



Title: Multi-sensor Operation

Purpose

This technical note demonstrates how a synchronized network of three or more Gocator sensors can be configured for system-wide exposure multiplexing, through the Gocator's Software Development Kit.

Equipment Used

Any Gocator model sensor
Master 400 (or higher)

Sensor Features Used

Software Development Kit
Ethernet Output

Table of Contents

1 Introduction	2
2 Setup	2
2.1 Installation	3
2.2 IP Configuration	4
2.3 Gocator Configuration	4
2.4 Master Selection	4
2.5 Operation Mode	5
2.6 Triggering	5
2.7 Exposure	6
2.8 Active Area	7
2.9 Calibration	7
2.10 Saving Configuration	8
2.11 Output	8
3 Multiplexing Rules for the Gocator	9
3.1 Simplified Exposure Rules	9
3.2 General Rules for Single Exposure	9
3.3 Dynamic Exposure Rules	10
3.4 Multiple Exposure Rules	10
3.5 Encoder Trigger Mode	10
4 Control Using the Gocator SDK	11
4.1 Initialization	11
4.2 Discover and Connect to Gocators	11
4.3 Configure the Multiplexing	12
4.4 Synchronously Start Gocators	13
4.5 Receive Data	13
4.6 Stop Gocators	14
4.7 Process Data	14



1 Introduction

This technical note explains how several Gocator sensors can be used together in a synchronized system in order to scan large objects at high resolution. The Gocator's built-in Buddy concept can handle most dual sensor configurations, but when creating systems with three or more sensors, the user has to work with the Gocator Software Development Kit (SDK) to build a custom solution.

A common requirement with multi sensor systems is to be able to precisely control the timing of each sensor to avoid interference in areas where the measurement zones overlap, often referred to as “cross talk” between sensors. The solution is to identify groups of sensors that are independent of each other (i.e. without cross talk issues) and logically group them into *banks*. Measurements in the system are then orchestrated in such a way that each bank take turn scanning the object, i.e. the banks are *multiplexed*.

Multi sensor systems also often require a system calibration procedure in order for all sensors to report data in a common coordinate frame. A calibration target of known shape and dimension is positioned in the system allowing for a calibration procedure to calculate translation parameters for each sensor. During run-time the translation parameters are then applied to the data points to create a single 3D model of the scanned object.

This document focuses on the Gocator SDK and multiplexing rules, while the system calibration topic is too broad to capture within the scope of this document. The content is divided into three sections.

- **Setup** describes how to physically connect the sensors in a network and how to configure the sensors using the Gocator's web-based user interface. Simply connect to the sensors through a web browser on any operating system. No driver installation required.
- **Multiplexing Rules** explain the detailed procedure for determining the exact timing of each sensor in a network of Gocator's to avoid cross talk. A simplified case, as well as the theoretical general case is covered. Both scenarios come with practical examples.
- **Gocator SDK** outlines how the user can implement their custom multiplex timing in software using the Software Development Kit. This technical note is bundled together with an open source example code project that the user can study, compile and run in parallel to reading this section. The code snippets in this document are taken directly from the example source code.

2 Setup

There are many potential Gocator applications that require three or more sensors and there are almost as many variations on the physical layout of such systems. Figure 1 shows a few real world examples.

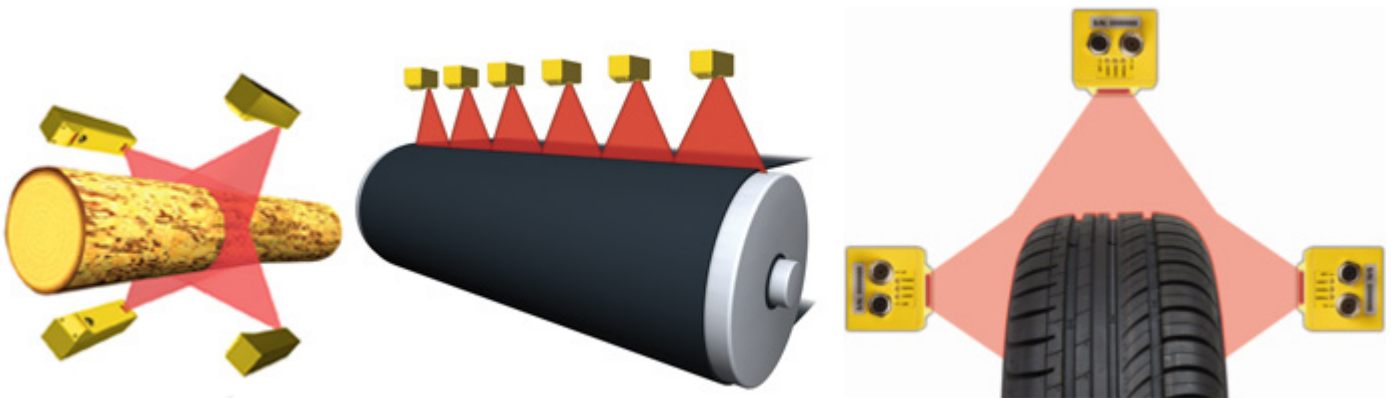


Figure 1: Multi Sensor Applications. Log scanning (left), Web scanning (middle), Tire scanning (right)



