Z240 ZigBee High-Level Library and Sample Application — End Device Guide

The sample application is configured at compile time to operate as an End Device.

1. はじめに

本書は、Z240を使用したシステムを容易に構築するために開発した **ZigBee High Level Library** と、ESP32 DevKit V1 + Z240-MP4 を使用したEndDeviceサンプルプログラムについて説明します。

本ライブラリは、ZigBeeモジュールとのUART通信やコマンドフレームの生成・解析などの低レベル処理をライブラリ内部で行うことにより、アプリケーションプログラムからHigh Level APIを呼び出してZigBee通信を利用できることを目的としています。

本プロジェクトでは、Coordinatorとの **Cluster `0xFC08` を使用した透過通信**に加え、ZigBee Home Automation Profileを利用した次のデバイス動作をサポートしています。

- 温・湿度センサー
- 漏水／冠水センサー（IAS Zone）
- EndDevice間の透過通信
- Coordinatorへのノード登録とShortAddress問い合わせ

本書を参照される前に、まずCoordinator側の説明書である、 Z240 ZigBee High-Level Library and Sample Application — Coordinator Guide をご参照ください。

1.1 本書の目的

本書は、ZigBee High Level Libraryを使用してEndDeviceを構築するために必要な、ライブラリの導入方法、基本的な使用方法、透過通信、およびHome Automationデバイスとして動作させるための設定方法について説明することを目的とします。

ZigBeeモジュール固有のコマンド生成、応答解析、ネットワーク接続処理などはライブラリ内部で処理します。
そのため、アプリケーション側ではZigBee通信プロトコルの詳細をできるだけ意識せず、HighLevel APIを使用して通信を行います。

1.2 ライブラリの特徴

本プロジェクトでは、用途に応じて `ZigBee.h` の動作モードを切り替えます。

```
// 1. 通常の透過送信
//#define TRANSPARENT

// 2. Cluster 0xFC08 を使用した透過送信
#define TRANSPARENT_ByFC08

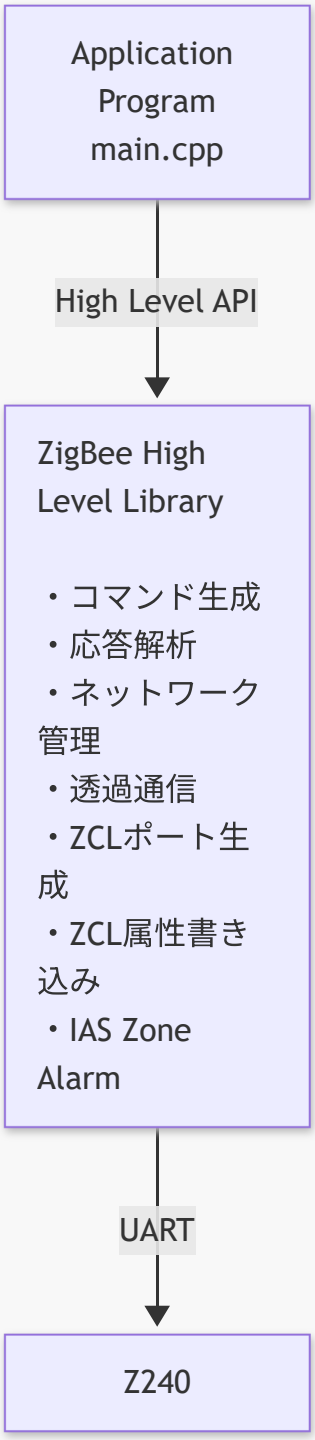
// 3. 温・湿度センサー
//#define THERMOSENSOR
```

```
// 4. 水漏れ・冠水センサー
//#define WATERDETECTOR
```

通常は使用するモードを1つ選択してビルドします。

---

2. ライブラリ概要



---

3. ファイル構成

```
project/
├─ platformio.ini
└─ src/
   ├─ main.cpp
   ├─ HighLayer.c
   ├─ HighLayer.h
   ├─ HL_Coordinator.c
   ├─ zb_def.c
   ├─ ZigBee.h
   ├─ SerialAndPrint.cpp
   ├─ SerialAndPrint.h
   ├─ eeprom_ex.cpp
   └─ eeprom_ex.h
```

ファイル 内容

`main.cpp` EndDeviceサンプル本体、初期化、ネットワーク参加、定期送受信

`HighLayer.c` High Level API、Z240制御、ZCLポート生成、属性書き込み、透過通信

`HighLayer.h` 定数、構造体、High Level API宣言

`HL_Coordinator.c` Coordinator関連処理および受信フレーム処理

`zb_def.c` 透過通信、温湿度、IAS Zone用ZCLポート定義

`ZigBee.h` デバイスタイプ、動作モード、UART等の設定

`SerialAndPrint.cpp/.h` Z240とのUART通信およびデバッグ表示

`eeprom_ex.cpp/.h` EEPROM処理

---

4. 基本的な使い方

4.1 EndDevice設定

本サンプルでは `ZigBee.h` でEndDeviceを指定しています。

```
#define ZIGBEE_DEVICE ENDDEVICE
```

自機番号は次の定義で設定します。

```
#define MY_DEVICE_NO 0x01
```

Coordinator側とEndDevice側で機器番号が重複しないように設定します。

4.2 UART

Z240との通信にはESP32の `Serial2` を使用します。

```
#define ZB_SERIAL_PORT Serial2
#define ZigBee_RxPin   16
#define ZigBee_TxPin   17
#define ZigBee_BaudRate 115200
```

4.3 ネットワーク参加

起動時に `zb_init()` でZ240の状態を確認します。

```
int retcode = zb_init(&zb_ctx, DeviceType, NeedFactoryReset);
```

未接続の場合は、

```
zb_begin(&zb_ctx);
zb_ConnectNetwork(&zb_ctx);
```

を実行してEndDeviceとしてネットワークへ参加します。

`zb_ConnectNetwork()` 内ではCoordinatorへのノード登録要求も行い、登録応答を受信するまで処理します。

---

5. EndDevice

本プロジェクトの `main.cpp` はCoordinator側サンプルと対になっています。

```
EndDevice → Coordinator              Coordinator → EndDevice

55 AA 55 02 +番号+SA+MAC(8)          55 AA 55 12 +可否+番号+SA
    ノード登録

55 AA 55 04 +番号+自SA               55 AA 55 14 +番号+相手SA
    ShortAddress問い合わせ
```

```
55 AA 55 08 +自SA+"1234"            55 AA 55 18 00 00+"67890"
    定期送信と応答
```

通常のサンプルでは、ネットワーク参加後に8秒周期で定期送信を行います。

```
#define SEND_INTERVAL 8000
```

---

6. 透過通信

本プロジェクトでは2種類の透過通信を想定しています。

6.1 通常の透過送信

```
#define TRANSPARENT
```

6.2 Cluster `0xFC08` を使用した透過送信

```
#define TRANSPARENT_ByFC08
```

このモードでは、ネットワーク参加前にCluster `0xFC08` のZCLポートを生成します。

```
zb_Set_TransparentSending_byFC08(&zb_ctx, &Transparent_val);
```

