對外秘

Time-of-Flight Sensor

Calibration Guide

Revision 2.3.3

Dec. 13, 2019

SAMSUNG ELECTRONICS

Beyond You Can Imagine

對外秘

Important Notice

Samsung Electronics Co. Ltd. (“Samsung”) reserves the right to make changes to the information in this publication at any time without prior notice. All information provided is for reference purpose only. Samsung assumes no responsibility for possible errors or omissions, or for any consequences resulting from the use of the information contained herein.

This publication on its own does not convey any license, either express or implied, relating to any Samsung and/or third-party products, under the intellectual property rights of Samsung and/or any third parties.

Samsung makes no warranty, representation, or guarantee regarding the suitability of its products for any particular purpose, nor does Samsung assume any liability arising out of the application or use of any product or circuit and specifically disclaims any and all liability, including without limitation any consequential or incidental damages.

Customers are responsible for their own products and applications. "Typical" parameters can and do vary in different applications. All operating parameters, including "Typicals" must be validated for each customer application by the customer's technical experts.

Samsung products are not designed, intended, or authorized for use in applications intended to support or sustain life, or for any other application in which the failure of the Samsung product could reasonably be expected to create a situation where personal injury or death may occur. Customers acknowledge and agree that they are solely responsible to meet all other legal and regulatory requirements regarding their applications using Samsung products notwithstanding any information provided in this publication. Customer shall indemnify and hold Samsung and its officers, employees, subsidiaries, affiliates, and distributors harmless against all claims, costs, damages, expenses, and reasonable attorney fees arising out of, either directly or indirectly, any claim (including but not limited to personal injury or death) that may be associated with such unintended, unauthorized and/or illegal use.

Samsung Electronics Co., Ltd.
San #24 Nongseo-Dong, Giheung-Gu
Yongin-City, Gyeonggi-Do, Korea 446-711

Contact Us: *kh74.bae@samsung.com*

Home Page: *http://www.samsungsemi.com*

Revision History

Revision No.	Date	Detailed of Modification	Drafter
1.0.0	Mar. 30, 2019	Initial Calibration Guide	K. Bae, H. Ko, D. Seo, M. Kang, C. Lee, M. Bae, S. Son, M. Keel
2.0.0	Jun. 10, 2019	Changed the calibration methods 1) Temperature drift calibration independent on wiggling and FPPN 2) Wiggling calibration uses a checkerboard chart which is the same equipment in Lens-FPPN. Removed the time-delay board part which is mentioned in a separate document.	K. Bae, H. Ko, D. Seo, C. Lee
2.1.0	Jul. 24, 2019	Changed the calibration methods 1) Temperature drift calibration uses wiggling	K. Bae, H. Ko, C. Lee
2.1.1	Jul. 31, 2019	Added API functions in Temperature Drift Library	C. Lee
2.2.0	Aug. 02, 2019	Added board calibration methods Modified calibration configuration XML	K. Bae, H. Ko
2.2.1	Aug. 19, 2019	Update Temperature drift and Calibration library	K. Bae, H. Ko
2.2.4	Sep. 05, 2019	Update Temperature drift and Calibration 1) APIs 2) Examples	H. Ko
2.3.0	Oct.24, 2019	Update board calibration methods 1) Board calibration using plain chart	K. Bae, H. Ko
2.3.1	Oct.29, 2019	Update wiggling calibration method 1) Remove temperature compensation	K. Bae, H. Ko
2.3.3	Dec.06, 2019	Update XML figure Added API function for checking error codes	H. Ko, C.Lee

Contents

List of Figures

1 Overview

A system calibration is a process to reduce the effect of the systematic errors in a ToF system, as shown in Figure 1.1. This document aims to present calibration method, equipment, and software.

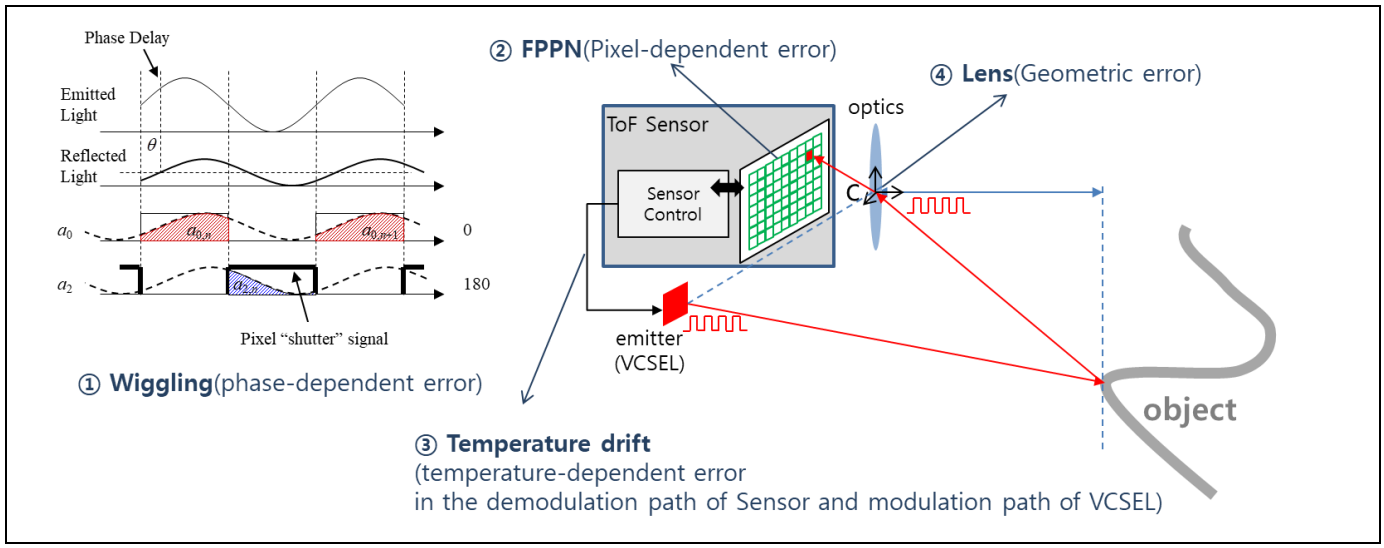

Figure 1.1 Systematic Error Sources

1.1 Systematic Errors

1.1.1 Wiggling

Phase (distance)-dependent error due to harmonic distortion.

1.1.2 Fixed Phase Pattern Noise (FPPN)

Pixel-dependent error due to the time-delay of demodulation signal depending on individual pixel position and the misalignment of a VCSEL and a sensor (which is geometric error).

1.1.3 Temperature Drift

Temperature-dependent error because the time-delay with respect to temperature can be changed in the demodulation path of a sensor and in the modulation path of a VCSEL.

1.1.4 **Lens**

Geometric error due to lens distortion, shifts, and tilts.

1.2 **Overall Calibration Procedure**

There are two types of ToF calibration

- One-time calibration: temperature drift calibration is typically performed once. All modules can use the same parameters.
- Module to Module (M2M) calibration: wiggling, lens, and FPPN have per-module variation. Each module must be calibrated.

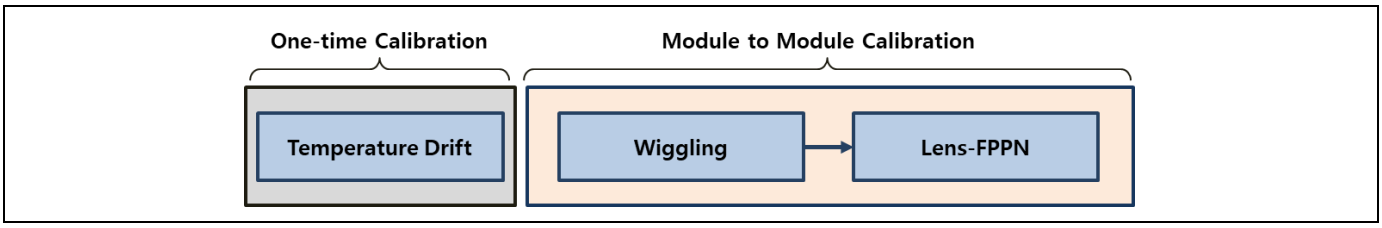

Figure 1.2 Overall Calibration Procedure

One-time calibration

1) Obtain the reference depth using the Temperature Drift Tool.
2) Obtain the statistics of temperature drift depending on the time and ambient temperature using the Temperature Drift Tool.
3) Calculate the coefficients using the statistics.
4) Two outputs; ① data for depth ISP ② data for M2M calibration.

M2M Calibration

: During the M2M calibration, the temperature in a sensor and a VCSEL driver will change over time. Temperature drift compensation is performed at each calibration.

1) Calibrate wiggling **with the external time-delay.**
2) Calibrate lens-FPPN **without the external time-delay (normal mode).**

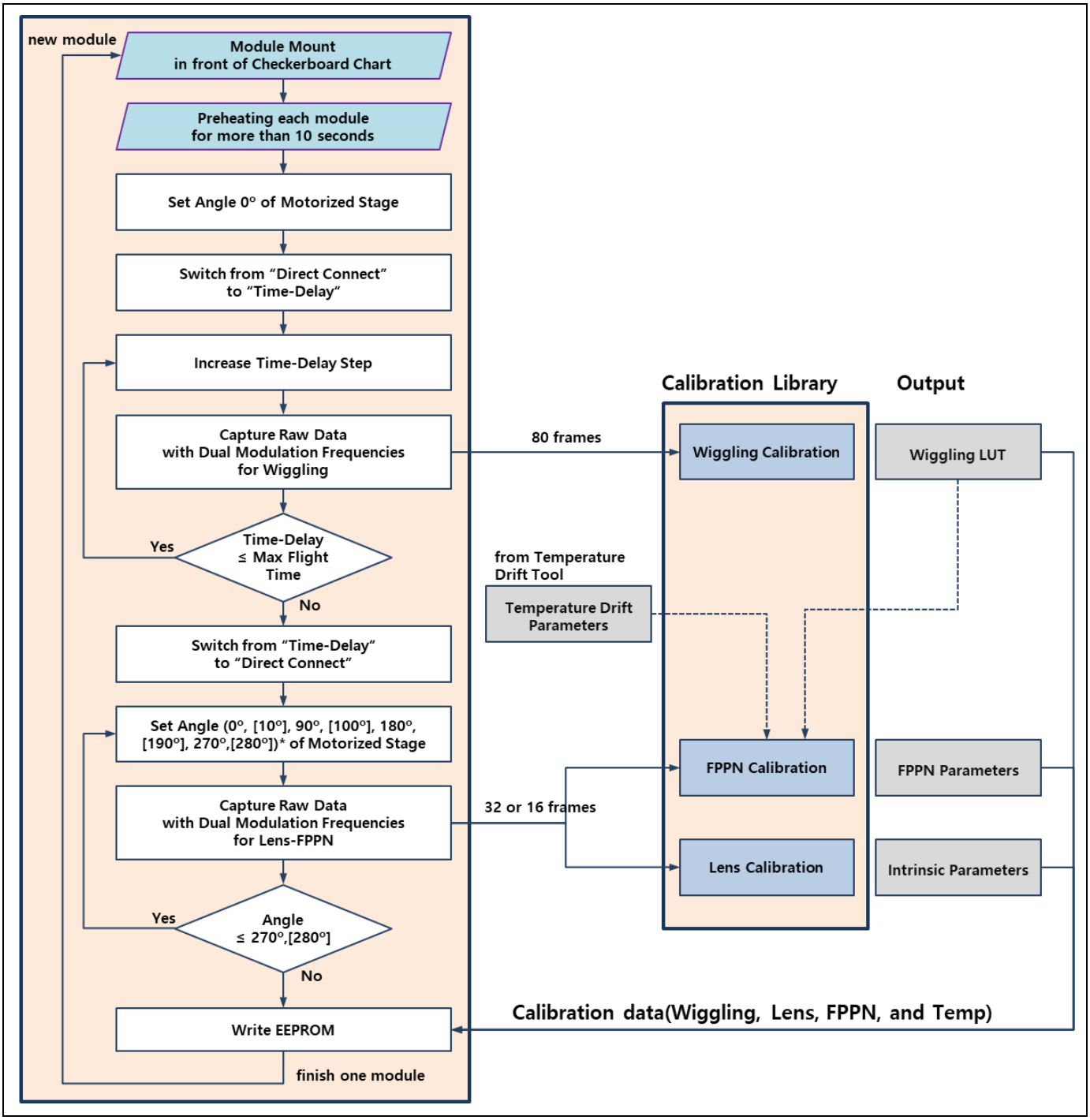

Figure 1.3 M2M Calibration Procedure

Table 1.1 M2M Calibration Processing Time (conditions: dual modulation frequencies; 100MHz & 80MHz, PC Spec.; i7 3.6GHz, RAM 16GB, frame grabber; Simmian MV3 board).

Calibration items	preheating time (sec)[1)]	data capturing time (sec)	data loading and processing time (sec)	Num of data (depth/frame)[4)]
wiggling	10	3.5 [2)]	1.5	20/80
lens-FPPN	-	40 [3)]	7.8	40/160
total	10	43.5	9.3	28/112
	62.8			

1) Preheating time can be reduced depending on the temperature drift performance.
2) Wiggling data capturing time measured with Samsung's Wiggling Calibration Board and Simmian board.
3) Lens-FPPN data capturing time depends on the speed of rotation and stabilization of motorized stage. 4 different views of a plane chart are used.
4) One depth requires 4 raw frames.

Figure 1.4 shows the compensation procedure using calibration data. Figure 1.5 represents the example of overall compensation procedure.

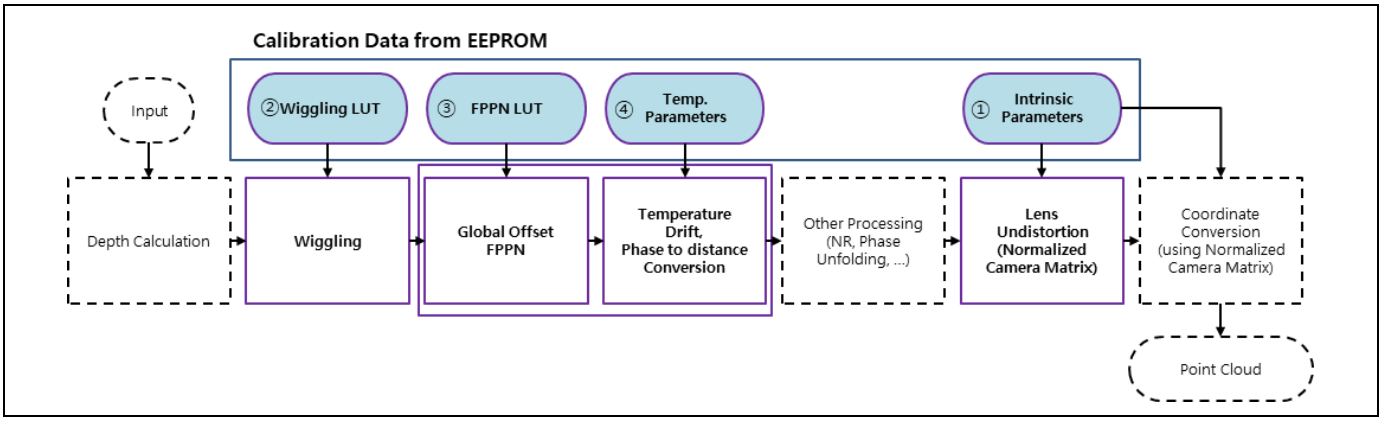

Figure 1.4 Overall Compensation Procedure

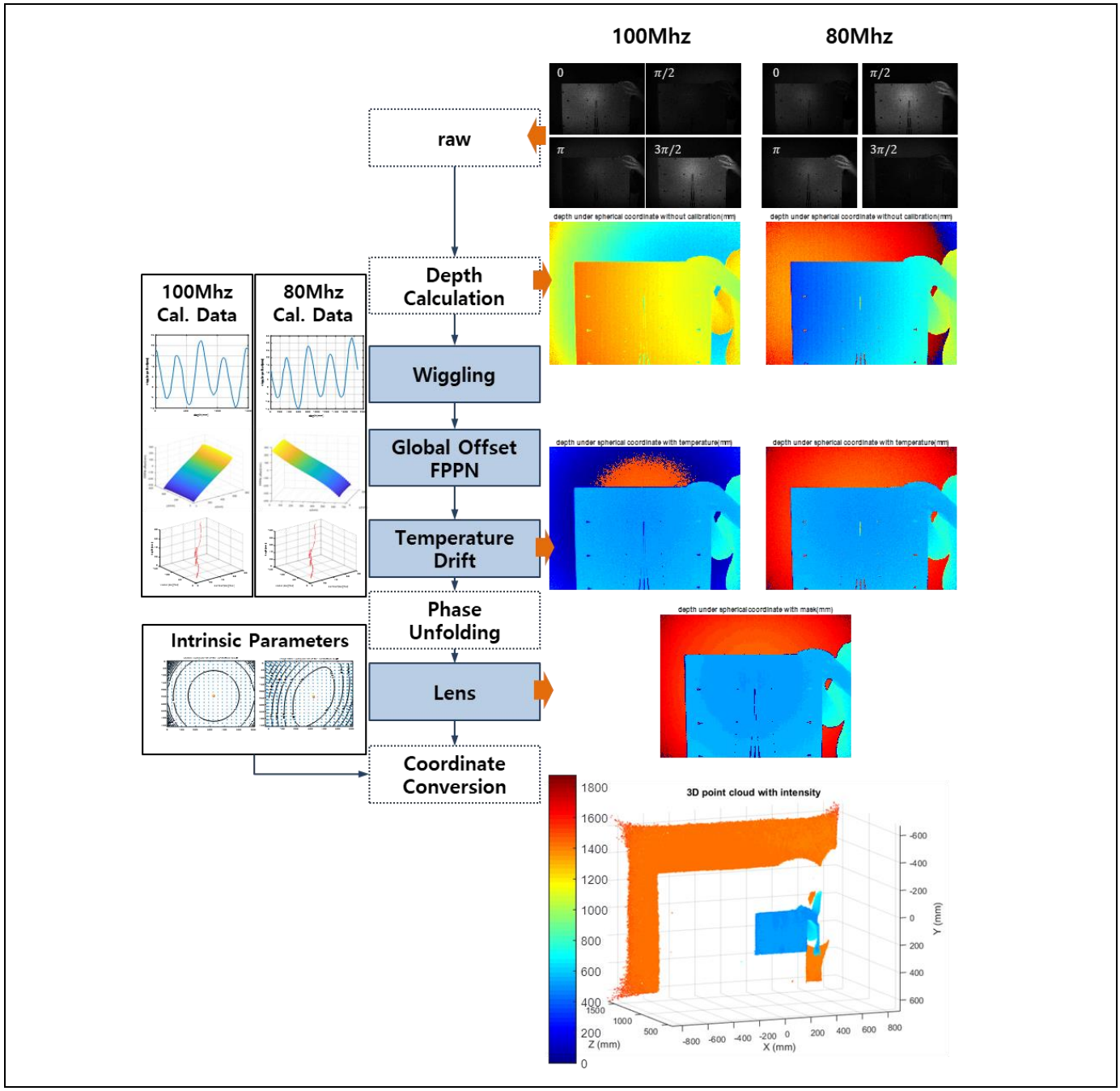

Figure 1.5 Example of Overall Compensation Procedure

2 Calibration

2.1 Temperature Drift

Temperature can change the depth offset, because the delay time in the VCSEL modulation and sensor demodulation path can be altered with temperature. Therefore, the distance offset compensation with temperature drift should be applied to the original distance, based on the temperatures in a sensor and a laser driver (here, we assume temperature of the VCSEL is represented by a laser driver's temperature). Compensation offset can be modeled as follows:

$$compensation\ offset = f(x, y)$$

where $x = temp_{sensor}$ and $y = temp_{vcsel}$ are the temperature data of a sensor and a laser driver, respectively, which can be obtained in the embedded line of sensor raw data (see more detailed information from the application note of the ToF sensor). $f(x, y)$ is a non-linear function. The coefficients of a function are calculated using temperature drift tool with respect to the temperature data. Temperature drift calibration is typically performed only one time from mass data, but not in M2M.

