EVOLUTIONS:

| Date | Revision | Description |
| --- | --- | --- |
| 27/10/2015 | F | Ethernet Interface for F28 light device. |
| 05/11/2015 | G | Update to 1.303 version, add VB.net samples. |
| 12/11/2015 | H | Mistakes fix. |
| 17/11/2015 | I | Add Auto calibration functions |
| 19/11/2015 | J | Add parameters Option's (page 46) and auto calibration function (page 69). |
| 24/11/2015 | K | Add parameters and explanation, DLL version 1.402. |
| 02/12/2015 | L | Add explanation for auto-calibration with 5 devices, DLL version 1.402. |
| 04/12/2015 | M | Add "Start Auto Cal Offset for more than one head in VB.net" DLL version 1.402. |
| 14/12/2015 | N | Add specifics error codes (§ 3.9.1) add electronic regulator option, DLL version 1.500. |
| 21/01/2016 | O | Update error codes on §3.9.4 "Result status and alarms". |
| 02/02/2016 | P | Update error codes on §3.9.4 "Result status and alarms". |
| 15/02/2016 | Q | Change to negatives values error codes on §3.9.4 "Result status and alarms". |
| 25/07/2016 | R | 1- Use F28_RemoveModule then F28_AddModule without reinitialize all.<br>2- :FIX: Some declarations that can cause Unbalanced Stack |
| 06/03/2017 | S | Add F28 Jet Check special cycle, update error messages, dll version 2.004. |
| 02/08/2017 | T | Units list modified (add several units). |
| 10/12/2018 | U | Add commun functions (all devices)<br>Add Auto-Ratio F28 function (V2.013)<br>Add B28 device |
| 09/01/2019 | V | Add Rise time (B28 step cycle and parameter) |
| 05/04/2019 | W | Add special cycles : compute volume test, |

| | | |
| --- | --- | --- |
| | | compute volume reference, infinite dump. |
| 11/04/2019 | X | Update test type : Flow test<br>Update status list : Incompatible device |
| 20/06/2019 | Y | Dll version 2.016<br>Update F28 parameters structure<br>Update F28 fill type enumeration<br>Update F28 step cycle enumeration |
| 28/06/2019 | Z | Dll version 2.017<br>Add special cycle Easy auto learning |
| 26/09/2019 | Z1 | Dll Version 2.018<br>B28 Modifications :<br>- Add alarm PST<br>- Add 2 Parameters (Pressure min and max)<br>- Add special cycle (regulator adjust) |
| 04/11/2019 | Z2 | F28 Modifications :<br>- Add leak unit ugH2O |

***Important!*** *This last DLL version works only with devices with firmware version ≥ 1.500.*

# Table of contents

# 1 Introduction

## 1.1 About this document

This manual describes the F28Light ***Application Programming Interface*** *(API)* and the containing functions. As a Win32 DLL for windows W7, W8, W10, it forms the interface between the user application and the B28 or F28Light.

## 1.2 What's needed for using the DLL

**<u>* 32 bits version</u>**

**vcredist_x86.exe :** Visual C++ Redistributable Packages for Visual Studio 2013.

**DLL Ethernet interface :** F28LightControl_ETH.dll

The Visual C++ Redistributable Packages install run-time components that are required to run applications that are developed by using Visual Studio 2013, on computers that don't have Visual Studio 2013 installed.

These packages install run-time components of these libraries: C Runtime (CRT), Standard C++, ATL, MFC, C++ AMP, and OpenMP.

**<u>* 64 bits version</u>**

**vcredist_x64.exe :** Visual C++ Redistributable Packages for Visual Studio 2013.

**DLL Ethernet interface :** F28LightControl_ETH64.dll

The Visual C++ Redistributable Packages install run-time components that are required to run applications that are developed by using Visual Studio 2013, on computers that don't have Visual Studio 2013 installed.

These packages install run-time components of these libraries: C Runtime (CRT), Standard C++, ATL, MFC, C++ AMP, and OpenMP.

***Nota** : All functions are the same between 32 bits or 64 bits. Just the Dll name is different. In this document, functions description is for 32 bits version, just replace F28LightControl_ETH.dll with F28LightControl_ETH64.dll in declaration for 64 bits version.*

***Nota** :*

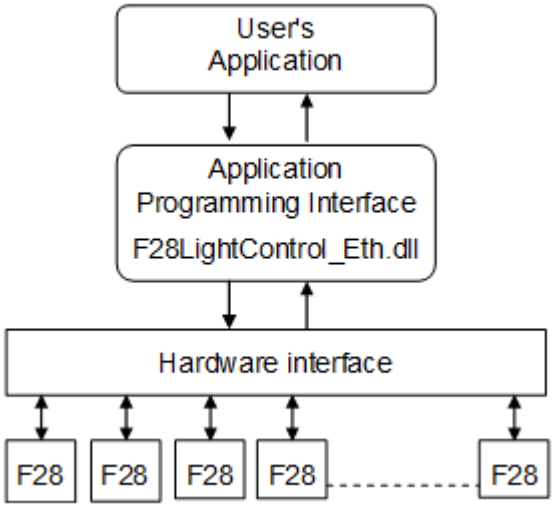
*- Functions starting with "X28_" concern all type of devices.*

*- Functions starting with "F28_" concern F28 device.*

*- Functions starting with "B28_" concern B28 device.*

***Nota**: All functions starting with "X28_" can be used also using the name "F28_" (for compatibility with older version).*

## 1.3 Application structures



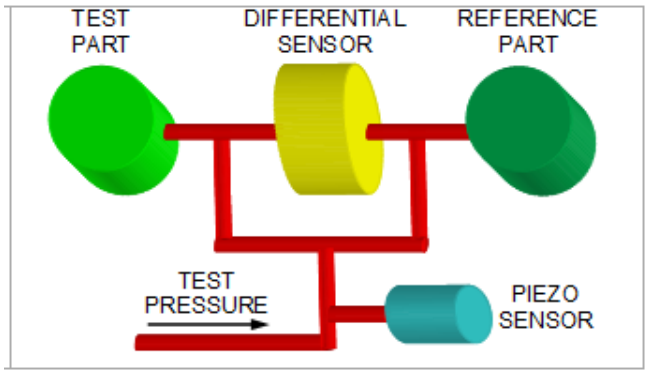

F28Light is a static slave : never send information by itself.

Only one master is supported in the network.

This is the same for B28 device.

## 1.4 Reminder of the ATEQ F28Light principle

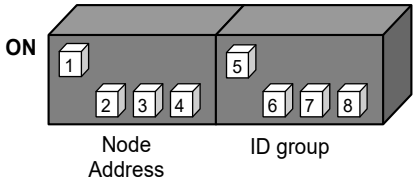
The **ATEQ F28 Light** is a compact air/air leak detector used to test the air-tightness of parts. The method used is based on the measurement of a small variation or drop in differential pressure between the **Test** and **Reference** parts, when both are filled to an identical pressure.



## 1.5 B28/F28 heads IP addresses

## 1.6 Switches configuration

On each head, on the main board, there's one switch to give a hardware address. The head must be configured as the following example.




Switch **#1** (node) = **On**

Switch **#5** (group) = **On**

Others switches = **Off**.

## 1.7 IP address configuration

The IP default configuration for the F28 Light/B28 device is in DHCP mode. This mode allows to automatically applying an IP address to the device by a rooter in the network.

For the first start, the device waits 10 seconds for a DHCP configuration. If it is not detected after these 10 seconds, the static **192.168.1.200** IP address is set.

To run the device in the network and update the boards, please use a static IP address.

This static IP address must be different for all the devices connected to the same network.

Keep one IP address for the PC and give different ones to the F28Light/B28 devices.

Example:

- **PC**: IP 192.168.1.**1**
- F28Light **#1**: IP 192.168.1.**2**
- F28Light **#2**: IP 192.168.1.**3**
- F28Light **#3**: IP 192.168.1.**4**
- F28Light **#4**: IP 192.168.1.**5** Etc…

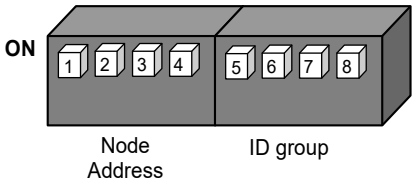
***Remind**: the network's rooter must have the same IP address root than the PC, example: **192.168.1**.X otherwise the device won't be detected.*

## 1.8 IP address loss

If the IP address is lost or nor recoverable, the communication between the device and the PC in the network is impossible.

To recover the communication, you must reset the IP address assignation, to be able to give another one.

For that, with the device powered off, set all the "Address" and the "Group" switches to 1.




Switches **#1** to **#8** (node) = **On**

Then power on the device for a few seconds and power off, the IP address is reseted.

## 1.9 Data definitions

| Definition | Description |
| --- | --- |
| BYTE, UCHAR | Unsigned char (8 bits) |
| char | Signed char (8 bits) |
| short | Signed word (2 bytes) |
| WORD | Unsigned word (2 bytes) |
| float | Floating point single precision (4 bytes) |
| DWORD | Unsigned word (4 bytes) |

# 2 Common structure and enumeration

This chapter concern all devices (B28/F28).

## 2.1 Enumeration

### 2.1.1 Boot/Application mode

Returned by function "**X28_GetMode**".

| Declaration | Data type | Value | Description |
| --- | --- | --- | --- |
| X28_MODE_BOOT | short | 1 | Boot mode |
| X28_MODE_APPLICATION | | 2 | Application mode |

**Declaration in C/C++:**

```
#define X28_MODE_BOOT 1
#define X28_MODE_APPLICATION 2
```

**Visual Basic (Vb.Net):**

```
Enum X28_MODE As Byte
     X28_MODE_BOOT = 1
     X28_MODE_APPLICATION = 2
End Enum
```

**C#.Net:**

```
public enum X28_MODE : byte
{
     X28_MODE_BOOT = 1,
     X28_MODE_APPLICATION = 2
};
```

### 2.1.2 Group identifier

Value for variable "**ucGroupID**".

| Variable | Data type | Value | Description |
| --- | --- | --- | --- |
| ucGroupID | BYTE | X28_GROUP_1 = 1,<br>X28_GROUP_2 = 2,<br>X28_GROUP_3 = 3,<br>X28_GROUP_4 = 4,<br>X28_GROUP_5 = 5,<br>X28_GROUP_6 = 6,<br>X28_GROUP_7 = 7 ,<br>X28_GROUP_8 = 8,<br>X28_GROUP_9 = 9,<br>X28_GROUP_10 = 10,<br>X28_GROUP_11 = 11,<br>X28_GROUP_12 = 12,<br>X28_GROUP_13 = 13,<br>X28_GROUP_14 = 14,<br>X28_GROUP_15 = 15 | Group identifier (1 -15) |

**Declaration in C/C++:**

```
enum
{
      X28_GROUP_1 = 1,
      X28_GROUP_2,
      X28_GROUP_3,
      X28_GROUP_4,
      X28_GROUP_5,
      X28_GROUP_6,
      X28_GROUP_7,
      X28_GROUP_8,
      X28_GROUP_9,
      X28_GROUP_10,
      X28_GROUP_11,
      X28_GROUP_12,
      X28_GROUP_13,
      X28_GROUP_14,
      X28_GROUP_15,
      X28_GROUP_MAX
};
```

**Visual Basic (Vb.Net):**

```
Enum X28_GROUP_ID As Byte
      X28_GROUP_1 = 1
      X28_GROUP_2
      X28_GROUP_3
      X28_GROUP_4
      X28_GROUP_5
      X28_GROUP_6
      X28_GROUP_7
      X28_GROUP_8
      X28_GROUP_9
      X28_GROUP_10
      X28_GROUP_11
      X28_GROUP_12
      X28_GROUP_13
      X28_GROUP_14
      X28_GROUP_15
      X28_GROUP_MAX
End Enum
```

**C#.Net:**

```
public enum X28_GROUP_ENUM : byte
{
      X28_GROUP_1 = 1,
      X28_GROUP_2,
      X28_GROUP_3,
      X28_GROUP_4,
      X28_GROUP_5,
      X28_GROUP_6,
      X28_GROUP_7,
      X28_GROUP_8,
      X28_GROUP_9,
      X28_GROUP_10,
      X28_GROUP_11,
      X28_GROUP_12,
```

```
      X28_GROUP_13,
      X28_GROUP_14,
      X28_GROUP_15;
      X28_GROUP_MAX
};
```

## 2.2 Module address

Value for variable "**ucModuleAddr**".

| Variable | Data type | Value | Description |
| --- | --- | --- | --- |
| ucModuleAddr | BYTE | X28_MODULE_ADDR_0 = 0<br>X28_MODULE_ADDR_1 = 1<br>X28_MODULE_ADDR_2 = 2<br>X28_MODULE_ADDR_3 = 3<br>X28_MODULE_ADDR_4 = 4<br>X28_MODULE_ADDR_5 = 5<br>X28_MODULE_ADDR_6 = 6<br>X28_MODULE_ADDR_7 = 7<br>X28_MODULE_ADDR_8 = 8<br>X28_MODULE_ADDR_9 = 9<br>X28_MODULE_ADDR_10 = 10<br>X28_MODULE_ADDR_11 = 11<br>X28_MODULE_ADDR_12 = 12<br>X28_MODULE_ADDR_13 = 13<br>X28_MODULE_ADDR_14 = 14<br>X28_MODULE_ADDR_15 = 15 | Station address (0 -15) |

**Declaration in C/C++:**

```
enum
{
      X28_MODULE_ADDR_0,
      X28_MODULE_ADDR_1,
      X28_MODULE_ADDR_2,
      X28_MODULE_ADDR_3,
      X28_MODULE_ADDR_4,
      X28_MODULE_ADDR_5,
      X28_MODULE_ADDR_6,
      X28_MODULE_ADDR_7,
      X28_MODULE_ADDR_8,
      X28_MODULE_ADDR_9,
      X28_MODULE_ADDR_10,
      X28_MODULE_ADDR_11,
      X28_MODULE_ADDR_12,
      X28_MODULE_ADDR_13,
      X28_MODULE_ADDR_14,
      X28_MODULE_ADDR_15,
      X28_MAX_MODULES_BY_GROUP
};
```

**Visual Basic (Vb.Net):**

```
Enum X28_MODULE_ADDR As Byte
      X28_MODULE_ADDR_0 = 0
      X28_MODULE_ADDR_1
      X28_MODULE_ADDR_2
      X28_MODULE_ADDR_3
      X28_MODULE_ADDR_4
      X28_MODULE_ADDR_5
      X28_MODULE_ADDR_6
      X28_MODULE_ADDR_7
      X28_MODULE_ADDR_8
      X28_MODULE_ADDR_9
      X28_MODULE_ADDR_10
      X28_MODULE_ADDR_11
      X28_MODULE_ADDR_12
      X28_MODULE_ADDR_13
      X28_MODULE_ADDR_14
      X28_MODULE_ADDR_15
      X28_MODULE_MAX
End Enum
```

**C#.Net:**

```
public enum X28_MODULE_ADDR_ENUM : byte
{
      X28_MODULE_ADDR_0,
      X28_MODULE_ADDR_1,
      X28_MODULE_ADDR_2,
      X28_MODULE_ADDR_3,
      X28_MODULE_ADDR_4,
      X28_MODULE_ADDR_5,
      X28_MODULE_ADDR_6,
      X28_MODULE_ADDR_7,
      X28_MODULE_ADDR_8,
      X28_MODULE_ADDR_9,
      X28_MODULE_ADDR_10,
      X28_MODULE_ADDR_11,
      X28_MODULE_ADDR_12,
      X28_MODULE_ADDR_13,
      X28_MODULE_ADDR_14,
      X28_MODULE_ADDR_15,
      X28_MAX_MODULES_BY_GROUP
};
```

## 2.3 Structure definition in C/C++

***Nota**: All structures are 1 byte packed, for easy portability and data exchange between API and Visual basic 2013 application.*

```
#pragma pack(push, 1 )

// Date structure
typedef struct
{
      WORD wYear;
      WORD wMonth;
      WORD wDay;
      WORD wHour;
      WORD wMinute;
      WORD wSecond;
} X28_DATE;

// Cycle statistics structure
typedef struct
{
      DWORD dwTotalCycles;
      DWORD dwFailCycles;
      DWORD dwSuccessCycles;
} X28_CYCLE_STATISTICS;

// Communication statistics structure
```

```
typedef struct
{
      DWORD dwTransmited;
      DWORD dwReceived;
      DWORD dwErrors;
} X28_COMMUNICATION_STATISTICS;

#pragma pack(pop)
```

## 2.4 Structure definition in C#.NET

```
[StructLayout(LayoutKind.Sequential, Pack = 1, CharSet = CharSet.Ansi)]
public struct X28_DATE
{
      public ushort usYear;
      public ushort usMonth;
      public ushort usDay;
      public ushort usHour;
      public ushort usMinute;
      public ushort usSecond;
};

[StructLayout(LayoutKind.Sequential, Pack = 1, CharSet = CharSet.Ansi)]
public struct X28_CYCLE_STATISTICS
{
      public uint uiTotalCycles;
      public uint uiFailCycles;
      public uint uiSuccessCycles;
};

[StructLayout(LayoutKind.Sequential, Pack = 1, CharSet = CharSet.Ansi)]
public struct X28_COMMUNICATION_STATISTICS
{
      public uint uiTransmited;
      public uint uiReceived;
      public uint uiErrors;
};
```

## 2.5 Structure definition in Visual Basic

```
' -------------------------
' Date structure
' -------------------------
<StructLayout(LayoutKind.Sequential, Pack:=1)> _
Structure X28_DATE
      Dim wYear As UShort
      Dim wMonth As UShort
```

```
      Dim wDay As UShort
      Dim wHour As UShort
      Dim wMinute As UShort
      Dim wSecond As UShort
End Structure

' -------------------------
' Statistic structure
' -------------------------
<StructLayout(LayoutKind.Sequential, Pack:=1)> _
Structure X28_CYCLE_STATISTICS
      Dim dwTotalCycles As UInteger
      Dim dwFailCycles As UInteger
      Dim dwSuccessCycles As UInteger
End Structure

' -------------------------
' Communication counter structure
' -------------------------
<StructLayout(LayoutKind.Sequential, Pack:=1)> _
Structure X28_COMMUNICATION_STATISTICS
      Dim dwTransmited As UInteger
      Dim dwReceived As UInteger
      Dim dwErrors As UInteger
End Structure
```

## 2.6 Function Return code

| Declaration | Data type | Value | Description |
| --- | --- | --- | --- |
| X28_FAIL | short | -1 | Error |
| X28_OK | | 0 | Ok |

**Declaration in C/C++:**

```
enum  X28_RETURN
{
      X28_FAIL = -1,
      X28_OK
};
```

**Visual Basic (Vb.Net):**

```
Enum X28_RETURN As Short
      X28_FAIL = -1
      X28_OK = 0
End Enum
```

**C#.Net:**

```
public enum X28_RETURN : byte
{
      X28_FAIL = -1,
      X28_OK
};
```

***Nota** : For F28_ functions, F28_FAIL and F28_OK can also be used. Definitions are the same as X28_FAIL and F28_OK.*

## 2.7 Connection status

This code is returned by X28_IsModuleConnected.

| Declaration | Data type | Value | Description |
| --- | --- | --- | --- |
| X28_OFFLINE | short | 0 | Offline |
| X28_CONNECTED | | 1 | Unit connected |

**Declaration in C/C++:**

```
enum  X28_CONNECTION
{
      X28_OFFLINE = 0,
      X28_CONNECTED
};
```

**Visual Basic (Vb.Net):**

```
Enum X28_CONNECTION As Short
      X28_OFFLINE = -0
      X28_CONNECTED
End Enum
```

**C#.Net:**

```
public enum X28_CONNECTION : byte
{
      X28_OFFLINE = 0,
      X28_CONNECTED
};
```

# 3 B28 structure and enumeration

## 3.1 Enumeration

### 3.1.1 Type of test

Test type parameter, to be use with "**wTypeTest**".

| Definition | Value | Description |
| --- | --- | --- |
| UNDEFINED_TEST | 0 | Not defined |
| IONIC | 1 | IONIQ test |

**Declaration in C/C++:**

```
enum B28_TYPE_TEST
{
      B28_UNDEFINED_TEST,
      B28_IONIQ
};
```

**Visual Basic (Vb.Net):**

```
Enum B28_TYPE_TEST    'Uses with wTypeTest parameter
      B28_UNDEFINED_TEST
      B28_IONIQ
      '
End Enum
```

**C#.Net:**

```
public enum B28_TYPE_TEST : byte
{
      B28_UNDEFINED_TEST,
      B28_IONIQ
};
```

### 3.1.2 Voltage unit

Voltage unit parameter, to be use with “**wVoltageUnit**”.

| Definition | Value | Description |
| --- | --- | --- |
| VOLTAGE_VOLT | 0 | Volt |
| VOLTAGE_POINT | 1 | Point (internal debug) |

**Declaration in C/C++:**

```
enum B28_VOLTAGE_UNITS
{
      B28_VOLTAGE_VOLT,
      B28_VOLTAGE_POINT,
      B28_NMAX_VOLTAGE_UNITS
```

```
};
```

**Visual Basic (Vb.Net):**

```
Enum B28_VOLTAGE_UNITS As Byte
      B28_VOLTAGE_VOLT
      B28_VOLTAGE_POINT
      B28_NMAX_PRESS_UNITS
End Enum
```

**C#.Net:**

```
public enum B28_VOLTAGE_UNITS : byte
{
      B28_VOLTAGE_VOLT,
      B28_VOLTAGE_POINT,
      B28_NMAX_VOLTAGE_UNITS
};
```

### 3.1.3 Measurement unit

Measurement unit parameter, to be use with “**wMeasurementUnit**”.

| Definition | Value | Description |
| --- | --- | --- |
| MEASUREMENT_POINT | 0 | Point |
| MEASUREMENT_RAW_POINT | 1 | Raw point (without autorange for internal debug) |

**Declaration in C/C++:**

```
enum B28_MEASUREMENT_UNITS
{
      B28_MEASUREMENT_POINT,
      B28_MEASUREMENT_RAW_POINT,
      B28_NMAX_MEASUREMENT_UNITS
};
```

**Visual Basic (Vb.Net):**

```
Enum B28_MEASUREMENT_UNITS As Byte
      B28_MEASUREMENT_POINT
      B28_MEASUREMENT_RAW_POINT
      B28_NMAX_MEASUREMENT_UNITS
End Enum
```

**C#.Net:**

```
public enum B28_MEASUREMENT_UNITS : byte
{
      B28_MEASUREMENT_POINT,
      B28_MEASUREMENT_RAW_POINT,
      B28_NMAX_MEASUREMENT_UNITS
};
```

## 3.2 Step code

Step code value for variable ucStatus inside B28_REALTIME_CYCLE structure.

| Variable | Data type | Value | Description |
| --- | --- | --- | --- |
| ucStatus | BYTE | READY = 0, | Out of cycle |
| | BYTE | NO_STEP1 = 1, | Not used |
| | BYTE | NO_STEP2 = 2, | Not used |
| | BYTE | TEST_STEP = 3, | Test step |
| | BYTE | FALL_STEP =4 | Fall step |

**Declaration in C/C++:**

```
enum B28_ENUM_STEP_CODE
{
      B28_READY,
      B28_RISE_STEP,
      B28_NO_STEP,
      B28_TEST_STEP,
      B28_FALL_STEP
};
```

**Visual Basic (Vb.Net):**

```
Enum B28_ENUM_STEP_CODE As Byte
      READY
      B28_RISE_STEP
      B28_NO_STEP
      B28_TEST_STEP
      B28_FALL_STEP
End Enum
```

**C#.Net:**

```
public enum B28_ENUM_STEP_CODE : byte
{
      READY,
      B28_RISE_STEP,
      B28_NO_STEP,
      B28_TEST_STEP,
      B28_FALL_STEP
};
```

## 3.3 Identifier of module

Value for variable sModuleID. The identifier of the module is unique. It returns by the function "**B28_AddModule**".

| Variable | Data type | Description |
| --- | --- | --- |
| sModuleID | short | High byte = index of channel.<br>Low byte = index of module. |

## 3.4 Structure definition in C/C++

***Nota:*** *All structures are 1 byte packed, for easy portability and data exchange between API and Visual basic 2013 application.*

```
#pragma pack(push, 1 )
```

```
// Result structure
typedef struct
{
      UCHAR ucStatus;
      float fVoltageValue;
      float fMeasurementValue;
      UCHAR ucUnitVoltage;
      UCHAR ucUnitMeasurement;
      BYTE ucGroupID;
      BYTE ucModuleAddr;
      X28_DATE dateReceived;
} B28_RESULT;

// Real time structure
typedef struct B28_REALTIME_CYCLE
{
      UCHAR ucEndCycle;
      UCHAR ucStatus;
      float fVoltageValue;
      float fMeasurementValue;
      UCHAR ucUnitVoltage;
      UCHAR ucUnitMeasurement;
      float fInternalTemperature;
      float fPatm;
} B28_REALTIME_CYCLE;

// Parameter structure
typedef struct B28_PARAMETERS
{
     WORD    wTypeTest;           // Test type
     WORD    wTestTime;           // Test time
     WORD    wFallTime;           // Fall time
     float   fVoltageSetPoint;    // Target voltage
     WORD    wVoltageUnit;        // Voltage unit
     float   fMeasurementMax;     // Maximum Measurement
     WORD    wMeasurementUnit;    // Measurement unit
     WORD    wRiseTime;           // Rise time
     WORD    wOptions;            // Options
     long    lPressureMin;        // Pressure Min (Pa)
     long    lPressureMax;        // Pressure Max (Pa)
} B28_PARAMETERS;

#pragma pack(pop)
```

## 3.5 Structure definition in Visual Basic

```
' -------------------------
' Result structure
' -------------------------
<StructLayout(LayoutKind.Sequential, Pack:=1)> _
Structure B28_RESULT
```

```
     Dim ucStatus As Byte
     Dim fVoltageValue As Single
     Dim fMeasurementValue As Single
     Dim ucUnitVoltage As Byte
     Dim ucUnitMeasurement As Byte
     Dim GroupID As Byte                    ' X28_GROUP_ID
     Dim ModuleAddr As Byte                 ' X28_MODULE_ADDR
     Dim dateReceived As X28_DATE
End Structure

' -------------------------
' real time result structure
' -------------------------
<StructLayout(LayoutKind.Sequential, Pack:=1)> _
Structure B28_REALTIME_CYCLE
     Dim ucEndCycle As Byte
     Dim ucStatus As Byte
     Dim fVoltageValue As Single
     Dim fMeasurementValue As Single
     Dim ucUnitVoltage As Byte
     Dim ucUnitMeasurement As Byte
     Dim fInternalTemperature As Single    ' Temperature in °C (NU)
     Dim fPatm As Single                   ' Abs pressure in Pa
End Structure

' -------------------------
' Parameter structure
' -------------------------
<StructLayout(LayoutKind.Sequential, Pack:=1)> _
Structure B28_PARAMETERS
     Dim wTypeTest As UShort
     Dim wTpsTest As UShort
     Dim wTpsFall As UShort
     Dim fVoltageSetPoint As Single
     Dim wVoltageUnit As UShort
     Dim fMeasurementMax As Single
     Dim wMeasurementUnit As UShort  ' See B28_MEASUREMENT_UNITS
     Dim wRiseTime As Ushort
     Dim wOptions As Ushort
     Dim lPressureMin As UInteger
     Dim lPressureMax As UInteger
End Structure
```

