OTA Update Sample Code

# tuya\_ble\_ota.c

#include "tuya\_ble\_ota.h"

#include "tuya\_ble\_log.h"

#include "tuya\_ble\_api.h"

#include "tuya\_ble\_utils.h"

#include "ty\_ble.h"

#include "nrf\_fstorage.h"

#include "nrf\_dfu\_settings.h"

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

 \* LOCAL CONSTANTS

 \*/

#define TUYA\_BLE\_OTA\_STATE\_UNKNOWN (-1)

#define TUYA\_BLE\_OTA\_VERSION (3)

#define TUYA\_BLE\_OTA\_PKG\_LEN (512)

#define TUYA\_BLE\_OTA\_FILE\_MD5\_LEN (16)

#define TUYA\_BLE\_OTA\_START\_ADDR BOARD\_FLASH\_OTA\_START\_ADDR

#define TUYA\_BLE\_OTA\_END\_ADDR BOARD\_FLASH\_OTA\_END\_ADDR

#define TUYA\_BLE\_OTA\_FILE\_MAX\_LEN (TUYA\_BLE\_OTA\_END\_ADDR-TUYA\_BLE\_OTA\_START\_ADDR)

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

 \* LOCAL STRUCT

 \*/

#pragma pack(1)

typedef struct{

 uint8\_t flag;

 uint8\_t ota\_version;

 uint8\_t type;

 uint32\_t version;

 uint16\_t package\_maxlen;

} tuya\_ble\_ota\_req\_rsp\_t;

typedef struct{

 uint8\_t type;

 uint8\_t pid[8];

 uint32\_t version;

 uint8\_t md5[TUYA\_BLE\_OTA\_FILE\_MD5\_LEN];

 uint32\_t file\_len;

 uint32\_t crc32;

} tuya\_ble\_ota\_file\_info\_t;

typedef struct{

 uint8\_t type;

 uint8\_t state;

 uint32\_t old\_file\_len;

 uint32\_t old\_crc32;

 uint8\_t old\_md5[TUYA\_BLE\_OTA\_FILE\_MD5\_LEN];

} tuya\_ble\_ota\_file\_info\_rsp\_t;

typedef struct{

 uint8\_t type;

 uint32\_t offset;

} tuya\_ble\_ota\_file\_offset\_t;

typedef struct{

 uint8\_t type;

 uint32\_t offset;

} tuya\_ble\_ota\_file\_offset\_rsp\_t;

typedef struct{

 uint8\_t type;

 uint16\_t pkg\_id;

 uint16\_t len;

 uint16\_t crc16;

 uint8\_t data[];

} tuya\_ble\_app\_ota\_data\_t;

typedef struct{

 uint8\_t type;

 uint8\_t state;

} tuya\_ble\_ota\_data\_rsp\_t;

typedef struct{

 uint8\_t type;

 uint8\_t state;

} tuya\_ble\_ota\_end\_rsp\_t;

typedef struct{

 uint32\_t len;

 uint32\_t crc32;

 uint8\_t md5[TUYA\_BLE\_OTA\_FILE\_MD5\_LEN];

} tuya\_ble\_ota\_file\_info\_storage\_t;

#pragma pack()

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

 \* LOCAL VARIABLES

 \*/

static volatile int8\_t s\_ota\_state = TUYA\_BLE\_OTA\_STATE\_UNKNOWN;

static volatile int32\_t s\_pkg\_id;

static volatile bool s\_ota\_success;

static uint32\_t s\_data\_len;

static uint32\_t s\_data\_crc;

static uint32\_t s\_file\_len;

static uint32\_t s\_file\_crc;

static uint8\_t s\_file\_md5[TUYA\_BLE\_OTA\_FILE\_MD5\_LEN];

//file info

static tuya\_ble\_ota\_file\_info\_storage\_t\* s\_old\_file;

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

 \* LOCAL FUNCTION

 \*/

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

 \* VARIABLES

 \*/

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

FN:

\*/

static uint32\_t tuya\_ble\_ota\_enter(void)