Temperature drift statistics can be obtained by "Temperature Drift Tool" depending on time and ambient temperature. Ambient temperature may be changed while obtaining statistics. "Temperature Drift Tool" calculates the coefficients from these statistics.

2.1.1 Procedure

Temperature Drift Calibration Procedure (see Figure 2.1)

1) Select a camera module for temperature drift calibration or golden samples.

   a) Wiggling Calibration

2) Set EIT with maximum value.

3) Set ambient temperature as 28℃

   a) Obtain the temperature statistics for about 10 minutes at 28℃.

      : If depth drift is stable, "Temperature Drift Tool" can stop and obtain the reference depth.

4) Increase sequentially ambient temperatures from 5℃ to 70℃ with 5℃ step.

   Decrease sequentially ambient temperatures from 70℃ to 5℃ with 5℃ step.

   a) Obtain the temperature statistics at each ambient temperature.

      : "Temperature Drift Tool" can obtain the first statistics of temperature drift with max EIT.

5) Set EIT with 15% of maximum value.

6) Increase sequentially ambient temperatures from 5℃ to 70℃ with 5℃ step.

Decrease sequentially ambient temperatures from 70℃ to 5℃ with 5℃ step.

a) Obtain the second temperature statistics at each ambient temperature with 15% EIT.

: “Temperature Drift Tool” can obtain the statistics of temperature drift.

7) Calculate coefficients using “Temperature Drift Tool”.
Obtain two sets; one set for EEPROM data and one set for M2M calibration. EEPROM data will be written on the all camera modules.

※ Note that if there is no temperature drift file for M2M calibration we provide the default file for M2M calibration. But we don’t recommend it. It is used for only testing mode.

※ We will define the ambient temperatures for calibration.

※ When increase or decrese the ambient temperature, please wait for stabilizing the setting temperature.

※ If difference between modules, please use a golden module instead of many modules.

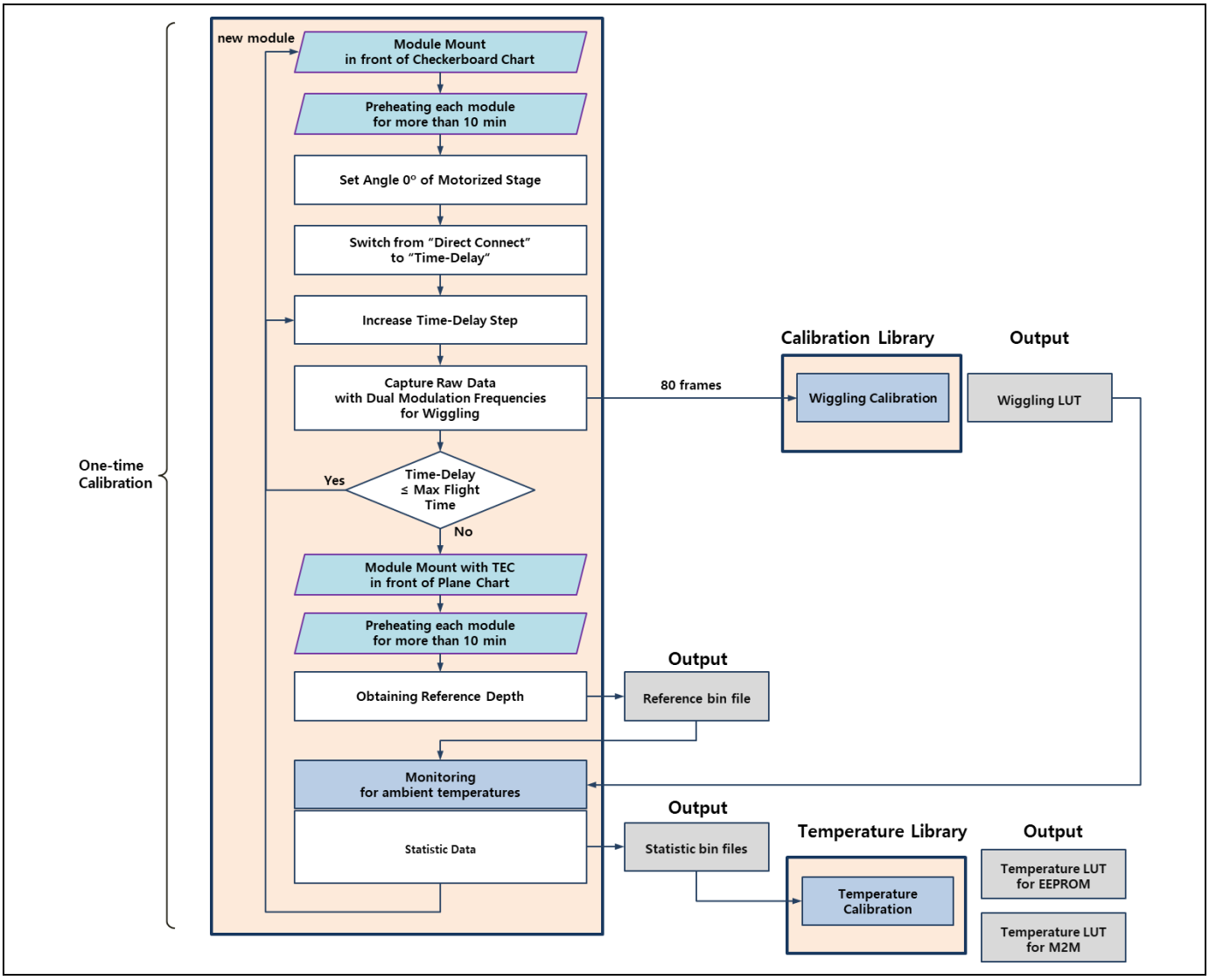

Figure 2.1 Temperature Drift Calibration Procedure

2.1.2 **Equipment**

For the temperature drift calibration, a plane chart whose reflectivity is higher than 80% is used in common. The distance from a module and a plane chart is fixed by monitoring the code to prevent saturation. This distance would be about 400 – 700 mm depending optical power of the VCSEL, integration time, and chart reflectivity.

Temperature drift calibration requires a sensor and a laser driver to be held at a specific stable temperature. For this purpose, a thermal chamber may be a candidate for adjusting temperature. However, size limitation of the thermal chamber can cause some problems. A small-size chamber will cause multiple reflections because a plane chart cannot be allowed to put into the chamber, and large-size chamber is costly and occupies huge space. If you have a large-size chamber you can use it.

We recommend a thermoelectric cooler (TEC) to regulate the calibration temperature. A TEC is a solid state device which can control heat flux using electric current; it is composed of a precision temperature controller and a base plate, as shown in Figure 2.2. Highly thermal conductive materials such as copper or aluminum will be used as the base plate for the chuck or thermal platform. And the TEC chip is directly connected to the base plate to control temperature fast and in bi-directional way. This allows for temperature settings above and below the ambient temperature. The TEC is particularly useful in small scale temperature control, providing fast temperature response and ultra-high temperature stability. Under low temperature test, it is important to create moisture free environment [1].

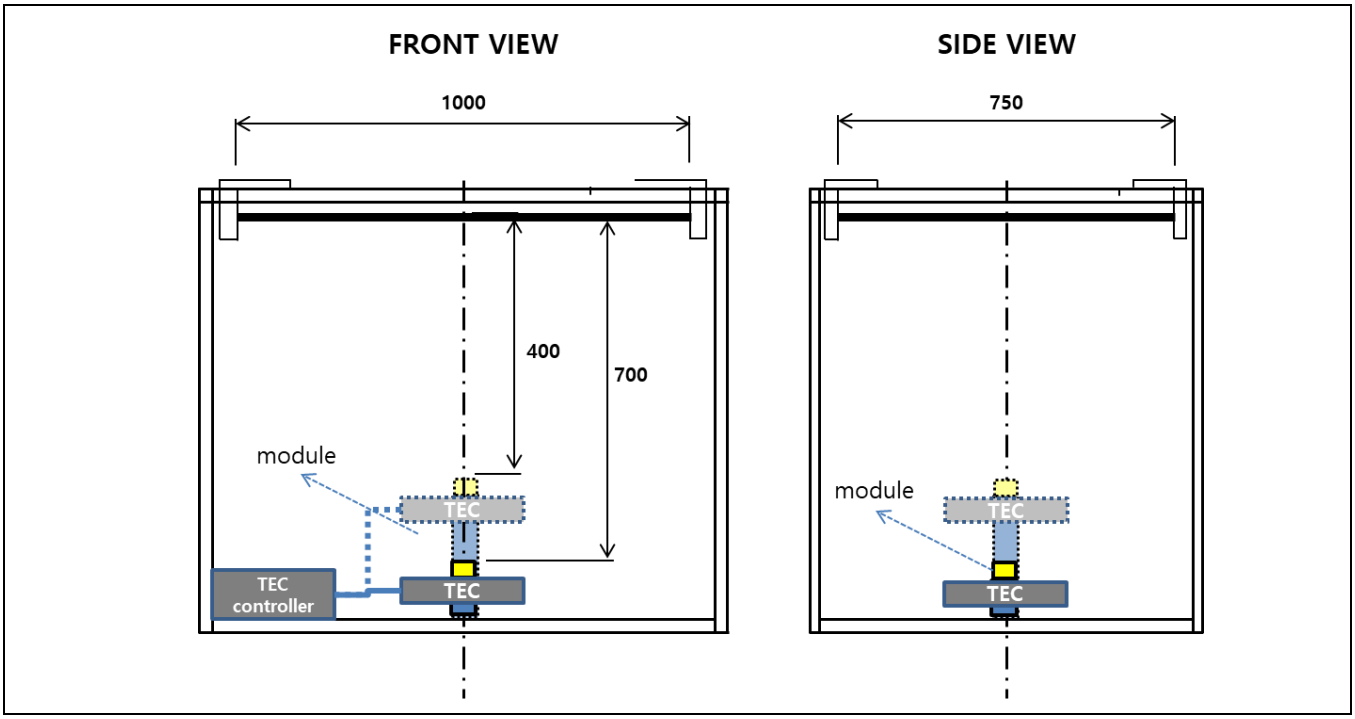

Figure 2.2 Example of temperature drift compensation equipment using a plane chart. Adjustable height stage mount for module (400~700mm) which is combined with TEC. Height will be fixed for a manufacturing environment.

When calibrating temperature drift, “Lens-FPPN & Temperature Drift Calibration Board” without the external time-delay as shown in Figure 2.3 has to be used.

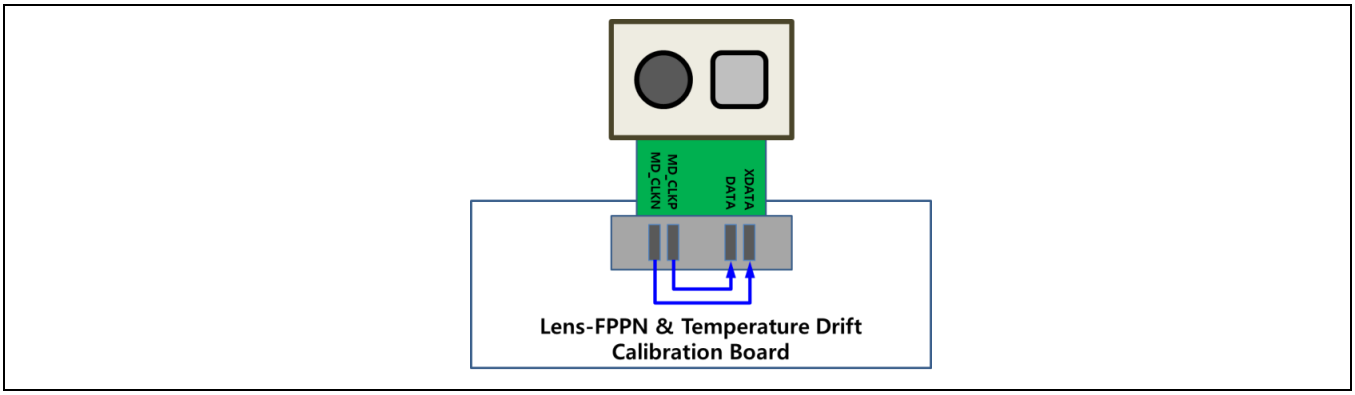

Figure 2.3 Temperature Calibration Board

2.2 Wiggling, Lens-FPPN

The wiggling error, also called as circular error, is a known systematic sinusoidal depth distortion. The error varies with the distance. Note that since the wiggling error is a nonlinear function of distance, in general, a look up table (LUT) is used; the residual error between the measured depth and the ground truth (GT) will be pre-calculated in the calibration procedure. In the conventional approach, the wiggling calibration is performed by a moving chart on a measurement track line. But in the mass production, this method cannot be used because of the space and capture time limitation, even though only the center region of interest (ROI) is used.

We will use an external time-delay board and a checkerboard chart to calibrate wiggling error. A checkerboard chart is the same as a chart for lens-FPPN. The procedure does not require moving chart on a measurement track line. Only a checkerboard chart in a fixed distance is sufficient. This method reduces calibration time and space for the mass production. It is possible to use only small data set. The time delay in the modulation signal for the VCSEL driver should be adjusted by an external time-delay board in order to add a phase shift between the modulation and the demodulation of the light signal. Delay in the modulation path is equivalent to a distance shift of the object at an optical center.

Note that the raw data of the charts obtained by an external time-delay board will be different from those obtained by a moving chart as shown in Figure 2.4. In case of the moving chart, the distance from the sensor and the chart increases gradually in z-direction on the Cartesian coordinates system of the sensor (see Figure 2.4 (a) and (b)). However, when using the time-delay circuit, the virtual distance from the sensor and the chart is consistently adjusted in r-direction on the Spherical coordinates system, but not in z-direction on the Cartesian coordinates system. Therefore, the resultant raw data looks like capturing curved surfaces as shown as in Figure 2.4 (b) and (d).

For the test of this method, the precision time-delay ICs such as NB6L295 from On-semiconductor® are used. The NB6L295 is a dual channel programmable delay chip [2]; the main specifications are 0 – 11.2 ns total time delay (extended mode), 11ps time delay per step, 511 step (9-bit control), linearity of +/-20ps (max.). The delay time can be digitally controlled by an MCU board such as Arduino. A single time step of the NB6L295 is equivalent to about 1.54 mm distance shift. Since the time-delay IC contains 2048 steps when used by two chips in series, the maximum delay represents a distance shift of about 3.1478m.

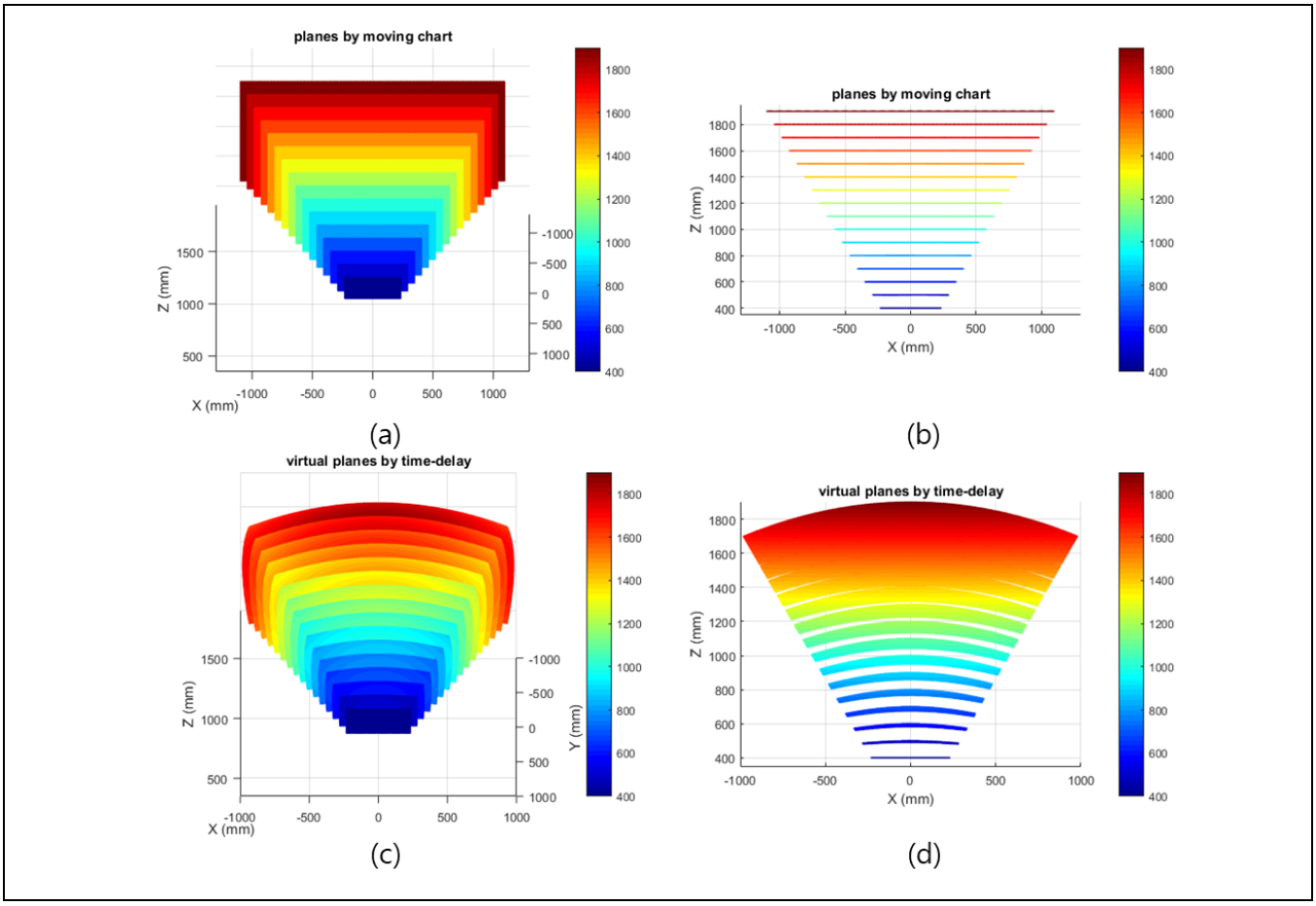

Figure 2.4 Comparison between a moving chart ((a) and (b)), and a virtual chart by time-delay ((c) and (d))

2.2.1 **Procedure**

Wiggling Calibration Procedure (see Figure 2.5)

1) Switch from “Direct Connect” to “Time-Delay” using Jumper or Relay Switch

2) Set a module to be able to capture a checkerboard chart

3) Preheat the module for more than 10 seconds.

