetRS485Config](#): RS485
- [BS2_GetWiegandConfig](#): Wiegand 가
- [BS2_SetWiegandConfig](#): Wiegand
- [BS2_GetWiegandDeviceConfig](#): Wiegand 가
- [BS2_SetWiegandDeviceConfig](#): Wiegand
- [BS2_GetInputConfig](#): Supervised 가
- [BS2_SetInputConfig](#): Supervised
- [BS2_GetWlanConfig](#): 가
- [BS2_SetWlanConfig](#):
- [BS2_GetTriggerActionConfig](#): Trigger action 가
- [BS2_SetTriggerActionConfig](#): Trigger action
- [BS2_GetEventConfig](#): Image log filter 가
- [BS2_SetEventConfig](#): Image log filter
- [BS2_GetWiegandMultiConfig](#): WiegandMulti 가
- [BS2_SetWiegandMultiConfig](#): WiegandMulti
- [BS2_GetCard1xConfig](#): V1.x Template On Card 가
- [BS2_SetCard1xConfig](#): V1.x Template On Card
- [BS2_GetSystemExtConfig](#): Master Slave 가

- [BS2_SetSystemExtConfig](#): Master Slave .
- [BS2_GetVoipConfig](#): Voip 가 .
- [BS2_SetVoipConfig](#): Voip .
- [BS2_GetFaceConfig](#): Face 가 .
- [BS2_SetFaceConfig](#): Face .
- [BS2_GetRS485ConfigEx](#): CoreStation RS485 가 .
- [BS2_SetRS485ConfigEx](#): CoreStation RS485 .
- [BS2_GetCardConfigEx](#): iClass SEOS 가 .
- [BS2_SetCardConfigEx](#): iClass SEOS .
- [BS2_GetDstConfig](#): DST 가 .
- [BS2_SetDstConfig](#): DST .
- [BS2_GetSupportedConfigMask](#): 가 .
- [BS2_GetIPConfigViaUDPEx](#): [+ 2.6.3] IP UDP broadcasting 가 . (host ip)
- [BS2_SetIPConfigViaUDPEx](#): [+ 2.6.3] IP UDP broadcasting . (host ip)
- [BS2_GetIPv6Config](#): [+ 2.6.3] IP V6 가 .
- [BS2_SetIPv6Config](#): [+ 2.6.3] IP V6 .
- [BS2_GetIPv6ConfigViaUDP](#): [+ 2.6.3] IP V6 UDP broadcasting 가 .
- [BS2_SetIPv6ConfigViaUDP](#): [+ 2.6.3] IP V6 UDP broadcasting .
- [BS2_GetIPv6ConfigViaUDPEx](#): [+ 2.6.3] IP V6 UDP broadcasting 가 . (host ip)
- [BS2_SetIPv6ConfigViaUDPEx](#): [+ 2.6.3] IP V6 UDP broadcasting . (host ip)
- [BS2_GetDesFireCardConfigEx](#): [+ 2.6.4] DesFire 가 .
- [BS2_SetDesFireCardConfigEx](#): [+ 2.6.4] DesFire .
- [BS2_GetAuthConfigExt](#): [+ 2.7.1] FSF2 가 .
- [BS2_SetAuthConfigExt](#): [+ 2.7.1] FSF2 .
- [BS2_GetFaceConfigExt](#): [+ 2.7.1] FSF2, FS2 가 .
- [BS2_SetFaceConfigExt](#): [+ 2.7.1] FSF2, FS2 .
- [BS2_GetThermalCameraConfig](#): [+ 2.7.1] FSF2, FS2 가 .
- [BS2_SetThermalCameraConfig](#): [+ 2.7.1] FSF2, FS2 .
- [BS2_GetBarcodeConfig](#): [+ 2.8] XS2 Barcode 가 .
- [BS2_SetBarcodeConfig](#): [+ 2.8] XS2 Barcode .
- [BS2_GetInputConfigEx](#): [+ 2.8.1] IM-120 Input 가 .
- [BS2_SetInputConfigEx](#): [+ 2.8.1] IM-120 Input .
- [BS2_GetRelayActionConfig](#): [+ 2.8.1] IM-120 RelayAction 가 .
- [BS2_SetRelayActionConfig](#): [+ 2.8.1] IM-120 RelayAction .
- [BS2_GetVoipConfigExt](#): [+ 2.8.3] BS3 VoIP 가 .
- [BS2_SetVoipConfigExt](#): [+ 2.8.3] BS3 VoIP .
- [BS2_GetRtspConfig](#): [+ 2.8.3] BS3 RTSP 가 .
- [BS2_SetRtspConfig](#): [+ 2.8.3] BS3 RTSP .
- [BS2_GetLicenseConfig](#): [+ 2.9.1] 가 .
- [BS2_GetOsdpStandardConfig](#): [+ 2.9.1] OSDP 가 .
- [BS2_GetOsdpStandardActionConfig](#): [+ 2.9.1] OSDP action LED/buzzer 가 .
- [BS2_SetOsdpStandardActionConfig](#): [+ 2.9.1] OSDP action LED/buzzer .
- [BS2_GetCustomCardConfig](#): [+ 2.9.4] Custom smart card 가 .
- [BS2_SetCustomCardConfig](#): [+ 2.9.4] Custom smart card .

BS2FactoryConfig

```
typedef struct {
    uint8_t major;
    uint8_t minor;
    uint8_t ext;
    uint8_t reserved[1];
} Version;

typedef struct {
    uint32_t deviceID;
    uint8_t macAddr[BS2_MAC_ADDR_LEN];
    uint8_t reserved[2];
    char modelName[BS2_MODEL_NAME_LEN];
    Version boardVer;
    Version kernelVer;
    Version bscoreVer;
    Version firmwareVer;
    char kernelRev[BS2_KERNEL_REV_LEN];
    char bscoreRev[BS2_BSCORE_REV_LEN];
    char firmwareRev[BS2_FIRMWARE_REV_LEN];
    uint8_t reserved2[32];
} BS2FactoryConfig;
```

1. *deviceID*
2. *macAddr*
3. *reserved*
4. *modelName*
5. *boardVer*
6. *kernelVer*
7. *bscoreVer*
BioStar core
8. *firmwareVer*
9. *kernelRev*

10. *bScoreRev*
BioStar core

11. *firmwareRev*

12. *reserved2*

BS2SystemConfig

```

typedef struct {
    uint8_t notUsed[16 * 16 * 3];
    int32_t timezone;
    uint8_t syncTime;
    uint8_t serverSync;
    uint8_t deviceLocked;
    uint8_t useInterphone;
    uint8_t useUSBConnection;
    uint8_t keyEncrypted;
    uint8_t useJobCode;
    uint8_t useAlphanumericID;
    uint32_t cameraFrequency;
    bool secureTamper;
    bool reserved0; // (write protected)
    uint8_t reserved[2];
    uint32_t useCardOperationMask;
    uint8_t reserved2[16];
} BS2SystemConfig;

```

1. *notUsed*

2. *timezone*
(sec)

3. *syncTime*
BioStar flag

4. *serverSync*

5. *deviceLocked*
()

6. *useInterphone*
flag

7. *useUSBConnection*
member (USB 가 .)

8. *keyEncrypted*

OSDP secure key flag .

9. *useJobCode*

Job code flag .

10. *useAlphanumericID*

AlphanumericID flag .

11. *cameraFrequency*

camera .

1	50Hz
2	60Hz

12. *secureTamper*

on flag . (, , , SSL)

13. *reserved0*

14. *reserved*

15. *useCardOperationMask*

[+ 2.6.4]

MASK 가 , 가

, 가 , 가 가

, EM CARD_OPERATION_USE useCardOperationMask 0x80000001

0xFFFFFFFF	CARD_OPERATION_MASK_DEFAULT
0x80000000	CARD_OPERATION_USE
0x00000000	CARD_OPERATION_MASK_NONE
0x00000001	CARD_OPERATION_MASK_LF_EM
0x00000002	CARD_OPERATION_MASK_LF_PROX
0x00000004	CARD_OPERATION_MASK_HF_CSN_MIFARE
0x00000008	CARD_OPERATION_MASK_HF_CSN_ICLASS
0x00000010	CARD_OPERATION_MASK_HF_SMART_MIFARE
0x00000020	CARD_OPERATION_MASK_HF_SMART_MIFARE_DESFIRE
0x00000040	CARD_OPERATION_MASK_HF_SMART_ICLASS
0x00000080	CARD_OPERATION_MASK_HF_SMART_ICLASS_SEOS
0x00000100	CARD_OPERATION_MASK_MOBILE_NFC
0x00000200	CARD_OPERATION_MASK_MOBILE_BLE
0x00000400	CARD_OPERATION_MASK_HF_CSN_OTHERS

16. reserved2

BS2AuthConfig

```
typedef struct {
    uint32_t authSchedule[BS2_NUM_OF_AUTH_MODE];
    uint8_t useGlobalAPB;
    uint8_t globalAPBFailAction;
    uint8_t useGroupMatching;
    uint8_t reserved
    uint8_t reserved[28];
    uint8_t usePrivateAuth;
    uint8_t faceDetectionLevel;
    uint8_t useServerMatching;
    uint8_t useFullAccess;
    uint8_t matchTimeout;
    uint8_t authTimeout;
    uint8_t numOperators;
    uint8_t reserved2[1];
    struct {
        char userID[BS2_USER_ID_SIZE];
        uint8_t level;
        uint8_t reserved[3];
    } operators[BS2_MAX_OPERATORS];
} BS2AuthConfig;
```

1. authSchedule

가
가 ,
0

0	BS2_AUTH_MODE_BIOMETRIC_ONLY	
1	BS2_AUTH_MODE_BIOMETRIC_PIN	+ PIN
2	BS2_AUTH_MODE_CARD_ONLY	
3	BS2_AUTH_MODE_CARD_BIOMETRIC	+
4	BS2_AUTH_MODE_CARD_PIN	+ PIN
5	BS2_AUTH_MODE_CARD_BIOMETRIC_OR_PIN	+ or PIN

6	BS2_AUTH_MODE_CARD_BIOMETRIC_PIN	+ + PIN
7	BS2_AUTH_MODE_ID_BIOMETRIC	ID +
8	BS2_AUTH_MODE_ID_PIN	ID + PIN
9	BS2_AUTH_MODE_ID_BIOMETRIC_OR_PIN	ID + or PIN
10	BS2_AUTH_MODE_ID_BIOMETRIC_PIN	ID + + PIN

2. useGlobalAPB

flag

3. globalAPBFailAction

BioStar

0	APB
1	Soft APB
2	Hard APB

4. useGroupMatching

flag

5. reserved

6. usePrivateAuth

flag

7. faceDetectionLevel

A2

Normal/Strict

가

가

0

0	
1	Normal mode
2	Strict mode

A2 가 , FaceStation2 FaceLite

8. *useServerMatching*

Matching server flag .