In the log scanning example above with four sensors, neighbouring sensors are suffering from cross talk. This means that the four sensors need to be logically divided into two banks, each consisting of two diagonally opposing sensors. As an example, assume that the sensors are operating at 100 Hz. This means that the time to acquire a single scan, referred to as the *frame period*, is 10 milliseconds long. Multiplexing will then be achieved by splitting the 10 ms frame period into two halves, each 5 ms long, giving one independent time slot for each bank. Figure 2 shows the system configuration and timing diagram of this simple example. Opposing sensors A and C belong to Bank 0, whereas sensors B and D belong to Bank 1.

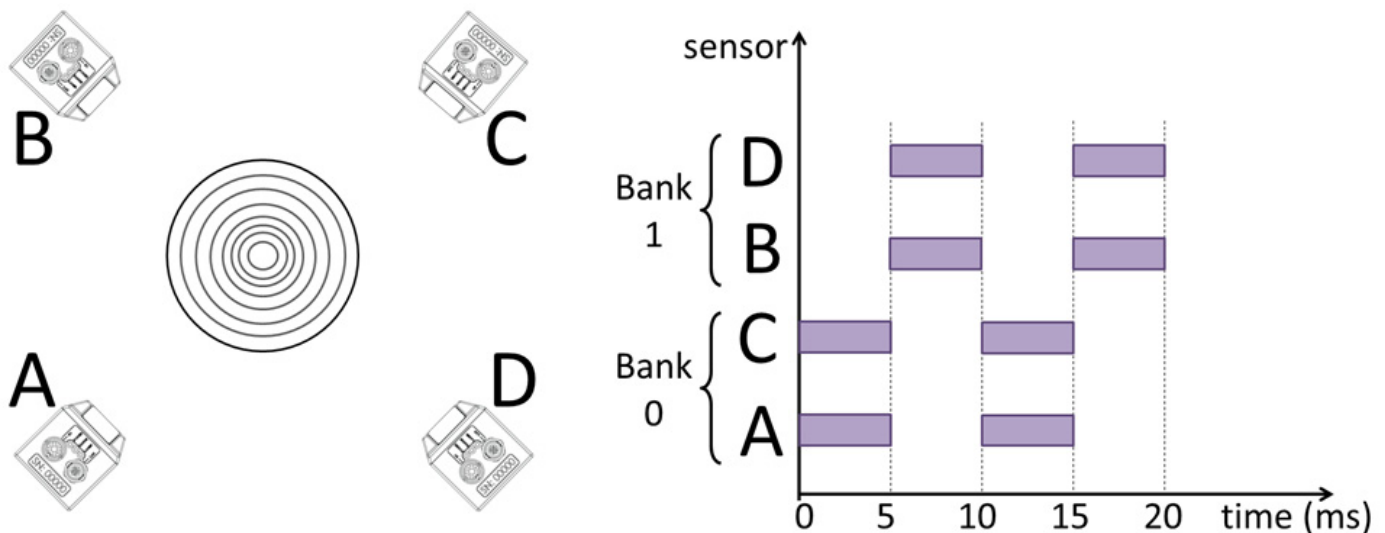


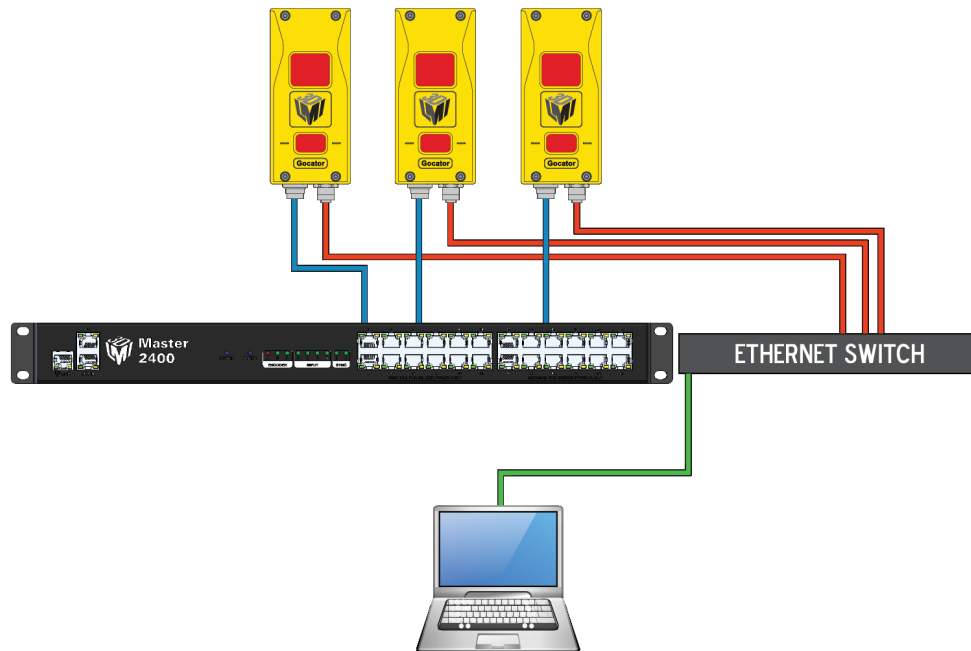
Figure 2: Example Timing Diagram for Log Scanning Application

2.1 Installation

Gocators are networked through a Master and an Ethernet switch. The Master accepts encoder and external input and provides power, laser safety and synchronization to the Gocators. The Ethernet switch provides the data communication paths for connecting Gocators to the client computer.

For a multi-sensor system, Master 400, 800, 1200 or 2400 could be used. These Masters have the same functionality and are different by the number of sensors that can be connected to it.

A multi-system system is connected as follows:

**Figure 3: Gocator Cable Connection**

Refer to section *Master 400/800* in the Gocator User Manual on how to connect power, laser safety and encoder signals to the Master 400. A very important consideration is to ensure that the shield of the Gocator cord set cable must be properly connected to earth ground to avoid electrostatic discharge.

