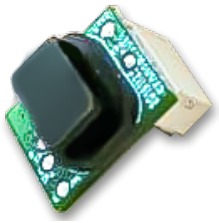


FSTOF200xC0x ToF module
Preliminary application guide

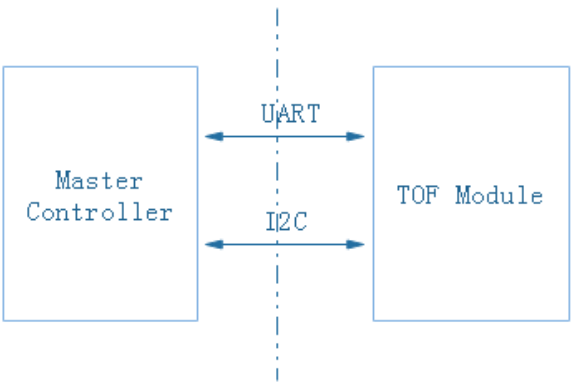
Features

- 940nm laser classified as class 1 under operation condition by IEC 60825-1:2014-3rd edition
- High speed distance measurement response
- Advanced optical cross-talk compensation
- Easy to set
- No additional optical calibration requirement
- Single power supply
- Lead-free, RoHS compliant

Contents

1. Introduction

The main feature of this TOF module is to quickly calculate the precise distance. Our TOF module also supports two interfaces: UART and I2C. When using the UART interface, the master controller sends the command, and then the TOF Module processes the response. When using I2C interface to access, TOF module is used as slave mode and master controller is used as master mode.



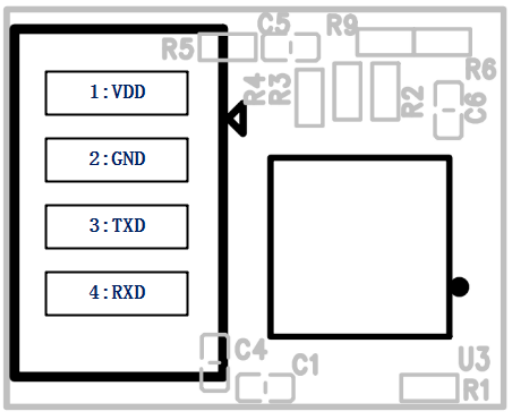
* Note: Regarding the command code, the user needs to check/modify at command "0x81" for measuring distance. Other commands are basically not used.

2. Introduction to Uart and I2C mode

The communication between Master Controller and TOF Module is mainly done by two modes: Uart Mode and I2C Mode. The modes needs to be selected according to your hardware design. Two modes are described in detail as below:

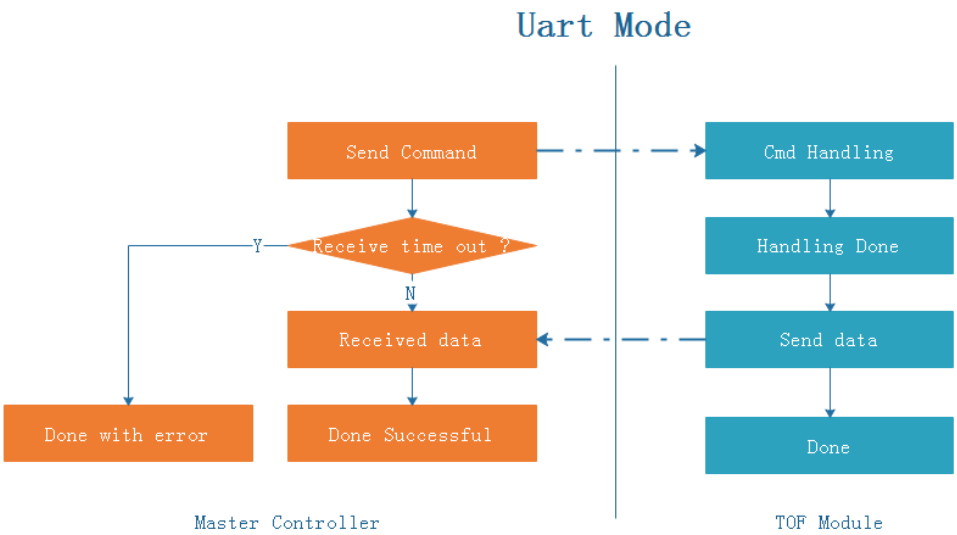
2.1: UART mode

- Hardware interface



- Program design of UART mode

The following figure describes the complete interaction process of master controller and TOF module in UART mode:



