
FMC Technologies

Important

All information and technical specifications in this documentation have been carefully checked and compiled by the author. However, we cannot completely exclude the possibility of errors. **Smith Meter GmbH** is always grateful to be informed of any errors.

TABLE OF CONTENTS

FIGURES

1. INTRODUCTION

1.1. GENERAL

This document contains a description of how to use the MPU Series B/ Ultra Series ModBus interfaces. Both the serial interface and the Ethernet interface are described.

1.2. References

	Doc. No	**Title**
1	USM-0000020565 (MNKS001)	MPU Series B User Manual: *Operators Manual – Configuring the ModBus Serial Ports*.
2		Andy Swales, Schneider Electric: *Open ModBus/TCP Specification*, Release 1.0 29 March 1999

1.3. Abbreviations

CP	Communications Processor
DSP	Digital Signal Processor
VOS	Velocity of Sound

1.4. Definitions

IP	Internet Protocol
MPU	MultiPath Ultrasonic
TCP	Transport Control Protocol

This page is intentionally left blank

2. PROTOCOL DETAILS

The MPU Series B/Ultra Series ModBus protocol will only operate in RTU mode.

All floating point values are represented as 32-bit real numbers in two consecutive 16-bit ModBus registers.

All integer values are represented as 32-bit integers in two consecutive 16-bit ModBus registers.

The MPU Series B/Ultra Series acts as a ModBus Slave.

2.1. ModBus On Serial Port

The MPU Series B/Ultra Series supports ModBus connections on one or both of its serial ports.

2.1.1. Serial Port 1

The settings for this port are shown in the table below.

Baud rate	9600
Parity	None
Data bits	8
Stop bits	1
ModBus node number	1

2.1.2. Serial Port 2

The following settings are fixed for port 2:

Parity	None
Data bits	8
Stop bits	1

The rest of the parameters are configurable on this port.

2.1.3. Hardware interface details

The actual hardware interface used can be one of the following:

1. RS-232 (Full Duplex)
2. 2-wire RS-485 (Half Duplex)
3. 4-wire RS-485, i.e. RS-422 (Full Duplex)

2.1.4. Timing considerations for RS-485

For the Half Duplex RS-485 (2-wire) interface, the following timing considerations must be taken into account by the software on the ModBus master:

There must be at least a 20 mS "silent interval" from receiving a ModBus reply from the MPU, before the next ModBus request message issued by the master to the MPU. The software on the ModBus master will typically need to follow this pattern of communication:

1. Send a request message A to the meter
2. Read the reply message from the meter
3. **Wait for at least 20 mS**
4. Send the next request message B to the meter

...etc...

This will ensure that the meter has sufficient amount of time to change the driver direction before the master sends the next request.

2.2. ModBus On TCP/IP

The MPU Series B/Ultra Series ModBus on TCP/IP implementation conforms to the definition in section 1.2 References [2].

The following requirements apply to the *ModBus Application Protocol* (MBAP) header used for TCP/IP:

Fields	Length	Client
Transaction Identifier	2 bytes	Must be 0
Protocol Identifier	2 bytes	Must be 0
Length	2 bytes	Minimum 6
Unit Identifier	1 byte	Must be 1

3. REGISTER MAPPINGS

This section lists the specific MPU Series B/Ultra Series database object numbers and ModBus addresses.

The following listings contain the most important objects available on the MODBUS protocol. For a complete listing of objects, please use the database report in the Winscreen program.

Object overview:

Object number	ModBus Address	Description	Format
0-199	0-398	Measured/Calculated values (Read only)	Float32
200-299	400-598	Alarm / Status / Accumulators / Version information (Read only)	Int32
400-699	800-1398	Parameters (Restricted). Use Winscreen	Float32
700-799	1400-1598	Modes (Restricted). Use Winscreen	Int32
1000-1100	2000-2198	External Input Registers (Write) (P / T / Z / Density / GC)	Float32

In case of a 4/3/1 path meter, the object numbers / Modbus addresses are the same, and objects with higher path numbers are not applicable.

3.1. Version information

The following registers contain the version information for the meter. All registers are read only and 32 bit integers.

