Vzense MIPI TOF Camera Module User Manual

Revision History

version number	Author
R01_20190701	Peter Liu, Jerry Deng, Sky Guo
R02_20190720	Sky Guo
R03_20200427	Peter Liu

1 Overview

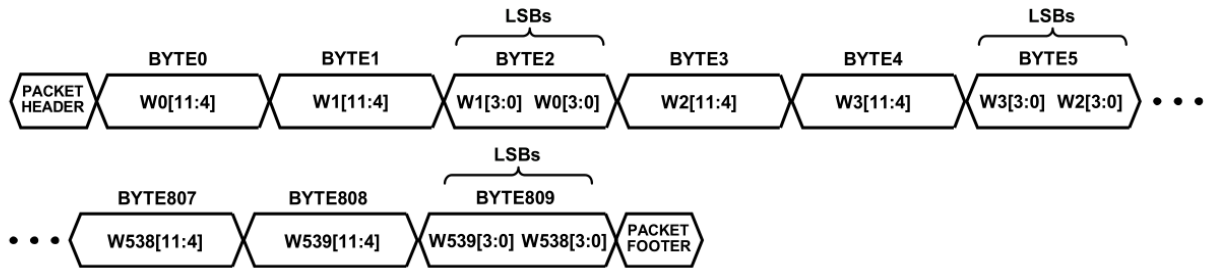
Welcome to the product software user manual for the Vzense MIPI TOF Camera Moduel (like DCAM10/20/80). The Module is a 3D camera hardware module, developed by Vzense, which uses time-of-flight (TOF) technology to capture depth information while recording imagery. The module using MIPI and I2C interface is designed for embedded platform in wide range of environments while providing high-precision depth information.

2 Interfaces

2.1 MIPI Interface

TOF module output 12-bit raw data. The MIPI interface can be configured for 1/2-lane via software setting. The one lane configuration can support 270Mbps, the two lanes configuration only can support 135Mbps each lane.

2.1.1 MIPI Raw Data Format



2.1.2 Hardware Waveforms Parameters(1-Lane)

Item	Value
MIPI Clock	270MHz
MIPI Prepare+Zero Time	108.00ns
MIPI High Speed Differential Voltage	200.00mV
MIPI Low Speed Voltage	1.20V

2.1.3 Hardware Waveforms Parameters(2-Lane)

Item	Value
MIPI Clock	135MHz
MIPI Prepare+Zero Time	111.00ns
MIPI High Speed Differential Voltage	200.00mV
MIPI Low Speed Voltage	1.20V

2.1.4 MIPI Data Reprocessing

Raw data is translated by MIPI lane, different SOC platform process raw data in the different ways. For example, on RV1108 platform, isp core converts 12bits raw data to 16bits data, guaranteed two bytes alignment. So in the user space, we need process 16bits data. We give sample code to process raw data like below.

TOF module output mode have three modes that is depth mode, IR mode and depth&IR mode. Depth mode format is the same as IR mode, the resolution is 640x480. Depth&IR mode resolution is 640x960.

Depth data format

Depth Data Content(640x480)
Depth Line 0
Depth Line 1
.....
Depth Line 478
Depth Line 479

IR data format

IR Data Content(640x480)
IR Line 0
IR Line 1
.....
IR Line 478
IR Line 479

Depht&IR data format

Depth & IR Data Content(640x960)
Depth Line 0
IR Line 0
Depth Line 1
IR Line 1
.....
.....
Depth Line 479
IR Line 479

2.2 I2C Interface

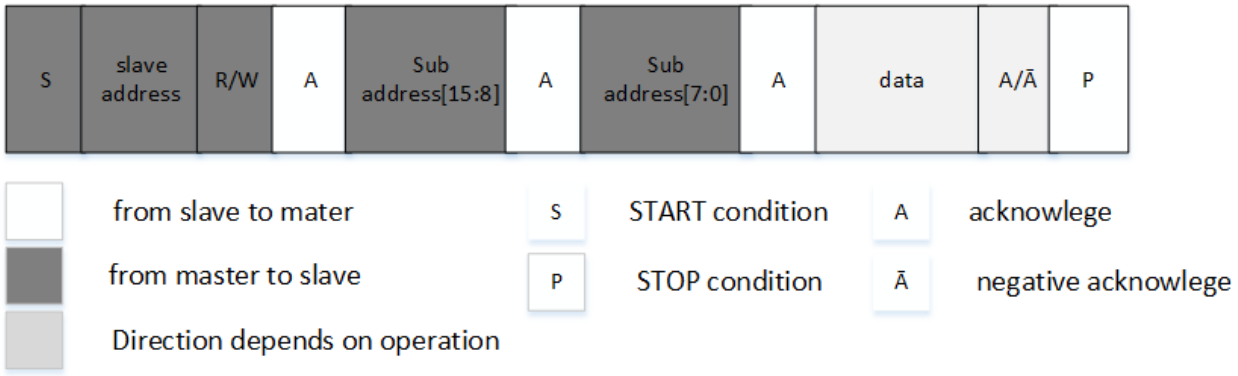
I2C interface control the TOF module operation. Refer the Vzense TOF protocol for detailed usage of the serial control port. I2C Bus have two wires and are named as Serial Clock Line (SCL) and Serial Data Line (SDA). The data to be transferred is sent through the SDA wire and is synchronized with the clock signal from SCL. The TOF module work as slave device, it responds commands from master device.

TOF module's clock frequency is 100Kbps. Slave address must be 0x64.