データ送信には次の関数を使用します。

```
zb_sendTPbyFC08(shortAddress, data, len);
```

6.3 EndDevice間通信

`End2End` を定義すると、送信前にCoordinatorへ相手EndDeviceのShortAddressを問い合わせ、そのShortAddress宛に直接透過送信します。

```
//#define End2End
```

---

7. HA（ホームオートメーション）サンプル

本プロジェクトでは、透過通信だけでなく、ZigBee Home Automation Profileを利用したデバイスとして動作させることができます。
以降に記載した例で、HAセンサーとして動作・確認をするためには、 市販のZigBeeゲートウェイ（ワイヤレスHUB）を用いる必要があります。
スマートフォン表示例で用いたものは、ゲートウェイとしてtuya 製 TYZG1、スマートフォンアプリ SmartLife です。

また、漏水／冠水センサーは IAS ALARM の通信（透過送信）を使用しますので、スマートホンアプリで表示する以外に、Coordinator 側では、透過送信(cluster 0xFC08)によりデータとして受信できます。

対応しているサンプルは次の2種類です。

1. 温・湿度センサー
2. 漏水／冠水センサー（IAS Zone）

7.1 温・湿度センサー

7.1.1 動作モード

`ZigBee.h` で次を有効にします。

```
#define THERMOSENSOR
```

透過通信モードを同時に使用しない場合は、該当する `TRANSPARENT` / `TRANSPARENT_ByFC08` の定義を無効にします。

7.1.2 Home Automation Profile

温・湿度センサーでは次のZigBee Clusterを使用します。

用途 Cluster ID

Identify `0x0003` Temperature Measurement `0x0402` Relative Humidity Measurement `0x0405`

ZCLポート生成用データは `zb_def.c` の `ThermoSensor_val` にまとめられています。

```
const struct portConfig ThermoSensor_val = {
    &ZCL_make_TH_Port_val,
    addZclPortAttribute_val,
    sizeof(addZclPortAttribute_val) /
        sizeof(addZclPortAttribute_val[0])
};
```

Profile IDにはHome Automation Profileを使用し、Device IDにはTemperature Sensorを指定しています。

7.1.3 温度属性

Temperature Measurement Clusterは `0x0402` です。

Measured Valueとして属性 `0x0000` を使用します。

```
Cluster ID : 0x0402
Attribute  : 0x0000
Data Type  : INT16
```

7.1.4 湿度属性

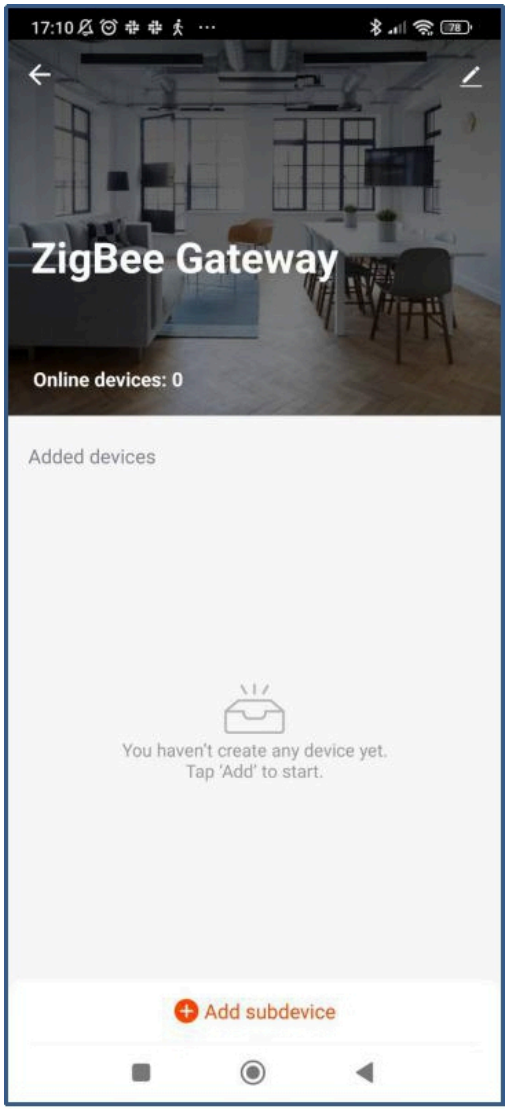
Relative Humidity Measurement Clusterは `0x0405` です。

```
Cluster ID : 0x0405
Attribute  : 0x0000
Data Type  : UINT16
```

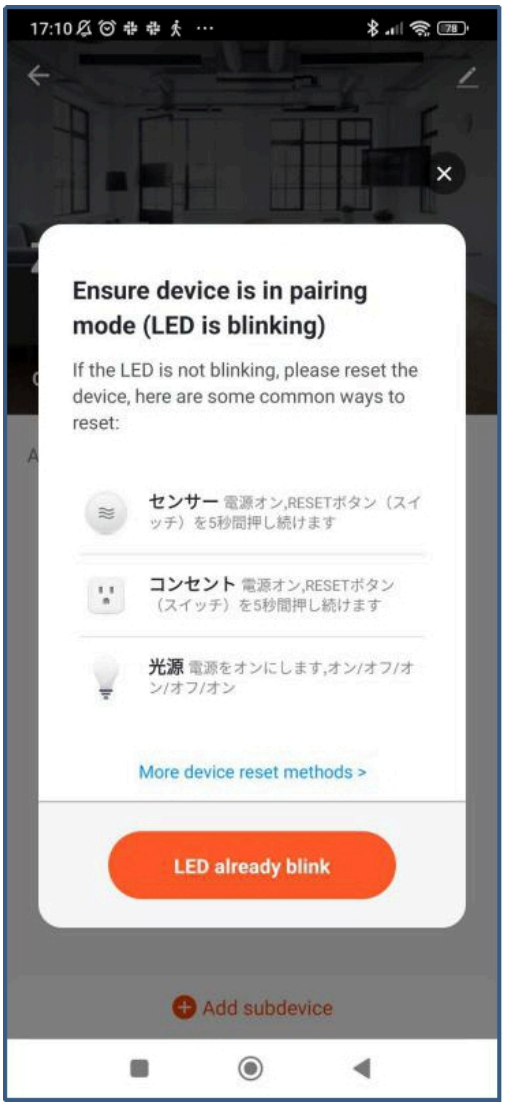
属性値の書き込みにはHigh Level APIの `zb_writeAttr()` を使用します。

```
zb_writeAttr(&zb_ctx, cluster_id, data);
```