{

 s\_pkg\_id = -1;

 s\_ota\_success = false;

 s\_data\_len = 0;

 s\_data\_crc = 0;

 s\_file\_len = 0;

 s\_file\_crc = 0;

 memset(s\_file\_md5, 0, TUYA\_BLE\_OTA\_FILE\_MD5\_LEN);

 ty\_ble\_set\_conn\_param(15, 30, 0, 6000);

 ty\_ble\_set\_dle();

 return 0;

}

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

FN:

\*/

static uint32\_t tuya\_ble\_ota\_exit(void)

{

 if(!s\_ota\_success) {

 s\_dfu\_settings.bank\_1.image\_crc = 0;

 s\_dfu\_settings.bank\_1.image\_size = 0;

 s\_dfu\_settings.bank\_1.bank\_code = 0;

 memset(&s\_dfu\_settings.progress, 0, sizeof(dfu\_progress\_t));

 s\_dfu\_settings.write\_offset = 0;

 s\_dfu\_settings.progress.update\_start\_address = 0;

// memset(&s\_dfu\_settings, 0, sizeof(nrf\_dfu\_settings\_t));

 nrf\_dfu\_settings\_write\_and\_backup(NULL);

 }

 s\_ota\_state = TUYA\_BLE\_OTA\_STATE\_UNKNOWN;

 tuya\_ble\_disconnect\_and\_reset\_timer\_start();

 return 0;

}

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

FN:

\*/

static uint32\_t tuya\_ble\_ota\_get\_crc32\_in\_flash(uint32\_t len)

{

 static uint8\_t buf[TUYA\_BLE\_OTA\_PKG\_LEN];

 if(len == 0) {

 return 0;

 }

 uint32\_t crc\_temp = 0;

 uint32\_t read\_addr = TUYA\_BLE\_OTA\_START\_ADDR;

 uint32\_t cnt = len/TUYA\_BLE\_OTA\_PKG\_LEN;

 uint32\_t remainder = len%TUYA\_BLE\_OTA\_PKG\_LEN;

 for(uint32\_t idx=0; idx<cnt; idx++) {

 tuya\_ble\_nv\_read(read\_addr, buf, TUYA\_BLE\_OTA\_PKG\_LEN);

 crc\_temp = tuya\_ble\_crc32\_compute(buf, TUYA\_BLE\_OTA\_PKG\_LEN, &crc\_temp);

 read\_addr += TUYA\_BLE\_OTA\_PKG\_LEN;

 }

 if(remainder > 0) {

 tuya\_ble\_nv\_read(read\_addr, buf, TUYA\_BLE\_OTA\_PKG\_LEN);

 crc\_temp = tuya\_ble\_crc32\_compute(buf, remainder, &crc\_temp);

 read\_addr += remainder;

 }

 return crc\_temp;

}

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

FN:

\*/

static void tuya\_ble\_ota\_setting\_write\_complete\_cb(nrf\_fstorage\_evt\_t\* p\_evt)

{

 UNUSED\_PARAMETER(p\_evt);

 tuya\_ble\_ota\_exit();

}

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

FN:

\*/

static uint32\_t tuya\_ble\_ota\_rsp(tuya\_ble\_ota\_response\_t\* rsp, void\* rsp\_data, uint16\_t data\_size)

{

// TUYA\_APP\_LOG\_HEXDUMP\_INFO("ota\_rsp\_data", rsp\_data, data\_size);

 rsp->p\_data = rsp\_data;

 rsp->data\_len = data\_size;

 return tuya\_ble\_ota\_response(rsp);

}

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

FN:

\*/

static uint32\_t tuya\_ble\_ota\_req\_handler(uint8\_t\* cmd, uint16\_t cmd\_size, tuya\_ble\_ota\_response\_t\* rsp)

{

 //param check

 if(s\_ota\_state != TUYA\_BLE\_OTA\_STATE\_UNKNOWN) {

 TUYA\_APP\_LOG\_ERROR("TUYA\_BLE\_OTA\_REQ- s\_ota\_state error");

 //rsp

 tuya\_ble\_ota\_req\_rsp\_t req\_rsp;

 memset(&req\_rsp, 0x00, sizeof(tuya\_ble\_ota\_req\_rsp\_t));

 req\_rsp.flag = 0x01; //refuse ota

 tuya\_ble\_ota\_rsp(rsp, &req\_rsp, sizeof(tuya\_ble\_ota\_req\_rsp\_t));

 tuya\_ble\_ota\_exit();

 return 1;