9. *useFullAccess*

.

10. *matchTimeout*

(sec) .

11. *authTimeout*

(sec) .

12. *numOperators*

operator .

13. *reserved2*

.

14. *userID*

.

15. *level*

가

.

0	
1	
2	
3	

Operator 가 , 가 operator **numOperators**

16. *reserved*

.

BS2StatusConfig

```
typedef struct {
    struct {
        uint8_t enabled;
        uint8_t reserved[1];
        uint16_t count;
        BS2LedSignal signal[BS2_LED_SIGNAL_NUM];
    } led[BS2_DEVICE_STATUS_NUM];
    uint8_t reserved1[32];
    struct {
        uint8_t enabled;
    }
}
```

```

    uint8_t reserved[1];
    uint16_t count;
    BS2BuzzerSignal signal[BS2_BUZZER_SIGNAL_NUM];
} buzzer[BS2_DEVICE_STATUS_NUM];
uint8_t configSyncRequired;
uint8_t reserved2[31];
} BS2StatusConfig;

```

1. *enabled*

led flag .

2. *reserved*

.

3. *count*

led signal , 0 .

4. *signal*

led signal pattern , 3 .

5. *reserved1*

.

6. *enabled*

buzzer flag .

7. *reserved*

.

8. *count*

buzzer signal , 0 .

9. *signal*

buzzer signal pattern , 3 . 10. *configSyncRequired*
configuration , true .

11. *reserved2*

.

BS2DisplayConfig

```

typedef struct {
    uint32_t language;
    uint8_t background;
    uint8_t volume;
    uint8_t bgTheme;
    uint8_t dateFormat;
    uint16_t menuTimeout;
    uint16_t msgTimeout;
    uint16_t backlightTimeout;
    uint8_t displayDateTime;
}

```

```

uint8_t useVoice;
uint8_t timeFormat;
uint8_t homeFormation;
BS2_B00L useUserPhrase;
BS2_B00L queryUserPhrase;
uint8_t shortcutHome[BS2_MAX_SHORTCUT_HOME];
uint8_t tnaIcon[16];
uint8_t useScreenSaver;
uint8_t reserved1[31];
} BS2DisplayConfig;

```

1. language

0	
1	
2	

2. background

0	LOGO
1	NOTICE
2	SLIDE
3	PDF

3. volume

0-100 . 0

4. bgTheme

0	
1	
2	Slide show
3	PDF

5. dateFormat

0	YYYY/MM/DD
1	MM/DD/YYYY
2	DD/MM/YYYY

6. menuTimeout

sec . 0 (sec) . 0-255

0	
10	10
20	20 ()
30	30
40	40
50	50
60	60

7. *msgTimeout*

(ms) . 500-5000 ms

500	500
1000	1
2000	2 ()
3000	3
4000	4
5000	5

8. *backlightTimeout*

(sec)

0	0
10	10
20	20 ()
30	30
40	40
50	50
60	60

9. *displayDateTime*

flag .

10. *useVoice*

voice instruction

flag .

11. *timeFormat*

0	12
1	24

, Linux BioStation 2, BioStation L2, BioLite Net2, FaceLite 가 . (0 = 24 hour / 1 = 12 hour)

12. homeFormation

Home

1	
2	Shortcut 1
3	Shortcut 2
4	Shortcut 3
5	Shortcut 4

13. useUserPhrase

14. queryUserPhrase

true

15. shortcutHome

homeFormation

16. tnalcon

17. useScreenSaver

FaceStation 2, FaceStation F2 true

가

18. reserved1

BS2IpConfig

```
typedef struct {
    uint8_t connectionMode;
    uint8_t useDHCP;
    uint8_t useDNS;
    uint8_t reserved[1];
    char ipAddress[BS2_IPV4_ADDR_SIZE];
    char gateway[BS2_IPV4_ADDR_SIZE];
    char subnetMask[BS2_IPV4_ADDR_SIZE];
    char serverAddr[BS2_IPV4_ADDR_SIZE];
    uint16_t port;
    uint16_t serverPort;
    uint16_t mtuSize;
    uint8_t baseband;
    uint8_t reserved2[1];
    uint16_t sslServerPort;
    uint8_t reserved3[30];
} BS2IpConfig;
```

1. connectionMode

BioStar

direct mode(0x0) server

mode(0x1) 가 BioStar	. direct mode	BioStar		server mode direct mode
2. <i>useDHCP</i> DHCP	flag	.		
3. <i>useDNS</i> server addresss	server URL		flag	.
4. <i>reserved</i>		.		
5. <i>ipAddress</i> IP		.		
6. <i>gateway</i> IP		.		
7. <i>subnetMask</i>		.		
8. <i>serverAddr</i> connectionMode가 server mode		, BioStar	IP	.
9. <i>port</i> IP		.		
10. <i>serverPort</i> connectionMode가 server mode		, BioStar		.
11. <i>mtuSize</i> TCP	MTU ¹⁾	.		
12. <i>baseband</i> baseband	10mb/s 100mb/s	가	.	
13. <i>reserved2</i>		.		
14. <i>sslServerPort</i> connectionMode가 server ssl mode		, BioStar		.
15. <i>reserved3</i>		.		

BS2IpConfigExt

```
typedef struct {
    char dnsAddr[BS2_IPV4_ADDR_SIZE];
    char serverUrl[BS2_URL_SIZE];
    uint8_t reserved[32];
};
```

```
} BS2IpConfigExt;
```

1. *dnsAddr*

dns

2. *serverUrl*

BioStar URL , 256

3. *reserved*

BS2TNAConfig

```
typedef struct {
    uint8_t tnaMode;
    uint8_t tnaKey;
    uint8_t tnaRequired;
    uint8_t reserved[1];
    uint32_t tnaSchedule[BS2_MAX_TNA_KEY];
    uint8_t unused[BS2_MAX_TNA_KEY];
} BS2TNAInfo;

typedef struct {
    char tnaLabel[BS2_MAX_TNA_KEY][BS2_MAX_TNA_LABEL_LEN];
    uint8_t unused[BS2_MAX_TNA_KEY];
} BS2TNAExtInfo;

typedef struct {
    BS2TNAInfo tnaInfo;
    BS2TNAExtInfo tnaExtInfo;
    uint8_t reserved2[32];
} BS2TNAConfig;
```

1. *tnaMode*

0	
1	
2	
3	가
4	

2. *tnaKey*

Device Type	T&A Code	Mapped Key	Value
BioStation 2	BS2_TNA_UNSPECIFIED	(N/A)	0
	BS2_TNA_KEY_1	F1	1
	BS2_TNA_KEY_2	F2	2
	BS2_TNA_KEY_3	F3	3
	BS2_TNA_KEY_4	F4	4
	BS2_TNA_KEY_5	1	5
	BS2_TNA_KEY_6	2	6
	BS2_TNA_KEY_7	3	7
	BS2_TNA_KEY_8	4	8
	BS2_TNA_KEY_9	5	9
	BS2_TNA_KEY_10	6	10
	BS2_TNA_KEY_11	7	11
	BS2_TNA_KEY_12	8	12
	BS2_TNA_KEY_13	9	13
	BS2_TNA_KEY_14	Call	14
	BS2_TNA_KEY_15	0	15
BS2_TNA_KEY_16	Esc	16	

3. *tnaRequired*
가 1

flag

4. *reserved*

5. *tnaSchedule*
가

6. *unused*

7. *tnaLabel*

8. *unused*

BS2CardConfig

```
typedef struct {
    uint8_t primaryKey[6];
    uint8_t reserved1[2];
    uint8_t secondaryKey[6];
    uint8_t reserved2[2];
    uint16_t startBlockIndex;
    uint8_t reserved[6];
} BS2MifareCard;
```



```

typedef struct {
    uint8_t primaryKey[8];
    uint8_t secondaryKey[8];
    uint16_t startBlockIndex;
    uint8_t reserved[6];
} BS2IClassCard;

typedef struct {
    uint8_t primaryKey[16];
    uint8_t secondaryKey[16];
    uint8_t appID[3];
    uint8_t fileID;
    uint8_t encryptionType;
    uint8_t operationMode;
    uint8_t reserved[2];
} BS2DesFireCard;

typedef struct {
    uint8_t byteOrder;
    uint8_t useWiegandFormat;
    uint8_t dataType;
    uint8_t useSecondaryKey;
    BS2MifareCard mifare;
    BS2IClassCard iclass;
    BS2DesFireCard desfire;
    uint8_t formatID;
    uint8_t cipher;
    uint8_t smartCardByteOrder;
    uint8_t reserved[22];
} BS2CardConfig;

```

1. *primaryKey*

Mifare card

2. *reserved1*

3. *secondaryKey*

Mifare card

4. *reserved2*

5. *startBlockIndex*

Mifare data storage start block index

6. *reserved*

7. *primaryKey*

IClass card

8. *secondaryKey*

IClass card

9. *startBlockIndex*

Mifare data storage start block index

10. *reserved*

11. *primaryKey*

DesFire card

12. *secondaryKey*

DesFire card

13. *appId*

DESFire

14. *fileID*

DESFire 가

15. *encryptionType*

0	DES/3DES
1	AES

16. *operationMode*

()

0	(PICC master key)
1	(App master key)

17. *reserved*

18. *byteOrder*

. 0 MSB²⁾ , 1 LSB³⁾ .