- How to set UART parameters

The baud rate of the TOF module is set to 9600, and the master controller needs to synchronize when setting UART parameters. The specific settings of all parameters are as follows

Parameter	Value	Description
UART Baud Rate	9600	This parameter represent the UART communication baud rate.
UART Word Length	8	Specifies the number of data bits transmitted or received in a frame.
UART Stop Bits	1	Specifies the number of stop bits transmitted.
UART Parity	None	Specifies the parity mode.
UART Hardware Flow Control	None	Specifies wether the hardware flow control mode is enabled or disabled.

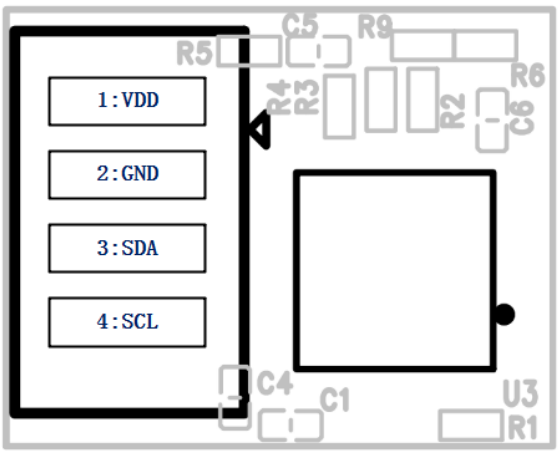
- Introduction of UART data package

When UART communication is used between master controller and TOF module, the format of transmission packet is fixed, and the specific protocol format is as follows:
Instruction format: packet header + command code + data length + data + end flag

Offset(bytes)	Length(bytes)	Type	Description
0	2	uint16_t	Packet header: 0x55 0xAA
2	1	uint8_t	Command Code
3	1	uint8_t	The length of the following parameter
4	N	Array of uint8_t	Parameter, N = Data Length, N+4 < 32, depending the TOF Module buf size
L	1	uint8_t	End flag:0xFA , Offset computational formula is L = Data Length + 4 .

2.2: I2C mode

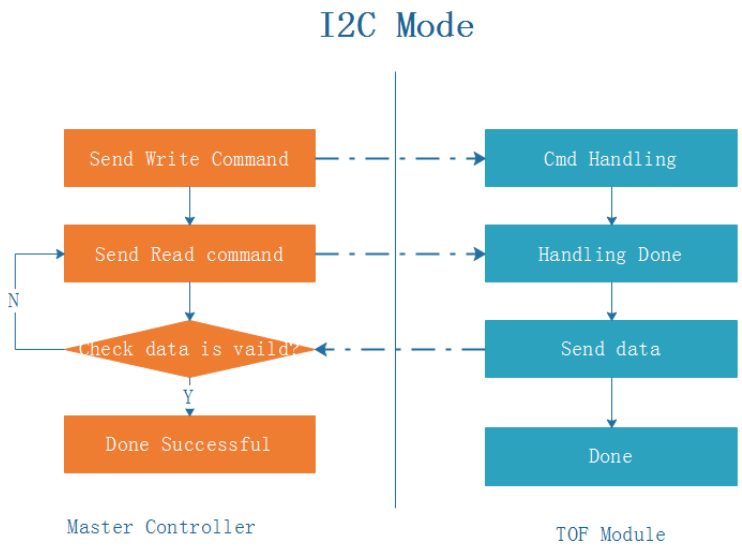
- hardware interface



- I2C mode programming

The following figure describes the complete interaction process between master controller and TOF module when in I2C mode:

* Note: send write CMD is required before send read CMD. When getting distance value, it is recommended to send read CMD after send write CMD for 36ms, because TOF module needs processing time. Less than 36 MS may read the last data.



