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 Message Processing specification (class 0x30)

REFERENCE :

AL/LT/1340-003

DATE :	VERSION :	AUTHOR :
13/01/2014	**1C**	**Lucien Thiriet / ALCIOM**

SUMMARY:

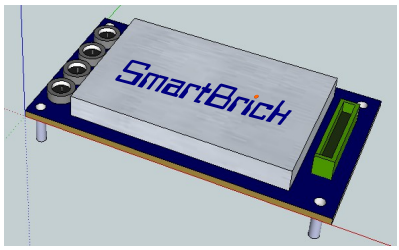
This document is the specification of the SmartBrick “SB-APP Generic Message Processing” applicative class. This class is supported by all modules which can receive and transmit messages, like wireless transceivers.

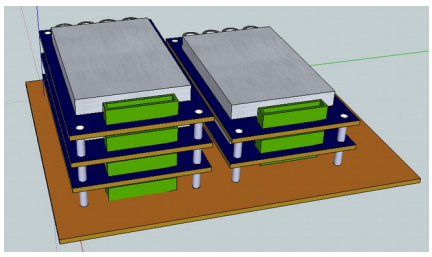
DOCUMENT HISTORY			
DATE	VERSION	AUTHOR	COMMENT
30/09/13	1A	L.Thiriet / ALCIOM	Initial version
11/10/13	1B	L.Thiriet / ALCIOM	Minor corrections
13/01/13	1C	R.Lacoste / ALCIOM	New absolute time triggering mode + notifications

CONTENTS

1 Introduction

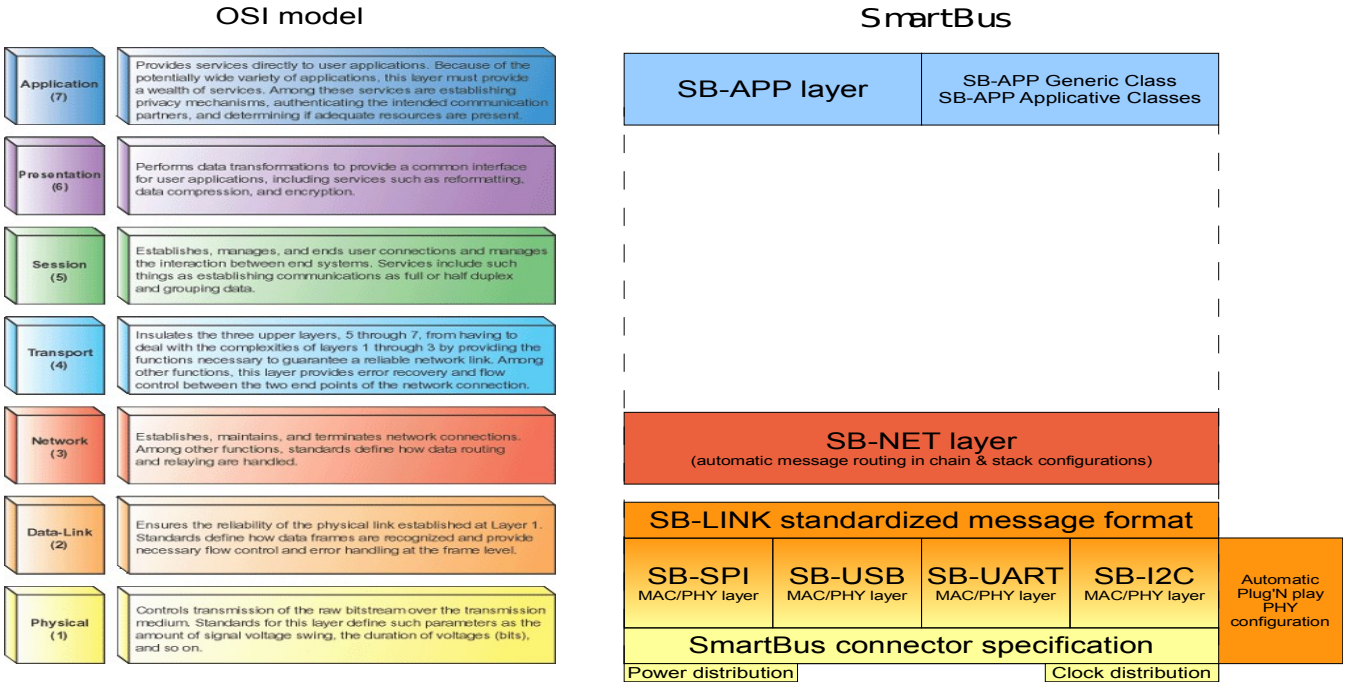
The SmartBrick products from ALCIOM are high performance analog and mixed-signal OEM modules. These modules are both daisy-chainable and stackable to build plug and play compact and efficient smart instrumentation and acquisition systems.





