Confidential

ToF Box Library
API specification

-ver 1.2.0-

SK hynix

CIS biz. / ISP group / Algorithm

Contents

Revision History

Rev No.	Date	Comments
1.2.0	October 5, 2021	First Version is released.

1. Overview

1.1 ToF Box Architecture

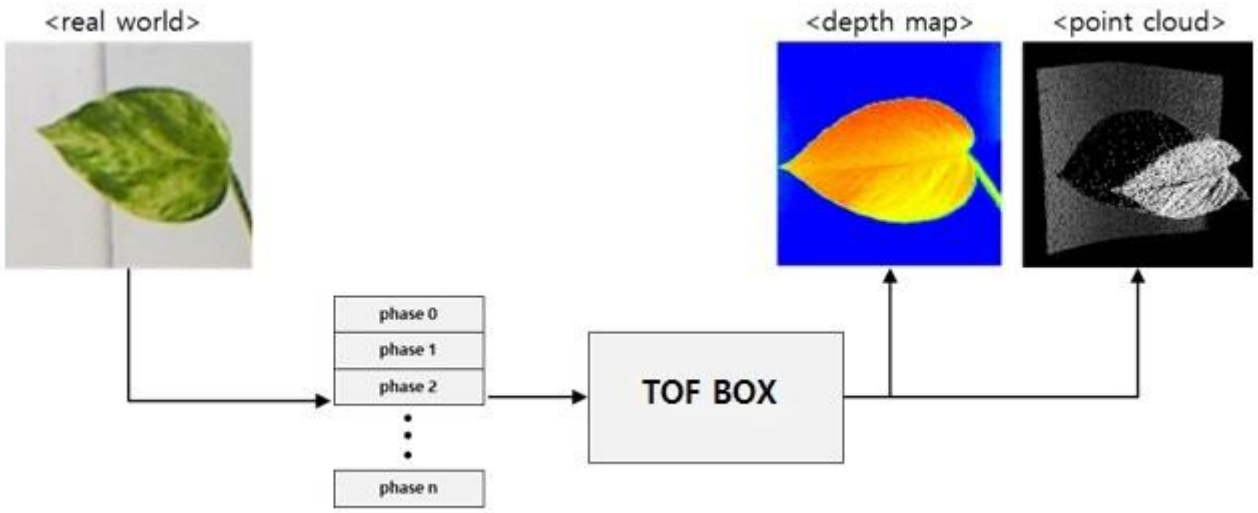


Figure 1. ToF Box Architecture

1.2 Abstract

ToF sensor detects the real world image with n phases. And ToF Box converts the phase data from ToF Sensor to the depth map that is a image that contains information of each pixel`s depth or point cloud data that a collection of 3D coordinates that represented object like **Figure 1** image.

2. Function Specification

Table 1. List of Functions

Function	Description
tof_ver	Return ToF Box version information
tof_init	Initializes library ・ Load and verify the tuning parameters ・ Initialize each function block
tof_proc	Process library ・ Process data
tof_deinit	De-initializes library

2.1 Function tof_ver

Syntax

```
tof_ver_t tof_ver(void);
```

Return Value

The tof_ver functions return version information structure. For detailed “tof_ver_t” structure, please refer to **Structure 1. ToF Box version**

Remarks

Gets current version of ToF Box library.

2.1.1 Structure tof_ver_t

```
typedef struct _tof_ver_t {
    uint32_t  major;
    uint32_t  minor1st;
    uint32_t  minor2nd;
} tof_ver_t;
```

Structure 1. ToF Box version

Members

major

If the main algorithm is changed, major number should be changed.

minor1st

If the functions are modified or added, minor version (1st) number should be changed.

minor2nd

If the wrong word or variable name is modified, minor version (2nd) number should be changed.

