12. February 2024

IMT.Analytics

USB Protocol DLL User Guide for Gas Flow Analysers PF-300 PRO and Citrex H5

CONTENT

1 TECHNICAL OVERVIEW

The USB communication to the CITREX and PF-300 Pro works through a virtual COM port with a binary protocol.

1.1 PREREQUISITES

1. USB connection to CITREX H5 or PF-300 Pro.
2. .NET 6.0 installed.
3. Microsoft Visual Studio installed.

2 SETUP THE PROJECT

2.1 ADD REFERENCE TO ASSEMBLY IN PROJECT

In your project, add reference to the assembly by left click on the project in Visual Studio and click on “Add”, choose “Project Reference” and browse to the assembly on your file system.

2.2 ESTABLISH CONNECTION TO THE DEVICE

You can create a Citrex or FlowAnalyserPro object to establish the connection to your device. For example, instantiate a Citrex class and call the “Connect” function to connect with your CITREX device. Identify the USB virtual COM port name from Windows Device Manager and pass the port name to the “Connect” function.

```
device = new Citrex();
device.Connect("COM14");
```

For PF-300 Pro, instantiate a FlowAnalyserPro class instead.

```
flowAnalyserPro = new FlowAnalyserPro();
flowAnalyserPro.Connect("COM10");
```

Once the connect call is made, the assembly will try to open the COM port and send device detection packet to device, and after receiving this device detection packet, the device will respond with a device detection packet immediately, and once the assembly receives the device detection packet from device, the connection is established. After the connection is established, settings requests are also sent to the device to retrieve device settings and trigger settings information.

To check the connection status, one can check against the “IsConnected” property in Citrex or FlowAnalyserPro class.

```
if (citrex.IsConnected) {
    // citrex is connected
}

if (flowAnalyserPro.IsConnected) {
    // PF-300 Pro is connected
}
```

To receive notification when the connection is disrupted, you can listen to “ExceptionOccurred” event.

```
citrex.ExceptionOccurred += Device_ExceptionOccurred;

flowAnalyserPro.ExceptionOccurred += Device_ExceptionOccurred;

private void Device_ExceptionOccurred(object? sender, Exception e) {
    Invoke(new Action(() => textBoxError.Text = e.Message));
}
```

3 SUBSCRIBE TO MEASUREMENT DATA

3.1 SUBSCRIBE TO CITREX MEASUREMENT DATA

Once the connection to CITREX is established, you can subscribe to fast data and all data to get the real time values.

For CITREX

```
            citrex.ValuesAllDataRemoteObject.AllDataHandler +=
HandleCitrexAllData;
            citrex.ValuesFastDataRemoteObject.FastDataHandler +=
HandleCitrexFastData;


        private void HandleCitrexAllData(object sender, CitrexAllData eventData)
{
            if (eventData != null) {
                Invoke(new Action(() => textBoxFlow.Text =
eventData.Flow.ToString()));
            }
        }

        private void HandleCitrexFastData(object sender, CitrexFastData
eventData) {
            if (eventData != null) {
                Invoke(new Action(() => textBoxFlow.Text =
eventData.Flow.ToString()));
            }
        }
```

3.1.1 CITREX ALL DATA DEFINITION

CITREX “All Data” includes all measurement and respiratory values and is sent from CITREX devices at 5Hz.

Value	Unit	Data Type	Comment
Flow	l/min	float	Flow
RawPressureDiffFlow	mbar	float	Raw differential pressure over LFE
PressureDiff	mbar	float	External differential pressure
PressureChannel	mbar	float	Pressure in channel (airway pressure)
PressureHigh	bar	float	High pressure
Volume	ml	float	Volume
Oxygen	%	float	Oxygen
Humidity	%	float	Humidity
Temperature	°C	float	Temperature
PressureAmbient	mbar	float	Pressure ambient
InspTime	s	float	Inspiration time
ExpTime	s	float	Expiration time
IToE	-	float	I:E
BreathRate	bpm	float	Breath rate
Vti	ml	float	Inspiration tidal volume

Vte	ml	float	Expiration tidal volume
Vi	l/min	float	Inspiration volume
Ve	l/min	float	Expiration volume
PeakPressure	mbar	float	Peak pressure
MeanPressure	mbar	float	Mean pressure
PEEP	mbar	float	PEEP
TiOverTcycle	%	float	Ti over Tcycle
PeakFlowInsp	l/min	float	Inspiratory peak flow
PeakFlowExp	l/min	float	Expiratory peak flow
PlateauPressure	mbar	float	Plateau pressure
Compliance	ml/mbar	float	Compliance
InspiratoryPositiveAirwayPressure	mbar	float	Inspiratory positive airway pressure
IrmaN2O	%	float	Irma sensor N2O
IrmaCO2	%	float	Irma sensor CO2
IrmaHalothane	%	float	Irma sensor Halothane
IrmaEnflurane	%	float	Irma sensor Enflurane
IrmaIsoflurane	%	float	Irma sensor Isoflurane
IrmaSevoflurane	%	float	Irma sensor Sevoflurane
IrmaDesflurane	%	float	Irma sensor Desflurane

