API Documentation

1.Background

2.Preparation and consideration

3.Getting started with API

4.Getting data from flat scan

5.Setting parameters for flat scan

1. Background
   1. Scope

This document is a specification for the API that can be used with LZR®-FlatScan.

* 1. Definitions, acronyms and abbreviations
  + SE: Self explanatory
  + HS: High-Speed mode
  + HD: High-Definition mode
  + LZR®-FlatScan: Detection Device
  + MDI: Measured Distance Information
  + FlatScan SDK: the program that acts as the controller
  + Host machine: the computer or server that runs the FlatScan SDK

1. Preparation and Consideration
   1. Preparation

Read the *README.md* file.

Sensors are connected to the controller.

Serial port number is specified.

* 1. Consideration

The choice of programming language python is based on following reasons:

* Concise nature of the language
* FlatScan sensors are soft real-time systems. A soft real-time system is defined by a system whose operation is degraded if results are not produced according to the specified timing requirement. Specific requirements should be found in the requirements document <https://docs.google.com/document/d/12kgEX397-gVZVA9J_FqP63cl6bP9HGk2nurZ5oEu87E/edit?usp=sharing>
* With HS mode, the measured distance information is sent every 10.75ms. Python is able to handle such frequency.

1. Files
   1. Directory

Relevant files are in the following directory:

source\_code \ flatscan-sdk-main \ api

* 1. *flatscan\_parameters*

This is essentially a header file with parameters (pound defines) and function declarations. The parameters are used in *flatscan\_utils*.

This customer should not include this file as a library in their code.

* 1. *flatscan\_utils*

This file contains code for handling low-level communication with the sensor. Functions defined in this file are called in *flatscan\_api*.

This customer should not include this file as a library in their code.

* 1. *flatscan\_api*

This customer should include this file as a library in their code.

*Example:* **import** sys

sys.path.append('C:/source\_code/flatscan-sdk-main/api/')

**import** flatscan\_api

By including this file, the customer can call functions defined in the class, Flatscan. This class contains API functions intended for use by the customer. These API is explored in the next section.

1. Getting Started with API
   1. Introduction

This section describes functions local to file, *flatscan\_api*.

The file contains 2 classes:

* **class Flatscan**
* **class FlatscanReadThread**
  1. Classes
     1. class Flatscan

This class contains the API functions intended for use by the customer.

The class constructor in the API is as follows:

**class Flatscan:**

**def**  \_\_init\_\_(self, port\_number, buffer\_maximum\_length = 100, baudrate = BAUDRATE\_DEFAULT)

After including *flatscan\_api* as a library in their code, the customer should create an object of the class, Flatscan:

*Example:* **import** api.flatscan\_api **as** API

com = 1

flatscan\_obj = API.Flatscan(com)

Note that baudrate is an argument of the Flatscan class constructor. However, the baud rate must be 921600 (or left as the default value) when first connecting to the sensor, i.e. when creating the class object.

Once the class object is created, the customer can call API functions as needed:

*Example:*flatscan\_obj.reset\_mdi\_counter()

Functions can be used to get (read) information from the connected sensor, or to set (write) parameters to the connected sensor.

* + 1. class FlatscanReadThread

As the nature of the detection device is a real-time system, the program receives information periodically, even without calling the API explicitly. It does so via the function read\_message().

By default, the FlatScan is in continuous mode, so there is no need to request the start of transmission. Via read\_message(), the following data structures are continuously received by the program:

* MDI\_frame
* heartbeat
* emergency

Note that when API function, get\_mdi() is called, continuous mode is disabled. (data\_field = 0.) Thereafter, only the Flatscan\_heartbeat and emergency data structures will be continuously received, and the MDI\_frame data structure will be received *only* when get\_mdi() is called explicitly.