 }

 s\_ota\_state = TUYA\_BLE\_OTA\_REQ;

 //param check

 if((cmd\_size != 0x0001) || (\*cmd != 0x00)) {

 TUYA\_APP\_LOG\_ERROR("TUYA\_BLE\_OTA\_REQ- param error");

 //rsp

 tuya\_ble\_ota\_req\_rsp\_t req\_rsp;

 memset(&req\_rsp, 0x00, sizeof(tuya\_ble\_ota\_req\_rsp\_t));

 req\_rsp.flag = 0x01; //refuse ota

 tuya\_ble\_ota\_rsp(rsp, &req\_rsp, sizeof(tuya\_ble\_ota\_req\_rsp\_t));

 tuya\_ble\_ota\_exit();

 return 1;

 }

 {

 tuya\_ble\_ota\_enter();

 //rsp

 tuya\_ble\_ota\_req\_rsp\_t req\_rsp;

 memset(&req\_rsp, 0x00, sizeof(tuya\_ble\_ota\_req\_rsp\_t));

 req\_rsp.flag = 0x00; //accept ota

 req\_rsp.ota\_version = TUYA\_BLE\_OTA\_VERSION;

 req\_rsp.type = 0x00; //firmware info

 req\_rsp.version = TY\_DEVICE\_FVER\_NUM;

 tuya\_ble\_inverted\_array((void\*)&req\_rsp.version, sizeof(uint32\_t));

 req\_rsp.package\_maxlen = TUYA\_BLE\_OTA\_PKG\_LEN;

 tuya\_ble\_inverted\_array((void\*)&req\_rsp.package\_maxlen, sizeof(uint16\_t));

 tuya\_ble\_ota\_rsp(rsp, &req\_rsp, sizeof(tuya\_ble\_ota\_req\_rsp\_t));

 s\_ota\_state = TUYA\_BLE\_OTA\_FILE\_INFO;

 }

 return 0;

}

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

FN:

\*/

static uint32\_t tuya\_ble\_ota\_file\_info\_handler(uint8\_t\* cmd, uint16\_t cmd\_size, tuya\_ble\_ota\_response\_t\* rsp)

{

 //param check

 if(s\_ota\_state != TUYA\_BLE\_OTA\_FILE\_INFO) {

 TUYA\_APP\_LOG\_ERROR("TUYA\_BLE\_OTA\_FILE\_INFO- s\_ota\_state error");

 //rsp none

 tuya\_ble\_ota\_exit();

 return 1;

 }

 //param check

 tuya\_ble\_ota\_file\_info\_t\* file\_info = (void\*)cmd;

 if(file\_info->type != 0x00) {

 TUYA\_APP\_LOG\_ERROR("TUYA\_BLE\_OTA\_FILE\_INFO- file\_info->type error");

 //rsp none

 tuya\_ble\_ota\_exit();

 return 1;

 }

 {

 //file info

 tuya\_ble\_inverted\_array((void\*)&file\_info->version, sizeof(uint32\_t));

 tuya\_ble\_inverted\_array((void\*)&file\_info->file\_len, sizeof(uint32\_t));

 tuya\_ble\_inverted\_array((void\*)&file\_info->crc32, sizeof(uint32\_t));

 s\_file\_len = file\_info->file\_len;

 s\_file\_crc = file\_info->crc32;

 memcpy(s\_file\_md5, file\_info->md5, TUYA\_BLE\_OTA\_FILE\_MD5\_LEN);

 //rsp

 tuya\_ble\_ota\_file\_info\_rsp\_t file\_info\_rsp;

 memset(&file\_info\_rsp, 0x00, sizeof(tuya\_ble\_ota\_file\_info\_rsp\_t));

 file\_info\_rsp.type = 0x00; //firmware info

 if(file\_info->version <= TY\_DEVICE\_FVER\_NUM) {

 file\_info\_rsp.state = 0x02; //version error

 }

 else if(file\_info->file\_len > TUYA\_BLE\_OTA\_FILE\_MAX\_LEN) {

 file\_info\_rsp.state = 0x03; //size error

 } else {

 file\_info\_rsp.state = 0x00;

 s\_ota\_state = TUYA\_BLE\_OTA\_FILE\_OFFSET\_REQ;