19. *useWiegandFormat*

Wiegand flag

20. *dataType*

Card

0	
1	
2	UTF16
3	BCD

21. useSecondaryKey

flag

22. formatID

BioStar

card configuration

가

23. cipher

Keypad card id

0, Xpass 2, Xpass D2 Gangbox Keypad

0	
1	

24. smartCardByteOrder

[+2.8.2]

smart card data

MSB

LSB

controller

, byte 가

smartCardByteOrder

, MSB/LSB

0	MSB
1	LSB

25. reserved

BS2FingerprintConfig

```
typedef struct {
    uint8_t securityLevel;
    uint8_t fastMode;
    uint8_t sensitivity;
    uint8_t sensorMode;
    uint16_t templateFormat;
    uint16_t scanTimeout;
    uint8_t successiveScan;
    uint8_t advancedEnrollment;
    uint8_t showImage;
    uint8_t lfdLevel;
    bool checkDuplicate;

    uint8_t reserved3[31];
} BS2FingerprintConfig;
```

1. securityLevel

9. *showImage*

flag

10. *lfdLevel*

0	
1	
2	
3	

11. *checkDuplicate*

[+ V2.6.4] true

12. *reserved3*

BS2Rs485Config

```
typedef struct {
    uint8_t supportConfig;
    uint8_t useExceptionCode;
    uint8_t exceptionCode[BS2_RS485_MAX_FAIL_CODE_LEN];
    uint8_t outputFormat;
    uint8_t osdpID;
    uint8_t reserved[4];
} BS2IntelligentPDInfo;

typedef struct {
    uint32_t baudRate;
    uint8_t channelIndex;
    uint8_t useRegistance;
    uint8_t numOfDevices;
    uint8_t reserved[1];
    BS2Rs485SlaveDevice slaveDevices[BS2_RS485_MAX_SLAVES_PER_CHANNEL];
} BS2Rs485Channel;

typedef struct {
    uint8_t mode;
    uint8_t numOfChannels;
    uint8_t reserved[2];
    BS2IntelligentPDInfo intelligentInfo;
    uint8_t reserved1[16];
    BS2Rs485Channel channels[BS2_RS485_MAX_CHANNELS];
} BS2Rs485Config;
```

1. *supportConfig*

[+V2.8] 0

Intelligent PD(Peripheral Device)

2. *useExceptionCode*

[+V2.8]

3. *exceptionCode*

[+V2.8]

가

가 0(0x0000000000000000)

가

4. *outputFormat*

[+V2.8]

0 ID가, 1

ID가

5. *osdpID*

[+V2.8]

ACU

0~127

unique

6. *reserved*

[+V2.8]

7. *baudRate*

RS485

9600
19200
38400
57600
115200

8. *channelIndex*

(가) RS485 network

9. *useRegistance*

flag

10. *numOfDevices*

11. *reserved*

12. *slaveDevices*

32

13. *mode*

RS485

flag

0

1	Master
2	Slave
3	Standalone

14. *numOfChannels*
RS485

15. *reserved*

16. *intelligentInfo*
[+V2.8] Intelligent Slave 가 , mode가 default(Standalone) OSDP

17. *reserved1*

18. *channels*
RS485 4

BS2WiegandConfig

```
typedef struct {
    uint32_t length;
    uint8_t idFields[BS2_WIEGAND_MAX_FIELDS][BS2_WIEGAND_FIELD_SIZE];
    uint8_t parityFields[BS2_WIEGAND_MAX_PARITIES][BS2_WIEGAND_FIELD_SIZE];
    BS2_WIEGAND_PARITY parityType[BS2_WIEGAND_MAX_PARITIES];
    uint8_t parityPos[BS2_WIEGAND_MAX_PARITIES];
} BS2WiegandFormat;

typedef struct {
    uint8_t mode;
    uint8_t useWiegandBypass;
    uint8_t useFailCode;
    uint8_t failCode;
    uint16_t outPulseWidth;
    uint16_t outPulseInterval;
    uint32_t formatID;
    BS2WiegandFormat format;
    uint16_t wiegandInputMask;
    uint16_t wiegandCardMask;
    uint8_t wiegandCSNIndex;
    uint8_t useWiegandUserID;
    uint8_t reserved[26];
} BS2WiegandConfig;
```

1. *length*

Wiegand

2. idFields

4 id field 가 , field id bit
 , Standard 26bit wiegand card data "P FFFFFFFF
 NNNNNNNNNNNNNNNNNN P" Facility Code " 0 11111111
 0000000000000000 0 " 0x01FE0000 가 , Card Number 0x0001FFFE

```
// for Facility Code
idFields[][28] = 0x01;
idFields[][29] = 0xFE;
idFields[][30] = 0x00;
idFields[][31] = 0x00;

// for Card Number
idFields[1][28] = 0x00;
idFields[1][29] = 0x01;
idFields[1][30] = 0xFF;
idFields[1][31] = 0xFE;
```

3. parityFields

4 가 , id Field

4. parityType

0	parity
1	parity
2	parity

5. parityPos

Wiegand

6. mode

Wiegand

0	
1	
2	

7. useWiegandBypass

0	
1	

8. useFailCode

Fail Code

9. failCode

Fail Code

0x00
0xFF

10. outPulseWidth

20 ~ 100 us 가

11. outPulseInterval

200 ~ 20000 us 가

12. formatID

Wiegand

13. format

WiegandFormat

14. wiegandInputMask

Master Slave wiegand wiegand mask

15. wiegandCardMask

Master mask

16. wiegandCSNIndex

Mifare EM Wiegand out
BS2CardConfig useWiegandFormat

17. useWiegandUserID

Wiegand Card ID ID

0	
1	Card ID
2	ID

18. reserved

BS2WiegandDeviceConfig

```
typedef struct {
    uint32_t deviceID;
    uint16_t port;
    uint8_t switchType;
    uint8_t reserved[1];
} BS2WiegandTamperInput;
```

```

typedef struct {
    uint32_t deviceID;
    uint16_t port;
    uint8_t reserved[10];
} BS2WiegandLedOutput;

typedef struct {
    uint32_t deviceID;
    uint16_t port;
    uint8_t reserved[34];
} BS2WiegandBuzzerOutput;

typedef struct {
    BS2WiegandTamperInput tamper;
    BS2WiegandLedOutput led[BS2_WIEGAND_STATUS_NUM];
    BS2WiegandBuzzerOutput buzzer;
    uint32_t reserved[32];
} BS2WiegandDeviceConfig;

```

1. *deviceID*

Wiegand card reader tamper

2. *port*

Wiegand card reader tamper

3. *switchType*

가 'off' 가 on trigger

0	Normally Open
1	Normally Closed

4. *reserved*

5. *deviceID*

Wiegand card reader led

6. *port*

Wiegand card reader led

7. *reserved*

8. *deviceID*

Wiegand card reader buzzer

9. *port*

Wiegand card reader buzzer

10. *reserved*

10. *led*

Wiegand card reader led

2

0	led
1	led

BS2InputConfig

```

typedef struct {
    uint16_t minValue;
    uint16_t maxValue;
} BS2SVInputRange;

typedef struct {
    uint32_t deviceID;
    uint16_t port;
    uint8_t reserved[10];
} BS2WiegandLedOutput;

typedef struct {
    BS2SVInputRange shortInput;
    BS2SVInputRange openInput;
    BS2SVInputRange onInput;
    BS2SVInputRange offInput;
} BS2SupervisedInputConfig;

typedef struct {
    uint8_t numInputs;
    uint8_t numSupervised;
    uint16_t reserved;
    struct {
        uint8_t portIndex;
        uint8_t enabled;
        uint8_t supervised_index;
        uint8_t reserved[5];
        BS2SupervisedInputConfig config;
    } supervised_inputs[BS2_MAX_INPUT_NUM];
} BS2InputConfig;

```

1. *minValue*

0 ~ 3300(3.3v)

2. *maxValue*

0 ~ 3300(3.3v)

3. *shortInput*

short input

4. *openInput*

open input

5. *onInput*
on input

6. *offInput*
off input

7. *numInputs*

8. *numSupervised*
supervised

9. *portIndex*

10. *enabled*
supervised input flag

11. *supervised_index*
supervised input

0	1k
1	2.2k
2	4.7k
3	10k
255	

12. *reserved*

13. *config*
supervised 가 , supervised input

BS2WlanConfig

```
typedef struct {
    uint8_t enabled;
    uint8_t operationMode;
    uint8_t authType;
    uint8_t encryptionType;
    char essid[BS2_WLAN_SSID_SIZE];
    char authKey[BS2_WLAN_KEY_SIZE];
    uint8_t reserved2[32];
} BS2WlanConfig;
```

1. *enabled*

2. *operationMode*

0	infrastructure
1	Ad-hoc

3. *authType*

0	Open
1	Shared
2	WPA-PSK
3	WPA2-PSK

4. *encryptionType*

0	
1	WEP
2	TKIP/AES
3	AES
3	TKIP

5. *ssid*

6. *authKey*

7. *reserved*

BS2Trigger

```
typedef struct {  
    uint16_t code;  
    uint8_t reserved[2];  
} BS2EventTrigger;  
  
typedef struct {  
    uint8_t port;  
    uint8_t switchType;  
    uint16_t duration;  
    uint32_t scheduleID;  
} BS2InputTrigger;  
  
typedef struct {  
    uint32_t type;
```

```

    uint32_t scheduleID;
} BS2ScheduleTrigger;

typedef struct {
    uint32_t deviceID;
    uint8_t type;
    uint8_t reserved[3];

    union {
        BS2EventTrigger event;
        BS2InputTrigger input;
        BS2ScheduleTrigger schedule;
    }
} BS2Trigger;

```

1. code

trigger event log .