3.1.2 CITREX FAST DATA DEFINITION

Citrex “Fast Data” includes the following measurement values and is sent from CITREX devices at 200Hz

Value	Unit	Data Type	Comment
Flow	l/min	float	Flow
RawPressureDiffFlow	mbar	float	Raw differential presure over LFE
PressureDiff	mbar	float	External differential pressure
PressureChannel	mbar	float	Pressure in channel (airway pressure)
PressureHigh	bar	float	High pressure
Volume	ml	float	Volume

3.2 SUBSCRIBE TO PF-300 PRO MEASUREMENT DATA

Once the connection to PF-300 Pro is established, you can subscribe to fast data, all data or IRMA data to get the real time values.

For PF-300 Pro

```
            flowAnalyserPro.ValuesAllDataRemoteObject.AllDataHandler +=
HandlePF300ProAllData;
            flowAnalyserPro.ValuesFastDataRemoteObject.FastDataHandler +=
HandlePF300ProFastData;
            flowAnalyserPro.ValuesIrmaDataRemoteObject.IrmaDataHandler +=
HandlePF300ProIrmaData;


        private void HandleCitrexFastData(object sender, CitrexFastData
eventData) {
            if (eventData != null) {
                Invoke(new Action(() => textBoxFlow.Text =
eventData.Flow.ToString()));
            }
        }

        private void HandlePF300ProFastData(object sender, PF300ProFastData
eventData) {
            if (eventData != null) {
                Invoke(new Action(() => textBoxFlow.Text =
eventData.Flow.ToString()));
            }
        }
        private void HandlePF300ProIrmaData(object sender, PF300ProIrmaData
eventData) {
            if (eventData != null) {
                Invoke(new Action(() => textBoxIrmaCO2.Text =
eventData.IrmaCO2.ToString()));
            }
        }
```

3.2.1 PF-300 PRO ALL DATA DEFINITION

PF-300 Pro “All Data” includes all measurement and respiratory values and is sent from PF-300 Pro at 5 Hz.

Value	Unit	Data Type	Comment
Flow	l/min	float	Flow
PressureDiffLow	mbar	float	Low Differential Pressure
PressureDiffHigh	mbar	float	High Differential Pressure
PressureChannel	mbar	float	Pressure in channel (airway pressure)
PressureHigh	bar	float	High pressure
Volume	ml	float	Volume
UltraLowFlow	l/min	float	Ultra low flow
Oxygen	%	float	Oxygen
Humidity	%	float	Humidity
Temperature	°C	float	Temperature
PressureAmbient	mbar	float	Pressure ambient
InspTime	s	float	Inspiration time
ExpTime	s	float	Expiration time
IToE	-	float	I:E
BreathRate	bpm	float	Breath rate
Vti	ml	float	Inspiration tidal volume
Vte	ml	float	Expiration tidal volume

Vi	l/min	float	Inspiration volume
Ve	l/min	float	Expiration volume
PeakPressure	mbar	float	Peak pressure
MeanPressure	mbar	float	Mean pressure
PEEP	mbar	float	PEEP
TiOverTcycle	%	float	Ti over Tcycle
PeakFlowInsp	l/min	float	Inspiratory peak flow
PeakFlowExp	l/min	float	Expiratory peak flow
PlateauPressure	mbar	float	Plateau pressure
Compliance	ml/mbar	float	Compliance
InspiratoryPositiveAirwayPressure	mbar	float	Inspiratory positive airway pressure
TiHold	s	float	Inspiration hold time
TeHold	s	float	Expiration hold time
RatioTp	%	float	%TP
IrmaN2O	%	float	Irma sensor N2O
IrmaCO2	%	float	Irma sensor CO2
IrmaHalothane	%	float	Irma sensor Halothane
IrmaEnflurane	%	float	Irma sensor Enflurane
IrmaIsoflurane	%	float	Irma sensor Isoflurane
IrmaSevoflurane	%	float	Irma sensor Sevoflurane
IrmaDesflurane	%	float	Irma sensor Desflurane

3.2.2 PF-300 PRO FAST DATA DEFINITION

PF-300 Pro “Fast Data” includes following values and is sent from PF-300 Pro at 1KHz

Value	Unit	Data Type	Comment
Flow	l/min	float	Flow
PressureDiffLow	mbar	float	Low Differential Pressure
PressureDiffHigh	mbar	float	High Differential Pressure
PressureChannel	mbar	float	Pressure in channel (airway pressure)
PressureHigh	bar	float	High pressure
Volume	ml	float	Volume
UltraLowFlow	l/min	float	Ultra low flow

3.2.3 PF-300 PRO IRMA DATA DEFINITION

PF-300 Pro “Irma Data” includes following values and is sent from PF-300 Pro at 20Hz.