2.2 IP Configuration

All Gocators are shipped with a default IP address of 192.168.1.10. Before connecting the Gocator to the Ethernet network, each Gocator requires a unique IP address.

Refer to section *Connecting to a New Sensor* in the User Manual on how to change the IP address of a Gocator.

2.3 Gocator Configuration

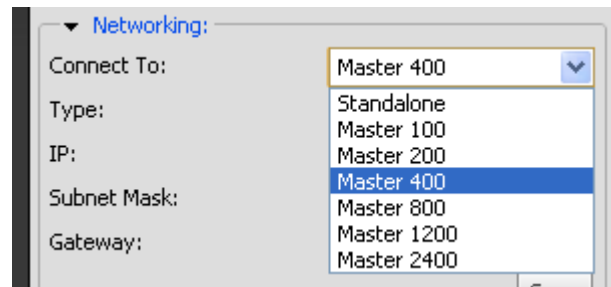
In a multi-sensor system, each Gocator is configured independently by logging in through a web browser using the Gocator's unique IP address.

Note that the Gocators are *not* configured to operate in Buddy mode.

The following sections explain the configurations that are required for each Gocator.

2.4 Master Selection

Select the correct Master model used in the system in the *Connect To* pull-down option under the Connect Page. For a system with three or more sensors a Master 400 or higher is required.

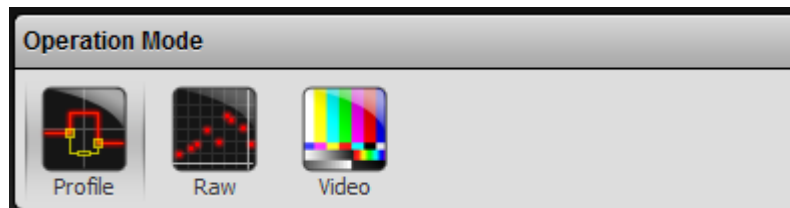
**Figure 4: Connect To Option**

2.5 Operation Mode

The Gocator can deliver profile data in two different operation modes, Profile and Raw. Either mode can be used in a multi sensor scanning application.

In Profile Mode the data has been resampled to an even interval across the laser line (X). This means that the data is delivered as a simple array of height values (Z).

In Raw Mode the data is delivered as unprocessed (X,Z) coordinate pairs. This leaves more processing power open on the sensor, but the user has to typically handle resampling on the client instead.