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 provide 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 Message Processing" applicative class, or class 0x30. This class is supported by all modules which can receive and transmit messages, like wireless transceivers.

2 Class presentation

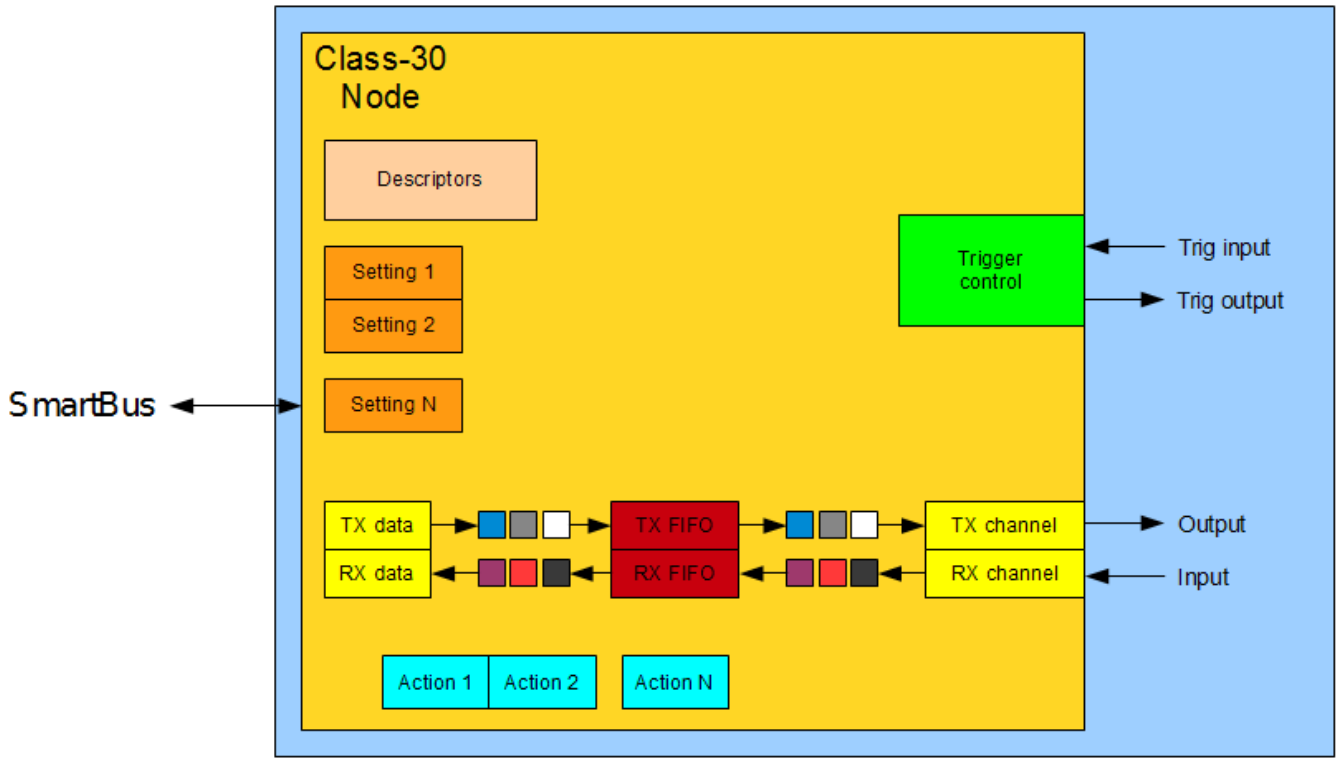
2.1 Class objectives

The SB-APP Generic Message Processing (class 0x30) is designed to be a flexible way to manage message sending and receiving on modules through a consistent command set. It allows either :