Value	Unit	Data Type	Comment
IrmaN2O	%	float	Irma sensor N2O
IrmaCO2	%	float	Irma sensor CO2
IrmaHalothane	%	float	Irma sensor Halothane
IrmaEnflurane	%	float	Irma sensor Enflurane
IrmaIsoflurane	%	float	Irma sensor Isoflurane
IrmaSevoflurane	%	float	Irma sensor Sevoflurane
IrmaDesflurane	%	float	Irma sensor Desflurane

4 GET AND SET DEVICE SETTINGS

4.1 GET AND SET CITREX DEVICE CONFIG SETTINGS

For the device settings of CITREX, they can be retrieved with the "DeviceConfigRemoteObject" member of the Citrex class. After the connection to CITREX device is established, the assembly will request device settings from the CITREX device and once the device settings are received, the "SettingsReceived" property in "DeviceConfigRemoteObject will be set to true. For example, to get the "Gas Type" of the Citrex, one can first check the "SettingsReceived" property in "DeviceConfigRemoteObject", if true, then read the device settings, refer to Chapter 4.1.1 for detailed device settings:

```
            if (citrex.DeviceConfigRemoteObject.SettingsReceived) {
                CitrexGasType gasTypeCitrex =
citrex.DeviceConfigRemoteObject.DeviceGasType;
            }
```

It is also possible to request settings to be sent by CITREX device explicitly after connection established, by calling the "RequestSettings" function in "DeviceConfigRemoteObject".

```
            if (citrex.IsConnected) {
                citrex.TriggerConfigRemoteObject.RequestSettings();
            }
```

To get notified when a device setting has been changed in device:

```
            citrex.DeviceConfigRemoteObject.PropertyChanged +=
DeviceConfigRemoteObject_PropertyChanged;
```

```
        private void DeviceConfigRemoteObject_PropertyChanged(object? sender,
System.ComponentModel.PropertyChangedEventArgs e) {
            switch (e.PropertyName) {
                case DeviceConfigRemoteObject.PROPERTY_GAS_TYPE:
                    Invoke(new Action(() => textBoxGasType.Text =
citrex.DeviceConfigRemoteObject.DeviceGasType.ToString()));
                    break;
                default:
                    break;
            }
        }
```

To change a device setting and send to device:

```
            if (citrex.IsConnected) {

      citrex.DeviceConfigRemoteObject.SendGasType(CitrexGasType.Heliox);
            }
```

4.1.1

CITREX DEVICECONFIGREMOTEOBJECT DEFINITIONS

The following table contains the properties of “DeviceConfigRemoteObject” which are setting values from CITREX devices.

Value	Setting from Device	Type
DeviceAnalogOut1ValueModelId	Config Analog Out 1	Flow = 6002 PressureDiff = 6003 PressureFlowChannel = 6004 HighPressure = 6010 Oxygen = 6007 Temperature = 6009
DeviceAnalogOut2ValueModelId	Config Analog Out 2	Flow = 6002 PressureDiff = 6003 PressureFlowChannel = 6004 HighPressure = 6010 Oxygen = 6007 Temperature = 6009
DeviceOwnerName	Owner	string
DeviceCompanyName	Company	string
DeviceScreenRotationLocked	Lock Screen Rotation	bool
SensorPressureCompensationType	Sensor Pressure Compensation	PressureInFlow = 0 PressureHigh = 1
DeviceFilterType	Filter Type	None = 0 Low = 1 Medium = 2 High = 3
DeviceTriggerMode	Trigger Mode	Adult = 0 Pediatric = 1 HighFreq = 2
DeviceGasType	Gas Type	Air = 0 AirO2xMan = 1 AirO2xAut = 2 N2OxO2xMan = 3 N2OxO2xAut = 4 Heliox = 5 HeO2xMan = 6 HeO2xAut = 7 N2 = 8 CO2 = 9
DeviceVolumeStandard	Volume Standard	ATP = 0 STP = 1 BTPS = 2 BTPD = 3 0x1013 = 4 20x981 = 5 15x1013 = 6 20x1013 = 7 25x991 = 8 AP21 = 9 STPH = 10 ATPD = 11 ATPS = 12 BTPS_A = 13 BTPD_A = 14 NTPD = 15 NTPS = 16
DeviceGasTypeOxygenConcentration	Oxygen Concentration	float (%)

DeviceHumidity	Humidity	float (%)
DeviceBaseFlowEnabled	Base Flow Enabled	bool
DeviceBaseFlowValue	Base Flow Value	float (l/min)
DeviceOxygenOptionEnabled	Option O2 Activated	bool
DeviceMonitoringOptionEnabled	Option Monitoring Activated	bool
DeviceTSI4000ProtocolOptionEnabled	Option TSI4000 Protocol Activated	bool
DeviceRS232Protocol	RS232 Protocol	ImtProtocol = 0 TSI4000Protocol = 1 Calibration = 2 ImtFastProtocol = 3 PrimaProtocol = 4 IrmaProtocol = 5
DeviceFlowLabOptionEnabled	Option FlowLab Activated	
DeviceVentilationParameterPressureSource	Ventilation Parameter Pressure Source	PressureInFlow = 0 PressureDiff = 1

The following table contains functions in “DevcieConfigRemoteObject” which can be used to send device settings to CITREX devices. It is not possible to send the “OxygenOptionEnabled”, “MonitoringOptionEnabled”, “TSI4000ProtocolOptionEnabled” and “FlowLabOptionEnabled” values to device because they are noly readable through the protocol.