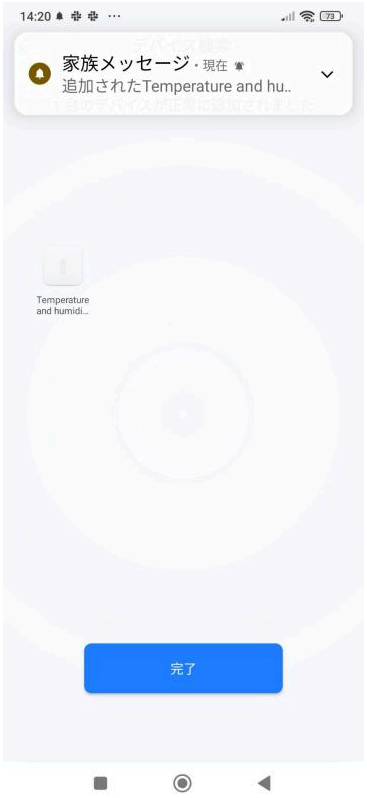
7.2 温・湿度センサーのスマートフォン表示例



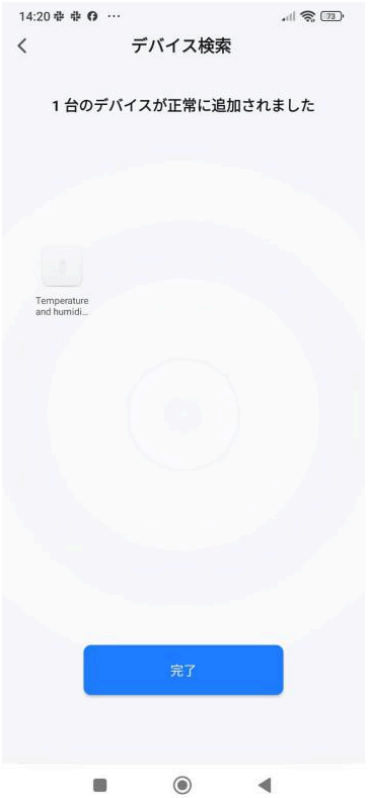
ZigBee Gateway画面



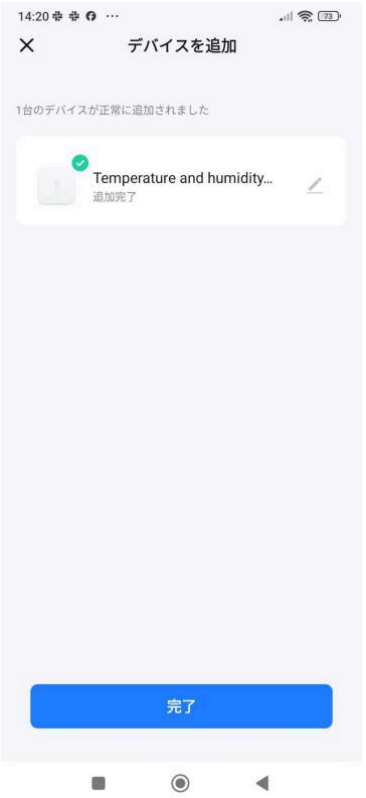
デバイス追加



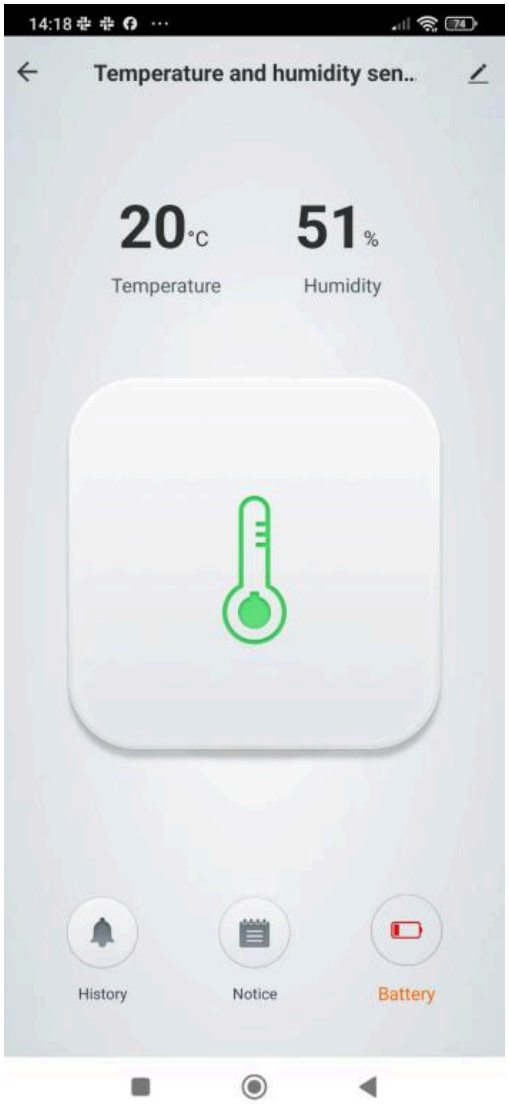
温湿度センサー登録



温湿度センサー登録完了



温湿度センサー登録表示



温湿度表示 20℃ 51%



[温湿度表示 30℃ 57%

温度と湿度の属性値を書き換えることにより、対応するZigBee HUB／スマートフォンアプリ上で温度・湿度センサーとして値を確認できます。

---

8. 漏水／冠水センサー（IAS Alarm）

漏水／冠水センサーでは、ZigBee Home Automation Profileの **IAS Zone Cluster (`0x0500`)** を使用します。

8.1 動作モード

`ZigBee.h` で次を有効にします。

```
#define WATERDETECTOR
```

8.2 Water Detectorのポート設定

Water Detectorでは次のClusterを入力Clusterとして使用します。

用途 Cluster ID

Basic `0x0000` Identify `0x0003` IAS Zone `0x0500`

`zb_def.c` では `WaterDetector_val` として設定値をまとめています。

```
const struct portConfig WaterDetector_val = {
    &ZCL_make_WD_Port_val,
    addZclPortAttribute_WD_val,
    sizeof(addZclPortAttribute_WD_val) /
        sizeof(addZclPortAttribute_WD_val[0])
};
```

8.3 IAS Zone属性

IAS Zone Clusterでは主に次の属性を定義しています。

```
Attribute ID 内容
```

```
    `0x0000` ZoneState
    `0x0001` ZoneType
    `0x0002` ZoneStatus
    `0x0010` IAS_CIE_Address
    `0x0011` ZoneID
    `0xFFFD` Cluster Revision
```

Water Detectorとして使用する場合、ZoneTypeにはWater Sensorを示す値を設定します。

`HighLayer.c` ではWater Detector用の初期属性設定コマンドも用意されています。

```
const byte WaterDetector[] = {
    0x55,0x0E,0x00,0x43,0x02,
    0x00,0x00,0x00,0x05,
    0x00,0x00,0x01,0x00,
    0x2A,0x00,0x6F
};
```