## 3.6 Structure definition in C#.NET

```
[StructLayout(LayoutKind.Sequential, Pack = 1, CharSet = CharSet.Ansi)]
public struct B28_RESULT
{
     public byte     bStatus;
     public float    fVoltageValue;
     public float    fMeasurementValue;
     public byte     bUnitVoltage;
```

```
     public byte     bUnitMeasurement;
     public byte     bGroupID;
     public byte     bModuleAddr;
     public X28_DATE dateReceived;
};


<StructLayout(LayoutKind.Sequential, Pack:=1)> _
[StructLayout(LayoutKind.Sequential, Pack = 1, CharSet = CharSet.Ansi)]
public struct B28_REALTIME_CYCLE
     public byte     ucEndCycle;
     public byte     ucStatus;
     public float    fVoltageValue;
     public float    fMeasurementValue;
     public byte     ucUnitVoltage;
     public byte     ucUnitMeasurement;
     public float    fInternalTemperature;
     public float    fPatm;
};


[StructLayout(LayoutKind.Sequential, Pack = 1, CharSet = CharSet.Ansi)]
public struct B28_PARAMETERS
{
     public ushort usTypeTest;
     public ushort usTestTime;
     public ushort usFallTime;
     public float fVoltageSetPoint;
     public ushort usVoltageUnit;
     public float fMeasurementMax;
     public ushort usMeasurementUnit;
     public ushort usRiseTime;
     public ushort usOptions;
     public uint uiPressureMin;
     public uint uiPressureMax;
};
```

# 4 F28 structure and enumeration

## 4.1 Enumeration

### 4.1.1 Type of test

Test type parameter, to be use with "**wTypeTest**".

| Definition | Value | Description |
| --- | --- | --- |
| UNDEFINED_TEST | 0 | Not defined |
| LEAK_TEST | 1 | Leak test |
| SEALED_COMPONENT_TEST | 2 | Sealed components test |
| DESENSITIZED_MODE_TEST | 3 | Desensitized mode test for measurement of large leaks. * |
| FLOW_TEST | 4 | Flow test |

**Declaration in C/C++:**

```
enum F28_TYPE_TEST
{
      UNDEFINED_TEST,
      LEAK_TEST,
      SEALED_COMPONENT_TEST,
      DESENSITIZED_MODE_TEST, // Since v1.500 only
      FLOW_TEST,
};
```

**Visual Basic (Vb.Net):**

```
Enum F28_TYPE_TEST    'Uses with wTypeTest parameter
      UNDEFINED_TEST
      LEAK_TEST
      SEALED_COMPONENT_TEST
      DESENSITIZED_MODE_TEST // Since v1.500 only
      FLOW_TEST
      '
End Enum
```

**C#.Net:**

```
public enum F28_TYPE_TEST : byte
{
      UNDEFINED_TEST,
      LEAK_TEST,
      SEALED_COMPONENT_TEST,
      DESENSITIZED_MODE_TEST, // Since v1.500 only
      FLOW_TEST,
};
```

***Nota**: the desensitized mode is used for the measurement of large leaks, when the reject level required is above the full scale of the differential sensor; the measurement is performed by the pressure sensor.*

### 4.1.2 Pressure unit

Pressure unit parameter, to be use with "**wPress1Unit**".

| Definition | Value | Description |
| --- | --- | --- |
| PRESS_PA | 0 | Pascal |
| PRESS_KPA, | 1 | Kilo pascal |
| PRESS_MPA | 2 | Mega pascal |
| PRESS_BAR | 3 | Bar |
| PRESS_mBAR | 4 | Millibar |
| PRESS_PSI | 5 | PSI |
| PRESS_POINTS | 6 | Points |

**Declaration in C/C++:**

```
enum F28_PRESS_UNITS
{
      PRESS_PA,
      PRESS_KPA,
      PRESS_MPA,
      PRESS_BAR,
      PRESS_mBAR,
      PRESS_PSI,
      PRESS_POINTS,
      NMAX_PRESS_UNITS
};
```

**Visual Basic (Vb.Net):**

```
Enum F28_PRESS_UNITS As Byte
      PRESS_PA
      PRESS_KPA
      PRESS_MPA
      PRESS_BAR
      PRESS_mBAR
      PRESS_PSI
      PRESS_POINTS
NMAX_PRESS_UNITS
End Enum
```

**C#.Net:**

```
public enum F28_PRESS_UNITS : byte
{
      PRESS_PA,
      PRESS_KPA,
      PRESS_MPA,
      PRESS_BAR,
      PRESS_mBAR,
      PRESS_PSI,
      PRESS_POINTS,
      NMAX_PRESS_UNITS
};
```

### 4.1.3 Leak units

Leak unit parameter, to be use with "**wLeakUnit**".

| Definition | Value | Description |
| --- | --- | --- |
| LEAK_PA | 0 | Pa |
| LEAK_PASEC | 1 | Pa/s |
| LEAK_PA_HR | 2 | Pa (High resolution) |
| LEAK_PASEC_H | 3 | Pa/s(High resolution) |
| LEAK_CAL_PA, | 4 | Calibrated Pascal. |
| LEAK_CAL_PASEC, | 5 | Calibrated Pascal/second |
| LEAK_CCMIN, | 6 | $cm^3$/min |
| LEAK_CCSEC, | 7 | $cm^3$/s |
| LEAK_CCH, | 8 | $cm^3$/h. |
| LEAK_MM3SEC, | 9 | $mm^3$/s |
| LEAK_CM3_SEC, | 10 | $cm^3$/s |
| LEAK_CM3_MIN, | 11 | $cm^3$/mn |
| LEAK_CM3_H, | 12 | $cm^3$/h |
| LEAK_ML_SEC, | 13 | ml/s |
| LEAK_ML_MIN, | 14 | ml/min |
| LEAK_ML_H, | 15 | ml/h |
| **USA** | | |
| LEAK_INCH3_SEC, | 16 | $Inch^3$/s |
| LEAK_INCH3_MIN, | 17 | $Inch^3$/mn |
| LEAK_INCH3_H, | 18 | $Inch^3$/h |
| LEAK_FT3_SEC, | 19 | $Feet^3$/s |
| LEAK_FT3_MIN, | 20 | $Feet^3$/mn |
| LEAK_FT3_H, | 21 | $Feet^3$/h |
| LEAK_MMCE, | 22 | mmWg |
| LEAK_MMCE_SEC, | 23 | mmWg/s |
| LEAK_SCCM, | 24 | sccm |
| LEAK_POINTS, | 25 | Points |
| LEAK_KPA, | 26 | kPa |
| LEAK_MPA, | 27 | MPa |
| LEAK_BAR, | 28 | Bar |
| LEAK_mBAR, | 29 | mbar |
| LEAK_PSI, | 30 | PSI |
| LEAK_L_MIN, | 31 | Litre/min |
| LEAK_CM_H2O, | 32 | Cm $H_2O$ |
| LEAK_UG_H2O | 33 | ug $H_2O$ |

**Declaration in C/C++:**

```
enum F28_LEAK_UNITS
{
      LEAK_PA,
      LEAK_PASEC,
      LEAK_PA_HR,
```

**Visual Basic (Vb.Net):**

```
Enum F28_LEAK_UNITS As Byte ' Uses with
                wLeakUnit parameter
      LEAK_PA
      LEAK_PASEC
      LEAK_PA_HR
```

```
    LEAK_PASEC_HR,
    LEAK_CAL_PA,
    LEAK_CAL_PASEC,
    LEAK_CCMIN,
    LEAK_CCSEC,
    LEAK_CCH,
    LEAK_MM3SEC,
    LEAK_CM3_SEC,
    LEAK_CM3_MIN,
    LEAK_CM3_H,
    LEAK_ML_SEC,
    LEAK_ML_MIN,
    LEAK_ML_H,
    LEAK_INCH3_SEC,
    LEAK_INCH3_MIN,
    LEAK_INCH3_H,
    LEAK_FT3_SEC,
    LEAK_FT3_MIN,
    LEAK_FT3_H,
    LEAK_MMCE,
    LEAK_MMCE_SEC,
    LEAK_SCCM,
    LEAK_POINTS,
    // 1.500 Leak units
    LEAK_KPA,
    LEAK_MPA,
    LEAK_BAR,
    LEAK_mBAR,
    LEAK_PSI,
    LEAK_L_MIN,
    LEAK_CM_H2O,
    LEAK_UG_H2O,
    NMAX_LEAK_UNITS
};
```

```
    LEAK_PASEC_HR
    LEAK_CAL_PA
    LEAK_CAL_PASEC
    LEAK_CCMIN
    LEAK_CCSEC
    LEAK_CCH
    LEAK_MM3SEC
    LEAK_CM3_SEC ' 10
    LEAK_CM3_MIN
    LEAK_CM3_H
    LEAK_ML_SEC
    LEAK_ML_MIN
    LEAK_ML_H
    LEAK_INCH3_SEC
    LEAK_INCH3_MIN
    LEAK_INCH3_H
    LEAK_FT3_SEC
    LEAK_FT3_MIN ' 20
    LEAK_FT3_H
    LEAK_MMCE
    LEAK_MMCE_SEC
    LEAK_SCCM
    LEAK_POINTS
    LEAK_KPA
    LEAK_MPA
    LEAK_BAR
    LEAK_mBAR
    LEAK_PSI
    LEAK_L_MIN
    LEAK_CM_H2O
    LEAK_UG_H2O
    NMAX_LEAK_UNITS
    LEAK_JET_CHECK = 255
End Enum
```

**C#.Net:**

```
public enum F28_LEAK_UNITS : byte
{
      LEAK_PA,
      LEAK_PASEC,
      LEAK_PA_HR,
      LEAK_PASEC_HR,
      LEAK_CAL_PA,
      LEAK_CAL_PASEC,
      LEAK_CCMIN,
      LEAK_CCSEC,
      LEAK_CCH,
      LEAK_MM3SEC,
      LEAK_CM3_SEC,
      LEAK_CM3_MIN,
      LEAK_CM3_H,
      LEAK_ML_SEC,
      LEAK_ML_MIN,
      LEAK_ML_H,
      LEAK_INCH3_SEC,
      LEAK_INCH3_MIN,
      LEAK_INCH3_H,
      LEAK_FT3_SEC,
      LEAK_FT3_MIN,
      LEAK_FT3_H,
      LEAK_MMCE,
      LEAK_MMCE_SEC,
      LEAK_SCCM,
      LEAK_POINTS,
      // 1.500 Leak units
      LEAK_KPA,
      LEAK_MPA,
      LEAK_BAR,
      LEAK_mBAR,
      LEAK_PSI,
      LEAK_L_MIN,
      LEAK_CM_H2O,
      LEAK_UG_H2O,
      NMAX_LEAK_UNITS,
      LEAK_JET_CHECK = 0xff // F28 check jet unit

}
```

### 4.1.4 Volume units

Volume unit, to be use with "**wVolumeUnit**" parameter.

| Definition | Value | Description |
| --- | --- | --- |
| VOLUME_CM3 | 0 | $cm^3$ |
| VOLUME_MM3 | 1 | $mm^3$ |
| VOLUME_ML, | 2 | ml |

| | | |
| --- | --- | --- |
| VOLUME_LITRE | 3 | l |
| VOLUME_INCH3 | 4 | $inch^3$ |
| VOLUME_FT3 | 5 | $Feet^3$ |

**Declaration in C/C++:**

```
enum F28_ENUM_VOLUME_UNIT
{
      VOLUME_CM3,
      VOLUME_MM3,
      VOLUME_ML,
      VOLUME_LITRE,
      VOLUME_INCH3,
      VOLUME_FT3,
      NMAX_VOLUME_UNITS
};
```

**Visual Basic (Vb.Net):**

```
Enum F28_ENUM_VOLUME_UNIT As Byte  'Uses with wVolumeUnit parameter
      VOLUME_CM3
      VOLUME_MM3
      VOLUME_ML
      VOLUME_LITRE
      VOLUME_INCH3
      VOLUME_FT3
      NMAX_VOLUME_UNITS
End Enum
```

**C#.Net:**

```
public enum F28_ENUM_VOLUME_UNIT : byte
{
      VOLUME_CM3,
      VOLUME_MM3,
      VOLUME_ML,
      VOLUME_LITRE,
      VOLUME_INCH3,
      VOLUME_FT3,
      NMAX_VOLUME_UNITS
};
```

### 4.1.5 Fill mode

Fill mode parameter, to be use with "**wFillMode**".

| Definition | Value | Description |
| --- | --- | --- |
| STD_FILL_MODE | 0 | Standard fill mode |
| AUTOFILL_MODE | 1 | Auto fill mode |
| INSTRUCTION_MODE | 2 | Fill mode with instruction, only with the electronic regulator option built-in (from v1.500) |
| RAMP_MODE | 3 | Fill with a ramp, only with the electronic regulator option built-in (from v1.500) |
| RAMP_CONTROL | 4 | |
| EASY | 5 | Basic electronic regulator mode without learning cycle |
| EASY_AUTO | 6 | Easy + auto correction to reach pressure instruction |

**Declaration in C/C++:**

```
enum F28_ENUM_FILL_MODE
{
      STD_FILL_MODE,
      AUTOFILL_MODE,
      INSTRUCTION_MODE,     // Electronic regulator option & from v1.500 only
      RAMP_MODE,            // Electronic regulator option & from v1.500 only
      RAMP_CONTROL,
      EASY,
      EASY_AUTO,
      NMAX_FILL_MODE

};
```

**Visual Basic (Vb.Net):**

```
Enum F28_ENUM_FILL_MODE As Byte
      STD_FILL_MODE
      AUTOFILL_MODE
      INSTRUCTION_MODE      'Electronic regulator option & from v1.500 only
      RAMP_MODE             'Electronic regulator option & from v1.500 only
      RAMP_CONTROL
      EASY
      EASY_AUTO
      NMAX_FILL_MODE

End Enum
```

**C#.Net:**

```
public enum F28_ENUM_FILL_MODE : byte
{
      STD_FILL_MODE,
      AUTOFILL_MODE,
      INSTRUCTION_MODE,     // Electronic regulator option & from v1.500 only
      RAMP_MODE,            // Electronic regulator option & from v1.500 only
      RAMP_CONTROL,
      EASY,
      EASY_AUTO,
      NMAX_FILL_MODE
};
```

### 4.1.6 Boot/Application mode

Returned by function "**F28_GetMode**".

Obsolete, see chapter 2.

### 4.1.7 Group identifier

Obsolete, see chapter 2.

## 4.2 Module address

Obsolete, see chapter 2.

Value for variable "**ucModuleAddr**".

| Variable | Data type | Value | Description |
| --- | --- | --- | --- |
| ucModuleAddr | BYTE | F28_MODULE_ADDR_0 = 0<br>F28_MODULE_ADDR_1 = 1<br>F28_MODULE_ADDR_2 = 2<br>F28_MODULE_ADDR_3 = 3<br>F28_MODULE_ADDR_4 = 4<br>F28_MODULE_ADDR_5 = 5<br>F28_MODULE_ADDR_6 = 6<br>F28_MODULE_ADDR_7 = 7<br>F28_MODULE_ADDR_8 = 8<br>F28_MODULE_ADDR_9 = 9<br>F28_MODULE_ADDR_10 = 10<br>F28_MODULE_ADDR_11 = 11<br>F28_MODULE_ADDR_12 = 12<br>F28_MODULE_ADDR_13 = 13<br>F28_MODULE_ADDR_14 = 14<br>F28_MODULE_ADDR_15 = 15 | Station address (0 -15) |

**Declaration in C/C++:**

```
enum
{
      F28_MODULE_ADDR_0,
      F28_MODULE_ADDR_1,
      F28_MODULE_ADDR_2,
      F28_MODULE_ADDR_3,
      F28_MODULE_ADDR_4,
      F28_MODULE_ADDR_5,
      F28_MODULE_ADDR_6,
      F28_MODULE_ADDR_7,
      F28_MODULE_ADDR_8,
      F28_MODULE_ADDR_9,
```

**Visual Basic (Vb.Net):**

```
Enum F28_MODULE_ADDR As Byte
      MODULE_ADDR_0 = 0
      MODULE_ADDR_1
      MODULE_ADDR_2
      MODULE_ADDR_3
      MODULE_ADDR_4
      MODULE_ADDR_5
      MODULE_ADDR_6
      MODULE_ADDR_7
      MODULE_ADDR_8
      MODULE_ADDR_9
      MODULE_ADDR_10
```

```
      F28_MODULE_ADDR_10,
      F28_MODULE_ADDR_11,
      F28_MODULE_ADDR_12,
      F28_MODULE_ADDR_13,
      F28_MODULE_ADDR_14,
      F28_MODULE_ADDR_15,
      F28_MAX_MODULES_BY_GROUP
 };
```

```
      MODULE_ADDR_11
      MODULE_ADDR_12
      MODULE_ADDR_13
      MODULE_ADDR_14
      MODULE_ADDR_15
      MODULE_MAX
End Enum
```

**C#.Net:**

```
public enum F28_MODULE_ADDR_ENUM : byte
{
      F28_MODULE_ADDR_0,
      F28_MODULE_ADDR_1,
      F28_MODULE_ADDR_2,
      F28_MODULE_ADDR_3,
      F28_MODULE_ADDR_4,
      F28_MODULE_ADDR_5,
      F28_MODULE_ADDR_6,
      F28_MODULE_ADDR_7,
      F28_MODULE_ADDR_8,
      F28_MODULE_ADDR_9,
      F28_MODULE_ADDR_10,
      F28_MODULE_ADDR_11,
      F28_MODULE_ADDR_12,
      F28_MODULE_ADDR_13,
      F28_MODULE_ADDR_14,
      F28_MODULE_ADDR_15,
      F28_MAX_MODULES_BY_GROUP
};
```

## 4.3 Step code

Step code value for variable ucStatus inside F28_REALTIME_CYCLE structure.

| Variable | Data type | Value | Description |
| --- | --- | --- | --- |
| ucStatus | BYTE | READY = 0, | Out of cycle |
| | BYTE | FILL_STEP = 1, | Fill step |
| | BYTE | STAB_STEP = 2, | Stabilization step |
| | BYTE | TEST_STEP = 3, | Test step |
| | BYTE | DUMP_STEP =4 | Dump step |

**Declaration in C/C++:**

```
enum F28_ENUM_STEP_CODE
{
      READY,
      FILL_STEP,
      STAB_STEP,
```

**Visual Basic (Vb.Net):**

```
Enum F28_ENUM_STEP_CODE As Byte
      READY
      FILL_STEP
      STAB_STEP
      TEST_STEP
```

```
      TEST_STEP,
      DUMP_STEP,
      FILL_VOLUME_STEP,
      TRANSFERT_STEP,
 };
```

```
      DUMP_STEP
      FILL_VOLUME_STEP
      TRANSFERT_STEP
End Enum
```

**C#.Net:**

```
public enum F28_ENUM_STEP_CODE : byte
{
      READY,
      FILL_STEP,
      STAB_STEP,
      TEST_STEP,
      DUMP_STEP,
      FILL_VOLUME_STEP,
      TRANSFERT_STEP,
};
```

## 4.4 Identifier of module

Value for variable sModuleID. The identifier of the module is unique. It returns by the function "**F28_AddModule**".

| Variable | Data type | Description |
| --- | --- | --- |
| sModuleID | short | High byte = index of channel.<br>Low byte = index of module. |

## 4.5 Structure definition in C/C++

***Nota**: All structures are 1 byte packed, for easy portability and data exchange between API and Visual basic 2013 application.*

```
#pragma pack(push, 1 )

// Date structure
typedef struct
{
      WORD wYear;
      WORD wMonth;
      WORD wDay;
      WORD wHour;
      WORD wMinute;
      WORD wSecond;
} F28_DATE;

// Result structure
typedef struct
{
```

```
      UCHAR ucStatus;
      float fPressureValue;
      float fLeakValue;
      UCHAR ucUnitPressure;
      UCHAR ucUnitLeak;
      BYTE ucGroupID;
      BYTE ucModuleAddr;
      F28_DATE dateReceived;
}F28_RESULT;

// Real time structure
typedef struct F28_REALTIME_CYCLE
{
      UCHAR ucEndCycle;
      UCHAR ucStatus;
      float fPressureValue;
      float fLeakValue;
      UCHAR ucUnitPressure;
      UCHAR ucUnitLeak;
      float fInternalTemperature;
      float fPatm;
}F28_REALTIME_CYCLE;

// Parameter structure
typedef struct F28_PARAMETERS
{
     WORD    wTypeTest;         // STANDARD LEAK
     WORD    wTpsFillVol;
     WORD    wTpsTransfert;
     WORD    wTpsFill;
     WORD    wTpsStab;
     WORD    wTpsTest;
     WORD    wTpsDump;
     WORD    wPress1Unit;       // See F28_PRESS_UNITS
     float   fPress1Min;
     float   fPress1Max;
     float   fSetFillP1;        //instruction auto-fill mode
     float   fRatioMax;         //LARGE LEAK mode only
     float   fRatioMin;         //LARGE LEAK mode only
     WORD    wFillMode;         //STD_FILL_MODE / AUTOFILL_MODE
     WORD    wLeakUnit;         //See F28_LEAK_UNITS
     WORD    wRejectCalc;       //Pa or Pa/s
     WORD    wVolumeUnit;       //See F28_ENUM_VOLUME_UNIT
     float   fVolume;
     float   fRejectMin;
     float   fRejectMax;
     float    fCoeffAutoFill;
     WORD    wOptions;          //Options parameters
     //V1.200
     float   fPatmSTD;          //Patm standard condition (hPa)
     float   fTempSTD;          //Temperature standard condition (in °C)
     float   fFilterTime;       //in (s)
     //V1.300
     float   fOffsetLeak;       //Offset on the leak
```

```
}F28_PARAMETERS;

// Special cycle Volume learning structure
typedef struct X28_VOLUME_LEARNING_TEST
{
    WORD    wTempsRempVolumeLearn;                // Fill time
    WORD    wTempsTransfertTestVolumeLearn;       // Transfert time
    float   fVolumePressCC;                       // Volume pressure (cm3)
    float   fVolumeMin;                           // Minimum volume
    float   fVolumeMax;                           // Maximum volume
} X28_VOLUME_LEARNING_TEST;

#pragma pack(pop)
```

## 4.6 Structure definition in Visual Basic

```
' -------------------------
' Date structure
' -------------------------
<StructLayout(LayoutKind.Sequential, Pack:=1)> _
Structure F28_DATE
     Dim wYear As UShort
     Dim wMonth As UShort
     Dim wDay As UShort
     Dim wHour As UShort
     Dim wMinute As UShort
     Dim wSecond As UShort
End Structure

' -------------------------
' Result structure
' -------------------------
<StructLayout(LayoutKind.Sequential, Pack:=1)> _
Structure F28_RESULT
     Dim ucStatus As Byte
     Dim fPressureValue As Single
     Dim fLeakValue As Single
     Dim ucUnitPressure As Byte
     Dim ucUnitLeak As Byte
     Dim GroupID As Byte                  ' F28_GROUP_ID
     Dim ModuleAddr As Byte               ' F28_MODULE_ADDR
     Dim dateReceived As F28_DATE
End Structure

' -------------------------
' real time result structure
' -------------------------
<StructLayout(LayoutKind.Sequential, Pack:=1)> _
Structure F28_REALTIME_CYCLE
     Dim ucEndCycle As Byte
     Dim ucStatus As Byte
     Dim fPressureValue As Single
```

```
     Dim fLeakValue As Single
     Dim ucUnitPressure As Byte
     Dim ucUnitLeak As Byte
     Dim fInternalTemperature As Single    ' Temperature in °C
     Dim fPatm As Single                   ' Abs pressure in hPa
End Structure

' -------------------------
' Statistic structure
' -------------------------
<StructLayout(LayoutKind.Sequential, Pack:=1)> _
Structure F28_CYCLE_STATISTICS
     Dim dwTotalCycles As UInteger
     Dim dwFailCycles As UInteger
     Dim dwSuccessCycles As UInteger
End Structure

' -------------------------
' Communication counter structure
' -------------------------
<StructLayout(LayoutKind.Sequential, Pack:=1)> _
Structure F28_COMMUNICATION_STATISTICS
     Dim dwTransmited As UInteger
     Dim dwReceived As UInteger
     Dim dwErrors As UInteger
End Structure

' -------------------------
' Parameter structure
' -------------------------
<StructLayout(LayoutKind.Sequential, Pack:=1)> _
Structure F28_PARAMETERS
     Dim wTypeTest As UShort ' STANDARD LEAK
     Dim wTpsFillVol As UShort
     Dim wTpsTransfert As UShort
     Dim wTpsFill As UShort
     Dim wTpsStab As UShort
     Dim wTpsTest As UShort
     Dim wTpsDump As UShort
     Dim wPress1Unit As UShort   ' See F28_PRESS_UNITS
     Dim fPress1Min As Single
     Dim fPress1Max As Single
     Dim fSetFillP1 As Single' Setpoint auto-fill
     Dim fRatioMax As Single
     Dim fRatioMin As Single
     Dim wFillMode As UShort ' STD_FILL_MODE / AUTOFILL_MODE
     Dim wLeakUnit As UShort ' See F28_LEAK_UNITS
     Dim wRejectCalc As UShort   ' Pa or Pa/s
     Dim wVolumeUnit As UShort   ' See F28_ENUM_VOLUME_UNIT
     Dim fVolume As Single
     Dim fRejectMin As Single
     Dim fRejectMax As Single
     Dim fCoeffAutoFill As Single
     Dim wOptions As UShort  ' Options parameters
```

```
     Dim fPatmSTD As Single  ' Patm  standard condition (hPa)
     Dim fTempSTD As Single  ' Temperature standard condition (°C)
     Dim fFilterTime As Single   ' in (s)
     Dim fOffsetLeak As Single' Offset on the leak
End Structure

' ----------------------------------------
' Special cycle Volume learning structure
' ----------------------------------------
<StructLayout(LayoutKind.Sequential, Pack:=1)> _
Structure X28_VOLUME_LEARNING_TEST
    Dim wTempsRempVolumeLearn As Ushort                ' Fill time
    Dim wTempsTransfertTestVolumeLearn As Ushort;      ' Transfert time
    Dim fVolumePressCC As Single;                      ' Volume pressure (cm3)
    Dim fVolumeMin;                                    ' Minimum volume
    Dim fVolumeMax;                                    ' Maximum volume
End Structure
```

## 4.7 Structure definition in C#.NET

```
[StructLayout(LayoutKind.Sequential, Pack = 1, CharSet = CharSet.Ansi)]
public struct F28_DATE
{
     public ushort usYear;
     public ushort usMonth;
     public ushort usDay;
     public ushort usHour;
     public ushort usMinute;
     public ushort usSecond;
};

[StructLayout(LayoutKind.Sequential, Pack = 1, CharSet = CharSet.Ansi)]
public struct F28_RESULT
{
     public byte     bStatus;
     public float    fPressureValue;
     public float    fLeakValue;
     public byte     bUnitPressure;
     public byte     bUnitLeak;
     public byte     bGroupID;
     public byte     bModuleAddr;
     public F28_DATE dateReceived;
};

[StructLayout(LayoutKind.Sequential, Pack = 1, CharSet = CharSet.Ansi)]
public struct F28_REALTIME_CYCLE
{
     public byte     bEndCycle;
     public byte     bStatus;
     public float    fPressureValue;
     public float    fLeakValue;
     public byte     bUnitPressure;
```

```
      public byte     bUnitLeak;
      public float    fInternalTemperature;
      public float    fPatm;
};

[StructLayout(LayoutKind.Sequential, Pack = 1, CharSet = CharSet.Ansi)]
public struct F28_CYCLE_STATISTICS
{
      public uint uiTotalCycles;
      public uint uiFailCycles;
      public uint uiSuccessCycles;
};

[StructLayout(LayoutKind.Sequential, Pack = 1, CharSet = CharSet.Ansi)]
public struct F28_REGLAGE
{
      long        lOffset;
      float       fCoeffA;
      float       fCoeffB;
      F28_DATE    date;
      [MarshalAs(UnmanagedType.LPStr)] string Operator;
};

[StructLayout(LayoutKind.Sequential, Pack = 1, CharSet = CharSet.Ansi)]
public struct F28_COMMUNICATION_STATISTICS
{
      public uint uiTransmited;
      public uint uiReceived;
      public uint uiErrors;
};

[StructLayout(LayoutKind.Sequential, Pack = 1, CharSet = CharSet.Ansi)]
public struct F28_PARAMETERS
{
      public ushort usTypeTest;    // STANDARD LEAK
      public ushort usTpsFillVol;
      public ushort usTpsTransfert;
      public ushort usTpsFill;
      public ushort usTpsStab;
      public ushort usTpsTest;
      public ushort usTpsDump;
      public ushort usPress1Unit;  // See F28_PRESS_UNITS
      public float fPress1Min;
      public float fPress1Max;
      public float fSetFillP1;     //auto-fill mode instruction
      public float fRatioMax;      //LARGE LEAK mode only
      public float fRatioMin;      //LARGE LEAK mode only
      public ushort usFillMode;    //STD_FILL_MODE / AUTOFILL_MODE
      public ushort usLeakUnit;    //See F28_LEAK_UNITS
      public ushort usRejectCalc;  //Pa or Pa/s
      public ushort usVolumeUnit;  //See F28_ENUM_VOLUME_UNIT
      public float fVolume;
      public float fRejectMin;
      public float fRejectMax;
```

```
     public float fCoeffAutoFill;
     public ushort usOptions;     //Options parameters
     public float fPatmSTD;       //Patm standard condition (hPa)
     public float fTempSTD;       //T° standard condition (in °C)
     public float fFilterTime;    //in (s)
     public float fOffsetLeak;    //Offset on the leak
};

// Special cycle Volume learning structure
[StructLayout(LayoutKind.Sequential, Pack = 1, CharSet = CharSet.Ansi)]
public struct X28_VOLUME_LEARNING_TEST
{
    public ushort wTempsRempVolumeLearn;              // Fill time
    public ushort wTempsTransfertTestVolumeLearn;     // Transfert time
    public float  fVolumePressCC;                     // Volume pressure (cm3)
    public float  fVolumeMin;                         // Minimum volume
    public float  fVolumeMax;                         // Maximum volume
};
```

## 4.8 Function return code

Obsolete, use X28_FAIL and X28_OK (see chapter 2.6).

| Declaration | Data type | Value | Description |
| --- | --- | --- | --- |
| F28_FAIL | short | -1 | Error |
| F28_OK | | 0 | Ok |

**Declaration in C/C++:**

```
enum  F28_RETURN
{
     F28_FAIL = -1,
     F28_OK
};
```

**Visual Basic (Vb.Net):**

```
Enum F28_RETURN As Short
     F28_FAIL = -1
     F28_OK = 0
End Enum
```

**C#.Net:**

```
public enum F28_RETURN : byte
{
     F28_FAIL = -1,
     F28_OK
};
```

# 5 Application programming interface

## 5.1 Functional groups in the API

- Driver related functions
- Network related functions
- General device functions
- Information related functions
- Unit Control related functions
- Group Control related functions
- Parameters related functions
- Result related functions

## 5.2 Driver related functions

These functions concern all devices (B28/F28).