Function	**Setting to Device**	**Input Data Type**
SendAnalogOut1ValueModelId	Config Analog Out 1	Flow = 6002 PressureDiff = 6003 PressureFlowChannel = 6004 HighPressure = 6010 Oxygen = 6007 Temperature = 6009
SendAnalogOut2ValueModelId	Config Analog Out 2	Flow = 6002 PressureDiff = 6003 PressureFlowChannel = 6004 HighPressure = 6010 Oxygen = 6007 Temperature = 6009
SendOwnerName	Owner	string
SendOwnerCompany	Company	string
SendScreenRotationLocked	Lock Screen Rotation	bool
SensorPressureCompensationType	Sensor Pressure Compensation	PressureInFlow = 0 PressureHigh = 1
SendFilterType	Filter Type	None = 0 Low = 1 Medium = 2 High = 3
SendTriggerMode	Trigger Mode	Adult = 0 Pediatric = 1 HighFreq = 2

SendGasType	Gas Type	Air = 0 AirO2xMan = 1 AirO2xAut = 2 N2OxO2xMan = 3 N2OxO2xAut = 4 Heliox = 5 HeO2xMan = 6 HeO2xAut = 7 N2 = 8 CO2 = 9
SendVolumeStandard	Volume Standard	ATP = 0 STP = 1 BTPS = 2 BTPD = 3 0x1013 = 4 20x981 = 5 15x1013 = 6 20x1013 = 7 25x991 = 8 AP21 = 9 STPH = 10 ATPD = 11 ATPS = 12 BTPS_A = 13 BTPD_A = 14 NTPD = 15 NTPS = 16
SendGasTypeOxygenConcentration	Oxygen Concentration	float (%)
SendHumidity	Humidity	float (%)
SendBaseFlowEnabled	Base Flow Enabled	Bool
SendBaseFlowValue	Base Flow Value	float (l/min)
SendRS232Protocol	RS232 Protocol	ImtProtocol = 0 TSI4000Protocol = 1 Calibration = 2 ImtFastProtocol = 3 PrimaProtocol = 4 IrmaProtocol = 5
SendVentilationParameterPressureSource	Ventilation Parameter Pressure Source	PressureInFlow = 0 PressureDiff = 1

4.2 GET AND SET PF-300 PRO DEVICE CONFIG SETTINGS

For the device settings of PF-300 Pro, they can be retrieved with the DeviceSettingsRemoteObject and MeasurementSettingsRemoteObject member of the FlowAnalyserPro class. After the connection to PF-300 Pro device is established, the assembly will request device settings from the PF-300 Pro device and once the device settings are received, the “SettingsReceived” property in “DeviceSettingsRemoteObject” and “MeasurementSettingsRemoteObject” will be set to true. For example, to get the Gas Type of the PF-300 Pro, one can first check the “SettingsReceived” property in “MeasurementSettingsRemoteObject”, if true, then read the device settings:

```
                if (flowAnalyserPro.MeasurementSettingsRemoteObject.SettingsReceived)
        {
                    Pf300ProGasType gasType =
        flowAnalyserPro.MeasurementSettingsRemoteObject.DeviceGasType;
                }

                if (flowAnalyserPro.DeviceSettingsRemoteObject.SettingsReceived) {
                    Pf300ProRS232Protocol rS232Protocol =
        flowAnalyserPro.DeviceSettingsRemoteObject.DeviceRS232Protocol;
                }
```

It is also possible to request device config settings from PF-300 Pro explicitly by calling the "RequestMeasurementSettings" function and "RequestDeviceSettings" function in "SettingRequestRemoteObject", once the connection is established.

```
                        if (flowAnalyserPro.IsConnected) {

        flowAnalyserPro.SettingRequestRemoteObject.RequestMeasurementSettings();

        flowAnalyserPro.SettingRequestRemoteObject.RequestDeviceSettings();
                }
```

