ALCIOM
3, rue des Vignes
78220 VIROFLAY - FRANCE

Tel/Fax : 01 47 09 30 51
✉ contact@alciom.com

www.alciom.com

Project :

SmartBrick

DOCUMENT :

SB-APP
Generic Input/Output Command Set specification (class 0x20)

REFERENCE :

AL/RL/1127/003

DATE :	VERSION :	AUTHOR :
02/10/2013	**1E**	**Robert Lacoste / ALCIOM**

SUMMARY:

This document is the specification of the SmartBrick "SB-APP Generic Input/Output Command Set" applicative class. This class is supported by all modules which measure discrete values of physical entities, like ADC's, wattmeters, inertial sensors, etc. and/or outputs discrete values, like DAC's, CW generators, switches, etc.

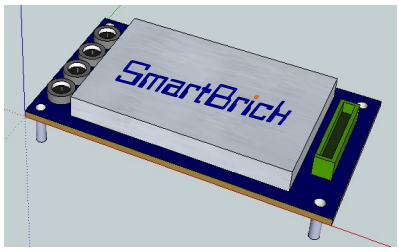
Nota : A separate class is dedicated to waveform digitizers or synthesizers. Class 20 manages discrete measurements and not time-sampled data.

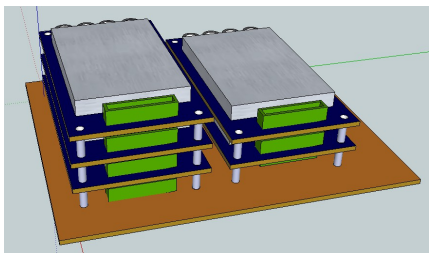
DOCUMENT HISTORY			
DATE	VERSION	AUTHOR	COMMENT
05/07/11	1A	R.Lacoste / ALCIOM	Initial version
17/08/11	1B	P.Rousseau / ALCIOM	Minor corrections
29/08/11	1C	P.Rousseau / ALCIOM	Minor corrections
07/09/11	1D	R.Lacoste / ALCIOM	Addition of discrete outputs
02/10/13	1E	L.Thiriet / ALCIOM	Minor corrections

CONTENTS

1 Introduction

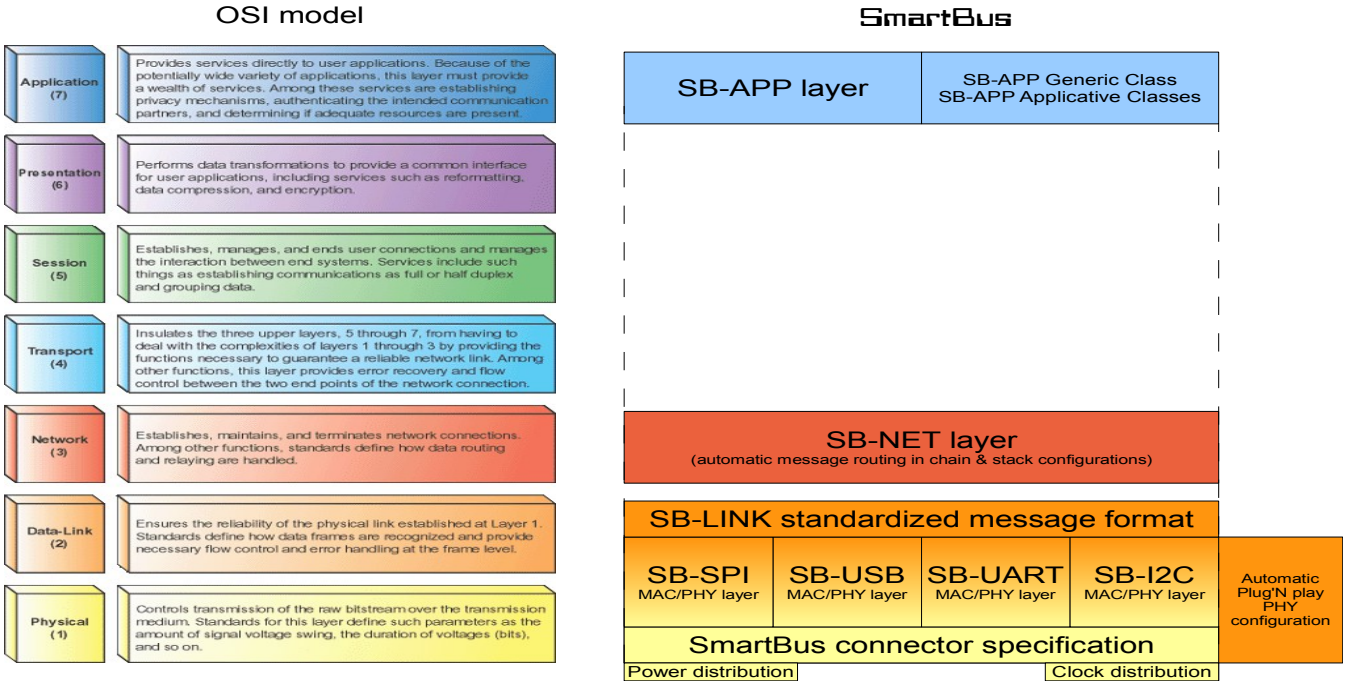
The SmartBrick products from ALCIOM are high performance analog and mixed-signal OEM modules. These modules are both daisy-chainable and stackable to build plug and play compact and efficient smart instrumentation and acquisition systems.