### 5.2.1 X28_Init

This function detects a head board and initializes a connection. It must be called first.

**Function call:**

**C++:**

```
short X28API X28_Init(void);
```

**Visual Basic (Vb.Net):**

```
Public Declare Function X28_Init Lib "F28LightControl_ETH.dll" () As Short
```

**C#.Net:**

```
[DllImport("F28LightControl_ETH.dll")]
private static extern short X28_Init();
```

**Arguments:**

| Argument | Data type | Description |
| --- | --- | --- |
| none | | |

**Return Value:**

**X28_OK**: if the function succeeds.

**X28_FAIL**: if the function fails.

***Nota :*** *Old version F28_Init() function is obsolete but can be used for compatibility reasons.*

### 5.2.2 X28_OpenChannel

This function opens a channel.

**Function call:**

**C++:**

```
short X28API X28_OpenChannel(void);
```

**Visual Basic (Vb.Net):**

```
Public Declare Function X28_OpenChannel Lib "F28LightControl_ETH.dll" () _ As Short
```

**C#.Net:**

```
[DllImport("F28LightControl_ETH.dll")]
private static extern short X28_OpenChannel();
```

**Arguments:**

| Argument | Data type | Description |
| --- | --- | --- |
| none | | |

**Return Value:**

**X28_OK**: if the function succeeds.

**X28_FAIL**: if the function fails.

***Nota :*** *Old version F28_OpenChannel() function is obsolete but can be used for compatibility reasons.*

### 5.2.3 X28_Close

This function closes all channels.

**Function call:**

**C++:**

```
void X28API X28_Close(void);
```

**Visual Basic (Vb.Net):**

```
Public Declare Sub X28_Close Lib "F28LightControl_ETH.dll" ()
```

**C#.Net:**

```
[DllImport("F28LightControl_ETH.dll")]
private static extern void X28_Close();
```

**Arguments:**

| Argument | Data type | Description |
| --- | --- | --- |
| none | | |

**Return Value:** None

***Note :*** *Old version F28_Close() function is obsolete but can be used for compatibility reasons.*

### 5.2.4 X28_GetDllMajorVersion

Read a major's version of the API.

**Function call:**

**C++:**

```
unsigned short X28API X28_GetDllMajorVersion(void);
```

**Visual Basic (Vb.Net):**

```
Public Declare Function X28_GetDllMajorVersion Lib _ "F28LightControl_ETH.dll" () As
             Short
```

**C#.Net:**

```
[DllImport("F28LightControl_ETH.dll")]
private static extern ushort X28_GetDllMajorVersion();
```

**Arguments:**

| Argument | Data type | Description |
| --- | --- | --- |
| None | | |

**Return Value:** *unsigned short*

*Major version*

***Note :*** *Old version F28_GetDllMajorVersion() function is obsolete but can be used for compatibility reasons.*

### 5.2.5 X28_GetDllMinorVersion

Read a minor's version of the API.

**Function call:**

**C++:**

```
unsigned short X28_GetDllMinorVersion()
```

**Visual Basic (Vb.Net):**

```
Public Declare Function X28_GetDllMinorVersion Lib _ "F28LightControl_ETH.dll" () As
             Short
```

**C#.Net:**

```
[DllImport("F28LightControl_ETH.dll")]
```

```
private static extern ushort X28_GetDllMinorVersion();
```

**Arguments:**

| Argument | Data type | Description |
| --- | --- | --- |
| none | | |

**Return Value:** *unsigned short*

*Minor version*

***Note :*** *Old version F28_GetDllMinorVersion() function is obsolete but can be used for compatibility reasons.*

## 5.3 Network related functions

### 5.3.1 F28 Functions

#### *5.3.1.1 F28_AddModule*

Add F28 unit to the network.

**Function call:**

**C++:**

```
short F28API  F28_AddModule(ULONG ulIP, BYTE ucModuleAddr, BYTE ucGroupID, BYTE
              ucTimeout);
```

**Visual Basic (Vb.Net):**

```
Public Declare Function F28_AddModule Lib "F28LightControl_ETH.dll" (ByVal ulIP As
              UInteger, ByVal ucModuleAddr As Byte, ByVal ucGroupID As Byte,
              ucTimeout As Byte) As Short
```

**C#.Net:**

```
[DllImport("F28LightControl_ETH.dll")]
private static extern short F28_AddModule(uint ulIP, byte bModuleAddr, byte bGroupID,
              byte bTimeout);
```

**Arguments:**

| Argument | Data type | Description |
| --- | --- | --- |
| ulIP | ULONG | IP address in long format |
| ucModuleAddr | BYTE | Module address |
| ucGroupID | BYTE | Group ID |
| ucTimeout | BYTE | Timeout in seconds |

**Return Value:** short.

**X28_FAIL**: if the function fails.

**sModuleID**: High byte = channel's index, Low byte = module's index

## 5.3.2 B28 Functions

### *5.3.2.1 B28_AddModule*

Add B28 unit to the network.

**Function call:**

**C++:**

```
short F28API  B28_AddModule(ULONG ulIP, BYTE ucModuleAddr, BYTE ucGroupID, BYTE
              ucTimeout);
```

**Visual Basic (Vb.Net):**

```
Public Declare Function B28_AddModule Lib "F28LightControl_ETH.dll" (ByVal ulIP As
              UInteger, ByVal ucModuleAddr As Byte, ByVal ucGroupID As Byte,
              ucTimeout As Byte) As Short
```

**C#.Net:**

```
[DllImport("F28LightControl_ETH.dll")]
private static extern short B28_AddModule(uint ulIP, byte bModuleAddr, byte bGroupID,
              byte bTimeout);
```

**Arguments:**

| Argument | Data type | Description |
| --- | --- | --- |
| ulIP | ULONG | IP address in long format |
| ucModuleAddr | BYTE | Module address |
| ucGroupID | BYTE | Group ID |
| ucTimeout | BYTE | Timeout in seconds |

**Return Value:** short.

**X28_FAIL**: if the function fails.

**sModuleID**: High byte = channel's index, Low byte = module's index

## 5.3.3 X28 Functions

These functions concern all devices (B28/F28).

### *5.3.3.1 X28_ReconnectModule*

Reconnect unit specified by module ID in the network.

**Function call:**

**C++:**

```
short X28API X28_ReconnectModule(short sModuleID);
```

**Visual Basic (Vb.Net):**

```
Public Declare Function X28_ReconnectModule Lib "F28LightControl_ETH.dll" _ (ByVal
              sModuleID As Short) As Short
```

**C#.Net:**

```
[DllImport("F28LightControl_ETH.dll")]
private static extern short X28_ReconnectModule(short sModuleID);
```

**Arguments:**

| Argument | Data type | Description |
| --- | --- | --- |
| sModuleID | short | Identifier of module |

**Return Value:** short

**X28_OK**: if the function succeeds.

**X28_FAIL**: if the function fails.

#### *5.3.3.2 X28_RemoveModule*

Remove unit specified by module ID from network.

**Function call:**

**C++:**

```
short X28API X28_RemoveModule(short sModuleID);
```

**Visual Basic (Vb.Net):**

```
Public Declare Function X28_RemoveModule Lib "F28LightControl_ETH.dll" _ (ByVal
             sModuleID As Short) As Short
```

**C#.Net:**

```
[DllImport("F28LightControl_ETH.dll")]
private static extern short X28_RemoveModule(short sModuleID);
```

**Arguments:**

| Argument | Data type | Description |
| --- | --- | --- |
| sModuleID | short | Identifier of module |

**Return Value:** short

**X28_OK**: if the function succeeds.

**X28_FAIL**: if the function fails.

#### *5.3.3.3 X28_RemoveAllModules*

Remove all units specified by channel's ID from the network.

**Function call:**

**C++:**

```
short F28API X28_RemoveAllModules(BYTE ucChannelID);
```

**Visual Basic (Vb.Net):**

```
Public Declare Function X28_RemoveAllModules Lib "F28LightControl_ETH.dll" _ (ByVal
             ucChannelID As Byte) As Short
```

**C#.Net:**

```
[DllImport("F28LightControl_ETH.dll")]
private static extern short X28_RemoveAllModules();
```

**Arguments:**

| Argument | Data type | Description |
| --- | --- | --- |
| ucChannelID | BYTE | Channel identifier |

**Return Value:** short

**X28_OK**: if the function succeeds.

**X28_FAIL**: if the function fails.

#### *5.3.3.4 X28_ResetEthernetModule*

To reset the Ethernet board, this is to fix a communication issue.

**Function call:**

**C++:**

```
short F28API X28_ResetEthernetModule(short sModuleID); (
```

**Visual Basic (Vb.Net):**

```
Public Declare Function X28_ResetEthernetModule Lib _ "F28LightControl_ETH.dll" _
             (ByVal sModuleID As Short) As Short
```

**C#.Net:**

```
[DllImport("F28LightControl_ETH.dll")]
private static extern short X28_ResetEthernetModule(short sModuleID);
```

**Arguments:**

| Argument | Data type | Description |
| --- | --- | --- |
| sModuleID | short | Identifier of module |

**Return Value:** short

**X28_OK**: if the function succeeds.

**X28_FAIL**: if the function fails.

#### *5.3.3.5 X28_IsModuleConnected*

To test if a unit is connected.

**Function call:**

**C++:**

```
short F28API X28_IsModuleConnected(short sModuleID); (
```

**Visual Basic (Vb.Net):**

```
Public Declare Function X28_IsModuleConnected Lib _ "F28LightControl_ETH.dll" _
              (ByVal sModuleID As Short) As Short
```

**C#.Net:**

```
[DllImport("F28LightControl_ETH.dll")]
private static extern short X28_IsModuleConnected(short sModuleID);
```

**Arguments:**

| Argument | Data type | Description |
| --- | --- | --- |
| sModuleID | short | Identifier of module |

**Return Value:** short

**X28_OFFLINE**: if unit is not connected.

**X28_CONNECTED**: if unit is connected.

**X28_FAIL**: if the function fails.

## 5.4 Information related functions

These functions concern all device (B28/F28).

### 5.4.1 X28_RefreshModuleInformations

Query information about the module.

**Function call:**

**C++:**

```
short F28API X28_RefreshModuleInformations(short sModuleID);
```

**Visual Basic (Vb.Net):**

```
Public Declare Function X28_RefreshModuleInformations Lib _ "F28LightControl_ETH.dll"
              (ByVal sModuleID As Short) As Short
```

**C#.Net:**

```
[DllImport("F28LightControl_ETH.dll")]
private static extern short X28_RefreshModuleInformations(short sModuleID);
```

**Arguments:**

| Argument | Data type | Description |
| --- | --- | --- |
| sModuleID | short | Identifier of module |

**Return Value:** short

**X28_OK**: if the function succeeds.

**X28_FAIL**: if the function fails.

***Nota**: this function must be called before X28_GetSerialNumber, X28_GetModuleSoftVersion, X28_GetModuleHardVersion.*

### 5.4.2 X28_GetSerialNumber

Retrieve the current serial number from Module Information.

**Function call:**

**C++:**

```
short X28API X28_GetSerialNumber(short sModuleID, LPSTR szSerialNumber, unsigned
             short wLength);
```

**Visual Basic (Vb.Net):**

```
Public Declare Function X28_GetSerialNumber Lib "F28LightControl_ETH.dll" _ (ByVal
             sModuleID As Short, ByVal szSerialNumber As String, _ ByVal Length As
             UShort) As Short
```

**C#.Net:**

```
[DllImport("F28LightControl_ETH.dll")]
private static extern short X28_GetSerialNumber(short sModuleID, StringBuilder
             strSerialNumber, ushort usLength);
```

**Arguments:**

| Argument | Data type | Description |
| --- | --- | --- |
| sModuleID | short | Identifier of module |
| szSerialNumber | array of char | Returned serial number |
| wLength | unsigned short | Length of char to read (20 chars max) |

**Return Value:** short

**X28_OK**: if the function succeeds.

**X28_FAIL**: if the function fails.

### 5.4.3 X28_GetModuleSoftVersion

Retrieve the version of unit's firmware from Module Information.

**Function call:**

**C++:**

```
short F28API X28_GetModuleSoftVersion(short sModuleID, LPSTR szVersion, unsigned
             short wLength);
```

**Visual Basic (Vb.Net):**

```
Public Declare Function X28_GetModuleSoftVersion Lib_ "F28LightControl_ETH.dll"
             (ByVal sModuleID As Short, ByVal _ szSoftVersion As String, ByVal
             Length As UShort) As Short
```

**C#.Net:**

```
[DllImport("F28LightControl_ETH.dll", CharSet = CharSet.Ansi)]
public static extern short X28_GetModuleSoftVersion(short sModuleID, StringBuilder
              strVersion, ushort usLength);
```

**Arguments:**

| Argument | Data type | Description |
| --- | --- | --- |
| sModuleID | short | Identifier of module |
| szVersion | array of char | Returned software's version of the B28/F28 |
| wLength | unsigned short | Length of char to read |

**Return Value:** short

**X28_OK**: if the function succeeds.

**X28_FAIL**: if the function fails.

### 5.4.4 X28_GetModuleHardVersion

Retrieve the board's hardware version from Module Information.

**Function call:**

**C++:**

```
short X28API X28_GetModuleHardVersion(short sModuleID, LPSTR szVersion, unsigned
              short wLength);
```

**Visual Basic (Vb.Net):**

```
Public Declare Function X28_GetModuleHardVersion Lib _ "F28LightControl_ETH.dll"
              (ByVal sModuleID As Short, ByVal _ szHardVersion As String, ByVal
              Length As UShort) As Short
```

**C#.Net:**

```
[DllImport("F28LightControl_ETH.dll", CharSet = CharSet.Ansi)]
private static extern short X28_GetModuleHardVersion(short sModuleID, StringBuilder
              strVersion, ushort usLength);
```

**Arguments:**

| Argument | Data type | Description |
| --- | --- | --- |
| sModuleID | short | Identifier of module |
| szVersion | array of char | Returned hardware's version of the B28/F28 |
| wLength | unsigned short | Lenght of byte to read |

**Return Value:** short

**X28_OK**: if the function succeeds.

**X28_FAIL**: if the function fails.

### 5.4.5 X28_GetAddressIP

Read the IP address of the Module in long format.

**Function call:**

**C++:**

```
short X28API X28_GetAddressIP(short sModuleID, ULONG* pAddressIP);
```

**Visual Basic (Vb.Net):**

```
Public Declare Function X28_GetAddressIP Lib "F28LightControl_ETH.dll" (ByVal
              sModuleID As Short, ByRef pAddressIP As UInteger) As Short
```

**C#.Net:**

```
[DllImport("F28LightControl_ETH.dll")]
private static extern short X28_GetAddressIP(short sModuleID, ref uint ulAddressIP);
```

**Arguments:**

| Argument | Data type | Description |
| --- | --- | --- |
| sModuleID | short | ID of module |
| pAddressIP | ULONG | Returned IP address of the module |

**Return Value:** short

**X28_OK**: if the function succeeds.

**X28_FAIL**: if the function fails.

### 5.4.6 X28_ETHSoftVersion

Read the version of Ethernet board firmware of the Module.

**Function call:**

**C++:**

```
short X28API X28_GetETHSoftVersion(short sModuleID, LPSTR szVersion, unsigned short
              wLength);
```

**Visual Basic (Vb.Net):**

```
Public Declare Function X28_GetETHSoftVersion Lib _
              "F28LightControl_ETH.dll" (ByVal sModuleID As Short, _
              ByVal szVersion As String, ByVal wLength As Short) As Short
```

**C#.Net:**

```
[DllImport("F28LightControl_ETH.dll")]
private static extern short X28_GetETHSoftVersion(short sModuleID, StringBuilder
              strVersion, ushort usLength);
```

**Arguments:**

| Argument | Data type | Description |
| --- | --- | --- |

| | | |
| --- | --- | --- |
| sModuleID | short | Identifier of module |
| szVersion | array of char | Returned software's version of the Ethernet board |
| wLength | unsigned short | Length of char to read |

**Return Value:** short

**X28_OK**: if the function succeeds.

**X28_FAIL**: if the function fails.

### 5.4.7 X28_GetETHHardVersion

Read the hard version of Ethernet board of the Module.

**Function call:**

**C++:**

```
short X28API X28_GetETHHardVersion(short sModuleID, LPSTR szVersion, unsigned short
             wLength);
```

**Visual Basic (Vb.Net):**

```
Public Declare Function X28_GetETHHardVersion Lib _ "F28LightControl_ETH.dll" (ByVal
             sModuleID As Short, _
             ByVal szVersion As String, ByVal wLength As Short) As Short
```

**C#.Net:**

```
[DllImport("F28LightControl_ETH.dll")]
private static extern short X28_GetETHHardVersion(short sModuleID, StringBuilder
             strVersion, ushort usLength);
```

**Arguments:**

| Argument | Data type | Description |
| --- | --- | --- |
| sModuleID | short | Identifier of module |
| szVersion | array of char | Returned hardware's version of the Ethernet board |
| wLength | unsigned short | Length of char to read |

**Return Value:** short

**X28_OK**: if the function succeeds.

**X28_FAIL**: if the function fails.

### 5.4.8 X28_GetSubnetMask

Read the Subnet mask of the Module in long format.

**Function call:**

**C++:**

```
short X28API X28_GetSubnetMask(short sModuleID, ULONG* pAddressIP);
```

**Visual Basic (Vb.Net):**

```
Public Declare Function X28_GetSubnetMask Lib "F28LightControl_ETH.dll" (ByVal
              sModuleID As Short, ByRef pAddressIP As UInteger) As Short
```

**C#.Net:**

```
[DllImport("F28LightControl_ETH.dll")]
private static extern short X28_GetSubnetMask(short sModuleID, ref uint ulAddressIP);
```

**Arguments:**

| Argument | Data type | Description |
| --- | --- | --- |
| sModuleID | short | ID of module |
| pAddressIP | ULONG | Returned Subnet mask of the module |

**Return Value:** short

**X28_OK**: if the function succeeds.

**X28_FAIL**: if the function fails.

### 5.4.9 X28_GetGatewayAddressIP

Read the Gateway of the Module in long format.

**Function call:**

**C++:**

```
short X28API X28_GetGatewayAddressIP(short sModuleID, ULONG* pAddressIP);
```

**Visual Basic (Vb.Net):**

```
Public Declare Function X28_GetGatewayAddressIP Lib "F28LightControl_ETH.dll" (ByVal
              sModuleID As Short, ByRef pAddressIP As UInteger) As Short
```

**C#.Net:**

```
[DllImport("F28LightControl_ETH.dll")]
private static extern short X28_GetGatewayAddressIP(short sModuleID, ref uint
              ulAddressIP);
```

**Arguments:**

| Argument | Data type | Description |
| --- | --- | --- |
| sModuleID | short | ID of module |
| pAddressIP | ULONG | Returned Gateway address of the module in long format |

**Return Value:** short

**X28_OK**: if the function succeeds.

**X28_FAIL**: if the function fails.

### 5.4.10 X28_GetMACAddress

Read the MAC address of the Module Information.

**Function call:**

**C++:**

```
short X28API X28_GetMACAddress(short sModuleID, LPSTR szMAC, unsigned short wLength);
```

**Visual Basic (Vb.Net):**

```
Public Declare Function X28_GetMACAddress Lib "F28LightControl_ETH.dll" _ (ByVal
              sModuleID As Short, ByVal szMAC As String, ByVal _ wLength As Short) As
              Short
```

**C#.Net:**

```
[DllImport("F28LightControl_ETH.dll")]
private static extern short X28_GetMACAddress(short sModuleID, StringBuilder strMAC,
              ushort usLength);
```

**Arguments:**

| Argument | Data type | Description |
| --- | --- | --- |
| sModuleID | short | ID of module |
| szMAC | LPSTR | Returned MAC address of the module in string |

**Return Value:** short

**X28_OK**: if the function succeeds.

**X28_FAIL**: if the function fails.

## 5.5 Unit control related functions

These functions concern all devices (B28/F28)

### 5.5.1 X28_IsModuleConnected

Check if the module is connected.

**Function call:**

**C++:**

```
short X28API X28_IsModuleConnected(short sModuleID);
```

**Visual Basic (Vb.Net):**

```
Public Declare Function X28_IsModuleConnected Lib _ "F28LightControl_ETH.dll" (ByVal
              sModuleID As Short) As Short
```

**C#.Net:**

```
[DllImport("F28LightControl_ETH.dll")]
private static extern short X28_IsModuleConnected(short sModuleID);
```

**Arguments:**

| Argument | Data type | Description |
| --- | --- | --- |
| sModuleID | short | Identifier of module |

**Return Value:** short

**X28_CONNECTED**: if the module is connected.

**X28_OFFLINE**: if the module is not connected.

**X28_FAIL**: if the function fails.

### 5.5.2 X28_StartCycle

The function starts the test cycle of the module.

**Function call:**

**C++:**

```
short X28API X28_StartCycle(short sModuleID);
```

**Visual Basic (Vb.Net):**

```
Public Declare Function X28_StartCycle Lib "F28LightControl_ETH.dll" _
              (ByVal sModuleID As Short) As Short
```

**C#.Net:**

```
[DllImport("F28LightControl_ETH.dll")]
private static extern short X28_StartCycle(short sModuleID);
```

**Arguments:**

| Argument | Data type | Description |
| --- | --- | --- |
| sModuleID | short | Identifier of module |

**Return Value:** short

**X28_OK**: if the function succeeds.

**X28_FAIL**: if the function fails.

### 5.5.3 X28_StopCycle

The function aborts the test cycle of the module.

**Function call:**

**C++:**

```
short X28API X28_StopCycle(short sModuleID);
```

**Visual Basic (Vb.Net):**

```
Public Declare Function X28_StopCycle Lib "F28LightControl_ETH.dll" _
              (ByVal sModuleID As Short) As Short
```

**C#.Net:**

```
[DllImport("F28LightControl_ETH.dll")]
private static extern short X28_StopCycle(short sModuleID);
```

**Arguments:**

| Argument | Data type | Description |
| --- | --- | --- |
| sModuleID | short | Identifier of module |

**Return Value:** short

**X28_OK**: if the function succeeds.

**X28_FAIL**: if the function fails.

## 5.6 Group control related functions

These functions concern all devices (B28/F28)

### 5.6.1 X28_StartCycleByGroup

The function starts the test cycle of all units cycle in the defined group.

**Function call:**

**C++:**

```
short X28API X28_StartCycleByGroup(BYTE ucGroupID);
```

**Visual Basic (Vb.Net):**

```
Public Declare Function X28_StartCycleByGroup Lib _ "F28LightControl_ETH.dll" (ByVal
              ucGroupID As Byte) As Short
```

**C#.Net:**

```
[DllImport("F28LightControl_ETH.dll")]
private static extern short X28_StartCycleByGroup(byte bGroupID);
```

**Arguments:**

| Argument | Data type | Description |
| --- | --- | --- |
| ucGroupID | BYTE | Identifier of the group |

**Return Value:** short

**X28_OK**: if the function succeeds.

**X28_FAIL**: if the function fails.

### 5.6.2 X28_StopCycleByGroup

The function aborts the test cycle of all units cycle in the defined group.