To get notified when a device setting has been changed in device:

```
            flowAnalyserPro.MeasurementSettingsRemoteObject.PropertyChanged +=
MeasurementSettingsRemoteObject_PropertyChanged;
            flowAnalyserPro.DeviceSettingsRemoteObject.PropertyChanged +=
DeviceSettingsRemoteObject_PropertyChanged;



        private void MeasurementSettingsRemoteObject_PropertyChanged(object?
sender, System.ComponentModel.PropertyChangedEventArgs e) {
            switch (e.PropertyName) {
                case MeasurementSettingsRemoteObject.PROPERTY_GAS_TYPE:
                    Invoke(new Action(() => textBoxGasType.Text =
flowAnalyserPro.MeasurementSettingsRemoteObject.DeviceGasType.ToString()));
                    break;
                default:
                    break;
            }
        }

        private void DeviceSettingsRemoteObject_PropertyChanged(object? sender,
System.ComponentModel.PropertyChangedEventArgs e) {
            switch (e.PropertyName) {
                case DeviceSettingsRemoteObject.PROPERTY_RS232_PROTOCOL:
                    Invoke(new Action(() => textBoxRs232.Text =
flowAnalyserPro.DeviceSettingsRemoteObject.DeviceRS232Protocol.ToString()));
                    break;
                default:
                    break;
            }
        }
```

To change a device setting and send to device after the connection is established:

```
            if (flowAnalyserPro.IsConnected) {

flowAnalyserPro.MeasurementSettingsRemoteObject.SendGasType(Pf300ProGasType.Air);
            }

            if (flowAnalyserPro.IsConnected) {

flowAnalyserPro.DeviceSettingsRemoteObject.SendRS232Protocol(Pf300ProRS232Protoco
l.ImtProtocol);
            }
```

4.2.1 PF-300 PRO MEASUREMENTSETTINGSREMOTEOBJECT DEFINITIONS

The following table contains properties from “MeasurementSettingsRemoteObject” which can be used to read measurement setting values from PF-300 Pro.

Value	Setting from Device	Type
DeviceVentilationParameterPressureSource	Ventilation Parameter Pressure Source	PressureInFlow = 0 PressureDiffHigh = 1 PressureDiffLow = 2
DynamicPressureCompensationEnabled	Dynamic Pressure Compensation Enabled	bool
DeviceFilterType	Filter Type	None = 0 Low = 1 Medium = 2 High = 3
DeviceTriggerMode	Trigger Mode	Adult = 0 Pediatric = 1 HighFreq = 2 AutoTrigger = 3
DeviceGasType	Gas Type	Air = 0 AirO2xMan = 1 AirO2xAut = 2 N2OxO2xMan = 3 N2OxO2xAut = 4 Heliox = 5 HeO2xMan = 6 HeO2xAut = 7 N2 = 8 CO2 = 9 Custom = 10 N2O = 11 He = 12

DeviceVolumeStandard	Volume Standard	ATP = 0 STP = 1 BTPS = 2 BTPD = 3 0x1013 = 4 20x981 = 5 15x1013 = 6 20x1013 = 7 25x991 = 8 AP21 = 9 STPH = 10 ATPD = 11 ATPS = 12 BTPS_A = 13 BTPD_A = 14 NTPD = 15 NTPS = 16 STP20 = 17 STPD0 = 18 AP25 = 19 23x1013 = 20 STPD21 = 21
DeviceGasTypeOxygenConcentration	Oxygen Concentration	float (%)
DeviceBaseFlowMode	Base Flow Mode	Off = 0 BaseFlow = 1 LeakageEstimation = 2
DeviceBaseFlowValue	Base Flow Value	float (l/min)
DeviceUlfGasType	ULF Gas Type	Air = 0 N2 = 1 CO2 = 2 N2O = 3 He = 4
DeviceUlfVolumeStandard	ULF Volume Standard	STP = 0 STPD0 = 1 STPD21 = 2 0x1013 = 3 20x981 = 4 15x1013 = 5 25x991 = 6 20x1013 = 7 23x1013 = 8 NTPD = 9 NTPS = 10

The following table contains functions in “MeasurementSettingsRemoteObject” which can be used to send measurement setting values to PF-300 Pro.

Function	**Setting to Device**	**Input Data Type**
SendDynamicPressureCompensationEnabled	Dynamic Pressure Compensation Enabled	bool
SendFilterType	Filter Type	None = 0 Low = 1 Medium = 2 High = 3
SendTriggerMode	Trigger Mode	Adult = 0 Pediatric = 1 HighFreq = 2 AutoTrigger = 3

SendGasType	Gas Type	Air AirO2xMan AirO2xAut N2OxO2xMan N2OxO2xAut Heliox HeO2xMan HeO2xAut N2 CO2 Custom N2O He	= 0 = 1 = 2 = 3 = 4 = 5 = 6 = 7 = 8 = 9 = 10 = 11 = 12
SendVolumeStandard	Volume Standard	ATP STP BTPS BTPD 0x1013 20x981 15x1013 20x1013 25x991 AP21 STPH ATPD ATPS BTPS_A BTPD_A NTPD NTPS STP20 STPD0 AP25 23x1013 STPD21	= 0 = 1 = 2 = 3 = 4 = 5 = 6 = 7 = 8 = 9 = 10 = 11 = 12 = 13 = 14 = 15 = 16 = 17 = 18 = 19 = 20 = 21
SendGasTypeOxygenConcentration	Oxygen Concentration	float (%)	
SendBaseFlowEnabled	Base Flow Mode	Off BaseFlow LeakageEstimation	= 0 = 1 = 2
SendBaseFlowValue	Base Flow Value	float (l/min)	
DeviceUlfGasType	ULF Gas Type	Air N2 CO2 N2O He	= 0 = 1 = 2 = 3 = 4
DeviceUlfVolumeStandard	ULF Volume Standard	STP STPD0 STPD21 0x1013 20x981 15x1013 25x991 20x1013 23x1013 NTPD NTPS	= 0 = 1 = 2 = 3 = 4 = 5 = 6 = 7 = 8 = 9 = 10
SendVentilationParameterPressureSource	Ventilation Parameter Pressure Source	PressureInFlow PressureDiffHigh PressureDiffLow	= 0 = 1 = 2