2. reserved

.

3. port

trigger .

4. switchType

가 'off' 가 on trigger .

0	Normally Open
1	Normally Closed

5. duration

trigger (ms) , 100 .

6. scheduleID

trigger .

7. type

schedule trigger .

0	schedule trigger
1	schedule trigger

8. scheduleID

trigger .

9. deviceID

trigger .

10. type

trigger .

0	None
1	Event trigger
2	Input trigger
3	Schedule trigger

BS2Action

```
typedef struct {
    uint32_t signalID;
    uint16_t count;
    uint16_t onDuration;
    uint16_t offDuration;
    uint16_t delay;
} BS2Signal;

typedef struct {
    uint8_t portIndex;
    uint8_t reserved[3];
    BS2Signal signal;
} BS2OutputPortAction;

typedef struct {
    uint8_t relayIndex;
    uint8_t reserved[3];
    BS2Signal signal;
} BS2RelayAction;

typedef struct {
    uint8_t color;
    uint8_t reserved[1];
    uint16_t duration;
    uint16_t delay;
} BS2LedSignal;

typedef struct {
    uint16_t count;
    uint8_t reserved[2];
    BS2LedSignal signal[3];
} BS2LedAction;

typedef struct {
    uint8_t tone;
    uint8_t fadeout;
    uint16_t duration;
    uint16_t delay;
} BS2BuzzerSignal;

typedef struct {
    uint16_t count;
```

```

    uint8_t reserved[2];
    BS2BuzzerSignal signal[3];
} BS2BuzzerAction;

typedef struct {
    uint8_t duration;
    uint8_t reserved[3];
    uint32_t displayID;
    uint32_t resourceID;
} BS2DisplayAction;

typedef struct {
    uint8_t count;
    uint16_t soundIndex;
    uint8_t reserved[5];
} BS2SoundAction;

typedef struct {
    uint32_t deviceID;
    uint8_t type;
    uint8_t stopFlag;
    uint16_t delay;
    union {
        BS2RelayAction relay;
        BS2OutputPortAction outputPort;
        BS2DisplayAction display;
        BS2SoundAction sound;
        BS2LedAction led;
        BS2BuzzerAction buzzer;
    };
} BS2Action;

```

1. *signalID*

2. *count*

3. *onDuration*

on (ms)

4. *offDuration*

off (ms)

5. *delay*

(ms) , count(2),
onDuration(100), offDuration(100), delay(50)

50ms	signal on(100)	signal off(100)	signal on(100)	signal off(100)
-------------	-----------------------	------------------------	-----------------------	------------------------

6. *portIndex*

TTL

7. reserved

8. relayIndex
Relay

9. reserved

10. color
LED

0	LED Off
1	LED
2	LED
3	LED
4	LED
5	LED
6	LED
7	LED

11. reserved

12. duration
LED (ms)

13. delay
LED (ms)

14. count
LED 0 -1

15. reserved

16. tone
Buzzer ()

0	
1	
2	
3	

17. count
Buzzer 0 -1

18. reserved

19. *duration*

Display (ms) .

20. *reserved*

Display (ms) .

21. *displayID*

22. *resourceID*

23. *count*

Sound

24. *soundIndex*

Sound resource

0	Welcome sound
1	Auth success sound
2	Auth fail sound

25. *deviceID*

Action

26. *type*

Action

[DoorModule-20, CoreStation-40]

Action type relay TTL(Output) , action 가
DM20, CS40 , action type relay action (6)
(TTL 가)

[DM20]

- Action type : Relay
- relay.relayIndex : 0 ~ 3 (RELAY 0 ~ 3)
- relay.relayIndex : 4 ~ 9 (OUTPUT 0 ~ 5)

[CS40]

- Action type : Relay
- relay.relayIndex : 0 ~ 3 (RELAY 0 ~ 3)
- relay.relayIndex : 4 ~ 11 (OUTPUT 0 ~ 7)

0	None
1	Lock device

2	Unlock device
3	Reboot device
4	Release alarm
5	General input
6	Relay action
7	TTL action
8	Sound action
9	Display action
10	Buzzer action
11	Led action
12	Fire alarm input
13	Auth Success(Access granted)
14	Auth Fail(Access denied)
15	Lift action

27. *stopFlag*

Action

1 door sensor 가 action
 2 action API id 가 ,
 가 action
 stopFlag 2 action

0	
1	
2	(V2.6.0 가)

28. *delay*

Action (ms)

BS2TriggerActionConfig

```
typedef struct {
    uint8_t numItems;
    uint8_t reserved[3];
    BS2TriggerAction items[BS2_MAX_TRIGGER_ACTION];
    uint8_t reserved2[32];
} BS2TriggerActionConfig;
```

1. *numItems*
 trigger action

2. *reserved*

3. *items*

trigger action 128

4. *reserved2*

BS2EventConfig

```
typedef struct {
    uint32_t numImageEventFilter;
    struct {
        uint8_t mainEventCode;
        uint8_t reserved[3];
        uint32_t scheduleID;
    } imageEventFilter[BS2_EVENT_MAX_IMAGE_CODE_COUNT];
    uint8_t reserved[32];
} BS2EventConfig;
```

1. *numImageEventFilter*

image log filter

2. *mainEventCode*

image log log main code

3. *reserved*

4. *scheduleID*

image log

5. *reserved*

BS2WiegandMultiConfig

```
typedef struct {
    uint32_t formatID;
    BS2WiegandFormat format;
    uint8_t reserved[32];
} BS2WiegandInConfig;

typedef struct {
    BS2WiegandInConfig formats[MAX_WIEGAND_IN_COUNT];
    uint8_t reserved[32];
} BS2WiegandMultiConfig;
```

1. *formatID*

WiegandFormat Index

2. *format*

WiegandFormat

3. *reserved*4. *formats*

WiegandInConfig 15 가

5. *reserved***BS1CardConfig**

```

typedef struct {
    enum {
        MIFARE_KEY_SIZE = 6,
        MIFARE_MAX_TEMPLATE = 4,

        VALID_MAGIC_NO = 0x1f1f1f1f,
    };

    // Options
    uint32_t    magicNo;
    uint32_t    disabled;
    uint32_t    useCSNOnly;          // default 0
    uint32_t    bioentryCompatible; // default 0

    // Keys
    uint32_t    useSecondaryKey;
    uint32_t    reserved1;
    uint8_t     primaryKey[MIFARE_KEY_SIZE];
    uint8_t     reserved2[2];
    uint8_t     secondaryKey[MIFARE_KEY_SIZE];
    uint8_t     reserved3[2];

    // Layout
    uint32_t    cisIndex;
    uint32_t    numOfTemplate;
    uint32_t    templateSize;
    uint32_t    templateStartBlock[MIFARE_MAX_TEMPLATE];

    uint32_t    reserve4[15];
} BS1CardConfig;

```

1. *magicNo*2. *disabled*

flag

3. *useCSNOnly*

CSN

4. *bioentryCompatible*

bioentry

5. *useSecondaryKey*

6. *reserved1*

7. *primaryKey*

8. *reserved2*

9. *secondaryKey*

10. *reserved3*

11. *cisIndex*

cis

12. *numOfTemplate*

13. *templateSize*

14. *templateStartBlock*

data storage start block index

15. *reserved4*

BS2SystemConfigExt

```
typedef struct {  
    uint8_t primarySecureKey[SEC_KEY_SIZE];  
    uint8_t secondarySecureKey[SEC_KEY_SIZE];  
  
    uint8_t reserved3[32];  
} BS2SystemConfigExt;
```

1. *primarySecureKey*

Master-Slave

2. *secondarySecureKey*

cMaster-Slave

3. reserved3

BS2VoipConfig

```

typedef struct {
    BS2_URL          serverUrl;          ///  

    BS2_PORT         serverPort;        ///  

    BS2_USER_ID      userID;            ///  

    BS2_USER_ID      userPW;           ///  

  

    uint8_t          exitButton;        ///  

    uint8_t          dtmfMode;          ///  

    BS2_B00L         bUse;              ///  

    uint8_t          reseverd[1];       ///  

  

    uint32_t         numPhonBook;
    BS2UserPhoneItem phonebook[BS2_VOIP_MAX_PHONEBOOK]; ///  

  

    uint8_t          reserved2[32];     ///  

} BS2VoipConfig;

```

1. serverUrl

BioStar URL , 256

2. serverPort

connectionMode가 server mode , BioStar

3. userID

4. userPW

5. exitButton

. (*, #, 0~9)

0	*
1	#
2 ~ 11	0 ~ 9

6. dtmfMode

7. bUse

8. *reseverd*

9. *numPhonBook*

10. *phonebook*

32

8. *reserved2*

BS2FaceConfig

```
typedef struct {
    uint8_t      securityLevel;
    uint8_t      lightCondition;
    uint8_t      enrollThreshold;
    uint8_t      detectSensitivity;

    uint16_t     enrollTimeout;
    uint8_t      lfdLevel;
    bool         quickEnrollment;

    uint8_t      previewOption;
    bool         checkDuplicate;
    uint8_t      operationMode;
    uint8_t      maxRotation;

    // Deprecated
    struct {
        uint16_t  min;
        uint16_t  max;
    } faceWidth;

    // Deprecated
    struct {
        uint16_t  x;
        uint16_t  width;
    } searchRange;

    struct {
        uint8_t  min;           // 30 ~ 100
        uint8_t  max;           // 40 ~ 100, 255
    } detectDistance;         ///< 2 bytes

    BS2_B00L wideSearch;      ///< 1 byte
    uint8_t  unused;

    uint8_t  reserved[14];    ///< 14 bytes (reserved)
}
```



```
} BS2FaceConfig;
```