The SmartBrick modules are driven by a host system (microcontroller, DSP or PC, either embedded or remote) through the patented ultra-flexible SmartBus interface, compatible with SPI, USB, UART or I2C links. Open source software libraries provides an easy interface with any user-developed application in virtually any language : Labview, C, C++, Python, C#, Visual Basic, MATLAB/Scilab, etc.

In order to be as portable and flexible as possible, the SmartBus protocol is structured as several independent protocol layers, following the OSI 7-layers protocol model as follows (cf SmartBus Specification, AL/RL/1048/004) :



In particular the SB-APP layer specifies the set(s) of messages that are supported by a SmartBrick module. These messages are grouped in **Classes**. All modules must support at least the message set defined in the SB-APP Generic Class, but supports also one or several custom SB-APP Applicative Classes.

This document is the specification of the SmartBrick "SB-APP Generic Input/Output Command Set" applicative class, or class 0x20. This class is supported by all modules which measure discrete values of physical entities, like ADC's, wattmeters, inertial sensors, etc., and/or outputs discrete values, like DAC's, CW generators, switches, etc. .

Nota : a separate class is dedicated to waveform digitizers or synthesizers. Class 20 manages discrete measurements and not time-sampled data.

2 Class presentation

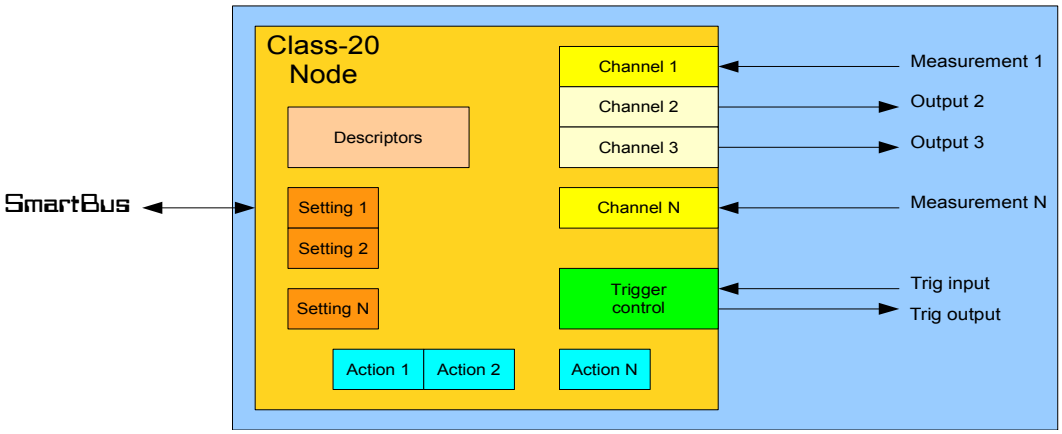
2.1 Class objectives

The SB-APP Generic Measurement Command Set (class 0x20) is designed to be a flexible way to manage virtually any measurement or generator module through a consistent command set. It allows either :

- Embedded applications to configure a module and get measurements in a very efficient and straightforward method with low protocol overloads ;
- PC-based applications to implement plug and play interfaces, with automatic discovery of the module's resources.