4.2.2 PF-300 PRO DEVICESETTINGSREMOTEOBJECT DEFINITIONS

The following table contains properties from “DeviceSettingsRemoteObject” which can be used to read device setting values from PF-300 Pro.

Value	Setting from Device	Type
DeviceRS232Protocol	RS232 Protocol	ImtProtocol = 0 ImtFastProtocol = 1 IrmaProtocol = 2 ImtExpressProtocol = 3

The following table contains functions from “DeviceSettingsRemoteObject” which can be used to send device setting values to PF-300 Pro

Function	Setting to Device	Input Data Type
SendRS232Protocol	RS232 Protocol	ImtProtocol = 0 ImtFastProtocol = 1 IrmaProtocol = 2 ImtExpressProtocol = 3

5 GET AND SET TRIGGER SETTINGS

5.1 TRIGGER CONFIG

We define a “TriggerConfig” class for easy access to the trigger settings. The “TriggerConfig” class has following properties:

Value	Setting	Type
Source	Trigger Source	Internal = 0 External = 1
StartSignal	Start Signal	Flow = 0 Pressure = 1
StartEdge	Start Edge	Rising = 0 Falling = 1
StartLevel	Start Level	float (l/min for Flow signal, mbar for Pressure Signal)
StartDelay	Start Delay	UInt16 (ms)
EndSignal	End Signal	Flow = 0 Pressure = 1
EndEdge	End Edge	Rising = 0 Falling = 1
EndLevel	End Level	float (l/min for Flow signal, mbar for Pressure Signal)
EndDelay	End Delay	UInt16 (ms)

5.2 GET AND SET CITREX TRIGGER CONFIG SETTINGS

For the trigger settings of CITREX, they can be retrieved with the “TriggerConfigRemoteObject” member of the Citrex class. After connection to CITREX is established, the assembly will request trigger settings from the CITREX device and once the trigger settings are received, the “SettingsReceived” property in “TriggerConfigRemoteObject” will be set to true.

Note: the trigger settings here do not include “Trigger Mode” which is in “DeviceConfigRemoteObject”, refer to Chapter 4.1.1 for reading and sending “Trigger Mode”.

For example, to get the “Trigger Source” of the CITREX Adult Trigger, Pediatric Trigger and High Frequency Trigger, one can first check the “SettingsReceived” property in “TriggerConfigRemoteObject”, if true, then read the trigger settings:

```
            if (citrex.TriggerConfigRemoteObject.SettingsReceived) {
                TriggerSource adultTriggerSourceCitrex =
citrex.TriggerConfigRemoteObject.DeviceAdultTrigger.Source;

                TriggerSource pediatricTriggerSourceCitrex =
citrex.TriggerConfigRemoteObject.DevicePediatricTrigger.Source;

                TriggerSource highFreqTriggerSourceCitrex =
citrex.TriggerConfigRemoteObject.DeviceHighFrequencyTrigger.Source;
            }
```

It is also possible to request trigger settings from CITREX explicitly by calling “RequestSettings” function in “TriggerConfigRemoteObject” once the connection is established.

```
            if (citrex.IsConnected) {
                citrex.TriggerConfigRemoteObject.RequestSettings();
            }
```

To get notified when a trigger setting has been changed in device:

```
            citrex.TriggerConfigRemoteObject.PropertyChanged +=
TriggerConfigRemoteObject_PropertyChanged;
```

```
        private void TriggerConfigRemoteObject_PropertyChanged(object? sender,
System.ComponentModel.PropertyChangedEventArgs e) {
            switch (e.PropertyName) {
                case TriggerConfigRemoteObject.PROPERTY_ADULT_TRIGGER:
                    Invoke(new Action(() => textBoxTriggerSource.Text =
citrex.TriggerConfigRemoteObject.DeviceAdultTrigger.Source.ToString()));
                    break;
                default:
                    break;
            }
        }
```