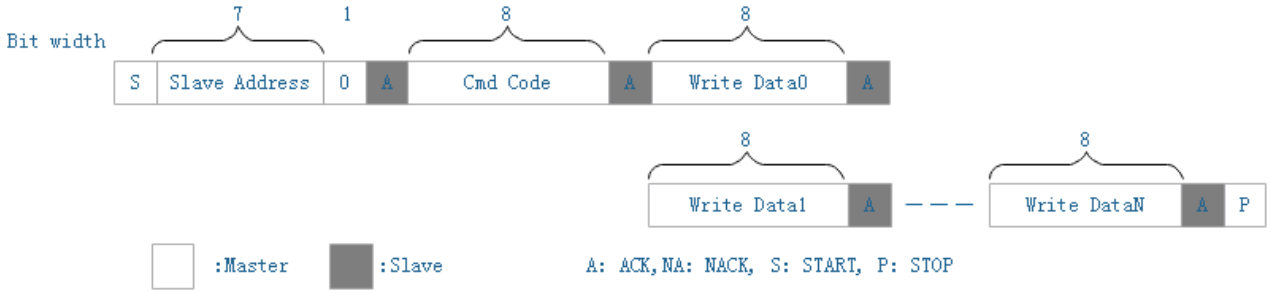
I2C bus interface

TOF module I2C works in slave mode, I2C slave address: 0x36

ADDRESS	A6	A5	A4	A3	A2	A1	A0	R/W
	0	1	1	0	1	1	0	X

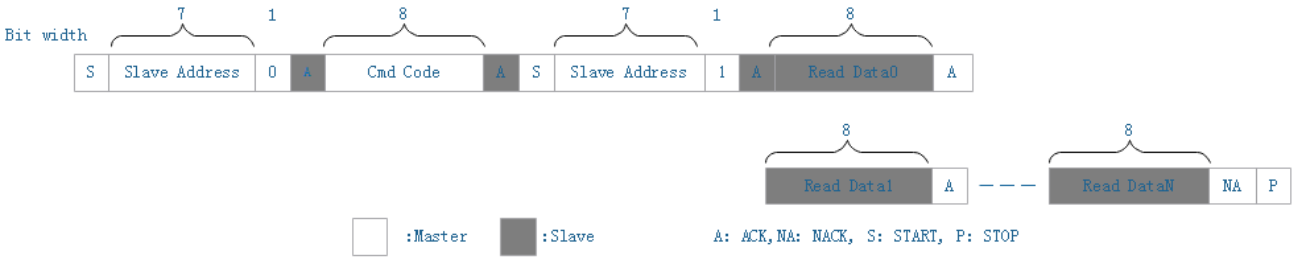
- Write format

The figure below describes in detail the data format to be sent when writing



- Read format

The figure below describes in detail the data format to be sent when reading multiple data through I2C.



I2C packet format

The format of I2C write and read is described in detail below. Here is the TOF module response data format. The first byte received is command code, followed by the data. If the first byte of response is CMD code, it means that there is valid data after it. If the first byte of the response is not the CMD code (0x00), the following data is invalid data. In this case, the TOF module does not complete the processing of the CMD.

TOF Module Response Data Format：

Offset(bytes)	Length(bytes)	Type	Description
0	1	uint8_t	Command Code
1	1	uint8_t	Data length
2 to N	N-2	uint8_t	Raw data. The data length is different for each command. The N computational formula is N = data length + 2.

3: Introduction to command code

Here we would like to introduce about the commands TOF module supports at present and the data format of each command reply. UART and I2C will encapsulate the valid data again to facilitate the correct analysis of data. The formats of UART and I2C packages are different. Please refer to the below sections for the specific format.

- **Command Code List**

All of the following CMD codes have the same effect on UART mode or I2C mode, as well as the limited data returned.

Command Code	Description
0x81	Measuring distance, Unit: mm
0x82	Crosstalk calibration
0x83	Offset calibration
0x84	Reset Tof module
0x85	Read factory data
0x86	Read version information
0x8A	Read debug parameter section 1
0x8B	Read debug parameter section 2
0x8C	Read debug parameter section 3
0x91	Set the threshold of interrupt output
0x92	Set the ranging mode

- **Factory Calibration Error Code List**

When the TOF module performs calibration CMD error, the error code response will be sent to the master controller.

Error Code	Description
0x00	No Error
0x01	Crosstalk calibration is error
0x02	Offset calibration is error

- **Command functions**

(1): Measuring distance cmd: 0x81
The valid data format of CMD response is as follows:

Offset(bytes)	Length(bytes)	Type	Description
0	2	int16_t	range1，measuring distance value.(Unit:mm)
2	1	uint8_t	range1_status, measurement status1.

