

DLL to interface FMod-IPDCMOT

FMod-IPDCMOT-DLLInterface

User Manual

Version 1.0

Version: 1.0
Last revision: August 14th 2007
Printed in Switzerland

© Copyright 2003-2007 FiveCo Sàrl. All rights reserved.
The contents of this manual may be modified by FiveCo without any warning.

Trademarks

Windows® is a registered trademark of Microsoft Corporation.
Ethernet® is a registered trademark of Xerox Corporation.
Borland® is a registered trademark of Borland Software Corporation.
Visual C++® is a registered trademark of Microsoft Corporation.
Philips® is a registered trademark of Koninklijke Philips Electronics N.V.

Support

e-mail: support@fiveco.ch

Table of Contents

| | |
|---|----|
| 1. Preliminary | 6 |
| Overview | 6 |
| Files | 6 |
| Revision history | 6 |
| 2. Structures of the DLL | 7 |
| FMod_SRegisterListRead | 7 |
| FMod_SModule | 11 |
| FMod_SHOMINGOPTIONS | 11 |
| 3. Functions of the DLL | 12 |
| General Functions | 12 |
| MAIN Port Functions | 12 |
| Repetitive asks Function | 16 |
| 4. General Functions description | 17 |
| FMod_IPDCMOT_ScanNetwork | 17 |
| FMod_IPDCMOT_ChangeIPAddress | 18 |
| FMod_IPDCMOT_VersionDLL | 19 |
| FMod_IPDCMOT_GetStateCommunication | 20 |
| FMod_IPDCMOT_GetLastError | 21 |
| 5. MAIN Port functions descriptions | 22 |
| FMod_IPDCMOT_OpenConnection_MAINPORT | 22 |
| FMod_IPDCMOT_CloseConnection | 23 |
| FMod_IPDCMOT_Read_TYPE | 24 |
| FMod_IPDCMOT_Read_VERSION | 25 |
| FMod_IPDCMOT_Write_RESETCPU | 26 |
| FMod_IPDCMOT_Write_SAVEUSERPARAMETERS | 27 |
| FMod_IPDCMOT_Write_RESTOREUSERPARAMETERS | 28 |
| FMod_IPDCMOT_Write_RESTOREFACTORYPARAMETERS | 29 |
| FMod_IPDCMOT_Write_SAVEFACTORYPARAMETERS | 30 |
| FMod_IPDCMOT_Read_VOLTAGE | 31 |
| FMod_IPDCMOT_Read_WARNINGS | 32 |
| FMod_IPDCMOT_Reset_WARNINGS | 33 |
| FMod_IPDCMOT_Read_COMMUNICATIONOPTIONS | 34 |
| FMod_IPDCMOT_Write_COMMUNICATIONOPTIONS | 35 |
| FMod_IPDCMOT_Read_ETHERNETMAC | 36 |
| FMod_IPDCMOT_Read_IPADDRESS | 37 |
| FMod_IPDCMOT_Write_IPADDRESS | 38 |
| FMod_IPDCMOT_Read_SUBNETMASK | 39 |
| FMod_IPDCMOT_Write_SUBNETMASK | 40 |
| FMod_IPDCMOT_Read_TCPTIMEOUT | 41 |
| FMod_IPDCMOT_Write_TCPTIMEOUT | 42 |
| FMod_IPDCMOT_Read_MODULENAME | 43 |
| FMod_IPDCMOT_Write_MODULENAME | 44 |
| FMod_IPDCMOT_Read_TCPCONNECTIONSOPENED | 45 |
| FMod_IPDCMOT_Read_REGULATIONMODE | 46 |

| | |
|---|------------------------------------|
| FMod_IPDCMOT_Write_REGULATIONMODE..... | 47 |
| FMod_IPDCMOT_Read_INPUT..... | 48 |
| FMod_IPDCMOT_Write_INPUT..... | 49 |
| FMod_IPDCMOT_Write_INPUTOFFSET..... | 50 |
| FMod_IPDCMOT_Write_INPUTOFFSETMEASURED..... | 51 |
| FMod_IPDCMOT_Read_INPUTMIN..... | 52 |
| FMod_IPDCMOT_Write_INPUTMIN..... | 53 |
| FMod_IPDCMOT_Read_INPUTMAX..... | 54 |
| FMod_IPDCMOT_Write_INPUTMAX..... | 55 |
| FMod_IPDCMOT_Read_POSITION..... | 56 |
| FMod_IPDCMOT_Write_POSITION..... | 57 |
| FMod_IPDCMOT_Write_POSITIONOFFSET..... | 58 |
| FMod_IPDCMOT_Read_SPEED..... | 59 |
| FMod_IPDCMOT_Read_TEMPERATURE..... | Erreur ! Signet non défini. |
| FMod_IPDCMOT_Read_CURRENTMAX..... | 60 |
| FMod_IPDCMOT_Write_CURRENTMAX..... | 61 |
| FMod_IPDCMOT_Read_OPTIONS..... | 62 |
| FMod_IPDCMOT_Write_OPTIONS..... | 63 |
| FMod_IPDCMOT_Read_LOOPTIME..... | 64 |
| FMod_IPDCMOT_Write_LOOPTIME..... | 65 |
| FMod_IPDCMOT_Read_OUTPUTVOLTAGEMAX..... | 66 |
| FMod_IPDCMOT_Write_OUTPUTVOLTAGEMAX..... | 67 |
| FMod_IPDCMOT_Read_DESIRED..... | 68 |
| FMod_IPDCMOT_Read_FEEDBACK..... | 69 |
| FMod_IPDCMOT_Read_COMMAND..... | 70 |
| FMod_IPDCMOT_Read_KP..... | 71 |
| FMod_IPDCMOT_Write_KP..... | 72 |
| FMod_IPDCMOT_Read_KI..... | 73 |
| FMod_IPDCMOT_Write_KI..... | 74 |
| FMod_IPDCMOT_Read_KD..... | 75 |
| FMod_IPDCMOT_Write_KD..... | 76 |
| FMod_IPDCMOT_Read_ANTIRESETWINDUP..... | 77 |
| FMod_IPDCMOT_Write_ANTIRESETWINDUP..... | 78 |
| FMod_IPDCMOT_Read_INTEGRALDELTA..... | 79 |
| FMod_IPDCMOT_Write_INTEGRALDELTA..... | 80 |
| FMod_IPDCMOT_Read_DERIVATIONOFDELTA..... | 81 |
| FMod_IPDCMOT_Read_AUTOTUNING..... | Erreur ! Signet non défini. |
| FMod_IPDCMOT_Write_AUTOTUNING..... | Erreur ! Signet non défini. |
| FMod_IPDCMOT_Read_ACCELERATION..... | 82 |
| FMod_IPDCMOT_Write_ACCELERATION..... | 83 |
| FMod_IPDCMOT_Read_DECELERATION..... | 84 |
| FMod_IPDCMOT_Write_DECELERATION..... | 85 |
| FMod_IPDCMOT_Read_TOPSPEED..... | 86 |
| FMod_IPDCMOT_Write_TOPSPEED..... | 87 |
| FMod_IPDCMOT_Read_DEADZONE..... | 88 |
| FMod_IPDCMOT_Write_DEADZONE..... | 89 |
| FMod_IPDCMOT_Read_HOMINGOPTIONS..... | 90 |
| FMod_IPDCMOT_Write_HOMINGOPTIONS..... | 91 |

| | |
|---|-----|
| FMod_IPDCMOT_Write_HOMING | 92 |
| FMod_IPDCMOT_Write_STOPHOMING | 93 |
| FMod_IPDCMOT_Read_HOMINGPOSITION | 94 |
| FMod_IPDCMOT_Write_HOMINGPOSITION | 95 |
| FMod_IPDCMOT_Read_HOMINGINPUT | 96 |
| FMod_IPDCMOT_Write_HOMINGINPUT | 97 |
| FMod_IPDCMOT_Read_LIMIT1SETUP | 98 |
| FMod_IPDCMOT_Write_LIMIT1SETUP | 99 |
| FMod_IPDCMOT_Read_LIMIT1REGULATIONMODE | 100 |
| FMod_IPDCMOT_Write_LIMIT1REGULATIONMODE | 101 |
| FMod_IPDCMOT_Read_LIMIT1POSITION | 102 |
| FMod_IPDCMOT_Write_LIMIT1POSITION | 103 |
| FMod_IPDCMOT_Read_LIMIT1XINPUT | 104 |
| FMod_IPDCMOT_Write_LIMIT1XINPUT | 105 |
| FMod_IPDCMOT_Read_LIMIT2SETUP | 106 |
| FMod_IPDCMOT_Write_LIMIT2SETUP | 107 |
| FMod_IPDCMOT_Read_LIMIT2REGULATIONMODE | 108 |
| FMod_IPDCMOT_Write_LIMIT2REGULATIONMODE | 109 |
| FMod_IPDCMOT_Read_LIMIT2POSITION | 110 |
| FMod_IPDCMOT_Write_LIMIT2POSITION | 111 |
| FMod_IPDCMOT_Read_LIMIT2XINPUT | 112 |
| FMod_IPDCMOT_Write_LIMIT2XINPUT | 113 |
| FMod_IPDCMOT_Read_AllRegister | 114 |
| FMod_IPDCMOT_ResetAllRegisterRead | 115 |
| FMod_IPDCMOT_SendData_MAINPORT | 116 |
| 6. Repetitive ask Function | 117 |
| FMod_IPDCMOT_RepetitiveAskStart | 118 |
| FMod_IPDCMOT_RepetitiveAskStop | 119 |
| FMod_IPDCMOT_RepetitiveAskSetTime | 120 |
| FMod_IPDCMOT_RepetitiveAskReset | 121 |
| 7. Communication events functions | 122 |
| Data_Received_MAINPORT | 123 |
| Com_Event | 124 |
| 8. Applications example | 125 |
| MAIN Port : Open and close communication | 125 |
| MAIN Port communication: Read and Write register | 126 |
| MAIN Port communication: Repetitive register read | 127 |

I. Preliminary

Overview

The main goal of this Dynamic Link Library (DLL) is to help users to interface FiveCo's Ethernet products. It manages both the communication and FiveCo protocols so that you can easily read and/or write information to the next products:

- FMod-IPDCMOT 48/1.5

This DLL is based on "C" language. It works with every compiler under "Windows 32 bits" operating system.

Files

The "FMod-IPDCMOT-DLLInterface" contains 3 different Files:

FMod_IPDCMOT_DLLInterface.dll

It contains the compiled software to interface FMod-IPDCMOT product.

FMod_IPDCMOT_DLLInterface.lib

This file is the library of the DLL's accessible functions. **WARNING:** this file could be specific to the compiler you use.

FMod_IPDCMOT_DLLInterface.h

This header file contains the declarations of the DLL's accessible functions.

Revision history

| DLL revision | Date | Author | Note | Manual version |
|--------------|----------|--------|-----------------|----------------|
| 1.00 | 14.08.07 | GF | - First release | 1.0 |

2. Structures of the DLL

Two structures, defined in the header file, are necessary in order to interface the DLL.

FMod_SRegisterListRead

This structure contains all the register value of the FMod-IPDCMOT. It also contains Boolean value to inform the user when a register value has been read or written.

Declaration of the FMod_SRegisterListRead structure:

```
struct FMod_SRegisterListRead
{
    int    TYPE[2];          // TYPE[0] : Type / TYPE[1] : Model
    bool   TYPE_Read;        // TYPE has been read

    int    VERSION[2];       // VERSION[0] : Version / VERSION[1] : Revision
    bool   VERSION_Read;     // VERSION has been read

    bool   RESETCPU_Written;  // RESETCPU has been written
    bool   SAVEUSERPARAMETERS_Written;
    // SAVEUSERPARAMETERS has been written
    bool   RESTOREUSERPARAMETERS_Written;
    // RESTOREUSERPARAMETERS has been written
    bool   RESTOREFACTORYPARAMETERS_Written;
    // RESTOREFACTORYPARAMETERS has been written
    bool   SAVEFACTORYPARAMETERS_Written;
    // SAVEFACTORYPARAMETERS has been written

    float  VOLTAGE; // Input Voltage (2Bytes.2Bytes)
    bool   VOLTAGE_Read;          // VOLTAGE has been read

    bool   WARNINGS[32];          // Warnings (32 individual bits)
    bool   WARNINGS_Read;         // WARNINGS has been read
    bool   WARNINGS_Reset;        // WARNINGS has been written

    bool   COMMUNICATIONOPTIONS[32]; // 32 individual bits
    bool   COMMUNICATIONOPTIONS_Read;
    // COMMUNICATIONOPTIONS has been read
    bool   COMMUNICATIONOPTIONS_Written;
    // COMMUNICATIONOPTIONS has been written

    int    ETHERNETMAC[6];        // Mac address (6 unsigned bytes)
    bool   ETHERNETMAC_Read;      // ETHERNETMAC has been read

    int    IPADDRESS[4];          // IP Address (4 unsigned bytes)
    bool   IPADDRESS_Read;        // IPADDRESS has been read
    bool   IPADDRESS_Written;     // IPADDRESS has been written

    int    SUBNETMASK[4];         // Network IP subnet mask (4 unsigned bytes)
    bool   SUBNETMASK_Read;       // SUBNETMASK has been read
    bool   SUBNETMASK_Written;    // SUBNETMASK has been written

    int    TCPTIMEOUT;            // TCP Timeout (1 unsigned byte)
    bool   TCPTIMEOUT_Read;       // TCPTIMEOUT has been read
    bool   TCPTIMEOUT_Written;    // TCPTIMEOUT has been written

    char   MODULENAME[16];
    // Name and/or description of the module (16 unsigned bytes (char))
    bool   MODULENAME_Read;       // MODULENAME has been read
    bool   MODULENAME_Written;    // MODULENAME has been written
}
```

```

int    TCPCONNECTIONSOPENED;
//Number of users connected to the card using TCP
bool   TCPCONNECTIONSOPENED_Read;
// TCPCONNECTIONOPENED has been read

int    REGULATIONMODE;
// Regulation Mode (1 unsigned byte) See value list
bool   REGULATIONMODE_Read;           // REGULATIONMODE has been read
bool   REGULATIONMODE_Written; // REGULATIONMODE has been written

int    INPUT;           // Input Value (4 bytes signed integer value)
bool   INPUT_Read;      // INPUT has been read
bool   INPUT_Written;   // INPUT has been written

bool   INPUTOFFSET_Written; // INPUTOFFSET has been written
bool   INPUTOFFSETMEASURED_Written;
// INPUTOFFSETMEASURED has been written

int    INPUTMIN;
// Input Min value (4 bytes signed integer value)
bool   INPUTMIN_Read;    // INPUTMIN has been read
bool   INPUTMIN_Written; // INPUTMIN has been written

int    INPUTMAX;
// Input Max value (4 bytes signed integer value)
bool   INPUTMAX_Read;    // INPUTMAX has been read
bool   INPUTMAX_Written; // INPUTMAX has been written

int    POSITION;         // Position (4 bytes signed integer value)
bool   POSITION_Read;    // POSITION has been read
bool   POSITION_Written; // POSITION has been written

bool   POSITIONOFFSET_Written; // POSITIONOFFSET has been written

int    SPEED;           // Speed (4 bytes signed integer value)
bool   SPEED_Read;      // SPEED has been read

float  CURRENTMAX;      // Current Max (2Bytes.2Bytes)
bool   CURRENTMAX_Read; // CURRENTMAX has been read
bool   CURRENTMAX_Written; // CURRENTMAX has been written

bool   OPTIONS[32];     // Options (32 individual bits)
bool   OPTIONS_Read;    // OPTIONS has been read
bool   OPTIONS_Written; // OPTIONS has been written

int    LOOPTIME;        // Loop Time (1 unsigned byte) See value list
bool   LOOPTIME_Read;   // POSITION has been read
bool   LOOPTIME_Written; // POSITION has been written

float  OUPUTVOLTAGEMAX; // Output Voltage Max (2Bytes.2Bytes)
bool   OUPUTVOLTAGEMAX_Read; // OUPUTVOLTAGEMAX has been read
bool   OUPUTVOLTAGEMAX_Written; // OUPUTVOLTAGEMAX has been written

int    DESIRED;
// Positive input of PID (4 bytes signed integer value)
bool   DESIRED_Read;    // DESIRED has been read

int    FEEDBACK;
// Negative input of PID (4 bytes signed integer value)
bool   FEEDBACK_Read;   // FEEDBACK has been read

int    COMMAND;         // Result of PID (4 bytes signed integer value)
bool   COMMAND_Read;    // COMMAND has been read

float  KP;              // PID Proportional gain (2Bytes.2Bytes)
bool   KP_Read;         // KP has been read
bool   KP_Written;      // KP has been written