8.4 Water DetectorのAlarm送信

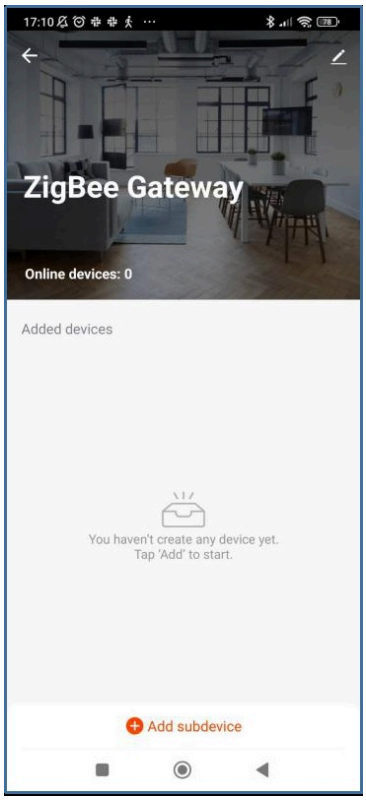
漏水／冠水状態はIAS ZoneのZone Statusで通知します。

代表的なbitは次のとおりです。

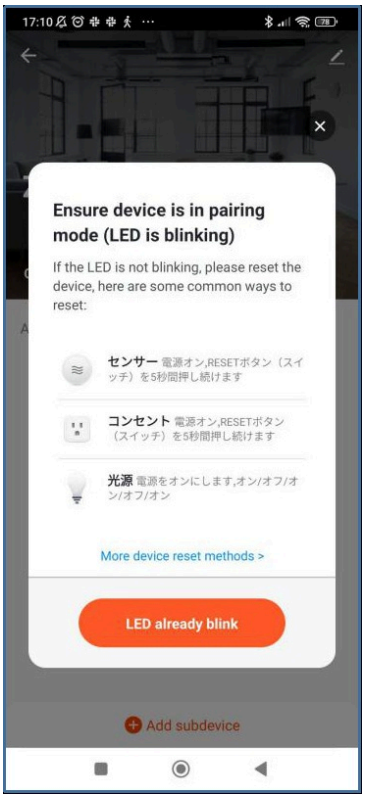
```
#define IAS_Zone_Alarm1 0b0000000000000001
#define IAS_Zone_Alarm2 0b0000000000000010
#define IAS_Zone_Temper 0b0000000000000100
```

サンプルプログラムを実行した Coordinator に接続した場合(Cluster 0xFC08 を使用した透過通信)は、EndDevice から送信されたデータを透過送信の受信データとして表示します。