**Function call:**

**C++:**

```
short X28API X28_StopCycleByGroup(BYTE ucGroupID);
```

**Visual Basic (Vb.Net):**

```
Public Declare Function X28_StopCycleByGroup Lib "F28LightControl_ETH.dll" _ (ByVal
             ucGroupID As Byte) As Short
```

**C#.Net:**

```
[DllImport("F28LightControl_ETH.dll")]
private static extern short X28_StopCycleByGroup(byte bGroupID);
```

**Arguments:**

| Argument | Data type | Description |
| --- | --- | --- |
| ucGroupID | BYTE | Identifier of the group |

**Return Value:** short

**X28_OK**: if the function succeeds.

**X28_FAIL**: if the function fails.

## 5.7 Parameters related functions

### 5.7.1 F28 parameters

#### *5.7.1.1 Parameters structure F28_PARAMETERS*

| Element | Data type | Description |
| --- | --- | --- |
| wTypeTest | WORD | Test type parameter |
| wTpsFillVol | WORD | Fill time for volume transfer in 0.01 sec (0 – 650 sec) |
| wTpsTransfert | WORD | Transfer time in 0.01 sec (0 – 650 sec) |
| wTpsFill | WORD | Fill time in 0.01 sec (0 – 650 sec) |
| wTpsStab | WORD | Stabilization time in 0.01 sec (0 – 650 sec) |
| wTpsTest | WORD | Test time in 0.01 sec (0 – 650 sec) |
| wTpsDump | WORD | Dump time in 0.01 sec (0 – 650 sec) |
| wPress1Unit | WORD | Unit of pressure # 1 |
| fPress1Min | float | Minimum pressure # 1 |
| fPress1Max | float | Maximum pressure # 1 |
| fSetFillP1 | float | Setpoint pressure # 1 |
| fRatioMax | float | Max reject value for ratio $P_{start}/P_{end}$ |
| fRatioMin | float | Min reject value for ratio $P_{start}/P_{end}$ |
| wFillMode | WORD | Fill mode |

| Element | Data type | Description | |
| --- | --- | --- | --- |
| wLeakUnit | WORD | Leak unit | |
| wRejectCalc | WORD | Pa or Pa/s | |
| wVolumeUnit | WORD | Volume unit | |
| fVolume | float | Volume value | |
| fRejectMin | float | Reject **Reference** side | *See reminder at the beginning of this manual.* |
| fRejectMax | float | Reject **Test** side | |
| fCoeffAutoFill | float | Reserved | |
| wOptions | WORD | See paragraph 3.7.1) Options ("**wOptions**" parameter). | |
| FPatmSTD | float | Patm standard condition (hPa) | |
| FTempSTD | float | Temperature standard condition (°C) | |
| FFilterTime | float | Filter time in sec | |
| fOffsetLeak | float | Offset on the leak | |
| fVolumeRef | float | Reference volume | |
| wTpsTestCompTemp | WORD | Test time for temperature compensation | |
| wPourcCompTemp | WORD | Percentage for temperature compensation | |
| wTpsWaitingTime | WORD | Waiting time for temperature compensation | |
| wLastConsigneDacEasy | WORD | DAC target for easy auto fill mode (read only) | |
| fNominalValue | float | Nominal value for auto-ratio | |
| fCoeffMax | float | Coeff max for auto-ratio (read only) | |
| fCoeffMin | float | Coeff min for auto-ratio (read only) | |

**Declaration in C/C++:**

```
typedef struct
{
    WORD    wTypeTest;          //STANDARD LEAK
    WORD    wTpsFillVol;
    WORD    wTpsTransfert;
    WORD    wTpsFill;
    WORD    wTpsStab;
    WORD    wTpsTest;
    WORD    wTpsDump;
    WORD    wPress1Unit;        //See F28_PRESS_UNITS
    float   fPress1Min;
    float   fPress1Max;
    float   fSetFillP1;         //instruction auto-fill mode
    float   fRatioMax;          //LARGE LEAK mode only
    float   fRatioMin;          //LARGE LEAK mode only
    WORD    wFillMode;          //STD_FILL_MODE / AUTOFILL_MODE
    WORD    wLeakUnit;          //See F28_LEAK_UNITS
    WORD    wRejectCalc;        //Pa or Pa/s
    WORD    wVolumeUnit;        //See F28_ENUM_VOLUME_UNIT
    float   fVolume;
    float   fRejectMin;
    float   fRejectMax;
    float   fCoeffAutoFill;
    WORD    wOptions;           //Options parameters
    float   fPatmSTD;           //Patm standard condition (hPa)
    float   fTempSTD;           //Temperature standard condition (°C)
    float   fFilterTime;        //in (s)
    float   fOffsetLeak;        //Offset on the leak
    float   fVolumeRef;
    WORD    wTpsTestCompTemp;
    WORD    wPourcCompTemp;
    WORD    wTpsWaitingTime;
    WORD    wLastConsigneDacEasy;
    float   fNominalValue;
    float   fCoeffMin;
    float   fCoeffMax;
}F28_PARAMETERS;
```

**Declaration in Visual Basic 2013:**

```
<StructLayout(LayoutKind.Sequential, Pack:=1)> _
Structure F28_PARAMETERS
      Dim wTypeTest As UShort       'STANDARD LEAK
      Dim wTpsFillVol As UShort
      Dim wTpsTransfert As UShort
      Dim wTpsFill As UShort
      Dim wTpsStab As UShort
      Dim wTpsTest As UShort
      Dim wTpsDump As UShort
      Dim wPress1Unit As UShort     'See F28_PRESS_UNITS
      Dim fPress1Min As Single
      Dim fPress1Max As Single
      Dim fSetFillP1 As Single      'Setpoint auto-fill
      Dim fRatioMax As Single
      Dim fRatioMin As Single
      Dim wFillMode As UShort       'STD_FILL_MODE / AUTOFILL_MODE
      Dim wLeakUnit As UShort       'See F28_LEAK_UNITS
      Dim wRejectCalc As UShort     'Pa or Pa/s
      Dim wVolumeUnit As UShort     'See F28_ENUM_VOLUME_UNIT
      Dim fVolume As Single
      Dim fRejectMin As Single
      Dim fRejectMax As Single
      Dim fCoeffAutoFill As Single
      Dim wOptions As UShort        'Options parameters
      Dim fPatmSTD As Single        'Patm standard condition (hPa)
      Dim fTempSTD As Single        'Temperature standard condition (°C)
      Dim fFilterTime As Single     'in (s)
      Dim fOffsetLeak As Single     'Offset on the leak
      DIM fVolumeRef As Single
      DIM wTpsTestCompTemp As UShort
      DIM wPourcCompTemp As UShort
      DIM wTpsWaitingTime As UShort
      DIM wLastConsigneDacEasy As UShort
      DIM fNominalValue As Single
      DIM fCoeffMin As Single
      DIM fCoeffMax As Single

End Structure
```

**Declaration in C#.Net:**

```
[StructLayout(LayoutKind.Sequential, Pack = 1, CharSet = CharSet.Ansi)]
public struct F28_PARAMETERS
{
      public ushort usTypeTest;    // STANDARD LEAK
      public ushort usTpsFillVol;
      public ushort usTpsTransfert;
      public ushort usTpsFill;
      public ushort usTpsStab;
      public ushort usTpsTest;
      public ushort usTpsDump;
      public ushort usPress1Unit;  // See F28_PRESS_UNITS
      public float fPress1Min;
      public float fPress1Max;
      public float fSetFillP1;     //auto-fill mode instruction
      public float fRatioMax;      //LARGE LEAK mode only
      public float fRatioMin;      //LARGE LEAK mode only
      public ushort usFillMode;    //STD_FILL_MODE / AUTOFILL_MODE
      public ushort usLeakUnit;    //See F28_LEAK_UNITS
      public ushort usRejectCalc;  //Pa or Pa/s
      public ushort usVolumeUnit;  //See F28_ENUM_VOLUME_UNIT
      public float fVolume;
      public float fRejectMin;
      public float fRejectMax;
      public float fCoeffAutoFill;
      public ushort usOptions;     //Options parameters
      public float fPatmSTD;       //Patm standard condition (hPa)
      public float fTempSTD;       //T° standard condition (in °C)
      public float fFilterTime;    //in (s)
      public float fOffsetLeak;    //Offset on the leak
      public float fVolumeRef;
      public ushort wTpsTestCompTemp;
      public ushort wPourcCompTemp;
      public WORD wTpsWaitingTime;
      public ushort wLastConsigneDacEasy;
      public float fNominalValue;
      public float fCoeffMin;
      public float fCoeffMax;
};
```

#### *5.7.1.2 Options*

Uses with "**wOptions**" parameter

| Element | Data type | Value | Description |
| --- | --- | --- | --- |
| wOptions | UShort | BIT_SIGN (bit 0)<br>BIT_NO_NEGATIVE_VALUE (bit 1)<br>BIT (bit 2 reserved)<br>BIT (bit 3 reserved)<br>BIT_TEST_PRESSURE_CORR (bit 4)<br>BIT_ELECTRONIC_REGULATOR (bit 5) | Sign option validation<br>No negative value validation<br>Reserved<br>Reserved<br>Test pressure correction validation<br>Electronic regulator option |

| Element | Data type | Value | Description |
| --- | --- | --- | --- |
| | | | validation |

| Bit | 15 to 6 | 5 | 4 | 3 | 2 | 1 | 0 |
| --- | --- | --- | --- | --- | --- | --- | --- |
| **Option** | Reserved | **Electronic regulator option** | **Pressure compensation** | Reserved | Reserved | **No Negative** | **Sign** |

**Declaration in C/C++:**

```
enum F28_OPTIONS
{
      BIT_SIGN = 0,
      BIT_NO_NEGATIVE_VALUE = 1,
      BIT = 2, // reserved
      BIT = 3, // reserved
      BIT_TEST_PRESSURE_CORR = 4
      BIT_ELECTRONIC_REGULATOR = 5
};
```

**Visual Basic (Vb.Net):**

```
Enum F28_OPTIONS As UShort
      BIT_SIGN = 0
      BIT_NO_NEGATIVE_VALUE = 1
      BIT = 2 'reserved
      BIT = 3 'reserved
      BIT_TEST_PRESSURE_CORR = 4
      BIT_ELECTRONIC_REGULATOR = 5
End Enum
```

**C#.Net:**

```
public enum F28_OPTIONS : UShort
{
      BIT_SIGN = 0,
      BIT_NO_NEGATIVE_VALUE = 1,
      BIT = 2, // reserved
      BIT = 3, // reserved
      BIT_TEST_PRESSURE_CORR = 4,
      BIT_ELECTRONIC_REGULATOR = 5
};
```

#### *5.7.1.3 F28_GetModuleParameters*

The function reads parameters from the defined module.

When querying the parameters the above F28_PARAMETERS structure is expected in the function call.

**Function call:**

**C++:**

```
short F28API F28_GetModuleParameters(short sModuleID, F28_PARAMETERS* pPara);
```

**Visual Basic (Vb.Net):**

```
Public Declare Function F28_GetModuleParameters Lib _ "F28LightControl_ETH.dll" (ByVal
              sModuleID As Short, _
              ByRef Para As F28_PARAMETERS) As Short
```

**C#.Net:**

```
[DllImport("F28LightControl_ETH.dll")]
private static extern short F28_GetModuleParameters(short sModuleID, ref F28_PARAMETERS
              tPara);
```

**Arguments:**

| Argument | Data type | Description |
| --- | --- | --- |
| sModuleID | short | Identifier of module |
| pPara | F28_PARAMETERS* | Pointer to a F28_PARAMETERS structure, to place returned values in. |

**Return Value:** short

**F28_OK**: if the function succeeds.

**F28_FAIL**: if the function fails.

#### *5.7.1.4 F28_SetModuleParameters*

The function writes parameters, F28_PARAMETERS structure, to the defined module.

**Function call:**

**C++:**

```
short F28API F28_SetModuleParameters(short sModuleID, F28_PARAMETERS* pPara);
```

**Visual Basic (Vb.Net):**

```
Public Declare Function F28_SetModuleParameters Lib _ "F28LightControl_ETH.dll"_ (ByVal
              sModuleID As Short, _
              ByRef Para As F28_PARAMETERS) As Short
```

**C#.Net:**

```
[DllImport("F28LightControl_ETH.dll")]
private static extern short F28_SetModuleParameters(short sModuleID, ref F28_PARAMETERS
              tPara);
```

**Arguments:**

| Argument | Data type | Description |
| --- | --- | --- |
| sModuleID | short | Identifier of module |
| pPara | F28_PARAMETERS* | Pointer to a F28_ PARAMETERS structure to write |

**Return Value:** short

**X28_OK**: if the function succeeds.

**X28_FAIL**: if the function fails.

***Nota**: the parameters must be written to the F28 module at least once after power on.*

## 5.7.2 B28 parameters

### *5.7.2.1 Parameters structure B28_PARAMETERS*

| Element | Data type | Description |
| --- | --- | --- |
| wTypeTest | WORD | Test type parameter |
| wTpsTest | WORD | Test time in 0.01 sec (0 – 650 sec) |
| wFallTime | WORD | Fall time in 0.01 sec (0 – 650 sec) |
| fVoltageSetPoint | float | Target voltage |
| wVoltageUnit | WORD | Voltage unit |
| fMeasurementMax | float | Maximum measurement |
| wMeasurementUnit | WORD | Measurement unit |
| wRiseTime | WORD | Rise time in 0.01 sec (0 – 650 sec) |
| wOptions | WORD | Options |
| lPressureMin | long | Pressure minimum (Pa) |
| lPresureMax | long | Pressure maximum (Pa) |

**Declaration in C/C++:**

```
typedef struct
{
    WORD    wTypeTest;
    WORD    wTestTime;
    WORD    wFallTime;
    float   fVoltageSetPoint;
    WORD    wVoltageUnit;
    float   fMeasurementMax;
    WORD    wMeasurementUnit;
    WORD    wRiseTime;
    WORD    wOptions;
    long    lPressureMin;
    long    lPressureMax;
}B28_PARAMETERS;
```

**Declaration in Visual Basic 2013:**

```
<StructLayout(LayoutKind.Sequential, Pack:=1)> _
Structure B28_PARAMETERS
      Dim wTypeTest As UShort
      Dim wTestTime As UShort
      Dim wFallTime As UShort
      Dim fVoltageSetPoint As Single
      Dim wVoltageUnit As UShort
      Dim fMeasurementMax As Single
      Dim wMeasurementUnit As Ushort
      Dim wRiseTime As UShort
      Dim wOptions As Ushort
      Dim uiPressureMin As Uinteger
      Dim uiPressureMax As UInteger
End Structure
```

**Declaration in C#.Net:**

```
[StructLayout(LayoutKind.Sequential, Pack = 1, CharSet = CharSet.Ansi)]
public struct B28_PARAMETERS
{
      public ushort usTypeTest;
      public ushort usTestTime;
      public ushort usFallTime;
      public float  fVoltageSetPoint;
      public ushort usVoltageUnit;
      public float  fMeasurementMax;
      public ushort usMeasurementUnit;
      public ushort usRiseTime;
      public ushort usOptions;
      public uint   uiPressureMin;
      public uint   uiPressureMax;
};
```

#### *5.7.2.2 B28_GetModuleParameters*

The function reads parameters from the defined module.

When querying the parameters the above B28_PARAMETERS structure is expected in the function call.

**Function call:**

**C++:**

```
short B28API B28_GetModuleParameters(short sModuleID, B28_PARAMETERS* pPara);
```

**Visual Basic (Vb.Net):**

```
Public Declare Function B28_GetModuleParameters Lib _ "F28LightControl_ETH.dll" (ByVal
              sModuleID As Short, _
              ByRef Para As B28_PARAMETERS) As Short
```

**C#.Net:**

```
[DllImport("F28LightControl_ETH.dll")]
private static extern short B28_GetModuleParameters(short sModuleID, ref B28_PARAMETERS
              tPara);
```

**Arguments:**

| Argument | Data type | Description |
| --- | --- | --- |
| sModuleID | short | Identifier of module |
| pPara | B28_PARAMETERS* | Pointer to a B28_PARAMETERS structure, to place returned values in. |

**Return Value:** short

**X28_OK**: if the function succeeds.

**X28_FAIL**: if the function fails.

#### *5.7.2.3 B28_SetModuleParameters*

The function writes parameters, B28_PARAMETERS structure, to the defined module.

**Function call:**

**C++:**

```
short B28API B28_SetModuleParameters(short sModuleID, B28_PARAMETERS* pPara);
```

**Visual Basic (Vb.Net):**

```
Public Declare Function B28_SetModuleParameters Lib _ "F28LightControl_ETH.dll"_ (ByVal
              sModuleID As Short, _
              ByRef Para As B28_PARAMETERS) As Short
```

**C#.Net:**

```
[DllImport("F28LightControl_ETH.dll")]
private static extern short B28_SetModuleParameters(short sModuleID, ref B28_PARAMETERS
              tPara);
```

**Arguments:**

| Argument | Data type | Description |
| --- | --- | --- |
| sModuleID | short | Identifier of module |
| pPara | B28_PARAMETERS* | Pointer to a B28_ PARAMETERS structure to write |

**Return Value:** short

**X28_OK**: if the function succeeds.

**X28_FAIL**: if the function fails.

***Nota**: the parameters must be written to the B28 module at least once after power on.*

## 5.8 Special cycle related functions

### 5.8.1 B28 functions

#### *5.8.1.1 B28_StartVoltageAdjust*

This function allows adjusting the distorsion voltage.

***Nota**: When running, a "**Reset cycle**" must be called to exit the function.*

**Function call:**

**C++:**

```
short B28API B28_StartVoltageAdjust(short sModuleID);
```

**Visual Basic (Vb.Net):**

```
Public Declare Function B28_ StartVoltageAdjust Lib _ "F28LightControl_ETH.dll"
              (ByVal sModuleID As Short) As Short
```

**C#.Net:**

```
[DllImport("F28LightControl_ETH.dll")]
private static extern short B28_StartVoltageAdjust(short sModuleID);
```

**Arguments:**

| Argument | Data type | Description |
| --- | --- | --- |
| sModuleID | short | Identifier of module |

**Return Value:** short

**X28_OK**: if the function succeeds.

**X28_FAIL**: if the function fails.

#### *5.8.1.2 B28_StartRegulatorAdjust*

This function allows adjusting manually the regulator.

***Nota**: When running, a "**Reset cycle**" must be called to exit the function.*

**Function call:**

**C++:**

```
short B28API B28_StartRegulatorAdjust(short sModuleID);
```

**Visual Basic (Vb.Net):**

```
Public Declare Function B28_StartRegulatorAdjust Lib _ "F28LightControl_ETH.dll"
              (ByVal sModuleID As Short) As Short
```

**C#.Net:**

```
[DllImport("F28LightControl_ETH.dll")]
private static extern short B28_StartRegulatorAdjust(short sModuleID);
```

**Arguments:**

| Argument | Data type | Description |
| --- | --- | --- |
| sModuleID | short | Identifier of module |

**Return Value:** short

**X28_OK**: if the function succeeds.

**X28_FAIL**: if the function fails.

### 5.8.2 F28 functions

#### *5.8.2.1 F28_StartAutoZeroPressure*

The function starts an auto-zero pressure special cycle of the defined module.

**Function call:**

**C++:**

```
short F28API F28_StartAutoZeroPressure(short sModuleID, float fDumpTime);
```

**Visual Basic (Vb.Net):**

```
Public Declare Function F28_StartAutoZeroPressure Lib _ "F28LightControl_ETH.dll"
              (ByVal sModuleID As Short, ByVal _ fDumpTime As Single) As Short
```

**C#.Net:**

```
[DllImport("F28LightControl_ETH.dll")]
private static extern short F28_StartAutoZeroPressure(short sModuleID, float
              fDumpTime);
```

**Arguments:**

| Argument | Data type | Description |
| --- | --- | --- |
| sModuleID | short | Identifier of module |
| fDumpTime | float | Dump time in seconds. |

**Return Value:** short

**X28_OK**: if the function succeeds.

**X28_FAIL**: if the function fails.

### *5.8.2.2 F28_StartRegulatorAdjust*

This function allows adjusting manually the regulator.

***Nota**: When running, a "**Reset cycle**" must be called to exit the function.*

**Function call:**

**C++:**

```
short F28API F28_StartRegulatorAdjust(short sModuleID);
```

**Visual Basic (Vb.Net):**

```
Public Declare Function F28_StartRegulatorAdjust Lib _ "F28LightControl_ETH.dll"
              (ByVal sModuleID As Short) As Short
```

**C#.Net:**

```
[DllImport("F28LightControl_ETH.dll")]
private static extern short F28_StartRegulatorAdjust(short sModuleID);
```

**Arguments:**

| Argument | Data type | Description |
| --- | --- | --- |
| sModuleID | short | Identifier of module |

**Return Value:** short

**X28_OK**: if the function succeeds.

**X28_FAIL**: if the function fails.

### 5.8.2.3 F28_StartLearningRegulator

The function starts an auto-zero for the pressure sensor and then starts an electronic regulator learning special cycle of the defined module.

This special cycle starts automatically at the module power on.

**Function call:**

**C++:**

```
short F28API F28_StartLearningRegulator(short sModuleID, float fDumpTime);
```

**Visual Basic (Vb.Net):**

```
Public Declare Function F28_ StartLearningRegulator Lib _ "F28LightControl_ETH.dll"
              (ByVal sModuleID As Short, ByVal _ fDumpTime As Single) As Short
```

**C#.Net:**

```
[DllImport("F28LightControl_ETH.dll")]
private static extern short F28_ StartLearningRegulator (short sModuleID, float
              fDumpTime);
```

**Arguments:**

| Argument | Data type | Description |
| --- | --- | --- |
| sModuleID | short | Identifier of module |
| fDumpTime | float | Dump time in seconds. |

**Return Value:** short

**X28_OK**: if the function succeeds.

**X28_FAIL**: if the function fails.

***Nota**: the Dump time (fDumpTime parameter) is the pressure sensor auto-zero time, after this time, the regulator learning cycle begins.*

### 5.8.2.4 F28_StartJetCheck

The function start a Jet check special cycle of the defined module.

**Function call:**

**C++:**

```
short F28API F28_StartJetCheck(short sModuleID);
```

**Visual Basic (Vb.net)**

```
Public Declare Function F28_StartJetCheck Lib _ "F28LightControl_Eth.dll" (ByVal
              sModuleID As Short) As Short
```

**C#.Net**

```
[dllImport("F28LightControl_ETH.dll")]
private static extern short F28_StartJetCheck(short sModuleID) ;
```

**Argument :**

| Argument | Data type | Description |
| --- | --- | --- |
| sModuleID | short | Identifier of module |

**Return Value:** short

**X28_OK**: if the function succeeds.

**X28_FAIL**: if the function fails.

***Warning!*** *When this special cycle has run, the leak unit value* `(wLeakUnit)` *in the results becomes 255 that is a millimeters unit (mm).*

### *5.8.2.5 X28_StartVolumeLearning*

The function start a volume learning (test or reference) special cycle of the defined module.

**Function call:**

**C++:**

```
short F28API X28_StartVolumeLearning(short sModuleID, X28_VOLUME_LEARNING_TEST
              *pPara, BOOL bTest);
```

**Visual Basic (Vb.net)**

```
Public Declare Function X28_StartVolumeLearning Lib _ "F28LightControl_Eth.dll"
              (ByVal sModuleID As Short, ByRef pPara As X28_VOLUME_LEARNING_TEST,
              Integer bTest) As Short
```

**C#.Net**

```
[dllImport("F28LightControl_ETH.dll")]
private static extern short X28_StartVolumeLearning(short sModuleID, ref
              X28_VOLUME_LEARNING_TEST pPara, int bTest) ;
```

**Argument :**

| Argument | Data type | Description |
| --- | --- | --- |
| sModuleID | short | Identifier of module |
| pPara | `X28_VOLUME_LEARNING_TEST` | Parameters for the test |
| bTest | int | 0 to compute reference volume<br>1 to compute test volume |

**Return Value:** short

**X28_OK**: if the function succeeds.

**X28_FAIL:** if the function fails.

#### *5.8.2.6 X28_StartVidageInfini*

The function start an infinite dump special cycle of the defined module.

**Function call:**

**C++:**

```
short F28API X28_StartVidageInfini(short sModuleID);
```

**Visual Basic (Vb.net)**

```
Public Declare Function X28_ StartVidageInfini Lib _ "F28LightControl_Eth.dll" (ByVal
              sModuleID As Short) As Short
```

**C#.Net**

```
[dllImport("F28LightControl_ETH.dll")]
private static extern short X28_StartVidageInfini(short sModuleID) ;
```

**Argument :**

| Argument | Data type | Description |
| --- | --- | --- |
| sModuleID | short | Identifier of module |

**Return Value:** short

**X28_OK**: if the function succeeds.

**X28_FAIL**: if the function fails.

## 5.9 Result related functions

### 5.9.1 F28 functions

#### *5.9.1.1 F28_ClearFIFOResults*

This function clears the result inside the FIFO.

***Note****: the FIFO contains only one result.*

**Function call:**

**C++:**

```
short F28API F28_ClearFIFOResults(short sModuleID);
```

**Visual Basic (Vb.Net):**

```
Public Declare Function F28_ClearFIFOResults Lib "F28LightControl_ETH.dll" _ (ByVal
              sModuleID As Short) As Short
```

**C#.Net:**

```
[DllImport("F28LightControl_ETH.dll")]
private static extern short F28_ClearFIFOResults(short sModuleID);
```

**Arguments:**

| Argument | Data type | Description |
| --- | --- | --- |
| sModuleID | short | Identifier of module |

**Return Value:** short

**X28_OK**: if the function succeeds.

**X28_FAIL**: if the function fails.

### *5.9.1.2 F28_GetResultsCount*

This function reads the number of results available in FIFO.

***Nota**: when the result is available, the result count is equal to 1.*

**Function call:**

**C++:**

```
WORD F28API F28_GetResultsCount(short sModuleID);
```

**Visual Basic (Vb.Net):**

```
Public Declare Function F28_GetResultsCount Lib "F28LightControl_ETH.dll" _ (ByVal
              sModuleID As Short) As UShort
```

**C#.Net:**

```
[DllImport("F28LightControl_ETH.dll")]
private static extern ushort F28_GetResultsCount(short sModuleID);
```

**Arguments:**

| Argument | Data type | Description |
| --- | --- | --- |
| sModuleID | Short | Identifier of module |

**Return Value:** short

**Number of results**: 0 / 1

***Nota**: the FIFO contains only one result.*

### *5.9.1.3 Result structure F28_RESULT*

| Element | Data type | Description |
| --- | --- | --- |
| ucStatus | UCHAR | Status of result |
| fPressureValue | float | Pressure value |
| fLeakValue | float | Leak value |
| ucUnitPressure | UCHAR | Pressure unit |
| ucUnitLeak | UCHAR | Leak unit |

| Element | Data type | Description |
|---|---|---|
| ucGroupID | UCHAR | Group identifier |
| ucModuleAddr | UCHAR | Module identifier |
| wYear | WORD | Year |
| wMonth | WORD | Month |
| wDay | WORD | Day |
| wHour | WORD | Hour |
| wMinute | WORD | Minute |
| wSecond | WORD | Second |

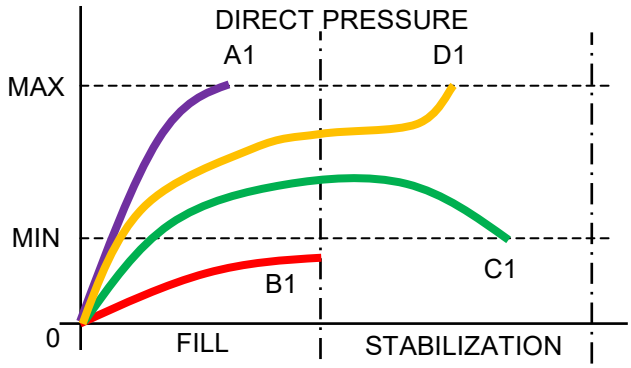
See result status below.