```



```

float KI;           // PID Integral gain (2Bytes.2Bytes)
boolKI_Read;       // KI has been read
boolKI_Written;     // KI has been written

float KD;           // PID Derivative gain (2Bytes.2Bytes)
bool  KD_Read;      // KD has been read
bool  KD_Written;   // KD has been written

int  ANTIRESETWINDUP;
// Integration saturation (4 bytes signed integer value)
bool ANTIRESETWINDUP_Read; // ANTIRESETWINDUP has been read
bool ANTIRESETWINDUP_Written; // ANTIRESETWINDUP has been written

int  INTEGRALDELTA;
// PID Integral result (4 bytes signed integer value)
bool INTEGRALDELTA_Read; // INTEGRALDELTA has been read
bool INTEGRALDELTA_Written; // INTEGRALDELTA has been written

int  DERIVATIONOFDELTA;
// PID derivative result (4 bytes signed integer value)
bool DERIVATIONOFDELTA_Read; // DERIVATIONOFDELTA has been read

int  ACCELERATION;
// Speed acceleration (4 bytes signed integer value)
bool ACCELERATION_Read; // ACCELERATION has been read
bool ACCELERATION_Written; // ACCELERATION has been written

int  DECELERATION;
// Speed deceleration (4 bytes signed integer value)
bool DECELERATION_Read; // DECELERATION has been read
bool DECELERATION_Written; // DECELERATION has been written

int  TOPSPEED;
// Maximum Speed (4 bytes signed integer value)
bool TOPSPEED_Read; // TOPSPEED has been read
bool TOPSPEED_Written; // TOPSPEED has been written

int  DEADZONE;
// Cancels speed zone (4 bytes signed integer value)
bool DEADZONE_Read; // DEADZONE has been read
bool DEADZONE_Written; // DEADZONE has been written

FMod_SHOMINGOPTIONS HOMINGOPTIONS;
// Homing Options (special structure)
bool HOMINGOPTIONS_Read; // HOMINGOPTIONS has been read
bool HOMINGOPTIONS_Written; // HOMINGOPTIONS has been written

bool HOMING_Written; // HOMING has been written (fct)
bool STOPHOMING_Written; // STOPHOMING has been written (fct)

int  HOMINGPOSITION;
// Sets the Home reference value (4 bytes signed integer value)
bool HOMINGPOSITION_Read; // HOMINGPOSITION has been read
bool HOMINGPOSITION_Written; // HOMINGPOSITION has been written

int  HOMINGINPUT;
// Sets a new goal when home is found (4 bytes signed integer value)
bool HOMINGINPUT_Read; // HOMINGINPUT has been read
bool HOMINGINPUT_Written; // HOMINGINPUT has been written

int  LIMIT1SETUP;
// Configures limit n° 1 (4 bytes unsigned integer value)
bool LIMIT1SETUP_Read; // LIMIT1SETUP has been read
bool LIMIT1SETUP_Written; // LIMIT1SETUP has been written

```

```

int    LIMIT1REGULATIONMODE;
// Limit 1 Regulation Mode (1 unsigned byte) See value list
bool   LIMIT1REGULATIONMODE_Read;
// LIMIT1REGULATIONMODE has been read
bool   LIMIT1REGULATIONMODE_Written;
// LIMIT1REGULATIONMODE has been written

int    LIMIT1POSITION;
// Sets a new position (4 bytes signed integer value)
bool   LIMIT1POSITION_Read;           // LIMIT1POSITION has been read
bool   LIMIT1POSITION_Written; // LIMIT1POSITION has been written

int    LIMIT1XINPUT;
// Sets a new input (4 bytes signed integer value)
bool   LIMIT1XINPUT_Read;             // LIMIT1XINPUT has been read
bool   LIMIT1XINPUT_Written;  // LIMIT1XINPUT has been written

int    LIMIT2SETUP;
// Configures limit n° 2 (4 bytes unsigned integer value)
bool   LIMIT2SETUP_Read;              // LIMIT2SETUP has been read
bool   LIMIT2SETUP_Written;           // LIMIT2SETUP has been written

int    LIMIT2REGULATIONMODE;
// Limit 2 Regulation Mode (1 unsigned byte) See value list
bool   LIMIT2REGULATIONMODE_Read;
// LIMIT2REGULATIONMODE has been read
bool   LIMIT2REGULATIONMODE_Written;
// LIMIT2REGULATIONMODE has been written

int    LIMIT2POSITION;
// Sets a new position (4 bytes signed integer value)
bool   LIMIT2POSITION_Read;           // LIMIT2POSITION has been read
bool   LIMIT2POSITION_Written; // LIMIT2POSITION has been written

int    LIMIT2XINPUT;
// Sets a new input (4 bytes signed integer value)
bool   LIMIT2XINPUT_Read;             // LIMIT2XINPUT has been read
bool   LIMIT2XINPUT_Written;           // LIMIT2XINPUT has been written
};

```

FMod_SModule

This structure contains the main information about a module. It is used to scan the network (FMod_IPDCMOT_ScanNetwork) and to receive information about all the FiveCo modules connected (not only FMod-IPDCMOT) on the network.

Declaration of the FMod_SModule structure:

```
struct FMod_SModule
{
    int     TYPE[2];
    int     VERSION[2];
    int     ETHERNETMAC[6];
    int     IPADDRESS[4];

    char    MODULENAME[16];
};
```

FMod_SHOMINGOPTIONS

This structure contains detailed information of the *HOMINGOPTIONS* register.

```
struct FMod_SHOMINGOPTIONS
{
    int     HomingMethod;
    bool    HomingAtPowerUp;
    int     RatioCURRENTMAX;
    int     RatioTOPSPEED;
    int     RatioACCELERATION;
    int     SaturatedTime;
};
```

3. Functions of the DLL

A set of functions is accessible from the FMod_IPDCMOT_DLLInterface DLL to communicate with the FMod-IPDCMOT product via Ethernet.

General Functions

| Name of the function | Description |
|------------------------------------|---|
| FMod_IPDCMOT_ScanNetwork | Retrieves a list of all FiveCo's modules connected on the network. |
| FMod_IPDCMOT_ChangeIPAddress | Sets the IP address and the subnet mask to the module with the specified Mac address. |
| FMod_IPDCMOT_VersionDLL | Gets version of the FMod-IPDCMOT-DLLInterface DLL. |
| FMod_IPDCMOT_GetStateCommunication | Gets the actual state of the communication. |
| FMod_IPDCMOT_GetLastError | Gets the last error that occurred in FMod-IPDCMOT-DLLInterface DLL. |

MAIN Port Functions

| Name of the function | Description |
|---|--|
| FMod_IPDCMOT_OpenConnection_MAINPORT | Opens the Ethernet communication on the MAIN Port. |
| FMod_IPDCMOT_CloseConnection | Closes the Ethernet communication. |
| FMod_IPDCMOT_Read_TYPE | Reads register TYPE. |
| FMod_IPDCMOT_Read_VERSION | Reads register VERSION. |
| FMod_IPDCMOT_Write_RESETCPU | Sends command RESETCPU. |
| FMod_IPDCMOT_Write_SAVEUSERPARAMETERS | Sends command SAVEUSERPARAMETERS. |
| FMod_IPDCMOT_Write_RESTOREUSERPARAMETERS | Sends command RESTOREUSERPARAMETERS. |
| FMod_IPDCMOT_Write_RESTOREFACTORYPARAMETERS | Sends command RESTOREFACTORYPARAMETERS. |
| FMod_IPDCMOT_Write_SAVEFACTORYPARAMETERS | Sends command SAVEFACTORYPARAMETERS. |
| FMod_IPDCMOT_Read_VOLTAGE | Reads register VOLTAGE. |
| FMod_IPDCMOT_Read_WARNINGS | Reads register WARNINGS. |
| FMod_IPDCMOT_Reset_WARNINGS | Resets register WARNINGS. |

| | |
|---|---------------------------------------|
| FMod_IPDCMOT_Read_COMMUNICATIONOPTIONS | Reads register COMMUNICATIONOPTIONS. |
| FMod_IPDCMOT_Write_COMMUNICATIONOPTIONS | Writes register COMMUNICATIONOPTIONS. |
| FMod_IPDCMOT_Read_ETHERNETMAC | Reads register ETHERNETMAC. |
| FMod_IPDCMOT_Read_IPADDRESS | Reads register IPADDRESS. |
| FMod_IPDCMOT_Write_IPADDRESS | Writes register IPADDRESS. |
| FMod_IPDCMOT_Read_SUBNETMASK | Reads register SUBNETMASK. |
| FMod_IPDCMOT_Write_SUBNETMASK | Writes register SUBNETMASK. |
| FMod_IPDCMOT_Read_TCPTIMEOUT | Reads register TCPTIMEOUT. |
| FMod_IPDCMOT_Write_TCPTIMEOUT | Writes register TCPTIMEOUT. |
| FMod_IPDCMOT_Read_MODULENAME | Reads register MODULENAME. |
| FMod_IPDCMOT_Write_MODULENAME | Writes register MODULENAME. |
| FMod_IPDCMOT_Read_TCPCONNECTIONSOPENED | Reads register TCPCONNECTIONSOPENED. |
| FMod_IPDCMOT_Read_REGULATIONMODE | Reads register REGULATIONMODE. |
| FMod_IPDCMOT_Write_REGULATIONMODE | Writes register REGULATIONMODE. |
| FMod_IPDCMOT_Read_INPUT | Reads register INPUT. |
| FMod_IPDCMOT_Write_INPUT | Writes register INPUT. |
| FMod_IPDCMOT_Write_INPUTOFFSET | Writes register INPUTOFFSET. |
| FMod_IPDCMOT_Write_INPUTOFFSETMEASURED | Writes register INPUTOFFSETMEASURED. |
| FMod_IPDCMOT_Read_INPUTMIN | Reads register INPUTMIN. |
| FMod_IPDCMOT_Write_INPUTMIN | Writes register INPUTMIN. |
| FMod_IPDCMOT_Read_INPUTMAX | Reads register INPUTMAX. |
| FMod_IPDCMOT_Write_INPUTMAX | Writes register INPUTMAX. |
| FMod_IPDCMOT_Read_POSITION | Reads register POSITION. |
| FMod_IPDCMOT_Write_POSITION | Writes register POSITION. |
| FMod_IPDCMOT_Write_POSITIONOFFSET | Writes register POSITIONOFFSET. |
| FMod_IPDCMOT_Read_SPEED | Reads register SPEED. |
| FMod_IPDCMOT_Read_CURRENTMAX | Reads register CURRENTMAX. |
| FMod_IPDCMOT_Write_CURRENTMAX | Writes register CURRENTMAX. |
| FMod_IPDCMOT_Read_OPTIONS | Reads register OPTIONS. |
| FMod_IPDCMOT_Write_OPTIONS | Writes register OPTIONS. |
| FMod_IPDCMOT_Read_LOOPTIME | Reads register LOOPTIME. |
| FMod_IPDCMOT_Write_LOOPTIME | Writes register LOOPTIME. |

| | |
|-------------------------------------|-----------------------------------|
| FMod_IPDCMOT_Read_OUTPUTVOLTAGEMAX | Reads register OUTPUTVOLTAGEMAX. |
| FMod_IPDCMOT_Write_OUTPUTVOLTAGEMAX | Writes register OUTPUTVOLTAGEMAX. |
| FMod_IPDCMOT_Read_DESIRED | Reads register DESIRED. |
| FMod_IPDCMOT_Read_FEEDBACK | Reads register FEEDBACK. |
| FMod_IPDCMOT_Read_COMMAND | Reads register COMMAND. |
| FMod_IPDCMOT_Read_KP | Reads register KP. |
| FMod_IPDCMOT_Write_KP | Writes register KP. |
| FMod_IPDCMOT_Read_KI | Reads register KI. |
| FMod_IPDCMOT_Write_KI | Writes register KI. |
| FMod_IPDCMOT_Read_KD | Reads register KD. |
| FMod_IPDCMOT_Write_KD | Writes register KD. |
| FMod_IPDCMOT_Read_ANTIRESETWINDUP | Reads register ANTIRESETWINDUP. |
| FMod_IPDCMOT_Write_ANTIRESETWINDUP | Writes register ANTIRESETWINDUP. |
| FMod_IPDCMOT_Read_INTEGRALDELTA | Reads register INTEGRALDELTA. |
| FMod_IPDCMOT_Write_INTEGRALDELTA | Writes register INTEGRALDELTA. |
| FMod_IPDCMOT_Read_DERIVATIONOFDELTA | Reads register DERIVATIONOFDELTA. |
| FMod_IPDCMOT_Read_ACCELERATION | Reads register ACCELERATION. |
| FMod_IPDCMOT_Write_ACCELERATION | Writes register ACCELERATION. |
| FMod_IPDCMOT_Read_DECELERATION | Reads register DECELERATION. |
| FMod_IPDCMOT_Write_DECELERATION | Writes register DECELERATION. |
| FMod_IPDCMOT_Read_TOPSPEED | Reads register TOPSPEED. |
| FMod_IPDCMOT_Write_TOPSPEED | Writes register TOPSPEED. |
| FMod_IPDCMOT_Read_DEADZONE | Reads register DEADZONE. |
| FMod_IPDCMOT_Write_DEADZONE | Writes register DEADZONE. |
| FMod_IPDCMOT_Read_HOMINGOPTIONS | Reads register HOMINGOPTIONS. |
| FMod_IPDCMOT_Write_HOMINGOPTIONS | Writes register HOMINGOPTIONS. |
| FMod_IPDCMOT_Write_HOMING | Writes register HOMING. |
| FMod_IPDCMOT_Write_STOPHOMING | Writes register STOPHOMING. |
| FMod_IPDCMOT_Read_HOMINGPOSITION | Reads register HOMINGPOSITION. |

| | |
|---|--|
| FMod_IPDCMOT_Write_HOMINGPOSITION | Writes register HOMINGPOSITION. |
| FMod_IPDCMOT_Read_HOMINGINPUT | Reads register HOMINGINPUT. |
| FMod_IPDCMOT_Write_HOMINGINPUT | Writes register HOMINGINPUT. |
| FMod_IPDCMOT_Read_LIMIT1SETUP | Reads register LIMIT1SETUP. |
| FMod_IPDCMOT_Write_LIMIT1SETUP | Writes register LIMIT1SETUP. |
| FMod_IPDCMOT_Read_LIMIT1REGULATIONMODE | Reads register LIMIT1REGULATIONMODE. |
| FMod_IPDCMOT_Write_LIMIT1REGULATIONMODE | Writes register LIMIT1REGULATIONMODE. |
| FMod_IPDCMOT_Read_LIMIT1POSITION | Reads register LIMIT1POSITION. |
| FMod_IPDCMOT_Write_LIMIT1POSITION | Writes register LIMIT1POSITION. |
| FMod_IPDCMOT_Read_LIMIT1XINPUT | Reads register LIMIT1XINPUT. |
| FMod_IPDCMOT_Write_LIMIT1XINPUT | Writes register LIMIT1XINPUT. |
| FMod_IPDCMOT_Read_LIMIT2SETUP | Reads register LIMIT2SETUP. |
| FMod_IPDCMOT_Write_LIMIT2SETUP | Writes register LIMIT2SETUP. |
| FMod_IPDCMOT_Read_LIMIT2REGULATIONMODE | Reads register LIMIT2REGULATIONMODE. |
| FMod_IPDCMOT_Write_LIMIT2REGULATIONMODE | Writes register LIMIT2REGULATIONMODE. |
| FMod_IPDCMOT_Read_LIMIT2POSITION | Reads register LIMIT2POSITION. |
| FMod_IPDCMOT_Write_LIMIT2POSITION | Writes register LIMIT2POSITION. |
| FMod_IPDCMOT_Read_LIMIT2XINPUT | Reads register LIMIT2XINPUT. |
| FMod_IPDCMOT_Write_LIMIT2XINPUT | Writes register LIMIT2XINPUT. |
| FMod_IPDCMOT_Read_AllRegister | Reads all registers. |
| FMod_IPDCMOT_ResetAllRegisterRead | Erases the list of registers to be read. |
| FMod_IPDCMOT_SendData_MAINPORT | Prepares and sends FiveCo packets to read and/or write the specified registers |

Repetitive asks Function

| Name of the function | Description |
|-----------------------------------|---|
| FMod_IPDCMOT_RepetitiveAskStart | Starts the repetitive asks of the specified registers. |
| FMod_IPDCMOT_RepetitiveAskStop | Stops the repetitive asks |
| FMod_IPDCMOT_RepetitiveAskSetTime | Sets the time between each repetitive asks. |
| FMod_IPDCMOT_RepetitiveAskReset | Erases the list of registers for the repetitive asks and Stops the repetitive asks. |

4. General Functions description

FMod_IPDCMOT_ScanNetwork

```
int    FMod_IPDCMOT_ScanNetwork(FMod_SModule *ModuleArray,
                                int ModuleArraySize);
```

Description

The *FMod_IPDCMOT_ScanNetwork* function retrieves the main information from all FiveCo's modules connected to the network.