1. *securityLevel*

0	
1	
2	

2. *lightCondition*

0	
1	
2	
3	[+ 2.8] (FaceStation F2)

3. *enrollThreshold*

0	THRESHOLD_0 ()
1	THRESHOLD_1
2	THRESHOLD_2
3	THRESHOLD_3
4	THRESHOLD_4 ()
5	THRESHOLD_5
6	THRESHOLD_6
7	THRESHOLD_7
8	THRESHOLD_8
9	THRESHOLD_9 ()

4. *detectSensitivity*

0	
1	
2	
3	

5. *enrollTimeout*

FaceStation2, FaceLite :

60

BS2_FACE_ENROLL_TIMEOUT_MIN	30
BS2_FACE_ENROLL_TIMEOUT_MAX	60
BS2_FACE_ENROLL_TIMEOUT_DEFAULT	BS2_FACE_ENROLL_TIMEOUT_MAX

FaceStation F2 : [+ V2.7.1]

20

BS2_FACE_EX_ENROLL_TIMEOUT_MIN	10
BS2_FACE_EX_ENROLL_TIMEOUT_MAX	20
BS2_FACE_EX_ENROLL_TIMEOUT_DEFAULT	BS2_FACE_EX_ENROLL_TIMEOUT_MAX

6. *lfdLevel*

[+ 2.6.3]

FaceStation2, FaceLite : 0

FaceStation F2 : [+ 2.7.1] 1

0	
1	
2	
3	

7. *quickEnrollment*

[+ 2.6.3]

true 가 1 , false 3
false

8. *previewOption*

[+ 2.6.3] IR 가 , preview

FaceLite

0	Preview
1	preview , 1/2
2	preview

9. *checkDuplicate*

[+ 2.6.4] true

10. *operationMode*

[+ 2.7.1] FaceStation F2 , Fusion

0	Fusion	Visual matching + IR matching	
1	Visual	Visual matching	
2	Visual + IR	Visual matching, IR	

11. *maxRotation*

[+ 2.7.1] FaceStation F2

FSF2 () 가

maxRotation , 15

12. *faceWidth*

[+ 2.7.1] FaceStation F2

[+ 2.8.3] BioStation 3

	(min)	(max)
FSF2	66	250
BS3	-	-

13. *searchRange*

[+ 2.7.1] FaceStation F2

[+ 2.8.3] BioStation 3

	(x)	(width)
FSF2	144	432
BS3	-	-

14. *detectDistance*

[+ 2.8.3] BioStation 3

faceWidth

()

cm

, 10

	()	()	()	()	()	()	()
BS3	30	100	60	40	100	255	100

15. *wideSearch*

[+ 2.8.3] BioStation 3

x width

searchRange

(false)

(true)

(true)

가

false

16. *unused*

17. *reserved*

BS2Rs485ConfigEX

```
typedef struct {
```

```

uint32_t baudRate;
uint8_t channelIndex;
uint8_t useRegistance;
uint8_t numOfDevices;
uint8_t reserved[1];
BS2Rs485SlaveDeviceEX slaveDevices[BS2_RS485_MAX_SLAVES_PER_CHANNEL];
} BS2Rs485ChannelEX;

typedef struct {
uint8_t mode[BS2_RS485_MAX_CHANNELS_EX];
uint8_t numOfChannels;
uint8_t reserved[2];
uint8_t reserved1[32];
BS2Rs485ChannelEX channels[BS2_RS485_MAX_CHANNELS];
} BS2Rs485ConfigEX;

```

1. *baudRate*

RS485

9600
19200
38400
57600
115200

2. *channelIndex*

RS485 network

3. *useRegistance*

flag

4. *numOfDevices*

5. *slaveDevices*

32

6. *mode*

RS485

flag

0	
1	Master
2	Slave
3	Standalone

7. *numOfChannels*

RS485

8. *reserved*

9. *reserved1*

10. *channels*

RS485

8

BS2CardConfigEx

```
typedef struct {
    uint8_t oid_ADF[13];           /// //
    // {0x2A, 0x85, 0x70, 0x81, 0x1E, 0x10, 0x00, 0x07, 0x00, 0x00, 0x02, 0x00, 0x00}
    uint8_t size_ADF;             //
    uint8_t reserved1[2];        ///
    uint8_t oid_DataObjectID[8];
    uint16_t size_DataObject[8];
    uint8_t primaryKeyAuth[16];  //
    uint8_t secondaryKeyAuth[16]; /// //
    uint8_t reserved2[24];
} BS2SeosCard;
typedef struct {
    BS2SeosCard seos;
    uint8_t reserved[24];
} BS2CardConfigEx;
```

1. *oid_ADF*

ADF . (.)

2. *size_ADF*

ADF size

3. *reserved1*

4. *oid_DataObjectID*

DataObjectID

5. *size_DataObject*

DataObject size

6. *primaryKeyAuth*

Seoscard

7. *secondaryKeyAuth*

Seoscard

8. *reserved2*

9. *seos*

BS2SeosCard

10. reserved

BS2DstConfig

```
enum {
    BS2_MAX_DST_SCHEDULE = 2,
};

typedef struct {
    uint16_t year;           // year, 0 means every year.
    uint8_t month;          // [0, 11] : months since January
    int8_t ordinal;         // [0, -1] : first, second, ..., last
    uint8_t weekDay;        // [0, 6] : days since Sunday
    uint8_t hour;           // [0, 23]
    uint8_t minute;         // [0, 59]
    uint8_t second;         // [0, 59]
} BS2WeekTime;

typedef struct {
    BS2WeekTime startTime;
    BS2WeekTime endTime;
    int32_t timeOffset;     // in seconds
    uint8_t reserved[4];
} BS2DstSchedule;

typedef struct {
    uint8_t numSchedules;
    uint8_t reserved[31];

    BS2DstSchedule schedules[BS2_MAX_DST_SCHEDULE];
} BS2DstConfig;
```

1. year

, 0

2. month

, 0 11 [1 -12]

가

3. ordinal

0

4. weekDay

, 0

, 1

5. hour

24

6. *minute*

7. *second*

8. *startTime*

9. *endTime*

10. *timeOffset*

DST

1, 3600

11. *reserved*

12. *numSchedules*

DST schedule

13. *schedules*

DST schedule 2

BS2Configs

```
typedef struct {
    uint32_t configMask;
    BS2FactoryConfig factoryConfig;
    BS2SystemConfig systemConfig;
    BS2AuthConfig authConfig;
    BS2StatusConfig statusConfig;
    BS2DisplayConfig displayConfig;
    BS2IpConfig ipConfig;
    BS2IpConfigExt ipConfigExt;
    BS2TNACConfig tnaConfig;
    BS2CardConfig cardConfig;
    BS2FingerprintConfig fingerprintConfig;
    BS2Rs485Config rs485Config;
    BS2WiegandConfig wiegandConfig;
    BS2WiegandDeviceConfig wiegandDeviceConfig;
    BS2InputConfig inputConfig;
    BS2WlanConfig wlanConfig;
    BS2TriggerActionConfig triggerActionConfig;
    BS2EventConfig eventConfig;
    BS2WiegandMultiConfig wiegandMultiConfig;
    BS1CardConfig card1xConfig;
    BS2SystemConfigExt systemExtConfig;
    BS2VoipConfig voipConfig;
    BS2FaceConfig faceConfig;
}
```

```
} BS2Configs;
```

1. configMask

configuration 가 mask .

0x0000	None
0x0001	Factory configuration
0x0002	System configuration
0x0004	TCP/IP configuration
0x0008	RS485 configuration
0x0010	Wireless LAN configuration
0x0020	Authentication configuration
0x0040	Card configuration
0x0080	Fingerprint configuration
0x0100	Face configuration
0x0200	Trigger Action configuration
0x0400	Display configuration
0x0800	Sound configuration
0x1000	Status Signal(LED, Buzzer) configuration
0x2000	Wiegand configuration
0x4000	USB configuration
0x8000	Time and Attendance configuration
0x10000	Videophone configuration
0x20000	Interphone configuration
0x40000	Voice over IP configuration
0x80000	Input(Supervised input) configuration
0x100000	Wiegand IO Device configuration
0x200000	Time and Attendance configuration
0x400000	DNS and Server url configuration
0x800000	Event configuration
0x1000000	1x Card configuration
0x2000000	Multi-Wiegand configuration
0x4000000	Extended System configuration
0x8000000	Daylight Saving configuration (Deprecated)
0x10000000	RS485 Extended configuration
0x20000000	Extended Card configuration
0x40000000	Daylight Saving configuration
0xFFFFFFFF	All configuration

BS2IPV6Config

```
enum {
    BS2_MAX_IPV6_ALLOCATED_ADDR = 8,
};
```



```

typedef struct {
    uint8_t useIPv6;
    uint8_t reserved1;
    uint8_t useDhcpV6;
    uint8_t useDnsV6;
    uint8_t reserved[1];
    char staticIpAddressV6[BS2_IPV6_ADDR_SIZE];
    char staticGatewayV6[BS2_IPV6_ADDR_SIZE];
    char dnsAddrV6[BS2_IPV6_ADDR_SIZE];
    char serverIpAddressV6[BS2_IPV6_ADDR_SIZE];
    uint16_t serverPortV6;
    uint16_t sslServerPortV6;
    uint16_t portV6;
    uint8_t numOfAllocatedAddressV6;
    uint8_t numOfAllocatedGatewayV6;
    uint8_t reserved[8];
    char
allocatedIpAddressV6[BS2_IPV6_ADDR_SIZE][BS2_MAX_IPV6_ALLOCATED_ADDR];
    char
allocatedGatewayV6[BS2_IPV6_ADDR_SIZE][BS2_MAX_IPV6_ALLOCATED_ADDR];
} BS2IpConfig;

```

1. *useIPv6*

IP V6 flag .