2.2 Module conceptual view

From the SB-APP Generic Measurement Command Set view point, the model of a class-20 compatible SmartBrick module is the following :



Each compatible module has the following elements :

- 0 or more **input channels** : An input channel is a measurement source. For example a voltmeter module will have one channel per voltage input, a 3D accelerometer will have 3 input channels.

- 0 or more **output channels** : A channel is a managed output value. For example a DAC module will have one output channel per voltage output, a quad SPDT switch will have 4 output channels, one per switch, a single output CW generator could have one output channel for the frequency and one output channel for the amplitude, etc.

- 0 or more **settings** : A setting is a configuration option for the module. Some settings allow the programmer to select a discrete option in a given list (for example AC/DC/GND coupling selection or input range for a given channel), other settings allow the programmer to select a value in a given pseudo-continuous range (for example a DC offset)

- A **trigger control** section, which allows to configure when the measurements should be made and how many successive measurements are needed. The trigger section also enables the synchronisation of several modules

- 0 or more **actions** : Actions are specific operations that could be executed on the selected module, independently from settings and measurements. The most usual action for a Measurement mode is to start a calibration procedure.

- A **descriptor table**, which allow the host processor to discover the characteristics of the module : number of channels, settings and actions, description of these resources, etc. The descriptor table is mainly used for plug and play PC-based applications.

2.3 Trigger modes

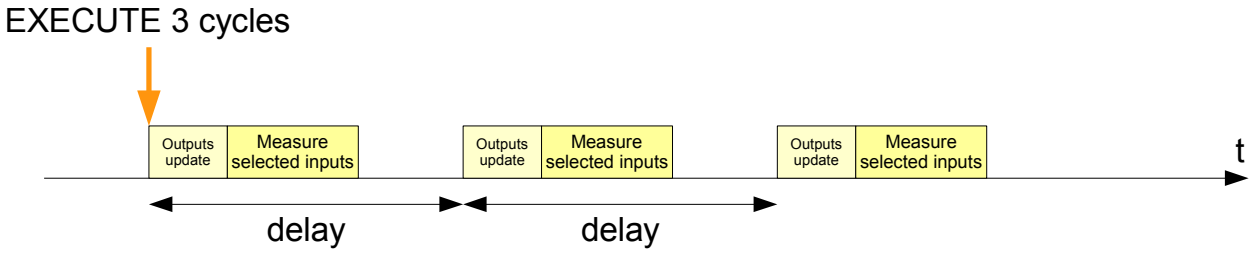
A module compliant with the Generic Input/Output Command Set class measures and/or generates physical values at discrete times.

When the user launches an acquisition and generation sequence, the module arms **t**he trigger section, **preparing for a set of measurements and/or signal generations. This could be** either for a single measurement on selected channels, a single update of output channels, or N successive output updates and synchronized measurements on these selected channels.

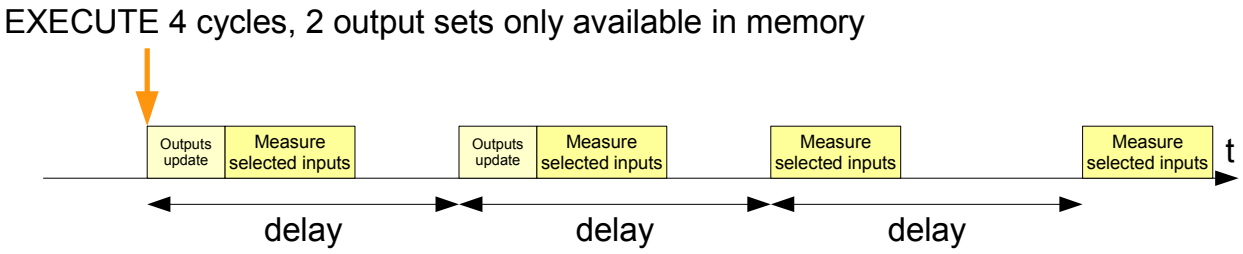
These events could be either executed immediately, or synchronised to other events through the hardware trigger line included in the SmartBus interface. The trigger subsystem supports the following modes :