Parameters

| | | |
|-----------------|-------|---|
| ModuleArray | [out] | Pointer on the first element of an array of FMod_SModule. |
| ModuleArraySize | [in] | Size of the array above. |

Return Values

Number of FiveCo modules connected on the network.

Notes

You have to create an array of FMod_SModule before you call *FMod_IPDCMOT_ScanNetwork*.

If the size of the array is smaller than the number of connected modules, you will only get information on the *ModuleArraySize* firsts modules. Check that the returned value is smaller or equal to the size of the array of FMod_SModule.

FMod_IPDCMOT_ChangeIPAddress

```
bool FMod_IPDCMOT_ChangeIPAddress(int MacAddress[6],
                                   int IPAddress[4], int SubnetMaskAddress[4]);
```

Description

The *FMod_IPDCMOT_ChangeIPAddress* function sets the IP address and the subnet mask of the module with the specified Mac address.

Parameters

| | | |
|-------------------|------|---|
| MacAddress | [in] | Mac address of the module to change IP address. |
| IPAddress | [in] | IP address to be set to the module. |
| SubnetMaskAddress | [in] | Subnet mask to be set to the module. |

Return Values

| | |
|-------|--|
| true | The function was completed successfully; both the IP address and the subnet mask are changed. |
| false | The function failed. See log file or call <i>FMod_IPDCMOT_GetLastError</i> to get error details. |

Notes

The *FMod_IPDCMOT_ChangeIPAddress* function takes some seconds to successfully complete.

FMod_IPDCMOT_VersionDLL

```
void FMod_IPDCMOT_VersionDLL(char* Version, int *Size);
```

Description

The *FMod_IPDCMOT_VersionDLL* function calls up the software version of the *FMod_IPDCMOT_DLLInterface.dll*.

Parameters

| | | |
|---------|-------|--|
| Version | [out] | Pointer on the first bytes of the buffer to receive the version. |
| Size | [in] | Size of the buffer <i>Version</i> . |
| | [out] | Nb of bytes copied in the buffer <i>Version</i> . |

Return Values

No return value.

Notes

The buffer *Version* must have a minimum size of 50 bytes in order to receive the entire DLL version.

FMod_IPDCMOT_GetStateCommunication

```
int    FMod_IPDCMOT_GetStateCommunication(void *ComID);
```

Description

The *FMod_IPDCMOT_GetStateCommunication* function calls up the current state of the communication.

Parameters

| | | |
|-------|------|-------------------------------|
| ComID | [in] | Pointer on the communication. |
|-------|------|-------------------------------|

Return Values

A 32-bit integer corresponding to the current state of the communication specified by the *ComID* pointer:

| | |
|---|---|
| 0 | The communication is closed. |
| 1 | The communication is opened. |
| 2 | An error occurred in the communication. |
| 3 | The communication is opening. |
| 4 | The communication is closing. |

In order to facilitate the use of *FMod_IPDCMOT_GetStateCommunication*, an enumeration of all the possible states is declared in the header file:

```
enum ComState { State_Closed=0, State_Opened, State_Error,
                State_Opening, State_Closing };
```

Notes

Don't use this function to verify the open state of a connection after calling up an open connection function. The Event function *Com_Event* is called up for all the main communication events.

FMod_IPDCMOT_GetLastError

```
void FMod_IPDCMOT_GetLastError (char* Error, int *Size);
```

Description

The *FMod_IPDCMOT_GetLastError* function returns the last error to have occurred in the *FMod_IPDCMOT_DLLInterface.dll*.

Parameters

| | | |
|-------|-------|--|
| Error | [out] | Pointer on the first Byte of the Buffer to write the description of the last error |
| Size | [in] | Size of the buffer <i>Error</i> . |
| | [out] | Nb of bytes copied in the buffer <i>Error</i> . |

Return Values

No return value.

Notes

A log file is created when an error occurs and is written to the same directory as the executable using the *FMod_IPDCMOT_DLLInterface.dll* is located.

5. MAIN Port functions descriptions

FMod_IPDCMOT_OpenConnection_MAINPORT

```
bool FMod_IPDCMOT_OpenConnection_MAINPORT(int AddressIP[4],
void (*Data_Received_MAINPORT)(FMod_SRegisterListRead
    *RegList, void *ComID),
void (*Com_Event)(int State, void *ComID),
void **ComID);
```

Description

The *FMod_IPDCMOT_OpenConnection_MAINPORT* function opens Ethernet communication on the Main Port with a specified IP Address. Its use will return a pointer (*ComID*).

Parameters

| | | |
|------------------------|-------|--|
| AddressIP | [in] | IP address in an array of 32bits integer values. |
| Data_Received_MAINPORT | [in] | Address of the <i>data receive</i> callback function. |
| Com_Event | [in] | Address of the <i>communication event</i> callback function. |
| ComID | [out] | Pointer on the opened communication. It is the reference for this communication and is used to call up most of the functions of the <i>FMod_IPDCMOT_DLLInterface.dll</i> . |

Return Values

| | |
|-------|--|
| true | The function was successfull, the communication is opening. |
| false | The function failed. See log file or call <i>FMod_IPDCMOT_GetLastError</i> to get error details. |

Notes

The communication is really opened only when the callback function *Com_Event* is called with *Com_State* = 1 (*State_Opened*).

If the communication is already opened (*ComID* not NULL), the communication will be closed and a new one created and opened. Before creating a new communication, make sure that the *ComID* pointer is NULL.

FMod_IPDCMOT_CloseConnection

```
bool FMod_IPDCMOT_CloseConnection (void **ComID);
```

Description

The *FMod_IPDCMOT_CloseConnection* function closes the Ethernet communication of the specified *ComID*.

Parameters

| | | |
|-------|------|-------------------------------|
| ComID | [in] | Pointer on the communication. |
|-------|------|-------------------------------|

Return Values

| | |
|------|---|
| true | The function is successful, the communication is closing. |
|------|---|

Notes

The communication is closed only when the callback function *Com_Event* is called through *Com_State* = 0 (*State_Closed*).

If *FMod_IPDCMOT_CloseConnection* is called with *ComID* = NULL, it will return true because the communication is already closed. However, the function *Com_Event* will not be called up.

FMod_IPDCMOT_Read_TYPE

bool FMod_IPDCMOT_Read_TYPE(void *ComID);

Description

The *FMod_IPDCMOT_Read_TYPE* function prepares the DLL to read the *TYPE* (0x00) register next time *FMod_IPDCMOT_SendData_MAINPORT* is called.

Parameters

ComID [in] Pointer on the communication.

Return Values

| | |
|-------|--|
| true | The function is successful. |
| false | The function failed. See log file or call <i>FMod_IPDCMOT_GetLastError</i> to get error details. |

FMod_IPDCMOT_Read_VERSION

```
bool FMod_IPDCMOT_Read_VERSION(void *ComID);
```

Description

The *FMod_IPDCMOT_Read_VERSION* function prepares the DLL to read the *VERSION* (0x01) register next time *FMod_IPDCMOT_SendData_MAINPORT* is called.

Parameters

| | | |
|-------|------|-------------------------------|
| ComID | [in] | Pointer on the communication. |
|-------|------|-------------------------------|

Return Values

| | |
|-------|--|
| true | The function is successful. |
| false | The function failed. See log file or call <i>FMod_IPDCMOT_GetLastError</i> to get error details. |

FMod_IPDCMOT_Write_RESETCPU

```
bool FMod_IPDCMOT_Write_RESETCPU(void *ComID);
```

Description

The *FMod_IPDCMOT_Write_RESETCPU* function prepares the DLL to write to the *RESETCPU* (0x02) register next time *FMod_IPDCMOT_SendData_MAINPORT* is called.

Parameters

| | |
|-------|------------------------------------|
| ComID | [in] Pointer on the communication. |
|-------|------------------------------------|

Return Values

| | |
|-------|--|
| true | The function is successful. |
| false | The function failed. See log file or call <i>FMod_IPDCMOT_GetLastError</i> to get error details. |

FMod_IPDCMOT_Write_SAVEUSERPARAMETERS

bool FMod_IPDCMOT_Write_SAVEUSERPARAMETERS(void *ComID);

Description

The *FMod_IPDCMOT_Write_SAVEUSERPARAMETERS* function prepares the DLL to launch the *SAVEUSERPARAMETERS* (0x03) function next time *FMod_IPDCMOT_SendData_MAINPORT* is called.

Parameters

| | | |
|-------|------|-------------------------------|
| ComID | [in] | Pointer on the communication. |
|-------|------|-------------------------------|

Return Values

| | |
|-------|--|
| true | The function is successful. |
| false | The function failed. See log file or call <i>FMod_IPDCMOT_GetLastError</i> to get error details. |

FMod_IPDCMOT_Write_RESTOREUSERPARAMETERS

```
bool FMod_IPDCMOT_Write_RESTOREUSERPARAMETERS(void  
*ComID);
```

Description

The *FMod_IPDCMOT_Write_RESTOREUSERPARAMETERS* function prepares the DLL to launch the *RESTOREUSERPARAMETERS* (0x04) function next time *FMod_IPDCMOT_SendData_MAINPORT* is called.

Parameters

| | | |
|-------|------|-------------------------------|
| ComID | [in] | Pointer on the communication. |
|-------|------|-------------------------------|

Return Values

| | |
|-------|--|
| true | The function is successful. |
| false | The function failed. See log file or call <i>FMod_IPDCMOT_GetLastError</i> to get error details. |

FMod_IPDCMOT_Write_RESTOREFACTORYPARAMETERS

```
bool FMod_IPDCMOT_Write_RESTOREFACTORYPARAMETERS(
    void *ComID);
```

Description

The *FMod_IPDCMOT_Write_RESTOREFACTORYPARAMETERS* function prepares the DLL to launch the *RESTOREFACTORYPARAMETERS* (0x05) function next time *FMod_IPDCMOT_SendData_MAINPORT* is called.

Parameters

| | | |
|-------|------|-------------------------------|
| ComID | [in] | Pointer on the communication. |
|-------|------|-------------------------------|

Return Values

| | |
|-------|--|
| true | The function is successful. |
| false | The function failed. See log file or call <i>FMod_IPDCMOT_GetLastError</i> to get error details. |

FMod_IPDCMOT_Write_SAVEFACTORYPARAMETERS

```
bool FMod_IPDCMOT_Write_SAVEFACTORYPARAMETERS(
    void *ComID);
```

Description

The *FMod_IPDCMOT_Write_SAVEFACTORYPARAMETERS* function prepares the DLL to launch the *RESTOREUSERPARAMETERS* (0x06) function next time *FMod_IPDCMOT_SendData_MAINPORT* is called.

Parameters

| | | |
|-------|------|-------------------------------|
| ComID | [in] | Pointer on the communication. |
|-------|------|-------------------------------|

Return Values

| | |
|-------|--|
| true | The function is successful. |
| false | The function failed. See log file or call <i>FMod_IPDCMOT_GetLastError</i> to get error details. |

FMod_IPDCMOT_Read_VOLTAGE

```
bool FMod_IPDCMOT_Read_VOLTAGE(bool Continuous, void *ComID);
```

Description

The *FMod_IPDCMOT_Read_VOLTAGE* function prepares the DLL to read the *VOLTAGE* (0x07) register next time *FMod_IPDCMOT_SendData_MAINPORT* is called.

Parameters

| | | |
|-------|------|-------------------------------|
| ComID | [in] | Pointer on the communication. |
|-------|------|-------------------------------|

Return Values

| | |
|-------|--|
| true | The function is successful. |
| false | The function failed. See log file or call <i>FMod_IPDCMOT_GetLastError</i> to get error details. |

FMod_IPDCMOT_Read_WARNINGS

bool FMod_IPDCMOT_Read_WARNINGS(bool Continous, void *ComID);

Description

The *FMod_IPDCMOT_Read_WARNINGS* function prepares the DLL to read the *WARNINGS* (0x08) register next time *FMod_IPDCMOT_SendData_MAINPORT* is called.

Parameters

ComID [in] Pointer on the communication.

Return Values

| | |
|-------|--|
| true | The function is successful. |
| false | The function failed. See log file or call <i>FMod_IPDCMOT_GetLastError</i> to get error details. |

FMod_IPDCMOT_Reset_WARNINGS

```
bool FMod_IPDCMOT_Reset_WARNINGS(void *ComID);
```

Description

The *FMod_IPDCMOT_Reset_WARNINGS* function prepares the DLL to Reset the *WARNINGS* (0x08) register next time *FMod_IPDCMOT_SendData_MAINPORT* is called.

Parameters

ComID [in] Pointer on the communication.

Return Values

| | |
|-------|--|
| true | The function is successful. |
| false | The function failed. See log file or call <i>FMod_IPDCMOT_GetLastError</i> to get error details. |

Notes

Each Bit of the *WARNINGS* register is set to 0.

FMod_IPDCMOT_Read_COMMUNICATIONOPTIONS

bool FMod_IPDCMOT_Read_COMMUNICATIONOPTIONS(void *ComID);

Description

The *FMod_IPDCMOT_Read_COMMUNICATIONOPTIONS* function prepares the DLL to read the *COMMUNICATIONOPTIONS* (0x10) register next time *FMod_IPDCMOT_SendData_MAINPORT* is called.

Parameters

| | | |
|-------|------|-------------------------------|
| ComID | [in] | Pointer on the communication. |
|-------|------|-------------------------------|

Return Values

| | |
|-------|--|
| true | The function is successful. |
| false | The function failed. See log file or call <i>FMod_IPDCMOT_GetLastError</i> to get error details. |

FMod_IPDCMOT_Write_COMMUNICATIONOPTIONS

```
bool FMod_IPDCMOT_Write_COMMUNICATIONOPTIONS(
    bool ComOpt[32], void *ComID);
```

Description

The *FMod_IPDCMOT_Write_COMMUNICATIONOPTIONS* function prepares the DLL to write to the *COMMUNICATIONOPTIONS* (0x10) register next time *FMod_IPDCMOT_SendData_MAINPORT* is called.