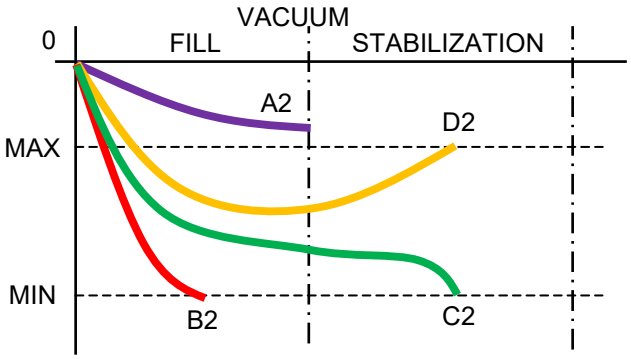
#### *5.9.1.4 Result status and alarms*

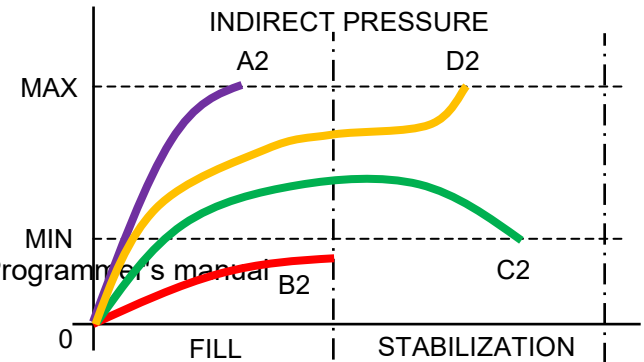
| Element | Data type | valueCode | Description | | Leak result value* |
|---|---|---|---|---|---|
| ucStatus | UCHAR | 0 | STATUS_GOOD_PART | Pass part | Value |
| | | 1 | STATUS_TEST_FAIL_PART | Test fail part. Not used (Reject level at 999) | Value |
| | | 2 | STATUS_REF_FAIL_PART | Reference fail part | Value |
| | | 3 | STATUS_ALARM_EEEE | Large leak on Test side, over full scale | Value |
| | | 4 | STATUS_ALARM_MMMM | Large leak on Reference side, over full scale | Value |
| | | 5 | STATUS_ALARM_PPPP | Pressure over the maximum pressure range (Tester error) | -399.99 |
| | | 6 | STATUS_ALARM_MPPP | Pressure below the minimum pressure range (Tester error) | -399.99 |
| | | 7 | STATUS_ALARM_OFFD_FUITE | Differential sensor auto-zero error (Tester error) | -399.99 |
| | | 8 | STATUS_ALARM_OFFD_PRESSION | Piezo sensor auto-zero error (Tester error) | -399.99 |
| | | 9 | STATUS_ALARM_PST | Over maximum pressure (pressure too high) | -399.99 |
| | | | | if "Sign" is checked (vacuum or indirect test) | Value |
| | | 10 | STATUS_ALARM_MPST | Below minimum pressure (pressure too low) | Value |
| | | | | if "Sign" is checked (vacuum or indirect test) | -399.99 |
| | | 11 | STATUS_ALARM_CS_VOLUME_PETIT | Fail Sealed components volume too small (Tester error) | -399.99 |
| | | 12 | STATUS_ALARM_CS_VOLUME_GRAND | Fail Sealed components volume too large (Tester error) | -399.99 |
| | | 13 | STATUS_ALARM_ERREUR_PRESS_CALIBRATION | Calibration pressure error (Tester error) | -399.99 |

| Element | Data type | valueCode | Description | | Leak result value* |
|---|---|---|---|---|---|
| | | 14 | STATUS_ALARM_ERREUR_LEAK_ CALIBRATION | Calibration leak error (Tester error) | -399.99 |
| | | 15 | STATUS_ALARM_ERREUR_LINE_ PRESS_CALIB | Calibration line pressure error (Tester error) | -399.99 |
| | | 16 | STATUS_ALARM_APPR_REG_ ELEC_ERROR | Electronic regulator learning fail | -399.99 |
| | | 17 | STATUS_ALARM_TEST_PART_ LARGE_LEAK | Large leak on Test side Alarm (no value) | +998.00 |
| | | 18 | STATUS_ALARM_REF_SIDE_ LARGE_LEAK | Large leak on Reference side Alarm (no value) | -399.99 |
| | | 19 | STATUS_ALARM_P_TOO_LARGE_ FILL ***See diagrams below*** | Over maximum pressure (pressure too high). **Case A1** | -399.99 |
| | | | | If "Sign" is checked (vacuum or indirect test) and over max pressure**. Case A2** | +999.00 |
| | | 20 | STATUS_ALARM_P_TOO_LOW_ FILL ***See diagrams below*** | Pressure Below min pressure (pressure too low). **Case B1** | +999.00 |
| | | | | If "Sign" is checked (vacuum or indirect test) and below min pressure**. CaseB2** | -399.99 |
| | | 21 | STATUS_ALARM_JET_CHECK_FAIL | Jet Check out of limits (Jet air supply out of limits or Jet damaged). | -399.99 |
| | | 22 | STATUS_ALARM_JET_CHECK_PASS | Jet Check special cycle succeed | -399.99 |
| | | 23 | INCOMPATIBLE_DEVICE | Used test type is not compatible with the device. | |

*The "**Leak result value**" is sent in the result frame, these specific values are only available from the 1.500 DLL version.



**Declaration in C/C++:**

```
typedef struct
{
    WORD wYear;
    WORD wMonth;
    WORD wDay;
    WORD wHour;
    WORD wMinute;
    WORD wSecond;
}X28_DATE;

typedef struct
{
    UCHAR ucStatus;
    float fPressureValue;
    float fLeakValue;
    UCHAR ucUnitPressure;
    UCHAR ucUnitLeak;
    BYTE ucGroupID;
    BYTE ucModuleAddr;
    X28_DATE dateReceived;
}F28_RESULT;
```

**Declaration in Visual Basic 2013:**

```
<StructLayout(LayoutKind.Sequential, Pack:=1)> _
Structure X28_DATE
    Dim wYear As UShort
    Dim wMonth As UShort
    Dim wDay As UShort
    Dim wHour As UShort
    Dim wMinute As UShort
    Dim wSecond As UShort
End Structure

<StructLayout(LayoutKind.Sequential, Pack:=1)> _
Structure F28_RESULT
    Dim ucStatus As Byte
    Dim fPressureValue As Single
    Dim fLeakValue As Single
    Dim ucUnitPressure As Byte
    Dim ucUnitLeak As Byte
    Dim GroupID As Byte           'F28_GROUP_ID
    Dim ModuleAddr As Byte        'F28_MODULE_ADDR
    Dim dateReceived As X28_DATE
End Structure
```

**Declaration in C#.Net:**

```
[StructLayout(LayoutKind.Sequential, Pack = 1, CharSet = CharSet.Ansi)]
public struct X28_DATE
{
      public ushort usYear;
      public ushort usMonth;
      public ushort usDay;
      public ushort usHour;
      public ushort usMinute;
      public ushort usSecond;
};

[StructLayout(LayoutKind.Sequential, Pack = 1, CharSet = CharSet.Ansi)]
public struct F28_RESULT
{
      public byte     bStatus;
      public float    fPressureValue;
      public float    fLeakValue;
      public byte     bUnitPressure;
      public byte     bUnitLeak;
      public byte     bGroupID;
      public byte     bModuleAddr;
      public X28_DATE dateReceived;
};
```

#### *5.9.1.5 F28_GetNextResult*

This function retrieves one result from the FIFO. When querying the above **F28_RESULT** structure is expected in the function call.

***Nota**: after reading, the result count is equal to 0.*

**Function call:**

**C++:**

```
short F28API F28_GetNextResult(short sModuleID, F28_RESULT* pResult);
```

**Visual Basic (Vb.Net):**

```
Public Declare Function F28_GetNextResult Lib "F28LightControl_ETH.dll" _ (ByVal
              sModuleID As Short, ByRef Result As F28_RESULT) As Short
```

**C#.Net:**

```
[DllImport("F28LightControl_ETH.dll")]
private static extern short F28_GetNextResult(short sModuleID, ref F28_RESULT
              tResult);
```

**Arguments:**

| Argument | Data type | Description |
| --- | --- | --- |
| sModuleID | short | Identifier of module |
| pResult | F28_RESULT* | Pointer to a F28_RESULT structure, to place returned |

| | | |
|---|---|---|
| | | values in. |

**Return Value:** short

**X28_OK**: if the function succeeds.

**X28_FAIL**: if the function fails.

#### *5.9.1.6 F28_GetLastResult*

This function retrieves the last cycle result. When querying the above **F28_RESULT** structure is expected in the function call.

**Function call:**

**C++:**

```
short F28API F28_GetLastResult(short sModuleID, F28_RESULT* pResult);
```

**Visual Basic (Vb.Net):**

```
Public Declare Function F28_GetLastResult Lib "F28LightControl_ETH.dll" _ (ByVal
              sModuleID As Short, ByRef Result As F28_RESULT) As Short
```

**C#.Net:**

```
[DllImport("F28LightControl_ETH.dll")]
private static extern short F28_GetLastResult(short sModuleID, ref F28_RESULT
              tResult);
```

**Arguments:**

| Argument | Data type | Description |
|---|---|---|
| sModuleID | short | Identifier of module |
| pResult | F28_RESULT* | Pointer to a F28_RESULT structure, to place returned values in. |

**Return Value:** short

**X28_OK**: if the function succeeds.

**X28_FAIL**: if the function fails.

***Note**: if the result is valid, this function can be called one or more times.*

### 5.9.2 B28 functions

#### *5.9.2.1 B28_ClearFIFOResults*

This function clears the result inside the FIFO.

***Nota**: the FIFO contains only one result.*

**Function call:**

**C++:**

```
short X28API X28_ClearFIFOResults(short sModuleID);
```

**Visual Basic (Vb.Net):**

```
Public Declare Function X28_ClearFIFOResults Lib "F28LightControl_ETH.dll" _ (ByVal
              sModuleID As Short) As Short
```

**C#.Net:**

```
[DllImport("F28LightControl_ETH.dll")]
private static extern short X28_ClearFIFOResults(short sModuleID);
```

**Arguments:**

| Argument | Data type | Description |
| --- | --- | --- |
| sModuleID | short | Identifier of module |

**Return Value:** short

**X28_OK**: if the function succeeds.

**X28_FAIL**: if the function fails.

#### *5.9.2.2 B28_GetResultsCount*

This function reads the number of results available in FIFO.

***Nota**: when the result is available, the result count is equal to 1.*

**Function call:**

**C++:**

```
WORD B28API B28_GetResultsCount(short sModuleID);
```

**Visual Basic (Vb.Net):**

```
Public Declare Function B28_GetResultsCount Lib "F28LightControl_ETH.dll" _ (ByVal
              sModuleID As Short) As UShort
```

**C#.Net:**

```
[DllImport("F28LightControl_ETH.dll")]
private static extern ushort B28_GetResultsCount(short sModuleID);
```

**Arguments:**

| Argument | Data type | Description |
| --- | --- | --- |
| sModuleID | Short | Identifier of module |

**Return Value:** short

**Number of results**: 0 / 1

***Nota**: the FIFO contains only one result.*

### *5.9.2.3 Result structure B28_RESULT*

| Element | Data type | Description |
|---|---|---|
| ucStatus | UCHAR | Status of result |
| fVoltageValue | float | Voltage value |
| fMeasurementValue | float | Measurement value |
| ucUnitVoltage | UCHAR | Voltage unit |
| ucUnitMeasurement | UCHAR | Measurement unit |
| ucGroupID | UCHAR | Group identifier |
| ucModuleAddr | UCHAR | Module identifier |
| wYear | WORD | Year |
| wMonth | WORD | Month |
| wDay | WORD | Day |
| wHour | WORD | Hour |
| wMinute | WORD | Minute |
| wSecond | WORD | Second |

See result status below.

### *5.9.2.4 Result status and alarms*

| Element | Data type | valueCode | Description | | Measurement result value* |
|---|---|---|---|---|---|
| | UCHARucStatus | 0 | STATUS_GOOD_PART | Pass part | Value |
| | | 1 | STATUS_TEST_FAIL_PART | Test fail part Over maximum measurement | Value |
| | | 2 | STATUS_VOLTAGE_OUT | Voltage out of limit | Value |
| | | 3 | STATUS_ALARM_EEEE | Large measurement, over full scale | 999.99 |
| | | 4 | STATUS_ALARM_PST | Pressure out of limit | Value |

*The "**Measurement result value**" is sent in the result frame.

**Declaration in C/C++:**

```
typedef struct
{
    WORD wYear;
    WORD wMonth;
    WORD wDay;
    WORD wHour;
    WORD wMinute;
    WORD wSecond;
}X28_DATE;

typedef struct
{
     UCHAR ucStatus;
     float fVoltageValue;
     float fMeasurementValue;
     UCHAR ucUnitVoltage;
     UCHAR ucUnitMeasurement;
     BYTE ucGroupID;
     BYTE ucModuleAddr;
     X28_DATE dateReceived;
} B28_RESULT;
```

**Declaration in Visual Basic 2013:**

```
<StructLayout(LayoutKind.Sequential, Pack:=1)> _
Structure X28_DATE
    Dim wYear As UShort
    Dim wMonth As UShort
    Dim wDay As UShort
    Dim wHour As UShort
    Dim wMinute As UShort
    Dim wSecond As UShort
End Structure

<StructLayout(LayoutKind.Sequential, Pack:=1)> _
Structure B28_RESULT
     Dim ucStatus As Byte
     Dim fVoltageValue As Single
     Dim fMeasurementValue As Single
     Dim ucUnitVoltage As Byte
     Dim ucUnitMeasurement As Byte
     Dim GroupID As Byte                  ' X28_GROUP_ID
     Dim ModuleAddr As Byte               ' X28_MODULE_ADDR
     Dim dateReceived As X28_DATE
End Structure
```

**Declaration in C#.Net:**

```
[StructLayout(LayoutKind.Sequential, Pack = 1, CharSet = CharSet.Ansi)]
public struct X28_DATE
{
      public ushort usYear;
      public ushort usMonth;
      public ushort usDay;
      public ushort usHour;
      public ushort usMinute;
      public ushort usSecond;
};

[StructLayout(LayoutKind.Sequential, Pack = 1, CharSet = CharSet.Ansi)]
public struct B28_RESULT
{
      public byte     bStatus;
      public float    fVoltageValue;
      public float    fMeasurementValue;
      public byte     bUnitVoltage;
      public byte     bUnitMeasurement;
      public byte     bGroupID;
      public byte     bModuleAddr;
      public X28_DATE dateReceived;
};
```

#### 5.9.2.5 B28_GetNextResult

This function retrieves one result from the FIFO. When querying the above **B28_RESULT** structure is expected in the function call.

***Nota**: after reading, the result count is equal to 0.*

**Function call:**

**C++:**

```
short B28API B28_GetNextResult(short sModuleID, B28_RESULT* pResult);
```

**Visual Basic (Vb.Net):**

```
Public Declare Function B28_GetNextResult Lib "F28LightControl_ETH.dll" _ (ByVal
              sModuleID As Short, ByRef Result As B28_RESULT) As Short
```

**C#.Net:**

```
[DllImport("F28LightControl_ETH.dll")]
private static extern short B28_GetNextResult(short sModuleID, ref B28_RESULT
              tResult);
```

**Arguments:**

| Argument | Data type | Description |
| --- | --- | --- |
| sModuleID | short | Identifier of module |
| pResult | B28_RESULT* | Pointer to a B28_RESULT structure, to place returned |

| | | |
|---|---|---|
| | | values in. |

**Return Value:** short

**X28_OK**: if the function succeeds.

**X28_FAIL**: if the function fails.

#### *5.9.2.6 B28_GetLastResult*

This function retrieves the last cycle result. When querying the above B**28_RESULT** structure is expected in the function call.

**Function call:**

**C++:**

```
short B28API B28_GetLastResult(short sModuleID, B28_RESULT* pResult);
```

**Visual Basic (Vb.Net):**

```
Public Declare Function B28_GetLastResult Lib "F28LightControl_ETH.dll" _ (ByVal
              sModuleID As Short, ByRef Result As B28_RESULT) As Short
```

**C#.Net:**

```
[DllImport("F28LightControl_ETH.dll")]
private static extern short B28_GetLastResult(short sModuleID, ref B28_RESULT
              tResult);
```

**Arguments:**

| Argument | Data type | Description |
|---|---|---|
| sModuleID | short | Identifier of module |
| pResult | B28_RESULT* | Pointer to a B28_RESULT structure, to place returned values in. |

**Return Value:** short

**X28_OK**: if the function succeeds.

**X28_FAIL**: if the function fails.

***Note**: if the result is valid, this function can be called one or more times.*

## 5.10 Real time cycle related functions

### 5.10.1 F28 functions

#### *5.10.1.1 Real time data structure F28_REALTIME_CYCLE*

| Element | Data type | Description |
| --- | --- | --- |
| ucEndCycle | UCHAR | 1 = End of cycle, 0 = Cycle in progress |
| ucStatus | UCHAR | see Step code |
| fPressureValue | float | Pressure value |
| fLeakValue | float | Leak value |
| ucUnitPressure | UCHAR | Pressure unit |
| ucUnitLeak | UCHAR | Leak unit |
| fInternalTemperature | float | Temperature in °C (NU) |
| fPatm | float | Absolute pressure in hPa |

**Declaration in C/C++:**

```
// Real time structure
typedef struct F28_REALTIME_CYCLE
{
      UCHAR ucEndCycle;
      UCHAR ucStatus;
      float fPressureValue;
      float fLeakValue;
      UCHAR ucUnitPressure;
      UCHAR ucUnitLeak;
      float fInternalTemperature;
      float fPatm;
}F28_REALTIME_CYCLE;
```

**Declaration in Visual Basic 2013:**

```
<StructLayout(LayoutKind.Sequential, Pack:=1)> _
Structure F28_REALTIME_CYCLE
     Dim ucEndCycle As Byte
     Dim ucStatus As Byte
     Dim fPressureValue As Single
     Dim fLeakValue As Single
     Dim ucUnitPressure As Byte
     Dim ucUnitLeak As Byte
     Dim fInternalTemperature As Single      ' Temperature in °C
     Dim fPatm As Single                     ' Abs pressure in hPa
End Structure
```

**C#.Net:**

```
[StructLayout(LayoutKind.Sequential, Pack = 1, CharSet = CharSet.Ansi)]
public struct F28_REALTIME_CYCLE
{
      public byte     bEndCycle;
      public byte     bStatus;
      public float    fPressureValue;
      public float    fLeakValue;
      public byte     bUnitPressure;
      public byte     bUnitLeak;
      public float    fInternalTemperature;
      public float    fPatm;
};
```

#### 5.10.1.2 F28_GetRealTimeData

The function reads real time data from the defined module. When querying, the above F28_REALTIME_CYCLE structure is expected in the function call.

**Function call:**

**C++:**

```
short F28API F28_GetRealTimeData(short sModuleID, F28_REALTIME_CYCLE* pCycle);
```

**Visual Basic (Vb.Net):**

```
Public Declare Function F28_GetRealTimeData Lib "F28LightControl_ETH.dll" _ (ByVal
              sModuleID As Short, ByRef Cycle As _ F28_REALTIME_CYCLE)  As Short
```

**C#.Net:**

```
[DllImport("F28LightControl_ETH.dll")]
private static extern short F28_GetRealTimeData(short sModuleID, ref
              F28_REALTIME_CYCLE tCycle);
```

**Arguments:**

| Argument | Data type | Description |
| --- | --- | --- |
| sModuleID | short | Identifier of module |
| pCycle | F28_REALTIME_CYCLE* | Pointer to a F28_REALTIME_CYCLE structure, to place returned values in. |

**Return Value:** short

**X28_OK**: if the function succeeds.

**X28_FAIL**: if the function fails.

### 5.10.2 B28 functions

#### 5.10.2.1 Real time data structure B28_REALTIME_CYCLE

| Element | Data type | Description |
| --- | --- | --- |
| ucEndCycle | UCHAR | 1 = End of cycle, 0 = Cycle in progress |
| ucStatus | UCHAR | see Step code |
| fVoltageValue | float | Voltage value |
| fMeasurementValue | float | Measurement value |
| ucUnitVoltage | UCHAR | Voltage unit |
| ucUnitMeasurement | UCHAR | Measurement unit |
| fInternalTemperature | float | Temperature in °C (NU) |
| fPatm | float | Abs pressure in Pa |

**Declaration in C/C++:**

```
// Real time structure
typedef struct B28_REALTIME_CYCLE
{
      UCHAR ucEndCycle;
```

```
      UCHAR ucStatus;
      float fVoltageValue;
      float fMeasurementValue;
      UCHAR ucUnitVoltage;
      UCHAR ucUnitMeasurement;
      float fInternalTemperature;
      float fPatm;
}B28_REALTIME_CYCLE;
```

**Declaration in Visual Basic 2013:**

```
<StructLayout(LayoutKind.Sequential, Pack:=1)> _
Structure B28_REALTIME_CYCLE
    Dim ucEndCycle As Byte
    Dim ucStatus As Byte
    Dim fVoltageValue As Single
    Dim fMeasurementValue As Single
    Dim ucUnitVoltage As Byte
    Dim ucUnitMeasurement As Byte
    Dim fInternalTemperature As Single      ' Temperature in °C
    Dim fPatm As Single                     ' Abs pressure in hPa
End Structure
```

**C#.Net:**

```
[StructLayout(LayoutKind.Sequential, Pack = 1, CharSet = CharSet.Ansi)]
public struct B28_REALTIME_CYCLE
{
      public byte     bEndCycle;
      public byte     bStatus;
      public float    fVoltageValue;
      public float    fMeasurementValue;
      public byte     bUnitVoltage;
      public byte     bUnitMeasurement;
      public float    fInternalTemperature;
      public float    fPatm;
};
```

### *5.10.2.2 B28_GetRealTimeData*

The function reads real time data from the defined module. When querying, the above B28_REALTIME_CYCLE structure is expected in the function call.

**Function call:**

**C++:**

```
short B28API B28_GetRealTimeData(short sModuleID, B28_REALTIME_CYCLE* pCycle);
```

**Visual Basic (Vb.Net):**

```
Public Declare Function B28_GetRealTimeData Lib "F28LightControl_ETH.dll" _ (ByVal
              sModuleID As Short, ByRef Cycle As _ B28_REALTIME_CYCLE)  As Short
```

**C#.Net:**

```
[DllImport("F28LightControl_ETH.dll")]
private static extern short B28_GetRealTimeData(short sModuleID, ref
              B28_REALTIME_CYCLE tCycle);
```

**Arguments:**

| Argument | Data type | Description |
| --- | --- | --- |
| sModuleID | short | Identifier of module |
| pCycle | B28_REALTIME_CYCLE* | Pointer to a B28_REALTIME_CYCLE structure, to place returned values in. |

**Return Value:** short

**X28_OK**: if the function succeeds.

**X28_FAIL**: if the function fails.

## 5.11 Statistics counter related functions

These functions concern all device B28/F28.

### 5.11.1 Cycle statistics X28_CYCLE_STATISTICS

| Element | Data type | Description |
| --- | --- | --- |
| dwTotalCycles | DWORD | Cycle counter |
| dwFailCycles | DWORD | Fail counter |
| dwSuccessCycles | DWORD | Pass counter |

**Declaration in C/C++:**

```
typedef struct
{
      DWORD dwTotalCycles;
      DWORD dwFailCycles;
      DWORD dwSuccessCycles;
}F28_CYCLE_STATISTICS;
```

**Declaration in Visual Basic 2013:**

```
<StructLayout(LayoutKind.Sequential, Pack:=1)> _
Structure F28_CYCLE_STATISTICS
     Dim dwTotalCycles As UInteger
     Dim dwFailCycles As UInteger
     Dim dwSuccessCycles As UInteger
End Structure
```

**Declaration in C#.Net:**

```
[StructLayout(LayoutKind.Sequential, Pack = 1, CharSet = CharSet.Ansi)]
public struct F28_CYCLE_STATISTICS
{
      public uint uiTotalCycles;
      public uint uiFailCycles;
```

```
      public uint uiSuccessCycles;
};
```

### 5.11.2 X28_GetCycleStatistics

This function allows reading the cycle statistics.

When querying the above cycle statistic, F28_CYCLE_STATISTICS, structure is expected in the function call.

**Function call:**

**C++:**

```
short X28API X28_GetCycleStatistics(short sModuleID, X28_CYCLE_STATISTICS* pInfo);
```

**Visual Basic (Vb.Net):**

```
Public Declare Function X28_GetCycleStatistics Lib _ "F28LightControl_ETH.dll" (ByVal
              sModuleID As Short, ByRef _ Info As X28_CYCLE_STATISTICS) As Short
```

**C#.Net:**

```
[DllImport("F28LightControl_ETH.dll")]
private static extern short X28_GetCycleStatistics(short sModuleID, ref
              X28_CYCLE_STATISTICS tInfo);
```

**Arguments:**

| Argument | Data type | Description |
| --- | --- | --- |
| sModuleID | short | Identifier of module |
| pInfo | X28_CYCLE_STATISTICS* | Pointer to a X28_CYCLE_STATISTICS structure, to place returned values in. |

**Return Value:** short

**X28_OK**: if the function succeeds.

**X28_FAIL**: if the function fails.

### 5.11.3 Communication statistics structure X28_COMMUNICATION_STATISTICS

| Element | Data type | Description |
| --- | --- | --- |
| dwTransmited | DWORD | Transmit counter |
| dwReceived | DWORD | Receive counter |
| dwErrors | DWORD | Error counter |

**Declaration in C/C++:**

```
typedef struct
{
      DWORD dwTransmited;
      DWORD dwReceived;
      DWORD dwErrors;
```

```
}X28_COMMUNICATION_STATISTICS;
```

**Declaration in Visual Basic 2013:**

```
<StructLayout(LayoutKind.Sequential, Pack:=1)> _
Structure X28_COMMUNICATION_STATISTICS
    Dim dwTransmited As UInteger
    Dim dwReceived As UInteger
    Dim dwErrors As UInteger
End Structure
```

**Declaration in C#.Net:**

```
[StructLayout(LayoutKind.Sequential, Pack = 1, CharSet = CharSet.Ansi)]
public struct X28_CYCLE_STATISTICS
{
     public uint uiTotalCycles;
     public uint uiFailCycles;
     public uint uiSuccessCycles;
};
```

### 5.11.4 X28_GetCommunicationStatistics

This function allows reading the communication statistics.

When querying the above communication statistic structure, X28_COMMUNICATION_STATISTICS, is expected in the function call.

**Function call:**

**C++:**

```
short X28API X28_GetCommunicationStatistics(short sModuleID,
             X28_COMMUNICATION_STATISTICS* pInfo);
```

**Visual Basic (Vb.Net):**

```
Public Declare Function X28_GetCommunicationStatistics Lib _
             "F28LightControl_ETH.dll" (ByVal sModuleID As Short, ByRef _ Info As
             X28_COMMUNICATION_STATISTICS) As Short
```

**C#.Net:**

```
[DllImport("F28LightControl_ETH.dll")]
private static extern short X28_GetCommunicationStatistics(short sModuleID, ref
             X28_COMMUNICATION_STATISTICS tInfo);
```

**Arguments:**

| Argument | Data type | Description |
| --- | --- | --- |
| sModuleID | short | Identifier of module |
| pInfo | X28_COMMUNICATION_STATISTICS* | Pointer to a X28_COMMUNICATION_STATISTICS structure, to place returned values in. |

**Return Value:** short

**X28_OK**: if the function succeeds.

**X28_FAIL**: if the function fails.

## 5.12 Auto calibration functions

### 5.12.1 F28 functions

#### *5.12.1.1 F28_GetEOCOffset*

This function allows reading the end of cycle for the offset calculation.