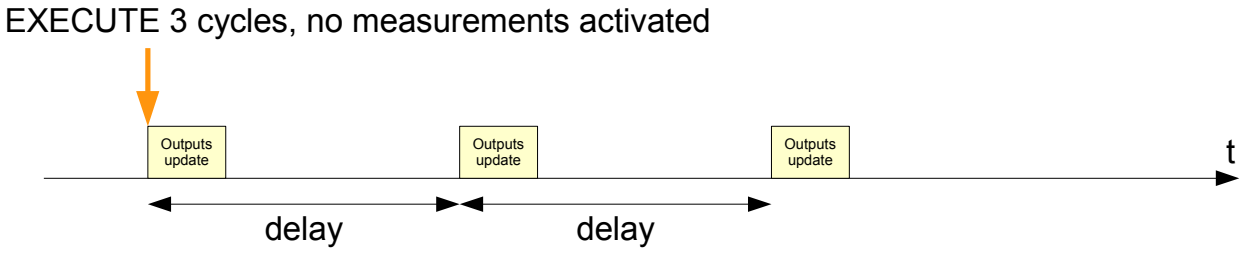
Autonomous mode :

In autonomous mode the measurements and output updates are started as soon as the Execute command is received. If successive measurements/updates have to be made then each set is separated by a configurable module-generated delay



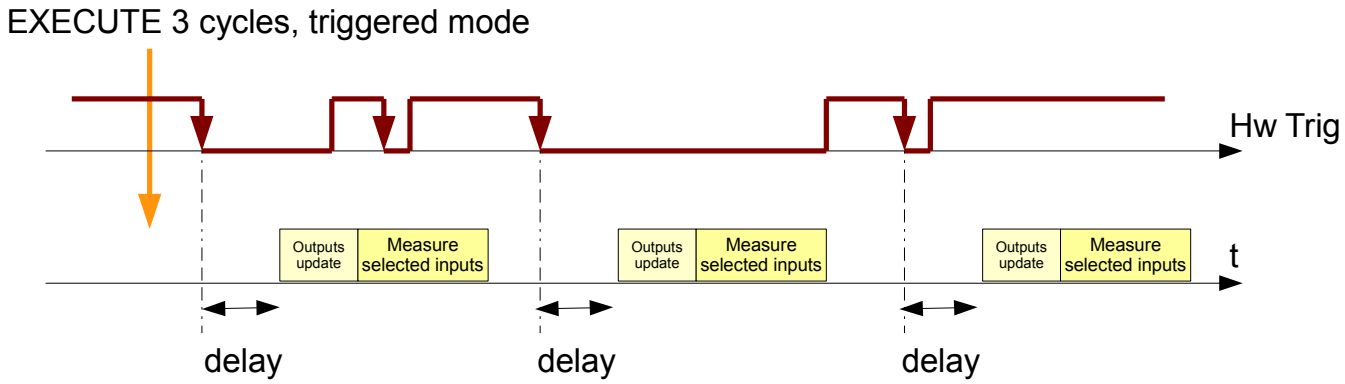
A cycle can also be started with less output updates records available in memory than the total number of requested measurement sets, or at the contrary output updates can be launched without asking for any measurements:





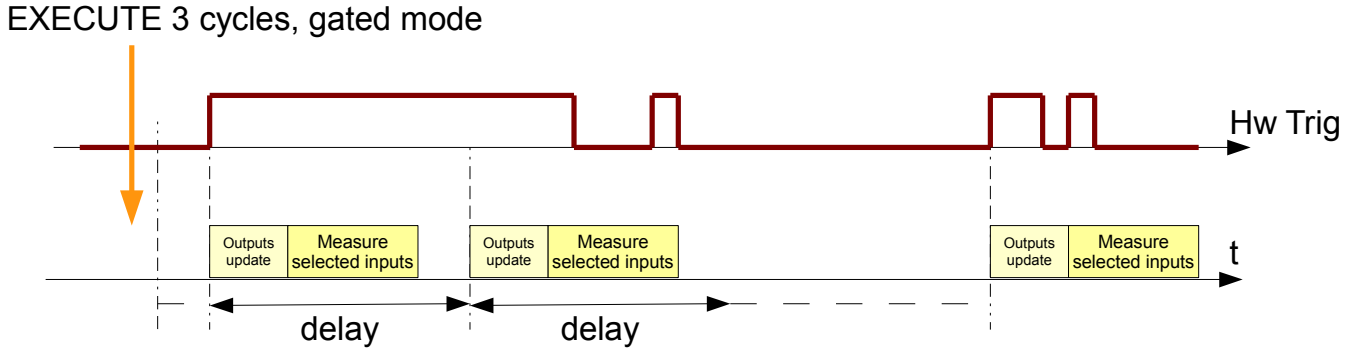
External trigger mode :

In external trigger mode, each measurement and/or output update is executed when the hardware trigger line has an active front (high to low as the line is OR'ed between modules). A programmed delay can be set between the active front and actual measurement/update :

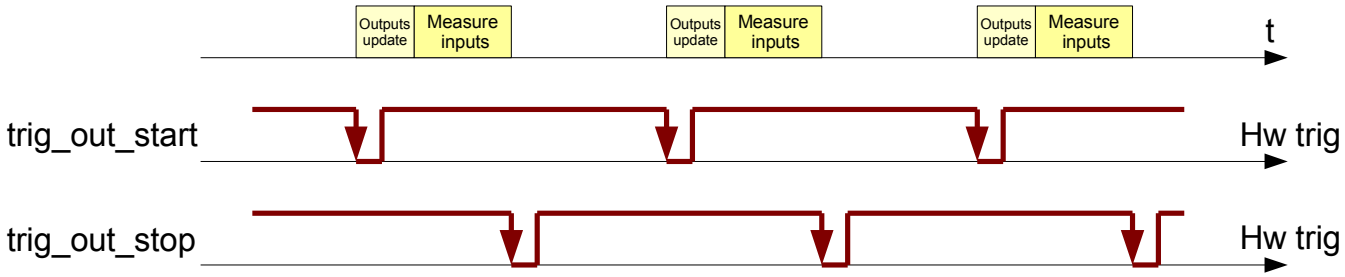


Gated mode :

In gated mode, as in autonomous mode, the measurements are started as soon as the arm command is received. If successive measurements have to be made then each measurement set is separated by a configurable module-generated delay. However, in addition, each measurement is differed until the hardware trigger line is in a preset logic state, either low or high :



In addition, each module compatible with the Generic Input/Output Command Set class can generate a pulse on the hardware trigger line either at the beginning or the end of each measurement/Update :



2.4 Command sequence example

The detailed command set is defined in chapter 3. For a simple data acquisition/generation application, the successive Class 20 commands that should be exchanged between the host and a compliant module are the following :

Configuration phase :

- **Read Descriptors** (only for plug and play application, to discover the module resources and implement the corresponding interface)
- **Write Settings** (to configure the required acquisition mode)
- **Read Units** (to know which units and scaling factor will be used for the reported measurements)
- **Set Trigger Mode** (to select when an individual measurement should be executed)
- **Select Active Channels** (to choose which channels to measure)
- **Write Output Records** **(to download to module's memory one or successive output values)**

Execution phase (can be repeated) :

- **Execute** (to launch a sequence of 1 to N measurements and/or output updates)

Data downloading phase :

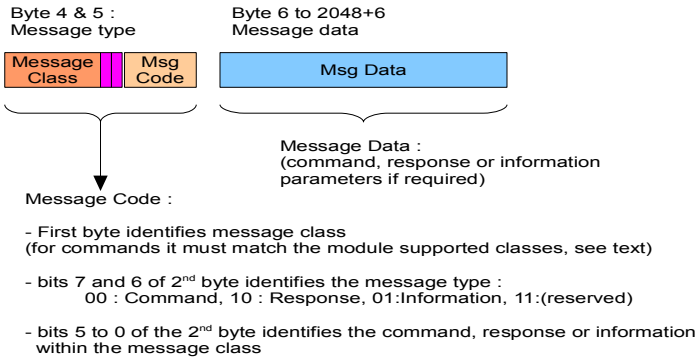
- **Read Measurements** (to recover measurements, until all measurements will be read)