2.2.1 Data Transfer Protocol

The command and parameter transfer of TOF module use the I2C protocol.

Message Format

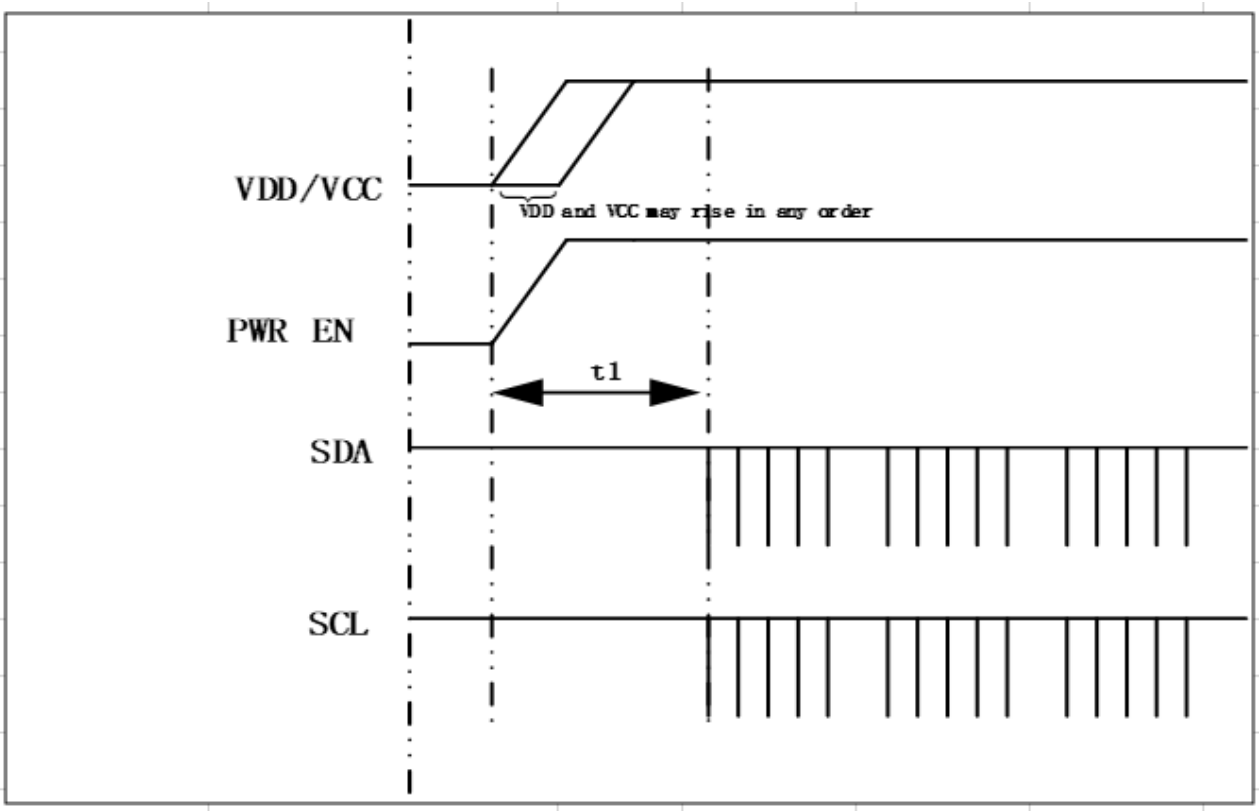


NOTE:

Slave address must be **0x64**.

3 Power management

3.1 Power Up Sequence



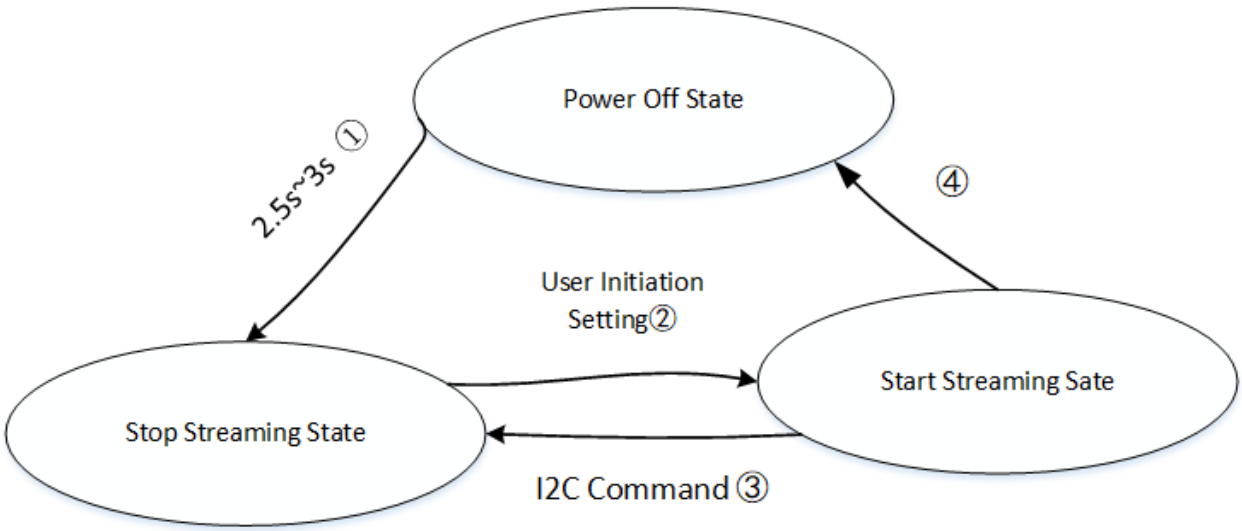
NOTE:

1. VDD can be powered on in any order.
2. After VDD on, the module will initial itself. The initialization need t1 time that is about 2.5s.
3. On different module version t1 is different.