Object number	ModBus Address	Description
250	500	UDSP Serial Number
251	502	UAFE Serial Number
252	504	UACF Serial Number
264	528	MID Protection Mode (0=Off, 1=MID)
265	530	Hardware Lock (0=NA, 1=Open, 2=Locked)
266	532	Database checksum of all W&M parameters
271	542	DB Reference checksum (stored)
272	544	Event counter
273	546	Software checksum
274	548	Software version

3.2. Results/Output Registers

The following registers are the results registers of the MPU Series B. They are all 32-bit float values. They are updated on the completion of every measurement cycle.

Object number	ModBus Address	Description	Unit metric	Unit US
0	0	Log count *1	-	-
1	2	Alarm Status *2	-	-
2	4	Flow Velocity	m/s	ft/s
3	6	Velocity of Sound	m/s	ft/s
4	8	Actual Volume Flowrate	m^3/h	ft^3/h *4 bbl/h *5
5	10	Accumulated Volume Forward	m^3	ft^3 *4 bbl *5
6	12	Accumulated Volume Reverse	m^3	ft^3 *4 bbl *5
7	14	Profile Flatness (all except single path meters)	%	%
8	16	Profile Symmetry (all except single path meters)	%	%
9	18	Swirl Flow (only 6 path meters) Accumulated Standard Volume, Forward (4 paths MPU and less)	% Sm^3	% Sft^3 *4
10	20	Cross Flow (only 6 path meters) Accumulated Standard Volume, Reverse (4 paths MPU and less)	% Sm^3	% Sft^3 *4
11	22	Increment Time Duration *3	sec	Sec
12	24	Used Line Pressure	barA	psiA
13	26	Used Line Temperature	°C	°F
14	28	Measured Flow Velocity path 1	m/s	ft/s
15	30	Measured Flow Velocity path 2	m/s	ft/s
16	32	Measured Flow Velocity path 3	m/s	ft/s
17	34	Measured Flow Velocity path 4	m/s	ft/s
18	36	Measured Flow Velocity path 5	m/s	ft/s
19	38	Measured Flow Velocity path 6	m/s	ft/s
20	40	Measured VOS path 1	m/s	ft/s
21	42	Measured VOS path 2	m/s	ft/s
22	44	Measured VOS path 3	m/s	ft/s
23	46	Measured VOS path 4	m/s	ft/s
24	48	Measured VOS path 5	m/s	ft/s
25	50	Measured VOS path 6	m/s	ft/s
26	52	Percentage Of Signals Used 1A	%	%
27	54	Percentage Of Signals Used 1B	%	%
28	56	Percentage Of Signals Used 2A	%	%

Object number	ModBus Address	Description	Unit metric	Unit US
29	58	Percentage Of Signals Used 2B	%	%
30	60	Percentage Of Signals Used 3A	%	%
31	62	Percentage Of Signals Used 3B	%	%
32	64	Percentage Of Signals Used 4A	%	%
33	66	Percentage Of Signals Used 4B	%	%
34	68	Percentage Of Signals Used 5A	%	%
35	70	Percentage Of Signals Used 5B	%	%
36	72	Percentage Of Signals Used 6A	%	%
37	74	Percentage Of Signals Used 6B	%	%
38	76	Gain 1A	-	-
39	78	Gain 1B	-	-
40	80	Gain 2A	-	-
41	82	Gain 2B	-	-
42	84	Gain 3A	-	-
43	86	Gain 3B	-	-
44	88	Gain 4A	-	-
45	90	Gain 4B	-	-
46	92	Gain 5A	-	-
47	94	Gain 5B	-	-
48	96	Gain 6A	-	-
49	98	Gain 6B	-	-
50	100	S/N Raw 1	dB	dB
51	102	S/N Raw 2	dB	dB
52	104	S/N Raw 3	dB	dB
53	106	S/N Raw 4	dB	dB
54	108	S/N Raw 5	dB	dB
55	110	S/N Raw 6	dB	dB
56	112	S/N Used 1	dB	dB
57	114	S/N Used 2	dB	dB
58	116	S/N Used 3	dB	dB
59	118	S/N Used 4	dB	dB
60	120	S/N Used 5	dB	dB
61	122	S/N Used 6	dB	dB
62	124	Turbulence level 1	%	%
63	126	Turbulence level 2	%	%
64	128	Turbulence level 3	%	%
65	130	Turbulence level 4	%	%
66	132	Turbulence level 5	%	%
67	134	Turbulence level 6	%	%

[*1] This is a counter that is incremented by one on the completion of every measurement cycle.