Parameters

| | | |
|--------|------|-------------------------------|
| ComOpt | [in] | Array of 32 Boolean value. |
| ComID | [in] | Pointer on the communication. |

Return Values

| | |
|-------|--|
| true | The function is successful. |
| false | The function failed. See log file or call <i>FMod_IPDCMOT_GetLastError</i> to get error details. |

Notes

Each Boolean value refers to one bit of the register value with the following rules:

- true → bit = 1.
- false → bit = 0.

FMod_IPDCMOT_Read_ETHERNETMAC

bool FMod_IPDCMOT_Read_ETHERNETMAC(void *ComID);

Description

The *FMod_IPDCMOT_Read_ETHERNETMAC* function prepares the DLL to read the *ETHERNETMAC* (0x11) register next time *FMod_IPDCMOT_SendData_MAINPORT* is called.

Parameters

ComID [in] Pointer on the communication.

Return Values

| | |
|-------|--|
| true | The function is successful. |
| false | The function failed. See log file or call <i>FMod_IPDCMOT_GetLastError</i> to get error details. |

FMod_IPDCMOT_Read_IPADDRESS

```
bool FMod_IPDCMOT_Read_IPADDRESS(void *ComID);
```

Description

The *FMod_IPDCMOT_Read_IPADDRESS* function prepares the DLL to read the *IPADDRESS* (0x12) register next time *FMod_IPDCMOT_SendData_MAINPORT* is called.

Parameters

| | | |
|-------|------|-------------------------------|
| ComID | [in] | Pointer on the communication. |
|-------|------|-------------------------------|

Return Values

| | |
|-------|--|
| true | The function is successful. |
| false | The function failed. See log file or call <i>FMod_IPDCMOT_GetLastError</i> to get error details. |

FMod_IPDCMOT_Write_IPADDRESS

```
bool FMod_IPDCMOT_Write_IPADDRESS(int IPAdd[4], void *ComID);
```

Description

The *FMod_IPDCMOT_Write_IPADDRESS* function prepares the DLL to write to the *IPADDRESS* (0x12) register next time *FMod_IPDCMOT_SendData_MAINPORT* is called.

Parameters

| | | |
|-------|------|-------------------------------------|
| IPAdd | [in] | Array of 4 "32 bits Integer value". |
| ComID | [in] | Pointer on the communication. |

Return Values

| | |
|-------|--|
| true | The function is successful. |
| false | The function failed. See log file or call <i>FMod_IPDCMOT_GetLastError</i> to get error details. |

Notes

The IPAdd array must contain valid value for an IP address.
 $0 \leq \text{value} \leq 255$.

FMod_IPDCMOT_Read_SUBNETMASK

```
bool FMod_IPDCMOT_Read_SUBNETMASK(void *ComID);
```

Description

The *FMod_IPDCMOT_Read_SUBNETMASK* function prepares the DLL to read the *SUBNETMASK* (0x13) register next time *FMod_IPDCMOT_SendData_MAINPORT* is called.

Parameters

| | | |
|-------|------|-------------------------------|
| ComID | [in] | Pointer on the communication. |
|-------|------|-------------------------------|

Return Values

| | |
|-------|--|
| true | The function is successful. |
| false | The function failed. See log file or call <i>FMod_IPDCMOT_GetLastError</i> to get error details. |

FMod_IPDCMOT_Write_SUBNETMASK

```
bool FMod_IPDCMOT_Write_SUBNETMASK(int SubnetMask[4],
                                     void *ComID);
```

Description

The *FMod_IPDCMOT_Write_SUBNETMASK* function prepares the DLL to write to the *SUBNETMASK* (0x13) register next time *FMod_IPDCMOT_SendData_MAINPORT* is called.

Parameters

| | | |
|------------|------|-------------------------------------|
| SubnetMask | [in] | Array of 4 "32 bits Integer value". |
| ComID | [in] | Pointer on the communication. |

Return Values

| | |
|-------|--|
| true | The function is successful. |
| false | The function failed. See log file or call <i>FMod_IPDCMOT_GetLastError</i> to get error details. |

Notes

The SubnetMask array must contain valid value for an IP address.
 $0 \leq \text{value} \leq 255$.

FMod_IPDCMOT_Read_TCPTIMEOUT

```
bool FMod_IPDCMOT_Read_TCPTIMEOUT(void *ComID);
```

Description

The *FMod_IPDCMOT_Read_TCPTIMEOUT* function prepares the DLL to read the *TCPTIMEOUT* (0x14) register next time *FMod_IPDCMOT_SendData_MAINPORT* is called.

Parameters

| | | |
|-------|------|-------------------------------|
| ComID | [in] | Pointer on the communication. |
|-------|------|-------------------------------|

Return Values

| | |
|-------|--|
| true | The function is successful. |
| false | The function failed. See log file or call <i>FMod_IPDCMOT_GetLastError</i> to get error details. |

FMod_IPDCMOT_Write_TCPTIMEOUT

```
bool FMod_IPDCMOT_Write_TCPTIMEOUT(int TCPTimeOut,
                                     void *ComID);
```

Description

The *FMod_IPDCMOT_Write_TCPTIMEOUT* function prepares the DLL to write to the *TCPTIMEOUT* (0x14) register next time *FMod_IPDCMOT_SendData_MAINPORT* is called.

Parameters

| | | |
|------------|------|-------------------------------|
| TCPTimeOut | [in] | 32 bits Integer value. |
| ComID | [in] | Pointer on the communication. |

Return Values

| | |
|-------|--|
| true | The function is successful. |
| false | The function failed. See log file or call <i>FMod_IPDCMOT_GetLastError</i> to get error details. |

Notes

The *TCPTimeOut* value must contain a valid value between 0 and 255.

FMod_IPDCMOT_Read_MODULENAME

```
bool FMod_IPDCMOT_Read_MODULENAME(void *ComID);
```

Description

The *FMod_IPDCMOT_Read_MODULENAME* function prepares the DLL to read the *MODULENAME* (0x15) register next time *FMod_IPDCMOT_SendData_MAINPORT* is called.

Parameters

| | |
|-------|------------------------------------|
| ComID | [in] Pointer on the communication. |
|-------|------------------------------------|

Return Values

| | |
|-------|--|
| true | The function is successful. |
| false | The function failed. See log file or call <i>FMod_IPDCMOT_GetLastError</i> to get error details. |

FMod_IPDCMOT_Write_MODULENAME

```
bool FMod_IPDCMOT_Write_MODULENAME(char Name[16],
                                     void *ComID);
```

Description

The *FMod_IPDCMOT_Write_MODULENAME* function prepares the DLL to write to the *MODULENAME* (0x15) register next time *FMod_IPDCMOT_SendData_MAINPORT* is called.

Parameters

| | | |
|-------|------|----------------------------------|
| Name | [in] | Array of 16 char (1 Byte) value. |
| ComID | [in] | Pointer on the communication. |

Return Values

| | |
|-------|--|
| true | The function is successful. |
| false | The function failed. See log file or call <i>FMod_IPDCMOT_GetLastError</i> to get error details. |

FMod_IPDCMOT_Read_TCPCONNECTIONSOPENED

```
bool FMod_IPDCMOT_Read_TCPCONNECTIONSOPENED(void *ComID);
```

Description

The *FMod_IPDCMOT_Read_TCPCONNECTIONSOPENED* function prepares the DLL to read the *TCPCONNECTIONSOPENED* (0x1A) register next time *FMod_IPDCMOT_SendData_MAINPORT* is called.

Parameters

| | | |
|-------|------|-------------------------------|
| ComID | [in] | Pointer on the communication. |
|-------|------|-------------------------------|

Return Values

| | |
|-------|--|
| true | The function is successful. |
| false | The function failed. See log file or call <i>FMod_IPDCMOT_GetLastError</i> to get error details. |

FMod_IPDCMOT_Read_REGULATIONMODE

```
bool FMod_IPDCMOT_Read_REGULATIONMODE(bool Continuous,
    void *ComID);
```

Description

The *FMod_IPDCMOT_Read_REGULATIONMODE* function prepares the DLL to read the *REGULATIONMODE* (0x20) register next time *FMod_IPDCMOT_SendData_MAINPORT* is called.

Parameters

| | | |
|------------|------|--|
| Continuous | [in] | Boolean for repetitive ask of this register. |
| ComID | [in] | Pointer on the communication. |

Return Values

| | |
|-------|--|
| true | The function is successful. |
| false | The function failed. See log file or call <i>FMod_IPDCMOT_GetLastError</i> to get error details. |

FMod_IPDCMOT_Write_REGULATIONMODE

```
bool FMod_IPDCMOT_Write_REGULATIONMODE(int RegMode,
void *ComID);
```

Description

The *FMod_IPDCMOT_Write_REGULATIONMODE* function prepares the DLL to write to the *REGULATIONMODE* (0x20) register next time *FMod_IPDCMOT_SendData_MAINPORT* is called.

Parameters

| | | |
|---------|------|-------------------------------|
| RegMode | [in] | 32 bits integer value. |
| ComID | [in] | Pointer on the communication. |

Return Values

| | |
|-------|--|
| true | The function is successful. |
| false | The function failed. See log file or call <i>FMod_IPDCMOT_GetLastError</i> to get error details. |

Notes

An enumerator in the header file *FMod_IPDCMOT_DLLInterface.h* lists all the possible values of *RegMode*:

```
enum    RegulationMode_Enum
{
    RegMode_Brake = 0,           // brake and stop the motor
    RegMode_DriverOpen,         // disconnect motor pins from ground and power
                                // supply
    RegMode_OpenLoop,           // Input reg is directly converted to PWM
                                // outputs
    RegMode_WaitMode,            // continue actual output (PWM), without
                                // regulation
    RegMode_SpeedControl,        // acceleration to Input speed, with PID
                                // algorithm
    RegMode_PositionControl      // acceleration, top speed and deceleration
                                // ramps with PID
};
```

FMod_IPDCMOT_Read_INPUT

```
bool FMod_IPDCMOT_Read_INPUT(bool Continuous, void *ComID);
```

Description

The *FMod_IPDCMOT_Read_INPUT* function prepares the DLL to read the *INPUT* (0x21) register next time *FMod_IPDCMOT_SendData_MAINPORT* is called.

Parameters

| | | |
|------------|------|--|
| Continuous | [in] | Boolean for repetitive ask of this register. |
| ComID | [in] | Pointer on the communication. |

Return Values

| | |
|-------|--|
| true | The function is successful. |
| false | The function failed. See log file or call <i>FMod_IPDCMOT_GetLastError</i> to get error details. |

FMod_IPDCMOT_Write_INPUT

```
bool FMod_IPDCMOT_Write_INPUT(int Input, void *ComID);
```

Description

The *FMod_IPDCMOT_Write_INPUT* function prepares the DLL to write to the *INPUT* (0x21) register next time *FMod_IPDCMOT_SendData_MAINPORT* is called.

Parameters

| | | |
|-------|------|-------------------------------|
| Input | [in] | 32 bits integer value. |
| ComID | [in] | Pointer on the communication. |

Return Values

| | |
|-------|--|
| true | The function is successful. |
| false | The function failed. See log file or call <i>FMod_IPDCMOT_GetLastError</i> to get error details. |

Notes

Active :

When *REGULATIONMODE* = Open-Loop, Position Control, Speed Control, or Wait-Mode

REGULATIONMODE = Open-Loop:

INPUT unit is % of PWM

| | | |
|-----|--------------------|--------------------------|
| Max | 0x0000FFFF = 65535 | PWM = 100% (65535/65536) |
|-----|--------------------|--------------------------|

| | | |
|------|----------------|--------------------|
| Zero | 0x00000000 = 0 | PWM = 0% (0/65536) |
|------|----------------|--------------------|

| | | |
|-----|---------------------|----------------------------|
| Min | 0xFFFF0001 = -65535 | PWM = -100% (-65535/65536) |
|-----|---------------------|----------------------------|

Higher and lower values are automatically saturated when converted to PWM

REGULATIONMODE = Position Control

INPUT is the position to reach, and the unit is pulses

| | |
|-----|----------------------------|
| Max | 0x7FFFFFFF = 2'147'483'647 |
|-----|----------------------------|

| | |
|-----|-----------------------------|
| Min | 0x80000000 = -2'147'483'648 |
|-----|-----------------------------|

REGULATIONMODE = Speed Control

INPUT is the speed to reach, and the unit is pulses/sec

| | |
|-----|----------------------------|
| Max | 0x7FFFFFFF = 2'147'483'647 |
|-----|----------------------------|

| | |
|-----|-----------------------------|
| Min | 0x80000000 = -2'147'483'648 |
|-----|-----------------------------|

Default:

0x00000000 = 0

FMod_IPDCMOT_Write_INPUTOFFSET

```
bool FMod_IPDCMOT_Write_INPUTOFFSET(int InputOffset,
                                     void *ComID);
```

Description

The *FMod_IPDCMOT_Write_INPUTOFFSET* function prepares the DLL to write to the *INPUTOFFSET* (0x22) register next time *FMod_IPDCMOT_SendData_MAINPORT* is called.

Parameters

| | | |
|-------------|------|-------------------------------|
| InputOffset | [in] | 32 bits integer value. |
| ComID | [in] | Pointer on the communication. |

Return Values

| | |
|-------|--|
| true | The function is successful. |
| false | The function failed. See log file or call <i>FMod_IPDCMOT_GetLastError</i> to get error details. |

Notes

Active:

When HOMING is not running, else it is ignored and cleared.

Limits:

Max 0x7FFFFFFF = 2'147'483'647

Min 0x80000000 = -2'147'483'648

Default:

0x00000000 = 0

FMod_IPDCMOT_Write_INPUTOFFSETMEASURED

```
bool FMod_IPDCMOT_Write_INPUTOFFSETMEASURED(
    int InputOffsetMeasured, void *ComID);
```

Description

The *FMod_IPDCMOT_Write_INPUTOFFSETMEASURED* function prepares the DLL to write to the *INPUTOFFSETMEASURED* (0x23) register next time *FMod_IPDCMOT_SendData_MAINPORT* is called.

Parameters

| | | |
|---------------------|------|-------------------------------|
| InputOffsetMeasured | [in] | 32 bits integer value. |
| ComID | [in] | Pointer on the communication. |

Return Values

| | |
|-------|--|
| true | The function is successful. |
| false | The function failed. See log file or call <i>FMod_IPDCMOT_GetLastError</i> to get error details. |

Notes

Active:

Only when its value is not equal to 0.

When HOMING is not running, else it is ignored and cleared.

Limits:

Max 0x7FFFFFFF = 2'147'483'647

Min 0x80000000 = -2'147'483'648

Default:

0x00000000 = 0

FMod_IPDCMOT_Read_INPUTMIN

```
bool FMod_IPDCMOT_Read_INPUTMIN(void *ComID);
```

Description

The *FMod_IPDCMOT_Read_INPUTMIN* function prepares the DLL to read the *INPUTMIN* (0x24) register next time *FMod_IPDCMOT_SendData_MAINPORT* is called.

Parameters

| | | |
|-------|------|-------------------------------|
| ComID | [in] | Pointer on the communication. |
|-------|------|-------------------------------|

Return Values

| | |
|-------|--|
| true | The function is successful. |
| false | The function failed. See log file or call <i>FMod_IPDCMOT_GetLastError</i> to get error details. |

FMod_IPDCMOT_Write_INPUTMIN

```
bool FMod_IPDCMOT_Write_INPUTMIN (int InputMin, void *ComID);
```

Description

The *FMod_IPDCMOT_Write_INPUTMIN* function prepares the DLL to write to the *INPUTMIN* (0x24) register next time *FMod_IPDCMOT_SendData_MAINPORT* is called.