4) Set the number of the steps to specify the number of images used in calibration (default, 20)

5) Raw data is captured automatically.

6) Calculate wiggling LUT using Calibration Library.

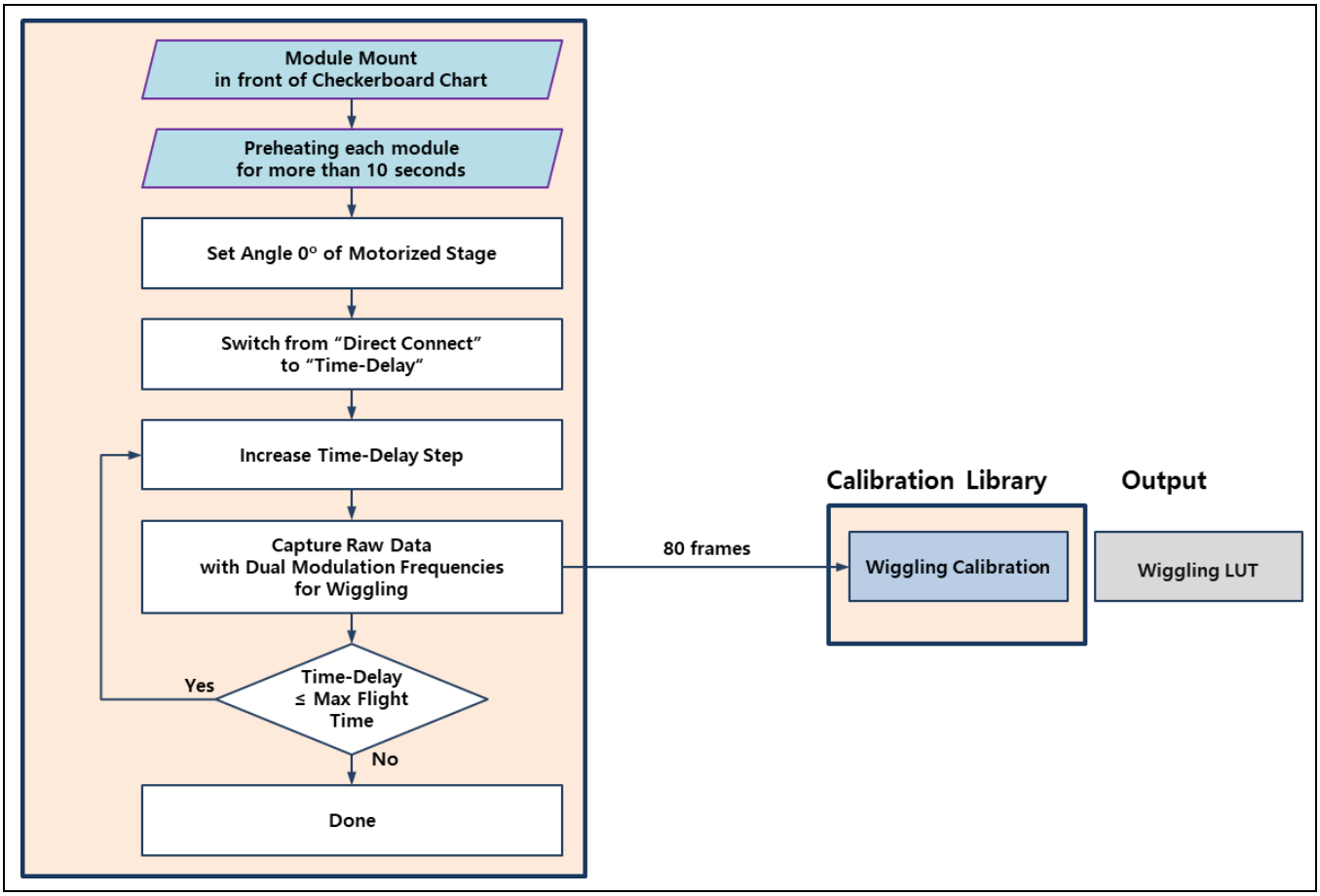

Figure 2.5 Wiggling Calibration Procedure

2.2.2 **Equipment**

For the wiggling calibration, a checkerboard chart can be used same as lens-FPPN. The reflectivity of white checker is more than 80%. The distance from a module and a plane chart is fixed by monitoring the code to prevent saturation. This distance is about 400 – 700 mm depending optical power of VCSEL, integration time, and chart reflectivity.

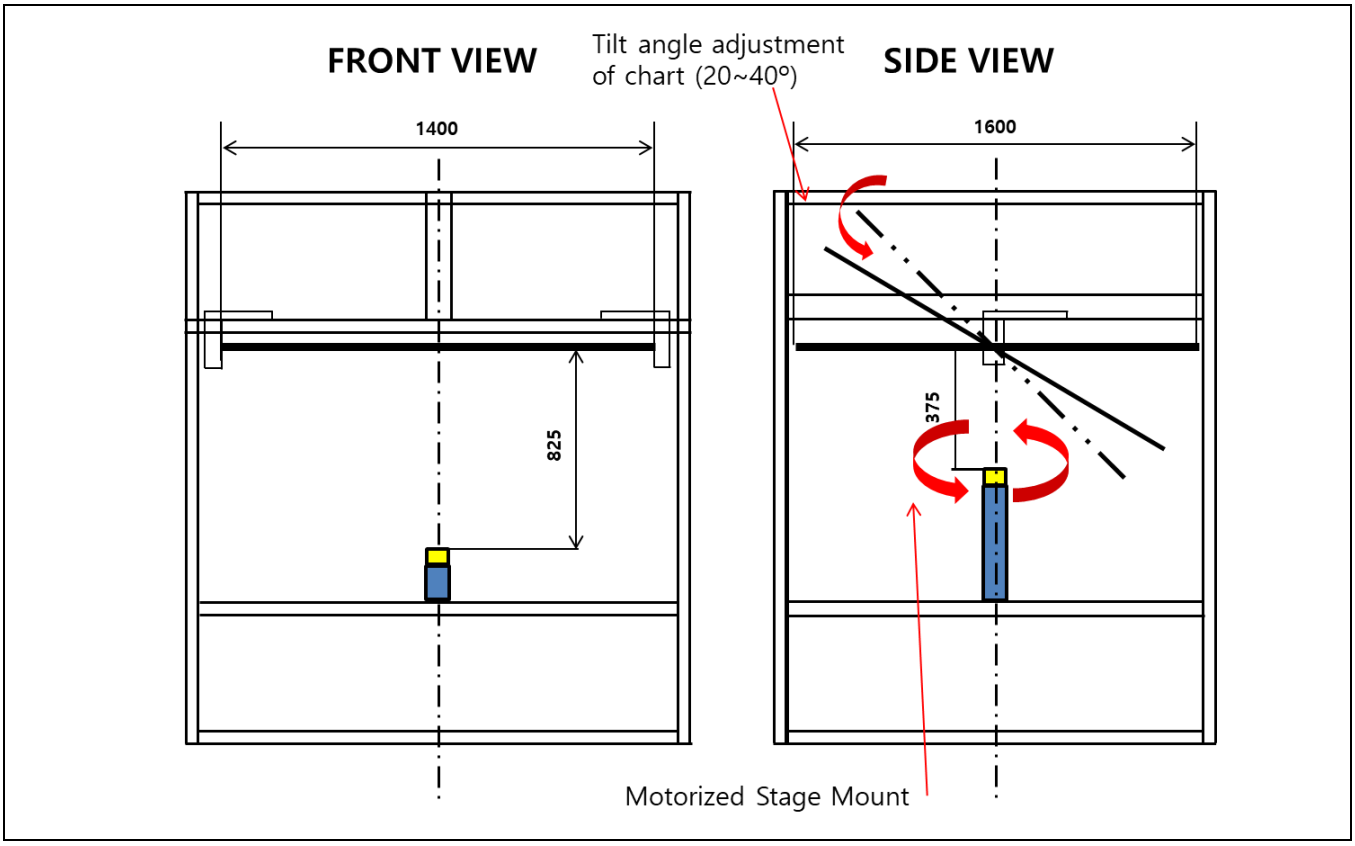

Figure 2.6 Wiggling and Lens-FPPN Calibration Chart. Height adjustable stage mount for module (375 – 825 mm) and an angle adjustable chart (20º – 40º). Height and angle will be fixed for a manufacturing environment.

Table 2.1 The Specification of Wiggling, Lens-FPPN Calibration Chart.

Chart type	Checkerboard
Chart size	1610 mm × 1410 mm
Material (lambertian reflectivity)	White > 80%, Dark > 10%
Square size	40 mm × 40 mm
Square count	40 x 35
Tilt angle adjustment	30º
Angle of Motorized Stage	0º or 180º for wiggling (default is 0º)
Distance from module	375 – 825 mm

When calibrating wiggling, “Wiggling Calibration Board” with the external time-delay as shown in Figure 2.7 has to be used.

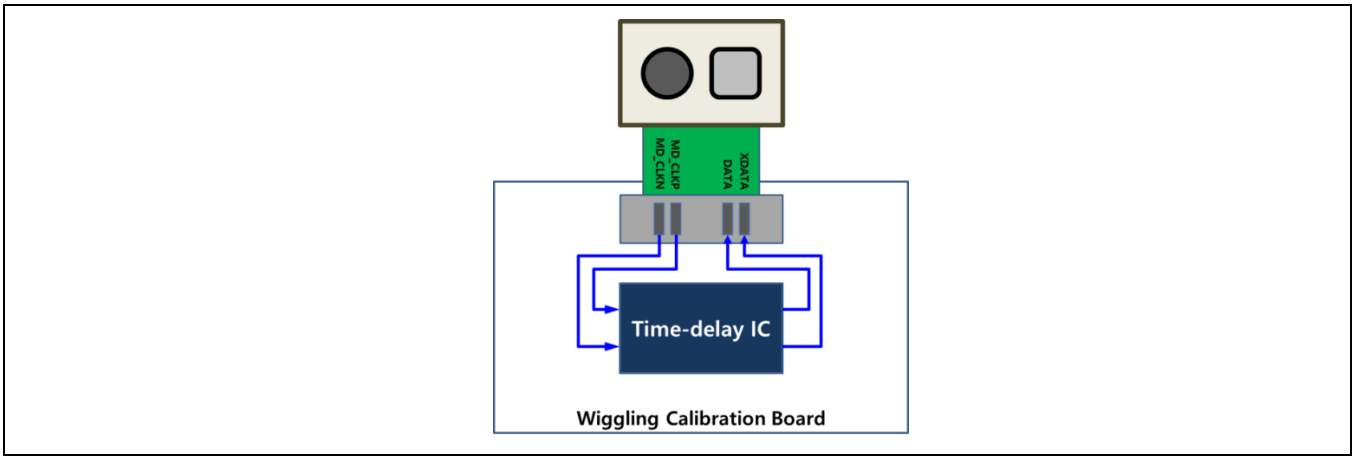

Figure 2.7 The Wiggling Calibration Board.

Precautions when using the Wiggling Calibration Board

The time-delay IC on the Wiggling Calibration Board has a delay-variation depending on the temperature. Time delay drift may occur over time, when the internal temperature of the ICs is stabilized. Therefore, sufficient warm-up time may be required when using the Wiggling Calibration Board. We recommend having a warm-up time of at least 15 minutes for stable results. This warm-up time varies with different board design, and is required only at the beginning of the equipment initialization; after the warm-up time, no additional warm-up is required for wiggling calibration process of each individual module.

2.3 Lens-FPPN

A geometric calibration uses the standard technique from Zhang [3]. The parameters are usually estimated through homography between 3D and 2D points. By theoretically using at least three different views of a planar checkerboard, the lateral intrinsic camera parameters (principal point, $\mathbf{cc}$, focal length, $\boldsymbol{f}_\mathbf{c}$ and lens distortions, $\mathbf{kc}$) are estimated. Lens-FPPN calibration uses at least four different views of a plane due to FPPN.

$\mathbf{K}$ is known as the camera matrix, and defined as follows:

$$\mathrm{K} = \begin{bmatrix} \boldsymbol{f}_\mathbf{c}(1) & 0 & \mathbf{cc}(1) \\ 0 & \boldsymbol{f}_\mathbf{c}(2) & \mathbf{cc}(2) \\ 0 & 0 & 1 \end{bmatrix}$$

Lens distortion uses the Brown's model as follows:

$$\mathbf{x_d} = \begin{bmatrix} \mathbf{x_d}(1) \\ \mathbf{x_d}(2) \end{bmatrix} = (1 + \mathbf{kc}(1)r^2 + \mathbf{kc}(2)r^4 + \mathbf{kc}(5)r^6)\mathbf{x_n} + \mathbf{dx}$$

where $\mathbf{dx}$ is the tangential distortion vector:

$$\mathbf{x_d} = \begin{bmatrix} 2\mathbf{kc}(1)xy + \mathbf{kc}(4)(r^2 + 2x^2) \\ \mathbf{kc}(3)(r^2 + 2y^2) + 2\mathbf{kc}(4)xy \end{bmatrix}$$

Therefore, the 5-vector $\mathbf{kc}$ contains both radial and tangential distortion coefficients.

Fixed pattern phase noise (FPPN) accounts for pixel-location-dependent phase deviations. The conventional approach is to estimate the FPPN with the aid of a plane and approximate the error with a polynomial fitting. Lens and FPPN calibration can be performed with the same equipment and environment simultaneously because extrinsic parameters of different views of a plane can be obtained in the lens calibration and ground truth (GT) of a plane can be calculated using extrinsic and intrinsic parameters.

2.3.1 **Procedure**

After completing wiggling calibration, lens calibration is performed. Therefore, intrinsic and extrinsic parameters can be obtained. Using extrinsic parameters, FPPN calibration can generate spatial depth offset LUT.

Lens-FPPN Calibration Procedure (as shown in Figure 2.8)

1) Switch from “Time-Delay” to “Direct Connect” using Jumper or Relay Switch

2) Set angles of checker chart as 0°, 90°, 180° and 270°. Additional 10°, 100°, 190° and 280° can be set. If 4 different views of a plane chart are used the number of data have to be increased after evaluating a module.

3) Obtain raw data images in each angle for dual modulation frequencies.

4) Lens calibration is performed using Calibration Library.

   a) Calculate intrinsic parameters for each module.

   b) Calculate extrinsic parameters for each raw data images.