- Embedded applications to configure a module and send/receive messages 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 Message Processing viewpoint, the model of a class-30 compatible SmartBrick module is the following :



Each compatible module has the following elements :

- 1 **input channel** : an input channel is a message receiving channel. For example, a wireless receiver has an RX channel, which allows it to receive data from a wireless transmitter.
- 1 **output channel** : an output channel is a message transmitting channel. For example, a wireless transmitter has a TX channel, which allows it to send data to 1 or more wireless receivers.
- 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 modulation or CRC mode), other settings allow the programmer to select a value in a given pseudo-continuous range (for example frequency deviation or bitrate).
- A **trigger control** section, which allows to configure when the message should be sent. 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 (for example clearing the FIFOs).
- A **descriptor table**, which allows the host processor to discover the characteristics of the module : settings and actions, description of these resources, etc. The descriptor table is mainly used for plug and play PC-based applications.

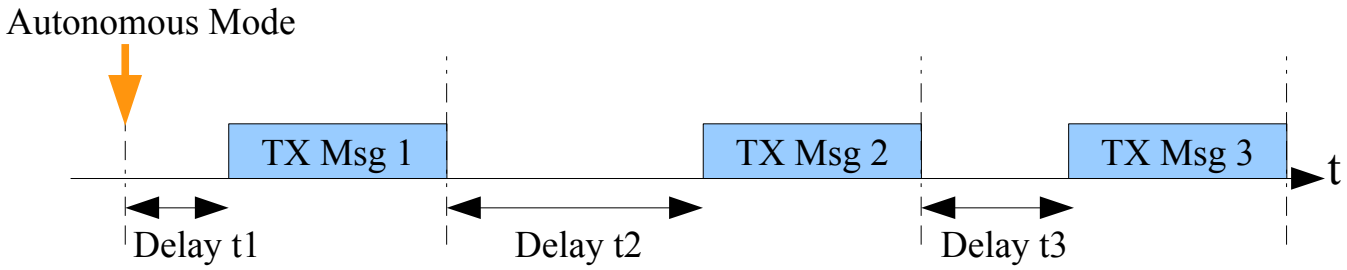
Trigger modes
A module compliant with the Generic Message Processing class sends and/or receives data messages at discrete times.

When the user activates the message processing through the Activate command, the module arms the trigger section, **preparing for a set of message receptions and emissions**. This could be either for a single message transmission, a single message reply, or several successive transmissions or replies.

The transmissions 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
- External trigger mode
- Reply mode
- Absolute time trigger mode

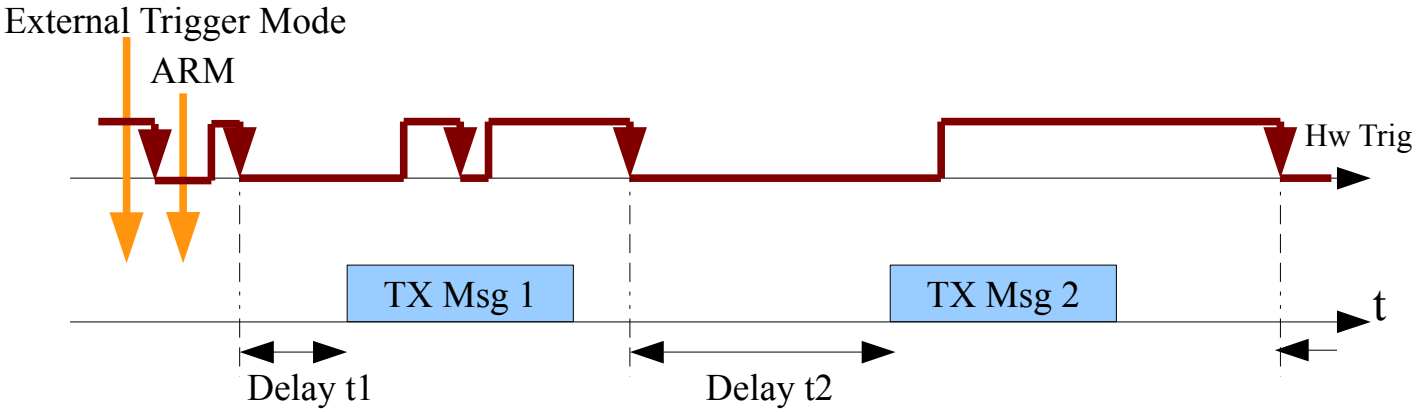
These modes are explained thereafter.