8.5 漏水／冠水センサーのスマートフォン表示例



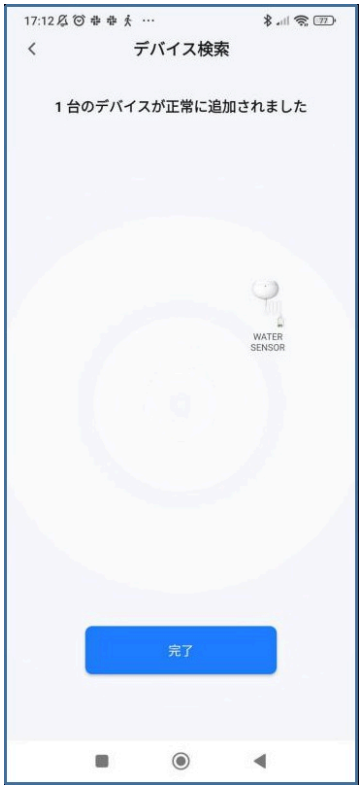
デバイス検索



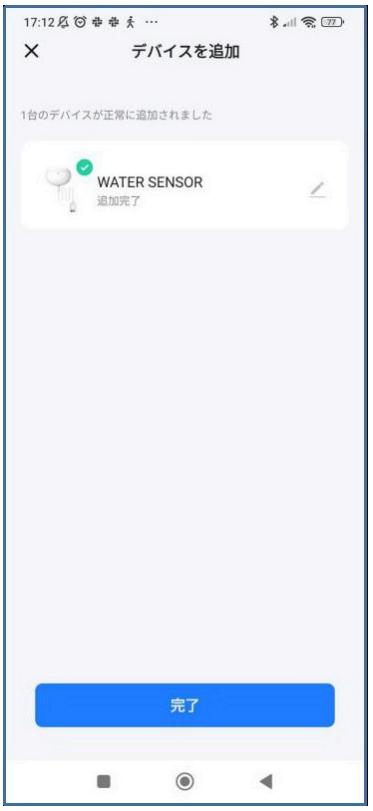
デバイス追加完了



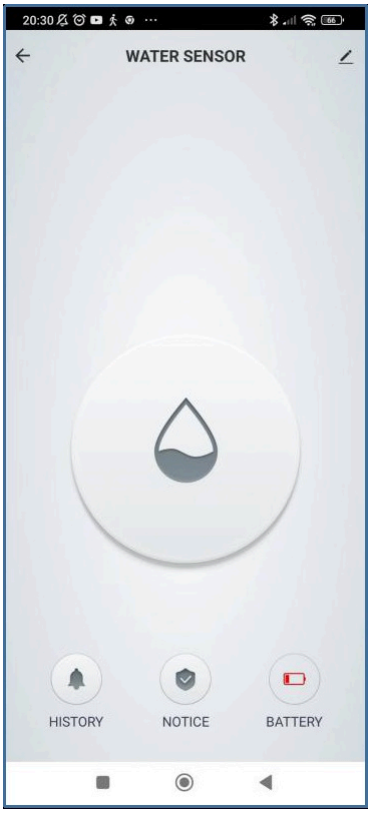
漏水・冠水センサー登録表示



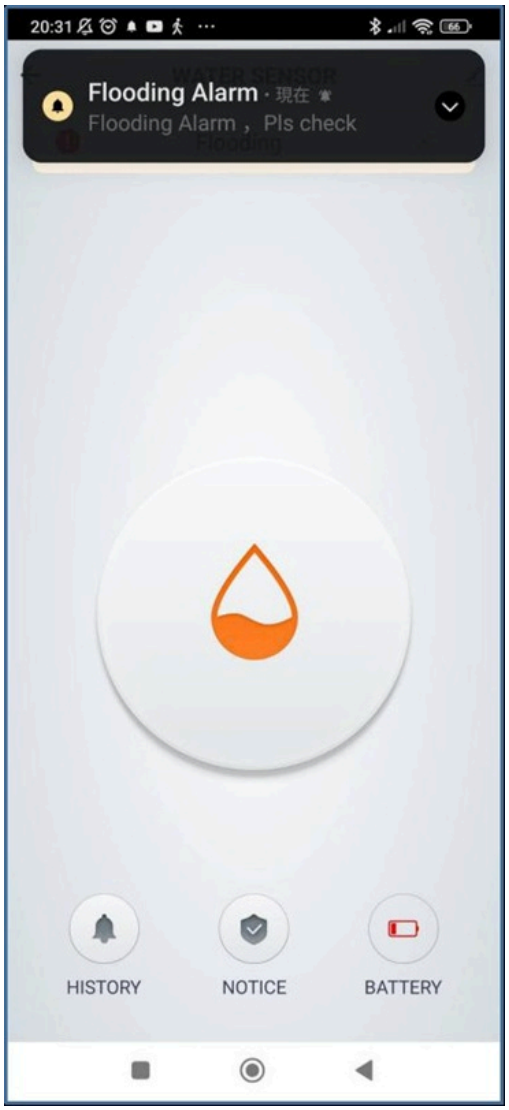
Water Sensor 通常状態



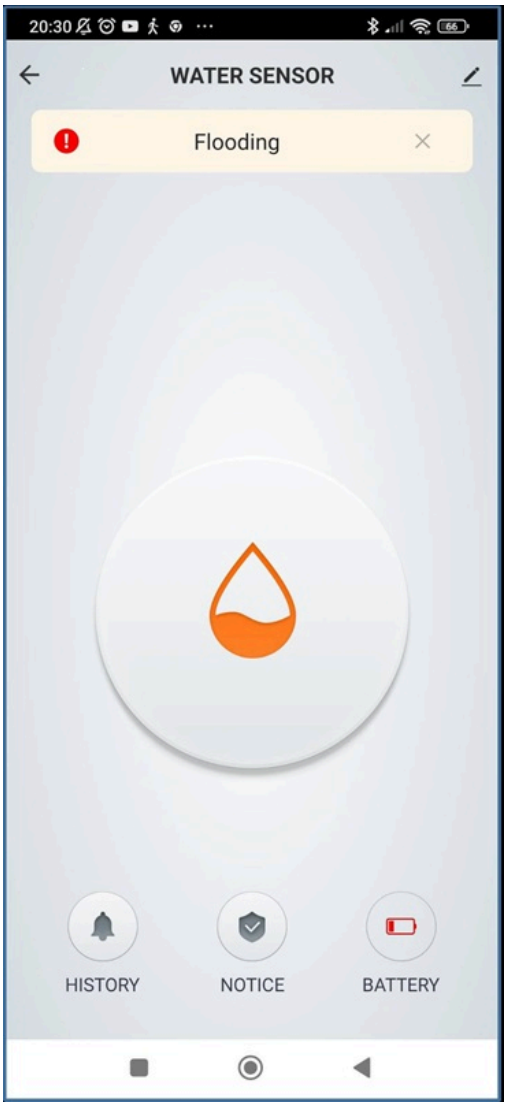
Flooding Alarm 通知



Flooding 状態



Sensor Lose 通知



Sensor Lose 状態

この表示例は、Water DetectorをIAS ZoneデバイスとしてZigBee HUBへ登録した場合のスマートフォンアプリ上の動作イメージです。

---

9. APIリファレンス

9.1 初期化・ネットワーク

zb_init()

```
int zb_init(zb_conn_t *c, int zbdev, bool FactoryReset);
```

Z240の状態を確認し、ネットワーク状態を取得します。

zb_begin()

```
bool zb_begin(zb_conn_t *c);
```

Z240のノードタイプなどを設定します。

zb_ConnectNetwork()

```
bool zb_ConnectNetwork(zb_conn_t *c);
```

EndDeviceではZigBeeネットワークへの参加を行います。

9.2 ZCL

zb_makePort()

```
bool zb_makePort(zb_conn_t *c, const struct portConfig *value);
```

ZCLポートを生成し、必要なClusterとAttributeを設定します。

zb_writeAttr()

```
bool zb_writeAttr(
    zb_conn_t *c,
    uint16_t cluster_id,
```

```
    int16_t data
);
```

温度、湿度などのZCL属性値を書き込みます。

9.3 透過通信

zb_write()

```
int zb_write(
    zb_conn_t *c,
    const uint8_t *data,
    size_t len
);
```

透過通信データを送信します。

zb_read()

```
int zb_read(zb_conn_t *c, uint16_t WaitTime);
```

透過通信データを受信します。

zb_Set_TransparentSending_byFC08()

```
void zb_Set_TransparentSending_byFC08(
    zb_conn_t *c,
    const struct portConfig *value
);
```

Cluster `0xFC08` の透過通信用ZCLポートを生成します。

zb_sendTPbyFC08()

```
void zb_sendTPbyFC08(
    uint16_t ShortAddr,
    const uint8_t *data,
    size_t len
);
```

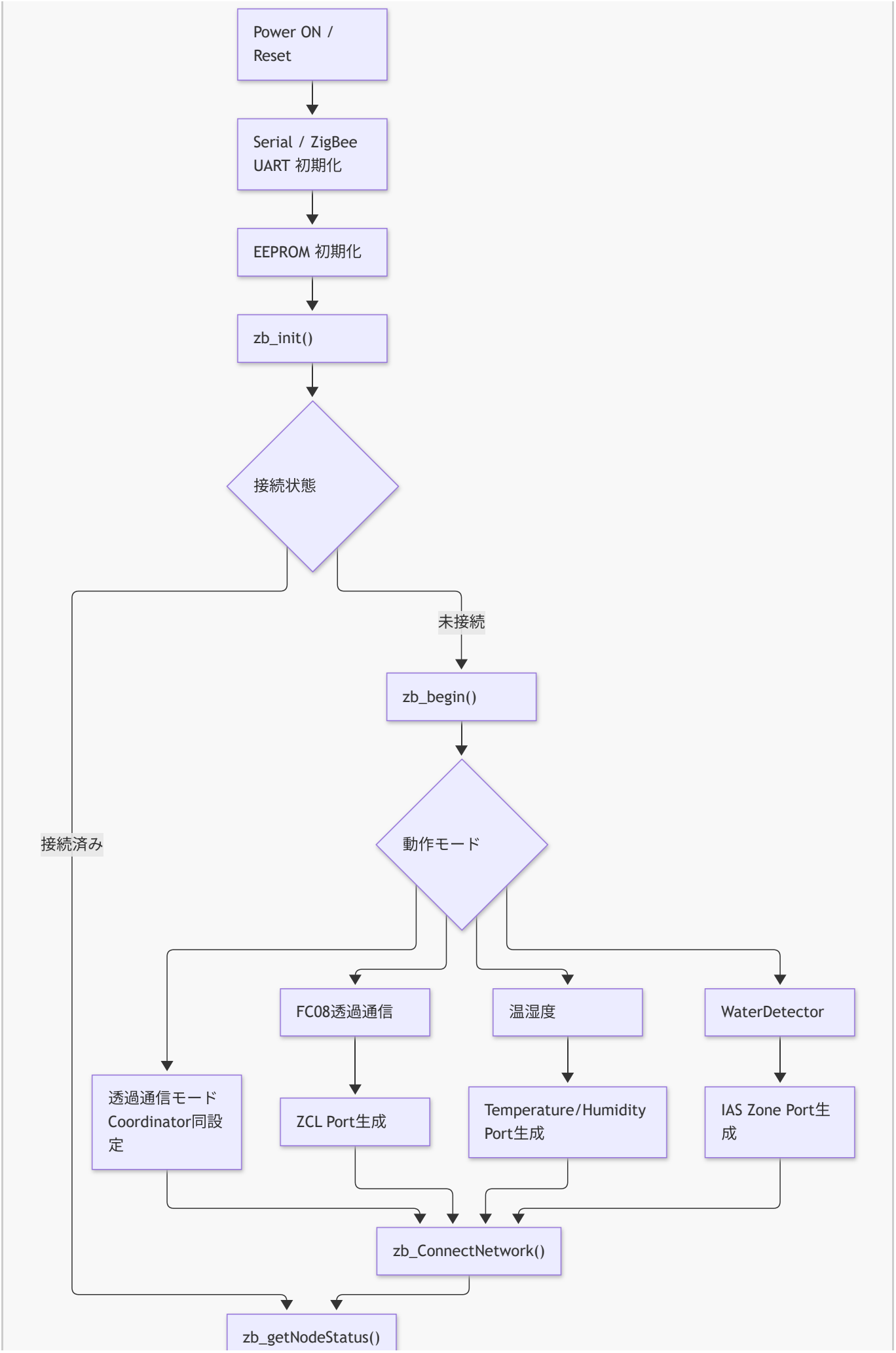
Cluster `0xFC08` を使用して指定ShortAddressへデータを送信します。