[*2] This is the alarm status word. A non-zero value here indicates that one or more alarms have been raised on the meter. See chapter 4.1 for details.

[*3] This is the time elapsed during the most recent measurement cycle

[*4] Gas meters

[*5] Liquid meters

3.3. High Resolution Accumulators

For transfer of volumetric data, the high resolution accumulators should be used. It has a very high and fixed resolution so that no volume is lost because of rounding. Do not use object 5 and 6 for transfer of volume. They are for information only. Rounding errors on the 32 bit floating point could be significant.

Use these high resolution accumulators to calculate volume increments by reading a certain intervals and subtract volume:
Volume increment = (current volume - previous volume).
All objects are 32 bit integer values and should be used in pairs.
To convert the volume using a 64 bit floating point, use this formula:
Volume = Integer part + Fractional part * 1.0E-9

Object number	ModBus Address	Description	Unit metric	Unit US
230	460	Accumulated Volume Forward Integer part	m^3	$ft^{3\,*1}$ bbl ^{*2}
231	462	Accumulated Volume Forward Fraction part (1.0E-9)		
232	464	Accumulated Volume Reverse Integer part	m^3	$ft^{3\,*1}$ bbl ^{*2}
233	466	Accumulated Volume Reverse Fraction part (1.0E-9)		
234	468	Acc. Error Volume Forward Integer part	m^3	$ft^{3\,*1}$ bbl ^{*2}
235	470	Acc. Error Volume Forward Fraction part (1.0E-9)		
236	472	Acc. Error Volume Reverse Integer part	m^3	$ft^{3\,*1}$ bbl ^{*2}
237	474	Acc. Error Volume Reverse Fraction part (1.0E-9)		
238	476	Acc. standard volume forward Integer part	m^3	$Sft^{3\,*1}$
239	478	Acc. standard volume forward Fraction part (1.0E-9)		
240	480	Acc. standard volume reverse Integer part	m^3	$Sft^{3\,*1}$
241	482	Acc. standard volume reverse Fraction part (1.0E-9)		

^{*1} Gas meters

^{*2} Liquid meters

3.4. Computerized Process Variables (Registers)

Object number	ModBus Address	Description	Unit metric	Unit US
102	204	Reynolds Number, Estimated	-	-
103	206	Viscosity, Estimated	cSt	cSt
104	208	Used Density at Line Conditions	kg/m^3	g/cm^3
106	212	VPC-X	-	-
107	214	VPC Correction	%	%

Note: This table is only valid for liquid Ultrasonic meters.

3.5. DECA/GERG Calculation results (MPU)

This section contains the results of the DECA/GERG calculations:

Object number	ModBus Address	Description	Unit
170	340	Calculated Velocity of Sound	m/s
171	342	Calculated average Mole Weight *1	kg/mol
172	344	Calculated DZ/DP *1	1/Pa
173	346	Calculated DZ/DT *1	1/K
174	348	Calculated Isobaric Heat Capacity (DECA) –OR- Superior heat value (GERG)	J/kmol (DECA) MJ/m^3 (GERG)
175	350	Calculated gas compressibility, line	-
176	352	Calculated Gas Density, line	kg/m^3
177	354	Calculated gas compressibility, reference *1	
178	356	K, compressibility ratio *1	
179	358	Inferior heat value	MJ/m^3
180	360	Mole Methane (C1) *1	% mol
181	362	Mole Nitrogen (N2) *1	% mol
182	364	Mole Carbon dioxide (CO2) *1	% mol
183	366	Mole Ethane (C2) *1	% mol
184	368	Mole Propane (C3) *1	% mol
185	370	Mole Water (H2O) *1	% mol
186	372	Mole Hydrogen sulphide (H2S) *1	% mol
187	374	Mole Hydrogen (H2) *1	% mol
188	376	Mole Carbon oxide (CO) *1	% mol
189	378	Mole Oxygen (O2) *1	% mol
190	380	Mole i-Butane (IC4) *1	% mol
191	382	Mole n-Butane (NC4) *1	% mol
192	384	Mole i-Pentane (IC5) *1	% mol
193	386	Mole n-Pentane (NC5) *1	% mol
194	388	Mole n-Hexane (NC6) *1	% mol
195	390	Mole n-Heptane (NC7) *1)	% mol
196	392	Mole n-Octane (NC8) *1	% mol

*1 DECA only (Not GERG)

NOTE: Only applicable for gas meters (MPU)
NOTE: All values are 32-bit float values.