**Function call:**

**C++:**

```
UCHAR F28API F28_GetEOCOffset(short sModuleID);
```

**Visual Basic (Vb.Net):**

```
Public Declare Function F28_GetEOCOffset Lib "F28LightControl_ETH.dll"_ (ByVal
              sModuleID As Short) As Byte
```

**C#.Net:**

```
[DllImport(strDllName)]
private static extern byte F28_GetEOCOffset(short sModuleID);
```

**Arguments:**

| Argument | Data type | Description |
| --- | --- | --- |
| sModuleID | short | Identifier of module |

**Return Value:** short

**0**: Cycle in progress.

**1**: End of cycle

#### *5.12.1.2 F28_GetEOCVolume*

This function allows reading the end of cycle for the volume measurement.

**Function call:**

**C++:**

```
UCHAR F28API F28_GetEOCVolume(short sModuleID);
```

**Visual Basic (Vb.Net):**

```
Public Declare Function F28_GetEOCVolume Lib "F28LightControl_ETH.dll"_ (ByVal
              sModuleID As Short) As Byte
```

**C#.Net:**

```
[DllImport(strDllName)]
```

```
private static extern byte F28_GetEOCVolume(short sModuleID);
```

**Arguments:**

| Argument | Data type | Description |
| --- | --- | --- |
| sModuleID | short | Identifier of module |

**Return Value:** short

**0**: Cycle in progress.

**1**: End of cycle.

### *5.12.1.3 F28_GetEOCRatio*

This function allows reading the end of cycle for the auto-ratio measurement.

**Function call:**

**C++:**

```
UCHAR F28API F28_GetEOCRatio(short sModuleID);
```

**Visual Basic (Vb.Net):**

```
Public Declare Function F28_ GetEOCRatio Lib "F28LightControl_ETH.dll"_ (ByVal
              sModuleID As Short) As Byte
```

**C#.Net:**

```
[DllImport(strDllName)]
private static extern byte F28_GetEOCRatio(short sModuleID);
```

**Arguments:**

| Argument | Data type | Description |
| --- | --- | --- |
| sModuleID | short | Identifier of module |

**Return Value:** short

**0**: Cycle in progress.

**1**: End of cycle.

### *5.12.1.4 F28_GetEOCEasyAutoLearning*

This function allows reading the end of cycle for the easy auto learning measurement.

**Function call:**

**C++:**

```
UCHAR F28API F28_GetEOCEasyAutoLearning(short sModuleID);
```

**Visual Basic (Vb.Net):**

```
Public Declare Function F28_ GetEOCEasyAutoLearning Lib "F28LightControl_ETH.dll"_
              (ByVal sModuleID As Short) As Byte
```

**C#.Net:**

```
[DllImport(strDllName)]
private static extern byte F28_GetEOCEasyAutoLearning(short sModuleID);
```

**Arguments:**

| Argument | Data type | Description |
| --- | --- | --- |
| sModuleID | short | Identifier of module |

**Return Value:** short

**0**: Cycle in progress.

**1**: End of cycle.

### *5.12.1.5 F28_StartAutoCalOffsetOnly*

This function allows calculating the offset of the measurement only.

**Function call:**

**C++:**

```
short F28API F28_StartAutoCalOffsetOnly(short sModuleID, WORD wNbCycles, WORD
             wInterCycleTime, float fOffsetMax);
```

**Visual Basic (Vb.Net):**

```
Public Declare Function F28_StartAutoCalOffsetOnly Lib _ "F28LightControl_ETH.dll"
             (ByVal sModuleID As Short, ByVal _ wNbCycles As UShort, ByVal
             wInterCycleTime As UShort, ByVal _ fOffsetMax As Single) As Short
```

**C#.Net:**

```
[DllImport(strDllName)]
private static extern short F28_StartAutoCalOffsetOnly(short sModuleID, ushort
             wNbCycles, ushort wInterCycleTime, float fOffsetMax);
```

**Arguments:**

| Argument | Data type | Description |
| --- | --- | --- |
| sModuleID | short | Identifier of module |
| wNbCycles | WORD | Number of cycles of offset calculation |
| wInterCycleTime | WORD | Time between each offset cycle (ms) |
| fOffsetMax | float | Maximum reject for the calculated offset (sccm) |

**Return Value:** short

**X28_OK**: if the function succeeds.

**X28_FAIL**: if the function fails.

***Important**! For complete calibration, the offset calibration must be carried on and succeed in first step and then the volume measurement succeed in second step.*

### 5.12.1.6 F28_StartAutoCalOffset (first step)

This function allows calculating the offset of the measurement; it is the first step of volume and offset calculation.

**Function call:**

**C++:**

```
short F28API F28_StartAutoCalOffset(short sModuleID, WORD wNbCycles, WORD
             wInterCycleTime, float fOffsetMax);
```

**Visual Basic (Vb.Net):**

```
Public Declare Function F28_StartAutoCalOffset Lib "F28LightControl_ETH.dll"_ (ByVal
             sModuleID As Short, ByVal wNbCycles As UShort, ByVal _ wInterCycleTime As
             UShort, ByVal fOffsetMax As Single) As Short
```

**C#.Net:**

```
[DllImport(strDllName)]
private static extern short F28_StartAutoCalOffset(short sModuleID, ushort wNbCycles,
             ushort wInterCycleTime, float fOffsetMax);
```

**Arguments:**

| Argument | Data type | Description |
| --- | --- | --- |
| sModuleID | short | Identifier of module |
| wNbCycles | WORD | Number of cycles of offset calculation |
| wInterCycleTime | WORD | Time between each offset cycle |
| fOffsetMax | float | Maximum reject for the calculated offset |

**Return Value:** short

**X28_OK**: if the function succeeds.

**X28_FAIL**: if the function fails.

***Nota**: if this function succeeds, the second step (volume measurement) is required to complete the calibration.*

### 5.12.1.7 F28_StartAutoCalVolume (second step)

This function allows measuring the volume of the installation; it is the second step of volume and offset calculation.

**Function call:**

**C++:**

```
short F28API F28_StartAutoCalVolume(short sModuleID, WORD wNbCycles, WORD
             wInterCycleTime, float fLeak, float fPressure, float fVolMin, float
             fVolMax);
```

**Visual Basic (Vb.Net):**

```
Public Declare Function F28_StartAutoCalVolume Lib "F28LightControl_ETH.dll"_ (ByVal
              sModuleID As Short, ByVal wNbCycles As UShort, ByVal _ wInterCycleTime As
              UShort, ByVal fLeak As Single, ByVal _ fPressure As Single, ByVal fVolMin
              As Single, ByVal fVolMax _
              As Single) As Short
```

**C#.Net:**

```
[DllImport(strDllName)]
private static extern short F28_StartAutoCalVolume(short sModuleID, ushort wNbCycles,
              ushort wInterCycleTime, float fLeak, float fPressure, float fVolMin,
              float fVolMax);
```

**Arguments:**

| Argument | Data type | Description |
| --- | --- | --- |
| sModuleID | short | Identifier of module |
| wNbCycles | WORD | Number of cycles of offset calculation |
| wInterCycleTime | WORD | Time between each offset cycle (ms) |
| fLeak | float | Value of the master leak (sccm) |
| fPressure | float | Pressure value of the master leak (bar) |
| fVolMin | float | Maximum reject for the measured volume ($cm^3$) |
| fVolMax | float | Minimum reject for the measured volume ($cm^3$) |

**Return Value:** short

**X28_OK**: if the function succeeds.

**X28_FAIL**: if the function fails.

***Nota**: if this second function succeeds, the calibration is complete.*

### *5.12.1.8 F28_StartAutoRatio*

The function start an auto-ratio special cycle of the defined module (for sealed component test).

**Function call:**

**C++:**

```
short F28API F28_StartAutoRatio(short sModuleID, WORD wNbCycles, WORD
wInterCycleTime, float  fRatioMax, float fRatioMin);
```

**Visual Basic (Vb.net)**

```
Public Declare Function F28_StartAutoRatio Lib _ "F28LightControl_Eth.dll" (ByVal
sModuleID As Short, ByVal wNbCycles As UShort, ByVal _ wInterCycleTime As UShort,
ByVal fRatioMax As Single, ByVal fRatioMin As Single) As Short
```

**C#.Net**

```
[dllImport("F28LightControl_ETH.dll")]
```

```
private static extern short F28_StartAutoRatio(short sModuleID, ushort wNbCycles,
ushort wInterCycleTime, float fRatioMax, float fRatioMin) ;
```

**Argument :**

| Argument | Data type | Description |
| --- | --- | --- |
| sModuleID | short | Identifier of module |
| wNbCycles | WORD | Number of cycles of auto-ratio cycle |
| wInterCycleTime | WORD | Time between each auto-ratio cycle (ms) |
| fRatioMax | float | Maximum ratio (%) |
| fRatioMin | float | Minimum ratio (%) |

**Return Value:** short

**X28_OK**: if the function succeeds.

**X28_FAIL**: if the function fails.

#### *5.12.1.9 F28_StartEasyAutoLearning*

The function start an easy auto learning special cycle of the defined module (for fill mode easy auto).

**Function call:**

**C++:**

```
short F28API F28_StartEasyAutoLearning(short sModuleID, WORD wNbCycles, WORD
wInterCycleTime);
```

**Visual Basic (Vb.net)**

```
Public Declare Function F28_StartEasyAutoLearning Lib _ "F28LightControl_Eth.dll"
(ByVal sModuleID As Short, ByVal wNbCycles As UShort, ByVal _ wInterCycleTime As
UShort) As Short
```

**C#.Net**

```
[dllImport("F28LightControl_ETH.dll")]
private static extern short F28_StartEasyAutoLearning(short sModuleID, ushort
wNbCycles, ushort wInterCycleTime) ;
```

**Argument :**

| Argument | Data type | Description |
| --- | --- | --- |
| sModuleID | short | Identifier of module |
| wNbCycles | WORD | Number of cycles of auto-ratio cycle |
| wInterCycleTime | WORD | Time between each auto-ratio cycle (ms) |

**Return Value:** short

**X28_OK**: if the function succeeds.

**X28_FAIL**: if the function fails.

### *5.12.1.10 F28_StopAutoCal*

This function allows aborting any auto calibration cycles. This function must be called to abort a current calibration process.

**Function call:**

**C++:**

```
short F28API F28_StopAutoCal(short sModuleID);
```

**Visual Basic (Vb.Net):**

```
Public Declare Function F28_StopAutoCal Lib "F28LightControl_ETH.dll" _
              (ByVal sModuleID As Short) As Short
```

**C#.Net:**

```
[DllImport(strDllName)]
private static extern short F28_StopAutoCal(short sModuleID);
```

**Arguments:**

| Argument | Data type | Description |
| --- | --- | --- |
| sModuleID | short | Identifier of module |

**Return Value:** short

**X28_OK**: if the function succeeds.

**X28_FAIL**: if the function fails.

### *5.12.1.11 F28_StopAutoRatio*

This function allows aborting any auto-ratio cycles. This function must be called to abort a current auto-ratio process.

**Function call:**

**C++:**

```
short F28API F28_StopAutoRatio(short sModuleID);
```

**Visual Basic (Vb.Net):**

```
Public Declare Function F28_StopAutoRatio Lib "F28LightControl_ETH.dll" _
              (ByVal sModuleID As Short) As Short
```

**C#.Net:**

```
[DllImport(strDllName)]
private static extern short F28_StopAutoRatio(short sModuleID);
```

**Arguments:**

| Argument | Data type | Description |
| --- | --- | --- |
| sModuleID | short | Identifier of module |

**Return Value:** short

**X28_OK**: if the function succeeds.

**X28_FAIL:** if the function fails.

#### *5.12.1.12 F28_StopEasyAutoLearning*

This function allows aborting any easy auto learning cycles. This function must be called to abort a current easy auto learning process.

**Function call:**

**C++:**

```
short F28API F28_StopEasyAutoLearning(short sModuleID);
```

**Visual Basic (Vb.Net):**

```
Public Declare Function F28_StopEasyAutoLearning Lib "F28LightControl_ETH.dll" _
              (ByVal sModuleID As Short) As Short
```

**C#.Net:**

```
[DllImport(strDllName)]
private static extern short F28_StopEasyAutoLearning(short sModuleID);
```

**Arguments:**

| Argument | Data type | Description |
| --- | --- | --- |
| sModuleID | short | Identifier of module |

**Return Value:** short

**X28_OK**: if the function succeeds.

**X28_FAIL:** if the function fails.

#### *5.12.1.13 F28_GetAutoCalAlarm*

This function allows reading the if an alarm has been triggered during the calibration cycles.

**Function call:**

**C++:**

```
UCHAR F28API F28_GetAutoCalAlarm(short sModuleID);
```

**Visual Basic (Vb.Net):**

```
Public Declare Function F28_GetAutoCalAlarm Lib "F28LightControl_ETH.dll"_ (ByVal
              sModuleID As Short) As Byte
```

**C#.Net:**

```
[DllImport(strDllName)]
private static extern byte F28_GetAutoCalAlarm(short sModuleID);
```

**Arguments:**

| Argument | Data type | Description |
| --- | --- | --- |
| sModuleID | short | Identifier of module |

**Return Value:** short

**= 0**: no Alarm.

**≠ 0**: Alarm triggered.

## 5.13 How to run calibration functions

### 5.13.1 F28

#### *5.13.1.1 Offset calculation only*

**1)** Start Offset calculation:

// Use: F28_StartAutoCalOffsetOnly(m_hDevice, m_wNbCycles, m_wInterCycle);

**2)** Wait End of Cycle of Offset calculation:

// Use: While (!F28_GetEOCOffset(m_hDevice))

**3)** Read Auto Calibration alarm:

// Use: F28_GetAutoCalAlarm( m_hDevice)

**4)** If no alarm, read and save parameters:

// Use: F28_GetParameters(m_DeviceInfo.hHandle, &tPara) == X28_OK)

#### *5.13.1.2 Volume and offset calculation*

**1)** State Offset calculation:

// Use: F28_StartAutoCalOffset(m_hDevice, m_wNbCycles, m_wInterCycle);

**2)** Wait End of Cycle of Offset calculation:

// Use: While (!F28_GetEOCOffset(m_hDevice))

**3)** Wait Master leak:

// Ask user to plug master leak

**4)** if (Wait master leak Ok) start volume calculation:

// Use: F28_StartAutoCalVolume(m_hDevice, m_wNbCycles, m_wInterCycle,
m_fVolumeLeak, m_fVolumePressure,
m_fVolumeMin, m_fVolumeMax)

**5)** Wait End of Cycle of Volume calculation:

// Use: (!F28_GetEOCVolume(m_hDevice))

**6)** Read Auto Calibration alarm:

// Use: F28_GetAutoCalAlarm( m_hDevice)

**7)** If no alarm, read and save parameters:

// Use: F28_GetParameters((m_hDevice, &tPara) == X28_OK)

## 5.14 Calibration code sample

### 5.14.1 F28

#### *5.14.1.1 Start calibration (first step)*

```
' ****************************************************
' Start Calibration
' ****************************************************
Public Function StartCal(sModuleID As Short, ucMode As Byte, _
wNbCycles As UShort, _
wInterCycle As UShort, _
fOffsetMax As Single, _
Optional fVolumeLeak As Single = 0, _
Optional fVolumePressure As Single = 0, _
Optional fVolumeMin As Single = 0, _
Optional fVolumeMax As Single = 0) As Boolean
Dim bRet As Boolean
bRet = False
If sModuleID > 0 Then
       m_wNbCycles = wNbCycles
       m_wInterCycle = wInterCycle
       m_fOffsetMax = fOffsetMax
       m_fVolumeLeak = fVolumeLeak
       m_fVolumePressure = fVolumePressure
       m_fVolumeMin = fVolumeMin
       m_fVolumeMax = fVolumeMax
       m_sModuleId = sModuleID
       m_ucMode = ucMode
       m_wError = 0
       m_ucPhase = CAL_AUTO_PHASES.AUTO_START_OFFSET ' CAL_AUTO_PHASES.AUTO_INIT
       m_bRunning = True
       bRet = True
End If
Return bRet
End Function
```

#### *5.14.1.2 Abort calibration*

```
' ****************************************************
' Stop Calibration
' ****************************************************
Public Function StopCal() As Short
Dim sRet As Short
sRet = F28_RETURN.F28_FAIL
If (m_sModuleId > 0) And (m_ucPhase <> CAL_AUTO_PHASES.AUTO_IDDLE) Then
```

```
        sRet = F28_StopAutoCal(m_sModuleId)
End If
m_ucPhase = CAL_AUTO_PHASES.AUTO_IDDLE
m_bRunning = False
Return sRet
End Function
```

#### 5.14.1.3 Continue calibration (second step)

```
' ****************************************************
' Continue calibration after validation -> Run Volume Calibration
' ****************************************************
Public Sub RunContinue(bForward As Boolean)
If (bForward = True) Then
        m_ucPhase = CAL_AUTO_PHASES.AUTO_START_VOLUME
        Else
        m_wError = m_ucPhase
        m_ucPhase = CAL_AUTO_PHASES.AUTO_IDDLE
        m_bRunning = False
        StopCal()
End If
End Sub
```

#### 5.14.1.4 Running calibration process

```
' ****************************************************
' Purpose : Run Calibration
' Return :
' - True : EOC calibration
' - False : Running
' ****************************************************
Public Function RunCal() As Boolean
Dim sRet As Short
Dim bReturn As Boolean
' Not End of Run
bReturn = False
Select Case m_ucPhase
Case CAL_AUTO_PHASES.AUTO_START_OFFSET ' Start auto Cal
        If (m_ucMode = MODE_AUTO_CAL.OFFSET) Then
                sRet = F28_StartAutoCalOffsetOnly(m_sModuleId, m_wNbCycles, m_wInterCycle, m_fOffsetMax)
                Else
                sRet = F28_StartAutoCalOffset(m_sModuleId, m_wNbCycles, m_wInterCycle, m_fOffsetMax)
        End If
        If (sRet = F28_RETURN.F28_OK) Then
                m_ucPhase = CAL_AUTO_PHASES.AUTO_WAIT_EOC_OFFSET
        Else
                m_wError = m_ucPhase
                m_ucPhase = CAL_AUTO_PHASES.AUTO_END
        End If
Case CAL_AUTO_PHASES.AUTO_WAIT_EOC_OFFSET ' Wait EOC Offset
        If (F28_GetEOCOffset(m_sModuleId) > 0) Then
                If (m_ucMode = MODE_AUTO_CAL.OFFSET) Then
                        m_wError = 0 ' Pas d'erreur
                        m_ucPhase = CAL_AUTO_PHASES.AUTO_END
                Else
                        m_wError = m_ucPhase
                        m_ucPhase = CAL_AUTO_PHASES.AUTO_WAIT_MASTER_LEAK
                End If
        End If
Case CAL_AUTO_PHASES.AUTO_WAIT_MASTER_LEAK ' Waiting master leak
```

```
        ' Wait validation from user
        ' Do nothing
Case CAL_AUTO_PHASES.AUTO_START_VOLUME ' Start auto volume
        If (F28_StartAutoCalVolume(m_sModuleId, _
                m_wNbCycles, _
                m_wInterCycle, _
                m_fVolumeLeak, _
                m_fVolumePressure, _
                m_fVolumeMin, _
                m_fVolumeMax) = F28_RETURN.F28_OK) Then
                m_ucPhase = CAL_AUTO_PHASES.AUTO_WAIT_EOC_VOLUME
        Else
                m_wError = m_ucPhase
                m_ucPhase = CAL_AUTO_PHASES.AUTO_END
        End If
Case CAL_AUTO_PHASES.AUTO_WAIT_EOC_VOLUME ' Wait EOC Auto volume
        If (F28_GetEOCVolume(m_sModuleId) > 0) Then
                m_wError = 0 ' Pas d'erreur
                m_ucPhase = CAL_AUTO_PHASES.AUTO_END
        End If
Case CAL_AUTO_PHASES.AUTO_END ' End of auto calibration
        m_wError = m_ucPhase
        m_ucPhase = CAL_AUTO_PHASES.AUTO_IDDLE
        m_bRunning = False
        bReturn = True
Case CAL_AUTO_PHASES.AUTO_IDDLE ' Ready do nothing
        ' do nothing
End Select
Return bReturn
End Function
```

## 5.15 How to run calibration functions for 5 devices

### 5.15.1 F28

#### *5.15.1.1 We have 5 devices*

The device ID's are in the:

```
arrayID(5) = {sModuleID1, sModuleID2, sModuleID3, sModuleID4, sModuleID5}
```

#### *5.15.1.2 Offset calculation only*

**1)** Start Offset Calculation for 5 devices.

**1.1)** Repeat `F28_StartAutoCalOffsetOnly` for each unit,

For Id = 0 to 4,

```
F28_StartAutoCalOffsetOnly(arrayID[i], m_wNbCycles, m_wInterCycle);
```

Next.

**2)** Wait EOC of Offset for 5 devices.

Do:

**2.1)** Read Real time for each unit,

```
    F28_GetRealTimeData(arrayID[i], m_realTime)
```

**2.2)** Display "Real time" for each unit,

**2.3)** If End of cycle, read last result,

```
    F28_GetLastResult(arrayID[i], m_Result)
```

**2.4)** Display last result,

**2.5)** If Number of cycles is not reached, start group (start next cycle for all unit),

```
    F28_StartCycleByGroup(ucGroup)
```

Or,

For Id = 0 to 4,

```
        F28_StartCycle(arrayID[i])
```

Next,

Or,

Wait till intercycle elapsed,

```
While (not F28_GetEOCOffset(arrayID[i]))
```

**3)** Read Auto-calibration alarm for the 5 units, at the end of calibration,

For Id = 0 to 4

Read alarm code

```
    F28_GetAutoCalAlarm(arrayID[i])
```

If no Alarm read and save parameters,

```
    F28_GetParameters(arrayID[i], &tPara)
```

Save parameters

Next

#### *5.15.1.3 Volume & offset calculation*

**1)** Start Offset calculation

**1.1)** Repeat `F28_StartAutoCalOffsetOnly` for each unit

For Id = 0 to 4

```
    F28_StartAutoCalOffsetOnly(arrayID[i], m_wNbCycles, m_wInterCycle);
```

Next

**2)** Wait EOC of Offset for 5 devices,

Do:

**2.1)** Read Real time for each unit,

```
    F28_GetRealTimeData(arrayID[i], m_realTime)
```

**2.2)** Correction leak to Pa/s,

```
    m_realTime.fLeakValue = m_realTime.fLeakValue * 1000
    m_realTime.ucUnitLeak = F28_LEAK_UNITS.LEAK_PASEC
```

**2.3)** Display Real time for each unit,

**2.4)** If end of cycle, last result,

```
F28_GetLastResult(arrayID[i], m_Result)
```

**2.5)** Correction leak to Pa/s

```
m_Result.fLeakValue = m_Result.fLeakValue * 1000
m_Result.ucUnitLeak = F28_LEAK_UNITS.LEAK_PASEC
```

**2.6)** Display Last Result,

**2.7)** If Number of Cycles is not reached, start group (start next cycle for all units),

```
F28_StartCycleByGroup(ucGroup)
```

Or,

For Id = 0 to 4,

```
    F28_StartCycle(arrayID[i])
```

Next,

Or,

Wait till intercycle elapsed,

```
While (not F28_GetEOCOffset(arrayID[i]))
```

**3)** Select/Plug master leak for all devices.

**4)** Start volume Calculation for all devices,

1.1 Repeat F28_StartAutoCalVolume for each unit

For Id = 0 to 4

```
F28_StartAutoCalVolume(arrayID[i], m_wNbCycles, m_wInterCycle,
            m_fVolumeLeak, m_fVolumePressure, m_fVolumeMin, m_fVolumeMax)
```

Next

**5)** Wait EOC of Volume Calibration for 5 devices

Do:

**5.1)** Read Real time for each unit,

```
    F28_GetRealTimeData(arrayID[i], m_realTime)
```

**5.2)** Correction leak to Pa/s,

```
m_realTime.fLeakValue = m_realTime.fLeakValue * 1000
m_realTime.ucUnitLeak = F28_LEAK_UNITS.LEAK_PASEC
```

**5.3)** Display Real time for each unit,

**5.4)** If end of cycle, Read last result,

```
F28_GetLastResult(arrayID[i], m_Result)
```

**5.5)** Correction leak to Pa/s,

```
m_Result.fLeakValue = m_Result.fLeakValue * 1000
m_Result.ucUnitLeak = F28_LEAK_UNITS.LEAK_PASEC
```

5.6 Display last result,

5.7 If Number Of Cycles Not Reached, Start next cycle for all unit,

```
- F28_StartCycleByGroup(ucGroup)
```

Or,

```
For Id = 0 to 4,
    F28_StartCycle(arrayID[i])
Next,
Or,
Wait till intercycle elapsed
While (not F28_GetEOCVolume(arrayID[i]))
```

**6)** Read Auto-calibration alarm for the 5 units, at the end of calibration,

```
For Id = 0 to 4,
    Read alarm code,
        F28_GetAutoCalAlarm(arrayID[i])
    If no alarm, read and save parameters,
        F28_GetParameters(arrayID[i], &tPara)
        Save parameters,
Next.
```

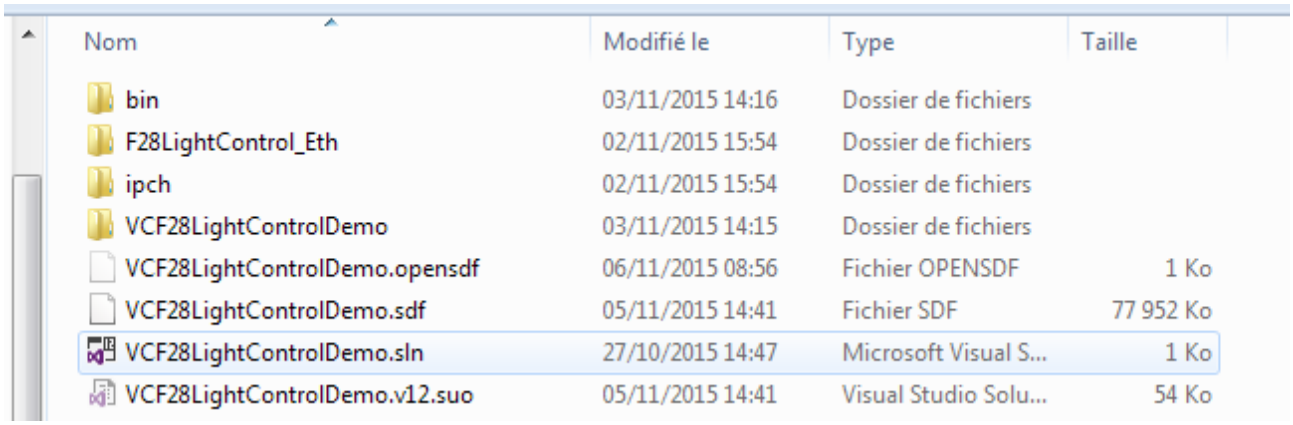
# 6 Appendices 1

## 6.1 What's needed for using the samples project C++/MFC/C#/VB.NET

- Microsoft Visual Studio 2013 Update5 must be installed,
- Microsoft.Net Framework 4.5,
- DLL Ethernet interface: **F28LightControl_ETH.dll** or **F28LightControl_ETH64.dll**