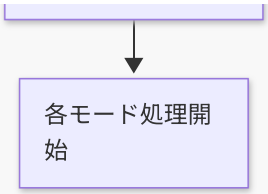
---

10. 動作モードまとめ

モード `ZigBee.h` の定義	主な用途
通常透過通信 `TRANSPARENT`	ZigBee透過通信
FC08透過通信 `TRANSPARENT_ByFC08`	Cluster `0xFC08` による透過通信
EndDevice間通信 `End2End`	CoordinatorにShortAddressを問い合わせてEndDeviceへ直接送信
温・湿度センサー `THERMOSENSOR`	HA Temperature / Humidity Sensor
漏水／冠水センサー `WATERDETECTOR`	HA IAS Zone Water Detector

11. 起動処理概要


Power ON / Reset
Serial / ZigBee UART 初期化
EEPROM 初期化
zb_init()
接続状態
接続済み
未接続
zb_begin()
動作モード
透過通信モード Coordinator同設定
FC08透過通信
温湿度
WaterDetector
ZCL Port生成
Temperature/Humidity Port生成
IAS Zone Port生成
zb_ConnectNetwork()
zb_getNodeStatus()



12. HA（ホームオートメーション）サンプル の 設定内容

ZigBee.hに以下の定義を記載

```
// profile
#define PROFILE_ID_HOME_AUTOMATION 0x0104

// device_id
#define DEVICE_ID_TEMP_SENSOR  0x0302
#define DEVICE_ID_IAS_ZONE     0x0402

// cluster
#define CLUSTER_ID_BASIC       0x0000
#define CLUSTER_ID_IDENTIFY    0x0003
#define CLUSTER_ID_TEMP_MEAS   0x0402
#define CLUSTER_ID_HUMID_MEAS  0x0405
#define CLUSTER_ID_IAS_ZONE    0x0500
```

12.1 温・湿度センサー

① 温・湿度センサー のポート設定 （ZCL コマンドコード：0x40）

アプリケーションのエンドポイントとなるポートを作成します。ポート番号は1～240 の範囲で未設定の番号を指定します。以下はポート1 を作成するコマンドの例です。後で追加する属性値の総数を予め調べておく必要があります。

Sample.cpp 内の記述

```
#ifdef THERMOSENSOR
      PrintF("\r\n  -- (2) 温・湿度センサーのZCLポート設定をします ...\r\n");
      zb_makePort(&zb_ctx, &ThermoSensor_val);
#endif
```

Zb_def.c での記述

```
const struct portConfig ThermoSensor_val = {
    &ZCL_make_TH_Port_val,
    addZclPortAttribute_val,
    sizeof(addZclPortAttribute_val) / sizeof(addZclPortAttribute_val[0])
};


// ZCLポート生成用クラスタ定義
// 0x40 (related)
const uint16_t cluster_attrs_in_data[] = {
                    CLUSTER_ID_IDENTIFY,
                    CLUSTER_ID_TEMP_MEAS,
                    CLUSTER_ID_HUMID_MEAS
                };
// ZCLポート生成用データ
// 0x40 (related)
static const ZCLmakeport ZCL_make_TH_Port_val={
        sizeof(addZclPortAttribute_val)/sizeof(addZclPortAttribute_val[0]),
//const uint8_t attr_cnt;
        1,                                      //const uint8_t port_number;
        PROFILE_ID_HOME_AUTOMATION,     //const uint16_t profile_id;
        DEVICE_ID_TEMP_SENSOR,            //const uint16_t device_id;
        sizeof(cluster_attrs_in_data)/sizeof(cluster_attrs_in_data[0]),
                                                //const size_t size_in;
        cluster_attrs_in_data,                  //const uint16_t
*cluster_attrs_in;
        0                                       //const size_t size_out;
};
```

作成されるポート作成 HEXコマンド例

```
55 11 00 40 0A 01 04 01 02 03 03 03 00 02 04 05 04 00 48
```

HEX コマンドバイト列の意味

- フレームヘッダ： 0x55 (固定)
- フレーム長： 0x11 (= 17 bytes)
- コマンド種別： 0x00（= ローカル設定）
- コマンドコード： 0x40（= Create ZCL Port）
- 属性値の総数： 0x0A（= 10）
- 作成するポート番号： 0x01（= ポート1）
- プロファイルID： 0x0104（= Home Automation）
- デバイスID： 0x0302（= Temperature Sensor）
- 入力クラスターID 数： 0x03（= 3）
- 入力クラスターID：
- 0x0003（= Identify）
- 0x0402（= Temperature Measurement）

- 0x0405（= Relative Humidity Measurement）
- 出力クラスターID 数： 0x00（= 0）
- XOR チェックサム： 0x48

② 設定したポートに属性追加 （ZCL コマンドコード：0x41）

```
// ZCLポート属性設定データ
// 0x41 (related)
static const struct addZclPortAttribute addZclPortAttribute_val[] = {
    {CLUSTER_ID_IDENTIFY,  0x0000, DATAID_UINT16, 0x03, 0},   // identify
    {CLUSTER_ID_IDENTIFY,  0xFFFD, DATAID_UINT16, 0x01, 0},   //
    {CLUSTER_ID_TEMP_MEAS, 0x0000, DATAID_INT16, 0x05, 0},   // Temperature
Measurement Cluster
    {CLUSTER_ID_TEMP_MEAS, 0x0001, DATAID_INT16, 0x01, 0},   //
    {CLUSTER_ID_TEMP_MEAS, 0x0002, DATAID_INT16, 0x01, 0},   //
    {CLUSTER_ID_TEMP_MEAS, 0xFFFD, DATAID_UINT16, 0x01, 0},   //
    {CLUSTER_ID_HUMID_MEAS, 0x0000, DATAID_UINT16, 0x05, 0},   // Humidity
Measurement Cluster
    {CLUSTER_ID_HUMID_MEAS, 0x0001, DATAID_UINT16, 0x01, 0},   //
    {CLUSTER_ID_HUMID_MEAS, 0x0002, DATAID_UINT16, 0x01, 0},   //
    {CLUSTER_ID_HUMID_MEAS, 0xFFFD, DATAID_UINT16, 0x01, 0}    //
};
```