**Figure 5: Operation Modes**

2.6 Triggering

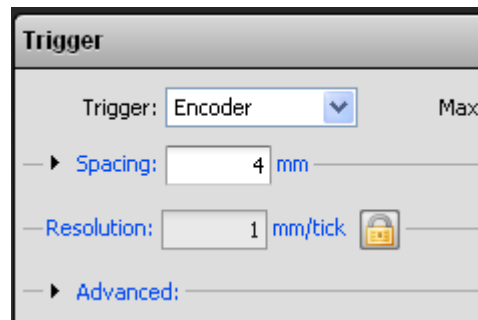
Triggering in a multi-sensor system is typically either time based or encoder based. Both options are supported in the Gocator.

With time based scanning, the sensors are simply free running at the specified speed. With encoder based scanning, the sensors are acquiring data at the specified distance interval. In either case, the delivered profile data is always tagged with both a timestamp and an encoder stamp (if wired into the Master) and it's up to the user to implement a suitable resampling strategy.

Encoder spacing is specified in mm, and the translation from encoder ticks to mm is controlled by the Encoder Resolution. The Gocator performs 4x sampling on the encoder signals, so the encoder value will increase by 4 for each physical forward tick from the encoder.

For example, to generate a profile at each physical encoder tick, the encoder resolution should be set to 1 mm/tick and the spacing set to 4.

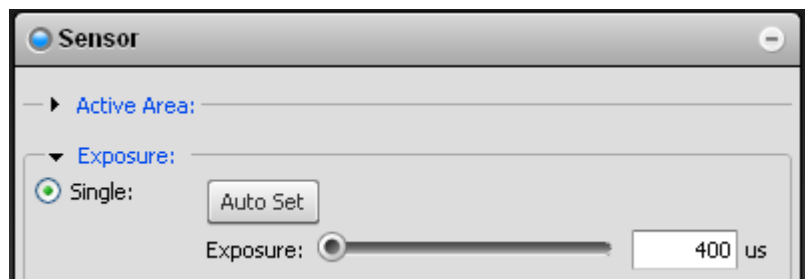
The Trigger option, Encoder Spacing and Encoder Resolution are configured in the Trigger panel. These parameters can be generated automatically by the sensor's built-in travel calibration, but can also be entered manually.

**Figure 6: Encoder Triggering Setup**

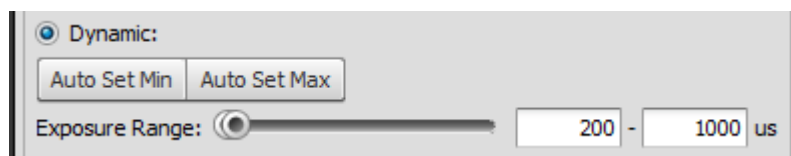
By default, the Encoder Resolution is set to 1 mm/tick. Users can reset the value by unlocking the Resolution parameter and entering their own values.

2.7 Exposure

Exposure is a critical parameter to set correctly in order to achieve good measurement results. Exposure determines the duration of camera and laser on-time. Longer exposures can be helpful to detect laser signals on dark or distant surfaces, but increasing exposure time decreases the maximum scanning speed. Exposure time can be automatically tuned or manually configured. There are three types of exposure modes in the Gocator, Single, Dynamic and Multiple. Under Single exposure the sensor is constantly operating with this setting, regardless of the characteristics of the target surface.

**Figure 7: Single Exposure Settings**

Dynamic exposure can be used when the surface target appearance is varying significantly in the direction of travel. In this mode there is a feedback loop that automatically adjusts the exposure time from frame to frame as the target is being scanned, within the configured limits. The feedback loop algorithm is implemented in hardware and has a very minimal effect on the maximum scanning speed of the sensor. However, this feature should not be used without some testing to ensure it performs as expected in the desired application.

**Figure 8: Dynamic Exposure Settings**



Multiple exposures can be used when the surface target is varying significantly across the Field of View of the sensor. Up to 5 exposures can be defined with each set to a different exposure level. When enabled, the laser profile from each exposure is combined into a single laser profile, referred to as the composite profile.

For each exposure the sensor will perform a complete scan at the current frame rate making the effective frame rate slower. For example, if two exposures are selected then the Gocator speed will be $\frac{1}{2}$ of the single exposure frame rate. The sensor will perform a complete scan for each external input or encoder trigger.

