R&D FIRE DEPARTMENT

F022_000 FA-Com Library

Library Description

Api Documentation

Rel. 1.12

DATE: 08/11/2024

Summary

Revision History

DOCUMENT DETAILS	
Authors:	Mariani Matteo
Department:	Fire R&D

VERSION	DATE	DESCRIPTION
1.00	07/10/2019	First issue
1.12	10/05/2024	Add SET_ClosePort() documentation.

FA-Com Constructors

FA_Com(string ip, ushort port)

Description: Initializes a new instance of the FA_Com class, sets the IP address and port, and retrieves the initial device configuration.

Parameters:

- **ip** (*string*) - The IP address of the device to communicate with.
- **port** (*ushort*) - The port number for communication.

Exceptions:

- **FAComLibraryException** - Thrown when there is an error during initialization.

Example Usage:

```
// Example usage of FA_Com constructor with IP and port
string ipAddress = "192.168.1.100";
ushort port = 5000;
FA_Com device = new FA_Com(ipAddress, port);
// Device is now initialized and ready for communication
```

FA_Com(string com)

Description: Initializes a new instance of the FA_Com class with the specified COM port for serial communication.

Parameters:

- **com** (*string*) - The COM port identifier for serial communication.

Exceptions:

- **FAComLibraryException** - Thrown when there is an error during initialization.

Example Usage:

```
// Example usage of FA_Com constructor with COM port
string comPort = "COM3";
FA_Com device = new FA_Com(comPort);
// Device is now initialized and ready for serial communication
```

GET_Ping()

Description: Checks the connectivity of the device by attempting to open a communication channel and retrieve the device version.

Returns: true if the device responds to the ping; otherwise, false.

Exceptions:

- **FAComLibraryException** - Thrown when there is an error during communication or version retrieval.

Example Usage:

```
// Example usage of GET_Ping method
bool isConnected = device.GET_Ping();
if (isConnected)
{
        Console.WriteLine("Device is responsive.");
}
else
{
        Console.WriteLine("Device is not responding.");
}
```

FA-Com Methods

GET_DetectorRealTimeStatus(int detector index)

Description: Retrieves the real-time status of a specified detector.

Parameters:

- **detector_index** (*int*) - The index of the detector to check.

Returns: A DetectorRealTimeData object containing the current status and metrics of the detector.

Exceptions:

- **FAComLibraryException** - Thrown when there is an error retrieving the real-time status of the detector.

Example Usage:

```
// Example usage of GET_DetectorRealTimeStatus method

int detectorIndex = 0; // Index of the detector

DetectorRealTimeData status = device.GET_DetectorRealTimeStatus(detectorIndex);

// Display the real-time status

Console.WriteLine($"Smoke Level: {status.S_SMOKELEVEL}%");

Console.WriteLine($"Temperature: {status.S_TEMPERATURE}°C");
```

GET_PanelRealTimeStatus()

Description: Retrieves the real-time status of the panel, including voltage, RPM, IO states, and various fault indicators.

Returns: A PanelRealTimeData structure populated with current panel status information.

Exceptions:

- **FAComLibraryException** - Thrown when there is an error retrieving the real-time status of the panel.

Example Usage:

```
// Example usage of GET_PanelRealTimeStatus method

PanelRealTimeData panelStatus = device.GET_PanelRealTimeStatus();

// Display the real-time status

Console.WriteLine($"Primary Voltage: {panelStatus.S_PRIMVolt}V");
```

```
Console.WriteLine($"Fan RPM: {panelStatus.S_RPM}");
```

GET_Configuration Method()

Description: Retrieves the current configuration of the device, including firmware, hardware, model, serial number, and detector programming details. Communicates with the device via serial or WiFi connection to gather version information and detector data.

Returns: A DeviceProg structure containing the configuration of the device and associated detectors.

Exceptions:

- **FAComLibraryException** - Thrown when there is an error retrieving the configuration from the device.