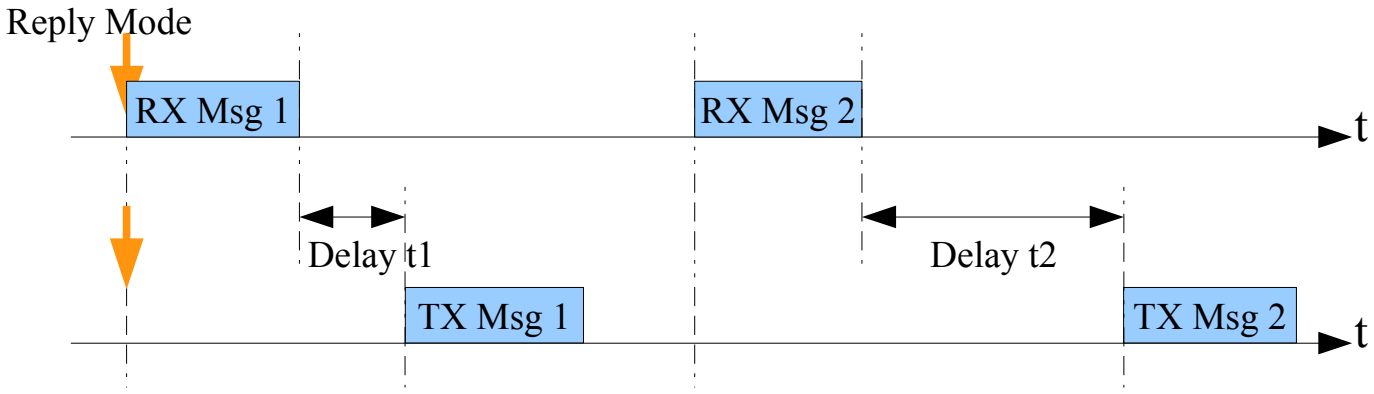
In autonomous mode, each message entered in the TX FIFO is associated with a given delay. The delay associated with the oldest message which has entered the TX FIFO is started as soon as the Activate command is executed. At the end of this delay, the oldest message is sent.

If the TX FIFO contains more than 1 message, the rest of the FIFO content is sent one message after the other, separated by the respetive delays.

If new messages are added to the TX FIFO when the message processing is active then the delay associated to the first message is immediately started.

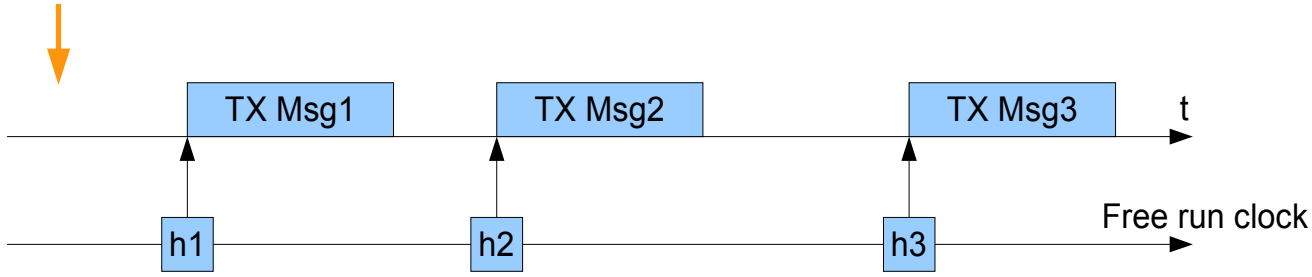


In external trigger mode, each message entered in the TX FIFO is associated with a given delay. The first message of the TX FIFO is sent when the hardware trigger line has an active front (high to low as the line is OR'ed between modules) and the programmed delay has ended.