2. *reserved1*

.

3. *useDhcpV6*

DHCP flag .

4. *useDnsV6*

server addresss server URL flag .

5. *staticIpAddressV6*

IP V6 .

6. *staticGatewayV6*

IP V6 .

7. *dnsAddrV6*

DNS V6 .

8. *serverIpAddressV6*

connectionMode가 server mode , BioStar IP V6 .

9. *serverPortV6*

connectionMode가 server mode , BioStar .

10. *sslServerPortV6*

connectionMode가 server mode , ssl .

- 11. *portV6*
IP V6
- 12. *numOfAllocatedAddressV6*
IP V6
- 13. *numOfAllocatedGatewayV6*
IP V6
- 14. *reserved*
- 15. *allocatedIpAddressV6*
IP V6 . numOfAllocatedAddressV6
- 16. *allocatedGatewayV6*
IP V6 . numOfAllocatedGatewayV6

BS2DesFireCardConfigEx

```

typedef struct {
    uint8_t appMasterKey[16];
    uint8_t fileReadKey[16];
    uint8_t fileWriteKey[16];
    uint8_t fileReadKeyNumber;
    uint8_t fileWriteKeyNumber;
    uint8_t reserved[2];
} BS2DesFireAppLevelKey;           ///< 52 bytes

typedef struct {
    BS2DesFireAppLevelKey desfireAppKey;    ///< 52 bytes
    uint8_t reserved[16];
} BS2DesFireCardConfigEx;          ///< 68 bytes

```

- 1. *appMasterKey*
DesFire application master key
- 2. *fileReadKey*
key
- 3. *fileWriteKey*
key
- 4. *fileReadKeyNumber*
key key index
- 5. *fileWriteKeyNumber*
key key index

6. reserved

7. *desfireAppKey*

DesFire

8. reserved

BS2AuthConfigExt

```
typedef struct {
    uint32_t extAuthSchedule[BS2_MAX_NUM_OF_EXT_AUTH_MODE];
    uint8_t useGlobalAPB;
    uint8_t globalAPBFailAction;
    uint8_t useGroupMatching;
    uint8_t reserved;

    uint8_t reserved2[4];

    uint8_t usePrivateAuth;
    uint8_t faceDetectionLevel;
    uint8_t useServerMatching;
    uint8_t useFullAccess;

    uint8_t matchTimeout;
    uint8_t authTimeout;
    uint8_t numOperators;
    uint8_t reserved3[1];

    struct {
        char userID[BS2_USER_ID_SIZE];
        uint8_t level;
        uint8_t reserved[3];
    } operators[BS2_MAX_OPERATORS];

    uint8_t reserved4[256];
} BS2AuthConfigExt;
```

1. *extAuthSchedule*

가

가

0

11	BS2_EXT_AUTH_MODE_FACE_ONLY	
12	BS2_EXT_AUTH_MODE_FACE_FINGERPRINT	+
13	BS2_EXT_AUTH_MODE_FACE_PIN	+ PIN

14	BS2_EXT_AUTH_MODE_FACE_FINGERPRINT_OR_PIN	+ /PIN
15	BS2_EXT_AUTH_MODE_FACE_FINGERPRINT_PIN	+ + PIN
16	BS2_EXT_AUTH_MODE_FINGERPRINT_ONLY	
17	BS2_EXT_AUTH_MODE_FINGERPRINT_FACE	+
18	BS2_EXT_AUTH_MODE_FINGERPRINT_PIN	+ PIN
19	BS2_EXT_AUTH_MODE_FINGERPRINT_FACE_OR_PIN	+ /PIN
20	BS2_EXT_AUTH_MODE_FINGERPRINT_FACE_PIN	+ + PIN
21	BS2_EXT_AUTH_MODE_CARD_ONLY	
22	BS2_EXT_AUTH_MODE_CARD_FACE	+
23	BS2_EXT_AUTH_MODE_CARD_FINGERPRINT	+
24	BS2_EXT_AUTH_MODE_CARD_PIN	+ PIN
25	BS2_EXT_AUTH_MODE_CARD_FACE_OR_FINGERPRINT	+ /
26	BS2_EXT_AUTH_MODE_CARD_FACE_OR_PIN	+ /PIN
27	BS2_EXT_AUTH_MODE_CARD_FINGERPRINT_OR_PIN	+ /PIN
28	BS2_EXT_AUTH_MODE_CARD_FACE_OR_FINGERPRINT_OR_PIN	+ / /PIN
29	BS2_EXT_AUTH_MODE_CARD_FACE_FINGERPRINT	+ +
30	BS2_EXT_AUTH_MODE_CARD_FACE_PIN	+ + PIN
31	BS2_EXT_AUTH_MODE_CARD_FINGERPRINT_FACE	+ +
32	BS2_EXT_AUTH_MODE_CARD_FINGERPRINT_PIN	+ + PIN
33	BS2_EXT_AUTH_MODE_CARD_FACE_OR_FINGERPRINT_PIN	+ / + PIN
34	BS2_EXT_AUTH_MODE_CARD_FACE_FINGERPRINT_OR_PIN	+ + /PIN
35	BS2_EXT_AUTH_MODE_CARD_FINGERPRINT_FACE_OR_PIN	+ + /PIN
36	BS2_EXT_AUTH_MODE_ID_FACE	ID +
37	BS2_EXT_AUTH_MODE_ID_FINGERPRINT	ID +
38	BS2_EXT_AUTH_MODE_ID_PIN	ID + PIN
39	BS2_EXT_AUTH_MODE_ID_FACE_OR_FINGERPRINT	ID + /
40	BS2_EXT_AUTH_MODE_ID_FACE_OR_PIN	ID + /PIN
41	BS2_EXT_AUTH_MODE_ID_FINGERPRINT_OR_PIN	ID + /PIN
42	BS2_EXT_AUTH_MODE_ID_FACE_OR_FINGERPRINT_OR_PIN	ID + / /PIN
43	BS2_EXT_AUTH_MODE_ID_FACE_FINGERPRINT	ID + +
44	BS2_EXT_AUTH_MODE_ID_FACE_PIN	ID + + PIN
45	BS2_EXT_AUTH_MODE_ID_FINGERPRINT_FACE	ID + +

46	BS2_EXT_AUTH_MODE_ID_FINGERPRINT_PIN	ID + PIN +
47	BS2_EXT_AUTH_MODE_ID_FACE_OR_FINGERPRINT_PIN	ID + / + PIN
48	BS2_EXT_AUTH_MODE_ID_FACE_FINGERPRINT_OR_PIN	ID + /PIN +
49	BS2_EXT_AUTH_MODE_ID_FINGERPRINT_FACE_OR_PIN	ID + /PIN +

2. useGlobalAPB

flag .

3. globalAPBFailAction

BioStar

0	APB
1	Soft APB
2	Hard APB

4. useGroupMatching

flag .

5. reserved

6. reserved2

7. usePrivateAuth

flag .

8. faceDetectionLevel

A2

Normal/Strict

가 ,

가

0 .

0	
1	Normal mode
2	Strict mode

A2 가 , FaceStation2 FaceLite .

9. useServerMatching

Matching server

flag .

10. *useFullAccess*

11. *matchTimeout*

(sec)

12. *authTimeout*

(sec)

13. *numOperators*

operator

14. *reserved3*

15. *userID*

16. *level*

가

0	
1	
2	
3	

Operator 가 , 가 operator **numOperators**

17. *reserved*

18. *reserved4*

BS2FaceConfigExt

```
typedef struct {
    uint8_t thermalCheckMode;
    uint8_t maskCheckMode;
    uint8_t reserved[2];

    uint8_t thermalFormat;
    uint8_t reserved2;

    uint16_t thermalThresholdLow;
    uint16_t thermalThresholdHigh;
}
```

```

uint8_t maskDetectionLevel;
uint8_t auditTemperature;

uint8_t useRejectSound;
uint8_t useOverlapThermal;
uint8_t useDynamicROI;
uint8_t faceCheckOrder;
} BS2FaceConfigExt;

```

1. thermalCheckMode

HARD , thermalThreshold ,
SOFT , thermalThreshold ,

thermalCheckMode가 (0)
thermalFormat, thermalThreshold, auditTemperature, useOverlapThermal
useRejectSound sound , faceCheckOrder

0		
1	(HARD)	
2	(SOFT)	

2. maskCheckMode

FaceStation F2

FaceStation 2

HARD , maskDetectionLevel ,
SOFT , maskDetectionLevel ,

maskCheckMode가 (0)
maskDetectionLevel
useRejectSound sound , faceCheckOrder 가

0		
1	(HARD)	
2	(SOFT)	

3. reserved

4. thermalFormat

0		
1		

5. reserved2

6. *thermalThresholdLow*

: FaceStation F2 V1.0.2, FaceStation 2 V1.5.0

100

가 100 (1°) 4500 (45°)

3200 (32°) 3200 (32°)

thermalThresholdHigh

7. *thermalThresholdHigh*

100

가 100 (1°) 4500 (45°)

3800(38°) 3800 (38°)

thermalThresholdLow

8. *maskDetectionLevel*

FaceStation F2

FaceStation 2

0		
1		
2		
3		

9. *auditTemperature*

10. *useRejectSound*

thermalThreshold maskDetectionLevel

11. *useOverlapThermal*

12. *useDynamicROI*

true

13. *faceCheckOrder*

ID , PIN

0		
1		
2		

BS2ThermalCameraConfig

```
typedef struct {
    uint8_t distance;
    uint8_t emissionRate;

    struct {
        uint16_t x;
        uint16_t y;
        uint16_t width;
        uint16_t height;
    } roi;

    uint8_t useBodyCompensation;
    int8_t compensationTemperature;
} BS2ThermalCameraConfig;
```

1. *distance* . cm 100 .
2. *emissionRate*
가 . 가 98 .
[95/97/98]
3. *roi*
ROI(Region of interest) , ,
(x, y) , (width, height) .
4. *useBodyCompensation* .
5. *compensationTemperature*
, 가 , ,
10 , -50 50 .