2.5 Class identifier

The SB-APP Generic Input/Output Command Set Class is identified by the following value

Class	Class identifier
SB-APP Generic Input/Output Command Set Class	0x20

This value is used as a prefix for all messages on this class, and allows the destination module to easily check if it supports the corresponding class or not :



2.6 Status informations

The generic SmartBus class 00 protocol includes a Get-Status command. As part of the answer to this command, each module compatible with the SB-APP Generic Measurement Command Set Class includes the following data set in the "Additional bytes" field of the Get-Status answer :

Byte	Content
0x20	Flag indicating class 20 specific status data
0x09	Length, in byte, of the class 20 status record
MeasurementCount	2 bytes, MSB first, providing the number of measurements available in memory through the Read-Measurements command
ActiveChannels	2 bytes, MSB first, current active channels (cf Set-Active-Channels command)
OutputRecCount	2 bytes, MSB first, providing the number of output data record still in memory, waiting to be sent to outputs
AvailRecMemory	2 bytes, MSB first, providing the number of output data records that could be received and stored in memory
TriggerMode	1 byte, current trigger mode (cf Set-Trigger-Mode command)

2.7 Supported modules

The SB-APP Generic Measurement Command Set Class is supported to date by the following modules :

Class	Supported modules
SB-APP Generic Measurement Command Set Class	SB_ADC24 SB_POW6000 SB_PGA4000 SB_INE9 SB_DAC16 SB_CW3000 SB_DDS350 SB_PGA4000 SB_CNV6000 SB_LPF50 SB_BPF3900 SB_MUX4000 SB_DEL10 SB_GPSLOCK

3 Class messages

Messages through a SmartBrick system are of three different types :

- **Commands** : Are sent by a requesting node to a target node, and request the target node to do a specific action. Commands are **always unicast messages** (one sender, one receiver). The target node must **always send a Response** back to the requesting node. If the target node is not existing then a Response will in any case be sent back to the requester by another node, flagging the error. The SmartBus protocol allows commands sent by host (usual case) or even from one module to another module for specific applications ;
- **Responses** : Are sent back by a target node to the requesting node after a command. Responses are always unicast messages too ;
- **Indications** (optional) : Could be send spontaneously by any module. Indications are always **broadcasted to all host clients.** In order to keep the use of the system as easy as possible Indications are disabled by default. If a host system supports Indications then it can enable Indications on one or several modules through a specific command.

The following paragraphs provide a detailed specification of all Commands, Responses and Indications supported by SmartBrick modules compliant to the **"SB-APP Generic Input/Output Command Set Class"** model.

3.1 Descriptors

3.1.1 Read Descriptors Command

Description : Read the descriptors published by the module

Command Format :

Message class	Message code	Message data
0x20	0x01	

Response format (module can also send back an Error Response, cf 9.2):

Message class	Message code	Message data	
0x20	0x01	Error code	0x00 if execution ok, error code if not (cf 4)
		Number of Channels Number of Actions Number of Settings	(one byte, 01 to 10) (one byte, 00 to FF) (one byte, 00 to FF)
		Output Mask	(two bytes, formatted as a logical OR of the following constants : 0x0001 if channel 1 is an output 0x0002 if channel 2 is an output … 0x8000 if channel 16 is an output
		Channels Names	(a zero terminated ASCII string containing the user-readable name of each input channel, separated by a semi-column. Example : "EXT INPUT1;EXT INPUT2;TEMP\0")
		Actions Names	(a zero terminated ASCII string containing the user-readable name of each action, separated by a semi-column. Example :

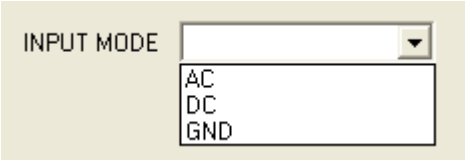
			“CALIBRATION;RESET OFFSET\0”)
		Setting 1 descriptor	(see here under)
		Setting 2 descriptor	(see here under)
		…	
		Setting N descriptor	(see here under)

For each setting the descriptor has one of the following forms :

Setting of a value in a discrete list (for example AC/DC/GND input selector) :

01	(one byte)
Number of options	(one byte)
Setting names	(a zero terminated ASCII string containing : - the user-readable name of the setting itself, - the user-readable name of each option for this setting, separated by a semi-column.

