APPLICATION NOTE

AT02698: ZAppSI User Guide

Features

- Splitting application logic and network functionality between two processors
- Same BitCloud API in the application code
- Serial connection between the host and the network processor
- Ready network processor's application

Introduction

The document describes ZAppSI (Zigbee Application Serial Interface), a solution that helps to split ZigBee network functions and the application logic between two platforms: one serving as the network processor, and the other being the host, the two connected by a serial interface. The application is executed on the host, which may be anything including a PC. Application's code is essentially the same as when the application runs on a single processor on top of BitCloud.

Chapter 1 gives a short overview and lists supported platforms. The architecture of the BitCloud ZAppSI solution in described in Chapter 2. Instructions on how to configure and run a ZAppSi-based application are given in Chapter 3. And Chapter 4 describes briefly how to port ZAppSI host environment to an arbitrary platform.

Table of Contents

1. Overview

ZigBee Application Serial Interface (ZAppSI) solution is provided as a part of BitCloud SDK [1]. It allows splitting ZigBee network functions and the application logic between two platforms: one serving as the network processor, and the other being the host, the two connected by a serial interface. The application is executed on the host, which may be anything including a PC. Application's code is essentially the same as when the application runs on a single processor on top of BitCloud. For details on BitCloud reference applications see [3], while development concepts and API can be found in [4] and [2].

Table 1-1 lists all platforms supported as a network processor and a host. Note that host side of ZAppSI is available as source code and, thus, can be ported by the user to other host platforms as well.

Table 1-1. Platforms, packages and interfaces supported by ZAppSI.

Platform	BitCloud packages	Network processor	Host	Serial interfaces
ATmega256RFR2 ATmega2564RFR2	BitCloud SDK for megaRF	x		USART(1) SPI USBFIFO
PC	All aforementioned packages		x (HADevices application)	Serial COM(2)

Notes: 1. The interface used in pre-compiled binaries and as default setting in source code.
2. NP may use a serial interface different from the one supported by a PC (for example, USART), but communicate with the PC through some kind of a bridge device.

BitCloud reference applications include the project files for compiling them as ZAppSI host application. Note that the host application uses the API provided by the host side of ZAppSI, which is exactly the same as the BitCloud API used by common applications.

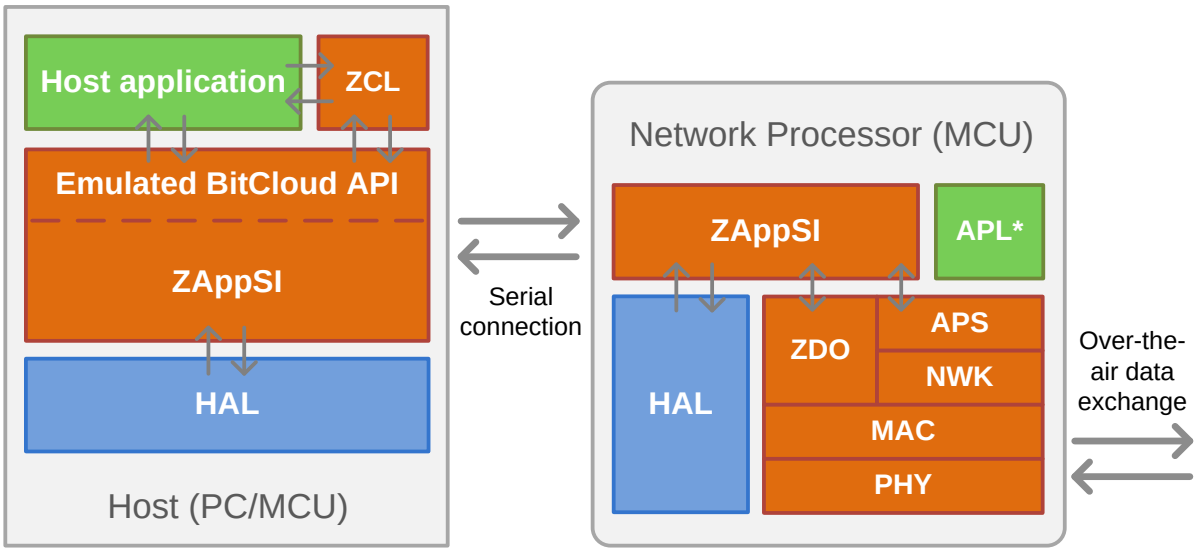
2. Architecture

ZAppSI is used for splitting execution of application logic and network functionality between two different processors: the host and the network processor. The host may be an MCU or a PC, and the network processor is a platform consisting of an MCU and a radio device.

The host's side includes the ZAppSI component, which emulates the APIs of other BitCloud components (except for ZCL which is a part of the host application). Invocation of any API function by an application will be correctly handled by ZAppSI. Because of this, the application code should not be changed when porting the application from a single-processor environment. An exception is made for compile-time configuration, which is partly moved from the application to the network processor – see Section 3.2. The network processor's side includes all BitCloud components (except for ZCL), ZAppSI, and the ZAppSINP application.