The range1 status error code is as follows:

Error Code	Error Status	Description
0x00	VALID_DATA	Valid data.
0x01	VCSEL_SHORT	When the VCSEL is short-circuited. If this error occurs, the VCSEL current will not flow inside the IC.
0x02	LOW_SIGNAL	The amount of reflected light obtained from the detected object is small.
0x04	LOW_SN	The ratio of reflected light from the detected object and disturbance light is small.
0x08	TOO_MUCH_AMB	Disturbance light is large.
0x10	WAF	Wrap around error.
0x20	CAL_ERROR	Internal calculation error.
0x80	CROSSTALK_ERROR	Crosstalk from the panel is large.

The following is the reference sample code, which mainly reflects the data formats sent and received by UART and I2C:

```
// Uart send cmd:(Master -> TOF)
uint8_t distance_cmd[] = {0x55, 0xAA, 0x81, 0x00, 0xFA};

// I2c send cmd:(Master ->
TOF)uint8_t CmdCode = 0x81;uint8_t data = 0x01;
i2c_write_byte(CmdCode, data);

/*****************************************/

//When range1 = 0x004d, range1_ When status = 0x00, the data format received by UART and I2C is as follows:
// Uart Recieve Data:(TOF -> Master)
uint8_t uart_recv_buf[] = {0x55, 0xAA, 0x81, 0x03, 0x00, 0x4D, 0x00, 0xFA};

// I2C Recieve Data:(TOF -> Master)
uint8_t i2c_recv_buf[256]={0};
uint8_t CmdCode = 0x81;
uint8_t len = 5;i2c_read_buffer(i2c_recv_buf, CmdCode, len);
//Finish I2C_ read_ After buffer function, I2C_ recv_ buf[256] = {0x81, 0x03, 0x00, 0x4D, 0x00};
```

(2): Crosstalk calibration cmd: 0x82

After the TOF module receives the CMD, it first performs the crosstalk calibration, then saves the calibration value to flash and feeds back to the master controller. The valid data format of CMD response is as follows:

Offset(bytes)	Length(bytes)	Type	Description
0	1	uint8_t	errCode · Refer to Factory Calibration Error Code List sections.
1	1	uint8_t	xtakLsb, crosstalk data_factory
2	1	uint8_t	xtakMsb, crosstalk data_factory

(3): Offset calibration cmd: 0x83

After receiving the CMD, TOF module first performs offset calibration, then saves the calibration value to flash and feeds back to the master controller. The valid data format of CMD response is as follows:

Offset(bytes)	Length(bytes)	Type	Description
0	1	uint8_t	errCode · Refer to Factory Calibration Error Code List sections.
1	1	uint8_t	offset_short1, offset for short range mode 1
2	1	uint8_t	offset_short2, offset for short range mode 2
3	1	uint8_t	offset_long1, offset for long range mode 1
4	1	uint8_t	offset_long2, offset for long range mode 2

*Note: TOF module uses gp2ap03vt sensor without offset_ Long1 and offset_ Only gp2ap02vt sensor has these two data.

(4): Read factory data cmd: 0x85

The main function of the CMD is to obtain the factory data saved on the flash after calibration. The valid data format of the CMD response is as follows:

Offset(bytes)	Length(bytes)	Type	Description
0	1	uint8_t	offset_short1, offset for short range mode 1
1	1	uint8_t	offset_short2, offset for short range mode 2
2	1	uint8_t	offset_long1, offset for long range mode 1
3	1	uint8_t	offset_long2, offset for long range mode 2
4	1	uint8_t	xtakLsb, crosstalk data_factory
5	1	uint8_t	xtakMsb, crosstalk data_factory
6	1	uint8_t	factory_calibrated: 0x00: Calibration is not done. 0x01: Offset calibration is passed. 0x02: Crosstalk calibration is passed. 0x03:Offset and Crosstalk calibration is passed.

*Note: TOF module uses gp2ap03vt sensor without offset_ Long1 and offset_ Only gp2ap02vt sensor has these two data.