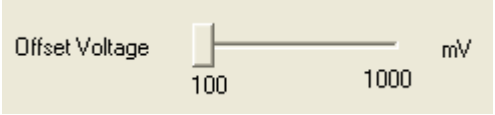
Example :
01
03
“INPUT MODE;DC;AC;GND\0”)



Setting of a value in a pseudo-continuous interval (for example DC offset selector) :

02	(one byte)
Min value	(two bytes, MSB first)
Max value	(two bytes, MSB first)
Setting names	(a zero terminated ASCII string containing : - the user-readable name of the setting itself, - the unit name of the setting; separated by a semi-column.

Example :
02
0064
03E8
“Offset Voltage;mV\0”)



3.2 Settings

3.2.1 Write Settings Command

Description : Modify the value of one or several settings

Command Format :

Message class	Message code	Message data	
0x20	0x08	Setting Number Setting Value Setting Number Setting Value etc (1 to 255 settings in a command)	0x01 to Number of Settings (two bytes, MSB first, begin at 0x0000) 0x01 to Number of Settings (two bytes, MSB first, begin at 0x0000)

Response format (module can also send back an Error Response, cf 9.2):

Message class	Message code	Message data	
0x20	0x08	Error code	0x00 if execution ok, error code if not (cf 4)

3.2.2 Read Settings Command

Description : Read back the value of one or several settings

Command Format :

Message class	Message code	Message data	
0x20	0x09	Setting Number Setting Number etc. (1 to 255 settings in a command)	0x01 to Number of Settings 0x01 to Number of Settings

Response format (module can also send back an Error Response, cf 9.2):

Message class	Message code	Message data	
0x20	0x09	Error code Setting Number Setting Value Setting Number Setting Value etc.	0x00 if execution ok, error code if not (cf 4) 0x01 to Number of Settings (two bytes, MSB first, begin at 0x0000) 0x01 to Number of Settings (two bytes, MSB first, begin at 0x0000)

3.3 Measurements and output values

3.3.1 Select Active Channels Command

Description : Select which channels should be used for the following **measurements** (nota : output channels can also be read back using this command)

Command Format :

Message class	Message code	Message data	
0x20	0x10	Channel Mask	(two bytes, formatted as a logical OR of the following constants : 0x0001 for channel 1 0x0002 for channel 2 … 0x8000 for channel 16

Response format (module can also send back an Error Response, cf 9.2):

Message class	Message code	Message data	
0x20	0x10	Error code	0x00 if execution ok, error code if not (cf 4)

3.3.2 Read Units Command

Description : Read the unit name, minimum/maximum values and scaling factor for each channel and associated with the current measurement settings. These informations don't change from measurement to measurement, so could be read only one time after any changes of the settings. One record is supplied for each requested channel.

Command Format :

Message class	Message code	Message data
0x20	0x11	

Response format (module can also send back an Error Response, cf 9.2):

Message class	Message code	Message data	
0x20	0x11	Error code	0x00 if execution ok, error code if not (cf 4)
		Channel count	(one byte, number of units provided in the message body)
		Min values	(four bytes per channel, MSB first, from first to last channel)
		Max values	(four bytes per channel, MSB first, from first to last channel)
		Numbers of decimals	(one byte per channel, for example if this value is 2 then a measurement of 0x03E8 (1000 decimal) should be read as “10.00”)

		Unit names	(a set of null-terminated strings giving the user-readable names of the unit for each channel, example : “mV\0mV\0V\0”

3.3.3 Write Output records Command

Description : Download one or several output records to the module. These output records will be sent to the physical outputs either immediately or later based on the trigger setting. If other records are already in memory then these new records will be added after the existing records (FIFO).