3.6. Input Registers

The following registers should be updated continuously by the external device.
Temperature and pressure (object 1000 and 1001) is used for correction of inner diameter, path lengths and path angles. It is also used for selection of transducer calibration node.
Density and compressibility (Object 1002-1005) is used for calculation of mass flow and standard volume flow rates. (MPU only)
Writing into these registers might be omitted if these functions are not needed, or if temperature and pressure is obtained via analog.

CAUTION: The 1000 series objects such as these are the only registers that should be updated continuously; updating other parameters unnecessarily will fill up the change log and stress the non-volatile storage devices.

Object number	ModBus Address	Description	Unit Metric	Unit US
1000	2000	Line Pressure, external source	barA	psiA
1001	2002	Line Temperature, external source	°C	F
1002 *1	2004	Gas Density at Line Conditions, external source	Kg/m^3	lbs/ft^3
1003 *1	2006	Gas Density at Ref Conditions, external source	Kg/Sm^3	lbs/Sft^3
1004 *1	2008	Gas Compressibility at Line Cond., external source	-	
1005 *1	2010	Gas Compressibility at Ref Cond., external source	-	
1006 *1	2012	Mole Methane (C1) *2	% mol	% mol
1007 *1	2014	Mole Nitrogen (N2)	% mol	% mol
1008 *1	2016	Mole Carbon dioxide (CO2)	% mol	% mol
1009 *1	2018	Mole Ethane (C2) *2	% mol	% mol
1010 *1	2020	Mole Propane (C3) *2	% mol	% mol
1011 *1	2022	Mole Water (H2O) *2	% mol	% mol
1012 *1	2024	Mole Hydrogen sulphide (H2S) *2	% mol	% mol
1013 *1	2026	Mole Hydrogen (H2) *2	% mol	% mol
1014 *1	2028	Mole Carbon oxide (CO) *2	% mol	% mol
1015 *1	2030	Mole Oxygen (O2) *2	% mol	% mol
1016 *1	2032	Mole i-Butane (IC4) *2	% mol	% mol
1017 *1	2034	Mole n-Butane (NC4) *2	% mol	% mol
1018 *1	2036	Mole i-Pentane (IC5) *2	% mol	% mol

1019 *1	2038	Mole n-Pentane (NC5) *2	% mol	% mol
1020 *1	2040	Mole n-Hexane (NC6) *2	% mol	% mol
1021 *1	2042	Mole n-Heptane (NC7) *2	% mol	% mol
1022 *1	2044	Mole c-Octane (NC8) *2	% mol	% mol
1023 *1	2046	Mole n-Nonane (NC9) *2	% mol	% mol
1024 *1	2048	Mole n-Decane (NC10) *2	% mol	% mol
1025 *1	2050	Mole Helium (HE) *2	% mol	% mol
1026 *1	2052	Mole Argon (AR) *2	% mol	% mol
1027 *1	2054	Mole Hexane+ (C6+) *2	% mol	% mol
1028 *1	2056	Mole rest *2	% mol	% mol

*1 Gas meters only (MPU)
*2 DECA only (Not GERG)

NOTE: All values are 32-bit float values.

4. REGISTER USAGE

Figure **1** below describes the most common way to implement a master application that is synchronized with the meter.

In this example, the master application running on an external flow computer is responsible for writing the correct pressure and temperature to the meter.

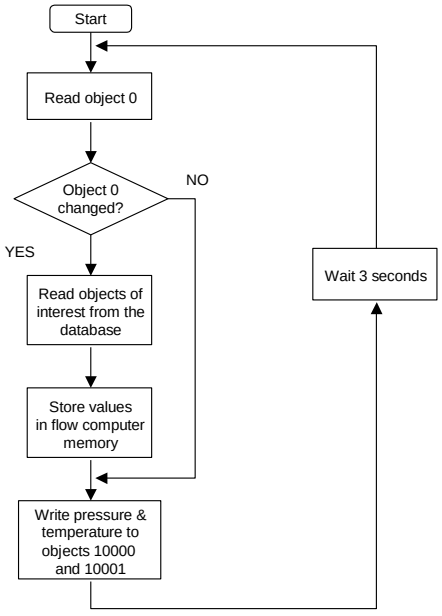

Figure 1 - Flow Computer Application Example

Alternatively, the master can read a whole block of data from object 0 to simplify and reduce the number of Modbus operations.