3.2 Power Down Sequence

VDD can be powered down in any order.

4 States Machine



TOF Module have three states that is Power off, Stop streaming, Start streaming.

Power off: the module do not have any power.

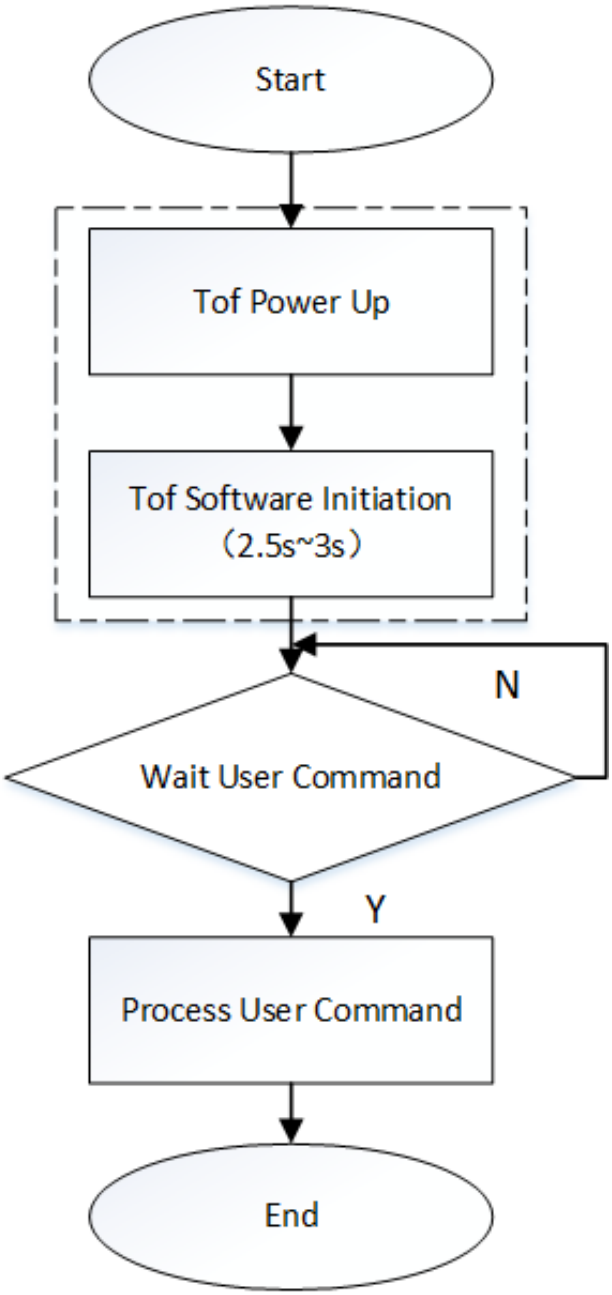
Stop streaming: module is end of default initialization, and can receive any user configuration.

Start streaming: image is streaming in the configuration.

When the TOF module is powered on, the module performs software initialization and enters the streaming stop state. This process takes about 2.5 seconds. When the module initialization is completed, it will wait for the user's initialization operation in stop streaming states. After the user setting configuration, it can enter start streaming state.

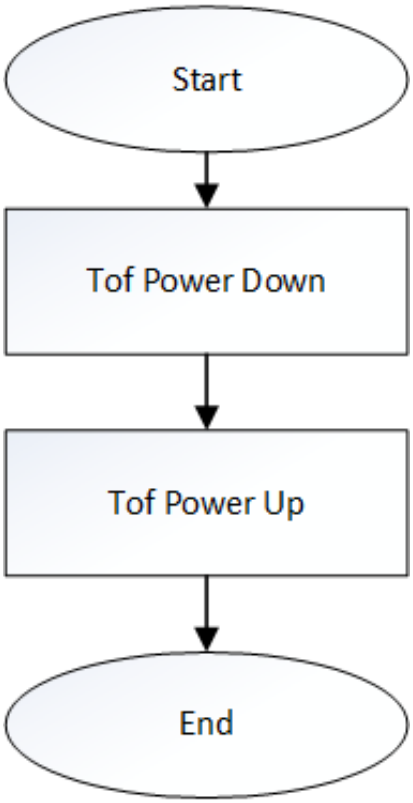
4.1 Power up Flow

TOF module's power up flow is like the below. Ensure the power supply of TOF module is fine. After that TOF module will initialize itself, then it enter the the stop streaming status. The TOF module will wait for user's command to configure or start streaming.



4.2 Reset Flow

To reset TOF module, it can power off the TOF module, and then power up the TOF module.



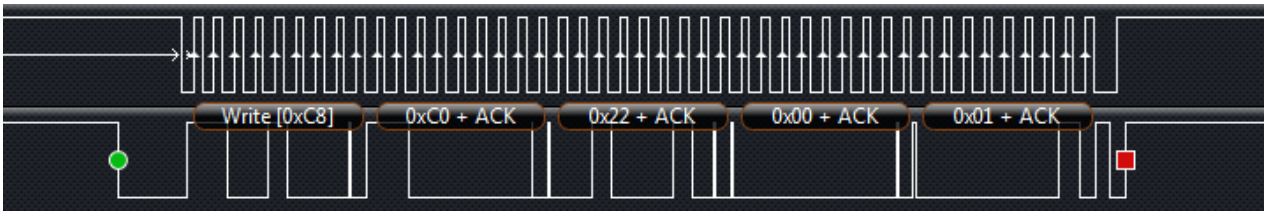
5 Command Operation

Set/Get operations can send/receive command to/from the controller immediately.