* 1. API Functions for Getting Data from FlatScan

|  |  |  |
| --- | --- | --- |
| Function | Description | Returns |
| **get\_mdi**() | Request measurements with transfer mode defined (HS or HD mode).  By calling this function, continuous mode is disabled (enter single-shot mode). | **MDI dictionary** |
| **get\_identity**() | Get the identity for the connected FlatScan. | **Identity dictionary** |
| **get\_parameters**() | Get FlatScan parameters. | **Parameters dictionary** |
| **get\_emergency**() | Get the emergency status of the FlatScan. | **Emergency dictionary** |
| **get\_can**() | Get CAN number of the detector (BEA serial number). | **int CAN\_number :**  Points to the variablelocal to *flatscan\_utils* function, flatscan\_parse\_identity\_  msg() |
| **get\_facet**() | For HS mode: Reference of the current mirror facet (1, 2, 3, 4).  For HD mode: 5. | **int ref\_facet** |

* 1. API Functions for Setting Parameters for FlatScan

|  |  |  |  |
| --- | --- | --- | --- |
| Function | Description | Arguments | Returns |
| **set\_led**(action,  color, blink\_color, blink\_freq) | Set display LED. | **str action :**  ‘set’, ‘blink’  **str color :**  ‘off’, ‘red’, ‘green’, ‘orange’  **str blink\_color :**  ‘off’, ‘red’, ‘green’, ‘orange’  **int blink\_freq :**  range from 1 to 10 (units: Hz) | Boolean ACK:  **True** (1) if success.  **False** (0) if failure to set led, or if an argument value is invalid (e.g. action = ’blnk’). |
| **set\_baudrate**(  baudrate)**\*** | Set the baud rate for serial communication.  To determine the baud rate, take into account the quantity of data, the period of the transmission, and the mode of transmission (HS or HD mode). | **int baudrate :**  57600, 115200, 230400, 460800, 921600 (units: bits/sec) | Boolean ACK:  **True** (1)if success.  **False** (0) if failure to set baud rate, or if argument value is invalid (e.g. baudrate = 100). |
| **reset\_mdi\_counter**() | Set the MDI counter to the default value of 1. |  | Boolean ACK:  **True** (1)if success.  **False** (0)if failure to reset MDI counter. |
| **set\_parameters**(  temperature, mdi\_info, detection\_field\_mode, sensitivity, num\_spots, angle\_first, angle\_last, can\_and\_frame\_counter, heartbeat\_period, facet\_number\_field, averaging) | Set parameters for FlatScan sensor.  Not all parameters must be set at once. | See rows below for more details on each parameter.  **int temperature** = None  **int mdi\_info** = None  **int detection\_field\_ mode** = None  **int sensitivity** = None  **int num\_spots** = None  **int angle\_first** = None  **int angle\_last** = None  **int can\_and\_frame\_ counter** = None  **int heartbeat\_period** = None  **int facet\_number\_field** = None  **int averaging** = None | Verification bits:**\*\***  A decimal value representing an 18-bit binary number. Each bit of the binary number represents the status (success or failure) of a parameter.  The bit value is 0 if success for a given parameter.  **The returned value is 0 if success for all parameters.** |
| **set\_temperature\_field**(  temperature) | Enable or disable CTN (temperature) field in measurement frames. | **int temperature** :  0: disable  1: enable | Boolean:  **True** (1) if success.  **False** (0) if failure to set field, or if an argument value is invalid (e.g. temperature = 10). |
| **set\_mdi\_info**(mdi\_info) | Set information in MDI. | **int mdi\_info** :  0: send distances only  1: send remissions only  2: send both | Boolean:  **True** (1) if success.  **False** (0) if failure to set info, or if an argument value is invalid (e.g. mdi\_info = 10). |
| **set\_detection\_field\_**  **mode**(detection\_field\_  mode) | Set detection field mode. | **int detection\_field\_ mode** :  0: HS mode  1: HD mode | Boolean:  **True** (1) if success.  **False** (0) if failure to set field, or if an argument value is invalid (e.g. detection\_ field\_mode = 10). |
| **set\_sensitivity\_and\_**  **optimization**(  sensitivity\_  optimization) | Set sensitivity and immunization optimization with respect to the size of the detection field. | **int sensitivity\_**  **optimization** :  0: No optimization (max sensitivity)  1: range from 0 to 2.5m (min sensitivity)  2: range from 0 to 3m  3: range from 0 to 3.5m  4: range longer than 3.5m (max sensitivity) | Boolean: 0 if set correctly. |
| **set\_num\_spots**(num\_  spots) | Set the number of spots in the field.  The maximum number of spots depends on the mode (HS or HD). | **int num\_spots** :  range from 0 to 100 (HS mode);  range from 0 to 400, number must be evenly divisible by 4 (HD mode) | Boolean: 0 if set correctly. |
| **set\_angle\_first**(angle\_  first) | Set lower limit of the detection field. | **int angle\_first** :  range from 0 to 10800 (units: 0.01 degrees) | Boolean: 0 if set correctly. |
| **set\_angle\_last**(angle\_  last) | Set upper limit of detection field. | **int angle\_last** :  range from 0 to 10800 (units: 0.01 degrees) | Boolean: 0 if set correctly. |
| **set\_can\_and\_frame\_**  **counter\_field**(can\_and\_  frame\_counter) | Enable or disable CAN and frame counter fields in measurement, heartbeat, emergency frame. | **int can\_and\_frame\_ counter** :  0: disable  1: enable | Boolean: 0 if set correctly. |
| **set\_heartbeat\_period**(  heartbeat\_period) | Set heartbeat period.  If value is 0, heartbeat is disabled. | **int heartbeat\_period** :  range from 0 to 255 (units: 1 sec) | Boolean: 0 if set correctly. |
| **set\_facet\_number\_field**(facet\_number\_field) | Enable or disable the facet number field in MDI. | **int facet\_number\_**  **field** :  0: disable  1: enable | Boolean: 0 if set correctly. |
| **et\_averaging\_setting**(  averaging\_setting) | Set averaging setting. | **int averaging\_setting** = None :  0: No averaging  1: Averaging 3 points at a time  2: Averaging 3 points and 2 neighbors  3: Averaging 5 points at a time  4: Averaging 5 points and 2 neighbors | Boolean: 0 if set correctly. |
| **reset\_emergency\_**  **counter**() | Set the emergency counter to the default value of 1. |  | Boolean ACK:  **True** (1) if success.  **False** (0) if failure to reset emergency counter. |
| **reset\_heartbeat\_**  **counter**() | Set the heartbeat counter to the default value of 1. |  | Boolean ACK:  **True** (1) if success.  **False** (0) if failure to reset heartbeat counter. |