5) FPPN calibration is performed using Calibration Library.

   a) Calculate spatial depth offset.

   b) Applies polynomial surface fitting and extracts coefficients

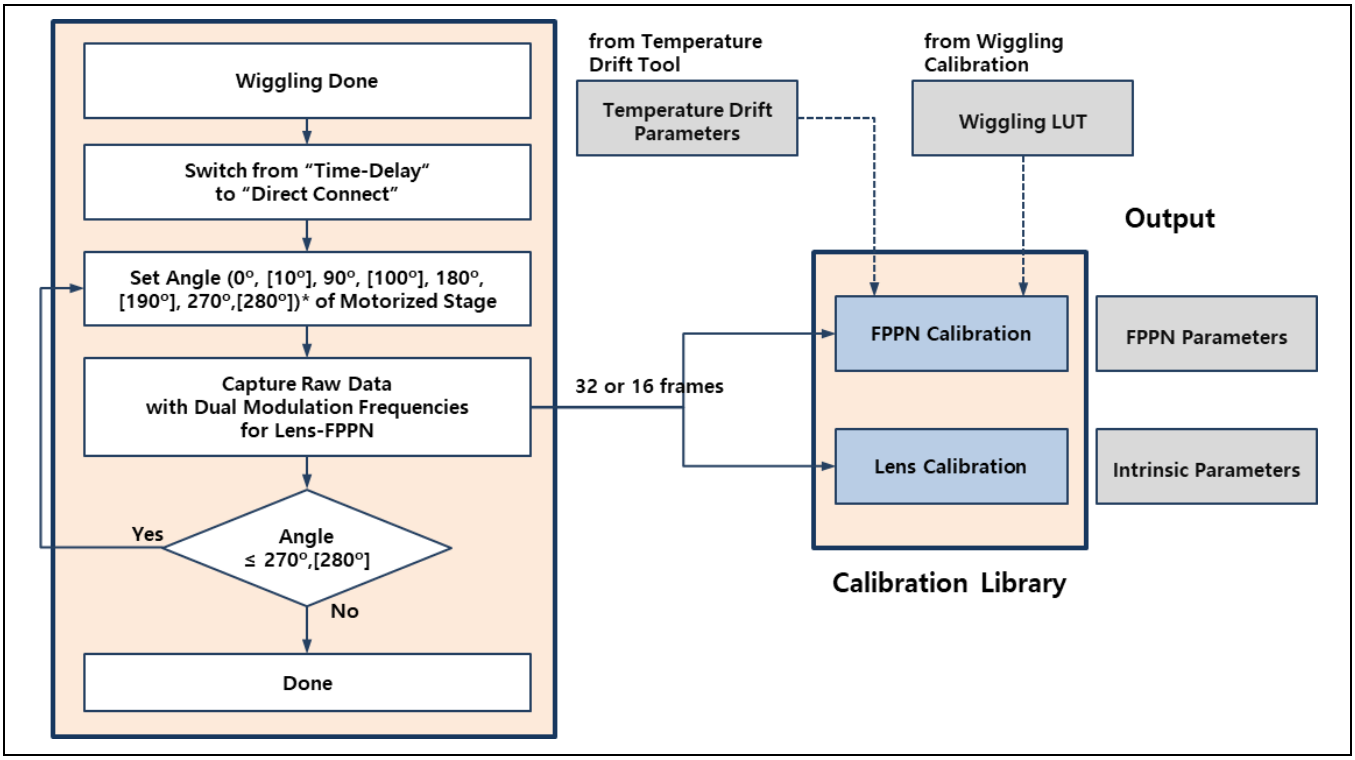

Figure 2.8 Lens-FPPN Calibration Procedure

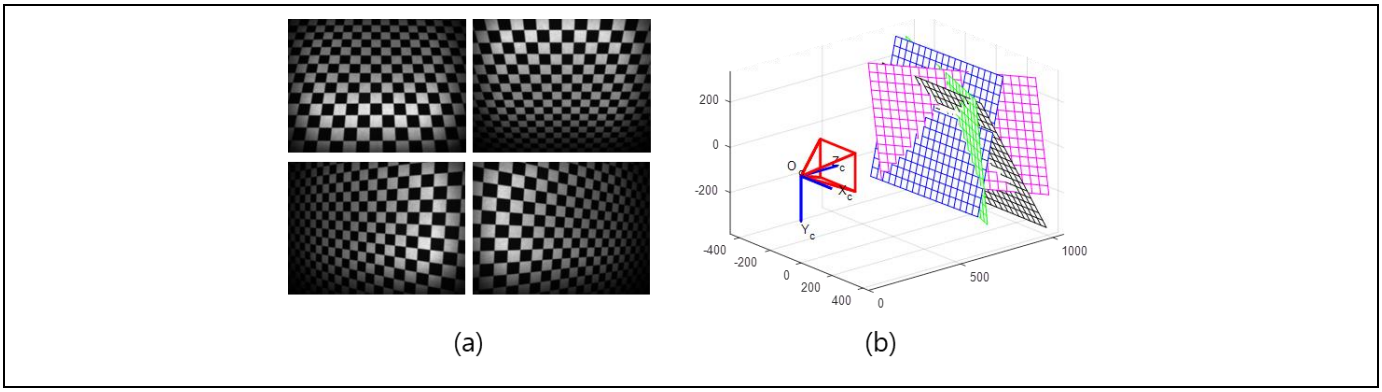

Figure 2.9 Lens Calibration Procedure.

Lens-FPPN Calibration Procedure in detail

FPPN calibration is to compensate pixel-location-dependent error due to the time-delay of the demodulation signal depending on pixel location and the misalignment of VCSEL and sensor (which is a geometric error). Each pixel of a ToF sensor can have different depth offset due to time-delay with respect to the injection of the reference control signal. Also, VCSEL and sensor can be misaligned due to limited space to be located in for small form factor. This misalignment can cause per-pixel depth offset.

As FPPN input data, same checker board images are used with different rotation angles which consists of basically 0°, 90°, 180° and 270°. In order to improve the performance, additional different rotation angles can be added such as 10°, 100°, 190° and 280°. Figure 2.10 show FPPN calibration procedure in detail while using 4 different rotation angle inputs. Ground truth depth image of each input checkerboard image can be generated with extrinsic parameters obtained from lens calibration. Input raw images are converted as measured depth. Low confidence depth data can be removed by using confidence map. Depth offsets are calculated in higher confidence area. Next, remained depth offset images are merged into one offset image.

Since areas of the low-confidence depth pixels are excluded, there are no depth offset data in some spatial positions as shown in Figure 2.11. In order to fill lack of depth offsets, special filtering is applied and then whole depth offsets can be obtained. FPPN depth offset can be estimated by polynomial surface fitting technique. Finally, calibration library calculates the FPPN depth offset LUT.

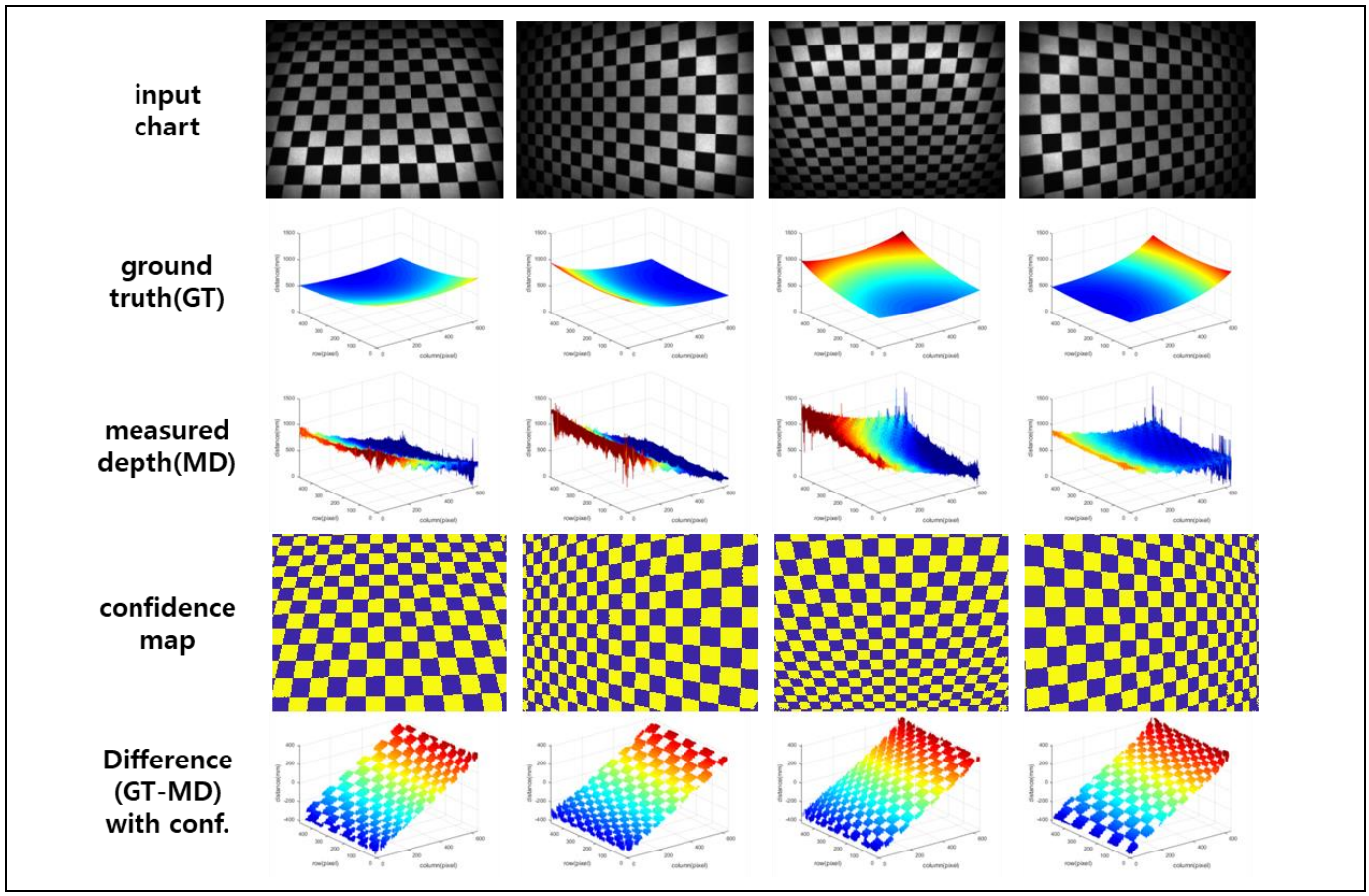

Figure 2.10 FPPN Calibration Procedure; each rotation.

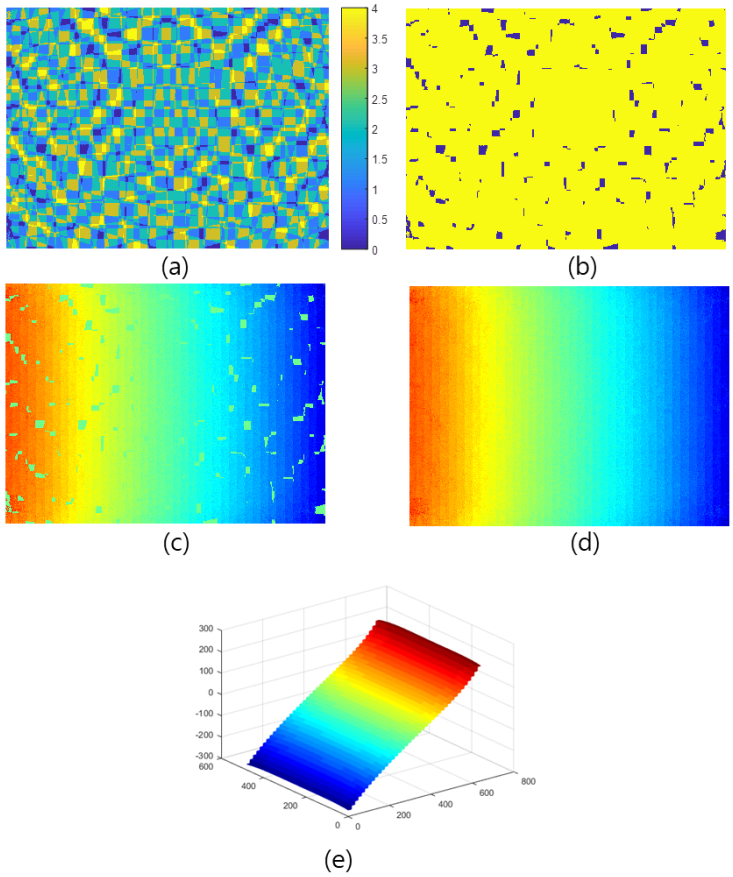

Figure 2.11 FPPN Calibration Procedure; merge all rotations.

2.3.2 **Equipment**

Figure 2.12 and Table 2.2 shows the example of lens-FPPN chart. The specification of lens-FPPN chart can be changed depending on the various conditions. The distance from a module and a plane chart is fixed by monitoring the code to prevent saturation. This distance is about 375 – 825 mm, depending on the optical power of VCSEL, integration time, and chart reflectivity.

When calibrating lens-FPPN, “Lens-FPPN & Temperature Drift Calibration Board” without the external time-delay as shown in Figure 2.3 has to be used.

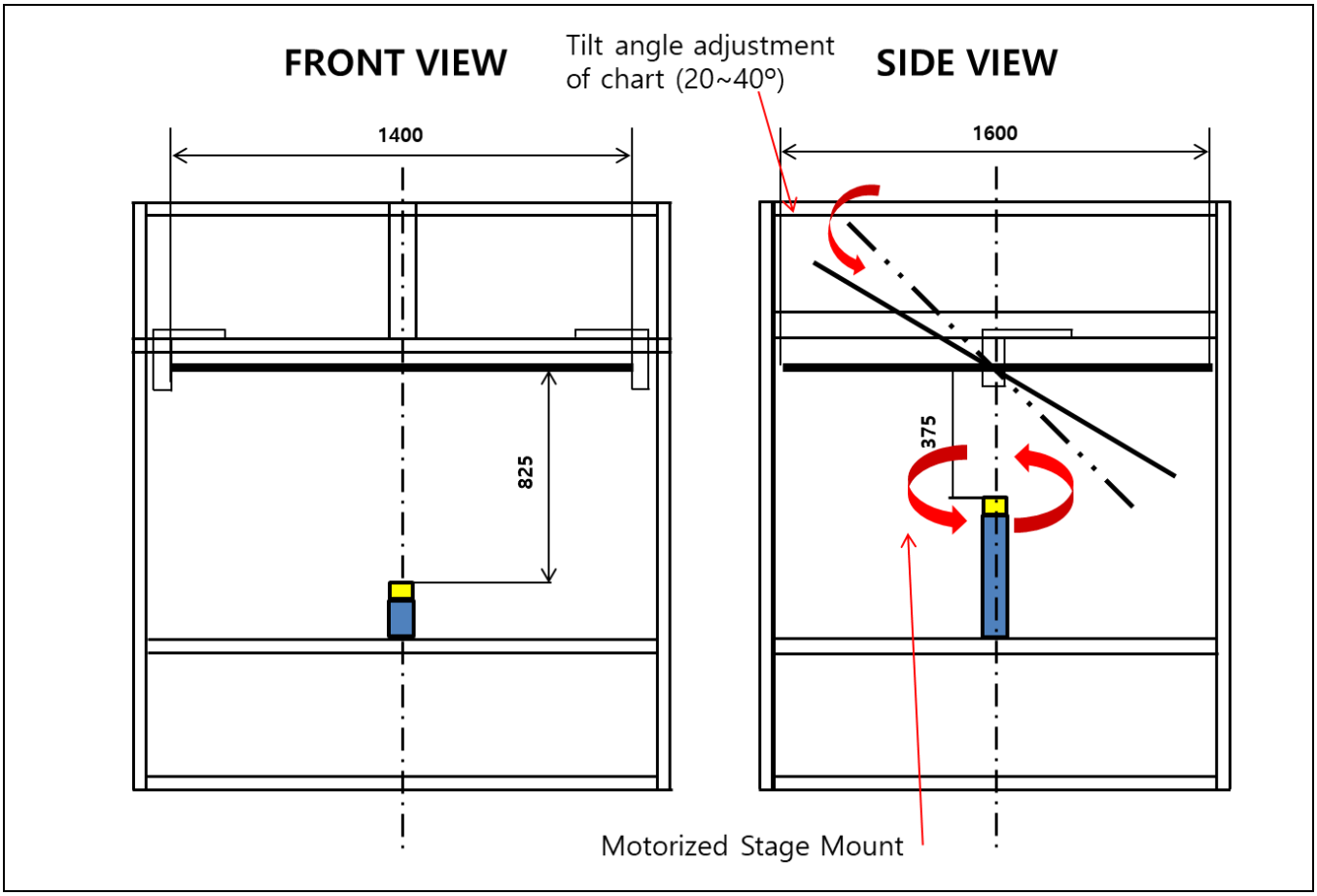

Figure 2.12 Lens-FPPN Calibration Chart. Height adjustable stage mount for module (375 – 825 mm) and an angle adjustable chart (20º – 40º). Height and angle will be fixed for a manufacturing environment.

Table 2.2 The Specification of Lens-FPPN Calibration Chart.

Chart type	Checkerboard
Chart size	1610 mm × 1410 mm
Material (lambertian reflectivity)	White > 80%, Dark > 10%
Square size	40 mm × 40 mm
Square count	40 x 35
Tilt angle adjustment	30º
Angle of Motorized Stage	0º, 90º, 180º, 270º + 10º, 100º, 190º, 280º (added angles depending on performance)
Distance from module	375 – 825 mm

3 Software

3.1 Overview

Software section is to explain usage of software calibration tool and library provided by Samsung Electronics. ToF calibration tools consist of temperature drift tool, calibration library and validation tool, as shown in Figure 3.1. In this section only temperature drift monitoring GUI (graphic user interface) tool for one-time calibration and calibration library for module to module calibration are covered except DEPS tool of validation tool. If board configuration is different between calibration and validation, board calibration should be additionally conducted in order to remove global offset using the Board Calibration Library.

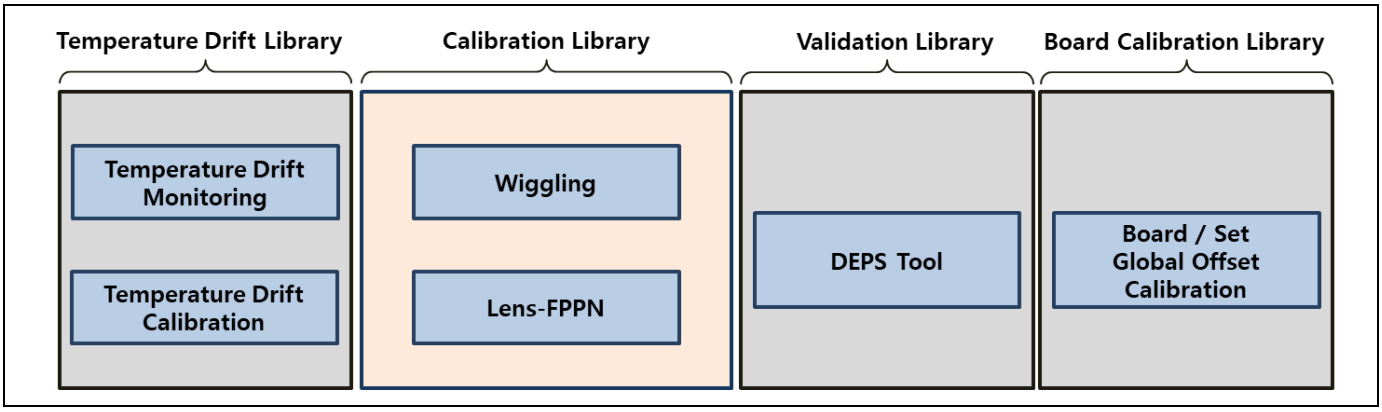

Figure 3.1 Released Softwares; 1) Temperature Drift Tool, 2) Calibration Library, 3) Validation Tool.

This document describes the following softwares.

Version Information

- Temperature Drift Library: version 2.0.x
- ToF Calibration Library: version 2.3.x
- Board Calibration Library: version 2.3.x (same as ToF Calibration Library)

3.2 Temperature Drift Tool

As mentioned in Section 2.1, temperature drift leads to a change in measured distance. In the calibration procedure, the systematic errors could be calibrated with the temperature drift parameters. For these parameters, Temperature Drift Tool calculates the calibration data at measurement of fixed distance.