Parameters

| | | |
|----------|------|-------------------------------|
| InputMin | [in] | 32 bits integer value. |
| ComID | [in] | Pointer on the communication. |

Return Values

| | |
|-------|--|
| true | The function is successful. |
| false | The function failed. See log file or call <i>FMod_IPDCMOT_GetLastError</i> to get error details. |

Notes

Active:

Each time the processor is running, but ignored during homing!

Limits:

Max 0x7FFFFFFF = 2'147'483'647

Min 0x80000000 = -2'147'483'648

Default:

Min -> never influences *INPUT* value

FMod_IPDCMOT_Read_INPUTMAX

```
bool FMod_IPDCMOT_Read_INPUTMAX(void *ComID);
```

Description

The *FMod_IPDCMOT_Read_INPUTMAX* function prepares the DLL to read the *INPUTMAX* (0x25) register next time *FMod_IPDCMOT_SendData_MAINPORT* is called.

Parameters

| | | |
|-------|------|-------------------------------|
| ComID | [in] | Pointer on the communication. |
|-------|------|-------------------------------|

Return Values

| | |
|-------|--|
| true | The function is successful. |
| false | The function failed. See log file or call <i>FMod_IPDCMOT_GetLastError</i> to get error details. |

FMod_IPDCMOT_Write_INPUTMAX

```
bool FMod_IPDCMOT_Write_INPUTMAX (int InputMax, void *ComID);
```

Description

The *FMod_IPDCMOT_Write_INPUTMAX* function prepares the DLL to write to the *INPUTMAX* (0x25) register next time *FMod_IPDCMOT_SendData_MAINPORT* is called.

Parameters

| | | |
|----------|------|-------------------------------|
| InputMax | [in] | 32 bits integer value. |
| ComID | [in] | Pointer on the communication. |

Return Values

| | |
|-------|--|
| true | The function is successful. |
| false | The function failed. See log file or call <i>FMod_IPDCMOT_GetLastError</i> to get error details. |

Notes

Active:

Each time the processor is running, but ignored during homing!

Limits:

Max 0x7FFFFFFF = 2'147'483'647

Min 0x80000000 = -2'147'483'648

Default:

Min -> never influences *INPUT* value

FMod_IPDCMOT_Read_POSITION

```
bool FMod_IPDCMOT_Read_POSITION(bool Continuous, void *ComID);
```

Description

The *FMod_IPDCMOT_Read_POSITION* function prepares the DLL to read the *POSITION* (0x26) register next time *FMod_IPDCMOT_SendData_MAINPORT* is called.

Parameters

| | | |
|------------|------|--|
| Continuous | [in] | Boolean for repetitive ask of this register. |
| ComID | [in] | Pointer on the communication. |

Return Values

| | |
|-------|--|
| true | The function is successful. |
| false | The function failed. See log file or call <i>FMod_IPDCMOT_GetLastError</i> to get error details. |

FMod_IPDCMOT_Write_POSITION

```
bool FMod_IPDCMOT_Write_POSITION (int Position, void *ComID);
```

Description

The *FMod_IPDCMOT_Write_POSITION* function prepares the DLL to write to the *POSITION* (0x26) register next time *FMod_IPDCMOT_SendData_MAINPORT* is called.

Parameters

| | | |
|----------|------|-------------------------------|
| Position | [in] | 32 bits integer value. |
| ComID | [in] | Pointer on the communication. |

Return Values

| | |
|-------|--|
| true | The function is successful. |
| false | The function failed. See log file or call <i>FMod_IPDCMOT_GetLastError</i> to get error details. |

Notes

Active:

Each time the processor is running

Limits:

Max 0x7FFFFFFF = 2'147'483'647

Min 0x80000000 = -2'147'483'648

Default:

After Power ON, *POSITION* is cleared (0x00000000). If *OPTIONS.6* bit is set, the *POSITION* used before the last shut down is reloaded.

FMod_IPDCMOT_Write_POSITIONOFFSET

```
bool FMod_IPDCMOT_Write_POSITIONOFFSET (int PositionOffset,
                                         void *ComID);
```

Description

The *FMod_IPDCMOT_Write_POSITIONOFFSET* function prepares the DLL to write to the *POSITIONOFFSET* (0x27) register next time *FMod_IPDCMOT_SendData_MAINPORT* is called.

Parameters

| | | |
|----------------|------|-------------------------------|
| PositionOffset | [in] | 32 bits integer value. |
| ComID | [in] | Pointer on the communication. |

Return Values

| | |
|-------|--|
| true | The function is successful. |
| false | The function failed. See log file or call <i>FMod_IPDCMOT_GetLastError</i> to get error details. |

Notes

Active:

Each time the processor is running

Limits:

Max 0x7FFFFFFF = 2'147'483'647

Min 0x80000000 = -2'147'483'648

FMod_IPDCMOT_Read_SPEED

```
bool FMod_IPDCMOT_Read_SPEED(bool Continuous, void *ComID);
```

Description

The *FMod_IPDCMOT_Read_SPEED* function prepares the DLL to read the *SPEED* (0x28) register next time *FMod_IPDCMOT_SendData_MAINPORT* is called.

Parameters

| | | |
|------------|------|--|
| Continuous | [in] | Boolean for repetitive ask of this register. |
| ComID | [in] | Pointer on the communication. |

Return Values

| | |
|-------|--|
| true | The function is successful. |
| false | The function failed. See log file or call <i>FMod_IPDCMOT_GetLastError</i> to get error details. |

FMod_IPDCMOT_Read_CURRENTMAX

```
bool FMod_IPDCMOT_Read_CURRENTMAX(void *ComID);
```

Description

The *FMod_IPDCMOT_Read_CURRENTMAX* function prepares the DLL to read the *CURRENTMAX* (0x2A) register next time *FMod_IPDCMOT_SendData_MAINPORT* is called.

Parameters

| | | |
|-------|------|-------------------------------|
| ComID | [in] | Pointer on the communication. |
|-------|------|-------------------------------|

Return Values

| | |
|-------|--|
| true | The function is successful. |
| false | The function failed. See log file or call <i>FMod_IPDCMOT_GetLastError</i> to get error details. |

FMod_IPDCMOT_Write_CURRENTMAX

```
bool FMod_IPDCMOT_Write_CURRENTMAX(float CurrentMax,
                                     void *ComID);
```

Description

The *FMod_IPDCMOT_Write_CURRENTMAX* function prepares the DLL to write to the *CURRENTMAX* (0x2A) register next time *FMod_IPDCMOT_SendData_MAINPORT* is called.

Parameters

| | | |
|------------|------|------------------------------------|
| CurrentMax | [in] | 32 bits floating-point data value. |
| ComID | [in] | Pointer on the communication. |

Return Values

| | |
|-------|--|
| true | The function is successful. |
| false | The function failed. See log file or call <i>FMod_IPDCMOT_GetLastError</i> to get error details. |

Notes

Active:

With every *REGULATIONMODE*. During WaitMode, *CURRENTMAX* is not refreshed.

Limits:

| | |
|------|--|
| Max | 0x000F0000 = 15.0 A (limited by <i>TEMPERATURE</i>) |
| Min | 0x00000000 = 0.0 A |
| Step | 0x00000640 = 0.025A |

Default:

| | |
|-------|--|
| Other | 0x00050000 = 327680, current limitation 5 A (327680/65536) |
|-------|--|

FMod_IPDCMOT_Read_OPTIONS

bool FMod_IPDCMOT_Read_OPTIONS(void *ComID);

Description

The *FMod_IPDCMOT_Read_OPTIONS* function prepares the DLL to read the *OPTIONS* (0x2C) register next time *FMod_IPDCMOT_SendData_MAINPORT* is called.

Parameters

ComID [in] Pointer on the communication.

Return Values

| | |
|-------|--|
| true | The function is successful. |
| false | The function failed. See log file or call <i>FMod_IPDCMOT_GetLastError</i> to get error details. |

FMod_IPDCMOT_Write_OPTIONS

```
bool FMod_IPDCMOT_Write_OPTIONS(bool Options[32],
                                void *ComID);
```

Description

The *FMod_IPDCMOT_Write_OPTIONS* function prepares the DLL to write to the *OPTIONS* (0x2C) register next time *FMod_IPDCMOT_SendData_MAINPORT* is called.

Parameters

| | | |
|---------|------|-------------------------------|
| Options | [in] | Array of 32 Boolean value. |
| ComID | [in] | Pointer on the communication. |

Return Values

| | |
|-------|--|
| true | The function is successful. |
| false | The function failed. See log file or call <i>FMod_IPDCMOT_GetLastError</i> to get error details. |

Notes

Each Boolean value refers to one bit of the register value with the following rules:

- true → bit = 1.
- false → bit = 0.

FMod_IPDCMOT_Read_LOOPTIME

```
bool FMod_IPDCMOT_Read_LOOPTIME(void *ComID);
```

Description

The *FMod_IPDCMOT_Read_LOOPTIME* function prepares the DLL to read the *LOOPTIME* (0x2D) register next time *FMod_IPDCMOT_SendData_MAINPORT* is called.

Parameters

| | | |
|-------|------|-------------------------------|
| ComID | [in] | Pointer on the communication. |
|-------|------|-------------------------------|

Return Values

| | |
|-------|--|
| true | The function is successful. |
| false | The function failed. See log file or call <i>FMod_IPDCMOT_GetLastError</i> to get error details. |

FMod_IPDCMOT_Write_LOOPTIME

```
bool FMod_IPDCMOT_Write_LOOPTIME (int LoopTime, void *ComID);
```

Description

The *FMod_IPDCMOT_Write_LOOPTIME* function prepares the DLL to write to the *LOPTIME* (0x2D) register next time *FMod_IPDCMOT_SendData_MAINPORT* is called.

Parameters

| | | |
|----------|------|-------------------------------|
| LoopTime | [in] | 32 bits integer value. |
| ComID | [in] | Pointer on the communication. |

Return Values

| | |
|-------|--|
| true | The function is successful. |
| false | The function failed. See log file or call <i>FMod_IPDCMOT_GetLastError</i> to get error details. |

Notes

Active:

Each time the processor is running.

Value Time Refresh rate

| | |
|------|-----------------|
| 0x00 | 50 ms, 20 Hz |
| 0x01 | 20 ms, 50 Hz |
| 0x02 | 10 ms, 100 Hz |
| 0x03 | 5 ms, 200 Hz |
| 0x04 | 2 ms, 500 Hz |
| 0x05 | 1 ms, 1000 Hz |
| 0x06 | 500 us, 2000 Hz |

Limits:

Max: 6

Default:

6 (500 us, 2000 Hz)

FMod_IPDCMOT_Read_OUTPUTVOLTAGEMAX

```
bool FMod_IPDCMOT_Read_OUTPUTVOLTAGEMAX(void *ComID);
```

Description

The *FMod_IPDCMOT_Read_OUTPUTVOLTAGEMAX* function prepares the DLL to read the *OUTPUTVOLTAGEMAX* (0x2E) register next time *FMod_IPDCMOT_SendData_MAINPORT* is called.

Parameters

| | | |
|-------|------|-------------------------------|
| ComID | [in] | Pointer on the communication. |
|-------|------|-------------------------------|

Return Values

| | |
|-------|--|
| true | The function is successful. |
| false | The function failed. See log file or call <i>FMod_IPDCMOT_GetLastError</i> to get error details. |

FMod_IPDCMOT_Write_OUTPUTVOLTAGEMAX

```
bool FMod_IPDCMOT_Write_OUTPUTVOLTAGEMAX(
    float OutPutVoltageMax, void *ComID);
```

Description

The *FMod_IPDCMOT_Write_CURRENTMAX* function prepares the DLL to write to the *CURRENTMAX* (0x2A) register next time *FMod_IPDCMOT_SendData_MAINPORT* is called.

Parameters

| | | |
|------------------|------|------------------------------------|
| OutPutVoltageMax | [in] | 32 bits floating-point data value. |
| ComID | [in] | Pointer on the communication. |

Return Values

| | |
|-------|--|
| true | The function is successful. |
| false | The function failed. See log file or call <i>FMod_IPDCMOT_GetLastError</i> to get error details. |

Notes

Active:

Each time the processor is running.

Limits:

| | |
|----------|--|
| Max | 0x7FFFFFFFxx = 32'767.996 |
| Min | 0x000000xx = 0.0, but it is strongly recommended that <i>OUTPUTVOLTAGEMAX > VOLTAGE / 2</i> |
| Step | 0x000001xx = 0.004 |
| Disabled | 0xFFFFFFFF, with this value, correction is never made |

Default:

| | |
|----------|------------|
| Disabled | 0xFFFFFFFF |
|----------|------------|

*FMod_IPDCMOT_Read_DESIRE*D

```
bool FMod_IPDCMOT_Read_DESIRED(bool Continuous, void *ComID);
```

Description

The *FMod_IPDCMOT_Read_DESIRED* function prepares the DLL to read the *DESIRE*D (0x30) register next time *FMod_IPDCMOT_SendData_MAINPORT* is called.

Parameters

| | | |
|------------|------|--|
| Continuous | [in] | Boolean for repetitive ask of this register. |
| ComID | [in] | Pointer on the communication. |

Return Values

| | |
|-------|--|
| true | The function is successful. |
| false | The function failed. See log file or call <i>FMod_IPDCMOT_GetLastError</i> to get error details. |

FMod_IPDCMOT_Read_FEEDBACK

```
bool FMod_IPDCMOT_Read_FEEDBACK(bool Continuous,
                                  void *ComID);
```

Description

The *FMod_IPDCMOT_Read_FEEDBACK* function prepares the DLL to read the *FEEDBACK* (0x31) register next time *FMod_IPDCMOT_SendData_MAINPORT* is called.

Parameters

| | | |
|------------|------|--|
| Continuous | [in] | Boolean for repetitive ask of this register. |
| ComID | [in] | Pointer on the communication. |

Return Values

| | |
|-------|--|
| true | The function is successful. |
| false | The function failed. See log file or call <i>FMod_IPDCMOT_GetLastError</i> to get error details. |

FMod_IPDCMOT_Read_COMMAND

```
bool FMod_IPDCMOT_Read_COMMAND(bool Continuous,
                                void *ComID);
```

Description

The *FMod_IPDCMOT_Read_COMMAND* function prepares the DLL to read the *COMMAND* (0x32) register next time *FMod_IPDCMOT_SendData_MAINPORT* is called.

Parameters

| | | |
|------------|------|--|
| Continuous | [in] | Boolean for repetitive ask of this register. |
| ComID | [in] | Pointer on the communication. |

Return Values

| | |
|-------|--|
| true | The function is successful. |
| false | The function failed. See log file or call <i>FMod_IPDCMOT_GetLastError</i> to get error details. |

FMod_IPDCMOT_Read_KP

```
bool FMod_IPDCMOT_Read_KP(void *ComID);
```

Description

The *FMod_IPDCMOT_Read_KP* function prepares the DLL to read the *KP* (0x33) register next time *FMod_IPDCMOT_SendData_MAINPORT* is called.

Parameters

| | | |
|-------|------|-------------------------------|
| ComID | [in] | Pointer on the communication. |
|-------|------|-------------------------------|

Return Values

| | |
|-------|--|
| true | The function is successful. |
| false | The function failed. See log file or call <i>FMod_IPDCMOT_GetLastError</i> to get error details. |

FMod_IPDCMOT_Write_KP

```
bool FMod_IPDCMOT_Write_KP(float Kp, void *ComID);
```

Description

The *FMod_IPDCMOT_Write_KP* function prepares the DLL to write to the *KP* (0x33) register next time *FMod_IPDCMOT_SendData_MAINPORT* is called.

