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 Low-Level Class specification (class 0x10)

REFERENCE :

AL/RL/1103/007

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

SUMMARY:

This document is the specification of the SmartBrick "SB-APP Low Level Class" applicative class. This class of messages is a low-level command set, allowing a deep access to the hardware resources of the target module.

SB-APP Low Level Class is the applicative message class for the SB-PROTO prototyping module, but is also supported as well as by the majority of SmartBrick modules for specific low-level applications.

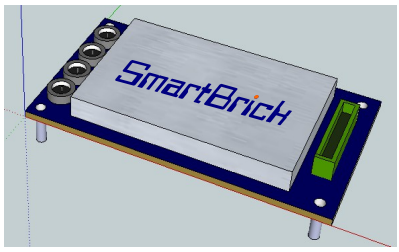
DOCUMENT HISTORY

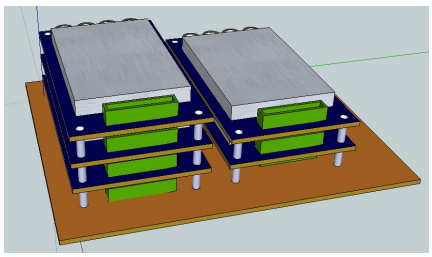
DATE	VERSION	AUTHOR	COMMENT
21/01/11	1A	R.Lacoste / ALCIOM	Initial version
31/01/11	1B	R.Lacoste / ALCIOM	Minor corrections
04/05/11	1C	T.Demarne / ALCIOM	Minor corrections
08/07/11	1D	R.Lacoste / ALCIOM	Change of command codes
11/07/11	1E	R.Lacoste / ALCIOM	Minor corrections
10/11/11	1F	R.Lacoste / ALCIOM	Suppression of unused commands
02/10/13	1G	L.Thiriet / ALCIOM	Minor corrections

CONTENTS

1 Introduction

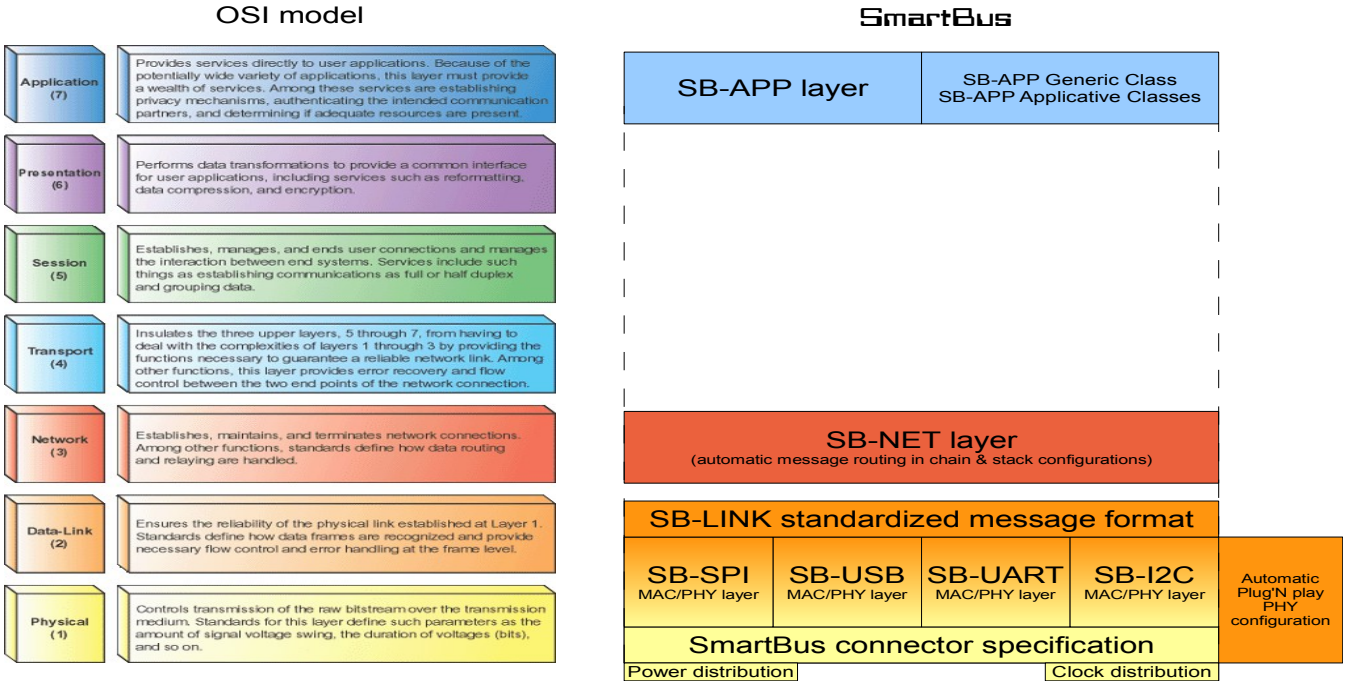
The SmartBrick products from ALCIOM are high performance analog and mixed-signal OEM modules. These modules are both daisy-chainable and stackable to build plug and play compact and efficient smart instrumentation and acquisition systems.