**\*** The baud rate can be set 1 of 2 ways:

1. By creating an object of the class, Flatscan
2. By calling the function, set\_baudrate(baudrate)

If setting the baud rate by creating a class object, the baud must be 921600 to start.

*Example:* **import** api.flatscan\_api **as** API

com = 1

baud = 921600

flatscan\_obj = API.Flatscan(com, baudrate = baud)

Then, the baud rate can be changed by calling set\_baudrate(baudrate). The new value is automatically saved in EEPROM. The new baud rate will take effect upon the next power cycle.

**\*\*** “Verification bits” is a decimal value representing an 18-bit binary number. See table

below for parameter-bit mapping.

Bits 0, 5, 6, 7, 8, 10, and 11 are reserved for future use; these bits are always 0:

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| Bit | | | | | | | | | | | | | | |
| 17 | 16 | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8…5 | 4 | 3 | 2 | 1 | 0 |
| averag-ing | facet\_  number\_  field | heartbeat\_  period | can\_  and\_  frame\_  counter | angle\_  last | angle\_  first | 0 | 0 | num\_  spots | 0000 | sensiti-vity | detection\_  field\_mode | mdi\_  info | temper-ature | 0 |

*Example:*

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| Bit | | | | | | | | | | | | | | | | | | Verification Bits (Decimal Value) | Status of Parameters |
| 17 | 16 | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
| 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | All parameters set correctly. |
| 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 1 | 0 | 0 | 4 | Error setting parameter, mdi\_info. |
| 0 | 0 | 0 | 0 | 1 | 1 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 1 | 0 | 1 | 0 | 12298 | Error setting parameters: temperature, detection\_field\_mode, angle\_first, and angle\_last. |
| 1 | 1 | 1 | 1 | 1 | 1 | 0 | 0 | 1 | 0 | 0 | 0 | 0 | 1 | 1 | 1 | 1 | 0 | 258590 | Error setting all parameters. |

1. Information Returned from the API
   1. Introduction

The dictionaries and parameters described in this section are local to file, *flatscan\_utils*.