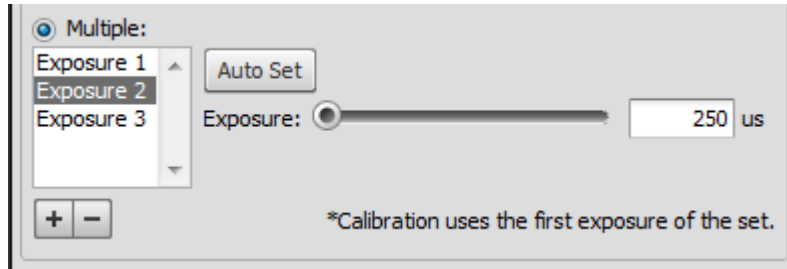


Figure 9: Multiple Exposure Settings

2.8 Active Area

The size of the active area is proportional to the maximum profiling speed achievable by the Gocator.

The active area is configured in the Sensor panel. Users can either configure the field of view graphically by selecting the Select button, or entering numerical values with the advanced panel.

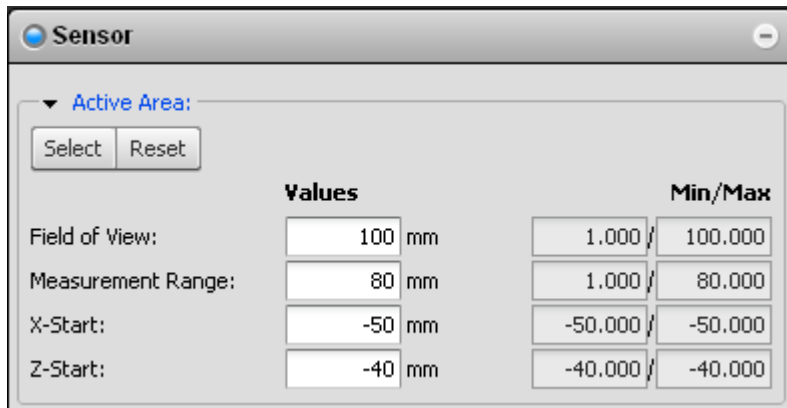


Figure 10: Active Area Configuration

Refer to section *Active Area* in the User's Manual for more details on how to setup the active area.

2.9 Calibration

The Gocator comes with several options for calibration. Alignment calibration defines a zero reference for the real world co-ordinates reported by the Gocator. Travel calibration uses a calibration disk of a known size to establish the relationship between the encoder and the direction of travel (mm/tick), as well as performing an alignment calibration.

If a Dual Sensor system has been setup with two sensors (Main and Buddy) there is built-in system calibration support to a common world coordinate frame.



For a detailed description of these procedures, refer to the User's Manual.

2.10 Saving Configuration

After configuring the Gocator, the configuration needs to be saved so that it will be automatically used after the Gocator is power cycled. To save the configuration, enter the configuration name, and then press Save in the tool bar.



Figure 11: Saving Configuration

The last saved configuration will be the one loaded by default. The choice of the default configuration can be changed in the File panel under the Connection Page.

2.11 Output

The data output is sent over Ethernet when configured in the Output Page by checking the *Main* sensor option. For multi-sensor applications, each Gocator must be configured to output their profile data to the client by checking the *Main* option in the Ethernet panel.

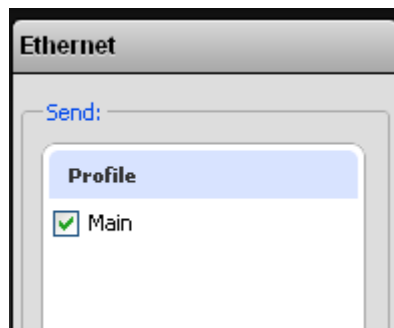


Figure 12: Ethernet Output Option in Output Panel



3 Multiplexing Rules for the Gocator

As described in the introduction, the basic principle to avoid sensor cross-talk is to employ a multiplexing strategy. Sensors that can operate independent of each other, without risk of cross talk, are grouped into logical banks. The overall system speed determines the length of the time period for one data acquisition cycle, the so called frame period. This frame period is then split up into shorter time slots, one for each bank of sensors to operate within. In practice, this means that multiplexing is achieved by aligning the exposure of each bank such that the exposure periods don't overlap, as explained by Figure 2 on page 3. The Exposure Delay parameter in the sensor configuration is the key mechanism to achieve various multiplexing behaviour

To correctly determine the exact timing configuration for every sensor in a larger system, use the rules described in these sections

3.1 Simplified Exposure Rules

In most applications all sensors in the system are scanning the same type of target surface, e.g. a log, the rubber of a tire, a road surface etc. The characteristics of the surface typically determine the minimum exposure time needed for a reliable measurement and *all sensors in the system are running with the same exposure time*.