BS2BarcodeConfig

```
typedef struct {
    uint8_t useBarcode;
    uint8_t scanTimeout;
    uint8_t bypassData;
    uint8_t treatAsCSN;

    uint8_t useVisualBarcode;
    uint8_t motionSensitivity;
    uint8_t visualCameraScanTimeout;
    uint8_t reserved[9];
}
```

} BS2BarcodeConfig;

1. useBarcode

XS2-QR Barcode flag .

2. scanTimeout

Barcode scan .
4 , 4~10 가 .

4	BS2_BARCODE_TIMEOUT_DEFAULT	
4	BS2_BARCODE_TIMEOUT_MIN	
10	BS2_BARCODE_TIMEOUT_MAX	

3. bypassData

[+2.8.2] barcode , barcode .
32 byte (BS2CSNCard data) ,
BS2_SetBarcodeScanListener , 512 byte barcode

4. treatAsCSN

[+2.8.2] Barcode CSN .
XS2-QR 1.1.3 , false , ASCII code 32 126 .
barcode
(BS2_WriteQRCode)
true , barcode CSN .
, 가 barcode
, card type , CSN barcode

5. useVisualBarcode

[+ 2.9.1] Visual barcode flag .

XS2-Finger	V1.2.0	
XS2-Card	V1.2.0	
BS3	V1.1.0	

Visual barcode QR code sensor가 , visual camera QR code
가
BS2_EnableBarcodeLicense .

6. motionSensitivity

[+ 2.9.1] Visual barcode , .

0	BS2_MOTION_SENSITIVITY_LOW	
1	BS2_MOTION_SENSITIVITY_NORMAL	
2	BS2_MOTION_SENSITIVITY_HIGH	

7. *visualCameraScanTimeout*

[+ 2.9.1] Visual camera scan

10 , 3~20 가

10	BS2_VISUAL_BARCODE_TIMEOUT_DEFAULT	
3	BS2_VISUAL_BARCODE_TIMEOUT_MIN	
20	BS2_VISUAL_BARCODE_TIMEOUT_MAX	

8. *reserved*

BS2InputConfigEx

```
typedef struct {
    uint8_t    numInputs;
    uint8_t    numSupervised;
    uint8_t    reserved[18];

    struct {
        uint8_t    portIndex;
        uint8_t    switchType;
        uint16_t   duration;

        uint8_t    reserved;
        uint8_t    supervisedResistor;
        uint8_t    reserved1[16];

        uint8_t    reserved2[26];
    } inputs[BS2_MAX_INPUT_NUM_EX];

    uint8_t    reserved2[200];
} BS2InputConfigEx;
```

1. *numInputs*

Input

2. *numSupervised*

supervised input

3. *reserved*

4. *portIndex*

Input

5. *switchType*

Input

0	Normally Open
1	Normally Closed

6. *duration*

Input (ms)

7. *reserved*

8. *supervisedResistor*

Supervised input, (unsupervised)

0	1K
1	2.2K
2	4.7K
3	10K
254	Unsupervised()

9. *reserved1*

10. *reserved2*

11. *reserved2*

BS2RelayActionConfig

```
typedef struct {
    uint32_t    deviceID;           ///< 4 bytes
    uint8_t     reserved[16];      ///< 16 bytes

    struct {
        uint8_t    port;           ///< 1 byte (relay port)
        uint8_t     reserved0;     ///< 1 byte
        uint8_t     disconnEnabled; ///< 1 byte (RS485
disconnection)
        uint8_t     reserved[9];   ///< 9 bytes

        struct {
            uint8_t    port;           ///< 1 byte (input port)
```

```

        uint8_t type;           ///< 1 byte (linkage/latching/release)
        uint8_t mask;          ///< 1 byte (alarm/fault)
        uint8_t reserved[9];   ///< 9 bytes
    } input[BS2_MAX_RELAY_ACTION_INPUT];    ///< 192 bytes
} relay[BS2_MAX_RELAY_ACTION];             ///< 816 bytes

    uint8_t reserved2[152];    ///< 152 bytes
} BS2RelayActionConfig;
    
```

1. *deviceId*

2. *reserved*

3. *relay*

Relay

4. *port*

Relay port

5. *reserved0*

6. *disconnEnabled*

true, RS485

7. *reserved*

8. *input*

relay port가 input port

9. *port*

Input port

10. *type*

input input
Linkage mask alarm 가

type		
NONE	0	
LINKAGE	1	input relay
LATCHING	2	
RELEASE	3	

11. *mask*

Input mask

type		
NONE	0	
ALARM	1	

type		
FAULT	2	/

12. *reserved*

13. *reserved2*

BS2VoipConfigExt

```
typedef struct {
    BS2_USER_ID phoneNumber;
    char description[48 * 3];

    uint8_t reserved[32];
} BS2ExtensionNumber;

typedef struct {
    BS2_B00L enabled;
    BS2_B00L useOutboundProxy;
    uint16_t registrationDuration;

    BS2_URL address;
    BS2_PORT port;

    struct {
        uint8_t speaker;    // 0 ~ 100
        uint8_t mic;        // 0 ~ 100
    } volume;              ///< 2 bytes

    BS2_USER_ID id;
    BS2_USER_ID password;
    BS2_USER_ID authorizationCode;

    struct {
        BS2_URL address;
        BS2_PORT port;
        uint8_t reserved[2];
    } outboundProxy;

    uint8_t exitButton;    ///< *, #, 0~9
    uint8_t reserved1;
    uint8_t numPhoneBook;
    BS2_B00L showExtensionNumber;

    BS2ExtensionNumber phonebook[128];

    uint8_t reserved2[32];    ///< 32 bytes (reserved)
```

```
} BS2VoipConfigExt;
```

1. *phoneNumber*

2. *description*

3. *reserved*

4. *enabled*

VoIP extension

5. *useOutboundProxy*

Outbound

6. *registrationDuration*

SIP

, 60~600

7. *address*

SIP (BioStar) IP

8. *port*

SIP 5060

9. *speaker*

0 100 50

10. *mic*

0 100 50

11. *id*

SIP ID

12. *password*

SIP

13. *authorizationCode*

SIP

14. *outboundProxy*

Outbound

15. *address*

Outbound IP

16. *port*

Outbound

17. *reserved*

18. *exitButton*

*	'*' ASCII code 42
#	'#' ASCII code 35
0~9	'0'~'9' ASCII code (48~57)

19. *reserved1*

20. *numPhoneBook*

21. *showExtensionNumber*

22. *phonebook*
128

23. *reserved2*

BS2RtspConfig

```
typedef struct {
    BS2_USER_ID id;
    BS2_USER_ID password;

    BS2_URL address;

    BS2_PORT port;
    BS2_BOOL enabled;
    uint8_t reserved;

    uint8_t reserved2[32];
} BS2RtspConfig;
```

1. *id*

RTSP

2. *password*

RTSP

3. *address*

RTSP

4. *port*

RTSP

554

5. *enabled*

RTSP

6. reserved

7. reserved2

BS2License

```

typedef struct {
    uint8_t      index;
    uint8_t      hasCapability;
    uint8_t      enable;
    uint8_t      reserved;
    BS2_LICENSE_TYPE licenseType;
    BS2_LICENSE_SUB_TYPE licenseSubType;
    uint32_t     enableTime;
    uint32_t     expiredTime;
    uint32_t     issueNumber;
    uint8_t      name[BS2_USER_ID_SIZE];
} BS2License;

```

1. index

2. hasCapability

가

1

3. enable

가

4. reserved

5. licenseType

0x0000	None
0x0001	Visual QR

6. licenseSubType

licenseType

0	None
1	Visual QR (CodeCorp)

7. enableTime

, POSIX time

8. *expiredTime*

, 0

9. *issueNumber*

10. *name*

BS2LicenseConfig

```
typedef struct {
    uint8_t    version;
    uint8_t    numOfLicense;
    uint8_t    reserved[2];
    BS2License license[BS2_MAX_LICENSE_COUNT];
    uint8_t    reserved1[16];
} BS2LicenseConfig;
```

1. *version*

2. *numOfLicense*

3. *reserved*

4. *license*

, 16

5. *reserved1*

BS2BarcodeConfig

```
typedef struct {
    uint8_t useBarcode;
    uint8_t scanTimeout;
    uint8_t bypassData;
    uint8_t treatAsCSN;

    uint8_t useVisualBarcode;
    uint8_t motionSensitivity;
    uint8_t visualCameraScanTimeout;
    uint8_t reserved[9];
} BS2BarcodeConfig;
```

1. useBarcode

XS2-QR Barcode flag

2. scanTimeout

Barcode scan 4, 4~10 가

4	BS2_BARCODE_TIMEOUT_DEFAULT	
4	BS2_BARCODE_TIMEOUT_MIN	
10	BS2_BARCODE_TIMEOUT_MAX	

3. bypassData

[+2.8.2] barcode, barcode, 32 byte (BS2CSNCard data), BS2_SetBarcodeScanListener, 512 byte barcode