Since the customer will not include *flatscan\_utils* as a library in their code, they will not have “direct” access to this information. Rather, the customer will “indirectly” access this information by calling functions in *flatscan\_api*.

* 1. Legend

Color denotes an indirect link between information local to *flatscan\_utils*, and information returned by a *flatscan\_api* function.

Lighter shades indicate information can be read. Darker shades indicate information can be written.

If information in this section is not colored, then there is no link, and the information is not accessible (without actually including *flatscan\_utils* as a library).

|  |  |  |
| --- | --- | --- |
| Dictionary / Function | Dictionary Key / Parameter(s) Returned | Description |
| **MDI dictionary**  Local to function, flatscan\_parse\_mdi\_msg() | **‘CAN\_number’** | CAN |
| **‘temperature’** | CTN |
| **‘facet’** | Facet number (1, 2, 3 or 4 for HS mode) or 5 for HD mode |
| **‘distances’** | Array of distances |
| **‘remissions’** | Array of remissions |
| **‘mdi\_frames\_**  **counter’** | CNTR |

|  |  |  |
| --- | --- | --- |
| **Identity dictionary**  Local to function, flatscan\_parse\_identity\_  msg() | **‘CAN\_number’** |  |
| **‘product\_part\_**  **number’** |  |
| **‘software\_**  **version’** |  |
| **‘software\_**  **Revision’** |  |
| **‘software\_**  **Prototype’** |  |

|  |  |  |
| --- | --- | --- |
| **Parameters dictionary**  Local to function, flatscan\_parse\_parameters\_msg() | **‘temperature’** | CTN |
| **‘mdi\_info’** | Information to be sent: distances, remissions, or both |
| **‘detection\_**  **field\_mode’** | Transfer mode: HS or HD |
| **‘sensitivity’** | Sensitivity and immunity optimization, with the respect to the size of the detection field. |
| **‘num\_spots’** | Number of spots in the field. |
| **‘angle\_first’** | Lower limit of the detection field (unit: 0.01 degree). |
| **‘angle\_last’** | Upper limit of the detection field (unit: 0.01 degree). |
| **‘can\_and\_frame\_**  **counter’** | Enabled or disabled. |
| **‘heartbeat\_**  **period’** | Heartbeat period (unit: 1 sec), if heartbeat is not disabled. |
| **‘facet\_number\_**  **field’** | Enabled or disabled. |
| **‘averaging\_**  **setting’** | Defines algorithm for finding the average. |
| **Emergency dictionary**  Local to function, flatscan\_parse\_emergency\_  msg() | **‘CAN\_number’** |  |
| **‘emergency\_**  **counter’** |  |
| **‘RS485\_error\_**  **code’** |  |
| **‘measuring\_head\_**  **error\_code’** |  |
| **flatscan\_parse\_heartbeat\_**  **msg()**  Returns an array:  [can\_num, counter] | **can\_num** |  |
| **counter** |  |
| **flatscan\_measuring\_head\_**  **error\_parse()** | Error code |  |
| **flatscan\_rs485\_error\_**  **parse()** | Error code |  |

Misc - Flatscan

\_\_init\_\_

export\_log

log\_action

enable\_log

log\_event

register\_heartbeat\_handler

register\_emergency\_handler

save\_parameters

load\_parameters

\_\_get\_msg

\_\_get\_ack

Misc - FlatscanThreading

run

stop

register\_heartbeat\_handler

register\_emergency\_handler

read\_message

enable\_logging