作成される属性追加HEXコマンドの例

```
55 0C 00 41 02 04 00 00 00 00 29 05 00 6B              // Temperature Measurement
Cluster
```

HEX コマンドバイト列の意味

- フレームヘッダ： 0x55 (固定)
- フレーム長： 0x0C (= 12 bytes)
- コマンド種別： 0x00（= ローカル設定）
- コマンドコード： 0x41（= Add properties）
- クラスターID： 0x0402（= Temperature Measurement）
- 製造者コード： 0x0000（= default value）
- 属性ID： 0x0000（= MeasuredValue）
- データ型： 0x29（= int16）
- 操作モード： 0x05（= read only / report）
- 属性方向： 0x00（= server）
- XOR チェックサム： 0x6B

④ センサー値の通知

モジュールに対し属性値の書き込みコマンドを実行することで、センサーの値の変更をHUB に通知することができます。

温度と湿度の値を書き込むコマンドの例は以下です。

HEX コマンド：

```
55 0E 00 43 02 00 00 02 04 00 00 00 00 01 02 44
```

温度データ書込みHEX コマンドバイト列の意味

- フレームヘッダ： 0x55 (固定)
- フレーム長： 0x0E (= 14 bytes)
- コマンド種別： 0x00（= ローカル設定）
- コマンドコード： 0x43（= Read and write properties）
- 操作： 0x02（= write attributes and save FLASH）
- ポートインデックス： 0x00（= ポート作成時に割り当てられたインデックス値）
- 属性の方向： 0x00（= server side）
- クラスターID： 0x0402（= Temperature Measurement）
- 製造者コード： 0x0000（= default value）
- 属性ID： 0x0000（= MeasuredValue）
- データ値： 0x0201（= 5.13℃）
- XOR チェックサム： 0x44

HEX コマンド：

```
55 0E 00 43 02 00 00 05 04 00 00 00 00 33 0E 7D
```

温度データ書込みHEX コマンドバイト列の意味

- フレームヘッダ： 0x55 (固定)
- フレーム長： 0x0E (= 14 bytes)
- コマンド種別： 0x00（= ローカル設定）
- コマンドコード： 0x43（= Read and write properties）
- 操作： 0x02（= write attributes and save FLASH）
- ポートインデックス： 0x00（= ポート作成時に割り当てられたインデックス値）
- 属性の方向： 0x00（= server side）
- クラスターID： 0x0405（= Relative Humidity Measurement）
- 製造者コード： 0x0000（= default value）
- 属性ID： 0x0000（= MeasuredValue）
- データ値： 0x0E33（= 36.35%）
- XOR チェックサム： 0x7D

12.2 漏水／冠水センサー（IAS Alarm）

① WaterDetector のポート設定 （ZCL コマンドコード：0x40)

Sample.cpp 内の記述

```
#ifdef WATERDETECTOR
      PrintF("\r\n  -- (3) IAS Alarm で 漏水センサー の ZCLポートを設定しています
...\r\n");
      zb_makePort(&zb_ctx, &WaterDetector_val);
#endifrmoSensor_val);
```

Zb_def.c での記述

```
const struct portConfig WaterDetector_val = {
    &ZCL_make_WD_Port_val,
    addZclPortAttribute_WD_val,
    sizeof(addZclPortAttribute_WD_val) / sizeof(addZclPortAttribute_WD_val[0])
};

//-----------------------------
// ZCLポート生成用クラスタ定義
// 0x40 (related)
const uint16_t cluster_attrs_in_WD_data[] = {
                    CLUSTER_ID_BASIC,
                    CLUSTER_ID_IDENTIFY,
                    CLUSTER_ID_IAS_ZONE
                };

//-----------------------------
// ZCLポート生成用データ
// 0x40 (related)
static const ZCLmakeport ZCL_make_WD_Port_val={
        sizeof(addZclPortAttribute_WD_val)/sizeof(addZclPortAttribute_WD_val[0]),
//const uint8_t attr_cnt;
        1,                                      //const uint8_t port_number;
        PROFILE_ID_HOME_AUTOMATION,             //const uint16_t profile_id;
        DEVICE_ID_IAS_ZONE,                     /const uint16_t device_id;
        sizeof(cluster_attrs_in_WD_data)/sizeof(cluster_attrs_in_WD_data[0]),
                                                //const size_t size_in;
        cluster_attrs_in_WD_data,               //const uint16_t *cluster_attrs_in;
        0                                       //const size_t size_out;

};
```

作成されるポート作成 HEXコマンド例

```
55 11 00 40 11 01 04 01 02 04 03 00 00 03 00 00 05 00 56
```

HEX コマンドバイト列の意味

- フレームヘッダ： 0x55 (固定)

- フレーム長：　　0x11 (= 17 bytes)
- コマンド種別：　0x00（= ローカル設定）
- コマンドコード：0x40（= Create ZCL Port）
- 属性値の総数：　0x11（= 17）
- 作成するポート番号：0x01（= ポート1）
- プロファイルID：0x0104（= Home Automation）
- デバイスID：　　0x0402（= IAS Zone）
- 入力クラスターID 数：0x03（= 3）
- 入力クラスターID：
- 0x0000 (Basic)
- 0x0003（Identify）
- 0x0500 (IAS ZONE)
- 出力クラスターID 数：0x00（= 0）
- XOR チェックサム：0x56

② 設定したポートに属性追加　（ZCL コマンドコード：0x41）

```
// ZCLポート属性設定データ
// 0x41 (related)
static const struct addZclPortAttribute addZclPortAttribute_WD_val[] = {
    {CLUSTER_ID_BASIC,  0x0000, DATAID_UINT8, 0x01, 0},   // Basic
    {CLUSTER_ID_BASIC,  0x0001, DATAID_UINT8, 0x01, 0},
    {CLUSTER_ID_BASIC,  0x0002, DATAID_UINT8, 0x01, 0},
    {CLUSTER_ID_BASIC,  0x0003, DATAID_UINT8, 0x01, 0},
    {CLUSTER_ID_BASIC,  0x0004, DATAID_ENUM8, 0x01, 0},
    {CLUSTER_ID_BASIC,  0x0005, DATAID_ENUM8, 0x01, 0},
    {CLUSTER_ID_BASIC,  0x0006, DATAID_ENUM8, 0x01, 0},
    {CLUSTER_ID_BASIC,  0x0007, DATAID_ENUM8, 0x01, 0},
    {CLUSTER_ID_BASIC,  0xFFFD, DATAID_UINT16, 0x01, 0},
    {CLUSTER_ID_IDENTIFY,  0x0000, DATAID_UINT16, 0x03, 0},   // identify
    {CLUSTER_ID_IDENTIFY,  0xFFFD, DATAID_UINT16, 0x01, 0},   //
    {CLUSTER_ID_IAS_ZONE, 0x0000, DATAID_ENUM8, 0x01, 0},   // IAS Zone Cluster
    {CLUSTER_ID_IAS_ZONE, 0x0001, DATAID_ENUM16, 0x01, 0},   //
    {CLUSTER_ID_IAS_ZONE, 0x0002, DATAID_BIT16, 0x01, 0},   //
    {CLUSTER_ID_IAS_ZONE, 0x0010, DATAID_EUI64, 0x03, 0},   //
    {CLUSTER_ID_IAS_ZONE, 0x0011, DATAID_UINT8, 0x01, 0},   //
    {CLUSTER_ID_IAS_ZONE, 0xFFFD, DATAID_UINT16, 0x01, 0}    //
};
```