5.1 Set Command Type

Set command to the controller, example like below.

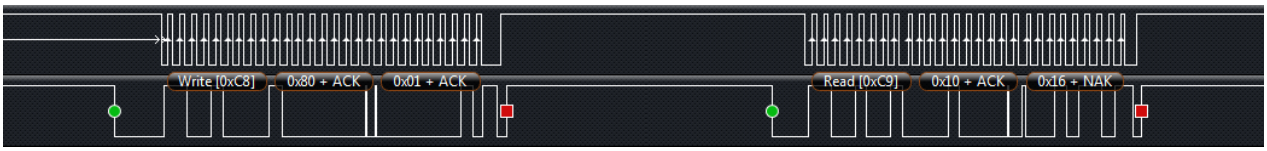
```
{0xC022, 0x0001},//write register 0xC022 value 0x0001
```



5.2 Get Command Type

Get command to the controller, example like below.

```
{0x8001, 0x1016}//read register 0x8001
```



5.3 Commands List

Index	CMD	Attribute	Address
1	Device ID	R	0x8001
2	Device Status	R / W	0x8002
3	Last Error	R	0x8003
4	Working OutMode	R / W	0x8004
5	Working Range A	R / W	0x8006
6	Range A Pulse Count	R / W	0x800a
7	Range A Gamma Gain	R / W	0x800b
8	Range A Threshold	R / W	0x800f
9	MIPI Prepare Value	W	0x8011
10	MIPI Zero Value	W	0x8012
11	IR Background Value	R / W	0x8013
12	Frame Rate Value	W	0x8017
13	MIPI Lane Number	W	0x801e
14	Work Mode Value	W	0x8005

1 Device ID (0x8001)

Read only command, the device id will always be 0x1016.

```
uint16_t device_id = iic_read(0x8001);
```

2 Device Status (0x8002)

This command will get device status machine value. If the driver want to manage module states accurately, it can use this command to query or check the status.

```
enum
{
    HW_CHECKING     =   0x00,
    INITIALING      =   0x01,
    STOP_SDREAMING  =   0x02,
    INIT_UPDATING   =   0x03,
    START_SDREAMING =   0x04,
};
```

3 Last Error (0x8003)

TBC

4 Working OutMode (0x8004)

```
enum
{
   DEPTH_MODE_NONE,
   DEPTH_30_MODE,
   IR_30_MODE,
   DEPTH_IR_30_MODE,
};
```

5 Working Range A(0x8006)

This command is used to change working range A. If TOF module works in the normal mode, only need to set working range A value.

```
enum
{
  DEPTH_RANGE0 = 0x00,
  DEPTH_RANGE1,
  DEPTH_RANGE2,
  DEPTH_RANGE3,
  DEPTH_RANGE4,
  DEPTH_RANGE5,
  DEPTH_RANGE6,
  DEPTH_RANGE7,
  DEPTH_RANGE8,
};
```

Using different range setting, module can calculate different distance range.

INDEX	RANGE SETTING	THEORETICAL MAXIMUM(mm)
0	Range 0	2394
1	Range 1	3331
2	Range 2	4944

6 Range A Pulse Count(0x800a)

This command is used to set pulse count value of range A separately. In general, we use the default values for the calibration, we do not need to modify the default value.

7 Range A Gamma Gain(0x800b)

This command is used to set gamma gain value of range A separately.

```
#define ADDI903X_MIN_GAMMA_GAIN 0
#define ADDI903X_MAX_GAMMA_GAIN  0x0fff
```

8 Range A Threshold(0x800f)

This command is used to set threshold value of range A separately.

```
#define ADDI903X_MIN_BG_THREHOLD 0
#define ADDI903X_MAX_BG_THREHOLD 0x3fff
```

9 MIPI Prepare Value(0x8011)

This command is used to change MIPI configure parameter, the value can change MIPI prepare time. Usually, use the default value, do not need change this value. When the MIPI lane number is configured to two lanes, the default prepare value is 3. When the MIPI lane number is configured to one lane, the default prepare value is 10.

If MIPI lane is configured to two lanes, the prepare value scope:

```
#define ADDI903X_MIN_MIPI_PREPARE_VALUE  3
#define ADDI903X_MAX_MIPI_PREPARE_VALUE 16
```

If MIPI lane is configured to one lane, the prepare value scope:

```
#define ADDI903X_MIN_MIPI_PREPARE_VALUE  6
#define ADDI903X_MAX_MIPI_PREPARE_VALUE 30
```

10 MIPI Zero Value(0x8012)

This command is used to change MIPI configure parameter, it can change the MIPI zero time. Usually, use the default value, do not need change this value.