 }

 file\_info\_rsp.old\_file\_len = s\_old\_file->len;

 tuya\_ble\_inverted\_array((void\*)&file\_info\_rsp.old\_file\_len, sizeof(uint32\_t));

 file\_info\_rsp.old\_crc32 = s\_old\_file->crc32;

 tuya\_ble\_inverted\_array((void\*)&file\_info\_rsp.old\_crc32, sizeof(uint32\_t));

 memset(file\_info\_rsp.old\_md5, 0x00, TUYA\_BLE\_OTA\_FILE\_MD5\_LEN);

 tuya\_ble\_ota\_rsp(rsp, &file\_info\_rsp, sizeof(tuya\_ble\_ota\_file\_info\_rsp\_t));

 if(file\_info\_rsp.state != 0x00) {

 TUYA\_APP\_LOG\_ERROR("TUYA\_BLE\_OTA\_FILE\_INFO- errorid: %d", file\_info\_rsp.state);

 tuya\_ble\_ota\_exit();

 }

 }

 return 0;

}

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

FN:

\*/

static uint32\_t tuya\_ble\_ota\_file\_offset\_handler(uint8\_t\* cmd, uint16\_t cmd\_size, tuya\_ble\_ota\_response\_t\* rsp)

{

 //param check

 if(s\_ota\_state != TUYA\_BLE\_OTA\_FILE\_OFFSET\_REQ) {

 TUYA\_APP\_LOG\_ERROR("TUYA\_BLE\_OTA\_FILE\_OFFSET\_REQ- s\_ota\_state error");

 //rsp none

 tuya\_ble\_ota\_exit();

 return 1;

 }

 //param check

 tuya\_ble\_ota\_file\_offset\_t\* file\_offset = (void\*)cmd;

 if(file\_offset->type != 0x00) {

 TUYA\_APP\_LOG\_ERROR("TUYA\_BLE\_OTA\_FILE\_OFFSET\_REQ- file\_offset->type error");

 //rsp none

 tuya\_ble\_ota\_exit();

 return 1;

 }

 {

 tuya\_ble\_inverted\_array((void\*)&file\_offset->offset, sizeof(uint32\_t));

 //rsp

 tuya\_ble\_ota\_file\_offset\_rsp\_t file\_offset\_rsp;

 memset(&file\_offset\_rsp, 0x00, sizeof(tuya\_ble\_ota\_file\_offset\_rsp\_t));

 file\_offset\_rsp.type = 0x00;

 {

 if(file\_offset->offset > 0)

 {

 if((memcmp(s\_old\_file->md5, s\_file\_md5, TUYA\_BLE\_OTA\_FILE\_MD5\_LEN) == 0)

 && (tuya\_ble\_ota\_get\_crc32\_in\_flash(s\_old\_file->len) == s\_old\_file->crc32)

 && (file\_offset->offset >= s\_old\_file->len)) {

 file\_offset\_rsp.offset = s\_old\_file->len;

 s\_data\_len = s\_old\_file->len;

 s\_data\_crc = s\_old\_file->crc32;

 } else {

 file\_offset\_rsp.offset = 0;

 s\_data\_len = 0;

 s\_data\_crc = 0;

 }

 }

 memcpy(s\_old\_file->md5, s\_file\_md5, TUYA\_BLE\_OTA\_FILE\_MD5\_LEN);

 nrf\_dfu\_settings\_write\_and\_backup(NULL);

 }

 tuya\_ble\_inverted\_array((void\*)&file\_offset\_rsp.offset, sizeof(uint32\_t));

 tuya\_ble\_ota\_rsp(rsp, &file\_offset\_rsp, sizeof(tuya\_ble\_ota\_file\_offset\_rsp\_t));

 s\_ota\_state = TUYA\_BLE\_OTA\_DATA;

 }

 return 0;

}

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

FN:

\*/

static uint32\_t tuya\_ble\_ota\_data\_handler(uint8\_t\* cmd, uint16\_t cmd\_size, tuya\_ble\_ota\_response\_t\* rsp)