In reply mode, each message entered in the TX FIFO is associated with a given delay. The device waits for a message received on the RX channel. Then, the oldest message in the TX FIFO is sent after its associated delay has finished.

Absolute time trigger mode



In Absolute time trigger mode each message is entered in the TX FIFO with an associated absolute time. The module updates a free running clock. The first message in the FIFO is sent when the value of the free running clock is **exactly equal** to the absolute time associated with this message.

The free running clock has a period of 1ms and is a 16-Bit value, so it loop back every 65,536s. Therefore the first message of the FIFO will be sent at most 65s after activation of the message processing. Similarly if two messages are entered in the FIFO with respective sending times T and T' and if T' is too close to T (ie if the module is busy sending the first message and time T' happens), then the second message will be sent 65.536s later.

2.3 Command sequence example

The detailed command set is defined in chapter 3. For a simple message transmission application, the successive Class 30 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 frequency, modulation type,...)
- **Write Message** (to load a message into the TX FIFO)

Execution phase :

- **Activate(TRUE)** (to launch a sequence of message transmissions and/or receptions)

Data downloading phase :

- **Read Message** (to read the oldest available message that have been received and stored in the RX FIFO)

End of execution

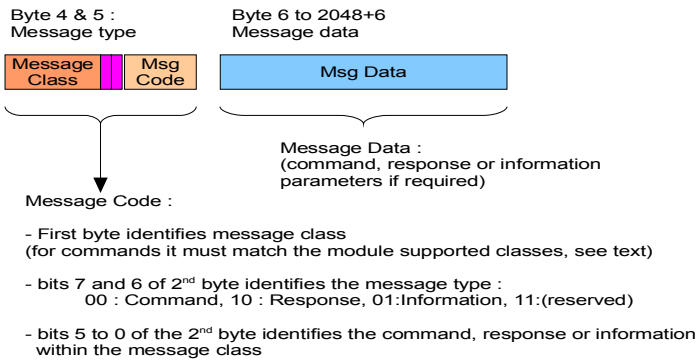
- **Activate(FALSE)**

2.4 Class identifier

The SB-APP Generic Message Processing class is identified by the following value :

Class	Class identifier
SB-APP Generic Message Processing class	0x30

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



2.5 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 Message Processing class includes the following data set in the “Additional bytes” field of the Get-Status answer :

Byte	Content
0x30	Flag indicating class 30 specific status data
0x05	Length, in byte, of the class 30 status record
RXMsgCount	1 byte, MSB first, providing the number of messages available in memory (RX FIFO) through the Read Message command
TXMsgCount	1 byte, MSB first, providing the number of output messages still in memory (TX FIFO), waiting to be sent
TXMsgAvailable	1 byte, providing the number of messages the TX FIFO can still host (free slots)
TriggerMode	1 byte, current trigger mode (cf Set-Trigger-Mode command)
ActiveMode	1 byte, current mode (message processing on or off)

2.6 Supported modules

The SB-APP Generic Message Processing class is supported to date by the following modules :

Class	Supported modules
SB-APP Generic Message Processing class	SB_CC1120

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 Message Processing 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
0x30	0x01	

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

Message class	Message code	Message data	
0x30	0x01	Error code	0x00 if execution ok, error code if not (cf 4)
		Number of Actions	(one byte, 00 to FF)
		Number of Settings	(one byte, 00 to FF)
		Actions Names	(a zero terminated ASCII string containing the user-readable name of each action, separated by a semi-column. Example : "Clear TX FIFO;Clear RX FIFO\0")
		Setting 1 descriptor	(see below)
		Setting 2 descriptor	(see below)
		…	
		Setting N descriptor	(see below)

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

Setting of a value in a discrete list (for example modulation type) :

01	(one byte, 01 : discrete list)
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
“MODULATION TYPE;FSK;GFSK;MSK\0”)

Setting of a value in a pseudo-continuous interval (for example RF emitting power) :

02	(one byte, 02 : pseudo-continuous interval)
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.