If MIPI lane is configured to two lanes, the zero value scope:

```
#define ADDI903X_MIN_MIPI_ZERO_VALUE  4
#define ADDI903X_MAX_MIPI_ZERO_VALUE 16
```

If MIPI lane is configured to one lane, the zero value scope:

```
#define ADDI903X_MIN_MIPI_ZERO_VALUE  8
#define ADDI903X_MAX_MIPI_ZERO_VALUE 30
```

11 IR Background Value(0x8013)

This command is used to control IR background on or off.

```
enum
{
  IR_BACKGROUND_OFF = 0,
  IR_BACKGROUND_ON  = 1,
};
```

12 Frame Rate Value(0x8017)

This command is used to set TOF module frame rate.

```
#define ADDI903X_MIN_SETTING_FRAME_RATE 10
#define ADDI903X_MAX_SETTING_FRAME_RATE 30
```

13 MIPI Lane Number (0x801e)

This command is used to configure MIPI lane number. The number can be set to one or two.

```
enum
{
  MIPI_ONE_LANE_CONFIGURE = 1,
  MIPI_TWO_LANE_CONFIGURE = 2,
};
```

14 Work Mode Value(0x8005)

This command is used to configure work mode whether working in the normal mode. To configure work Mode Value to be 1, TOF module will work in normal mode.

6 Parameter Operation

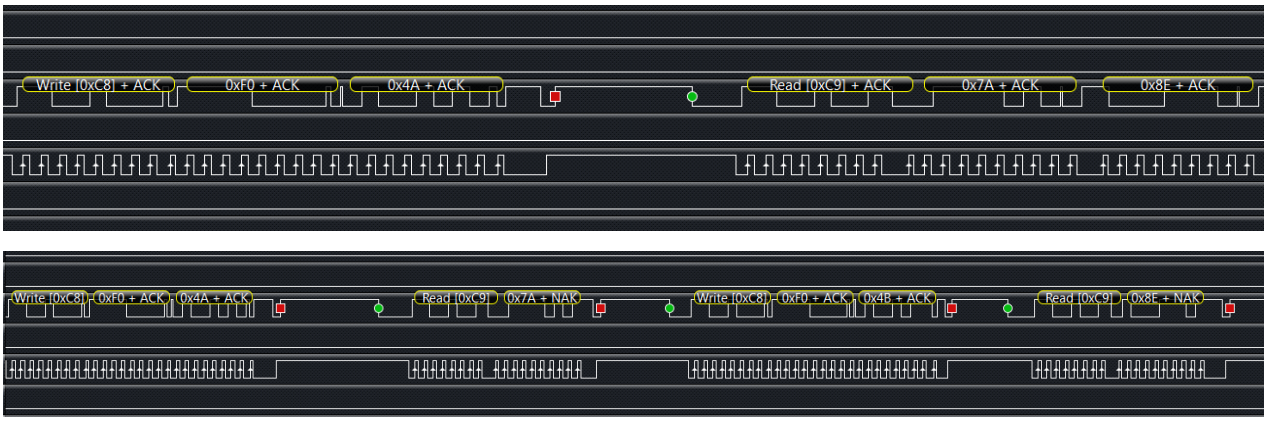
Write/Read operations can not send/receive parameter to/from the controller immediately. The controller need time to prepare the data after receive the write information. Master can check the status of data prepare, if it is ready, master can read it.

6.1 Write Parameter Type

Write parameter operation like the set command, but the content length of the data is different. So if the parameter length is bigger than 2 bytes, the IIC need write the data continuously after write address.

6.2 Read Parameter Type

Read parameter operation like the get command, but the content length of the data is different. If the parameter length is bigger than 2 bytes, the IIC can read the data continuously after write address, and also support single byte read option.



6.3 Parameters

Parameters include calibration data and module informations. All parameters are stored by little-endian.

CMD	Attribute	Address	Length(Byte)
Mipi Prepare Value Record	R / W	0xEC00	2
Mipi Zero Value Record	R / W	0xEC04	2
Max Depth And Range	R	0xEFFE	54
Depth Intrinsic	R	0xF04A	72
Depth Distortion	R	0xF094	64
RGB Intrinsic	R	0xF0D6	72
RGB Distortion	R	0xF120	64
Rotation Mat	R	0xF162	72
Transfer Mat	R	0xF1AC	24
E Mat	R	0xF1C6	72
F Mat	R	0xF210	72
SN	R	0xF25A	20
Gamma Gain	R / W	0xF273	2
Firmware Version	R	0xF2FC	20

1 MIPI Prepare Value(0xEC00)

This saved parameter is used to configure MIPI parameters. This parameter store the value in module and use it on next initialization.

2 MIPI Zero Value(0xEC04)

This saved parameter is also used to configure MIPI parameters. This parameter store the value in module and use it on next initialization.

3 Depth Intrinsic(0xF04A)

Depth camera's internal parameter.

This parameter is used for algorithm.

```
double matrix[3 * 3] =
{
  Fx, 0,  Cx,
  0,  Fy, Cy,
  0,  0,  1,
};
```

Fx, Fy : pixel focal length

Cx, Cy: camera's principal point

Each variable in the matrix takes 8 bytes. The number of variables in the matrix is 9.

0	1	2	3	4	5	6	7
Fx							
8	9	10	11	12	13	14	15
0							
16	17	18	19	20	21	22	23
Cx							
24	25	26	27	28	29	30	31
0							
32	33	34	35	36	37	38	39
Fy							
40	41	42	43	44	45	46	47
Cy							
48	49	50	51	52	53	54	55
0							
56	57	58	59	60	61	62	63
0							
64	65	66	67	68	69	70	71
1							

4 Depth Distortion(0xF04A)

Depth camera's internal parameter.

This parameter is used for algorithm.

```
double matrix[1 * 8] = {K1,K2,P1,P2,K3,K4,K5,K6};
```

K1~6: the coefficients of the radial distortion

P1, P2: the coefficients of the tangential distortion

Each variable in the matrix takes 8 bytes. The number of variables in the matrix is 8.