{

 //param check

 if(s\_ota\_state != TUYA\_BLE\_OTA\_DATA) {

 TUYA\_APP\_LOG\_ERROR("TUYA\_BLE\_OTA\_DATA- s\_ota\_state error");

 //rsp

 tuya\_ble\_ota\_data\_rsp\_t ota\_data\_rsp;

 memset(&ota\_data\_rsp, 0x00, sizeof(tuya\_ble\_ota\_data\_rsp\_t));

 ota\_data\_rsp.state = 0x04; //unknow error

 tuya\_ble\_ota\_rsp(rsp, &ota\_data\_rsp, sizeof(tuya\_ble\_ota\_data\_rsp\_t));

 tuya\_ble\_ota\_exit();

 return 1;

 }

 //param check

 tuya\_ble\_app\_ota\_data\_t\* ota\_data = (void\*)cmd;

 if(ota\_data->type != 0x00) {

 TUYA\_APP\_LOG\_ERROR("TUYA\_BLE\_OTA\_DATA- ota\_data->type error");

 //rsp

 tuya\_ble\_ota\_data\_rsp\_t ota\_data\_rsp;

 memset(&ota\_data\_rsp, 0x00, sizeof(tuya\_ble\_ota\_data\_rsp\_t));

 ota\_data\_rsp.state = 0x04; //unknow error

 tuya\_ble\_ota\_rsp(rsp, &ota\_data\_rsp, sizeof(tuya\_ble\_ota\_data\_rsp\_t));

 tuya\_ble\_ota\_exit();

 return 1;

 }

 {

 tuya\_ble\_inverted\_array((void\*)&ota\_data->pkg\_id, sizeof(uint16\_t));

 tuya\_ble\_inverted\_array((void\*)&ota\_data->len, sizeof(uint16\_t));

 tuya\_ble\_inverted\_array((void\*)&ota\_data->crc16, sizeof(uint16\_t));

 //rsp

 tuya\_ble\_ota\_data\_rsp\_t ota\_data\_rsp;

 memset(&ota\_data\_rsp, 0x00, sizeof(tuya\_ble\_ota\_data\_rsp\_t));

 ota\_data\_rsp.type = 0x00;

 if(s\_pkg\_id+1 != ota\_data->pkg\_id) {

 ota\_data\_rsp.state = 0x01; //package id error

 }

 else if(cmd\_size-7 != ota\_data->len) {

 ota\_data\_rsp.state = 0x02; //size error

 }

 else if(tuya\_ble\_crc16\_compute(ota\_data->data, ota\_data->len, NULL) != ota\_data->crc16) {

 ota\_data\_rsp.state = 0x03; //crc error

 } else {

 ota\_data\_rsp.state = 0x00;

 //erase

 bool flag\_4k = false;

 if((s\_data\_len == 0) || ((s\_data\_len + ota\_data->len) >= (((s\_data\_len/TUYA\_NV\_ERASE\_MIN\_SIZE) + 1)\*TUYA\_NV\_ERASE\_MIN\_SIZE)))

 {

 if(s\_data\_len == 0) {

 tuya\_ble\_nv\_erase(TUYA\_BLE\_OTA\_START\_ADDR, TUYA\_NV\_ERASE\_MIN\_SIZE);

 } else {

 uint32\_t erase\_addr = TUYA\_BLE\_OTA\_START\_ADDR + (((s\_data\_len/TUYA\_NV\_ERASE\_MIN\_SIZE) + 1)\*TUYA\_NV\_ERASE\_MIN\_SIZE);

 tuya\_ble\_nv\_erase(erase\_addr, TUYA\_NV\_ERASE\_MIN\_SIZE);

 }

 flag\_4k = true;

 }

 if(0 != tuya\_ble\_nv\_write(TUYA\_BLE\_OTA\_START\_ADDR + s\_data\_len, ota\_data->data, ota\_data->len)) {

 ota\_data\_rsp.state = 0x04; //write error

 } else {

 s\_data\_len += ota\_data->len;

 if(s\_data\_len < s\_file\_len) {

 s\_ota\_state = TUYA\_BLE\_OTA\_DATA;

 } else if(s\_data\_len == s\_file\_len) {

 s\_ota\_state = TUYA\_BLE\_OTA\_END;