## 6.2 Visual C++/MFC sample

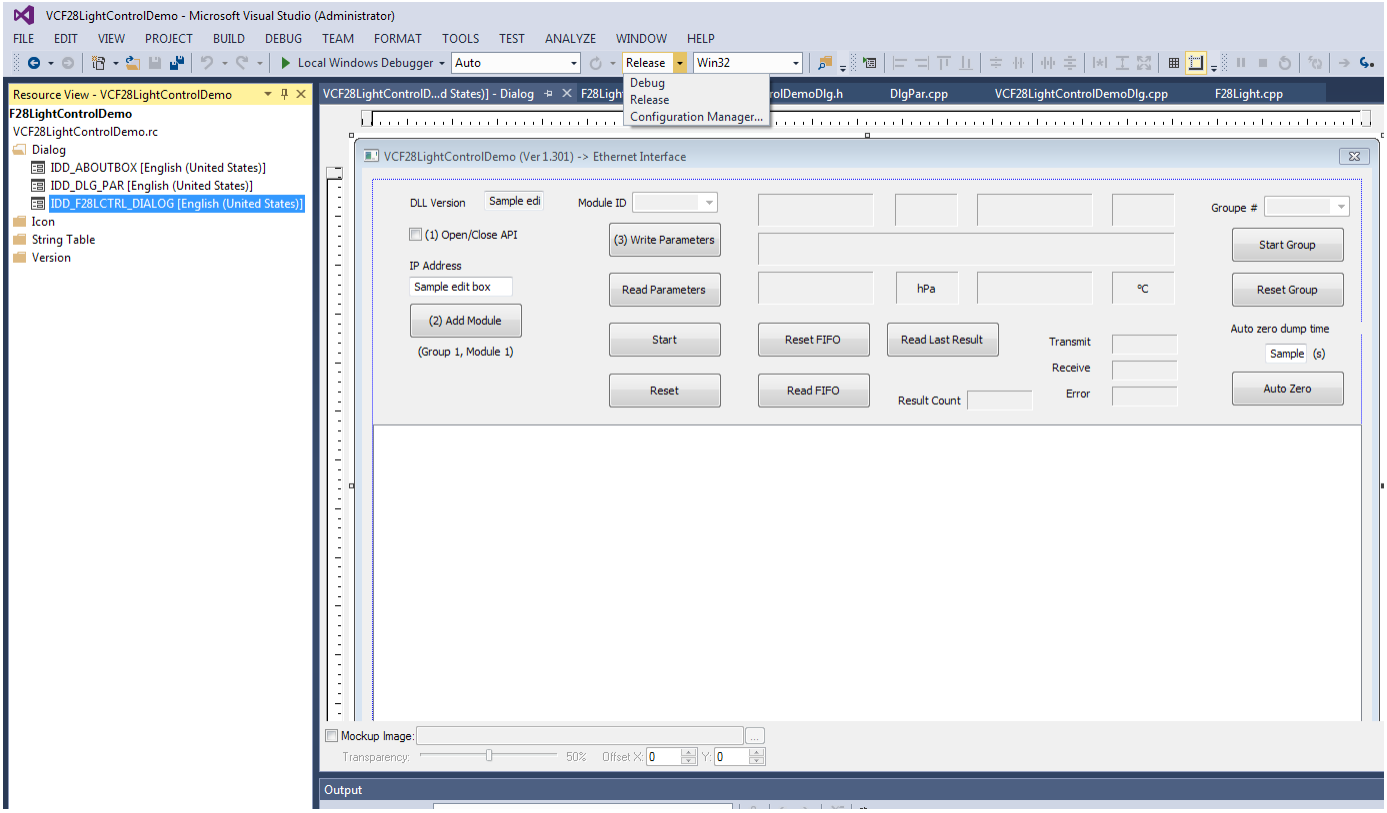
### 6.2.1 Build project

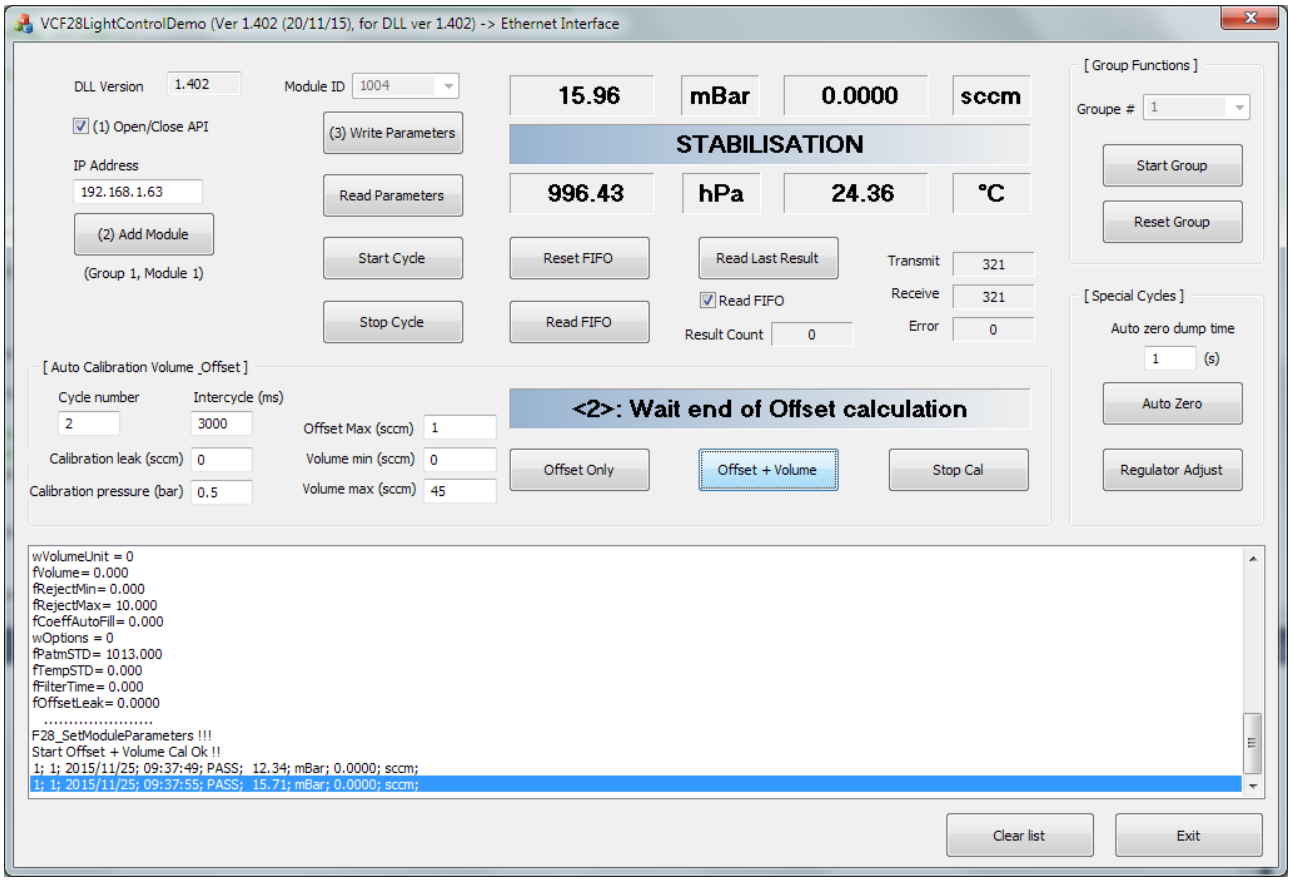


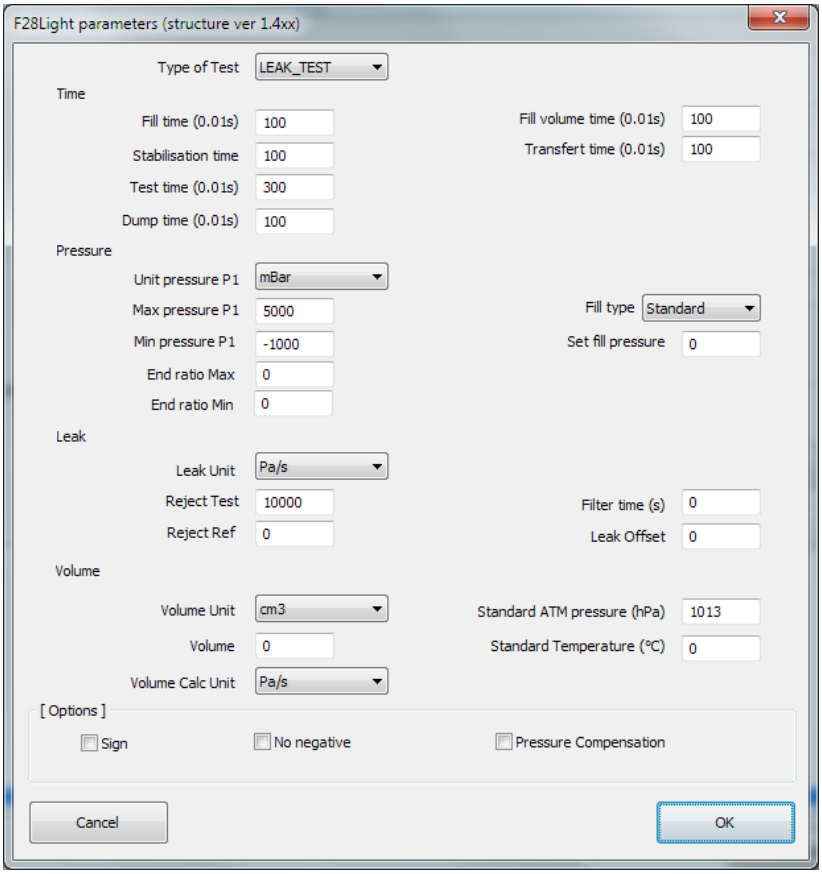
| Nom | Modifié le | Type | Taille |
| --- | --- | --- | --- |
| bin | 03/11/2015 14:16 | Dossier de fichiers | |
| F28LightControl_Eth | 02/11/2015 15:54 | Dossier de fichiers | |
| ipch | 02/11/2015 15:54 | Dossier de fichiers | |
| VCF28LightControlDemo | 03/11/2015 14:15 | Dossier de fichiers | |
| VCF28LightControlDemo.opensdf | 06/11/2015 08:56 | Fichier OPENSDF | 1 Ko |
| VCF28LightControlDemo.sdf | 05/11/2015 14:41 | Fichier SDF | 77 952 Ko |
| VCF28LightControlDemo.sln | 27/10/2015 14:47 | Microsoft Visual S... | 1 Ko |
| VCF28LightControlDemo.v12.suo | 05/11/2015 14:41 | Visual Studio Solu... | 54 Ko |

Release → for Release,

Debug → for debug.

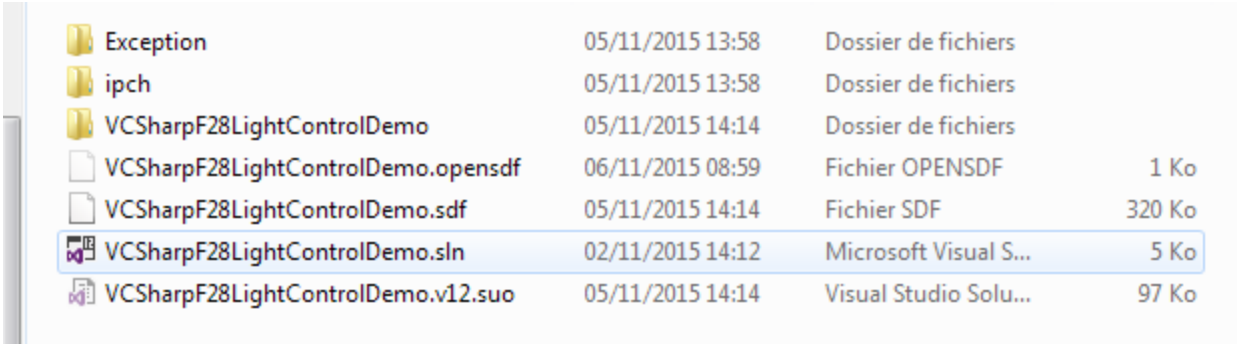

VCF28LightControlDemo - Microsoft Visual Studio (Administrator)
FILE EDIT VIEW PROJECT BUILD DEBUG TEAM FORMAT TOOLS TEST ANALYZE WINDOW HELP
Local Windows Debugger
Auto
Release
Win32
Debug
Release
Configuration Manager...
Resource View - VCF28LightControlDemo
F28LightControlDemo
VCF28LightControlDemo.rc
Dialog
IDD_ABOUTBOX [English (United States)]
IDD_DLG_PAR [English (United States)]
IDD_F28LCTRL_DIALOG [English (United States)]
Icon
String Table
Version
DlgPar.cpp
VCF28LightControlDemoDlg.cpp
F28Light.cpp
VCF28LightControlDemo (Ver 1.301) -> Ethernet Interface
DLL Version
Sample edi
Module ID
(1) Open/Close API
(3) Write Parameters
Groupe #
Start Group
IP Address
Sample edit box
Read Parameters
hPa
°C
Reset Group
(2) Add Module
(Group 1, Module 1)
Start
Reset FIFO
Read Last Result
Transmit
Receive
Error
Auto zero dump time
Sample
(s)
Reset
Read FIFO
Result Count
Auto Zero
Mockup Image:
Transparency:
50%
Offset X: 0
Y: 0
Output


VCF28LightControlDemo (Ver 1.402 (20/11/15), for DLL ver 1.402) -> Ethernet Interface
DLL Version
1.402
Module ID
1004
15.96
mBar
0.0000
sccm
[ Group Functions ]
(1) Open/Close API
(3) Write Parameters
STABILISATION
Groupe #
1
Start Group
IP Address
192.168.1.63
Read Parameters
996.43
hPa
24.36
°C
Reset Group
(2) Add Module
(Group 1, Module 1)
Start Cycle
Reset FIFO
Read Last Result
Transmit
321
Receive
321
Read FIFO
Stop Cycle
Read FIFO
Result Count
0
Error
0
[ Special Cycles ]
Auto zero dump time
1
(s)
[ Auto Calibration Volume _Offset ]
Cycle number
2
Intercycle (ms)
3000
Offset Max (sccm)
1
<2>: Wait end of Offset calculation
Auto Zero
Calibration leak (sccm)
0
Volume min (sccm)
0
Offset Only
Offset + Volume
Stop Cal
Regulator Adjust
Calibration pressure (bar)
0.5
Volume max (sccm)
45
wVolumeUnit = 0
fVolume= 0.000
fRejectMin= 0.000
fRejectMax= 10.000
fCoeffAutoFill= 0.000
wOptions = 0
fPatmSTD= 1013.000
fTempSTD= 0.000
fFilterTime= 0.000
fOffsetLeak= 0.0000
F28_SetModuleParameters !!!
Start Offset + Volume Cal Ok !!
1; 1; 2015/11/25; 09:37:49; PASS; 12.34; mBar; 0.0000; sccm;
1; 1; 2015/11/25; 09:37:55; PASS; 15.71; mBar; 0.0000; sccm;
Clear list
Exit




## 6.3 Visual C# sample

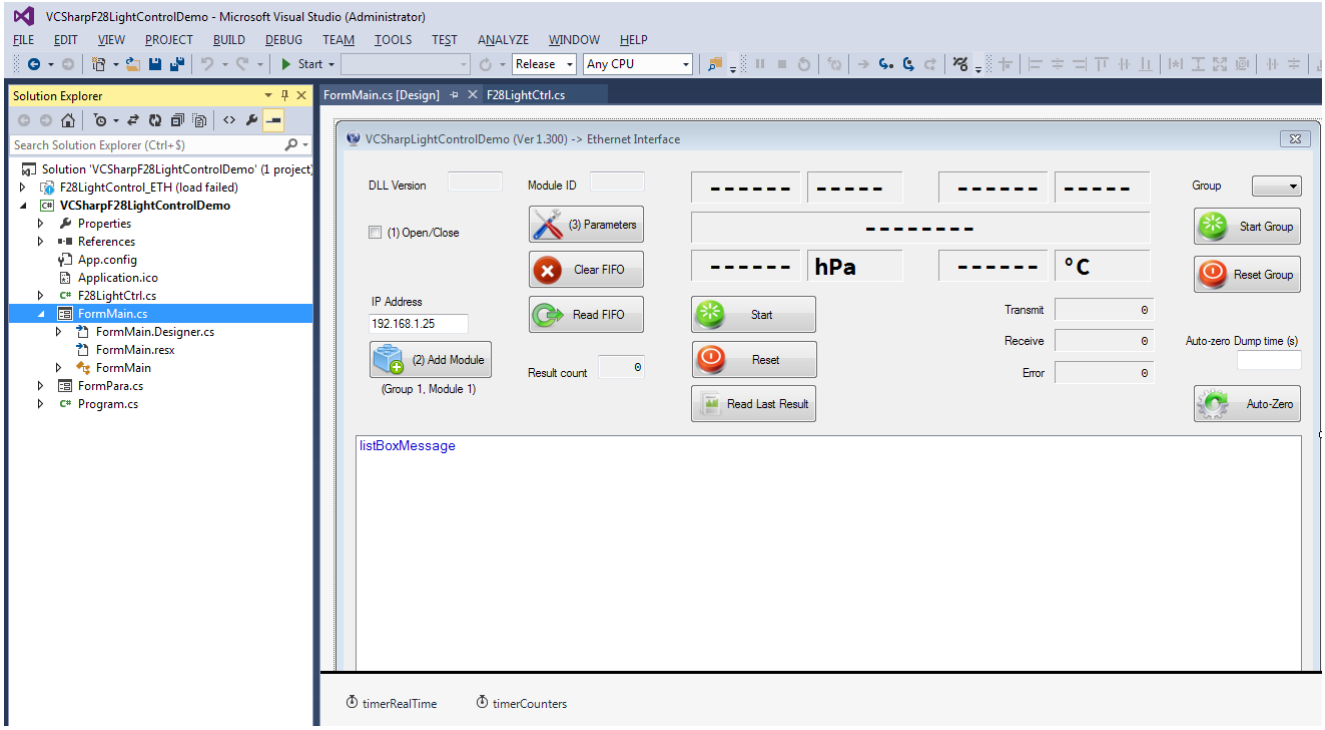
### 6.3.1 Build project

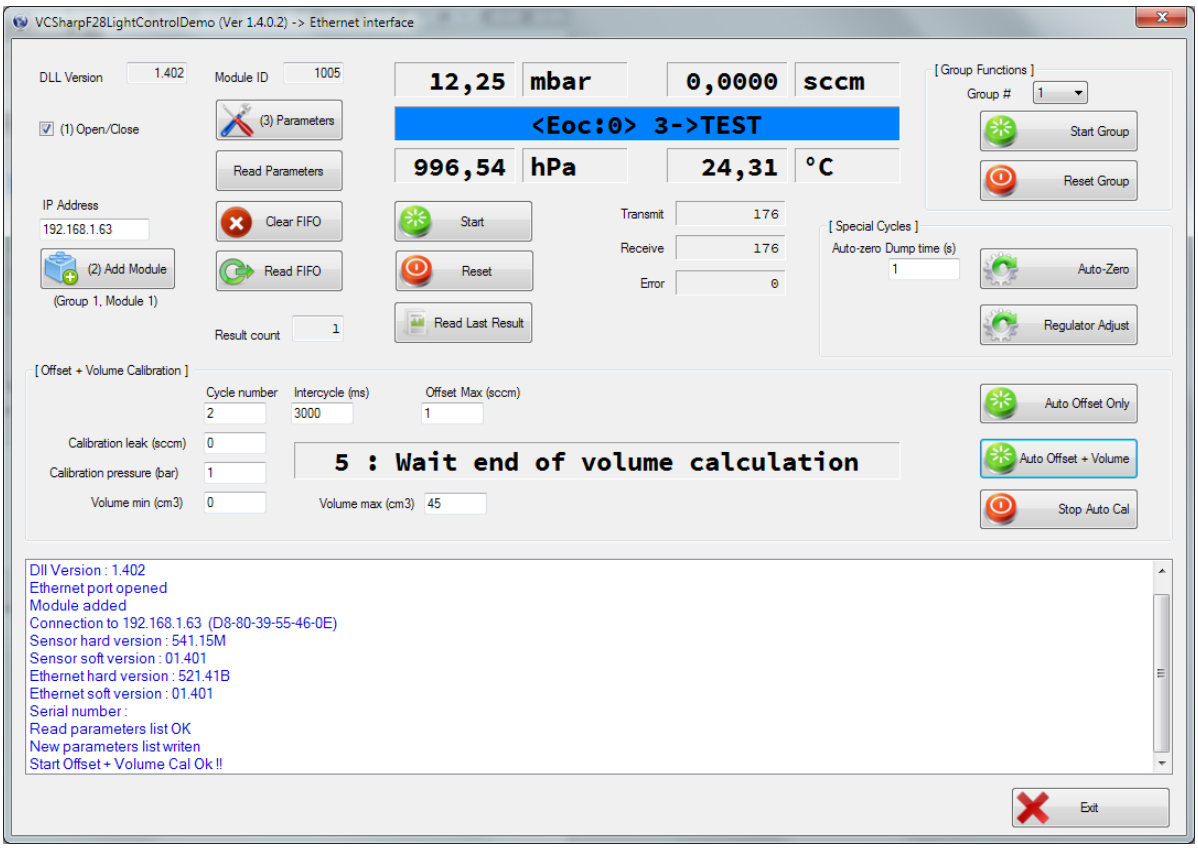


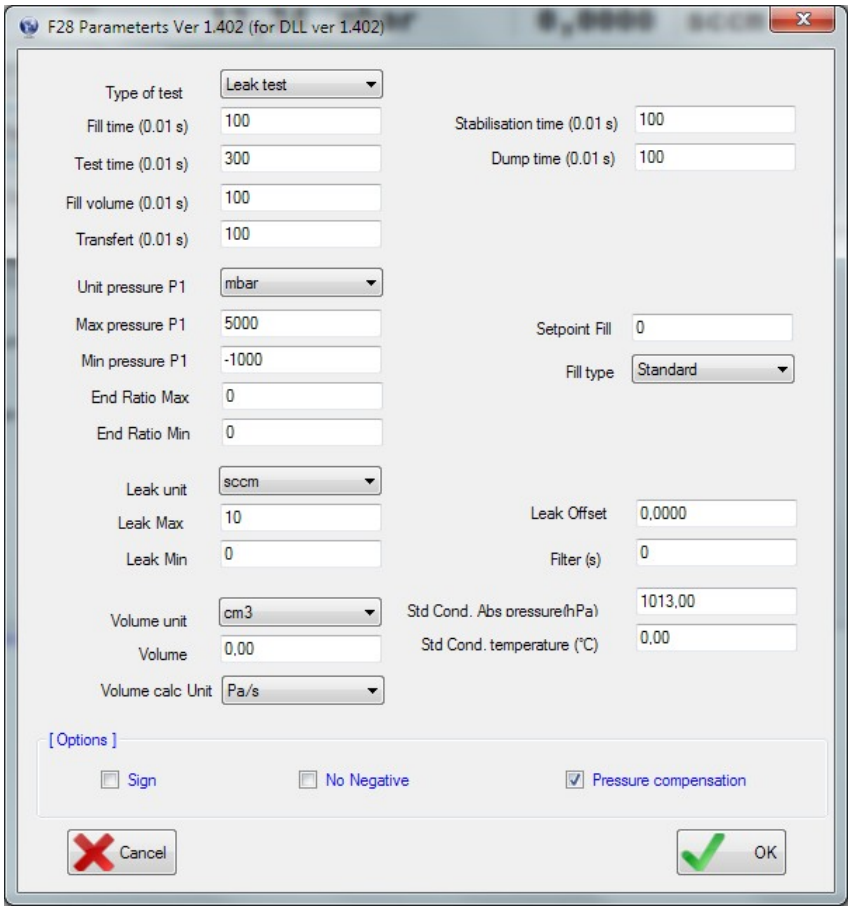
| | | | |
| --- | --- | --- | --- |
| Exception | 05/11/2015 13:58 | Dossier de fichiers | |
| ipch | 05/11/2015 13:58 | Dossier de fichiers | |
| VCSharpF28LightControlDemo | 05/11/2015 14:14 | Dossier de fichiers | |
| VCSharpF28LightControlDemo.opensdf | 06/11/2015 08:59 | Fichier OPENSDF | 1 Ko |
| VCSharpF28LightControlDemo.sdf | 05/11/2015 14:14 | Fichier SDF | 320 Ko |
| VCSharpF28LightControlDemo.sln | 02/11/2015 14:12 | Microsoft Visual S... | 5 Ko |
| VCSharpF28LightControlDemo.v12.suo | 05/11/2015 14:14 | Visual Studio Solu... | 97 Ko |

Release → for Release,

Debug → for debug.