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 Low Level Class" applicative class, or class 0x10. This class of messages is a low-level command set, allowing a deep access to the hardware resources of the target module.

SB-APP Low Level Class is the applicative message class for the **SB-PROTO prototyping module**, but is also supported as well as by the majority of SmartBrick modules for specific low-level applications.

2 Class presentation

2.1 Class objectives

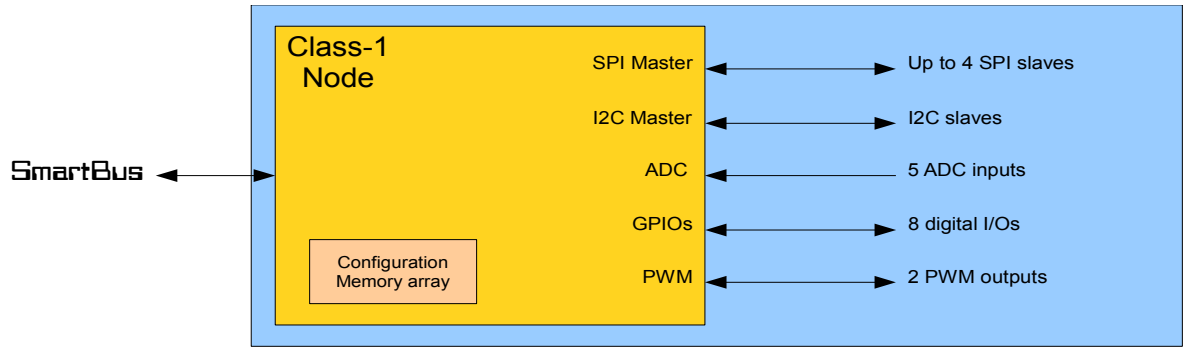
The SB-APP Low Level Class is aimed at providing a low level access to the hardware components of compliant SmartBrick module. This class is mainly used internally by ALCIOM for factory operations. This class is also the only supported class for the **SB-PROTO** prototyping module : Through this class the host can sends any low level commands to the built-in peripherals or to any custom peripherals through the supported communication interfaces.

As the SB-PROTO architecture is also the basis of many SmartBrick modules this class is also supported natively by several modules. However end-users are not expected to use it for these modules as the internal architecture and peripherals of SmartBrick modules is not documented and could evolve between different versions of the same module. Specific applications could however be discussed specifically with ALCIOM.

WARNING : misuse of the Low Level Class with a SmartBrick module can potentially destroy the module or its calibration parameters. Be informed that the use of any Low Level command is registered in non-volatile memory inside any module and will definitively void the warranty.