Example Usage:

```
// Example usage of GET_Configuration method

DeviceProg detector_prog; // Detector configuration

FA_Com com = new FA_Com("10.105.151.18", (ushort)3060);

detector_prog = com.GET_Configuration();

// Display the real-time status

Console.WriteLine($"Boot ver.: {detector_prog.S_BOOT}%");

Console.WriteLine($"Detector 1 CLASS: {detector_prog.S_DETECTOR[0].S_CLASS}");

Console.WriteLine($"Detector 2 CLASS: {detector_prog.S_DETECTOR[1].S_CLASS}");
```

SET_ClosePort ()

Description: Close Connection Port on Device The port connection on the device does not need to be manually closed after every subsequent call. Ports will automatically close after 60 seconds of inactivity. Once the port is closed, there is no need to reopen it manually for subsequent call.

Returns: - true if the device is closed; false - if thrown an exception.

Exceptions:

FAComLibraryException - Thrown when there is an error during closing port.

Example Usage:

```
// Example usage of GET_Configuration method
   DeviceProg detector_prog; // Detector configuration
   FA_Com com = new FA_Com("192.168.1.100", (ushort)3060);


   DetectorRealTimeData status = device.GET_DetectorRealTimeStatus(detectorIndex);
   Console.WriteLine($"Smoke Level: {status.S_SMOKELEVEL}%");
   status = device.GET_DetectorRealTimeStatus(detectorIndex);
   Console.WriteLine($"Smoke Level: {status.S_SMOKELEVEL}%");
   status = device.GET_DetectorRealTimeStatus(detectorIndex);
   Console.WriteLine($"Smoke Level: {status.S_SMOKELEVEL}%");
   com.SET_ClosePort();`
```

FA-Com Structs

DetectorRealTimeData

Description: Struct representing real-time data and status indicators for a detector.

Fields:

- **S_DATETIME** (*DateTime*) - Date and time of the recorded data sample.
- **S_SMOKE** (*double*) - Smoke level, obscuration level in db/m.
- **S_SMOKELEVEL** (*double*) - Smoke level, percentage of obscuration level (0% .. 255%); 100% corresponds to reaching the alarm threshold alarmThr.
- **S_RELIABILITY** (*double*) - Reliability level, range 0 .. 200 where 0 represents the highest probability of a false alarm, and 200 represents the highest probability of a fire.
- **S_CONTAMINATION** (*double*) - Contamination level, represented as a percentage.
- **S_FLOW** (*double*) - Air flow rate measurement, represented as l/min. Range: 0 to 655,35 l/s. Resolution: 0.01 l/min.
- **S_FLOW_PERC** (*double*) - Flow rate measurement, represented as a percentage of the nominal flow.
- **S_TEMPERATURE** (*double*) - Measured air temperature, represented in degrees Celsius. Range: 64 to +63.5 °C. Resolution: 0.5 °C.
- **S_ALARM** (*bool*) - Indicates the smoke level exceeded the alarm threshold.
- **S_WARNING** (*bool*) - Indicates if the smoke level exceeded the warning threshold.
- **S_CUSTOM_EVENT** (*bool*) - Indicates if the smoke level exceeded the custom threshold.
- **S_FAULT_NOT_PRESENT** (*bool*) - Indicates if detector is not inserted in the slot (measurements are not valid).
- **S_FAULT_NO_RESPONSE** (*bool*) - Indicates if detector is not responding or returns FAIL (measurements are not valid).
- **S_FAULT_OPTICS** (*bool*) - Indicates an optics fault, rendering certain optical measurements invalid.
- **S_FAULT_CONTAMINATION** (*bool*) - Indicates the detector is contaminated (measurements are valid).
- **S_FAULT_HIGH_FLOWRATE** (*bool*) - Indicates a high flow rate, which may suggest a broken tube.
- **S_FAULT_LOW_FLOWRATE** (*bool*) - Indicates a low flow rate, potentially indicating a blockage.
- **S_FAULT_FLOWMETER** (*bool*) - Flow meter malfunction indicator; speed and temperature data may be invalid.
- **S_TEMPERATURE_EVENT** (*bool*) - Indicates that air temperature has surpassed a programmed threshold.