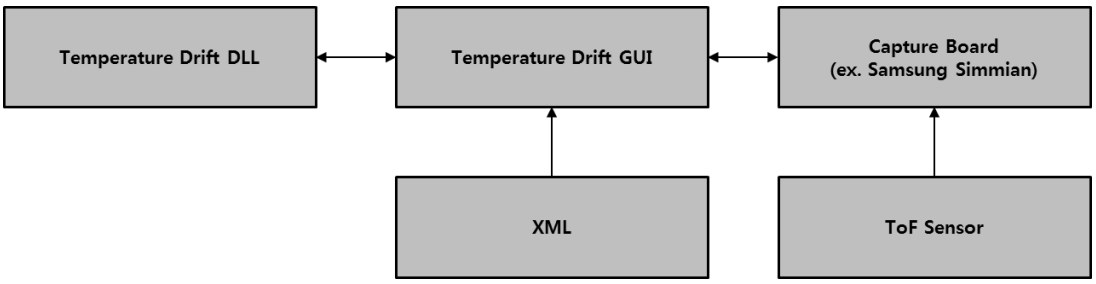

Figure 3.2 Temperature Drift Calibration Block Diagram

The Temperature Drift Tool is composed of data-processing, graphic-user-interface (GUI) and data-capture blocks. The data-processing block is a dynamic link library (DLL) type. Therefore, users can design their own user interface. The usage guide of Temperature Drift DLL will be explained in Section 3.2.1. For example, the Temperature Drift GUI is shown in Fig. 3.3 and controlled by configuration parameters in xml file. Drift Monitor GUI uses Simmain MV3 board and NxSimmian Software for a frame grabber. Your own frame grabber can be combined with Temperature Drift DLL. The Simmian software (NxSimmian) and device drivers must be installed for operation of example, Temperature Drift Monitor GUI.

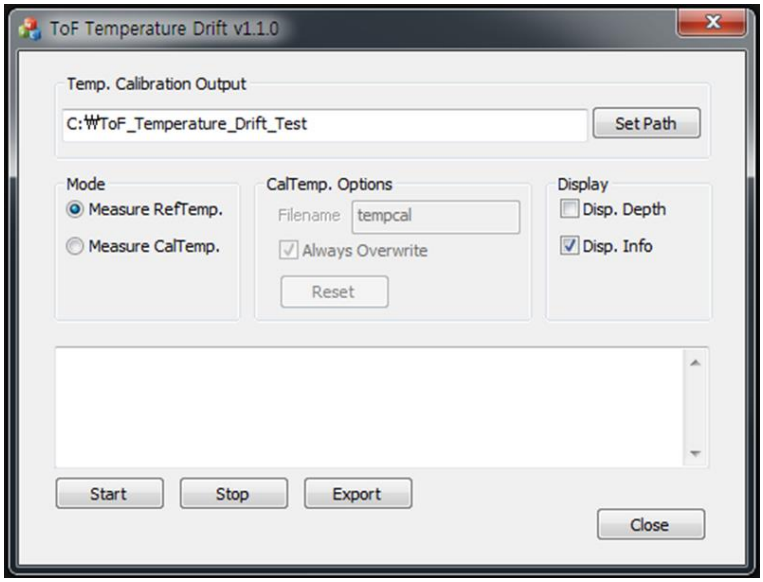

Figure 3.3 Temperature Drift Monitor GUI

3.2.1 **Usage**

The Temperature Drift Monitor options are shown in Fig. 3.4.

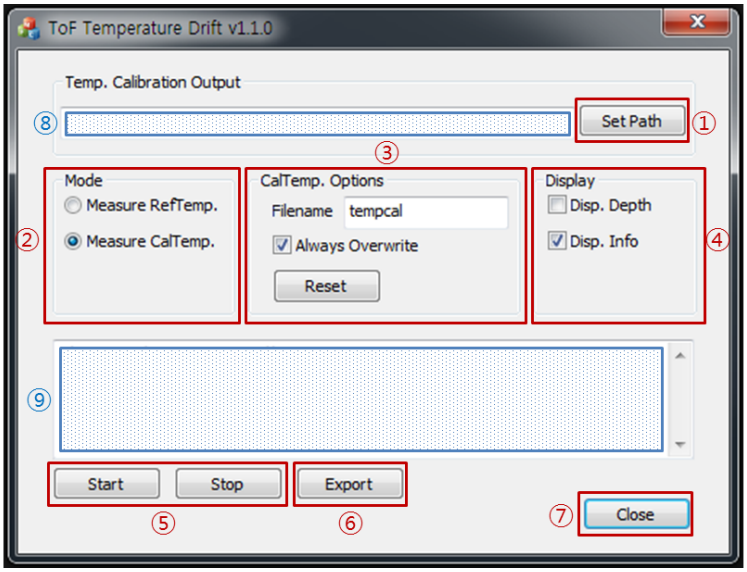

Figure 3.4 Temperature Drift Monitor Options

1. Set Path: Selecting a folder for exporting temperature calibration data. The selected path is shown in the path window (EditControl Num.8). The path of folder should be configured in English only. After all measurements have been completed, the temperature compensation result is saved in this path.

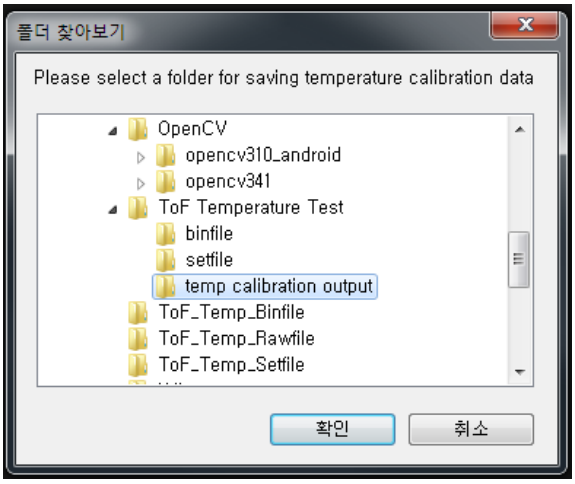

Figure 3.5 Set Path Dialog

2. Mode: Selecting temperature drift monitoring mode. 'Measure RefTemp.' is the mode to measure a reference temperature drift statistics after some warm-up time, t_w. During the warm-up time, distance measurement may not be correct (see Figure 3.6). 'Measure CalTemp.' is the mode to measure a calibrated temperature drift statistics based on reference temperature drift statistics for varying temperature.

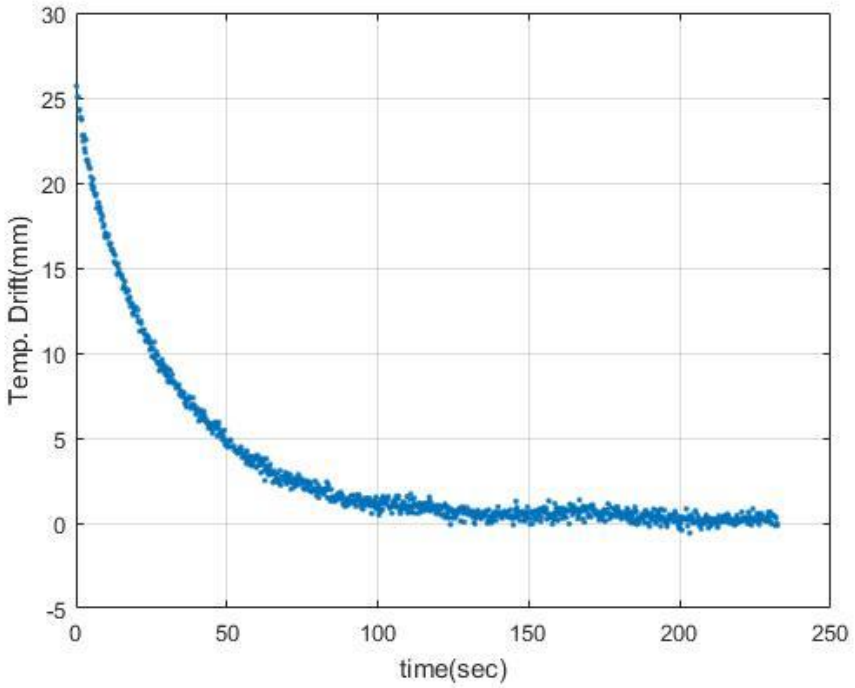

Figure 3.6 Temperature drift offset from reference depth caused by self-induced heating

3. CalTemp. Options: When the ‘Measure CalTemp’ mode is checked, the options of CalTemp will be enable. The save filename of CalTemp measurement can be set by filename editbox. It can be to check whether the same filename exists by checkbox ‘Always Overwrite’, and then all previous measurement data could be deleted by the reset button.

4. Display: The checkbox ‘Disp. Depth’ controls a display window of depth map as shown in Figure 3.7. This checkbox is disable during mode operation. ‘Disp. Info’ turn on and off about information window (Edit Control Num.9), as shown in Figure 3.8.

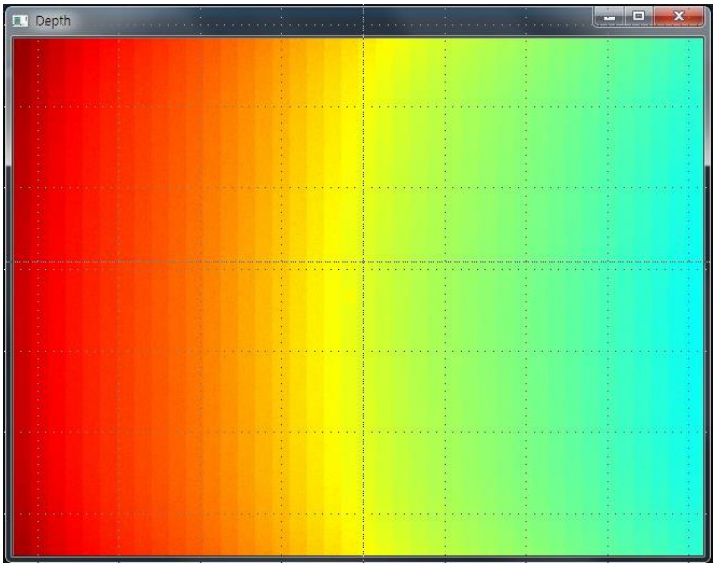

Figure 3.7 Display Depth Map

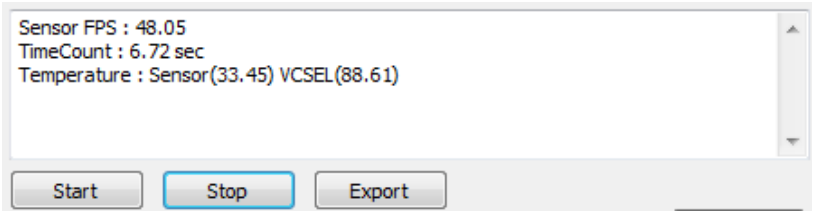

Figure 3.8 Display Information

5. Start and Stop: Start and stop the selected mode operation. After starting a mode, the start button is disabled. The disabled start button is enabled after finishing the mode operation or clicking the stop button.

6. Export: It exports the temperature calibration output file with all CalTemp measurement data. The output files will be saved in the path of EditControl Num.8.

7. Close: Close Temperature Drift Monitor tool

3.2.2 **Deliverables**

Below items are deliverables to measure temperature calibration data.

- ToFTemperatureDriftMonitor.exe : execution program
- ToFTemperatureMonitor_DLL.dll : temperature drift monitor library
- ToFTempMonitor.xml : configuration file in XML
- ToFTemperatureDrift_ API.h : API definition C++ header file

- RegisterMap, Setfile
- opencv_world341.dll : OpenCV library

3.2.3 **Temperature Drift Monitor Configuration file**

Temperature Drift Monitor is controlled by configuration parameters in XML files (see Figure 3.9). The path of folder should be configured in English only and all parameters should be enclosed by quotation marks.Note that should use muti-byte character set for configuration.

```
<TempMonitorXML>
  <!-- Path Config -->
  <RawFilePath>"C:\ToF_Temp_Rawfile\"</RawFilePath>
  <SetFilePath>"C:\ToF_Temp_Setfile\"</SetFilePath>
  <BinFilePath>"C:\ToF_Temp_Binfile\"</BinFilePath>
  <!-- FileName Config -->
  <RegisterMapFileName>"ToF_RegisterMap.bin"</RegisterMapFileName>
  <SetFileName>"ToF_SensorSetfile.set"</SetFileName>
  <!-- Control Config -->
  <UseExtEmbeddData>1</UseExtEmbeddData></TempMonitorXML>
```

Figure 3.9 Temperature Drift Monitor Configuration XML example

- RawFilePath: folder where captured data will be saved
- SetFilePath: folder of setfiles
- BinFilePath: folder where temperature drift monitor data will be saved
- RegisterMapFileName: register map filename
- SetFileName: setfile filename
- UseExtEmbeddData: set embedded data (0: internal and 1: external)

3.2.4 **Temperature Drift Monitor DLL API**

Temperature Drift Monitor DLL API is class structure supported for processing temperature data.

```
typedef enum _RET_CODE {
        E_ERROR_TEMP_SUCCESS = (0x00000000),
        E_ERROR_TEMP_FAIL = (0x00000001),
        E_ERROR_TEMP_CREATEINSTANCE = (0x00000002),
        E_ERROR_TEMP_DELETEINSTANCE = (0x00000003),
        E_ERROR_TEMP_SETRAWSIZE = (0x00000004),
        E_ERROR_TEMP_RESETTIME = (0x00000005),
        E_ERROR_TEMP_ELAPSEDTIME = (0x00000006),
```

```
        E_ERROR_TEMP_SETREFDATA = (0x00000007),
        E_ERROR_TEMP_LOADREFDATA = (0x00000008),
        E_ERROR_TEMP_SETREFCOUNT = (0x00000009),
        E_ERROR_TEMP_SETTOFCALDATA = (0x0000000A),
        E_ERROR_TEMP_SETWIGGCALPATH = (0x0000000B),
        E_ERROR_TEMP_SETFPPNCALPATH = (0x0000000C),
        E_ERROR_TEMP_RUNTEMPERATUREDRIFT = (0x0000000D),
        E_ERROR_TEMP_GETTEMPCALVAL = (0x0000000E),
        E_ERROR_TEMP_GETCENTERDEPTH = (0x0000000F),
        E_ERROR_TEMP_GETTEMPINFO = (0x00000010),
        E_ERROR_TEMP_GETTEMPREFINFO = (0x00000011),
        E_ERROR_TEMP_GETTOFVERSION = (0x00000012),
        E_ERROR_TEMP_CTRLDISPDEPTH = (0x00000013),
        E_ERROR_TEMP_CLOSEREFFINAL = (0x00000014),
        E_ERROR_TEMP_CLOSECALDATA = (0x00000015),
        E_ERROR_TEMP_CLEARALLPARAM = (0x00000016),
        E_ERROR_TEMP_GetDebugXML = (0x00000017),
        E_ERROR_TEMP_MODEL = (0x00000018),
        E_ERROR_TEMP_READIMAGE = (0x00000019),
        E_ERROR_TEMP_FILEOPEN = (0x0000001A),

}RET_CODE;

TOFTEMPERATUREMONITOR_DLL_API RET_CODE CreateInstance( );
TOFTEMPERATUREMONITOR_DLL_API RET_CODE DeleteInstance( );

TOFTEMPERATUREMONITOR_DLL_API RET_CODE SetRawSize(int row, int col);
TOFTEMPERATUREMONITOR_DLL_API RET_CODE ResetTime();
TOFTEMPERATUREMONITOR_DLL_API RET_CODE ElapsedTime(float& elapsedTime);

TOFTEMPERATUREMONITOR_DLL_API RET_CODE SetRefData(CString bin_path, CString
ref_filename);
TOFTEMPERATUREMONITOR_DLL_API RET_CODE LoadRefData(CString bin_path, CString
ref_filename, CString cal_filename);
TOFTEMPERATUREMONITOR_DLL_API RET_CODE SetRefCount(int refCount);
TOFTEMPERATUREMONITOR_DLL_API RET_CODE SetToFCalData (CString path);
TOFTEMPERATUREMONITOR_DLL_API RET_CODE RunTemperatureDrift(int calmode, CString
raw_path, CString raw_filename, bool use_ext_embedddata);


TOFTEMPERATUREMONITOR_DLL_API RET_CODE GetTempInfo(float& sensorTemp, float&
vcselTemp);
TOFTEMPERATUREMONITOR_DLL_API RET_CODE GetTempRefInfo(float* refCalData, float*
sensorTemp, float* vcselTemp);

TOFTEMPERATUREMONITOR_DLL_API RET_CODE GetTempCalVal(CString bin_path, CString
save_path, CString* filenames, int filename_number);
TOFTEMPERATUREMONITOR_DLL_API RET_CODE GetToFVersion(int& getVersion);

TOFTEMPERATUREMONITOR_DLL_API RET_CODE CloseRefFinal();
TOFTEMPERATUREMONITOR_DLL_API RET_CODE CloseCalData();
TOFTEMPERATUREMONITOR_DLL_API RET_CODE ClearAllParam();

// Debug
TOFTEMPERATUREMONITOR_DLL_API RET_CODE CtrlDispDepth(bool dispDepth);
```

```
TOFTEMPERATUREMONITOR_DLL_API RET_CODE GetError_TEMP(const RET_CODE errCode);
```