(5): Read version information cmd: 0x86

The main function of the CMD is to obtain the software and hardware related information of the TOF module. The valid data format of the CMD response is as follows:

Offset(bytes)	Length(bytes)	Type	Description
0	1	uint8_t	sensor_ic : 0x02: The sensor is gp2ap02vt. 0x03: The sensor is gp2ap03vt.
1	1	uint8_t	port: 0x41('A'): Firmware supports both UART and I2C interfaces. 0x49('I'): Firmware supports I2C interface. 0x55('U'): Firmware supports UART interface.
2	1	uint8_t	Firmware version information for the TOF module.

(6): Read debug parameter section 1 cmd: 0x8A

The CMD is mainly used to track the debug information of the problem when there is a problem. The data format of section 1 is as follows:

Offset(bytes)	Length(bytes)	Type	Description
0	2	int16_t	range1, range1 mm
2	1	uint8_t	range1_status, measurement status1
3	2	uint16_t	range1_raw, range1 mm for checking value before error processing
5	4	int32_t	ret_sig, return signal
9	4	int32_t	data_xtalk, crosstalk data

(7): Read debug parameter section 2 cmd: 0x8B

The CMD is mainly used to track the debug information of the problem when there is a problem. The data format of section 2 is as follows:

Offset(bytes)	Length(bytes)	Type	Description
0	4	int32_t	ret_maxcnt_fx28p
4	4	int32_t	ref_sig, reference signal
8	4	int32_t	amb_fx28p

(8): Read debug parameter section 3 cmd: 0x8C

The CMD is mainly used to track the debug information of the problem when there is a problem. The data format of section 3 is as follows:

Offset(bytes)	Length(bytes)	Type	Description
0	4	int32_t	data_xtalk_factory, crosstalk data_factory
4	1	int8_t	offset_short1, offset for short range mode 1
5	4	int32_t	data_xtalk_pre, crosstalk data

NOTE:

1: The debug information returned by the TOF module is large-scale. If the master controller is a small-scale one, it needs to be converted here. You can refer to the following swap 32 and swap 16 large and small end sequence conversion.

2: Pay attention to the problem of structure word alignment. Defining structure needs attribute ((packed)) modification, and tells the compiler to cancel the optimized alignment of structure in the compilation process and align according to the actual number of bytes occupied.

The following is the sample code of read debug parameter:

test_debug_info.h

```
/**
* @brief TOF module Command code
*
*/
typedef enum {
    TOFM_CMD_NONE = 0x00,
    TOFM_CMD_START_FLAG = 0x80,
    TOFM_CMD_ST_MM = 0x81,
    TOFM_CMD_CALI_XTALK = 0x82,
TOFM_CMD_CALI_OFS = 0x83,
    TOFM_CMD_RESET = 0x84,
    TOFM_CMD_RD_FACTORY_DATA = 0x85,
    TOFM_CMD_RD_VERSION_INFO = 0x86,
    TOFM_CMD_RD_DEBUG_PARA1 = 0x8A,
    TOFM_CMD_RD_DEBUG_PARA2 = 0x8B,
    TOFM_CMD_RD_DEBUG_PARA3 = 0x8C,

    TOFM_CMD_CODE_COUNT,
}TOFM_CMD_CODE_E;

/**
 * @brief I2C data playload
 *
 */

typedef struct{
    uint8_t cmdCode;   /**< command code, from TOFM_CMD_CODE_E */
    uint8_t dataLen; /**< length of the following parameter */
    uint8_t data[0];
}__attribute__((packed))I2C_DATA_PAYLOAD_T;

/**
 * @brief Debug parameters
 *
 */typedef struct
{    int16_t range1; /*!< range1 mm */
    uint8_t range1_status; /*!< measurement status1 */
uint16_t range1_raw; /* < range1 mm for checking value before error processing */
    int32_t ret_sig; /*!< return signal */
    int32_t data_xtalk; /*!< crosstalk data */
}__attribute__((packed))TOFM_DEBUG_PARA1_PAYLOAD_T;

typedef struct
{
    int32_t ret_maxcnt_fx28p;
    int32_t ref_sig; /*!< reference signal */
    int32_t amb_fx28p;
}__attribute__((packed))TOFM_DEBUG_PARA2_PAYLOAD_T;

typedef struct
{
    int32_t data_xtalk_factory; /*!< crosstalk data_factory */
int8_t offset_short1; /*!< offset for short range mode 1 */
    int32_t data_xtalk_pre; /*!< crosstalk data */
}__attribute__((packed))TOFM_DEBUG_PARA3_PAYLOAD_T;
```