4. treatAsCSN

[+2.8.2] Barcode CSN, XS2-QR 1.1.3, false, ASCII code 32 126, (BS2_WriteQRCode), true, barcode CSN, 가 barcode, card type, CSN barcode

5. useVisualBarcode

[+ 2.9.1] Visual barcode flag

XS2-Finger	V1.2.0	
XS2-Card	V1.2.0	
BS3	V1.1.0	

Visual barcode QR code sensor가, visual camera QR code 가, BS2_EnableBarcodeLicense

6. motionSensitivity

[+ 2.9.1] Visual barcode

0	BS2_MOTION_SENSITIVITY_LOW	
1	BS2_MOTION_SENSITIVITY_NORMAL	
2	BS2_MOTION_SENSITIVITY_HIGH	

7. visualCameraScanTimeout

[+ 2.9.1] Visual camera scan 가 . .
 10 , 3~20

10	BS2_VISUAL_BARCODE_TIMEOUT_DEFAULT	
3	BS2_VISUAL_BARCODE_TIMEOUT_MIN	
20	BS2_VISUAL_BARCODE_TIMEOUT_MAX	

8. reserved

BS2OsdpStandardConfig

```
typedef struct {
    uint32_t          baudRate;           ///< 4 bytes
    uint8_t          channelIndex;       ///< 1 byte
    uint8_t          useRegistance;      ///< 1 byte
    uint8_t          numOfDevices;      ///< 1 byte
    BS2 OSDP_CHANNEL_TYPE channelType;  ///< 1 byte
    BS2OsdpStandardDevice
slaveDevices[BS2_RS485_MAX_SLAVES_PER_CHANNEL];  ///< 28 * 32 = 896 bytes
    uint8_t          reserved[4];       ///< 4 bytes
} BS2OsdpStandardChannel;  ///< 908 bytes

typedef struct {
    uint8_t          mode[BS2_RS485_MAX_CHANNELS_EX];  ///< 8 byte
    uint16_t         numOfChannels;  ///< 2 byte
    uint8_t          reserved[2];    ///< 2 bytes
    (packing)
    uint8_t          reserved1[32];  ///< 32 bytes
    (reserved)
    BS2OsdpStandardChannel channels[BS2_RS485_MAX_CHANNELS_EX];  ///<
908 * 8 bytes = 7264 bytes
} BS2OsdpStandardConfig;  ///< 7308 bytes
```

1. baudRate
 OSDP

9600
19200
38400
57600

115200

2. *channelIndex*

OSDP 가 RS485

3. *useRegistance*

flag

4. *numOfDevices*

5. *channelType*

RS485

가

CoreStation40

,

가

0~4

5

,

OSDP 가

가

, Suprema

, OSDP

0

channelType 1

Suprema

가

.

OSDP

가

Suprema

OSDP

가

,

OSDP

, channelType

2

Suprema

가

CoreStation40

Suprema

, OSDP

OSDP

가

가

2

channelType 3

가

0	Normal
1	Suprema
2	OSDP
3	OSDP FULL

6. *slaveDevices*

7. *reserved*

8. *mode*

RS485

flag

Osdp standard config

CoreStation40

2023/1/12

master

0	
1	Master
2	Slave
3	Standalone ()

9. *numOfChannels*

. CoreStation40

5

10. reserved

11. reserved1

12. channels

OSDP

8

가

, CoreStation40 5

가

0~4

BS2OsdpStandardActionConfig

```
typedef struct{
    BS2_B00L use;          ///< 1 byte
    uint8_t readerNumber;  ///< 1 byte
    uint8_t ledNumber;     ///< 1 byte

    BS2_OSDP_STANDARD_LED_COMMAND tempCommand;  ///< 1 byte
    uint8_t tempOnTime;          ///< 1 byte
    uint8_t tempOffTime;        ///< 1 byte
    BS2_OSDP_STANDARD_COLOR tempOnColor;        ///< 1 byte
    BS2_OSDP_STANDARD_COLOR tempOffColor;       ///< 1 byte
    uint16_t tempRunTime;        ///< 2 bytes

    BS2_OSDP_STANDARD_LED_COMMAND permCommand;  ///< 1 byte
    uint8_t permOnTime;          ///< 1 byte
    uint8_t permOffTime;        ///< 1 byte
    BS2_OSDP_STANDARD_COLOR permOnColor;        ///< 1 byte
    BS2_OSDP_STANDARD_COLOR permOffColor;       ///< 1 byte

    uint8_t reserved;           ///< 1 byte
} BS2osdpStandardLedAction;  ///< 16 bytes

typedef struct {
    BS2_B00L use;          ///< 1 byte
    uint8_t readerNumber;  ///< 1 byte
    BS2_OSDP_STANDARD_TONE tone;          ///< 1 byte
    uint8_t onTime;        ///< 1 byte
    uint8_t offTime;       ///< 1 byte
    uint8_t numOfCycle;    ///< 1 byte
    uint8_t reserved[2];   ///< 2 bytes
} BS2osdpStandardBuzzerAction;  ///< 8 bytes

typedef struct {
    BS2_OSDP_STANDARD_ACTION_TYPE actionType;  ///< 1 byte
    uint8_t reserved[3];                       ///< 3 bytes
    BS2osdpStandardLedAction led[2];          ///< 16 x 2 = 32 bytes
    BS2osdpStandardBuzzerAction buzzer;       ///< 8 bytes
}
```

```

} BS20sdpStandardAction;           ///< 44 bytes

typedef struct
{
    uint8_t          version;        ///< 1 byte
    uint8_t          reserved[3];    ///< 3 bytes
    BS20sdpStandardAction actions[BS2_OSDP_STANDARD_ACTION_MAX_COUNT];
    ///< 44 x 32 = 1408
} BS20sdpStandardActionConfig;     ///< 1412 bytes
    
```

1. use

LED action

2. readerNumber

OSDP

3. ledNumber

OSDP 가 LED

4. tempCommand

Temporary command

0	No Operation
1	Cancel
2	Set

5. tempOnTime

Temporary command LED가 , 100ms
 2 LED on 20

6. tempOffTime

Temporary command LED가 , 100ms
 1 LED off 10

7. tempOnColor

Temporary command On LED

0	BLACK
1	RED
2	GREEN
3	AMBER
4	BLUE
5	MAGENTA
6	CYAN
7	WHITE

8. tempOffColor

Temporary command Off LED

tone On/Off . 0 .

22. reserved

23. *actionType*
action

0	None
1	Success
2	Fail
3	Wait input

24. reserved

25. *led*
OSDP LED

26. *buzzer*
OSDP buzzer

27. *version*
Action configuration . 0 .

28. reserved

29. *actions*
OSDP LED/buzzer , 32 .

BS2CustomMifareCard

```
typedef struct {
    uint8_t primaryKey[6];
    uint8_t reserved1[2];
    uint8_t secondaryKey[6];
    uint8_t reserved2[2];
    uint16_t startBlockIndex;
    uint8_t dataSize;
    uint8_t skipBytes;
    uint8_t reserved[4];
} BS2CustomMifareCard;
```

1. *primaryKey*
Mifare card

2. *reserved1*

3. *secondaryKey*

Mifare card

4. *reserved2*5. *startBlockIndex*

Mifare data storage start block index

6. *dataSize*

가 byte

7. *skipBytes*

가

0

byte

8. *reserved***BS2CustomDesFireCard**

```

typedef struct {
    uint8_t primaryKey[16];
    uint8_t secondaryKey[16];
    uint8_t appID[3];
    uint8_t fileID;
    uint8_t encryptionType;           // 0: DES/3DES, 1: AES
    uint8_t operationMode;           // 0: legacy(use picc master
key), 1: new mode(use app master, file read, file write key)
    uint8_t dataSize;
    uint8_t skipBytes;
    uint8_t reserved[4];
    BS2DesFireAppLevelKey desfireAppKey;   ///<52 bytes
} BS2CustomDesFireCard;               ///<96 Bytes

```

1. *primaryKey*

DesFire card

. ()

2. *secondaryKey*

DesFire card

. ()

3. *appID*

DESFire

4. *fileID*

DESFire

5. *encryptionType*

0	DES/3DES
1	AES

6. *operationMode*

0	(PICC master key)
1	(App master key)

7. *dataSize*

가 byte

8. *skipBytes*

가

0 , byte

9. *reserved*

10. *desfireAppKey*

DesFire card

()

BS2CustomCardConfig

```
typedef struct {
    BS2_CARD_DATA_TYPE dataType;
    BS2_BOOL useSecondaryKey;
    uint8_t reserved1[2];

    BS2CustomMifareCard mifare;
    BS2CustomDesFireCard desfire;
    uint8_t reserved2[24];
    uint8_t reserved3[96];

    BS2_CARD_BYTE_ORDER smartCardByteOrder;
    uint8_t reserved4[3];
    BS2_UID formatID;
    uint8_t reserved5[8];
} BS2CustomCardConfig;
```

1. *dataType*

Card

0	
1	

2	UTF16
3	BCD

2. *useSecondaryKey*

flag

3. *reserved1*

4. *mifare*

Mifare custom card

5. *desfire*

DESFire custom card

6. *reserved2*

7. *reserved3*

8. *smartCardByteOrder*

MSB LSB

0	MSB
1	LSB

9. *reserved4*

10. *formatID*

BioStar

card configuration

가

11. *reserved5*

1)

, Maximum Transmission Unit

2)

, Most Significant Bit

3)

, Least Significant Bit

From:
<http://kb.supremainc.com/bs2sdk/> - **BioStar 2 Device SDK**

Permanent link:
http://kb.supremainc.com/bs2sdk./doku.php?id=ko:configuration_api&rev=1691718721

Last update: **2023/08/11 10:52**