2.2 Module conceptual view

From the SB-APP low level Class view point, the model of a compatible SmartBrick module is the following :



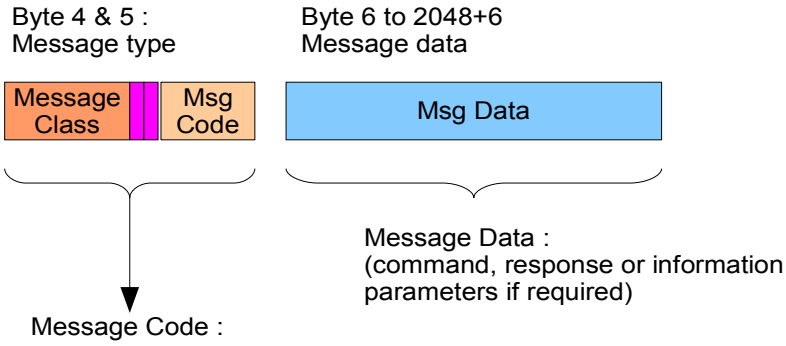
The target module therefore consist in several low level interfaces, which will be individually accessible through the provided messages, as well as a non-volatile memory array which can be read and written. The content of this memory can be freely used on the SB-PROTO module, but will contain application-related parameters (like undocumented factory calibration data) for other modules so shouldn't be altered.

2.3 Class identifier

The SB-APP Low Level Class is identified by the following value

Class	Class identifier
SB-APP Low Level Class	0x10

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 :



- First byte identifies message class
(for commands it must match the module supported classes, see text)

- bits 7 and 6 of 2nd byte identifies the message type :
00 : Command, 10 : Response, 01:Information, 11:(reserved)

- bits 5 to 0 of the 2nd byte identifies the command, response or information within the message class

2.4 Supported modules

The SB-APP Low Level Class is officially supported by the following modules :

Class	Supported modules
SB-APP Low Level Class	SB_PROTO
SB-APP Low Level Class	All modules that shares the same base as SB_PROTO

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 provides a detailed specification of all Commands, Responses and Indication supported by SmartBrick modules compliant to the **"SB-APP Low Level Class"** model.

3.1 SPI bus management

3.1.1 SPI configure Command

Description : Configure the SPI peripheral for future transfers

Command Format :

Message class	Message code	Message data	
0x10	0x01	SMP	SDI Sample Phase (1 byte) 0x01 = Input data sampled at end of data output time 0x00 = Input data sampled at middle of data output time
		CKE	SPIx Clock Edge Select (1 byte) 0x01 = Serial output data changes on transition from active clock state to Idle clock state (see bit 6) 0x00 = Serial output data changes on transition from Idle clock state to active clock state
		CKP	Clock Polarity Select (1 byte) 0x01 = Idle state for clock is a high level; active state is a low level 0x00 = Idle state for clock is a low level; active state is a high level
		Speed	SPI bus sampling speed in MHz (0x00 to 0x1F) bit 4-2 : Secondary Prescale 111 = 1/1 110 = 2/1 ... 000 = 8/1 bit 1-0 : Primary Prescale 11 = 1/1 10 = 4/1 01 = 16/1 00 = 64/1

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

Message class	Message code	Message data	
0x10	0x01	Error code	0x00 if execution ok, error code if not (cf 4)

3.1.2 SPI send/receive Command

Description : Do a SPI transaction on one of the 4 available SPI slaves

Command Format :

Message class	Message code	Message data	
0x10	0x02	SlaveId	Chip Select line to use (1 byte, $01 to $04)
		ByteRxCnt_MSB ByteRxCnt_LSB	MSB of Number of bytes to receive (1 byte) LSB of Number of bytes to receive (1 byte) (the number of actual SPI cycles will be the max of this number and the number of bytes to send)
		BytesToSend	0 to 2043 bytes to send to the specified slave (0x00's will be sent if the provided bytes are less than ByteCnt, extra bytes will be ignored)

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

Message class	Message code	Message data	
0x10	0x02	Error code	0x00 if execution ok, error code if not (cf 4)
		SlaveId	Chip Select line used (1 byte, $01 to $04)
		BytesReceived	ByteRxCnt bytes (0 to 2043)