Parameters

| | | |
|--------------|------|------------------------------------|
| <i>Kp</i> | [in] | 32 bits floating-point data value. |
| <i>ComID</i> | [in] | Pointer on the communication. |

Return Values

| | |
|-------|--|
| true | The function is successful. |
| false | The function failed. See log file or call <i>FMod_IPDCMOT_GetLastError</i> to get error details. |

Notes

Active:

Used with every regulation refresh when PID is selected.

Limits:

| | |
|------|-----------------------------|
| Max | 0x7FFFFFFF = 32'767.9999847 |
| Min | 0x00000000 = 0.0 |
| Step | 0x00000001 = 0.000015 |

Default:

KP is between 0.5 and 50.

Active:

Used with every regulation refresh when PID is selected.

FMod_IPDCMOT_Read_KI

```
bool FMod_IPDCMOT_Read_KI(void *ComID);
```

Description

The *FMod_IPDCMOT_Read_KI* function prepares the DLL to read the *KI* (0x34) register next time *FMod_IPDCMOT_SendData_MAINPORT* is called.

Parameters

| | | |
|-------|------|-------------------------------|
| ComID | [in] | Pointer on the communication. |
|-------|------|-------------------------------|

Return Values

| | |
|-------|--|
| true | The function is successful. |
| false | The function failed. See log file or call <i>FMod_IPDCMOT_GetLastError</i> to get error details. |

FMod_IPDCMOT_Write_KI

```
bool FMod_IPDCMOT_Write_KI(float Ki, void *ComID);
```

Description

The *FMod_IPDCMOT_Write_KI* function prepares the DLL to write to the *KI* (0x34) register next time *FMod_IPDCMOT_SendData_MAINPORT* is called.

Parameters

| | | |
|-------|------|------------------------------------|
| Ki | [in] | 32 bits floating-point data value. |
| ComID | [in] | Pointer on the communication. |

Return Values

| | |
|-------|--|
| true | The function is successful. |
| false | The function failed. See log file or call <i>FMod_IPDCMOT_GetLastError</i> to get error details. |

Notes

Active:

Used with every regulation refresh when PID is selected.

Limits:

| | |
|------|-----------------------------|
| Max | 0x7FFFFFFF = 32'767.9999847 |
| Min | 0x00000000 = 0.0 |
| Step | 0x00000001 = 0.000015 |

Default:

KI is between 0.01 and 1.0.

Active:

Used with every regulation refresh when PID is selected.

FMod_IPDCMOT_Read_KD

```
bool FMod_IPDCMOT_Read_KD(void *ComID);
```

Description

The *FMod_IPDCMOT_Read_KD* function prepares the DLL to read the *KD* (0x35) register next time *FMod_IPDCMOT_SendData_MAINPORT* is called.

Parameters

| | | |
|-------|------|-------------------------------|
| ComID | [in] | Pointer on the communication. |
|-------|------|-------------------------------|

Return Values

| | |
|-------|--|
| true | The function is successful. |
| false | The function failed. See log file or call <i>FMod_IPDCMOT_GetLastError</i> to get error details. |

FMod_IPDCMOT_Write_KD

```
bool FMod_IPDCMOT_Write_KD(float Kd, void *ComID);
```

Description

The *FMod_IPDCMOT_Write_KD* function prepares the DLL to write to the *KD* (0x35) register next time *FMod_IPDCMOT_SendData_MAINPORT* is called.

Parameters

| | | |
|-------|------|------------------------------------|
| Kd | [in] | 32 bits floating-point data value. |
| ComID | [in] | Pointer on the communication. |

Return Values

| | |
|-------|--|
| true | The function is successful. |
| false | The function failed. See log file or call <i>FMod_IPDCMOT_GetLastError</i> to get error details. |

Notes

Active:

Used with every regulation refresh when PID is selected.

Limits:

| | |
|------|-----------------------------|
| Max | 0x7FFFFFFF = 32'767.9999847 |
| Min | 0x00000000 = 0.0 |
| Step | 0x00000001 = 0.000015 |

Default:

KD is not used.

Active:

Used with every regulation refresh when PID is selected.

FMod_IPDCMOT_Read_ANTIRESETWINDUP

```
bool FMod_IPDCMOT_Read_LOOPTIME(void *ComID);
```

Description

The *FMod_IPDCMOT_Read_ANTIRESETWINDUP* function prepares the DLL to read the *ANTIRESETWINDUP* (0x36) register next time *FMod_IPDCMOT_SendData_MAINPORT* is called.

Parameters

| | | |
|-------|------|-------------------------------|
| ComID | [in] | Pointer on the communication. |
|-------|------|-------------------------------|

Return Values

| | |
|-------|--|
| true | The function is successful. |
| false | The function failed. See log file or call <i>FMod_IPDCMOT_GetLastError</i> to get error details. |

FMod_IPDCMOT_Write_ANTIRESETWINDUP

```
bool FMod_IPDCMOT_Write_ANTIRESETWINDUP(
    int AntiResetWindup, void *ComID);
```

Description

The *FMod_IPDCMOT_Write_ANTIRESETWINDUP* function prepares the DLL to write to the *ANTIRESETWINDUP* (0x36) register next time *FMod_IPDCMOT_SendData_MAINPORT* is called.

Parameters

| | | |
|-----------------|------|-------------------------------|
| AntiResetWindup | [in] | 32 bits integer value. |
| ComID | [in] | Pointer on the communication. |

Return Values

| | |
|-------|--|
| true | The function is successful. |
| false | The function failed. See log file or call <i>FMod_IPDCMOT_GetLastError</i> to get error details. |

Notes

Active:

Used for every PID calculation, when *KI* is not 0.

Limits:

Max 0x7FFFFFFF = 2'147'483'647

Min 0x00000000 = 0

Default:

0x7FFFFFFF. This value should not be modified.

FMod_IPDCMOT_Read_INTEGRALDELTA

```
bool FMod_IPDCMOT_Read_INTEGRALDELTA(bool Continuous,
                                       void *ComID);
```

Description

The *FMod_IPDCMOT_Read_INTEGRALDELTA* function prepares the DLL to read the *INTEGRALDELTA* (0x37) register next time *FMod_IPDCMOT_SendData_MAINPORT* is called.

Parameters

| | | |
|------------|------|--|
| Continuous | [in] | Boolean for repetitive ask of this register. |
| ComID | [in] | Pointer on the communication. |

Return Values

| | |
|-------|--|
| true | The function is successful. |
| false | The function failed. See log file or call <i>FMod_IPDCMOT_GetLastError</i> to get error details. |

FMod_IPDCMOT_Write_INTEGRALDELTA

```
bool FMod_IPDCMOT_Write_INTEGRALDELTA(int IntegralDelta,
                                         void *ComID);
```

Description

The *FMod_IPDCMOT_Write_INTEGRALDELTA* function prepares the DLL to write to the *INTEGRALDELTA* (0x37) register next time *FMod_IPDCMOT_SendData_MAINPORT* is called.

Parameters

| | | |
|---------------|------|-------------------------------|
| IntegralDelta | [in] | 32 bits integer value. |
| ComID | [in] | Pointer on the communication. |

Return Values

| | |
|-------|--|
| true | The function is successful. |
| false | The function failed. See log file or call <i>FMod_IPDCMOT_GetLastError</i> to get error details. |

Notes

Active:

Updated with every regulation refresh when PID is selected.

Limits:

| | |
|------|-----------------------------|
| Max | 0x7FFFFFFF = 2'147'483'647 |
| Zero | 0x00000000 = 0 |
| Min | 0x80000000 = -2'147'483'648 |

FMod_IPDCMOT_Read_DERIVATIONOFDELTA

```
bool FMod_IPDCMOT_Read_DERIVATIONOFDELTA(bool Continuous,
void *ComID);
```

Description

The *FMod_IPDCMOT_Read_DERIVATIONOFDELTA* function prepares the DLL to read the *DERIVATIONOFDELTA* (0x38) register next time *FMod_IPDCMOT_SendData_MAINPORT* is called.

Parameters

| | | |
|------------|------|--|
| Continuous | [in] | Boolean for repetitive ask of this register. |
| ComID | [in] | Pointer on the communication. |

Return Values

| | |
|-------|--|
| true | The function is successful. |
| false | The function failed. See log file or call <i>FMod_IPDCMOT_GetLastError</i> to get error details. |

FMod_IPDCMOT_Read_ACCELERATION

```
bool FMod_IPDCMOT_Read_ACCELERATION(void *ComID);
```

Description

The *FMod_IPDCMOT_Read_ACCELERATION* function prepares the DLL to read the *ACCELERATION* (0x40) register next time *FMod_IPDCMOT_SendData_MAINPORT* is called.

Parameters

| | |
|-------|------------------------------------|
| ComID | [in] Pointer on the communication. |
|-------|------------------------------------|

Return Values

| | |
|-------|--|
| true | The function is successful. |
| false | The function failed. See log file or call <i>FMod_IPDCMOT_GetLastError</i> to get error details. |

FMod_IPDCMOT_Write_ACCELERATION

```
bool FMod_IPDCMOT_Write_ACCELERATION(int Acceleration,
                                       void *ComID);
```

Description

The *FMod_IPDCMOT_Write_ACCELERATION* function prepares the DLL to write to the *ACCELERATION* (0x40) register next time *FMod_IPDCMOT_SendData_MAINPORT* is called.

Parameters

| | | |
|--------------|------|-------------------------------|
| Acceleration | [in] | 32 bits integer value. |
| ComID | [in] | Pointer on the communication. |

Return Values

| | |
|-------|--|
| true | The function is successful. |
| false | The function failed. See log file or call <i>FMod_IPDCMOT_GetLastError</i> to get error details. |

Notes

Active:

Used when *REGULATIONMODE* is in Position or Speed Control mode.

Limits:

| | |
|-----|----------------------------|
| Max | 0x7FFFFFFF = 2'147'483'647 |
| Min | 0x00000000 = 0 |

Default:

If you have no idea, set *ACCELERATION* = maximum speed of the motor. The motor will accelerate during approximately 1 second.

FMod_IPDCMOT_Read_DECELERATION

```
bool FMod_IPDCMOT_Read_DECELERATION(void *ComID);
```

Description

The *FMod_IPDCMOT_Read_DECELERATION* function prepares the DLL to read the *DECELERATION* (0x41) register next time *FMod_IPDCMOT_SendData_MAINPORT* is called.

Parameters

| | |
|-------|------------------------------------|
| ComID | [in] Pointer on the communication. |
|-------|------------------------------------|

Return Values

| | |
|-------|--|
| true | The function is successful. |
| false | The function failed. See log file or call <i>FMod_IPDCMOT_GetLastError</i> to get error details. |

FMod_IPDCMOT_Write_DECELERATION

```
bool FMod_IPDCMOT_Write_DECELERATION(int Deceleration,
                                       void *ComID);
```

Description

The *FMod_IPDCMOT_Write_DECELERATION* function prepares the DLL to write to the *DECELERATION* (0x41) register next time *FMod_IPDCMOT_SendData_MAINPORT* is called.

Parameters

| | | |
|--------------|------|-------------------------------|
| Deceleration | [in] | 32 bits integer value. |
| ComID | [in] | Pointer on the communication. |

Return Values

| | |
|-------|--|
| true | The function is successful. |
| false | The function failed. See log file or call <i>FMod_IPDCMOT_GetLastError</i> to get error details. |

Notes

Active:

Used when *REGULATIONMODE* is in Position Control mode.

Limits:

| | |
|-----|----------------------------|
| Max | 0x7FFFFFFF = 2'147'483'647 |
| Min | 0x00000000 = 0 |

Default:

If you have no idea, set *DECELERATION* = *TOPSPEED*. The motor will decelerate during approximately 1 second.

FMod_IPDCMOT_Read_TOPSPEED

```
bool FMod_IPDCMOT_Read_TOPSPEED(void *ComID);
```

Description

The *FMod_IPDCMOT_Read_TOPSPEED* function prepares the DLL to read the *TOPSPEED* (0x42) register next time *FMod_IPDCMOT_SendData_MAINPORT* is called.

Parameters

| | | |
|-------|------|-------------------------------|
| ComID | [in] | Pointer on the communication. |
|-------|------|-------------------------------|

Return Values

| | |
|-------|--|
| true | The function is successful. |
| false | The function failed. See log file or call <i>FMod_IPDCMOT_GetLastError</i> to get error details. |

FMod_IPDCMOT_Write_TOPSPEED

```
bool FMod_IPDCMOT_Write_TOPSPEED(int TopSpeed, void *ComID);
```

Description

The *FMod_IPDCMOT_Write_TOPSPEED* function prepares the DLL to write to the *TOPSPEED* (0x42) register next time *FMod_IPDCMOT_SendData_MAINPORT* is called.

Parameters

| | | |
|----------|------|-------------------------------|
| TopSpeed | [in] | 32 bits integer value. |
| ComID | [in] | Pointer on the communication. |

Return Values

| | |
|-------|--|
| true | The function is successful. |
| false | The function failed. See log file or call <i>FMod_IPDCMOT_GetLastError</i> to get error details. |

Notes

Active:

Used when *REGULATIONMODE* is in Position Control mode only.

Limits:

Max 0x7FFFFFFF = 2'147'483'647

Min 0x00000000 = 0

Default:

0x7FFFFFFF, no speed limitations.

FMod_IPDCMOT_Read_DEADZONE

```
bool FMod_IPDCMOT_Read_DEADZONE(void *ComID);
```

Description

The *FMod_IPDCMOT_Read_DEADZONE* function prepares the DLL to read the *DEADZONE* (0x43) register next time *FMod_IPDCMOT_SendData_MAINPORT* is called.

Parameters

| | | |
|-------|------|-------------------------------|
| ComID | [in] | Pointer on the communication. |
|-------|------|-------------------------------|

Return Values

| | |
|-------|--|
| true | The function is successful. |
| false | The function failed. See log file or call <i>FMod_IPDCMOT_GetLastError</i> to get error details. |

FMod_IPDCMOT_Write_DEADZONE

```
bool FMod_IPDCMOT_Write_DEADZONE(int DeadZone,
                                   void *ComID);
```

Description

The *FMod_IPDCMOT_Write_DEADZONE* function prepares the DLL to write to the *DEADZONE* (0x43) register next time *FMod_IPDCMOT_SendData_MAINPORT* is called.

Parameters

| | | |
|----------|------|-------------------------------|
| DeadZone | [in] | 32 bits integer value. |
| ComID | [in] | Pointer on the communication. |

Return Values

| | |
|-------|--|
| true | The function is successful. |
| false | The function failed. See log file or call <i>FMod_IPDCMOT_GetLastError</i> to get error details. |

Notes

Active:

Used when *REGULATIONMODE* is in Position or Speed Control mode only.

Limits:

| | |
|-----|----------------------------|
| Max | 0x7FFFFFFF = 2'147'483'647 |
| Min | 0x00000000 = 0 |

Default:

0x00000001 (1) when no overshoot of goal position is configured (*KP*, *KI*, *KD*, *DECELERATION*).

FMod_IPDCMOT_Read_HOMINGOPTIONS

```
bool FMod_IPDCMOT_Read_HOMINGOPTIONS(void *ComID);
```

Description

The *FMod_IPDCMOT_Read_HOMINGOPTIONS* function prepares the DLL to read the *HOMINGOPTIONS* (0x48) register next time *FMod_IPDCMOT_SendData_MAINPORT* is called.

Parameters

| | |
|-------|------------------------------------|
| ComID | [in] Pointer on the communication. |
|-------|------------------------------------|

Return Values

| | |
|-------|--|
| true | The function is successful. |
| false | The function failed. See log file or call <i>FMod_IPDCMOT_GetLastError</i> to get error details. |

FMod_IPDCMOT_Write_HOMINGOPTIONS

```
bool FMod_IPDCMOT_Write_HOMINGOPTIONS(
    FMod_SHOMINGOPTIONS HomingOptions, void *ComID);
```

Description

The *FMod_IPDCMOT_Write_HOMINGOPTIONS* function prepares the DLL to write to the *HOMINGOPTIONS* (0x48) register next time *FMod_IPDCMOT_SendData_MAINPORT* is called.