2.2 Function tof_init

Syntax

```
int32_t tof_init (
    uint16_t *cal_data,
    uint16_t *work,
    int32_t width_max,
    int32_t height_max
);
```

Parameters

**cal_data*

ToF sensor module calibration data. (256byte)

Temperature calibration data	16Byte
Wiggling calibration data	40Byte
Offset calibration data	96Byte
Camera calibration data	88Byte
Reference temperature	16Byte

**work*

working data buffer

size : 64 * width * height * sizeof(uint16_t)

same buffer pointer with i_param.work

width_max

ToF sensor module max width.

height_max

ToF sensor module max height.

Return Value

Returns 0 after ToF Box library is initialized successfully.

Returns error code (non-zero) if an error is occurred while initialized. For more information about error codes, please refer to **Table 2. Error values**.

Remarks

Loads calibration data set from read to OTP or EEPROM or File Data. This function initializes each block of algorithms to operate successfully.

Bit	Error List	Description
[0]	D_ERR_UNKNOWN	Unexpected error.
[1]	D_ERR_WIDTH	Invalid width (under 0)
[2]	D_ERR_HEIGHT	Invalid height (under 0)
[3]	D_ERR_ENABLE	Invalid enable option (under 0)
[4]	D_ERR_PACKED_BUFFER	Packed buffer allocation failed (NULL)
[5]	D_ERR_SRC_BUFFER	Source buffer allocation failed (NULL)
[6]	D_ERR_WORK_BUFFER	Work buffer allocation failed (NULL)
[7]	D_ERR_D_DST_BUFFER	Destination depth map buffer allocation failed (NULL)
[8]	D_ERR_PC_DST_BUFFER	Destination point cloud buffer allocation failed (NULL)
[9]	D_ERR_EDL_BUFFER	Embedded data line buffer allocation failed (NULL)
[10]	D_ERR_CAL_BUFFER	Calibration data buffer allocation failed (NULL)

Table 2. Error values

2.3 Function tof_proc

Syntax

```
int32_t tof_proc (
    tof_proc_t *info
);
```

Parameters

Tof_proc_t * info

Set of environment variables for processing the ToF Box. For detailed “tof_proc_t” structure, refer to **Structure 2. ToF Process**

Return Value

Returns 0 after ToF Box library process successfully.

Returns error code (non-zero) if an error is occurred while initialized. For more information about error codes, please refer to **Table 2. Error values**

Remarks

Outputs depth or Point Cloud data through process. For detailed “point_cloud_t” structure, refer to **Structure 3. Point Cloud**

2.3.1 Structure tof_proc_t

```
typedef struct _tof_proc_t
{
	int32_t		width;
	int32_t		height;
	int32_t		enable;
	float		cur_vcsel_temp;
	float		cur_sensor_temp;
	int32_t		int_time_current_hex;
	int32_t		*int_time_output_hex;
	uint16_t	*input;
	uint16_t	*input_packed;
	uint16_t	*edl_input;
	uint16_t	*work;
	uint16_t	*output_d;
	point_cloud_t	*output_pc;
} tof_proc_t;
```

Structure 2. ToF Process

Members

width

Horizontal image size of sensor.

height

Vertical image size of sensor.

enable

Sensor operation mode

cur_vcsel_temp

current vcsel temperature

cur_sensor_temp

current sensor temperature

int_time_current_hex

current interval time (hexadecimal)