3.2 I2C bus management

3.2.1 I2C configure Command

Description : Configure the I2C master module

Command Format :

Message class	Message code	Message data	
0x10	0x10	Speed	0x04 for 400kbps, 0x01 for 100Khz (1 byte)

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

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

3.2.2 I2C read Command

Description : Do a I2C read of a specific I2C slave

Command Format :

Message class	Message code	Message data	
0x10	0x11	I2CSlaveId	I2C slave address (1 byte, 0x00 to 0x7F)
		ByteRxCnt	Number of bytes to read (1 byte, 0x00 to 0xFF)

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

Message class	Message code	Message data	
0x10	0x11	Error code	0x00 if execution ok, error code if not (cf 4)
		I2CSlaveId	I2C slave address (1 byte, 0x00 to 0x7F)
		BytesReceived	Number of read bytes (0 to 256) (could be lower than requested in case of STOP condition on the bus)

3.2.3 I2C write Command

Description : Do a I2C write to a specific I2C slave

Command Format :

Message class	Message code	Message data	
0x10	0x12	I2CSlaveId	I2C slave address (1 byte, 0x00 to 0x7F)
		BytesToWrite	1 to 256 bytes to write to the specified slave

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

Message class	Message code	Message data	
0x10	0x12	Error code	0x00 if execution ok, error code if not (cf 4)
		I2CSlaveId	I2C slave address (0x00 to 0x7F)

3.3 ADC management

3.3.1 ADC read Command

Description : Read the five ADC channels

Command Format :

Message class	Message code	Message data
0x10	0x18	None

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

Message class	Message code	Message data	
0x10	0x18	Error code	0x00 if execution ok, error code if not (cf 4)
		ADC1_MSB ADC1_LSB	ADC values, 10 bits for each input channel coded on 2 bytes
		ADC2_MSB ADC2_LSB	
		ADC3_MSB ADC3_LSB	
		ADC4_MSB ADC4_LSB	
		ADC5_MSB ADC5_LSB	

3.4 GPIO management

3.4.1 GPIO configure Command

Description : Configure the GPIO pins

Command Format :

Message class	Message code	Message data	
0x10	0x20	Direction (1 byte)	One bit per GPIO line, allows to configure each line as input or output. If the data direction bit is a 1, then the pin is an input (input by default)

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

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

3.4.2 GPIO set Command

Description : Set the level of the GPIO pins configured as outputs

Command Format :

Message class	Message code	Message data	
0x10	0x21	Value (1 byte)	One bit per GPIO line, allows to configure each output line at level 0 or 1. Bits for inputs are ignored

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

Message class	Message code	Message data	
0x10	0x21	Error code	0x00 if execution ok, error code if not (cf 4)

3.4.3 GPIO get Command

Description : Get the level of the GPIO pins configured as inputs

Command Format :

Message class	Message code	Message data
0x10	0x22	None

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

Message class	Message code	Message data	
0x10	0x22	Error code	0x00 if execution ok, error code if not (cf 4)
		Value (1 byte)	One bit per GPIO line, gives the level of the pins configured as inputs, and the set level for outputs

3.5 PWM management

3.5.1 PWM set Command

Description : Configure the PWM frequencies and ratios

Command Format :

Message class	Message code	Message data
0x10	0x28	PWM1PERIOD_MSB PWM1PERIOD_LSB PWM period for output 1 PWM1ONPER_MSB PWM1ONPER_LSB "ON" period for output 1 PWM2PERIOD_MSB PWM2PERIOD_LSB PWM period for output 2 PWM2ONPER_MSB PWM2ONPER_LSB "ON" period for output 2 Nota : - periods are in 1/16MHz units : 65000 corresponds to 246Hz - on period must be lower than the PWM period

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

Message class	Message code	Message data
0x10	0x28	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, ie 0x00 to 0x2F, please refer to the overall SmartBus specification)

Error code	Description	Comment
0x30	Unsupported bus speed	
0x40	I2C slave not responding	
0x41	I2C transaction error	