In most applications there is also a certain speed at which the system has to run in order to solve the problem at hand. This required speed then sets the available frame period.

In this simplified scenario the exposure rule is easy. The exposure time cannot be longer than:
(frame period / number of banks).

The actual multiplexing is then achieved by delaying the start of the exposure for each bank.

Example

A system of three Gocator 2080 sensors is scanning a log and the sensors are mounted in a ring with 120° separation. Each sensor's FOV is overlapping with both its neighbours causing cross-talk, meaning that three banks are required. The application requires the sensors to run at 300 Hz, which means that the frame period is approximately 3333 µs. According to the simplified rule, the exposure time for each sensor cannot be longer than $3333 \mu\text{s} / 3 \text{ banks} = 1111 \mu\text{s}$. With a small margin added, a safe upper limit for the exposure time in this system is 1100 µs.

The multiplexing in this system is then implemented such that sensor 0 is exposing in the time slot from 0 µs to 1100 µs, sensor 1 is exposing in the time slot from 1111 µs to 2211 µs and sensor 2 is exposing in the time slot from 2222 µs to 3322 µs.

The example source code project included with this document is showing how to use the Gocator SDK to accomplish this exact scenario.

3.2 General Rules for Single Exposure

In this theoretical general case, every sensor in the system may run with a *different* exposure time. Although this scenario is less common in multi sensor systems, it nevertheless provides a detailed understanding for how the Gocator operates. The procedure for establishing settings for multiplexing in this case is:

- Determine the desired exposure time for each sensor in the system.
- Determine how many multiplexing banks are needed.
- Within each bank, make note of the longest exposure time.



- Determine the exposure delay for each bank, such that the delay for a bank is the sum of the longest exposure in each preceding bank. For example, the exposure delay for Bank 2 should be the longest exposure in Bank 0, plus the longest exposure in Bank 1.
- Apply the exposure time and exposure delay to each sensor in the system. These settings will affect the maximum speed at which the Gocator can run. When the exposure settings are applied, the Gocator will automatically calculate this maximum speed and update a property called Maximum Frame Rate.
- Determine the overall frame rate the system can run at. The system will not be able to run faster than the slowest sensor, so query the Maximum Frame Rate property from each sensor in order to determine the slowest maximum frame rate. To *guarantee* that there is no cross talk in the system, add a safety margin of 10 μ s to the frame period, i.e. invert the slowest maximum frame rate to get the period, then add the 10 μ s margin and then invert this result back to a frame rate. This is the fastest overall frame rate the system can run at while honouring all the desired exposure times.

Example

In a system of 7 Gocator 2030 sensors it is determined that 3 banks are needed to avoid sensor cross talk. Application testing has determined that an exposure time between 500 μ s and 1000 μ s is required to get a good measurement off the target. The longest exposure time in Bank 0 is 750 μ s, in Bank 1 it is 900 μ s and in Bank 2 it is 600 μ s. This will give an exposure delay of 0 μ s for all sensors in Bank 0, 750 μ s for Bank 1 and 1650 μ s for Bank 2.

All the exposure times and exposure delays are applied to the 7 sensors and each sensor is then queried for the Maximum Frame Rate. In theory, the slowest sensors in the system should be the one with 600 μ s in Bank 2, since it has the longest exposure delay of 1650 μ s, giving a total frame period of 2250 μ s. This corresponds to approximately 440 Hz, but the result of the query is that the slowest sensor's maximum rate is reported as 300 Hz. The reason for this discrepancy is that the camera in the Gocator 2030 cannot run faster than 300 Hz at full measurement range. The Gocator can be configured to run much faster by manipulating the Active Area settings to narrow down the measurement range, but that's a detailed discussion outside the scope of this document.

3.3 Dynamic Exposure Rules

The rules for multiplexing a system of sensors that are running Dynamic Exposure are identical to the rules for Single Exposure, with one exception. The upper bound limit for the dynamic exposure should be used in place for the single exposure in the rules outlined above in section 3.2.

3.4 Multiple Exposure Rules

When creating a multiplexed system of sensors that use the Multiple Exposure mode, all sensors *must* use the same number of exposure steps. The rules are again the same as for Single Exposure, with one exception. The sum of the exposures of all exposure steps should be used in place for the single exposure in the rules outlined above in section 3.2.

3.5 Encoder Trigger Mode

The calculations outlined above are the same for encoder trigger mode. Exposure and exposure delays are always specified in time. The user only needs to ensure that the *encoder spacing is longer than the frame period*.