 } else {

 ota\_data\_rsp.state = 0x04;

 }

 s\_pkg\_id++;

 TUYA\_APP\_LOG\_INFO("s\_pkg\_id: %d", s\_pkg\_id);

 s\_data\_crc = tuya\_ble\_crc32\_compute(ota\_data->data, ota\_data->len, &s\_data\_crc);

 if(flag\_4k) {

 s\_old\_file->len = s\_data\_len;

 s\_old\_file->crc32 = s\_data\_crc;

 nrf\_dfu\_settings\_write\_and\_backup(NULL);

 }

 }

 }

 tuya\_ble\_ota\_rsp(rsp, &ota\_data\_rsp, sizeof(tuya\_ble\_ota\_data\_rsp\_t));

 if(ota\_data\_rsp.state != 0x00) {

 TUYA\_APP\_LOG\_ERROR("TUYA\_BLE\_OTA\_DATA- errorid: %d", ota\_data\_rsp.state);

 tuya\_ble\_ota\_exit();

 }

 }

 return 0;

}

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

FN:

\*/

static uint32\_t tuya\_ble\_ota\_end\_handler(uint8\_t\* cmd, uint16\_t cmd\_size, tuya\_ble\_ota\_response\_t\* rsp)

{

 //param check

 if(s\_ota\_state != TUYA\_BLE\_OTA\_END) {

 TUYA\_APP\_LOG\_ERROR("TUYA\_BLE\_OTA\_END- s\_ota\_state error");

 //rsp

 tuya\_ble\_ota\_end\_rsp\_t end\_rsp;

 memset(&end\_rsp, 0x00, sizeof(tuya\_ble\_ota\_end\_rsp\_t));

 end\_rsp.state = 0x03; //unknow error

 tuya\_ble\_ota\_rsp(rsp, &end\_rsp, sizeof(tuya\_ble\_ota\_end\_rsp\_t));

 tuya\_ble\_ota\_exit();

 return 1;

 }

 //param check

 if((cmd\_size != 0x0001) || (\*cmd != 0x00)) {

 TUYA\_APP\_LOG\_ERROR("TUYA\_BLE\_OTA\_END- type error");

 //rsp

 tuya\_ble\_ota\_end\_rsp\_t end\_rsp;

 memset(&end\_rsp, 0x00, sizeof(tuya\_ble\_ota\_end\_rsp\_t));

 end\_rsp.state = 0x03; //unknow error

 tuya\_ble\_ota\_rsp(rsp, &end\_rsp, sizeof(tuya\_ble\_ota\_end\_rsp\_t));

 tuya\_ble\_ota\_exit();

 return 1;

 }

 {

 //rsp

 tuya\_ble\_ota\_end\_rsp\_t end\_rsp;

 memset(&end\_rsp, 0x00, sizeof(tuya\_ble\_ota\_end\_rsp\_t));

 end\_rsp.type = 0x00;

 if(s\_data\_len != s\_file\_len) {

 end\_rsp.state = 0x01; //total size error

 } else if(s\_file\_crc != tuya\_ble\_ota\_get\_crc32\_in\_flash(s\_data\_len)) {

 end\_rsp.state = 0x02; //crc error

 } else {

 end\_rsp.state = 0x00;

 s\_ota\_success = true;

 TUYA\_APP\_LOG\_INFO("ota success");

 {

 memset(&s\_dfu\_settings, 0, sizeof(nrf\_dfu\_settings\_t));

 s\_dfu\_settings.bank\_1.image\_crc = s\_file\_crc;

 s\_dfu\_settings.bank\_1.image\_size = s\_file\_len;

 s\_dfu\_settings.bank\_1.bank\_code = NRF\_DFU\_BANK\_VALID\_APP;

 memset(&s\_dfu\_settings.progress, 0, sizeof(dfu\_progress\_t));

 s\_dfu\_settings.write\_offset = 0;

 s\_dfu\_settings.progress.update\_start\_address = TUYA\_BLE\_OTA\_START\_ADDR;

 nrf\_dfu\_settings\_write\_and\_backup((nrf\_dfu\_flash\_callback\_t)tuya\_ble\_ota\_setting\_write\_complete\_cb);