Variables

- RET_CODE : Pre-defined return value of API functions

```
        E_ERROR_TEMP_SUCCESS = (0x00000000) / return success
        E_ERROR_TEMP_FAIL = (0x00000001) / return fail
        E_ERROR_TEMP_CREATEINSTANCE = (0x00000002) / CreateInstance is failed
        E_ERROR_TEMP_DELETEINSTANCE = (0x00000003) / DeleteInstance is failed
        E_ERROR_TEMP_SETRAWSIZE = (0x00000004) / SetRawSize is failed
        E_ERROR_TEMP_RESETTIME = (0x00000005) / ResetTime is failed
        E_ERROR_TEMP_ELAPSEDTIME = (0x00000006) / ElapsedTime is failed
        E_ERROR_TEMP_SETREFDATA = (0x00000007) / SetRefData is failed
        E_ERROR_TEMP_LOADREFDATA = (0x00000008) /    LoadRefData is failed
        E_ERROR_TEMP_SETREFCOUNT = (0x00000009) / SetRefCount is failed
        E_ERROR_TEMP_SETTOFCALDATA = (0x0000000A) / SetTofCalData is failed
        E_ERROR_TEMP_SETWIGGCALPATH = (0x0000000B) / SetWiggCalpath is failed
        E_ERROR_TEMP_SETFPPNCALPATH = (0x0000000C) / SetFppnCalPath is failed
        E_ERROR_TEMP_RUNTEMPERATUREDRIFT = (0x0000000D) / RunTemperatureDrift is failed
        E_ERROR_TEMP_GETTEMPCALVAL = (0x0000000E) / GetTempCalVal is failed
        E_ERROR_TEMP_GETCENTERDEPTH = (0x0000000F) / GetCenterDepth is failed
        E_ERROR_TEMP_GETTEMPINFO = (0x00000010) / GetTempInfo is failed
        E_ERROR_TEMP_GETTEMPREFINFO = (0x00000011) / GetTempRefInfo is failed
        E_ERROR_TEMP_GETTOFVERSION = (0x00000012) / GetToFVersion is failed
        E_ERROR_TEMP_CTRLDISPDEPTH = (0x00000013) / CtrlDispDepth is failed
        E_ERROR_TEMP_CLOSEREFFINAL = (0x00000014) / CloseRefFinal is failed
        E_ERROR_TEMP_CLOSECALDATA = (0x00000015) / CloseCalData is failed
        E_ERROR_TEMP_CLEARALLPARAM = (0x00000016) / ClearAllParam is failed
        E_ERROR_TEMP_GetDebugXML = (0x00000017) / Cannot find the configuration file to
read
        E_ERROR_TEMP_MODEL = (0x00000018) /
        E_ERROR_TEMP_READIMAGE = (0x00000019) / Cannot read image in the file
        E_ERROR_TEMP_FILEOPEN = (0x0000001A) / Cannot open the file
```

-

Functions

- RET_CODE CreateInstance()

  create temperature dll instance

- RET_CODE DeleteInstance()

  delete temperature dll instance

- RET_CODE SetRawSize(int row, int col)

set raw image size
row – height of raw image
col – width of raw image

- RET_CODE ResetTime()

  time initialization

- RET_CODE ElapsedTime(float& elapsedTime)

  return elapsed time in sec after time initialization

- RET_CODE SetRefData(CString bin_path, CString ref_filename)

  set configurations of reference file to measure the reference data

  bin_path – binary file path
  ref_filename – reference data filename

- RET_CODE LoadRefData(CString bin_path, CString ref_filename, CString cal_filename)

  set configurations of reference data to load and calibration data to measure

  bin_path – binary file path
  ref_filename – reference data filename
  cal_filename – calibration data filename

- RET_CODE SetRefCount(int refCount)

  set frame numbers for averaging reference temperature data

  refCount – frame numbers to average for reference data

- RET_CODE SetToFCalData (CString path)

  set and load calibration data

  path – calibration binary file path

- RET_CODE RunTemperatureDrift(int calmode, CString raw_path, CString raw_filename, bool use_ext_embedddata)

  process and read raw data file based on raw data file path, mode parameter
  calmode – 0 is 'Measure RefTemp.' to measure the reference data
  and 1 is 'Measure CalTemp.' To measure the calibration data

  raw_path – raw file path to read (captured raw and embedded file)
  raw_filename – captured raw filename to read
  use_ext_embedddata – 0 is internal embedded data and 1 is external embedded data

- RET_CODE GetTempCalVal(CString bin_path, CString save_path, CString* filenames, int

filename_number)

export the final temperature calibration data to save_path

bin_path – binary file path
save_path – temperature calibration output file path
filenames – measured calibration data filenames
filename_number – the number of filenames

- RET_CODE GetTempInfo(float& sensorTemp, float& vcselTemp)

  get sensor and vcsel temperature

  sensorTemp – sensor temperarue in degree

  vcselTemp – vcsel temperarue in degree

- RET_CODE GetTempRefInfo (float* refCalData, float* sensorTemp, float* vcselTemp)

  get reference temperature calibration results for each modulation frequency

  refCalData[2] – reference calibration depth data

  sensorTemp[2] – reference sensor temperarue in degree

  vcselTemp[2] – reference vcsel temperarue in degree

- RET_CODE CloseRefFinal()

  close reference data file

- RET_CODE CloseCalData()

  close calibration data file

- RET_CODE ClearAllParam()

  clear all parameter (time, count)

- RET_CODE CtrlDispDepth(bool dispDepth)

  control display depth map window

  dispDepth – true for displaying the depth map

- RET_CODE GetError_TEMP(const RET_CODE errCode)

  Get ErrorDescription

This is a block diagram for software development guide with Temperature Drift Monitor API.

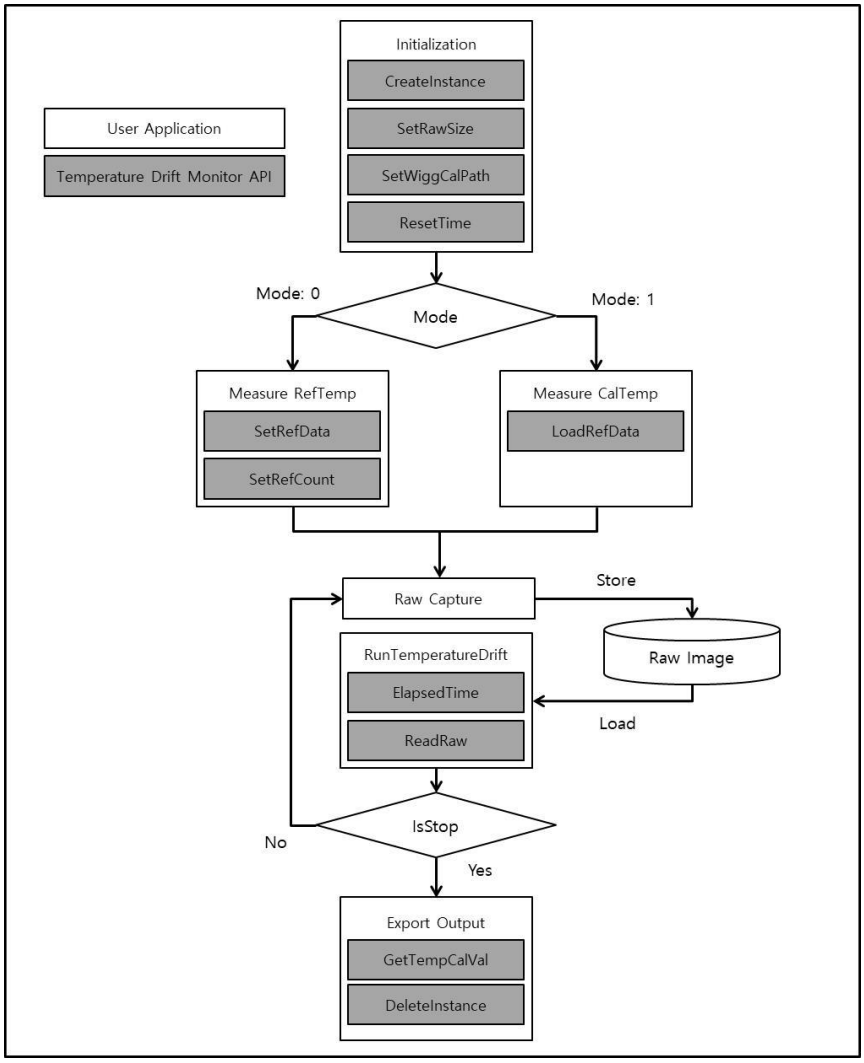

Figure 3.10 Temperature Drift Monitor API Guide

This is example code for Temperature Drift Monitor API.

```
#include "stdafx.h"
#include <atlstr.h>
#include "ToFTemperatureDrift_API.h"
#include <io.h>
#include <vector>


using namespace std;


int main()
{
```

```
	/* mode (0: Measure RefTemp. 1: Measure CalTemp. 2: Extract cal. data) */
	int TemperatureMonitorMode = 0;
	bool bRunning = (TemperatureMonitorMode == 2) ? false : true;

	int row = 480; // depth height from raw image
	int col = 640; // depth width from raw image
	int nRefCount = 10; // averaging 10 frames
	float sensorTemp = 0.0f; // sensor temperature for checking
	float vcselTemp = 0.0f; // vcsel temperature for checking
	const float mode1_timeOut = 600; // timeOut in sec for mode1
	bool bUseExtEmb = true; // true : use external embedded info, false : use internal
embedded info
	int getVersion = 0; // Monitoring tool version read
	float elapsedTime = 0.f;

	RET_CODE ret = E_ERROR_TEMP_SUCCESS; //return error code initialize

	CString strRawFilePath = _T("c:\\ToF_Temp_Rawfile\\");
	CString strBinFilePath = _T("c:\\ToF_Temp_Binfile\\");
	CString strCalOutputPath = _T("c:\\ToF_Temp_Output\\");

	CString strRefFilename = _T("refTempFinalData.bin");
	CString strCalFilename = _T("calTempData.bin");
	CString strRawFilename = _T("ToF_0000");

	/* create ToF temperature dll instance */
	ret = CreateInstance();

	if (E_ERROR_TEMP_SUCCESS != ret) {
		GetError_TEMP(ret);
		return 0;
	}

	ret = GetToFVersion(getVersion);

	if (E_ERROR_TEMP_SUCCESS != ret) {
		GetError_TEMP(ret);
		return 0;
	}

	int version = getVersion;
	printf("Temperature drift version : %d \n ", version);

	/* raw file size */
	ret = SetRawSize(row, col);

	/* check calibration data folder existence */
	if (GetFileAttributes(strRawFilePath) == INVALID_FILE_ATTRIBUTES)
		CreateDirectory(strRawFilePath, NULL);

	if (GetFileAttributes(strBinFilePath) == INVALID_FILE_ATTRIBUTES)
		CreateDirectory(strBinFilePath, NULL);

	if (GetFileAttributes(strCalOutputPath) == INVALID_FILE_ATTRIBUTES)
		CreateDirectory(strCalOutputPath, NULL);
```

```
	/* set & load wiggling calibration data */
	ret = SetToFCalData(strCalOutputPath);
	if (E_ERROR_TEMP_SUCCESS != ret)
	{
		printf("[ERROR] SetToFCalData : can not find data %s \n ", strCalOutputPath);
		GetError_TEMP(ret);
		return 0;
	}

	if (TemperatureMonitorMode == 0)
	{
		/* measure reference temperature drift statistics */
		ret = SetRefData(strBinFilePath, strRefFilename);
		if (E_ERROR_TEMP_SUCCESS != ret)
		{
			printf("[ERROR] SetRefData : can not find data %s, %s\n ",
strBinFilePath, strRefFilename);
			GetError_TEMP(ret);
			return 0;
		}

		ret = SetRefCount(nRefCount);
		if (E_ERROR_TEMP_SUCCESS != ret)
		{
			GetError_TEMP(ret);
			return 0;
		}
	}
	else if (TemperatureMonitorMode == 1)
	{
		/* check calibration data folder existence */
		if (GetFileAttributes(strBinFilePath + _T("caldata\\")) ==
INVALID_FILE_ATTRIBUTES)
			CreateDirectory(strBinFilePath + _T("caldata\\"), NULL);

		/* measure calibrated temperature drift statistics */
		CString strFreq1RefFilename, strFreq2RefFilename;

		int idx = strRefFilename.ReverseFind(_T('.'));
		strFreq1RefFilename = strRefFilename.Left(idx) + _T("_f1.bin");
		strFreq2RefFilename = strRefFilename.Left(idx) + _T("_f2.bin");

		/* check reference data existence */
		if ((GetFileAttributes(strBinFilePath + strFreq1RefFilename) ==
INVALID_FILE_ATTRIBUTES)    && (GetFileAttributes(strBinFilePath + strFreq2RefFilename) ==
INVALID_FILE_ATTRIBUTES))
		{
			/* error */
			printf("Could not find the reference data files");
			return -1;
		}
		else
		{
			if (E_ERROR_TEMP_SUCCESS != LoadRefData(strBinFilePath,
```

```
strRefFilename, strCalFilename))
                        {
                                printf("[ERROR] LoadRefData : can not find data \n ");
                                ret = LoadRefData(strBinFilePath, strRefFilename,
strCalFilename);
                                GetError_TEMP(ret);
                                return 0;
                        }
                }
        }

        /* init. time */
        ResetTime();

        while (bRunning)
        {
                if (!bRunning) break;

                /* user capture code */
                // TODO : place code here to capture raw file
                // user client capture code with raw filename(strRawFilename)
                /* read raw file */

                ret = RunTemperatureDrift(TemperatureMonitorMode, strRawFilePath,
strRawFilename, bUseExtEmb);

                if (E_ERROR_TEMP_SUCCESS == ret) {
                        bRunning = true;
                }
                else if (E_ERROR_TEMP_FAIL == ret) {
                        bRunning = false;
                }
                else {
                        GetError_TEMP(ret);
                        return 0;
                }

                /* user read variables */
                // TODO : place code here to read variables
                /* get elapsed time */

                ret = ElapsedTime(elapsedTime);
                if (E_ERROR_TEMP_SUCCESS != ret)
                {
                        GetError_TEMP(ret);
                        return 0;
                }

                float elapsed_time = elapsedTime;

                /* get temperature data */
                ret = GetTempInfo(sensorTemp, vcselTemp);
                if (E_ERROR_TEMP_SUCCESS != ret)
                {
                        GetError_TEMP(ret);
```

```
                        return 0;
                }

                /* Debugging for operation */
                printf("[%.2f sec] sensorTemp = %f, vcselTemp = %f \n", elapsed_time,
sensorTemp, vcselTemp);

                /* user time out condition code for mode 1 */
                // TODO : place code here to capture raw file
                if ((TemperatureMonitorMode == 1) & (elapsed_time > mode1_timeOut))
                {
                        bRunning = false;
                }

                if (!bRunning) break;

        }

        if (TemperatureMonitorMode == 0)
        {
                /* get reference calibration data for checking */
                float calData[2] = { 0.0f };
                float sensorData[2] = { 0.0f };
                float vcselData[2] = { 0.0f };

                ret = GetTempRefInfo(calData, sensorData, vcselData);
                if (E_ERROR_TEMP_SUCCESS != ret)
                {
                        GetError_TEMP(ret);
                        return 0;
                }
                /* Debugging for result */
                for (int i = 0; i < 2; i++)
                {
                        printf("[%d] calData = %f, sensorData = %f, vecselData = %f \n", i,
calData[i], sensorData[i], vcselData[i]);
                }

                CloseRefFinal();
                ClearAllParam();
        }
        else if (TemperatureMonitorMode == 1)
        {
                CloseCalData();
                ClearAllParam();
        }
        else if (TemperatureMonitorMode == 2)
        {
                /* make measured calibration data list */
                // The measured calibration data is stored in subfolder (‘caldata’) of
                // binary file path. We can export the final temperature calibration output
                // by filename list in the subfolder.

                vector<CString> vecCalFile;
                CString findpath = strBinFilePath + _T("caldata\\");
```

```
		CString findfile = _T("*.bin");

		intptr_t handle;
		int result = 1;
		_finddata_t fd;

		handle = _findfirst((CStringA)(findpath + findfile), &fd);

		while (result != -1 && handle != -1)
		{
			vecCalFile.push_back(CString(fd.name));
			result = _findnext(handle, &fd);
		}

		_findclose(handle);
		/* export calibration output */
		ret = GetTempCalVal(findpath, strCalOutputPath, &vecCalFile[0],
vecCalFile.size());
		if (E_ERROR_TEMP_SUCCESS != ret)
		{
			GetError_TEMP(ret);
			return 0;
		}
		printf("Exporting is done \n");
	}

	/* delete ToF temperature dll instance */
	DeleteInstance();

	return 0;
}
```

3.2.5 **Temperature Drift Monitor Output**

Temperature Drift Monitor tool generates outputs as binary format of temperature drift statistics. The some file name can be changed by user.

- `tempcal.bin`

  temperature drift data (The filename can be changed. It is stored for each frequency)
  (float elapsed_time, float sensor_temperature, float vcsel_temperature, float depth)

- `refTempFinalData.bin`

  temperature drift data after warm-up time (mean value of data the number of `SetRefCount set value`)
  (float sensor_temperature_mean, float vcsel_temperature_mean, float depth_mean)

- `temp.bin`

  temperature calibration output
  (float calibration_val)

3.3 **Calibration Library**

Calibration library is common dynamic library (*.dll) developed in C++. Since calibration library internally uses OpenCV library's functions and data structures, OpenCV dynamic library (currently 3.4.1 version) is included in deliverables. Also, two sets of calibrarion library will be provided according to operating systems x64 and x86.

3.3.1 **Deliverables**

Below items are deliverables to calibrate ToF sensor.