4 Control Using the Gocator SDK

The multi sensor system can be configured and controlled using the Gocator SDK. Commands to start, stop, receive data and so on are all available through an API called Go2. In fact, everything that can be done in the web based UI can also be done programmatically via the SDK.

An example project is included with this technical note. Users can reference this project as a template to start developing their application.

The following section walks through the example code to illustrate how to start, stop and collect data from a multi-sensor system.

4.1 Initialization

A call to *Go2Api_Initialize()* is used to initialize the Go2 API SDK.

After the API is initialized, a *Go2System* needs to be constructed for each Gocator in the network.

```
// When working with multiple sensors, a Go2System has to be created for each individual sensor
for (i = 0; i < APP_SENSOR_COUNT; i++)
{
    if ((status = Go2System_Construct(&obj->systemSensor[i])) != GO2_OK)
    {
        printf("Error: Go2System_Construct:%d\n", status);
        return APP_ERROR;
    }
}
```

The array *obj->systemSensor* is an array of *Go2System* objects. After the system is initialized, the next step is to discover and establish a control connection to the Gocator within the network.

4.2 Discover and Connect to Gocators

Each Gocator needs to be connected to separately. Each Gocator is connected to the client using two TCP channels; control and data. The control channel is used to send commands to the Gocator and the data channel is used to receive data from the Gocator.

The function *Go2System_Discover* returns a list of Gocators that are connected in the network.

```
Go2System_Discover(&deviceIds, &addresses, &deviceCount);
```

The function *Go2System_Connect* establishes a control connection to a Gocator.

```
//For each sensor, use the system object to establish a connection
for (i = 0; i < APP_SENSOR_COUNT; i++)
{
    if ((s = Go2System_Connect(obj->systemSensor[i], addresses[i].address, GO2_USER_ADMIN, "")) != GO2_OK)
    {
        printf("Error: Go2System_Connect:%d\n", s);
        return APP_ERROR;
    }
    if ((status = Go2IPAddress_Format(addresses[i].address, text, GO2_COUNT(text))) != GO2_OK)
    {
        printf("Error: Go2IPAddress_Format:%d\n", status);
        return APP_ERROR;
    }
}
```



```
}
```

Similarly, the function *Go2System_ConnectData* establishes a data connection to a Gocator. Additionally, if using the SDK's asynchronous callback method to read the data from the sensor, it is a required step to store the sensor's ID in the context used for the particular callback. This field in the context will then be used in the callback to identify from which sensor the data is delivered. The code snippet below shows how this can be accomplished.

```
// Connect to the data channels of each sensor
for (i = 0; i < APP_SENSOR_COUNT; i++)
{
    //find the serial number and store in context, so that the callback can know
    //from which sensor the data is coming
    sensor = Go2System_SensorAt(obj->systemSensor[i], 0);
    obj->context[i].serialNumber = Go2Sensor_Id(sensor);

    //connect to the data channel and assign callback function
    if ((s = Go2System_ConnectData(obj->systemSensor[i], onData, &obj->context[i])) != GO2_OK)
    {
        printf("Error: Go2System_ConnectData:%d\n", status);
        return APP_ERROR;
    }
}
```

4.3 Configure the Multiplexing

To implement the type of timing scenario described in section 3.1, i.e. the multiplexing of the data capture, the key functions are *Go2System_SetFrameRate*, *Go2Sensor_SetExposure* and *Go2Sensor_SetExposureDelay*. Note that the frame rate is a property of the System object, whereas the exposure settings apply to the sensor object.

```
// configure sensors
for (i = 0; i < APP_SENSOR_COUNT; i++)
{
    // set sensor speed
    if ((status = Go2System_SetFrameRate(obj->systemSensor[i], APP_SENSOR_SPEED)) != GO2_OK)
    {
        printf("Error: Go2System_SetFrameRate:%d\n", status);
        return APP_ERROR;
    }

    sensor = Go2System_SensorAt(obj->systemSensor[i], 0);

    // set exposure time
    if ((s = Go2Sensor_SetExposure(sensor, APP_MAX_EXPOSURE - APP_EXPOSURE_MARGIN)) != GO2_OK)
    {
        printf("Error: Go2Sensor_SetExposure:%d\n", s);
        return APP_ERROR;
    }

    // set exposure delay - the actual multiplexing!
    if ((status = Go2Sensor_SetExposureDelay(sensor, i * APP_MAX_EXPOSURE)) != GO2_OK)
    {
        printf("Error: Go2Sensor_SetExposureDelay:%d\n", status);
        return APP_ERROR;
    }
}
```



4.4 Synchronously Start Gocators

With a Master in the system, all Gocators' time and encoder values are synchronized. The system can be configured for time based triggering or encoder based triggering. In order to start all sensors at the exact same moment in time, this start event has to be scheduled at a future point in time (or future encoder count if using encoder triggering).