Whenever object 0 changes values, there are updated values in the block.

4.1. The MPU Series B Alarm Status

The MPU Series B Alarm Status is a bit coded value indicating the state of the MPU Series B alarms. To correctly interpret the bits, the 32-bit float value has to be converted into a 32-bit integer value. The table below shows the alarm bits used and their interpretation.

Bit	Value	Alarm interpretation
0	1	High Flow Alarm
1	2	Hardware Error
2	4	Transducer Failure
3	8	Calculation Failure
4	16	Burst Percent Low
5	32	Gain Error
6	64	Velocity-of-sound difference too large
7	128	Flow corrections active
8	256	Parameter error
9	512	S/N ratio low
10	1024	Turbulence level high
11	2048	Profile deviation high

4.2. The ModBus Protocol – Message Exchange Example

This chapter describes the exchange of messages taking place in a typical Flow Computer - MPU communication.

4.2.1. Modbus Read Message Example

In the following example the flow computer performs the following task:

- Read VOS and flow rate from the MPU Series B

The database objects used for VOS and flowrate object numbers 3 and 4. These have the ModBus addresses 6 (07D0 Hex) and 8. These registers are read with ONE message with function code 3. Figure 2 below shows the contents of this message.

Figure 2 - ModBus Read Message Example

Note: All bytes are shown in hexadecimal format.

The reply from the MPU Series B will be on the format described in Figure 3 below.

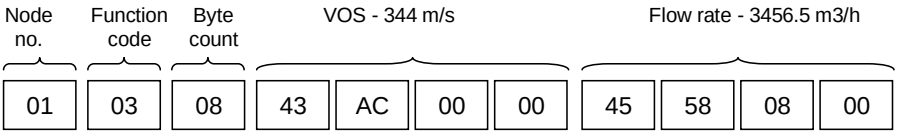

Figure 3 - ModBus Read Reply Message Example

The message contains the content of database objects 3 (VOS) and 4 (flow rate) represented as two 32-bit real values (least significant byte first).

4.2.2. ModBus Write Message Example

In this example the flow computer performs the following task:

- Write temperature and pressure from the MPU Series B

The database objects used for pressure and temperature from an external source are object numbers 1000 and 1001. These have the ModBus addresses 2000 (07D0 hex) and 2002. These registers are written with ONE message with function code 16. Figure 4 below shows the contents of this message.

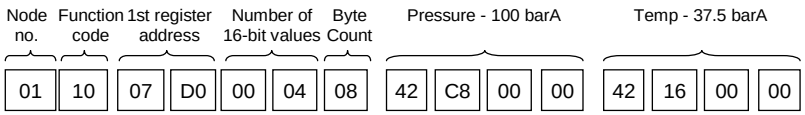

Figure 4 - ModBus Write Message Example

The reply from the MPU Series B will be on the format described in the figure below.

Figure 5 - ModBus Write Reply Message Example

The message contains a copy of the first 6 bytes of the request message.

5. WINDOWS COMMUNICATION SOFTWARE

5.1. MPUCOMM Dynamic Link Library

This communication link provides a fast and simple way of communicating with the MPU Series B using the network interface. The interface is based on a Microsoft Windows DLL. This DLL encapsulate all the communication with the MPU Series B, and offers a set of library functions that can be called from your program. (DLLs are callable from most programming languages including Microsoft visual C++, Microsoft Visual Basic, National Instrument Lab View/Lab Windows, Office applications that can be programmed with Visual Basic Macros i.e. Excel).

The PC to use this DLL has to have the following "pieces":

1. Running Microsoft Windows 98/NT//2000 or later versions.
2. Ethernet network (or a Ethernet card in your computer)
3. TCP/IP protocol installed under Windows.
4. Application can then call a Windows DLL (Dynamic Link Library). The DLL contains simple functions to read or write to the MPU Series B database via the network.