0	1	2	3	4	5	6	7
K1							
8	9	10	11	12	13	14	15
K2							
16	17	18	19	20	21	22	23
P1							
24	25	26	27	28	29	30	31
P2							
32	33	34	35	36	37	38	39
K3							
40	41	42	43	44	45	46	47
K4							
48	49	50	51	52	53	54	55
K5							
56	57	58	59	60	61	62	63
K6							

5 RGB Intrinsic(0xF04A)

RGB camera's internal parameter.

This parameter is used for algorithm.

```
double matrix[3 * 3] =
{
  Fx, 0,  Cx,
  0,  Fy, Cy,
  0,  0,  1,
};
```

Fx, Fy : pixel focal length

Cx, Cy: camera's principal point

Each variable in the matrix takes 8 bytes. The number of variables in the matrix is 9.

0	1	2	3	4	5	6	7
Fx							
8	9	10	11	12	13	14	15
0							
16	17	18	19	20	21	22	23
Cx							
24	25	26	27	28	29	30	31
0							
32	33	34	35	36	37	38	39
Fy							
40	41	42	43	44	45	46	47
Cy							
48	49	50	51	52	53	54	55
0							
56	57	58	59	60	61	62	63
0							
64	65	66	67	68	69	70	71
1							

6 RGB Distortion(0xF120)

RGB camera's internal parameter.

This parameter is used for algorithm.

```
double matrix[1*8] = {K1,K2,P1,P2,K3,K4,K5,K6};
```

K1~6: the coefficients of the radial distortion

P1, P2: the coefficients of the tangential distortion

Each variable in the matrix takes 8 bytes. The number of variables in the matrix is 8.

0	1	2	3	4	5	6	7
K1							
8	9	10	11	12	13	14	15
K2							
16	17	18	19	20	21	22	23
P1							
24	25	26	27	28	29	30	31
P2							
32	33	34	35	36	37	38	39
K3							
40	41	42	43	44	45	46	47
K4							
48	49	50	51	52	53	54	55
K5							
56	57	58	59	60	61	62	63
K6							

7 Rotation Mat(0xF162)

Camera's external parameter.

This parameter is used for algorithm.

```
double matrix[3 * 3] =
{
  r11,  r12,  r13,
  r21,  r22,  r23,
  r31,  r32,  r33,
};
```

Each variable in the matrix takes 8 bytes. The number of variables in the matrix is 9.

0	1	2	3	4	5	6	7
R11							
8	9	10	11	12	13	14	15
R12							
16	17	18	19	20	21	22	23
R13							
24	25	26	27	28	29	30	31
R21							
32	33	34	35	36	37	38	39
R22							
40	41	42	43	44	45	46	47
R23							
48	49	50	51	52	53	54	55
R31							
56	57	58	59	60	61	62	63
R32							
64	65	66	67	68	69	70	71
R33							

8 Transfer Mat(0xF1AC)

Camera's external parameter.

This parameter is used for algorithm.

```
double matrix[1 * 3] = {t1, t2, t3};
```

Each variable in the matrix takes 8 bytes. The number of variables in the matrix is 3.

0	1	2	3	4	5	6	7
t1							
8	9	10	11	12	13	14	15
t2							
16	17	18	19	20	21	22	23
t3							

9 Essential Mat(0xF1C6)

Camera's external parameter.

This parameter is used for algorithm.

```
double matrix[3 * 3] =
{
  e11,  e12,  e13,
  e21,  e22,  e23,
  e31,  e32,  e33,
};
```

Each variable in the matrix takes 8 bytes. The number of variables in the matrix is 9.

0	1	2	3	4	5	6	7
e11							
8	9	10	11	12	13	14	15
e12							
16	17	18	19	20	21	22	23
e13							
24	25	26	27	28	29	30	31
e21							
32	33	34	35	36	37	38	39
e22							
40	41	42	43	44	45	46	47
e23							
48	49	50	51	52	53	54	55
e31							
56	57	58	59	60	61	62	63
e32							
64	65	66	67	68	69	70	71
e33							

10 F Mat(0xF210)

Camera's external parameter.

This parameter is used for algorithm.

```
double matrix[3 * 3] =
{
  f11,  f12,  f13,
  f21,  f22,  f23,
  f31,  f32,  f33,
};
```

Each variable in the matrix takes 8 bytes. The number of variables in the matrix is 9.

0	1	2	3	4	5	6	7
f11							
8	9	10	11	12	13	14	15
f12							
16	17	18	19	20	21	22	23
f13							
24	25	26	27	28	29	30	31
f21							
32	33	34	35	36	37	38	39
f22							
40	41	42	43	44	45	46	47
f23							
48	49	50	51	52	53	54	55
f31							
56	57	58	59	60	61	62	63
f32							
64	65	66	67	68	69	70	71
f33							

11 SN(0xF25A)

This parameter is used to store factory serial number.

12 Gamma gain(0xF273)

This parameter is used to store default gamma gain value.

13 Max Depth And Range(0xEFFE)

This parameter is used to store every range's effective and max distance. The first 18 bytes save the theoretical maximum of nine kinds of ranges, and the theoretical maximum of each range occupies two bytes. The middle 18 bytes save the effective maximum value of the calibration of nine kinds of ranges, and the effective maximum value of each range occupies two bytes. The last 18 bytes store the effective minimum values calibrated by nine kinds of ranges, and the effective minimum values of each range occupy two bytes.