test_debug_info.c

```
#define I2C_READ_BUFFER_SIZE    256
static uint8_t I2c_Buf_Read[I2C_READ_BUFFER_SIZE];

#define SWAP32(a) \
 ((a&0x000000ff)<<24) | ((a&0x0000ff00)<<8) | ((a&0x00ff0000)>>8) | ((a&0xff000000)>>24)
#define SWAP16(a) \
 ((a&0x00ff)<<8) | ((a&0xff00)>>8)

void Test_Debug_Info(void)
{
    uint8_t wdata;
    I2C_DATA_PAYLOAD_T *i2cDataBuf = (I2C_DATA_PAYLOAD_T *)I2c_Buf_Read;
    memset(I2c_Buf_Read, 0, I2C_READ_BUFFER_SIZE);
    wdata = 0x01;
    I2C_TOFM_BufferWrite(TOFM_CMD_ST_MM, &wdata, sizeof(uint8_t));
Delay_ms(40);
    I2C_TOFM_BufferRead(TOFM_CMD_ST_MM, I2c_Buf_Read, sizeof(I2C_DATA_PAYLOAD_T) +
sizeof(TOFM_MEAS_DATA_PAYLOAD_T));
if((i2cDataBuf->cmdCode == TOFM_CMD_ST_MM) && (i2cDataBuf->dataLen == sizeof(TOFM_MEAS_DATA_PAYLOAD_T)))
    {
        TOFM_MEAS_DATA_PAYLOAD_T *measDataPayload = (TOFM_MEAS_DATA_PAYLOAD_T *)i2cDataBuf->data;
        printf("range1 = %dmm\r\n", measDataPayload->range1_msb << 8 | measDataPayload->range1_lsb);
    }

    printf("########################Debug Info#######################\r\n");

    wdata = 0x01;
    I2C_TOFM_BufferWrite(TOFM_CMD_RD_DEBUG_PARA1, &wdata, sizeof(uint8_t));
    Delay_ms(1);
    I2C_TOFM_BufferRead(TOFM_CMD_RD_DEBUG_PARA1, I2c_Buf_Read, sizeof(I2C_DATA_PAYLOAD_T) +
sizeof(TOFM_DEBUG_PARA1_PAYLOAD_T));

    if((i2cDataBuf->cmdCode == TOFM_CMD_RD_DEBUG_PARA1) && (i2cDataBuf->dataLen ==
sizeof(TOFM_DEBUG_PARA1_PAYLOAD_T)))
    {
    TOFM_DEBUG_PARA1_PAYLOAD_T *dbgPara1Payload = (TOFM_DEBUG_PARA1_PAYLOAD_T *)i2cDataBuf->data;
        printf("range1 = 0x%04x\r\n", SWAP16(dbgPara1Payload->range1));
        printf("range1_status = 0x%02x\r\n",dbgPara1Payload->range1_status);
        printf("range1_raw = 0x%04x\r\n",SWAP16(dbgPara1Payload->range1_raw));
        printf("ret_sig = 0x%08x\r\n",SWAP32(dbgPara1Payload->ret_sig));
        printf("data_xtalk = 0x%08x\r\n",SWAP32(dbgPara1Payload->data_xtalk));
    }
    wdata = 0x01;
    I2C_TOFM_BufferWrite(TOFM_CMD_RD_DEBUG_PARA2, &wdata, sizeof(uint8_t));
    Delay_ms(1);
    I2C_TOFM_BufferRead(TOFM_CMD_RD_DEBUG_PARA2, I2c_Buf_Read, sizeof(I2C_DATA_PAYLOAD_T) +
sizeof(TOFM_DEBUG_PARA2_PAYLOAD_T));

if((i2cDataBuf->cmdCode == TOFM_CMD_RD_DEBUG_PARA2) && (i2cDataBuf->dataLen == sizeof(TOFM_DEBUG_PARA2_PAYLOAD_T)))
    {
        TOFM_DEBUG_PARA2_PAYLOAD_T *dbgPara2Payload = (TOFM_DEBUG_PARA2_PAYLOAD_T *)i2cDataBuf->data;
        printf("ret_maxcnt_fx28p = 0x%08x\r\n", SWAP32(dbgPara2Payload->ret_maxcnt_fx28p));
        printf("ref_sig = 0x%08x\r\n", SWAP32(dbgPara2Payload->ref_sig));
        printf("amb_fx28p = 0x%08x\r\n", SWAP32(dbgPara2Payload->amb_fx28p));
    }

    wdata = 0x01;
    I2C_TOFM_BufferWrite(TOFM_CMD_RD_DEBUG_PARA3, &wdata, sizeof(uint8_t));
    Delay_ms(1);
    I2C_TOFM_BufferRead(TOFM_CMD_RD_DEBUG_PARA3, I2c_Buf_Read, sizeof(I2C_DATA_PAYLOAD_T) +
sizeof(TOFM_DEBUG_PARA3_PAYLOAD_T));

    if((i2cDataBuf->cmdCode == TOFM_CMD_RD_DEBUG_PARA3) && (i2cDataBuf->dataLen ==
sizeof(TOFM_DEBUG_PARA3_PAYLOAD_T)))
    {
        TOFM_DEBUG_PARA3_PAYLOAD_T *dbgPara3Payload = (TOFM_DEBUG_PARA3_PAYLOAD_T *)i2cDataBuf->data;
    printf("data_xtalk_factory = 0x%08x\r\n", SWAP32(dbgPara3Payload->data_xtalk_factory));
        printf("offset_short1 = 0x%02x\r\n",dbgPara3Payload->offset_short1);
        printf("data_xtalk_pre = 0x%08x\r\n", SWAP32(dbgPara3Payload->data_xtalk_pre));
    }
    printf("############################################################\r\n");
}
```