The function that provides this control is called *Go2System_ScheduledStart*. Since a scheduled start is a one-to-one command, the client has to send the command to each Gocator one by one. To handle the time variation at which each Gocator receives the command, a safety delay is added to the current system time (or current encoder count). The application first obtains the current time (or encoder) value using the *Go2System_GetTime* (or *GetEncoder*) command, adds a delay to the returned value and then issues the *ScheduledStart* command. This sequence is illustrated below.

```
//when starting a multi-sensor system it's a required step to get the current system time from
//one sensor and then schedule a synchronized future start for all sensors
if ((status = Go2System_GetTime(obj->systemSensor[0], &currentTime)) != GO2_OK)
{
    printf("Error: Go2System_GetTime:%d\n", status);
    return;
}
startTime = currentTime + APP_START_DELAY;
for (i = 0; i < APP_SENSOR_COUNT; i++)
{
    if ((status = Go2System_ScheduledStart(obj->systemSensor[i], startTime, GO2_NULL)) != GO2_OK)
    {
        printf("Error: Go2System_SheduledStart:%d\n", status);
        return APP_ERROR;
    }
}
```

The start delay value depends on:

1. The time required to send the commands to all Gocators
2. The start-up time required for each Gocator
3. If operating with encoder trigger, whether the encoder is running when the system is started

If operating with time based triggering, a safe time delay is 1 second *per sensor*.

If operating with encoder trigger *and the encoder is running*, the start delay (in encoder ticks now) should be large enough to cover the same 1 second per sensor. However, if the encoder is standing still a delay of 1 encoder tick is sufficient.

4.5 Receive Data

Profile data can be received from the Gocator using the function *Go2System_ReceiveData*, which is a synchronous blocking call. However in most real-time industrial applications an asynchronous callback method is preferred. The example project included with this technical note is using a callback function *onData*, which shows how to read out profile data from the sensor.

Each Gocator data message contains encoder stamps, profile header and actual measurement data.

```
timeStamp = Go2Data_Timestamp(data);
encoder = Go2Data_Encoder(data);
encoderIndex = Go2Data_EncoderIndexValue(data);

for (j = 0; j < Go2Data_ItemCount(data); ++j)
```



```
{
    Go2Object dataItem = Go2Data_ItemAt(data, j);
    if (Go2Object_Type(dataItem) == GO2_TYPE_PROFILE_DATA)
    {
        short* d;
        unsigned int dataWidth;

        dataWidth = Go2ProfileData_Width(data);
        d = Go2ProfileData_Ranges(dataItem);

        // Have the pointer to the data. Now do something with it!
        ...
        ...
        ...
    }
}
```

Actual profile data is returned as a fixed size array. The pointer to the data array is returned from the call to the function *Go2ProfileData_Ranges*. Depending on whether the Operation Mode is Profile or Raw, the format is slightly different. In the case of resampled data in Profile Mode, each element in the array corresponds to the X location defined by $XOffset + \text{element index} * X \text{ Resolution}$. In the case of unprocessed data in Raw Mode, the X coordinate is sent explicitly in an X,Z coordinate pair.

Z values can be converted to real world coordinates by $ZOffset + \text{value} * ZResolution$.

4.6 Stop Gocators

Each Gocator can be stopped by sending the Stop command to each Gocator, using the function *Go2System_Stop*. The data channel should also be disconnected.

```
for (i = 0; i < APP_SENSOR_COUNT; i++)
{
    if ((status = Go2System_Stop(obj->systemSensor[i])) != GO2_OK)
    {
        printf("Error: Go2System_Stop:%d\n", status);
        return APP_ERROR;
    }
    if ((Go2System_DisconnectData(obj->systemSensor[i])) != GO2_OK)
    {
        printf("Error: Go2System_DisconnectData:%d\n", status);
        return APP_ERROR;
    }
}
```

4.7 Process Data

Data processing itself is not discussed in detail within the scope of this document. It can vary dramatically depending on the nature of each particular application, but the basic steps are often:

- Data buffering and memory management
- Transformation of profile data into World Coordinates
- Resampling to an even coordinate grid
- Application specific algorithms to produce final results

Note that when configuring a multi sensor system to multiplex the data capture, the data is not acquired from the exact same location in the direction of travel (Y). The standard solution to this dilemma is to employ a resampling strategy to an even interval in Y that is the same across the whole system.