PanelRealTimeData

Description: Represents real-time data and status indicators for a control panel.

Fields:

- **S_RPM** (*double*) - Fan speed in RPM.
- **S_PRIMVolt** (*double*) - Primary supply voltage.
- **S_AUXVolt** (*double*) - Auxiliary supply voltage.
- **S_IO_POLARITY** (*int[]*) - Array of polarity statuses for I/O channels: 0 = Normally Closed (NC), 1 = Normally Open (NO).
- **S_IO_DIRECTION** (*int[]*) - Direction of each I/O channel: 0 = Input, 1 = Output.
- **S_IO_ACTIVE** (*int[]*) - Activation status of each I/O channel: 0 = Restore, 1 = Active.
- **S_IO_REFERENCES** (*int[]*) - References configuration for each I/O channel: 0 = Positive, 1 = Negative.
- **S_IO_OUT_FAULT** (*bool[]*) - Fault status for each output I/O channel, indicating interconnection issues.
- **S_IO_IN_FAULT_SHORT** (*bool[]*) - Fault status for input I/O channels indicating a short circuit condition.
- **S_IO_IN_FAULT_OPEN** (*bool[]*) - Fault status for input I/O channels indicating an open circuit condition.
- **S_STATUS_OK** (*bool*) - True if no issues are detected in the system.
- **S_PROGRAMMING_FAULT** (*bool*) - Indicates a data programming error or corruption.
- **S_FW_UPDATE_FAULT** (*bool*) - Indicates a failure during firmware update.
- **S_PRI_SUPPLY_FAULT** (*bool*) - Indicates a primary power supply issue.
- **S_AUX_SUPPLY_FAULT** (*bool*) - Indicates an auxiliary power supply issue.
- **S_FAN_FAULT** (*bool*) - Indicates a fault in the blower.
- **S_LOOP_NOT_RESPONDING** (*bool*) - Indicates that a loop is not responding.
- **S_LOOP_ISOLATOR** (*bool*) - Indicates that the loop isolator is open.
- **S_INTERCONNECTION_FAULT** (*bool*) - Indicates a fault in the interconnections.
- **S_SERVICE_MODE** (*bool*) - Indicates that the system is in service mode.

DeviceProg

Description: Represents the programming information for a device, including firmware, hardware, model, and associated detectors.

Fields:

- **S_BOOT** (*string*) - Boot version of the device.
- **S_FIRMWARE** (*string*) - Firmware version of the device.
- **S_HARDWARE** (*string*) - Hardware version of the device.
- **S_MODEL** (*string*) - Model identifier of the device.
- **S_SERIAL** (*string*) - Serial number of the device.
- **S_DETECTOR** (*DetectorProg[]*) - Array of detectors associated with the device.

DetectorProg

Description: Represents the programming details for a single detector, including flow settings, thresholds, and custom labels.

Fields:

- **S_ENABLED** (*bool*) - Indicates if the detector is enabled (true or false).
- **S_DETECTORLABEL** (*string*) - Custom label assigned to the detector.
- **S_MDETECTOR** (*string*) - Detector sensitivity; range 235 .. 20000 expressed in 10^-5 dB / m.
- **S_CLASS** (*string*) - Classification or type of the detector.
- **S_NOMINAL_FLOW** (*double*) - Nominal air flow rate; range 0.1 to 180 l/min.
- **S_WARNINGTHR** (*int*) - Threshold for warning notifications.
- **S_CUSTOMTHR** (*int*) - Custom threshold value set for the detector.
- **S_CUSTOMLABEL** (*string*) - Custom label for identifying or categorizing the detector.
- **S_DATALOGGERTHR** (*int*) - Threshold for data logging triggers in the detector.
- **S_HIGHFLOWTHR** (*int*) - Threshold for high flow alerts.
- **S_LOWFLOWTHR** (*int*) - Threshold for low flow alerts.
- **S_TEMPERATURETHR** (*int*) - Temperature threshold for the detector.