5.1.1. Installation

The IP address of the MPU Series B must be defined in the host file. For example:

```
128.1.221.121 MPUsn11
```

where '*128.1.221.121*' is the IP address, and '*MPUsn11*' is the name the machine can be referenced by.

This file is usually found in directory C:\WINNT\system32\drivers\etc\ on PCs running Windows NT, and under C:\WINDOWS\…. on PCs running Windows 98.

The following files are included in the MPU Series B network communication toolkit:

1. *MPUBComm.dll* - Callable library containing the implementation of the communication system.
2. *MPUBComm.lib* - Linkable library for Microsoft visual C++.
3. *DLLTester.exe* – An executable test program with a simple user interface
4. *SimpleTalk.cpp* – A very simple working example program in C showing the use of the DLL.

5. *MPUBCommExample.exe* - Executable of the above.
6. *W95ws2setup.exe* – Installation of windows socket version 2 for Windows 95.

Note: The library uses windows socket version 2. This component is standard in Windows NT 4.0, and Windows 98. Some versions of Windows 95 haven't got this component installed. If there are problems in Windows 95, run the "W95ws2setup.exe" program. This installs windows sockets version 2.

Warning: Do not run "W95ws2setup.exe" if you have Windows NT 4.0, Windows 98 or later.

5.1.2. Running The Test Programs

Both test programs must be run from the same directory the *MPUBComm.dll* file is located.

DLLTester.exe

This test program displays a simple dialog where the user can read and write values to the MPU Series B database. This is shown in Figure 6 below.

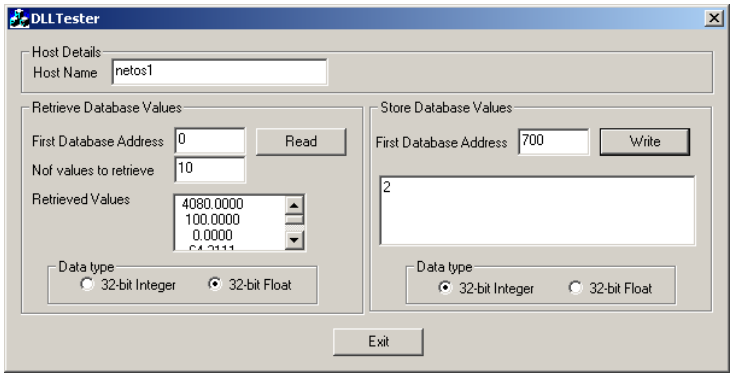

Figure 6 - DLLTester User Interface

SimpleTalk.exe

This is a simple console application that reads 5 values from the MPU Series B database. It always reads the same 5 objects starting from database object number 0.

Open a DOS window and enter the command:

`SimpleTalk` *`<machine name>`*

Figure 7 shows a sample run of this application.

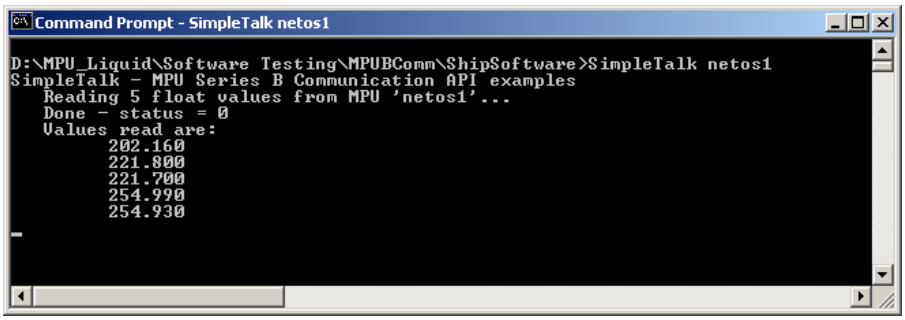

Figure 7 - Running the SimpleTalk.exe application

5.1.3. The Network Programmers Interface

The following functions are implemented in the DLL:

GetFloatValues

This functions reads object as float values from the MPU Series B database.

```
int DLLAPI getFloatValues(
  // Input
  const char     *lpszHost,           //
Hostname to connect to (MPUsn11)
  unsigned short nFirstObjectNumber,  //
First database object to read
  unsigned short nNOfObject,          // The
number of objects to read
  // Output
  float          *pfValueBuffer       //
Buffer to store the values
);
```

Parameters :

- **LpszHost**
  Specifies the host to connect. The host must be defined in the host file on your PC. The string must be null terminated.