(1) Opencv_world341.dll : OpenCV library

(2) ToFCalParam.xml : calibration operation configuration file in XML

(3) ToF_Calibration_DLL_x64.dll/ ToF_Calibration_DLL_x64.lib : calibration dynamic libraries for x64

(4) ToF_Calibration_DLL_Win32.dll / ToF_Calibration_DLL_Win32.lib: calibration dynamic libraries for x86

(5) ToF_Calibration_EXE_x64/ToF_Calibration_EXE_Win32 : execution example program

(6) ToF_Calibration_API.h : API definition C++ header file

(7) ToF_Calibration_EEPROM_MAP.pdf : Memory map of calibration output result

(8) _ToF_cal.bin : calibration test binary file

3.3.2 **Calibration Configuration file**

In order to control and set the proper calibration environmnent, ToFCalParam.xml is provided. ToFCalParam.xml includes input/output interfaces and specific paramaters in each calibration module. Note that the path of folder should be configured in English only and all parameters should be enclosed by quotation marks and should use muti-byte character set for configuration.

```
<!-- Common Calibration Parameters -->
<COMM_CalFinalBinFilePath>"..\DB\Calibration_result\"</COMM_CalFinalBinFilePath>
<COMM_EnableWigg>1</COMM_EnableWigg>
<COMM_EnableLens>1</COMM_EnableLens>
<COMM_EnableFPPN>1</COMM_EnableFPPN>
<COMM_EnableExtE>1</COMM_EnableExtE>
<COMM_ImageWidth>640</COMM_ImageWidth>
<COMM_ImageHeight>480</COMM_ImageHeight>
<COMM_ModulationFrequency1>100</COMM_ModulationFrequency1>
<COMM_ModulationFrequency2>80</COMM_ModulationFrequency2>
<COMM_PixelSize>7</COMM_PixelSize>
<COMM_ModuleTTL>5</COMM_ModuleTTL>
<COMM_ValidationDistance>600</COMM_ValidationDistance>
<COMM_GlobalOffsetFrequency1>0</COMM_GlobalOffsetFrequency1>
<COMM_GlobalOffsetFrequency2>0</COMM_GlobalOffsetFrequency2>
<!-- Temperature Calibration Parameters -->
<TEMP_TempBinFilePath>"..\DB\Calibration_temp\"</TEMP_TempBinFilePath>
<TEMP_DriftBinFilePath>"..\DB\Calibration_drift\"</TEMP_DriftBinFilePath>
<!-- Lens & FPPN Calibration Parameters -->
<LENS_BinFilePath>"..\DB\Calibration_lensfppn\"</LENS_BinFilePath>
<LENS_RawFilePath>"..\DB\lens_fppn\"</LENS_RawFilePath>
<LENS_RawFilePrefix>"lensfppn_"</LENS_RawFilePrefix>
<LENS_DirNum>4</LENS_DirNum>
<LENS_AvgNum>10</LENS_AvgNum>
<!-- Wiggling Calibration Parameters -->
<WIGG_BinFilePath>"..\DB\Calibration_wigg\"</WIGG_BinFilePath>
<WIGG_RawFilePath>"..\DB\wiggling\"</WIGG_RawFilePath>
<WIGG_RawFilePrefix>"wigg_"</WIGG_RawFilePrefix>
<WIGG_DisStr>0</WIGG_DisStr>
<WIGG_DisEnd>1392</WIGG_DisEnd>
<WIGG_DisStep>72</WIGG_DisStep>
<WIGG_DisCap>550</WIGG_DisCap></CalParameter>
```

Figure 3.11 Calibration Configuration XML example

3.3.2.1 Common Calibration Parameters

Common paramters for all calibration modes can be set.

- COMM_FinalCalFilePath: calibration binary final output file location in English
  - Default: “..\DB\Calibration_result\”
- COMM_EnableWigg: wiggling calibration on/off
  - Default : 1 (on:1, off:0)
- COMM_EnableLens: lens calibration on/off
  - Default : 1 (on:1, off:0)

- COMM_EnableFPPN: fppn calibration on/off
  - Default : 1 (on:1, off:0)
- COMM_EnableExtE: external information file file interface on/off, in case of off, information will be in embedded line in phase raw input
  - Default : 1 (on:1, off:0)
- COMM_ImageWidth: final depth output image width
  - Default : 640
- COMM_ImageHeight: final depth output image height
  - Default : 480
- COMM_ModulationFrequency1: first modulation frequency value in MHz
  - Default : 100
- COMM_ModulationFrequency2: second modulation frequency value in MHz
  - Default : 80
- COMM_PixelSize: pixel size of RX sensor in um
  - Default : 7
- COMM_ModuleTTL: module TTL in mm
  - Default : 5
- COMM_ValidataionDistance: validataion distance for board calibration in mm
  - Default : 600
- COMM_GlobalOffsetFrequency1: global offset in mm for first modulation frequency
  - Default : 0 (0 ~ 1 normalized depth value)
- COMM_GlobalOffsetFrequency2: global offset in mm for second modulation frequency
  - Default : 0 (0 ~ 1 normalized depth value)

3.3.2.2 Temperature Calibration Parameters

Temperature calibration parameters can be set.

- TEMP_TempBinFilePath: temperature calibration intermediate binary output file location in English corresponding memory map
  - Default: "..\DB\Calibration_temp\"

- TEMP_DriftBinFilePath: temperature drift monitoring tool binary input file location in English. Note that temperature drift binary files must be included in this folder.
    - Default: "..\DB\Calibration_drift\"

3.3.2.3 Lens & FPPN Calibration Paramters

Lens & FPPN calibration parameters can be set.

- LENS_BinFilePath: lensfppn calibration intermediate binary output file location in English corresponding memory map
    - Default: "..\DB\Calibration_lensfppn\"
- LENS_RawFilePath: lens & fppn calibration input file location in English
    - Default: "..\DB\lensfppn\"
- LENS_RawFilePrefix: prefix of input raw file name
    - Default : "lensfppn_" (actual raw file name is set as "lensfppn_~~.raw")
- LENS_DirNum: total calibration input image direction number same as converted depth image number
    - Default : 4 (4 direction includes 0°, 90°, 180° and 270° as default, actual raw file number is same as LENS_Num * 4 in 4 Tab, shuffle and dual frequency mode)
- LENS_AvgNum: average calibration input image number for each direction same as converted depth image number
    - Default : 10 (actual raw file number is same as LENS_Num * 4 in 4 Tab, shuffle and dual frequency mode)

3.3.2.4 Wiggling Calibration Paramters

Wiggling calibration parameters can be set.

- WIGG_BinFilePath: wiggling calibration intermediate binary output file location in English corresponding memory map
    - Default: "..\DB\Calibration_wigg\"
- WIGG_RawFilePath: wiggling calibration input file location in English
    - Default: "..\DB\wiggling\"
- WIGG_RawFilePrefix: prefix of input raw file name
    - Default : "wigg_" (actual raw file name is set as "wiggling_~~.raw")
- WIGG_DisStr: starting control code setting of the time-delay circuit

  - Default : 0

- WIGG_DisEnd: ending control code setting of the time-delay circuit
  - Default : 1392
- WIGG_DisStep: increment of time-delay control code
  - Default : 72
- WIGG_DistCap: initial estimate of the distance between the sensor and the captured chart in mm
  - Default : 550

3.3.3 **Calibration DLL API**

Only one main API is needed to execute calibration for module to module(M2M) calibration by calling in execution program. Calibration tool can be controlled by initial paramters.

```
typedef enum _RET_CODE {
	E_ERROR_TOF_SUCCESS		=		(0x00000000),
	E_ERROR_TOF_INIT_COMM		=		(0x00000110),
	E_ERROR_TOF_INIT_TEMP		=		(0x00000120),
	E_ERROR_TOF_INIT_WIGG		=		(0x00000130),
	E_ERROR_TOF_INIT_FPPN		=		(0x00000140),
	E_ERROR_TOF_INIT_LENS		=		(0x00000150),
	E_ERROR_TOF_INIT_BOARD		=		(0x00000160),
	E_ERROR_TOF_INIT_PHONE		=		(0x00000170),
	E_ERROR_TOF_RUN_COMM		=		(0x00000210),
	E_ERROR_TOF_RUN_TEMP		=		(0x00000220),
	E_ERROR_TOF_RUN_WIGG		=		(0x00000230),
	E_ERROR_TOF_RUN_FPPN		=		(0x00000240),
	E_ERROR_TOF_RUN_LENS		=		(0x00000250),
	E_ERROR_TOF_RUN_BOARD		=		(0x00000260),
	E_ERROR_TOF_RUN_PHONE		=		(0x00000270),
	E_ERROR_TOF_RUN_INPUT		=		(0x00000280),
	E_ERROR_TOF_RUN_OUTPUT		=		(0x00000290),
	E_ERROR_TOF_NOT_SUPPORT	=	(0x00000A10),
}RET_CODE;

typedef struct _CAL_INIT_PARAM {
	// Common Cal. Parameters
	char* strCalBinFilePath;
	int enWigg;
	int enLens;
	int enFPPN;
	int enExtE;
	int imgWidth;
	int imgHeight;
	float fModulationFreq1;
	float fModulationFreq2;
	float pixSize;
	float modTTL;
	float valDist;
	float globalOffsetFreq1;
```

```
        float globalOffsetFreq2;

        // Temperature Cal. Parameters
        char* strTempBinFilePath;
        char* strTempDriftBinFilePath;

        // Lens & FPPN Cal. Parameters
        char* strLensBinFilePath;
        char* strLensRawPath;
        char* strLensPrefix;
        int lensDirNum;
        int lensAvgNum;

        // Wiggling Cal. Parameters
        char* strWiggBinFilePath;
        char* strWiggRawPath;
        char* strWiggPrefix;
        int wiggDisStr;
        int wiggDisEnd;
        int wiggDisStep;
        int wiggDisCap;

}CAL_INIT_PARAM, *PCAL_INIT_PARAM;

TOF_CALBRATION_API RET_CODE Calibrate_ToF(const CAL_INIT_PARAM& calInitParams);
TOF_CALBRATION_API RET_CODE GetError_ToF(const RET_CODE errCode);
```

Variables

- `RET_CODE` : Pre-defined return value of API functions
- `CAL_INIT_PARAM` : calibration initial structure including variables which are same as XML paramters, please refer to calibration configuration file section

Functions

- `RET_CODE Calibrate_ToF(CAL_INIT_PARAM& calInitParams)`
    - Run calibration tool
    - Return : RET_CODE (refer to variables)
    - Argument :

      CAL_INIT_PARAM : calibration initial paramters controlling calibration tool
- `RET_CODE GetError_ToF(const int errCode)`
    - Display and described calibration tool with pre-defined error codes
    - Return : RET_CODE (refer to variables)

- Argument :

  RET_CODE (refer to variables)

Calibration execution example is given below. Firstly control paramters are set and then function `Calibrate_ToF_Board()` calling is carried out. Finally, calibration execution result can be monitored by checking return value. As a result, "_ToF_cal.bin" can be generated.

```
#include <iostream>
#include <string>
#include "ToF_Calibration_API.h"

using namespace std;

void setInitParams(CAL_INIT_PARAM& calInitParams)
{
	// Common Cal. Parameters
	calInitParams.strCalBinFilePath = "..\\DB\\Calibration_result\\"; // path in English
	calInitParams.enWigg = 1; //on : 1, off : 0
	calInitParams.enLens = 1; //on : 1, off : 0
	calInitParams.enFPPN = 1; //on : 1, off : 0
	calInitParams.enExtE = 0; //on : 1, off : 0
	calInitParams.imgWidth = 640; // depth output width
	calInitParams.imgHeight = 480; // depth output height
	calInitParams.fModulationFreq1 = 100; // Mhz
	calInitParams.fModulationFreq2 = 80; // Mhz
	calInitParams.pixSize = 7; // um
	calInitParams.modTTL = 5; // mm
	calInitParams.valDist = 600; // mm
	calInitParams.globalOffsetFreq1 = 0; // 0 ~ 1
	calInitParams.globalOffsetFreq2 = 0; // 0 ~ 1

	// Temperature Cal. Parameters
	calInitParams.strTempBinFilePath = "..\\DB\\Calibration_temp\\"; // path in English
	calInitParams.strTempDriftBinFilePath = "..\\DB\\Calibration_drift\\"; // path in English

	// Lens & FPPN Cal. Parameters
	calInitParams.strLensBinFilePath = "..\\DB\\Calibration_lensfppn\\"; // path in English
	calInitParams.strLensRawPath = "..\\DB\\lens_fppn\\"; // path in English
	calInitParams.strLensPrefix = "lensfppn_"; // path in English
	calInitParams.lensDirNum = 4; // checker board direction
	calInitParams.lensAvgNum = 10; // average number for each direction

	// Wiggling Cal. Parameters
	calInitParams.strWiggBinFilePath = "..\\DB\\Calibration_wigg\\"; // path in English
	calInitParams.strWiggRawPath = "..\\DB\\wiggling\\"; // path in English
	calInitParams.strWiggPrefix = "wigg_"; // path in English
	calInitParams.wiggDisStr = 0; // time-delay start value
	calInitParams.wiggDisEnd = 1392; // time-delay end value
	calInitParams.wiggDisStep = 72;  // time-delay step value
	calInitParams.wiggDisCap = 550;    // input image initial distance in mm
}
```

```
int main()
{
	RET_CODE ret = E_ERROR_TOF_SUCCESS;

	CAL_INIT_PARAM calInitParams;
	// Initialize calibration control paramters
	 setInitParams(calInitParams);

	// Run calibration tool
	ret = Calibrate_ToF(calInitParams);
	if(ret == E_ERROR_TOF_SUCCESS)
		cout<<endl<< ">>>>>>> Tof Calibration is done <<<<<<< " << endl;
	else
	{
		ret = GetError_ToF(ret);
		cout<<endl<< ">>>>>>> Tof Calibration is fail <<<<<<< " << endl;
	}

	return 0;
}
```

3.3.4 **Calibration Output**

Calibration library generates each calibration block result and merge full calibration results in a binary format. Each calibration block results can be used internally. Also full calibration results can be used to write in a memory such as EEPROM on the camera module.

Note that in order to build full calibration binary, temperature calibration binaries from the temperature drift tool should be included in same calibration folder previsouly set in XML.

3.3.4.1 Calibration intermediate binary output in each calibration block

- temp_f1.bin/ temp_f2.bin: temperature calibration results in binary format for each modulation frequency corresponding memory map
- wigg_f1_tbl.bin/ wigg_f2_tbl.bin: wiggling calibration results in binary format for each modulation frequency corresponding memory map
- fppn_f1_tbl.bin/ fppn_f2_tbl.bin: FPPN calibration results in binary format for each modulation frequency corresponding memory map
- lens.bin: lens calibration result in binary format

3.3.4.2 Calibration final output

- _ToF_cal.bin : merged full calibration final binary result according to memory map.

3.4 **Board Calibration Tool**

Board configuration difference between calibration and validation can exist shown in Figure 3.12. The length difference of conntection between Rx and Tx can result in global depth offset during validataion. In order to remove global offset, board calibration should be additionally conducted before validation.

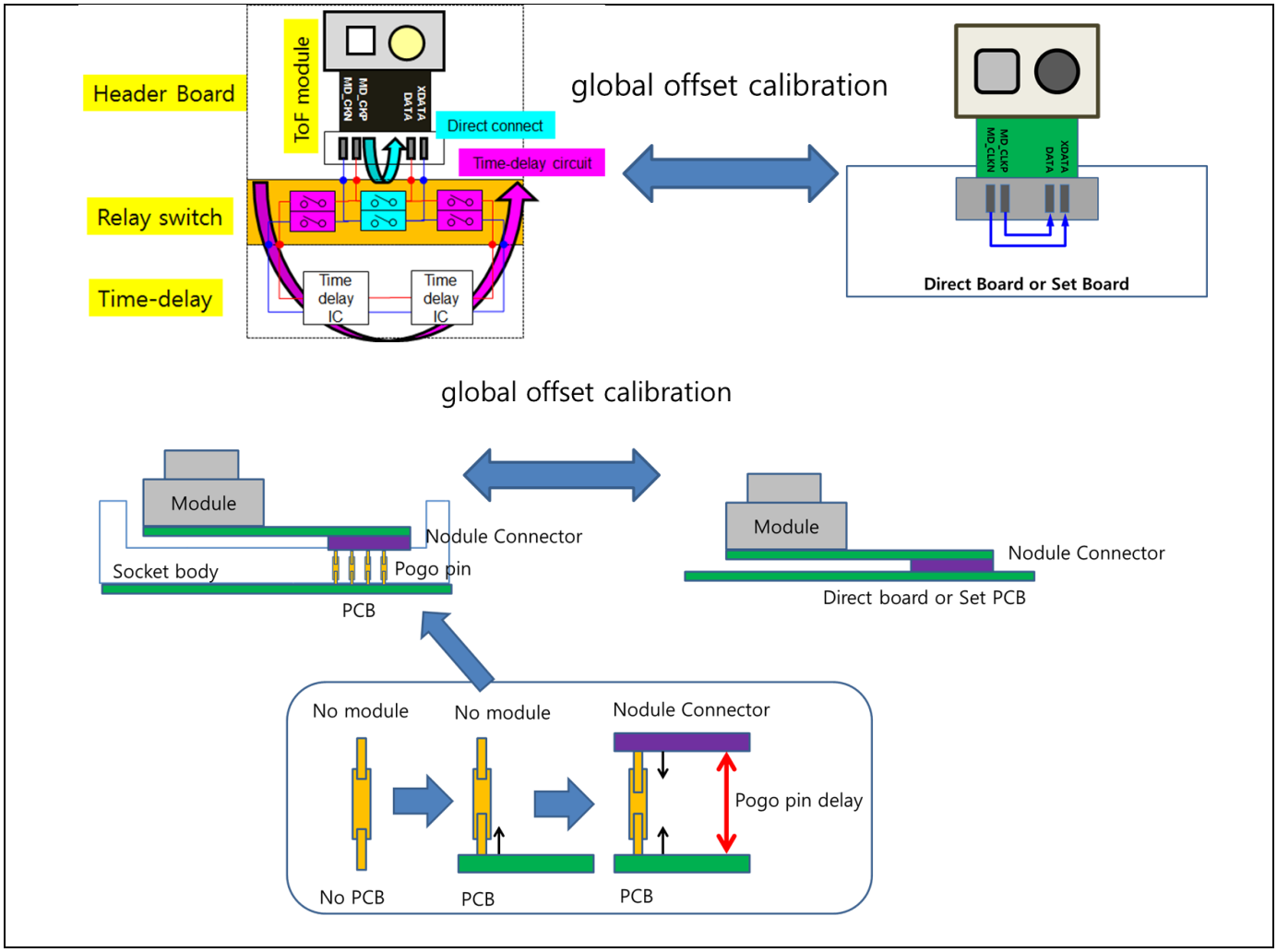

Figure 3.12 Different Board Conifguration Examples

3.4.1 **Procedure**

There are two type board calibrations using different charts for input images, checker board chart and plain chart respectively. Global depth offset can be acquired after board calibration and then this result can be applied during validation.

Board calibration procedure using checkerboard chart is shown in Figure 3.13. It is similar to Lens-FPPN Calibration Procedure (2.3.1). Board calibration procedure using plain chart is also described in Figure 3.14. Plain

chart method is used when the switch of time-delay board is not stable. Unstable switch makes the delay unstable and then constant delay values cannot be applied to direct board. In this case, validation chart is used for calibrating board delay when validation step.

Note that in order to choose method, please set input chart direction number in XML.

Board Calibration Procedure using checker board chart (as shown in Figure 2.8)

1) Obtain the checker chart images from validataion board and set images as input to calibration tool

2) Set and applied calibration results of calibration board into checker images of validation board.