Command Format :

Message class	Message code	Message data	
0x20	0x14	Number of output records Number of set output channels Channel numbers Channel values output channel, MSB first :	(one byte) (one byte) (one byte per set output channel) (four bytes per output record and per set record 1, channel 1 record 1, channel 2 … record 2, channel 1 etc,

Response format (module can also send back an Error Response, cf 9.2):

Message class	Message code	Message data	
0x20	0x14	Error code	0x00 if execution ok, error code if not (cf 4)

3.3.4 Read Measurements Command

Description : Read back the measurements results. This command will return to the host all measurements that have been made since the last call to this command. This command could also read back the outputs records.

Command Format :

Message class	Message code	Message data	
0x20	0x18	MaxMeasCount	(one byte, maximum number of measurements to be read in one command)

Response format (module can also send back an Error Response, cf 9.2):

Message class	Message code	Message data	
0x20	0x18	Error code	0x00 if execution ok, error code if not (cf 4) In particular an error will be returned if no measurement is available or if memory is exhausted due to too long time between two calls to this command
		Number of measurements returned	(one byte)
		Number of measurements not read	(one byte)
		Number of channels per measurement	(one byte)

		Channel Mask (two bytes, cf Select Active Channels) Oldest measurement : Channel value (four bytes, MSB first) … Channel value (four bytes, MSB first) etc, Most recent measurement : Channel value (four bytes, MSB first) … Channel value (four bytes, MSB first)

3.4 Trigger

3.4.1 Set Trigger Mode Command

Description : Set when a measurement should be made

Command Format :

Message class	Message code	Message data	
0x20	0x20	Trigger Mode	(one byte, which could be either : 00 for autonomous mode 01 for triggered mode 02 for gated mode, active low 03 for gated mode, active high cf 2.3)
		Delay	(four bytes, MSB first, delay between successive measurements or between trigger event and measurement, expressed in units of 1µs from 0 to 1h11'34")
		Trigger out mode	(one byte, which could be either : 00 : don't output trigger pulses 01 : output pulses after each measurement 02 : output pulses before each measurement

Response format (module can also send back an Error Response, cf 9.2):

Message class	Message code	Message data	
0x20	0x20	Error code	0x00 if execution ok, error code if not (cf 4)

3.4.2 Execute Command

Description : Arm the trigger subsystem and launch a set of 1 to N measurements and/or output updates

Command Format :

Message class	Message code	Message data	
0x20	0x21	Cycle Count	(two bytes, MSB first, 0001 to FFFF. The specific value FFFF means indefinitely, the value 0 stops the current execution)

Response format (module can also send back an Error Response, cf 9.2):

Message class	Message code	Message data	
0x20	0x21	Error code	0x00 if execution ok, error code if not (cf 4)

3.5 Actions

3.5.1 Execute Action Command

Description : Executes immediately a specific action

Command Format :

Message class	Message code	Message data	
0x20	0x30	Action Number	(one byte, 01 to max action number)

Response format (module can also send back an Error Response, cf 9.2):

Message class	Message code	Message data	
0x20	0x30	Error code	0x00 if execution ok, error code if not (cf 4)

3.6 Unsolicited Indication messages

None

4 Class-specific error codes

The following error codes could be returned as part of SB-APP Low-Level Class 0 responses (error codes 0x30 to 0xFF are module specific, for generic error codes, i.e. 0x00 to 0x2F, please refer to the overall SmartBus specification)

Error code	Description	Comment
0x30	Unsupported setting number	Additional data : offending setting number
0x31	Unsupported setting value	Additional data : offending setting number and value
0x32	Illegal channel number	Additional data : channel number
0x40	No measurements available now	
0x41	Measurements lost	Indicates that the Read Measurement command was not executed soon enough, the memory was not enough to store all measurements.
0x44	Memory full	
0x50	Unsupported trigger mode	Additional data : offending trigger mode
0x51	Unsupported trigger output mode	Additional data : offending trigger output mode
0x60	Unsupported action number	Additional data : offending action number
0x70	Cannot execute command: cycles running	