- **NFirstObjectNumber**
  Specifies the object number of the first database value to read. These object numbers are defined in section 3.

- **NNOfObject**
  Specifies how many objects to read. This must be a number between 1 and 64. The parameter must be a 16 bit integer.

- **PfValueBuffer**
  Specifies the memory address where the function will place the read values. The parameter must be a pointer to an array of 32 bits single precision floating point values. The size of the array must at least be `nFirstObjectNumber` * 4 bytes.

GetIntValues

This functions reads object as float values from the MPU Series B database.

```
int DLLAPI getFloatValues(
  // Input
  const char     *lpszHost,           //
Hostname to connect to (MPUsn11)
  unsigned short nFirstObjectNumber,  //
First database object to read
  unsigned short nNOfObject,          // The
number of objects to read
  // Output
  int            *pnValueBuffer       //
Buffer to store the values
);
```

Parameters :

- **LpszHost**
  Specifies the host to connect. The host must be defined in the host file on your PC. The string must be null terminated.

- **NFirstObjectNumber**
  Specifies the object number of the first database value to read. These object numbers are defined in section 3.

- **NNOfObject**
  Specifies how many objects to read. This must be a number between 1 and 64. The parameter must be a 16 bit integer.

- **PnValueBuffer**
  Specifies the memory address where the function will place the read values. The parameter must be a pointer to an array of 64 bits integer values. The size of the array must at least be `nFirstObjectNumber` * 8 bytes.

SetFloatValues:

This function writes floating point values to the MPU Series B database.

```
int DLLAPI setFloatValues(
   // Input
   const char     *lpszHost,           //
Hostname to connect to (MPUsn11)
   unsigned short nFirstObjectNumber,  //
First database object to read
   unsigned short nNOfObject,          // The
number of objects to read
   float          *pfValueBuffer       //
Buffer with the values
 );
```

Parameters :

- **LpszHost**
  Specifies the host to connect. The host must be defined in the host file on your PC. The string must be null terminated.

- **NFirstObjectNumber**
  Specifies the object number of the first database value to write. These object numbers are defined in section 3.

- **NNOfObject**
  Specifies how many objects to write. This must be a number between 1 and 64. The parameter must be a 16 bit integer.

- **PfValueBuffer**
  Specifies the memory address where the values to write are fetched from. The parameter must be a pointer to an array of 32 bits single precision values. The size of the array must at least be `nFirstObjectNumber` * 4 bytes.

setIntValues

This function writes integer values to the MPU Series B database.

```
int DLLAPI setIntValues(
   // Input
   const char      *lpszHost,           //
Hostname to connect to (MPUsn11)
   unsigned short  nFirstObjectNumber,  //
First database object to read
   unsigned short  nNOfObject,          //
The number of objects to read
   int             *pnValueBuffer       //
Buffer with the values
 );
```

Parameters :

- **LpszHost**
  Specifies the host to connect. The host must be defined in the host file on your PC. The string must be null terminated.

- **NFirstObjectNumber**
  Specifies the object number of the first database value to write. These object numbers are defined in section 3.

- **NNOfObject**
  Specifies how many objects to write. This must be a number between 1 and 64. The parameter must be a 16 bit integer.

- **PnValueBuffer**
  Specifies the memory address where the values to write are fetched from. The parameter must be a pointer to an array of 64 bits integer values. The size of the array must at least be `nFirstObjectNumber` * 8 bytes.

5.1.4. Status Return Codes

The functions implemented in DLL return a single status code upon return. The code is returned as a 64-bit integer.

The table below shows the valid return values.

Return Code	Description
0	The function call succeeded
1	The function call failed
2	The connection attempt to the MPU Series B failed
3	The specified database address was invalid

5.2. MPU Series B WinScreen

The MPU Series B WinScreen is a stand-alone application for communicating with the MPU Series B. It can be used with both the TCP/IP and the serial ModBus interface. Refer to (MNKS001) PRD-0000020565 “User Manual, MPU Series B” for more details.