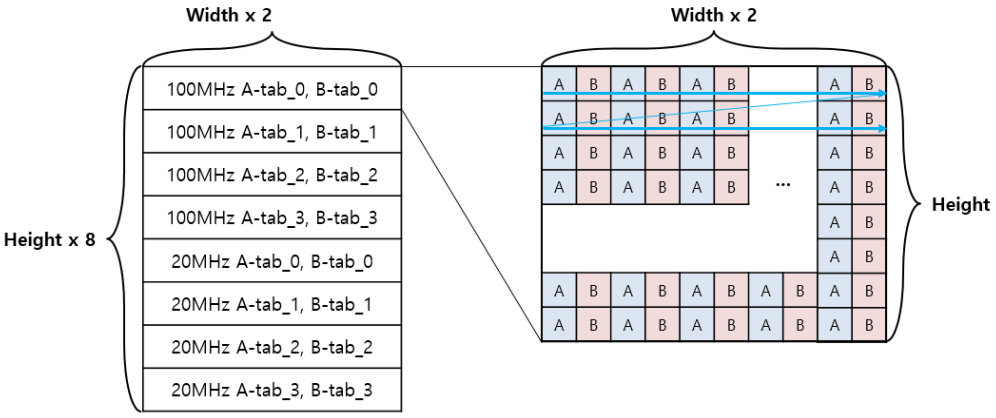
**int_time_output_hdx*

output interval time pointer (hexadecimal)

**input*

input image data buffer.

size : 16 * width * height * sizeof(uint16_t)



**input_packed*

If unpack block enable, user should assign input image buffer to *this pointer variable*.

**edl_input*

input edl(embedded data line) data buffer

size : 8 * 16 * sizeof(uint16_t)

**work*

working data buffer

size : 64 * width * height * sizeof(uint16_t)

**output_d*

Output buffer in depth format

size : width * height * sizeof(uint16_t)

**output_pc*

Output buffer in Point Cloud format

size : width * height * sizeof(point_cloud_t)

For detailed Point Cloud format, refer to **Structure 3. Point Cloud**

2.3.2 Structure point_cloud_t

```
typedef struct _point_cloud_t
{
	float  x;
	float  y;
	float  z;
} point_cloud_t;
```

Structure 3. Point Cloud

Point Cloud buffer is XYZ formatted structure

[0]			[1]			[2]			[3]		
x	y	z	x	y	z	x	y	z	x	y	Z

2.4 Function tof_deinit

Syntax

```
void tof_deinit(void);
```

Remarks

This function terminate ToF Box library normally

2.5 Sample Code

Example

```
// This program ToF Box library sample code.
#include “skhynix_tof_lib.h”

int32_t tof_sample(
   uint16_t *src,                      // source buffer pointer
   uint16_t *output_depth,             // destination depth buffer pointer
   point_cloud_t *output_point_cloud,  // destination point cloud buffer pointer
   uint16_t *cal_buf,                  // calibration buffer pointer
   uint16_t *edl_data,                 // edl data
   int32_t width,                      // horizontal image size of sensor
   int32_t height,                     // vertical image size of sensor
   float cur_vcsel_temp,               // current vcsel temperature
   float cur_sensor_temp,              // current sensor temperature
```

```
    int32_t int_time_current,          // current interval time
    int32_t *int_time_output           // output interval time
)
{
  int32_t ret = 0;

  // get ToF Box version
  tof_ver_t ver = tof_ver();
  printf("tof_box ver : %d.%d.%d\n", ver.major, ver.minor1st, ver.minor2nd);

  uint16_t *work;
  work = (uint16_t*)malloc(width * height * 64 * sizeof(uint16_t));

  // set tof_box initialize
  ret = tof_init(cal_buf, work, width, height);

  if (ret != 0) {
    printf("tof_box initialize fail. [0x%01x]\n", ret);
    return ret;
  }

  // ToF Box param
  tof_proc_t tof_param;
  tof_param.width = width;
  tof_param.height = height;
  tof_param.enable = 0x1AE4;
  tof_param.int_time_current_hex = int_time_current;
  tof_param.int_time_output_hex = int_time_output;

  tof_param.edl_input = edl_data;
  tof_param.cur_sensor_temp = 0;
  tof_param.cur_vcsel_temp = 0;

  tof_param.packed = NULL;
  tof_param.src = src
  tof_param.output_d = output_depth;
  tof_param.output_pc = output_point_cloud;
```

```
  ret = tof_proc(&tof_param);

  if (ret != 0) {
    printf("tof_box process fail. [0x%01x]\n", ret);
    return ret;
  }

  tof_deinit();

  free(work);

  return ret;
}
```