作成される属性追加HEXコマンドの例

```
55 0C 00 41 00 05 00 00 00 00 30 01 00 75            // IAS Zone Cluster
```

HEX コマンドバイト列の意味

【詳細】

name	クラスターID	製造者コード	属性ID	データ型	サポート操作	属性方向
ZCL version	0x0000	0x0000	0x0000	0x20 (uint8)	0x01 (read only)	Server
ApplicationVersion	0x0000	0x0000	0x0001	0x20 (uint8)	0x01 (read only)	Server
Software version	0x0000	0x0000	0x0002	0x20 (uint8)	0x01 (read only)	Server
hardware version	0x0000	0x0000	0x0003	0x20 (uint8)	0x01 (read only)	Server
ManufacturerName	0x0000	0x0000	0x0004	0x42 (string)	0x01 (read only)	Server
Product number	0x0000	0x0000	0x0005	0x42 (string)	0x01 (read only)	Server
Production Date	0x0000	0x0000	0x0006	0x42 (string)	0x01 (read only)	Server
Power supply	0x0000	0x0000	0x0007	0x30 (enum8)	0x01 (read only)	Server
property terminator	0x0000	0x0000	0xFFFF	0x21 (uint16)	0x01 (read only)	Server
Identity tag status	0x0003	0x0000	0x0000	0x21 (uint16)	0x03 (r/w)	Server
property terminator	0x0003	0x0000	0xFFFF	0x21 (uint16)	0x01 (read only)	Server
Zone State	0x0500	0x0000	0x0000	0x30 (enum8)	0x01 (read only)	Server
Zone Type	0x0500	0x0000	0x0001	0x31 (enum16)	0x01 (read only)	Server
Zone Status(Alarm)	0x0500	0x0000	0x0002	0x19 (bitmap16)	0x01 (read only)	Server
IAS CIE Address	0x0500	0x0000	0x0010	0xF0 (EUI 64)	0x03 (r/w)	Server
Zone ID	0x0500	0x0000	0x0011	0x20 (uint8)	0x01 (read only)	Server
property terminator	0x0500	0x0000	0xFFFF	0x21 (uint16)	0x01 (read only)	Server

③ IAS Zone を WaterDetector にする属性初期設定 （ZCL コマンドコード：0x43）

HEXコマンドの例 55 0E 00 43 [02] [00] [00] [00 05] [00 00] [01 00] [2A 00] 6F ☐内の説明

- 操作モード = 0x02　　属性の設定とFLASHへの保存
- ポートインデックス = 0x00 ポートの作成時にモジュールに割り当てられたポートインデックス番号
- 属性方向 = 0x00　　サーバ側属性
- クラスタ ID=0x0500　　属性のクラスタID
- 製造者コード = 0x0000　　属性定義の製造者コード
- 属性 ID=0x0001　　属性定義のID番号　（ZoneType）
- 属性のデータ（属性値）= 0x002A　　：Water Detector

④ Water Detector の Alarm 送信　　（ZCLコマンドコード：0x0F）

Alarmデータの送信コマンド

ZCL制御コマンドを使用して、Alarm（状態）を送信するコマンドです。 以下の定義中のCommandParameter内のZone Statusにbitデータをセットし、送信します。

- 0x55
- 0x15 // Data Length
- 0x02,0x0F // ZCL 制御コマンド送信
- 0x00, // Send mode = 0x00, port index 0 sends command, no mode.
- 0x00,0x00, // Target short address
- 0x01, // Destination port
- 0x75, // Frame number
- 0x01, // Command direction=Server->Client
- 0x00,0x05, // Cluster ID
- 0x00,0x00, // Manufacturer code
- 0x00, // Reply mode
- 0x00, // Command ID
- // Command Parameter
- 0x00,0x00, // Zone Status : Alarm (distance16 FromTop) <-- IAS Zone Status bits(Binary)
- 0x00, // Extended Status
- 0xFE, // Defence Zone ID
- 0x01,0x00, // Delay
- //
- 0x82 // Check Code

Zone Status にセットする bit値

アラーム	ビット	名称
IAS_Zone_Alarm1	0b0000000000000001	Alarm 1
IAS_Zone_Alarm2	0b0000000000000010	Alarm 2(SmartLifeでは表示なし)
IAS_Zone_Temper	0b0000000000000100	Tamper

IAS Zoneのセキュリティデバイスにおけるアラーム AlarmZoneStatus属性の値

ビット番号	意味	値	本サンプルでSmartLifeの表示
0	Alarm1	1 - opened or alarmed 0 - closed or not alarmed	○
1	Alarm2	1 - opened or alarmed 0 - closed or not alarmed	×（Alarm2はなし）
2	Tamper	1 - Tampered 0 - Not tampered	○（Alarm2をTamperで）
3	Battery	1 - Low battery 0 - Battery OK	×
4	Supervision Report	1 - Notify 0 - Does not notify	×
5	Restore Report	1 - Notify restore 0 - Does not notify of restore	×
6	Trouble	1 - Trouble/Failure 0 - OK	×
7	AC(mains)	1 - AC/Mains fault 0 - AC/Mains OK	×
8	Test	1 - Sensor is in test mode 0 - Sensor is in operation mode	×
9	Battery Defect	1 - Sensor detects a defective battery 0 - Sensor battery is functioning normally	×

13. ライセンス

本プロジェクトのソースコードおよびドキュメントは、MIT License のもとで公開しています。

個人利用・商用利用を問わず、MIT License の条件に従って、自由に使用・複製・変更・再配布することができます。

詳細については `LICENSE` ファイルを参照してください。

14. サポートについて

本プロジェクトは無償公開のサンプルおよびライブラリであり、技術サポートは提供していません。

本プロジェクトのソースコードおよびドキュメントは現状のまま（AS IS）で提供されます。
使用環境や接続機器などによる動作を保証するものではありません。

本プロジェクトの利用、変更、組み込み等は、利用者自身の責任において行ってください。

なお、本プロジェクトのソースコード、README およびその他のドキュメントは、予告なく変更される場合があります。
ソースコードの変更にあたっては、可能な限り既存バージョンとの互換性を維持する方針としています。

改訂履歴

日付	内容
2026-09-01	初版