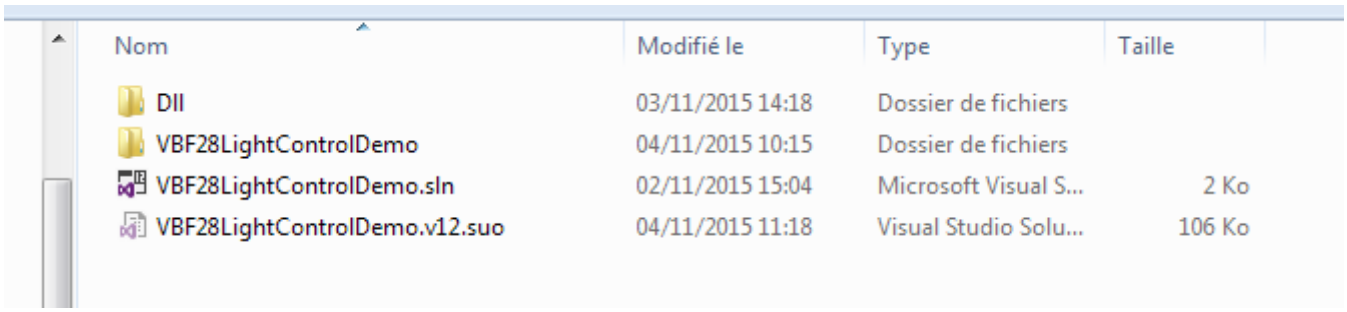

VCSharpF28LightControlDemo - Microsoft Visual Studio (Administrator)
FILE EDIT VIEW PROJECT BUILD DEBUG TEAM TOOLS TEST ANALYZE WINDOW HELP
Solution Explorer
Search Solution Explorer (Ctrl+$)
Solution 'VCSharpF28LightControlDemo' (1 project)
F28LightControl_ETH (load failed)
VCSharpF28LightControlDemo
Properties
References
App.config
Application.ico
F28LightCtrl.cs
FormMain.cs
FormMain.Designer.cs
FormMain.resx
FormMain
FormPara.cs
Program.cs
FormMain.cs [Design]
F28LightCtrl.cs
VCSharpLightControlDemo (Ver 1.300) -> Ethernet Interface
DLL Version
Module ID
(1) Open/Close
(3) Parameters
Clear FIFO
Read FIFO
IP Address
192.168.1.25
(2) Add Module
(Group 1, Module 1)
Result count
Start
Reset
Read Last Result
hPa
°C
Transmit
Receive
Error
Group
Start Group
Reset Group
Auto-zero Dump time (s)
Auto-Zero
listBoxMessage
timerRealTime
timerCounters


VCSharpF28LightControlDemo (Ver 1.4.0.2) -> Ethernet interface
DLL Version 1.402
Module ID 1005
(1) Open/Close
(3) Parameters
Read Parameters
IP Address
192.168.1.63
(2) Add Module
(Group 1, Module 1)
Clear FIFO
Read FIFO
Result count 1
12,25 mbar
0,0000 sccm
<Eoc:0> 3->TEST
996,54 hPa
24,31 °C
Start
Reset
Read Last Result
Transmit 176
Receive 176
Error 0
[ Group Functions ]
Group # 1
Start Group
Reset Group
[ Special Cycles ]
Auto-zero Dump time (s) 1
Auto-Zero
Regulator Adjust
[ Offset + Volume Calibration ]
Cycle number 2
Intercycle (ms) 3000
Offset Max (sccm) 1
Calibration leak (sccm) 0
Calibration pressure (bar) 1
Volume min (cm3) 0
Volume max (cm3) 45
5 : Wait end of volume calculation
Auto Offset Only
Auto Offset + Volume
Stop Auto Cal
Dll Version : 1.402
Ethernet port opened
Module added
Connection to 192.168.1.63 (D8-80-39-55-46-0E)
Sensor hard version : 541.15M
Sensor soft version : 01.401
Ethernet hard version : 521.41B
Ethernet soft version : 01.401
Serial number :
Read parameters list OK
New parameters list writen
Start Offset + Volume Cal Ok !!
Exit




## 6.4 Visual Basic.NET sample

### 6.4.1 Build project

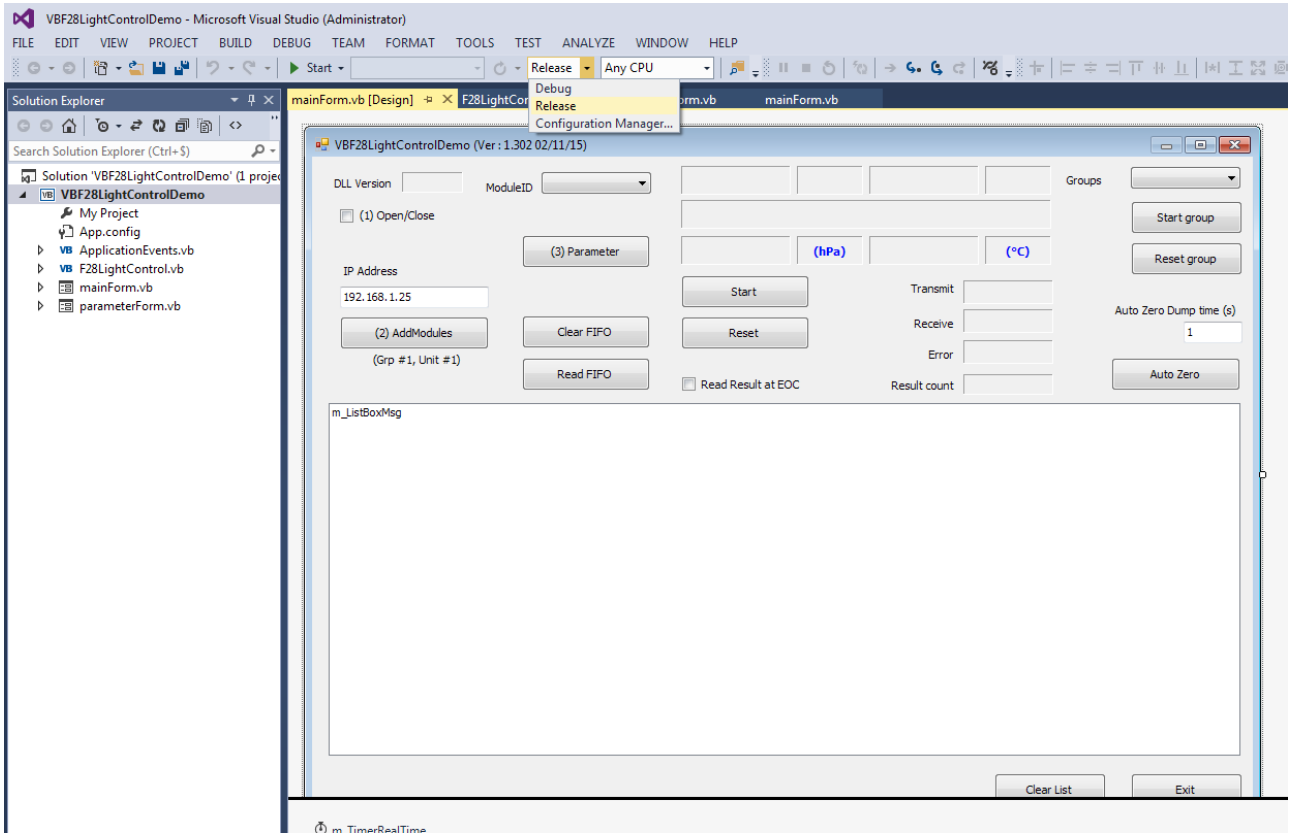


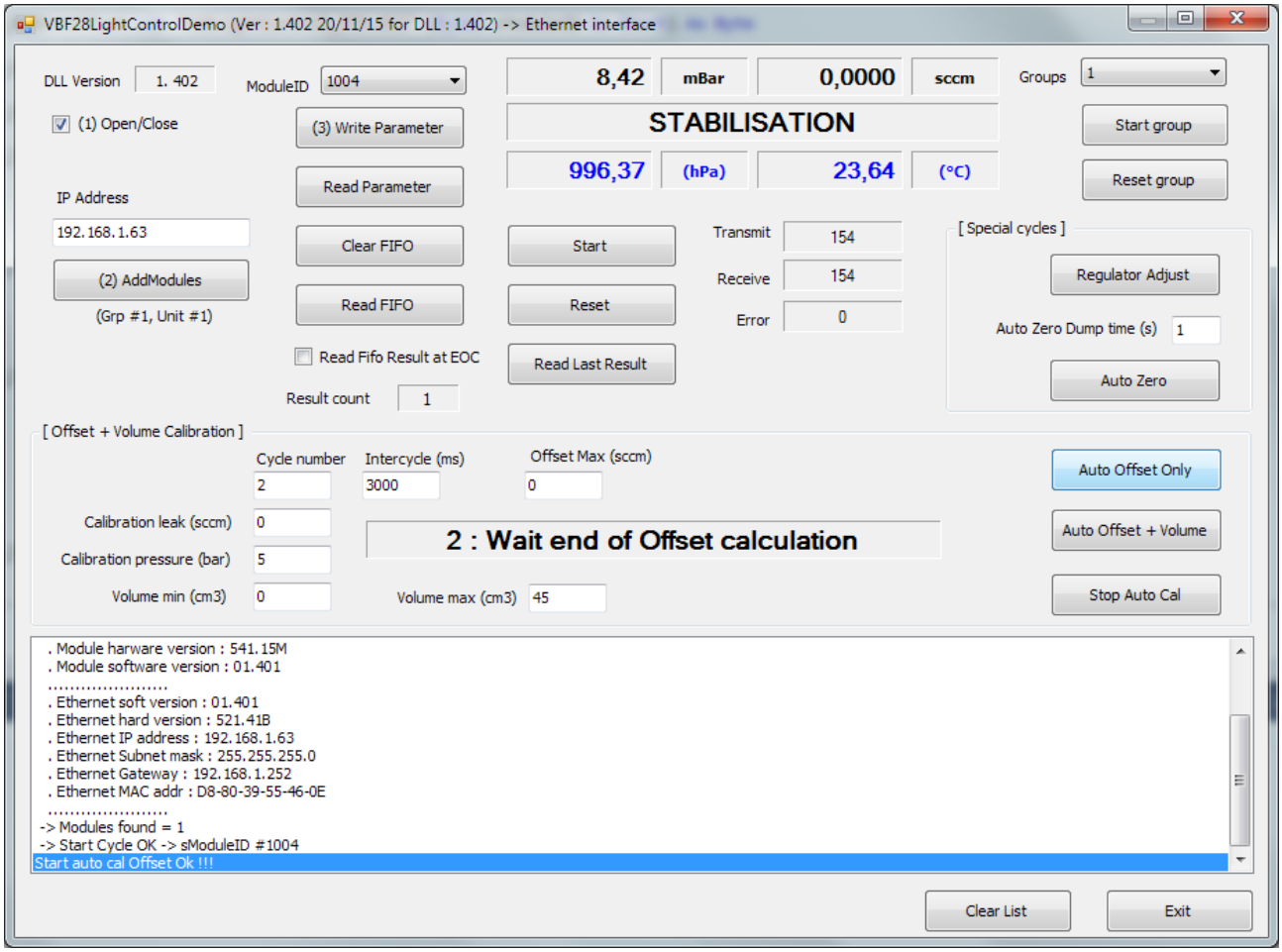
| Nom | Modifié le | Type | Taille |
| --- | --- | --- | --- |
| Dll | 03/11/2015 14:18 | Dossier de fichiers | |
| VBF28LightControlDemo | 04/11/2015 10:15 | Dossier de fichiers | |
| VBF28LightControlDemo.sln | 02/11/2015 15:04 | Microsoft Visual S... | 2 Ko |
| VBF28LightControlDemo.v12.suo | 04/11/2015 11:18 | Visual Studio Solu... | 106 Ko |

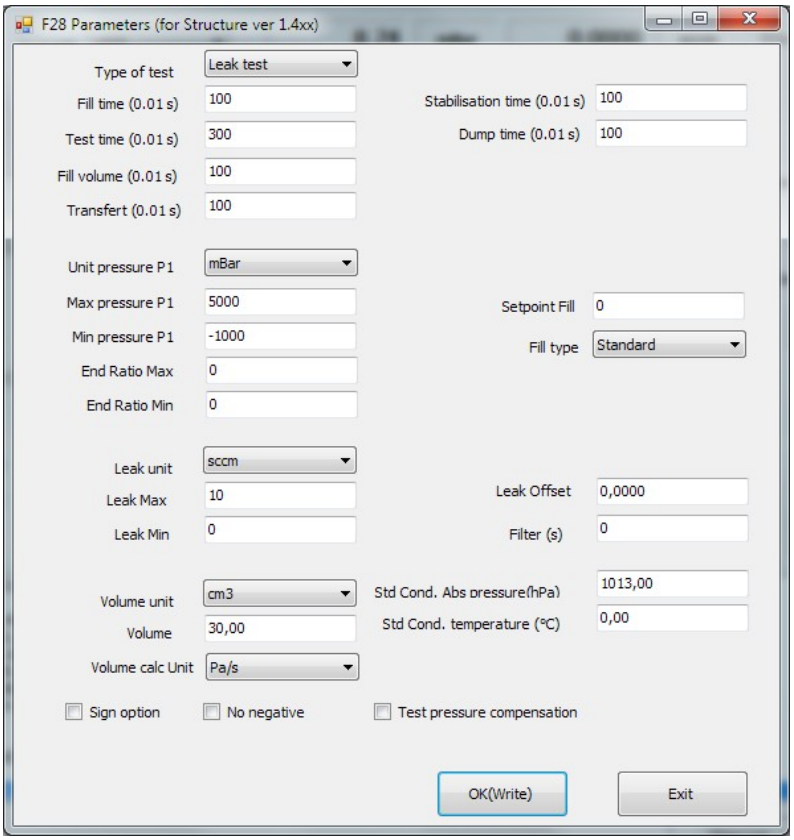
Release → for Release,

Debug → for debug.

| Nom | Modifié le | Type | Taille |
|---|---|---|---|
| Debug | 02/11/2015 14:56 | Dossier de fichiers | |
| Release | 02/11/2015 15:07 | Dossier de fichiers | |
| F28LightControl_ETH.dll | 02/11/2015 14:15 | Extension de l'app... | 77 Ko |
| F28LightControl_ETHD.dll | 02/11/2015 14:14 | Extension de l'app... | 337 Ko |




VBF28LightControlDemo (Ver : 1.402 20/11/15 for DLL : 1.402) -> Ethernet interface
DLL Version
1. 402
ModuleID
1004
8,42
mBar
0,0000
sccm
Groups
1
(1) Open/Close
(3) Write Parameter
STABILISATION
Start group
Read Parameter
996,37
(hPa)
23,64
(°C)
Reset group
IP Address
192.168.1.63
Clear FIFO
Start
Transmit
154
[ Special cycles ]
(2) AddModules
Read FIFO
Reset
Receive
154
Regulator Adjust
(Grp #1, Unit #1)
Error
0
Auto Zero Dump time (s)
1
Read Fifo Result at EOC
Read Last Result
Auto Zero
Result count
1
[ Offset + Volume Calibration ]
Cycle number
Intercycle (ms)
Offset Max (sccm)
Auto Offset Only
2
3000
0
Calibration leak (sccm)
0
2 : Wait end of Offset calculation
Auto Offset + Volume
Calibration pressure (bar)
5
Volume min (cm3)
0
Volume max (cm3)
45
Stop Auto Cal
. Module harware version : 541.15M
. Module software version : 01.401
. Ethernet soft version : 01.401
. Ethernet hard version : 521.41B
. Ethernet IP address : 192.168.1.63
. Ethernet Subnet mask : 255.255.255.0
. Ethernet Gateway : 192.168.1.252
. Ethernet MAC addr : D8-80-39-55-46-0E
-> Modules found = 1
-> Start Cycle OK -> sModuleID #1004
Start auto cal Offset Ok !!!
Clear List
Exit


F28 Parameters (for Structure ver 1.4xx)
Type of test
Leak test
Fill time (0.01 s)
100
Stabilisation time (0.01 s)
100
Test time (0.01 s)
300
Dump time (0.01 s)
100
Fill volume (0.01 s)
100
Transfert (0.01 s)
100
Unit pressure P1
mBar
Max pressure P1
5000
Setpoint Fill
0
Min pressure P1
-1000
Fill type
Standard
End Ratio Max
0
End Ratio Min
0
Leak unit
sccm
Leak Max
10
Leak Offset
0,0000
Leak Min
0
Filter (s)
0
Volume unit
cm3
Std Cond. Abs pressure(hPa)
1013,00
Volume
30,00
Std Cond. temperature (°C)
0,00
Volume calc Unit
Pa/s
Sign option
No negative
Test pressure compensation
OK(Write)
Exit

## 6.5 Sample code in VB.NET

### 6.5.1 Get & Display Ethernet information

```
' ----------------------------------------------------------------
' Convert from IP string to Long
' ----------------------------------------------------------------
Private Function IPString2Long(ByVal DottedIP As String) As Long
	Dim arrDec() As String
	Dim lResult As Long
	lResult = 0
	If DottedIP <> "" Then
		arrDec = DottedIP.Split(".")
		If (arrDec.Length = 4) Then
			lResult = CLng(arrDec(3)) * 2 ^ 24 + CLng(arrDec(2)) * 2 ^ 16 + CLng(arrDec(1)) * 2 ^ 8
			+ CLng(arrDec(0))
		End If
	End If
	Return lResult
End Function

' ----------------------------------------------------------------
' Read & display Ethernet information
' ----------------------------------------------------------------
'
Private Function GetEthernetInformation(ByVal sModuleID As Short, ByRef Info As T_ETH_INFO) As Short
	Dim sRet As Short
	Dim strBuff As String
	Dim ulIP As ULong
	Dim strMsg As String
	Const ucMaxBuff As Byte = 30


	strMsg = " ......................."
	DisplayTxt(strMsg)

	' Read soft version
	If (sRet = F28_RETURN.F28_OK) Then
		strBuff = Space(ucMaxBuff)
		sRet = F28_GetETHSoftVersion(sModuleID, strBuff, ucMaxBuff - 1)
		If (sRet = F28_RETURN.F28_OK) Then
			Info.strVersion = strBuff

			strMsg = " . Ethernet soft version : " + Info.strVersion
			DisplayTxt(strMsg)

		End If
	End If

	' Read hard version
	If (sRet = F28_RETURN.F28_OK) Then
		strBuff = Space(ucMaxBuff)
		sRet = F28_GetETHHardVersion(sModuleID, strBuff, ucMaxBuff - 1)
		If (sRet = F28_RETURN.F28_OK) Then
			Info.strHardVersion = strBuff

			strMsg = " . Ethernet hard version : " + Info.strHardVersion
			DisplayTxt(strMsg)
```

```
            End If
        End If

        ' Read IP address
        sRet = F28_GetAddressIP(sModuleID, ulIP)
        If (sRet = F28_RETURN.F28_OK) Then
            Dim curIPAdd As New IPAddress(ulIP)
            Info.strIP = curIPAdd.ToString()

            strMsg = " . Ethernet IP address : " + Info.strIP
            DisplayTxt(strMsg)

        End If

        ' Read Mask
        If (sRet = F28_RETURN.F28_OK) Then
            sRet = F28_GetSubnetMask(sModuleID, ulIP)
            If (sRet = F28_RETURN.F28_OK) Then
                Dim curIPAdd As New IPAddress(ulIP)
                Info.strSubnetMask = curIPAdd.ToString()

                strMsg = " . Ethernet Subnet mask : " + Info.strSubnetMask
                DisplayTxt(strMsg)

            End If
        End If

        ' Read gateway
        If (sRet = F28_RETURN.F28_OK) Then
            sRet = F28_GetGatewayAddressIP(sModuleID, ulIP)
            If (sRet = F28_RETURN.F28_OK) Then
                Dim curIPAdd As New IPAddress(ulIP)
                Info.strGateway = curIPAdd.ToString()

                strMsg = " . Ethernet Gateway : " + Info.strGateway
                DisplayTxt(strMsg)

            End If
        End If

        ' Read MAC address
        If (sRet = F28_RETURN.F28_OK) Then
            strBuff = Space(ucMaxBuff)
            sRet = F28_GetMACAddress(sModuleID, strBuff, ucMaxBuff - 1)
            If (sRet = F28_RETURN.F28_OK) Then
                Info.strMACAddress = strBuff

                strMsg = " . Ethernet MAC addr : " + Info.strMACAddress
                DisplayTxt(strMsg)

            End If
        End If

        strMsg = " ......................."
        DisplayTxt(strMsg)

        Return sRet

End Function
```

### 6.5.2 Get module information

```
'-----------------------------------------------------------------------
' GetModuleInfo
'-----------------------------------------------------------------------
'
Private Function GetModuleInfo(ByVal sModuleID) As Short
	Dim sRetCode As Short
	Dim strBuff As String
	Dim strMsg As String

	sRetCode = F28_RefreshModuleInformations(sModuleID)

	If (sRetCode = F28_RETURN.F28_OK) Then
		strBuff = Space(100)
		sRetCode = F28_GetSerialNumber(sModuleID, strBuff, 20)
		If (sRetCode = F28_RETURN.F28_OK) Then
		strMsg = strBuff.Insert(0, " . Serial number : ")
		DisplayTxt(strMsg)
		End If
	End If

	If (sRetCode = F28LightControl.F28_RETURN.F28_OK) Then
		strBuff = Space(100)
		sRetCode = F28_GetModuleHardVersion(sModuleID, strBuff, 20)
		If (sRetCode = F28LightControl.F28_RETURN.F28_OK) Then
			strMsg = strBuff.Insert(0, " . Module harware version : ")
			DisplayTxt(strMsg)
		End If
	End If

	If (sRetCode = F28LightControl.F28_RETURN.F28_OK) Then
		strBuff = Space(100)
		sRetCode = F28_GetModuleSoftVersion(sModuleID, strBuff, 20)
		If (sRetCode = F28LightControl.F28_RETURN.F28_OK) Then
			strMsg = strBuff.Insert(0, " . Module software version : ")
			DisplayTxt(strMsg)
		End If
	End If

	' 1.301 Get Ethernet info
	If (sRetCode = F28_RETURN.F28_OK) Then
		sRetCode = GetEthernetInformation(sModuleID, m_deviceEthernetInfo)
	End If

	GetModuleInfo = sRetCode

End Function
```

### 6.5.3 Read real time status & Read result cycle

```
' ---------------------------------------------------------------
' Read & display real time status & Measurement
' ---------------------------------------------------------------
Private Sub m_TimerRealTime_Tick(sender As Object, e As EventArgs) Handles m_TimerRealTime.Tick
	Dim sRetCode As Short
```

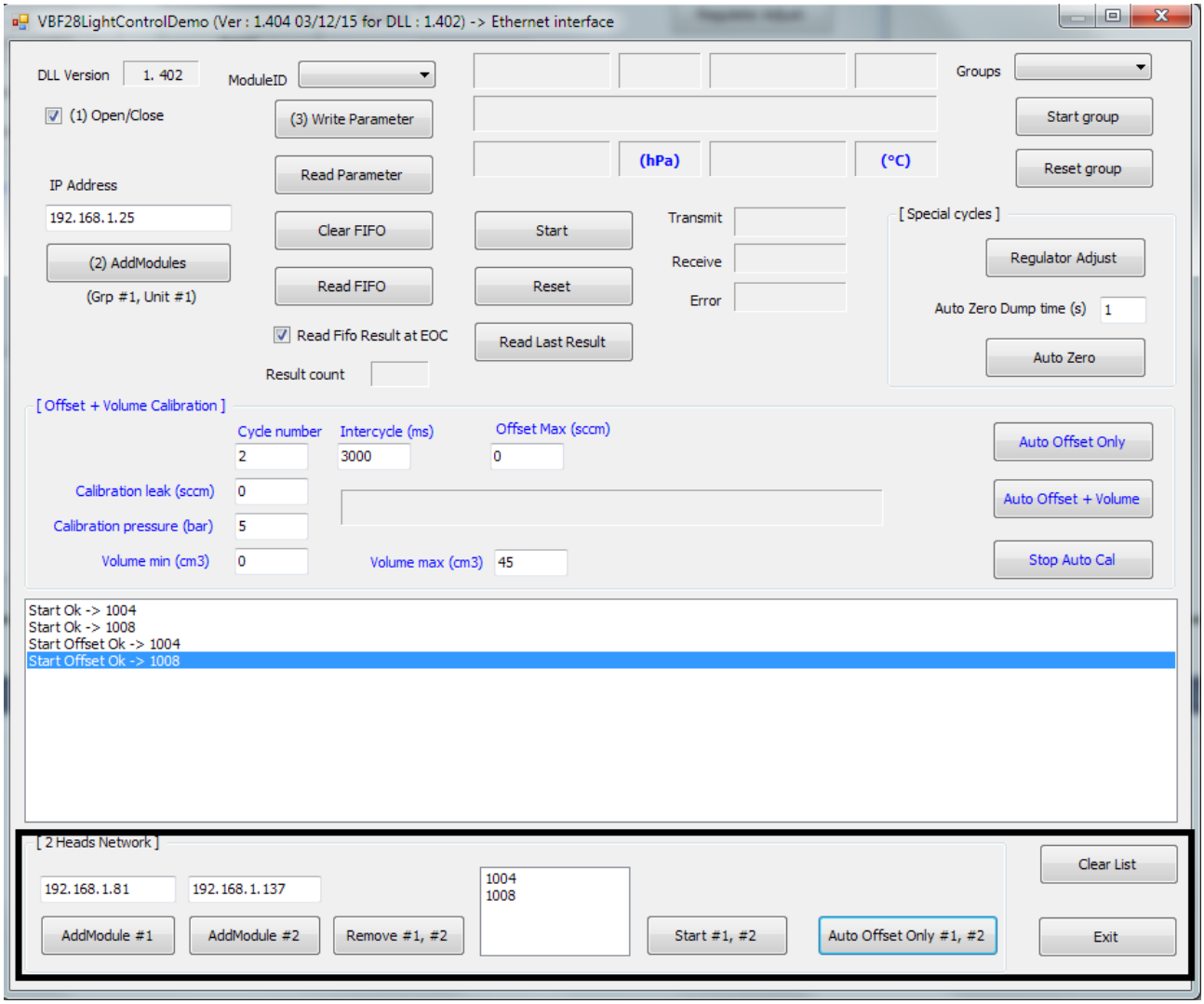
```
        Dim wCount As UShort
        If m_bAPIOpened And F28_IsModuleConnected(m_sModuleID) Then

                ' Read real time status & measurement
                sRetCode = F28_GetRealTimeData(m_sModuleID, m_realTime)
                If sRetCode = F28_RETURN.F28_OK Then

                        ' Display real time
                        DisplayRealTime()

                        ' If end of cycle -> Read last result & display
                        If (m_realTime.ucEndCycle > 0) Then

                                ' Stop real time reading at EOC
                                m_TimerRealTime.Stop()

                                ' Read Last Result
                                sRetCode = F28_GetLastResult(m_sModuleID, m_Result)
                                If sRetCode = F28_RETURN.F28_OK Then
                                        DisplayResult(0)
                                End If

                                ' Read Get fifo Result count
                                wCount = F28_GetResultsCount(m_sModuleID)
                                m_labelFifoCount.Text = wCount.ToString

                                ' Read fifo if demands
                                If wCount > 0 And m_chkReadFifo.Checked Then
                                        ' Read fifo
                                        sRetCode = F28_GetNextResult(m_sModuleID, m_Result)
                                        If sRetCode = F28_RETURN.F28_OK Then
                                                DisplayResult(1)
                                        End If

                                        wCount = F28_GetResultsCount(m_sModuleID)
                                        m_labelFifoCount.Text = wCount.ToString
                                End If
                        End If
                End If
        End If

' Read & display counter
If sRetCode = F28_RETURN.F28_OK Then
        sRetCode = F28_GetCommunicationStatistics(m_sModuleID, m_rCptComm)
        If sRetCode = F28_RETURN.F28_OK Then
                DisplayCounter()
        End If
End If
End Sub
```

### 6.5.4 Auto-zero pressure

```
' ----------------------------------------------------------------
' Auto Zero pressure
' ----------------------------------------------------------------
'
```

```
Private Sub btnAZPressure_Click(sender As Object, e As EventArgs) Handles btnAZPressure.Click

      Dim fDumpTime As Single
      Dim sRetCode As Short

      If m_bAPIOpened And F28_IsModuleConnected(m_sModuleID) Then

            ' Get dump time in sec
            fDumpTime = Convert.ToSingle(textBoxAZDumpTime.Text)

            sRetCode = F28_StartAutoZeroPressure(m_sModuleID, fDumpTime)

            If (sRetCode = F28_RETURN.F28_OK) Then
                  DisplayTxt("Start auto zero Ok !!!")
            Else
                  DisplayTxt("Start auto zero error !!!")
            End If
      End If

End Sub
```

## 6.6 Start cal offset for more than one head in VB.NET




Below is a example of VB.net code, how to do a Start /Start Auto Cal Offset only for more than one heads.

We need only to repeat the command to each heads.

```
 ' ------------------------------------------------------------------
    ' Start auto cal offset for all heads inside the listBox
' ------------------------------------------------------------------
'
Private Sub Button5_Click(sender As Object, e As EventArgs) Handles btnOffset2.Click
Dim n As Integer
      Dim strBuff As String
      Dim sNum As Short
      Dim sRet As Short
      Dim wNbCycles As UShort
      Dim wInterCycle As UShort
      Dim fOffsetMax As Single

      wNbCycles = Convert.ToInt16(textBoxCycleNumber.Text)
      wInterCycle = Convert.ToInt16(textBoxIntercycle.Text)
      fOffsetMax = Convert.ToSingle(textBoxOffset.Text)
```

```
        ' Get number of heads inside the listbox
        n = m_listBox2Heads.Items.Count

        ' If not empty
        If n > 0 Then

' Repeat for all heads
        For i = 0 To n - 1

                ' Get sModuleID for head i
                strBuff = m_listBox2Heads.Items.Item(i)
                sNum = CShort(strBuff)

                ' Check if the module is connected
                If m_bAPIOpened And F28_IsModuleConnected(sNum) Then

                        ' Start auto Cal Offset for head i
                        sRet = F28_StartAutoCalOffsetOnly(sNum, wNbCycles, wInterCycle, fOffsetMax)

                        If (sRet = F28_RETURN.F28_OK) Then
                                DisplayTxt("Start Offset Ok -> " + sNum.ToString())
                        Else
                                DisplayTxt("Start Offset error -> " + sNum.ToString())
                        End If

                End If

                Next
        End If

End Sub
```