1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18
range0 theoretical maximum		range1 theoretical maximum		range2 theoretical maximum		range3 theoretical maximum		range4 theoretical maximum		range5 theoretical maximum		range6 theoretical maximum		range7 theoretical maximum		range8 theoretical maximum	
19	20	21	22	23	24	25	26	27	28	29	30	31	32	33	34	35	36
range0 calibration effective maximum		range1 calibration effective maximum		range2 calibration effective maximum		range3 calibration effective maximum		range4 calibration effective maximum		range5 calibration effective maximum		range6 calibration effective maximum		range7 calibration effective maximum		range8 calibration effective maximum	
37	38	39	40	41	42	43	44	45	46	47	48	49	50	51	52	53	54
range0 calibration effective minimum		range1 calibration effective minimum		range2 calibration effective minimum		range3 calibration effective minimum		range4 calibration effective minimum		range5 calibration effective minimum		range6 calibration effective minimum		range7 calibration effective minimum		range8 calibration effective minimum	

Distance formula:

$$RealDistance = RangexTheoreticalMaximum * 4.0f/16384 * (RawData) + 0.5f$$

$$MaxOffset = 0.004f * RangexTheoreticalMaximum$$

```
if(RealDistance + MaxOffset <  RangexTheoreticalMaximum)
  RealDistance = RangexTheoreticalMaximum * 4.0f / 16384 * (RawData) + 0.5f;
else
  RealDistance = RangexTheoreticalMaximum;
```

NOTE:

1. RangexTheoreticalMaximum is every range's theoretical maximum of the measurement.
2. RawData is the raw pixel value of the sensor.

14 Firmware Version(0xEFFE)

The version of TOF module, like "DCAM10_20190211".

7 Software Example and Reference Code

7.1 Normal Mode Initialization Example

Initial device need to set some device configures, like normal mode. So the initial method will use a command sequence, like the below example.

If you want to init TOF module working in：

the normal mode,

outmode is depth mode,

range is near,

frame rate is 30FPS,

you can configure like this:

```
init_normal_mode_setting[] = {
  {ADI_CAMERA_MODULE_REG_TYPE_DATA,0x801e, 0x0002},  //mipi lane number
  {ADI_CAMERA_MODULE_REG_TYPE_TIMEOUT, 0x0000, 1},   //command time delay
  {ADI_CAMERA_MODULE_REG_TYPE_DATA,0x8004, 0x0001},  //depth 30 mode
  {ADI_CAMERA_MODULE_REG_TYPE_TIMEOUT, 0x0000, 1},   //command time delay
  {ADI_CAMERA_MODULE_REG_TYPE_DATA,0x8006, 0x0000},  //range a 0 - near
  {ADI_CAMERA_MODULE_REG_TYPE_TIMEOUT, 0x0000, 1},   //command time delay

    /*if you want to modify rangeA pulsecount and threhold default value
    you can add rangeA pulsecount gamma and threhold configure value here*/

  {ADI_CAMERA_MODULE_REG_TYPE_DATA,0x8005, 0x0001},  //work mode:normal mode

  {ADI_CAMERA_MODULE_REG_TYPE_TIMEOUT, 0x0000, 1},   //command time delay
  {ADI_CAMERA_MODULE_REG_TYPE_DATA,0x8017, 0x0001e}, //0x1e 30FPS Frame Rate
0xa 10FPS
  {ADI_CAMERA_MODULE_REG_TYPE_TIMEOUT, 0x0000, 5},   //command time delay
};
```

NOTE:

1. There is a minimum delay of one millisecond between each instruction in the initialization queue.
2. It is best to set frame rate with a delay of at least 5 ms.

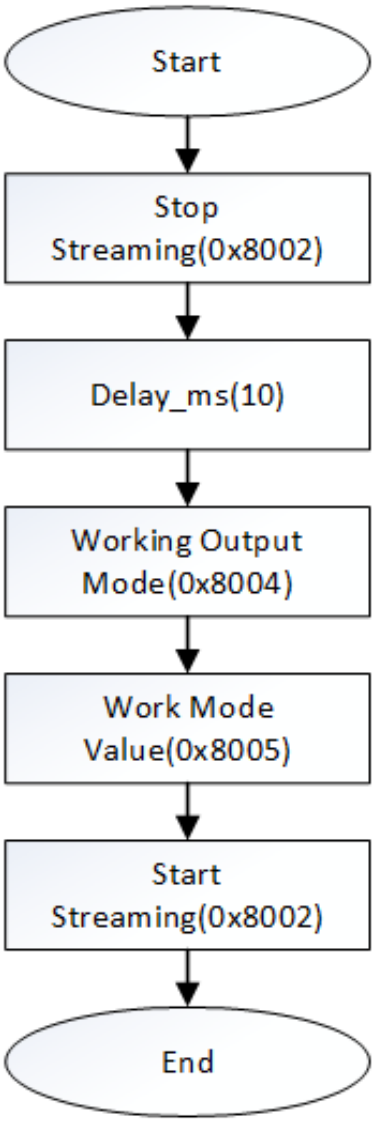
7.2 Setting Command Example

7.2.1 Working Mode Operation

TOF module have three kinds of output mode, depth output mode(640x480), IR output mode(640x480) and Depth&IR output mode(640x960).