Example :
02
FFF8
0016
“Emitting Power;dBm\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	
0x30	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	
0x30	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	
0x30	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	
0x30	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 Messages

3.3.1 Write Message Command

Description : Download a message to the module's TX FIFO memory. This message will be output either immediately or later based on the trigger setting and the associated delay. If other messages are already in memory then this new message will be added after the existing ones.

Delay (two bytes, MSB first) : delay between trigger event and message transmission or between successive transmissions, expressed in units of 1ms, from 0 to 65.535s.

Command Format :

Message class	Message code	Message data	
0x30	0x14	Delay or Absolute time	(two bytes, MSB first, in ms)
		Message content	(one byte min.)

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

Message class	Message code	Message data	
0x30	0x14	Error code	0x00 if execution ok, error code if not (cf 4)

3.3.2 Read Message Command

Description : Read the oldest available message that has been received and stored in the RX FIFO. The slot containing the message will then be cleared to leave room for a new message.

Command Format :

Message class	Message code	Message data
0x30	0x18	

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

Message class	Message code	Message data	
0x30	0x18	Error code	0x00 if execution ok, error code if not (cf 4). In particular an error will be returned if : - No message is available - Memory is exhausted due to too long time between two calls to this command - A message received at that location of the RX FIFO has been lost due to an error (cf 4)
		TimeStamp	(two bytes, MSB first, in millisecond, corresponds to the value of the internal free running clock when the message was actually fully received (end of current reception)
		Message content	(one byte min., none if execution error)

3.4 Trigger

3.4.1 Set Trigger Mode Command

Description : Set when a message should be sent

Command Format :

Message class	Message code	Message data	
0x30	0x20	Trigger mode	(one byte, which could be either : 00 for autonomous mode (default) 01 for external trigger mode 02 for reply mode 03 for absolute time mode cf 2.3)
		Trigger out mode	(one byte, which could be either : 00 : don't output trigger pulses 01 : output pulses after each transmission 02 : output pulses before each transmission 03 : output pulses after each reception)

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

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

Set Activate Mode Command

Description : Activates or deactivates the message processing (TX and RX) as well as the trigger subsystem

Command Format :

Message class	Message code	Message data	
0x30	0x21	Activation Mode	(one byte, which could be either : 00 : deactivate message processing (default) 01 : activate Default 00)

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

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

3.6 Unsolicited Indication messages

The following unsollicited indications can be enabled with the Enable-Indication command (class 00, see SmartBus specification) :

Indication bit mask	Indication code	Indication data
0x01	0x30	Same as ReadMessage data content, see thereafter (disabled by default)

Nota : if indication bit mask 01 is enabled then the received messages are immediately sent through an unsollicited indication message and are not stored in the RX FIFO. Therefore the RX fifo stays empty and the ReadMessage command shouldn't be used.

4 Class-specific error codes

The following error codes could be returned as part of SB-APP Low-Level Class 0 responses (error codes 0x20 to 0xFF are module specific, for generic error codes, i.e. 0x00 to 0x2F, please refer to the overall SmartBus specification)

Error code	Description	In response to command	Comment
0x30	Unsupported setting number	Write Settings	Additional data : offending setting number
0x31	Unsupported setting value	Write Settings	Additional data : offending setting number and value
0x40	No message available now	Read message	RX FIFO is empty
0x41	RX message lost	Read message	Indicates that there was no free slot in RX FIFO for the arriving message OR the message received was too long. This message was ignored.
0x44	TX message rejected	Write Message	Indicates that there is no free slot in TX FIFO for the message asked to transmit OR the message is too long. This message is ignored.

0x50	Unsupported trigger mode	Set Trigger Mode	Additional data : offending trigger mode
0x51	Unsupported trigger output mode	Set Trigger Mode	Additional data : offending trigger output mode
0x60	Unsupported action number	Execute Action	Additional data : offending action number
0x70	Cannot execute command		Module is busy