Parameters

| | | |
|---------------|------|--------------------------------|
| HomingOptions | [in] | FMod_SHOMINGOPTIONS Structure. |
| ComID | [in] | Pointer on the communication. |

Return Values

| | |
|-------|--|
| true | The function is successful. |
| false | The function failed. See log file or call <i>FMod_IPDCMOT_GetLastError</i> to get error details. |

Notes

See *FMod_SHOMINGOPTIONS* for structure details.

FMod_IPDCMOT_Write_HOMING

```
bool FMod_IPDCMOT_Write_HOMING(void *ComID);
```

Description

The *FMod_IPDCMOT_Write_HOMING* function prepares the DLL to write to the *HOMING* (0x49) register next time *FMod_IPDCMOT_SendData_MAINPORT* is called.

Parameters

| | | |
|-------|------|-------------------------------|
| ComID | [in] | Pointer on the communication. |
|-------|------|-------------------------------|

Return Values

| | |
|-------|--|
| true | The function is successful. |
| false | The function failed. See log file or call <i>FMod_IPDCMOT_GetLastError</i> to get error details. |

FMod_IPDCMOT_Write_STOPHOMING

```
bool FMod_IPDCMOT_Write_STOPHOMING(void *ComID);
```

Description

The *FMod_IPDCMOT_Write_STOPHOMING* function prepares the DLL to write to the *STOPHOMING* (0x4A) register next time *FMod_IPDCMOT_SendData_MAINPORT* is called.

Parameters

| | | |
|-------|------|-------------------------------|
| ComID | [in] | Pointer on the communication. |
|-------|------|-------------------------------|

Return Values

| | |
|-------|--|
| true | The function is successful. |
| false | The function failed. See log file or call <i>FMod_IPDCMOT_GetLastError</i> to get error details. |

FMod_IPDCMOT_Read_HOMINGPOSITION

bool FMod_IPDCMOT_Read_HOMINGPOSITION(void *ComID);

Description

The *FMod_IPDCMOT_Read_HOMINGPOSITION* function prepares the DLL to read the *HOMINGPOSITION* (0x4B) register next time *FMod_IPDCMOT_SendData_MAINPORT* is called.

Parameters

| | | |
|-------|------|-------------------------------|
| ComID | [in] | Pointer on the communication. |
|-------|------|-------------------------------|

Return Values

| | |
|-------|--|
| true | The function is successful. |
| false | The function failed. See log file or call <i>FMod_IPDCMOT_GetLastError</i> to get error details. |

FMod_IPDCMOT_Write_HOMINGPOSITION

```
bool FMod_IPDCMOT_Write_HOMINGPOSITION(int HomingPosition,
void *ComID);
```

Description

The *FMod_IPDCMOT_Write_HOMINGPOSITION* function prepares the DLL to write to the *HOMINGPOSITION* (0x4B) register next time *FMod_IPDCMOT_SendData_MAINPORT* is called.

Parameters

| | | |
|----------------|------|-------------------------------|
| HomingPosition | [in] | 32 bits integer value. |
| ComID | [in] | Pointer on the communication. |

Return Values

| | |
|-------|--|
| true | The function is successful. |
| false | The function failed. See log file or call <i>FMod_IPDCMOT_GetLastError</i> to get error details. |

Notes

Active:

After *HOMING* function, when *HOMINGOPTIONS* conditions are reached.

Limits:

| | |
|-----|-----------------------------|
| Max | 0x7FFFFFFF = 2'147'483'647 |
| Min | 0x80000000 = -2'147'483'648 |

Default:

0x00000000 = 0

FMod_IPDCMOT_Read_HOMINGINPUT

bool FMod_IPDCMOT_Read_HOMINGINPUT(void *ComID);

Description

The *FMod_IPDCMOT_Read_HOMINGINPUT* function prepares the DLL to read the *HOMINGINPUT* (0x4C) register next time *FMod_IPDCMOT_SendData_MAINPORT* is called.

Parameters

| | | |
|-------|------|-------------------------------|
| ComID | [in] | Pointer on the communication. |
|-------|------|-------------------------------|

Return Values

| | |
|-------|--|
| true | The function is successful. |
| false | The function failed. See log file or call <i>FMod_IPDCMOT_GetLastError</i> to get error details. |

FMod_IPDCMOT_Write_HOMINGINPUT

```
bool FMod_IPDCMOT_Write_HOMINGINPUT(int HomingInput,
                                     void *ComID);
```

Description

The *FMod_IPDCMOT_Write_HOMINGINPUT* function prepares the DLL to write to the *HOMINGINPUT* (0x4C) register next time *FMod_IPDCMOT_SendData_MAINPORT* is called.

Parameters

| | | |
|-------------|------|-------------------------------|
| HomingInput | [in] | 32 bits integer value. |
| ComID | [in] | Pointer on the communication. |

Return Values

| | |
|-------|--|
| true | The function is successful. |
| false | The function failed. See log file or call <i>FMod_IPDCMOT_GetLastError</i> to get error details. |

Notes

Active:

After *HOMING* function, when *HOMINGOPTIONS* conditions are reached.

Limits:

| | |
|-----|-----------------------------|
| Max | 0x7FFFFFFF = 2'147'483'647 |
| Min | 0x80000000 = -2'147'483'648 |

Default:

0x00000000 = 0

FMod_IPDCMOT_Read_LIMITISETUP

```
bool FMod_IPDCMOT_Read_LIMITISETUP(void *ComID);
```

Description

The *FMod_IPDCMOT_Read_LIMITISETUP* function prepares the DLL to read the *LIMITISETUP* (0x50) register next time *FMod_IPDCMOT_SendData_MAINPORT* is called.

Parameters

| | | |
|-------|------|-------------------------------|
| ComID | [in] | Pointer on the communication. |
|-------|------|-------------------------------|

Return Values

| | |
|-------|--|
| true | The function is successful. |
| false | The function failed. See log file or call <i>FMod_IPDCMOT_GetLastError</i> to get error details. |

FMod_IPDCMOT_Write_LIMIT1SETUP

```
bool FMod_IPDCMOT_Write_LIMIT1SETUP(int Limit1Setup,
                                     void *ComID);
```

Description

The *FMod_IPDCMOT_Write_LIMIT1SETUP* function prepares the DLL to write to the *LIMIT1SETUP* (0x50) register next time *FMod_IPDCMOT_SendData_MAINPORT* is called.

Parameters

| | | |
|-------------|------|---|
| Limit1Setup | [in] | Unsigned Int 32 bits, each bit independent. |
| ComID | [in] | Pointer on the communication. |

Return Values

| | |
|-------|--|
| true | The function is successful. |
| false | The function failed. See log file or call <i>FMod_IPDCMOT_GetLastError</i> to get error details. |

Notes

When set to 1:

| Bit number | Bit name | Bit description |
|------------------|------------------------|--|
| LIMIT1SETUP.0 | bLimit1 Enable | Activates the limit number1 detection |
| LIMIT1SETUP.1 | bLimit1 ActivHigh | When clear, it is active low |
| LIMIT1SETUP.2 | bLimit1 RegulationMode | Copy LIMIT1 REGULATIONMODE to REGULATIONMODE |
| LIMIT1SETUP.3 | bLimit1 Position | Copy LIMIT1 POSITION to POSITION (once) |
| LIMIT1SETUP.4 | bLimit1 Index | not used (0), future functionality |
| LIMIT1SETUP.5 | bLimit1 xInputL | Defines where LIMIT1xINPUT must be copied to |
| LIMIT1SETUP.6 | bLimit1 xInputH | with previous bit |
| LIMIT1SETUP.5-31 | none | not used (0) |

FMod_IPDCMOT_Read_LIMITIREGULATIONMODE

```
bool FMod_IPDCMOT_Read_LIMITIREGULATIONMODE(void  
*ComID);
```

Description

The *FMod_IPDCMOT_Read_LIMITIREGULATIONMODE* function prepares the DLL to read the *LIMITIREGULATIONMODE* (0x51) register next time *FMod_IPDCMOT_SendData_MAINPORT* is called.

Parameters

| | | |
|-------|------|-------------------------------|
| ComID | [in] | Pointer on the communication. |
|-------|------|-------------------------------|

Return Values

| | |
|-------|--|
| true | The function is successful. |
| false | The function failed. See log file or call <i>FMod_IPDCMOT_GetLastError</i> to get error details. |

FMod_IPDCMOT_Write_LIMITIREGULATIONMODE

```
bool FMod_IPDCMOT_Write_LIMITIREGULATIONMODE(
    int LimitIRegMode, void *ComID);
```

Description

The *FMod_IPDCMOT_Write_LIMITIREGULATIONMODE* function prepares the DLL to write to the *LIMITIREGULATIONMODE* (0x51) register next time *FMod_IPDCMOT_SendData_MAINPORT* is called.

Parameters

| | | |
|---------------|------|-------------------------------|
| LimitIRegMode | [in] | 32 bits integer value. |
| ComID | [in] | Pointer on the communication. |

Return Values

| | |
|-------|--|
| true | The function is successful. |
| false | The function failed. See log file or call <i>FMod_IPDCMOT_GetLastError</i> to get error details. |

Notes

Active:

When LimitI is reached.

FMod_IPDCMOT_Read_LIMITIPOSITION

```
bool FMod_IPDCMOT_Read_LIMITIPOSITION(void *ComID);
```

Description

The *FMod_IPDCMOT_Read_LIMITIPOSITION* function prepares the DLL to read the *LIMITIPOSITION* (0x52) register next time *FMod_IPDCMOT_SendData_MAINPORT* is called.

Parameters

| | | |
|-------|------|-------------------------------|
| ComID | [in] | Pointer on the communication. |
|-------|------|-------------------------------|

Return Values

| | |
|-------|--|
| true | The function is successful. |
| false | The function failed. See log file or call <i>FMod_IPDCMOT_GetLastError</i> to get error details. |

FMod_IPDCMOT_Write_LIMITIPOSITION

```
bool FMod_IPDCMOT_Write_LIMITIPOSITION(int LimitIPosition,
    void *ComID);
```

Description

The *FMod_IPDCMOT_Write_LIMITIPOSITION* function prepares the DLL to write to the *LIMITIPOSITION* (0x52) register next time *FMod_IPDCMOT_SendData_MAINPORT* is called.

Parameters

| | | |
|----------------|------|-------------------------------|
| LimitIPosition | [in] | 32 bits integer value. |
| ComID | [in] | Pointer on the communication. |

Return Values

| | |
|-------|--|
| true | The function is successful. |
| false | The function failed. See log file or call <i>FMod_IPDCMOT_GetLastError</i> to get error details. |

Notes

Active:

When LimitI is reached.

FMod_IPDCMOT_Read_LIMITIXINPUT

```
bool FMod_IPDCMOT_Read_LIMITIXINPUT(void *ComID);
```

Description

The *FMod_IPDCMOT_Read_LIMITIXINPUT* function prepares the DLL to read the *LIMITIXINPUT* (0x53) register next time *FMod_IPDCMOT_SendData_MAINPORT* is called.

Parameters

| | |
|-------|------------------------------------|
| ComID | [in] Pointer on the communication. |
|-------|------------------------------------|

Return Values

| | |
|-------|--|
| true | The function is successful. |
| false | The function failed. See log file or call <i>FMod_IPDCMOT_GetLastError</i> to get error details. |

FMod_IPDCMOT_Write_LIMITIXINPUT

```
bool FMod_IPDCMOT_Write_LIMITIXINPUT(int LimitIXInput,
                                       void *ComID);
```

Description

The *FMod_IPDCMOT_Write_LIMITIXINPUT* function prepares the DLL to write to the *LIMITIXINPUT* (0x53) register next time *FMod_IPDCMOT_SendData_MAINPORT* is called.

Parameters

| | | |
|--------------|------|-------------------------------|
| LimitIXInput | [in] | 32 bits integer value. |
| ComID | [in] | Pointer on the communication. |

Return Values

| | |
|-------|--|
| true | The function is successful. |
| false | The function failed. See log file or call <i>FMod_IPDCMOT_GetLastError</i> to get error details. |

Notes

Active:
When LimitI is reached.

FMod_IPDCMOT_Read_LIMIT2SETUP

bool FMod_IPDCMOT_Read_LIMIT2SETUP(void *ComID);

Description

The *FMod_IPDCMOT_Read_LIMIT2SETUP* function prepares the DLL to read the *LIMIT2SETUP* (0x58) register next time *FMod_IPDCMOT_SendData_MAINPORT* is called.

Parameters

ComID [in] Pointer on the communication.

Return Values

| | |
|-------|--|
| true | The function is successful. |
| false | The function failed. See log file or call <i>FMod_IPDCMOT_GetLastError</i> to get error details. |

FMod_IPDCMOT_Write_LIMIT2SETUP

```
bool FMod_IPDCMOT_Write_LIMIT2SETUP(int Limit2Setup,
                                     void *ComID);
```

Description

The *FMod_IPDCMOT_Write_LIMIT2SETUP* function prepares the DLL to write to the *LIMIT2SETUP* (0x58) register next time *FMod_IPDCMOT_SendData_MAINPORT* is called.

Parameters

| | | |
|-------------|------|---|
| Limit2Setup | [in] | Unsigned Int 32 bits, each bit independent. |
| ComID | [in] | Pointer on the communication. |

Return Values

| | |
|-------|--|
| true | The function is successful. |
| false | The function failed. See log file or call <i>FMod_IPDCMOT_GetLastError</i> to get error details. |

Notes

When set to 1:

| Bit number | Bit name | Bit description |
|------------------|------------------------|--|
| LIMIT2SETUP.0 | bLimitI Enable | Activates the limit numberI detection |
| LIMIT2SETUP.1 | bLimitI ActivHigh | When clear, it is active low |
| LIMIT2SETUP.2 | bLimitI RegulationMode | Copy LIMIT2REGULATIONMODE to REGULATIONMODE |
| LIMIT2SETUP.3 | bLimitI Position | Copy LIMIT2POSITION to POSITION (once) |
| LIMIT2SETUP.4 | bLimitI Index | not used (0), future functionality |
| LIMIT2SETUP.5 | bLimitI xInputL | Defines where LIMIT2xINPUT must be copied to |
| LIMIT2SETUP.6 | bLimitI xInputH | with previous bit |
| LIMIT2SETUP.5-31 | none | not used (0) |

FMod_IPDCMOT_Read_LIMIT2REGULATIONMODE

```
bool FMod_IPDCMOT_Read_LIMIT2REGULATIONMODE(void
*ComID);
```

Description

The *FMod_IPDCMOT_Read_LIMIT2REGULATIONMODE* function prepares the DLL to read the *LIMIT2REGULATIONMODE* (0x59) register next time *FMod_IPDCMOT_SendData_MAINPORT* is called.

Parameters

| | | |
|-------|------|-------------------------------|
| ComID | [in] | Pointer on the communication. |
|-------|------|-------------------------------|

Return Values

| | |
|-------|--|
| true | The function is successful. |
| false | The function failed. See log file or call <i>FMod_IPDCMOT_GetLastError</i> to get error details. |

FMod_IPDCMOT_Write_LIMIT2REGULATIONMODE

```
bool FMod_IPDCMOT_Write_LIMIT2REGULATIONMODE(
    int Limit2RegMode, void *ComID);
```

Description

The *FMod_IPDCMOT_Write_LIMIT2REGULATIONMODE* function prepares the DLL to write to the *LIMIT2REGULATIONMODE* (0x59) register next time *FMod_IPDCMOT_SendData_MAINPORT* is called.