TOF OUTMODE		
Index	Mode	Resolution
0	Detph	640*480
1	IR	640*480
2	Depth&IR	640*960

Set outmode as follows:



Refer to the command queue:

```
{0x8002, 0x0002}, //stop streaming
{0x8004, 0x0001}, //0x01 :depth outmode 0x02:ir outmode 0x03 :depth&IR mode
{0x8005, 0x0001}, //work mode: normal mode
{0x8002, 0x0004}, //start streaming
```

NOTE:

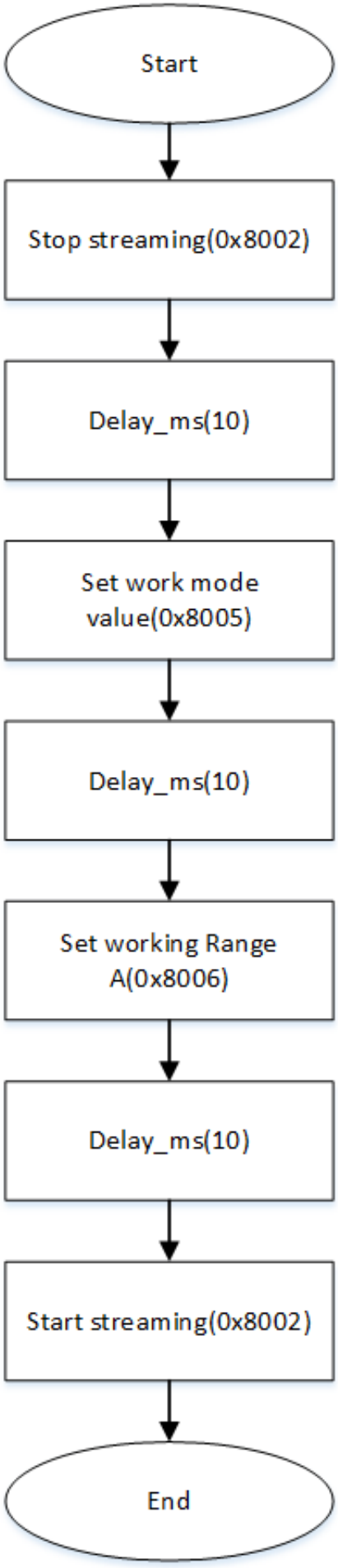
1. A delay must be added after stopping the streaming, 10 ms is recommend.
2. Other instructions may add 10 ms latency, not necessarily.

7.2.2 Range Operation

TOF module has nine kinds of range, range0~range8.

Index	Range Type
0	Range 0
1	Range 1
2	Range 2
3	Range 3
4	Range 4
5	Range 5
6	Range 6
7	Range 7
8	Range 8

Depending on your needs, configure TOF module in which range you need. Set range as follows:



Refer to the command queue:

```
{0x8002, 0x0002}, //stop streaming
{0x8005, 0x0001}, //work mode :normal mode
{0x8006, 0x0001}, //working rangeA, 1:range1 2: range2  ......
{0x8002, 0x0004}, //start streaming
```

NOTE:

1. A delay must be added after stopping the streaming, 10 ms is recommend.
2. Other instructions may add 10 ms latency, not necessarily.

7.2.3 Gamma Gain Operation

Set normal mode gamma gamma gain

Refer to the command queue:

```
{0x800b,0x00c8}, //set normal mode gamma gain 200
```

7.3 Example C code

Depth process code

```
#define PICTURESIZE 460800
uint8_t raw[460800] = {0x0};
uint16_t temp1,temp2,temp3,temp4;
for (i = 0; i < PICTURESIZE - 1; i = i + 3)
{
    temp1 = dst[i];
    temp2 = dst[i + 1];
    temp3 = (dst[i + 2] & 0xf0) >> 4;
    temp4 = dst[i + 2] & 0x0f;
    raw[j] = temp1 << 4 | temp4;
    raw[j + 1] = temp2 << 4 | temp3;
    j = j + 2;
}
```

IR process code

```
#define PICTURESIZE 460800
uint8_t raw[460800] = {0x0};
uint16_t temp1,temp2,temp3,temp4;
for(i = 0; i < PICTURESIZE - 1; i = i + 3)
{
    temp1 = dst[i];
    temp2 = dst[i + 1];
    temp3 = (dst[i + 2] & 0xf0) >> 4;
    temp4 = dst[i + 2] & 0x0f;
    raw[j] = temp1 << 4 | temp4;
    raw[j + 1] = temp2 <<4 | temp3;
    j = j + 2;
}
```

Depth&IR process code

```
#define PICTURESIZE 7372800
uint16_t raw[614400]       = {0x0};
uint16_t depht_raw[307200] = {0x0};
uint16_t ir_raw[307200]    = {0x0};
uint16_t temp1,temp2,temp3,temp4;

//based on mipi raw data format translate raw to active pixcel data
for(i = 0; i < PICTURESIZE-1;i = i+3)
{
    temp1  = dst[i];
    temp2  = dst[i + 1];
    temp3  = (dst[i + 2] & 0xf0) >> 4;
    temp4  = dst[i + 2] & 0x0f;
    raw[j] = temp1 << 4 | temp4;
    raw[j + 1] = temp2 << 4 | temp3;
    j = j + 2;
}

//separate depth and IR data
for(j = 0; j < 960; j = j + 1)
{
    //depth data
    if(0 == (j % 2))
    {
       for(i = 0; i < 480; i = i + 1)
       {
            depth_raw[i + (j / 2) * 480] = raw[i + 480 * j];
       }
    }
    //IR data
    else
    {
```

```
        for(i = 0;i < 480;i = i + 1)
        {
             ir_raw[i + ((j - 1) / 2) * 480] = raw[ i + 480 * j];
        }
    }
}
```