 }

 }

 tuya\_ble\_ota\_rsp(rsp, &end\_rsp, sizeof(tuya\_ble\_ota\_end\_rsp\_t));

 if(end\_rsp.state != 0x00) {

 TUYA\_APP\_LOG\_ERROR("TUYA\_BLE\_OTA\_END- errorid: %d", end\_rsp.state);

 tuya\_ble\_ota\_exit();

 }

 }

 return 0;

}

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

FN:

\*/

uint32\_t tuya\_ble\_ota\_init(void)

{

 s\_old\_file = (tuya\_ble\_ota\_file\_info\_storage\_t\*)(s\_dfu\_settings.init\_command);

 uint32\_t ret = nrf\_dfu\_settings\_init(true);

 APP\_ERROR\_CHECK(ret);

 return ret;

}

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

FN:

\*/

void tuya\_ble\_ota\_handler(tuya\_ble\_ota\_data\_t\* ota)

{

 tuya\_ble\_ota\_response\_t rsp;

 rsp.type = ota->type;

 if(ota->type != TUYA\_BLE\_OTA\_DATA) {

 TUYA\_APP\_LOG\_INFO("ota\_cmd\_type: %d", ota->type);

 TUYA\_APP\_LOG\_HEXDUMP\_INFO("ota\_cmd\_data", ota->p\_data, ota->data\_len);

 }

 switch(ota->type)

 {

 case TUYA\_BLE\_OTA\_REQ: {

 tuya\_ble\_ota\_req\_handler(ota->p\_data, ota->data\_len, &rsp);

 } break;

 case TUYA\_BLE\_OTA\_FILE\_INFO: {

 tuya\_ble\_ota\_file\_info\_handler(ota->p\_data, ota->data\_len, &rsp);

 } break;

 case TUYA\_BLE\_OTA\_FILE\_OFFSET\_REQ: {

 tuya\_ble\_ota\_file\_offset\_handler(ota->p\_data, ota->data\_len, &rsp);

 } break;

 case TUYA\_BLE\_OTA\_DATA: {

 tuya\_ble\_ota\_data\_handler(ota->p\_data, ota->data\_len, &rsp);

 } break;

 case TUYA\_BLE\_OTA\_END: {

 tuya\_ble\_ota\_end\_handler(ota->p\_data, ota->data\_len, &rsp);

 } break;

 case TUYA\_BLE\_OTA\_UNKONWN: {

 } break;

 default: {

 } break;

 }

}

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

FN:

\*/

uint32\_t tuya\_ble\_ota\_get\_state(void)

{

 return s\_ota\_state;

}

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

FN:

\*/

uint32\_t tuya\_ble\_ota\_disconn\_handler(void)

{

 if(s\_ota\_state >= TUYA\_BLE\_OTA\_REQ) {

 return tuya\_ble\_ota\_exit();

 } else {

 return 0;

 }

}

# tuya\_ble\_ota.h

/\*\*

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

\* @file tuya\_ble\_ota.h

\* @brief tuya\_ble\_ota

\* @author suding

\* @version V1.0.0

\* @date 2019-09-11

\* @note

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

\* @attention

\*

\* <h2><center>&copy; COPYRIGHT 2019 Tuya </center></h2>

\*/

#ifndef \_\_TUYA\_BLE\_OTA\_H\_\_

#define \_\_TUYA\_BLE\_OTA\_H\_\_

#ifdef \_\_cplusplus

extern "C"

{

#endif

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

 \* INCLUDES

 \*/

#include "stdint.h"

#include "tuya\_ble\_type.h"

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

 \* CONSTANTS

 \*/

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

 \* STRUCT

 \*/

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

 \* EXTERNAL VARIABLES

 \*/

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

 \* EXTERNAL FUNCTIONS

 \*/

uint32\_t tuya\_ble\_ota\_init(void);

void tuya\_ble\_ota\_handler(tuya\_ble\_ota\_data\_t\* ota);

uint32\_t tuya\_ble\_ota\_get\_state(void);

uint32\_t tuya\_ble\_ota\_disconn\_handler(void);

#ifdef \_\_cplusplus

}

#endif

#endif //\_\_TUYA\_BLE\_OTA\_H\_\_