Test results:

```
range1 = 316mm
########################Debug Info#########################
range1 = 0x013c
range1_status = 0x00
range1_raw = 0x013c
ret_sig = 0x00000488
data_xtalk = 0x0000016c
ret_maxcnt_fx28p = 0x00558000
ref_sig = 0x0000a000
amb_fx28p = 0x00000366
data_xtalk_factory = 0x00000000
offset_short1 = 0x00
data_xtalk_pre = 0x00000000
############################################################
```

(9): Set the threshold of interrupt output: 0x91

The CMD is mainly used to set the threshold of the output interrupt signal, which includes: short distance and long distance. The specific data format is as follows:

Offset(bytes)	Length(bytes)	Type	Description
0	2	uint16_t	Close range threshold
2	2	uint16_t	Far range threshold

```
For example: 55 AA 91 04 64 00 E8 03 FA
Proximity threshold: 0x0064
Proximity threshold: 0x03E8
```

(10): Set the ranging mode: 0x92

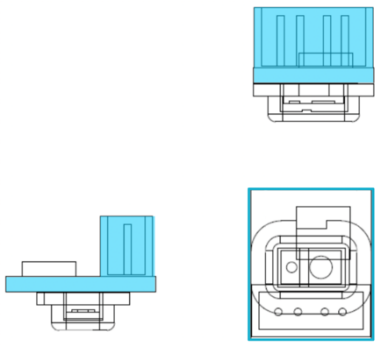
The CMD is mainly used to set the test distance mode, which can be set as the continuous output distance. The specific data format is as follows:

Offset(bytes)	Length(bytes)	Type	Description
0	1	uint8_t	0：Close continuous output data 1：Open continuous output data

For example:
Turn off continuous ranging mode: 55 AA 92 01 00 FA
Open continuous ranging mode: 55 AA 92 01 01 FA

4: Description on the design recommendation / installation method

When choosing a place for mounting, please consider the following recommendations:



- Please design appropriate housing or compartment to mechanically fit the tof module. Or, use glue to affix the module to your application , and pay attention to using non-conductive materials and do not touch any components on the PCB to avoid affecting the measurement accuracy and possibly even causing damage.
  (It is recommended to install on the surface with the color as shown in the figure above)
- Mounting close to sources of heat or strong electromagnetic fields can decrease the sensing performance
- Do not mount anything directly in front of the sensor or in a cone of approximately 25° around the central optical axis of the sensor
- It is better to avoid having other sources of Continuous Wave or modulated IR light close to the sensor
- Please consider that dust, dirt and condensation can affect the sensor performance

Date	Version	Description
20201215	1.01	Description on the design recommendation / installation method
20201005	1.0	Initial version