To change settings of Adult Trigger, Pediatric Trigger and High Frequency Trigger and send to de

```
            if (citrex.IsConnected) {
                citrex.TriggerConfigRemoteObject.SendTriggerConfigAdult(new
TriggerConfig() {
                    Source = TriggerSource.Internal,
                    StartSignal = TriggerSignal.Flow,
                    StartLevel = 3.0f,
                    StartEdge = TriggerEdge.Rising,
                    StartDelay = 60,
                    EndSignal = TriggerSignal.Flow,
                    EndLevel = -3.0f,
                    EndEdge = TriggerEdge.Falling,
                    EndDelay = 60,
                });

                citrex.TriggerConfigRemoteObject.SendTriggerConfigPediatric(new
TriggerConfig() {
                    Source = TriggerSource.Internal,
                    StartSignal = TriggerSignal.Flow,
                    StartLevel = 1.0f,
                    StartEdge = TriggerEdge.Rising,
                    StartDelay = 60,
                    EndSignal = TriggerSignal.Flow,
                    EndLevel = -1.0f,
                    EndEdge = TriggerEdge.Falling,
                    EndDelay = 60,
                });


citrex.TriggerConfigRemoteObject.SendTriggerConfigHighFrequency(new
TriggerConfig() {
                    Source = TriggerSource.Internal,
                    StartSignal = TriggerSignal.Flow,
                    StartLevel = 3.0f,
                    StartEdge = TriggerEdge.Rising,
                    StartDelay = 30,
                    EndSignal = TriggerSignal.Flow,
                    EndLevel = -3.0f,
                    EndEdge = TriggerEdge.Falling,
                    EndDelay = 30,
                });
             }
```

5.2.1 CITREX TRIGGERCONFIGREMOTEOBJECT DEFINITIONS

The following table contains properties from “TriggerConfigRemoteObject” which can be used to read trigger config values from CITREX devices. Please refer to chapter 5.1 for details on “TriggerConfig”.

For example, the “DeviceAdultTrigger” contains the trigger settings for “Adult Trigger” in the device and is of type “TriggerConfig”, please refer to Chapter 5.1 for details on “TriggerConfig”.

Value	Setting	Type
DeviceAdultTrigger	Adult Trigger settings in the device, such as trigger signal, trigger edge, trigger delay etc.	TriggerConfig
DevicePediatricTrigger	Pediatric Trigger settings in the device, such as trigger signal, trigger edge, trigger delay etc.	TriggerConfig

DeviceHighFrequencyTrigger	High Freq Trigger settings in the device, such as trigger signal, trigger edge, trigger delay etc.	TriggerConfig

The following table contains functions from “TriggerConfigRemoteObject” which can be used to send trigger config values to CITREX devices. There are three functions to send trigger settings to the device. For example, if need to send trigger settings for “Adult Trigger” in device, call the “SendTriggerConfigAdult” function and with input data type “TriggerConfig”, to send trigger settings for “Pediatric Trigger” in device, call “SendTriggerConfigPediatric”, and to send trigger settings for “High Frequency Trigger” in device, call “SendTriggerConfigHighFrequency”.

Function	**Setting**	**Input Data Type**
SendTriggerConfigAdult	Adult Trigger settings in the device, such as trigger signal, trigger edge, trigger delay etc.	TriggerConfig
SendTriggerConfigPediatric	Pediatric Trigger settings in the device, such as trigger signal, trigger edge, trigger delay etc.	TriggerConfig
SendTriggerConfigHighFrequency	High Freq Trigger settings in the device, such as trigger signal, trigger edge, trigger delay etc.	TriggerConfig

5.3 GET AND SET PF-300 PRO TRIGGER CONFIG SETTINGS

For the trigger settings of PF-300 Pro, they can be retrieved with the “TriggerSettingsRemoteObject” member of the FlowAnalyserPro class. After the connection to device is established, the assembly will request trigger settings from PF-300 Pro device, and once the trigger settings are received, the “SettingsReceived” property in “TriggerSettingsRemoteObject” will be set to true. For example, to get the “Trigger Source” of the PF-300 Pro Adult Trigger, Pediatric Trigger and High Frequency Trigger, one can first check the “SettingsReceived” property in “TriggerSettingsRemoteObject”, if true, then read the trigger settings:

```
            if (flowAnalyserPro.TriggerSettingsRemoteObject.SettingsReceived) {
                TriggerSource adultTriggerSourcePf300Pro =
flowAnalyserPro.TriggerSettingsRemoteObject.DeviceAdultTrigger.Source;
                TriggerSource pediatricTriggerSourcePf300Pro =
flowAnalyserPro.TriggerSettingsRemoteObject.DevicePediatricTrigger.Source;
                TriggerSource highFreqTriggerSourcePf300Pro =
flowAnalyserPro.TriggerSettingsRemoteObject.DeviceHighFrequencyTrigger.Source;
            }
```

It is also possible to request trigger settings from PF-300 Pro explicitly by calling “RequestTriggerSettings” in “SettingRequestRemoteObject” after connection is established.

```
            if (!flowAnalyserPro.IsConnected) {

flowAnalyserPro.SettingRequestRemoteObject.RequestTriggerSettings();
            }
```