3) Global depth offsets are calculated between these board configurations.

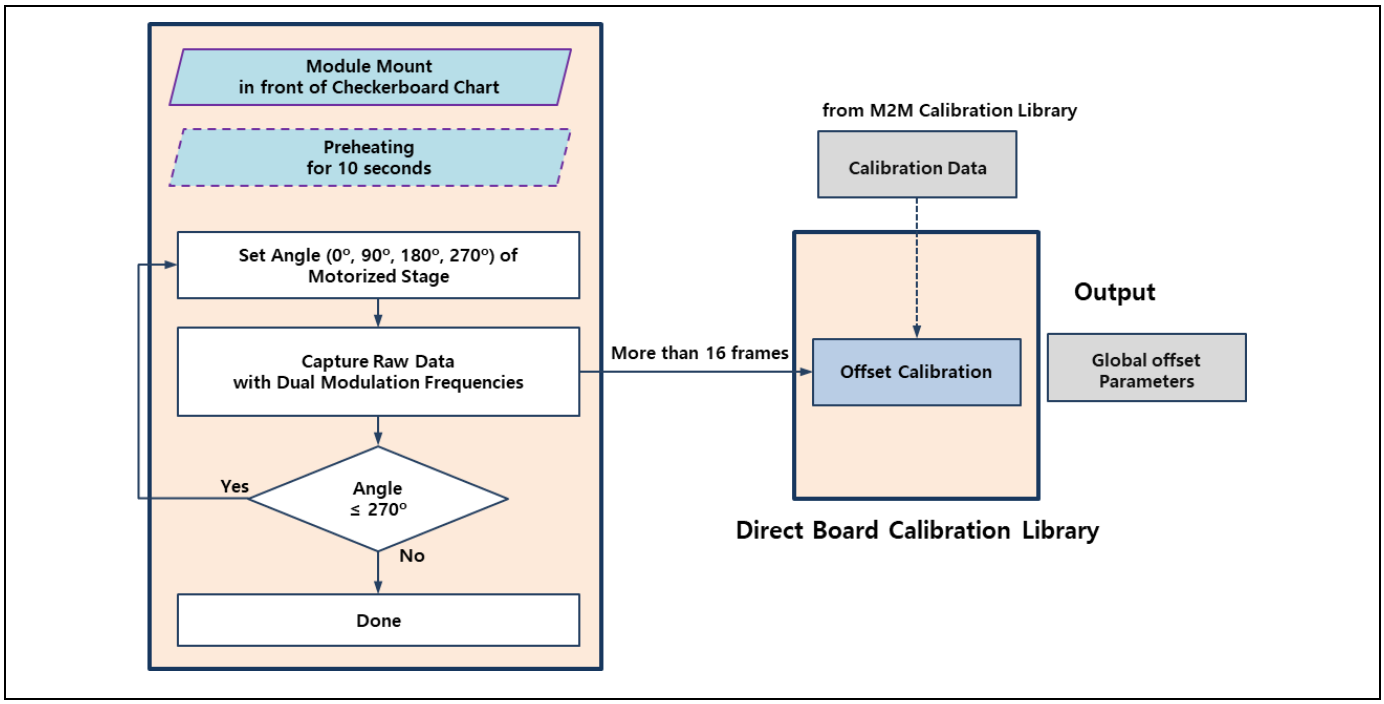

Figure 3.13 Board Calibration Procedure

Board Calibration Procedure using plain chart when validation step

1) Obtain the plain chart images at specific distance (currently, 600mm is recommended) from validataion board and set images as input to calibration tool, also this specific distance should be set in XML for board calibration library.

2) Set and applied calibration results of calibration board into plain chart image of validation board.

3) Global depth offsets are calculated between these board configurations.

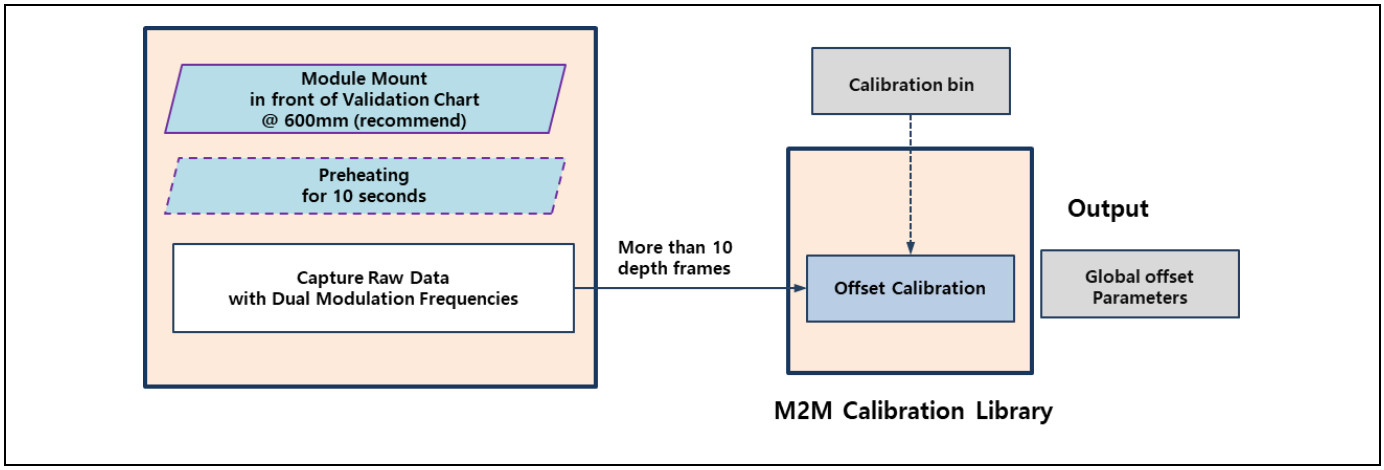

Figure 3.14 Board Calibration Procedure

3.4.2 **Deliverables**

Below items are deliverables to calibrate board configuration difference.

(1) Opencv_world341.dll : OpenCV library

(2) ToFCalParamBoard.xml : calibration opearation configuration file in XML

(3) ToF_Calibration_DLL_x64.dll/ ToF_Calibration_DLL_x64.lib : calibration dynamic libraries for x64

(4) ToF_Calibration_DLL_Win32.dll / ToF_Calibration_DLL_Win32.lib: calibration dynamic libraries for x86

(5) ToF_Calibration_EXE_Board_x64/ToF_Calibration_EXE_Board_Win32 : execution example program

(6) ToF_Calibration_API.h : API definition C++ header file

3.4.3 **Board Calibration Configuration file**

In order to control and set the proper board calibration environmnent, ToFCalParam.xml is provided same as calibration tool. Only paramters in gray boxes are used. ToFCalParam.xml includes input/output interfaces and specific paramaters in each calibration module. Note that the path of folder should be configured in English only and all parameters should be enclosed by quotation marks and should use muti-byte character set for configuration.

```
<!-- Common Calibration Parameters -->
<COMM_CalFinalBinFilePath>"..\DB\Calibration_result\"</COMM_CalFinalBinFilePath>
<COMM_EnableWigg>1</COMM_EnableWigg>
<COMM_EnableLens>1</COMM_EnableLens>
<COMM_EnableFPPN>1</COMM_EnableFPPN>
<COMM_EnableExtE>1</COMM_EnableExtE>
<COMM_ImageWidth>640</COMM_ImageWidth>
<COMM_ImageHeight>480</COMM_ImageHeight>
<COMM_ModulationFrequency1>100</COMM_ModulationFrequency1>
<COMM_ModulationFrequency2>80</COMM_ModulationFrequency2>
<COMM_PixelSize>7</COMM_PixelSize>
<COMM_ModuleTTL>5</COMM_ModuleTTL>
<COMM_ValidationDistance>600</COMM_ValidationDistance>
<COMM_GlobalOffsetFrequency1>0</COMM_GlobalOffsetFrequency1>
<COMM_GlobalOffsetFrequency2>0</COMM_GlobalOffsetFrequency2>
<!-- Temperature Calibration Parameters -->
<TEMP_TempBinFilePath>""</TEMP_TempBinFilePath>
<TEMP_DriftBinFilePath>""</TEMP_DriftBinFilePath>
<!-- Lens & FPPN Calibration Parameters -->
<LENS_BinFilePath>""</LENS_BinFilePath>
<LENS_RawFilePath>"..\DB\lens_fppn\"</LENS_RawFilePath>
<LENS_RawFilePrefix>"lensfppn_"</LENS_RawFilePrefix>
<LENS_DirNum>4</LENS_DirNum>
<LENS_AvgNum>10</LENS_AvgNum>
<!-- Wiggling Calibration Parameters -->
<WIGG_BinFilePath>""</WIGG_BinFilePath>
<WIGG_RawFilePath>""</WIGG_RawFilePath>
<WIGG_RawFilePrefix>""</WIGG_RawFilePrefix>
<WIGG_DisStr>0</WIGG_DisStr>
<WIGG_DisEnd>1392</WIGG_DisEnd>
<WIGG_DisStep>72</WIGG_DisStep>
<WIGG_DisCap>550</WIGG_DisCap></CalParameter>
```

Figure 3.15 Board Calibration Configuration XML example

3.4.3.1 Common Calibration Parameters

Common paramters for all calibration modes can be set.

- COMM_FinalCalFilePath: calibration binary input file location of calibration board in English
  - Default: “..\DB\Calibration_result\”
- COMM_ImageWidth: final depth output image width
  - Default : 640
- COMM_ImageHeight: final depth output image height
  - Default : 480
- COMM_ModulationFrequency1: first modulation frequency value in MHz

- Default : 100

- COMM_ModulationFrequency2: second modulation frequency value in MHz
  - Default : 80
- COMM_PixelSize: pixel size of RX sensor in um
  - Default : 7
- COMM_ModuleTTL: module TTL in mm
  - Default : 5
- COMM_ValidataionDistance: validataion distance for board calibration in mm
  - Default : 600

3.4.3.2 Lens & FPPN Calibration Paramters

Lens & FPPN calibration parameters for validation board can be set.

- LENS_RawFilePath: lens & fppn calibration input file location of validation board in English
  - Default: "..\DB\lensfppn\"
- LENS_RawFilePrefix: prefix of input raw file name
  - Default : "lensfppn_" (actual raw file name is set as "lensfppn_~~.raw")
- LENS_DirNum: total calibration input image direction number same as converted depth image number
  - Default : 4 (if use checker board, set to 4 direction(0°, 90°, 180° and 270°) as default, or or use plain chart set to 1. Also actual raw file number is same as LENS_Num * 4 in 4 Tab, shuffle and dual frequency mode)
- LENS_AvgNum: average calibration input image number for each direction same as converted depth image number
  - Default : 10 (actual raw file number is same as LENS_Num * 4 in 4 Tab, shuffle and dual frequency mode)

3.4.4 **Board Calibration API**

Only one main API is needed to execute calibration for board calibration by calling in execution program. Board calibration tool can be controlled by initial paramters.

```
typedef enum _RET_CODE {
	E_ERROR_TOF_SUCCESS		=		(0x00000000),
	E_ERROR_TOF_INIT_COMM		=		(0x00000110),
```

```
        E_ERROR_TOF_INIT_TEMP               =       (0x00000120),
        E_ERROR_TOF_INIT_WIGG               =       (0x00000130),
        E_ERROR_TOF_INIT_FPPN               =       (0x00000140),
        E_ERROR_TOF_INIT_LENS               =       (0x00000150),
        E_ERROR_TOF_INIT_BOARD              =       (0x00000160),
        E_ERROR_TOF_INIT_PHONE              =       (0x00000170),
        E_ERROR_TOF_RUN_COMM                =       (0x00000210),
        E_ERROR_TOF_RUN_TEMP                =       (0x00000220),
        E_ERROR_TOF_RUN_WIGG                =       (0x00000230),
        E_ERROR_TOF_RUN_FPPN                =       (0x00000240),
        E_ERROR_TOF_RUN_LENS                =       (0x00000250),
        E_ERROR_TOF_RUN_BOARD               =       (0x00000260),
        E_ERROR_TOF_RUN_PHONE               =       (0x00000270),
        E_ERROR_TOF_RUN_INPUT               =       (0x00000280),
        E_ERROR_TOF_RUN_OUTPUT              =       (0x00000290),
        E_ERROR_TOF_NOT_SUPPORT     =       (0x00000A10),
}RET_CODE;

typedef struct _CAL_INIT_PARAM {
        // Common Cal. Parameters
        char* strCalBinFilePath;
        int enWigg;
        int enLens;
        int enFPPN;
        int enExtE;
        int imgWidth;
        int imgHeight;
        float fModulationFreq1;
        float fModulationFreq2;
        float pixSize;
        float modTTL;
        float valDist;
        float globalOffsetFreq1;
        float globalOffsetFreq2;

        // Temperature Cal. Parameters
        char* strTempBinFilePath;
        char* strTempDriftBinFilePath;

        // Lens & FPPN Cal. Parameters
        char* strLensBinFilePath;
        char* strLensRawPath;
        char* strLensPrefix;
        int lensDirNum;
        int lensAvgNum;

        // Wiggling Cal. Parameters
        char* strWiggBinFilePath;
        char* strWiggRawPath;
        char* strWiggPrefix;
        int wiggDisStr;
        int wiggDisEnd;
        int wiggDisStep;
        int wiggDisCap;
```

```
}CAL_INIT_PARAM, *PCAL_INIT_PARAM;

TOF_CALBRATION_API RET_CODE Calibrate_ToF_Board(const CAL_INIT_PARAM& calInitParams);
TOF_CALBRATION_API RET_CODE GetError_ToF(const RET_CODE errCode);
```

Variables

- `RET_CODE` : Pre-defined return value of API functions
- `CAL_INIT_PARAM` : calibration initial structure including variables which are same as XML paramters, please refer to calibration configuration file section

Functions

- `RET_CODE Calibrate_ToF_Board (CAL_INIT_PARAM& calInitParams)`
  - Run board calibration tool
  - Return : RET_CODE (refer to variables)
  - Argument :

    CAL_INIT_PARAM : calibration initial paramters controlling calibration tool
- `RET_CODE GetError_ToF(const int errCode)`
  - Display and described calibration tool with pre-defined error codes
  - Return : RET_CODE (refer to variables)
  - Argument :

    RET_CODE (refer to variables)

Calibration execution example is given below. Firstly control paramters are set and then function `Calibrate_ToF_Board()` calling is carried out. Finally calibration execution result can be monitored by checking return value.

```
#include <iostream>
#include <string>

#include "ToF_Calibration_API.h"

using namespace std;

void setInitParams(CAL_INIT_PARAM& calInitParams)
{
	// Common Cal. Parameters
	calInitParams.strCalBinFilePath = "..\\DB\\Calibration_result\\"; // path in English
```

```
	/* No need for board cal */ calInitParams.enWigg = 1; //on : 1, off : 0
	/* No need for board cal */ calInitParams.enLens = 1; //on : 1, off : 0
	/* No need for board cal */ calInitParams.enFPPN = 1; //on : 1, off : 0
	calInitParams.enExtE = 1; //on : 1, off : 0
	calInitParams.imgWidth = 640; // depth output width
	calInitParams.imgHeight = 480; // depth output height
	calInitParams.fModulationFreq1 = 100; // Mhz
	calInitParams.fModulationFreq2 = 80; // Mhz
	calInitParams.pixSize = 7; // um
	calInitParams.modTTL = 5; // mm
	calInitParams.valDist = 600; // mm
	/* No need for board cal */ calInitParams.globalOffsetFreq1 = 0; // 0 ~ 1
	/* No need for board cal */ calInitParams.globalOffsetFreq2 = 0; // 0 ~ 1

	// Temperature Cal. Parameters
	/* No need for board cal */ calInitParams.strTempBinFilePath = "";
	/* No need for board cal */ calInitParams.strTempDriftBinFilePath = "";

	// Lens & FPPN Cal. Parameters
	/* No need for board cal */ calInitParams.strLensBinFilePath = ""; // path in English
	calInitParams.strLensRawPath = "..\\DB\\lens_fppn\\"; // path in English
	calInitParams.strLensPrefix = "lensfppn_"; // path in English
	calInitParams.lensDirNum = 4; // if use checker board, set to 4 direction, or use plain
chart set to 1
	calInitParams.lensAvgNum = 10; // average number for each direction

	// Wiggling Cal. Parameters
	/* No need for board cal */ calInitParams.strWiggBinFilePath = ""; // path in English
	/* No need for board cal */ calInitParams.strWiggRawPath = "";
	/* No need for board cal */ calInitParams.strWiggPrefix = ""; // path in English
	/* No need for board cal */ calInitParams.wiggDisStr = 0; // time-delay start value
	/* No need for board cal */ calInitParams.wiggDisEnd = 1392; // time-delay end value
	/* No need for board cal */ calInitParams.wiggDisStep = 72;  // time-delay step value
	/* No need for board cal */ calInitParams.wiggDisCap = 550;    // input image initial
distance in mm
}

int main()
{
	RET_CODE ret = E_ERROR_TOF_SUCCESS;

	CAL_INIT_PARAM calInitParams;
	// Initialize calibration control paramters
	setInitParams(calInitParams);

	// Run calibration tool
	ret = Calibrate_ToF_Board(calInitParams);
	if (ret == E_ERROR_TOF_SUCCESS)
		cout << endl << ">>>>>>> Tof Board Calibration is done <<<<<<<" << endl;
	else
	{
		ret = GetError_ToF(ret);
		cout << endl << ">>>>>>> Tof Board Calibration is fail <<<<<<<" << endl;
	}
```

```
        return 0;
}
```

3.4.5 **Board Calibration Output**

ToF board calibration library generates global depth offset between different board configurations. If board calibration succeed, global offsets for each frequency are displayed normalized depth (0~1) and in mm. These values should be included in calibration XML as globalOffsets.

3.5 **DEPS Tool**

DEPS tool and document will be updated.

References

[1] https://en.wikipedia.org/wiki/Thermoelectric_cooling

[2] https://www.onsemi.com/pub/Collateral/NB6L295-D.PDF

[3] Z. Zhang, "Flexible Camera Calibration by Viewing a Plane from Unknown Orientations," ICCV99, vol. 1, pp. 666-673, 1999.5.

[4] D. Brown, "Decentering distortion of lenses". Photogrammetric Engineering. 32 (3), pp.444-462, 1966.

[5] https://opencv.org/