Parameters

| | | |
|---------------|------|-------------------------------|
| Limit2RegMode | [in] | 32 bits integer value. |
| ComID | [in] | Pointer on the communication. |

Return Values

| | |
|-------|--|
| true | The function is successful. |
| false | The function failed. See log file or call <i>FMod_IPDCMOT_GetLastError</i> to get error details. |

Notes

Active:

When Limit2 is reached.

FMod_IPDCMOT_Read_LIMIT2POSITION

```
bool FMod_IPDCMOT_Read_LIMIT2POSITION(void *ComID);
```

Description

The *FMod_IPDCMOT_Read_LIMIT2POSITION* function prepares the DLL to read the *LIMIT2POSITION* (0x5A) register next time *FMod_IPDCMOT_SendData_MAINPORT* is called.

Parameters

| | |
|-------|------------------------------------|
| ComID | [in] Pointer on the communication. |
|-------|------------------------------------|

Return Values

| | |
|-------|--|
| true | The function is successful. |
| false | The function failed. See log file or call <i>FMod_IPDCMOT_GetLastError</i> to get error details. |

FMod_IPDCMOT_Write_LIMIT2POSITION

```
bool FMod_IPDCMOT_Write_LIMIT2POSITION(int Limit2Position,
    void *ComID);
```

Description

The *FMod_IPDCMOT_Write_LIMIT2POSITION* function prepares the DLL to write to the *LIMIT2POSITION* (0x5A) register next time *FMod_IPDCMOT_SendData_MAINPORT* is called.

Parameters

| | | |
|----------------|------|-------------------------------|
| Limit2Position | [in] | 32 bits integer value. |
| ComID | [in] | Pointer on the communication. |

Return Values

| | |
|-------|--|
| true | The function is successful. |
| false | The function failed. See log file or call <i>FMod_IPDCMOT_GetLastError</i> to get error details. |

Notes

Active:
When Limit2 is reached.

FMod_IPDCMOT_Read_LIMIT2XINPUT

```
bool FMod_IPDCMOT_Read_LIMIT2XINPUT(void *ComID);
```

Description

The *FMod_IPDCMOT_Read_LIMIT2XINPUT* function prepares the DLL to read the *LIMIT2XINPUT* (0x5B) register next time *FMod_IPDCMOT_SendData_MAINPORT* is called.

Parameters

| | | |
|-------|------|-------------------------------|
| ComID | [in] | Pointer on the communication. |
|-------|------|-------------------------------|

Return Values

| | |
|-------|--|
| true | The function is successful. |
| false | The function failed. See log file or call <i>FMod_IPDCMOT_GetLastError</i> to get error details. |

FMod_IPDCMOT_Write_LIMIT2XINPUT

```
bool FMod_IPDCMOT_Write_LIMIT2XINPUT(int Limit2XInput,
                                       void *ComID);
```

Description

The *FMod_IPDCMOT_Write_LIMIT2XINPUT* function prepares the DLL to write to the *LIMIT2XINPUT* (0x5B) register next time *FMod_IPDCMOT_SendData_MAINPORT* is called.

Parameters

| | | |
|--------------|------|-------------------------------|
| Limit2XInput | [in] | 32 bits integer value. |
| ComID | [in] | Pointer on the communication. |

Return Values

| | |
|-------|--|
| true | The function is successful. |
| false | The function failed. See log file or call <i>FMod_IPDCMOT_GetLastError</i> to get error details. |

Notes

Active:
When Limit2 is reached.

FMod_IPDCMOT_Read_AllRegister

```
bool FMod_IPDCMOT_Read_AllRegister(void *ComID);
```

Description

The *FMod_IPDCMOT_Read_AllRegister* function prepares the DLL to read all the registers of the module next time *FMod_IPDCMOT_SendData_MAINPORT* is called.

Parameters

| | | |
|-------|------|-------------------------------|
| ComID | [in] | Pointer on the communication. |
|-------|------|-------------------------------|

Return Values

| | |
|-------|--|
| true | The function is successful. |
| false | The function failed. See log file or call <i>FMod_IPDCMOT_GetLastError</i> to get error details. |

FMod_IPDCMOT_ResetAllRegisterRead

```
bool FMod_IPDCMOT_ResetAllRegisterRead(void *ComID);
```

Description

The *FMod_IPDCMOT_ResetAllRegisterRead* function resets the DLL of all registers set to be read.

Parameters

| | | |
|-------|------|-------------------------------|
| ComID | [in] | Pointer on the communication. |
|-------|------|-------------------------------|

Return Values

| | |
|-------|--|
| true | The function is successful. |
| false | The function failed. See log file or call <i>FMod_IPDCMOT_GetLastError</i> to get error details. |

FMod_IPDCMOT_SendData_MAINPORT

```
bool FMod_IPDCMOT_SendData_MAINPORT(void *ComID);
```

Description

The *FMod_IPDCMOT_SendData_MAINPORT* function sends a FiveCo packet to the device with all of the user's requests.

Parameters

| | | |
|-------|------|-------------------------------|
| ComID | [in] | Pointer on the communication. |
|-------|------|-------------------------------|

Return Values

| | |
|-------|--|
| true | The function is successful, data is sent. |
| false | The function failed. See log file or call <i>FMod_IPDCMOT_GetLastError</i> to get error details. |

Notes

Sent data (read values and/or write acknowledge) results in a calling up of the callback function *Data_Received_MAINPORT*.

6. Repetitive ask Function

This set of function allows you to read some registers repetitively within a specified delay.

For example, if you want to read the register VOLTAGE every 10[ms], you only have to follow the next steps:

1. Set the register INPUTS in continuous read mode:
Call FMod_IPDCMOT_Read_VOLTAGE with continuous variable = true,
2. Set the repetitive time to 10[ms]
Call FMod_IPDCMOT_RepetitiveAskSetTime with TimeMiliSec = 10.
3. Start the repetitive ask
Call FMod_IPDCMOT_RepetitiveAskStart

Each 10[ms], the callback function Data_Received_MAINPORT will then be called up with the updated value for the register VOLTAGE.

To stop the repetitive ask, just call FMod_IPDCMOT_RepetitiveAskStop.

To add another register to the list for repetitive asks, you have to call the FMod_IPDCMOT_Read_XXX function with continuous = true. You can perform this task even if the repetitive ask is already running.

FMod_IPDCMOT_RepetitiveAskStart

```
bool FMod_IPDCMOT_RepetitiveAskStart(void *ComID);
```

Description

The *FMod_IPDCMOT_RepetitiveAskStart* start the automatic repetitive ask of specified registers.

Parameters

| | | |
|-------|------|-------------------------------|
| ComID | [in] | Pointer on the communication. |
|-------|------|-------------------------------|

Return Values

| | |
|-------|--|
| true | The function is successful. |
| false | The function failed. See log file or call <i>FMod_IPDCMOT_GetLastError</i> to get error details. |

Notes

To specify which register (XXX) will be read, use *FMod_IPDCMOT_Read_XXX* function with `continuous = true`.
 You can also erase the list of registers for repetitive asks with the function *FMod_IPDCMOT_RepetitiveAskReset*.
 To set the time of the repetition, call up *FMod_IPDCMOT_RepetitiveAskSetTime*.
 The default time value between two automatic repetitive asks is 100[ms].

FMod_IPDCMOT_RepetitiveAskStop

bool FMod_IPDCMOT_RepetitiveAskStop(void *ComID);

Description

The *FMod_IPDCMOT_RepetitiveAskStop* stops the automatic repetitive asks of specified registers.

Parameters

| | | |
|-------|------|-------------------------------|
| ComID | [in] | Pointer on the communication. |
|-------|------|-------------------------------|

Return Values

| | |
|-------|---|
| true | The function is successful. |
| false | The function failed. See log file or call FMod_IPDCMOT_GetLastError to get error details. |

Notes (note?)

These function only stops the loop of repetitive asks. The list of registers to read remains unchanged.

FMod_IPDCMOT_RepetitiveAskSetTime

```
bool FMod_IPDCMOT_RepetitiveAskSetTime(int TimeMiliSec,
                                         void *ComID);
```

Description

The *FMod_IPDCMOT_RepetitiveAskSetTime* sets the delay between two automatic repetitive asks.

Parameters

| | | |
|-------------|------|-------------------------------|
| TimeMiliSec | [in] | Delay in milliseconds. |
| ComID | [in] | Pointer on the communication. |

Return Values

| | |
|-------|--|
| true | The function is successful. |
| false | The function failed. See log file or call <i>FMod_IPDCMOT_GetLastError</i> to get error details. |

Notes

The TimeMiliSec must be positive and greater than zero.
 The smallest value for this time is 1[ms]. The DLL can follow this repetitive time value if the CPU is powerful enough and if the network to the connected module is fast enough.

FMod_IPDCMOT_RepetitiveAskReset

```
bool FMod_IPDCMOT_RepetitiveAskReset(void *ComID);
```

Description

The *FMod_IPDCMOT_RepetitiveAskReset* erases the list of registers for the automatic repetitive ask and stops the repetitive ask.

Parameters

| | | |
|-------|------|-------------------------------|
| ComID | [in] | Pointer on the communication. |
|-------|------|-------------------------------|

Return Values

| | |
|-------|--|
| true | The function is successful. |
| false | The function failed. See log file or call <i>FMod_IPDCMOT_GetLastError</i> to get error details. |

7. Communication events functions

The *FMod_IPDCMOT_DLLInterface.dll* is based on event interaction. This way, one doesn't have to call a function to know if data has been received. You only have to wait for the call of one of the next callback functions to know about important events in the communication process, such as data being received or the main states of the communication (ie: closed, opened or error).

The following functions are given in parameters of the next functions:

- FMod_IPDCMOT_OpenConnection_MAINPORT

| Name of the function | Description |
|------------------------|---|
| Data_Received_MAINPORT | This function is called when data is received from the connected module on the Main Port. |
| Com_Event | This function is called to inform user of a major communication event. |

It is recommended to use a different Data_Received_MAINPORT and Com_Event function for each opened communication in order to facilitate the reception of information from the DLL.

Data_Received_MAINPORT

```
void Data_Received_MAINPORT(FMod_SRegisterListRead*RegList,  
void *ComID)
```

Description

The *Data_Received_MAINPORT* function is called by the DLL when data received from the connected module on the Main Port is ready to be read.

Parameters

| | | |
|---------|------|---|
| RegList | [in] | Pointer on the structure containing the module's information. |
| ComID | [in] | Pointer on the communication. |

Return Values

No return value.

Notes

The Pointer on the structure is at your disposal during this event. When this function returns, the DLL could change the data of this structure when new data is incoming. Therefore, you have to read and/or store data before returning this event function.

Until the user returns this function, the communication will be blocked even if a repetitive ask has been activated.

Com_Event

```
void ComEvent(int State, void *ComID);
```

Description

The *ComEvent* function is called by the DLL when a main event occurs during the communication process.

Parameters

| | | |
|-------|------|------------------------------------|
| State | [in] | Actual state of the communication. |
| ComID | [in] | Pointer on the communication. |

Return Values

No return value.

Notes

This event informs the user of the TCP communication's next events:

- The communication is opened *State = 1 (State_Opened)*
- The communication is closed *State = 0 (State_Closed)*
- An error occurred during communication *State = 2 (State_Error)*

The pointer *ComID* on the communication received by this event function is a copy of the one received *by the functions*:

- FMod_IPDCMOT_OpenConnection_MAINPORT

You can use this pointer to call up any FMod_IPDCMOT_DLLInterface DLL function. However, if the function modifies the pointer *ComID* (like FMod_IPDCMOT_CloseConnection), you will have to copy the result value in your own *ComID* pointer for this communication.

8. Applications example

MAIN Port : Open and close communication

Header File

```
#include " FMod_IPDCMOT_DLLInterface.h"

//-----
//  MAIN port communication parameters
//-----

void *ComID_MAINPORT;

// Callback functions
void Data_Received_MAINPORT(FMod_SRegisterListRead *RegList, void *ComID);
void ComEvent_MAINPORT(int State, void *ComID);
```

Source File

```
//-----
//  MAIN port communication callback : Data Receive
//-----
void Data_Received_MAINPORT (FMod_SRegisterListRead *RegList, void *ComID);
{
}

//-----
//  MAIN port communication callback : Communication Event
//-----
void ComEvent_MAINPORT (int State, void *ComID)
{
}

//-----
//  MAIN port open communication
//-----
void OpenConnection_MainPORT( )
{
    int add[4] = {169, 254, 5, 5};
    ComID_MAINPORT = NULL;

    FMod_IPDCMOT_OpenConnection_MAINPORT (add, Data_Received_MAINPORT,
                                           ComEvent_MAINPORT,
                                           &ComID_MAINPORT);
}

//-----
//  MAIN port close communication
//-----
void CloseConnection_MainPORT ( )
{
    FMod_IPDCMOT_CloseConnection(&ComID_MAINPORT);
}

//-----
```

MAIN Port communication: Read and Write register

```
//-----
// MAIN port communication callback : Data Receive
//-----
void Data_Received_MAINPORT (FMod_SRegisterListRead *RegList, void *ComID);
{
    int tcpTime;

    if(RegList->TCPTIMEOUT_Read)
    {
        tcpTime = RegList->TCPTIMEOUT;
    }

    if(RegList->TCPTIMEOUT_Written)
    {
        // acknowledge of TCPTIMEOUT register write
    }
}
//-----
// MAIN port communication : Read register TCPTIMEOUT
//-----
void Read_TCPTIMEOUT ( );
{
    // Prepare to read Register TCPTIMEOUT
    FMod_IPDCMOT_Read_TCPTIMEOUT(ComID_MAINPORT);

    // Makes and send the packet to the module
    FMod_IPDCMOT_SendData_MAINPORT(ComID_MAINPORT);
}
//-----
// MAIN port communication: Write register TCPTIMEOUT
//-----
void Write_TCPTIMEOUT( );
{
    // Prepare to write Register TCPTIMEOUT (value = 25sec)
    FMod_IPDCMOT_Write_TCPTIMEOUT(25, ComID_MAINPORT);

    // Makes and send the packet to the module
    FMod_IPDCMOT_SendData_MAINPORT(ComID_MAINPORT);
}
//-----
// MAIN port communication: Read all registers
//-----
void Read_ALLREGISTER( );
{
    // Prepare to read all Registers
    FMod_IPDCMOT_Read_AllRegister(ComID_MAINPORT);

    // Makes and send the packet to the module
    FMod_IPDCMOT_SendData_MAINPORT(ComID_MAINPORT);
}
```

MAIN Port communication: Repetitive register read

```

//-----
//  MAIN port communication callback : Data Receive
//-----
void Data_Received_MAINPORT (FMod_SRegisterListRead *RegList, void *ComID);
{
    float Volt;

    if(RegList->VOLTAGE_Read)
    {
        Volt = RegList->VOLTAGE;
    }
}
//-----
//  MAIN port communication : Read register VOLTAGE in continuous
//-----
void Start_Read_VOLTAGE_Rep( );
{
    // Prepare to read Register VOLTAGE in continuous
    FMod_IPDCMOT_Read_VOLTAGE(true, ComID_MAINPORT);

    // Set repetitive time to 10[ms]
    FMod_IPDCMOT_RepetitiveAskSetTime(10, ComID_MAINPORT);

    // Start the repetitive ask
    FMod_IPDCMOT_RepetitiveAskStart(ComID_MAINPORT);
}
//-----
//  MAIN port communication : Stop repetitive ask
//-----
void Stop_Read_VOLTAGE_Rep( );
{
    // Start the repetitive ask
    FMod_IPDCMOT_RepetitiveAskStop(ComID_MAINPORT);
}

```

Contact address :

FiveCo - Innovative Engineering
PSE-C
CH-1015 Lausanne
Switzerland
Tel: +41 21 693 86 71
Fax: +41 21 693 86 70

www.fiveco.ch
info@fiveco.ch



Updates, Support and further Information available on the following web page:

http://www.fiveco.ch/section_motion/support_motion_E.htm