To get notified when a trigger setting has been changed in device:

```
            flowAnalyserPro.TriggerSettingsRemoteObject.PropertyChanged +=
TriggerSettingsRemoteObject_PropertyChanged;


        private void TriggerSettingsRemoteObject_PropertyChanged(object? sender,
System.ComponentModel.PropertyChangedEventArgs e) {
            switch (e.PropertyName) {
                case TriggerSettingsRemoteObject.PROPERTY_ADULT_TRIGGER:
                    Invoke(new Action(() => textBoxTriggerSource.Text =
flowAnalyserPro.TriggerSettingsRemoteObject.DeviceAdultTrigger.Source.ToString())
);
                    break;
                default:
                    break;
            }
        }
```

To change trigger setting of Adult Trigger, Pediatric Trigger and High Frequency Trigger and send to device:

```
                    if (flowAnalyserPro.IsConnected) {

flowAnalyserPro.TriggerSettingsRemoteObject.SendTriggerConfigAdult(new
TriggerConfig() {
                        Source = TriggerSource.Internal,
                        StartSignal = TriggerSignal.Flow,
                        StartLevel = 3.0f,
                        StartEdge = TriggerEdge.Rising,
                        StartDelay = 60,
                        EndSignal = TriggerSignal.Flow,
                        EndLevel = -3.0f,
                        EndEdge = TriggerEdge.Falling,
                        EndDelay = 60,
                    });


flowAnalyserPro.TriggerSettingsRemoteObject.SendTriggerConfigPediatric(new
TriggerConfig() {
                        Source = TriggerSource.Internal,
                        StartSignal = TriggerSignal.Flow,
                        StartLevel = 1.0f,
                        StartEdge = TriggerEdge.Rising,
                        StartDelay = 60,
                        EndSignal = TriggerSignal.Flow,
                        EndLevel = -1.0f,
                        EndEdge = TriggerEdge.Falling,
                        EndDelay = 60,
                    });


flowAnalyserPro.TriggerSettingsRemoteObject.SendTriggerConfigHighFrequency(new
TriggerConfig() {
                        Source = TriggerSource.Internal,
                        StartSignal = TriggerSignal.Flow,
                        StartLevel = 3.0f,
                        StartEdge = TriggerEdge.Rising,
                        StartDelay = 30,
                        EndSignal = TriggerSignal.Flow,
                        EndLevel = -3.0f,
                        EndEdge = TriggerEdge.Falling,
                        EndDelay = 30,
                    });
                }
```

5.3.1 PF-300 PRO TRIGGERSETTINGSREMOTEOBJECT DEFINITIONS

The following table contains properties from “TriggerSettingsRemoteObject” which can be used to read trigger settings values from PF-300 Pro.

Note: the trigger settings here do not include “Trigger Mode” which is in “MeasurementSettingsRemoteObject”, refer to Chapter 4.2.1 for reading and sending “Trigger Mode”.

For example, the “DeviceAdultTrigger” contains the trigger settings for “Adult Trigger” in the device and is of type “TriggerConfig”, please refer to Chapter 5.1 for details on “TriggerConfig”.

Value	Setting	Type
DeviceAdultTrigger	Adult Trigger settings in the device, such as trigger signal, trigger edge, trigger delay etc.	TriggerConfig

DevicePediatricTrigger	Pediatric Trigger settings in the device, such as trigger signal, trigger edge, trigger delay etc.	TriggerConfig
DeviceHighFrequencyTrigger	High Freq Trigger settings in the device, such as trigger signal, trigger edge, trigger delay etc.	TriggerConfig

The following table contains functions from “TriggerSettingsRemoteObject” which can be used to send trigger settings values to PF-300 Pro. There are three functions to send trigger settings to the device. For example, if need to send trigger settings for “Adult Trigger” in device, call the “SendTriggerConfigAdult” function and with input data type “TriggerConfig”, to send trigger settings for “Pediatric Trigger” in device, call “SendTriggerConfigPediatric”, and to send trigger settings for “High Frequency Trigger” in device, call “SendTriggerConfigHighFrequency”.

Function	**Setting**	**Input Data Type**
SendTriggerConfigAdult	Adult Trigger settings in the device, such as trigger signal, trigger edge, trigger delay etc.	TriggerConfig
SendTriggerConfigPediatric	Pediatric Trigger settings in the device, such as trigger signal, trigger edge, trigger delay etc.	TriggerConfig
SendTriggerConfigHighFrequency	High Freq Trigger settings in the device, such as trigger signal, trigger edge, trigger delay etc.	TriggerConfig

6 DEBUG LOG

It is possible to enable the debug log from the assembly, which will log the debug information and exceptions to log files under “C:\ProgramData\ImtUsbProtocol\” directory.

By calling the function “SetDebugOutput” in “DebugOutput” class, the debug output will be enabled.

There are three levels of debug output: “None”, “Standard” and “Detail”.

“None”: write no debug output.
“Standard”: write standard debug output.
“Detail”: write detailed debug output.

How to enable the debug log:

```
DebugOutput.SetDebugOutput();
DebugOutput.DebugLevel = DebugLevel.Standard;
```