ZAppSI architecture is shown in Figure 2-1.

Figure 2-1. ZAppSI architecture: interactions between host's and network processor's components.



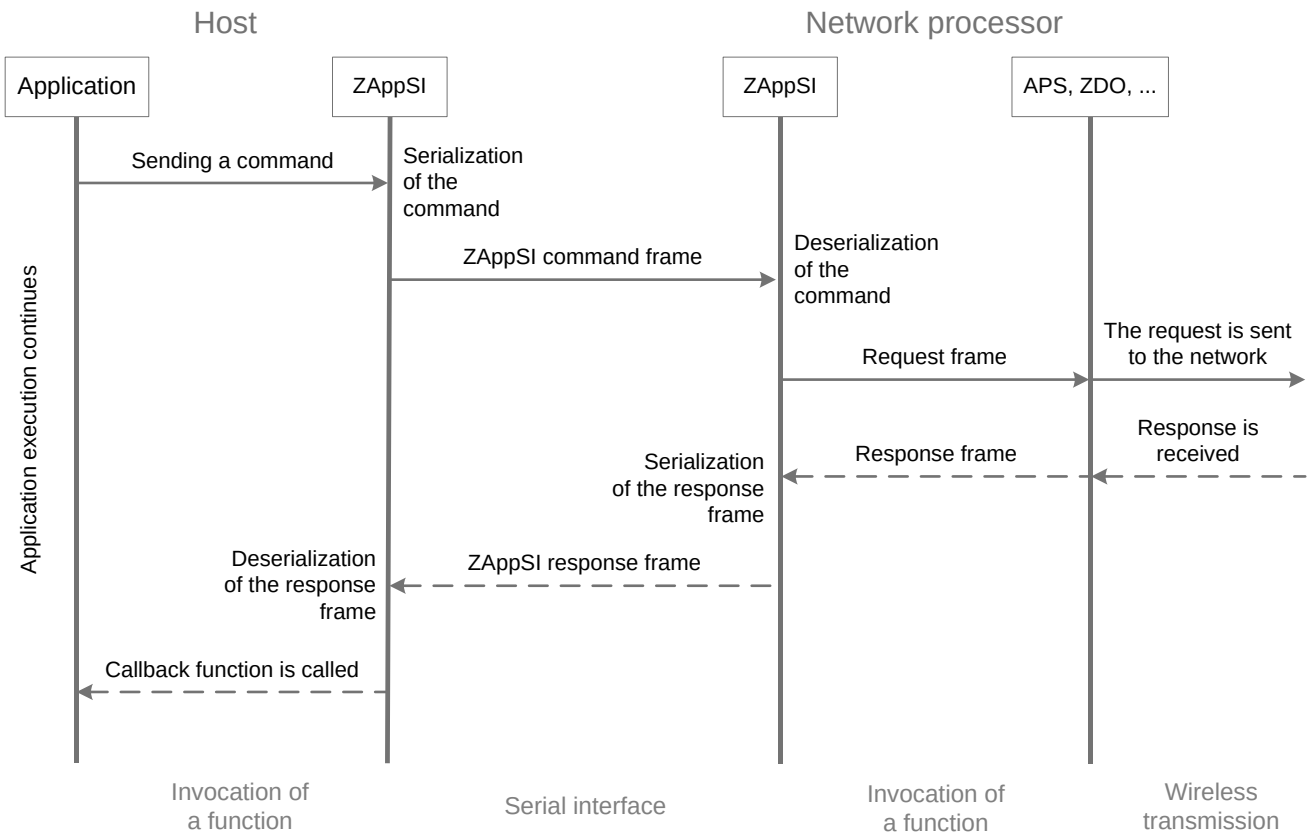
* Should not be modified by the user

Requests issued by the host application to BitCloud components (except for ZCL, which is a part of the host application) are serialized by ZAppSI and transferred over a serial connection to the network processor. On the network processor's side, ZAppSI deserializes these requests and passes them to corresponding BitCloud component for execution. The response, once it is ready, is sent in the opposite direction: from the BitCloud core to ZAppSI, to ZAppSI on the host, to the host application. If an asynchronous request has been sent, the callback function provided by the application is called when the response is received and deserialized. In case of synchronous requests, program execution on the host side is blocked until the response is received from the network processor.

Note: ZCL runs on the host's side, wrapping application requests into APS requests and passing them to ZAppSI.

Figure 2-2 illustrates processing of an asynchronous request sent by the application.

Figure 2-2. Processing of an asynchronous request by ZAppSI.



The network processor's ZAppSINP application is provided in pre-built images, as well as in source code, but, generally, the source code should not be changed. The user may only need to change some compile-time parameters affecting sizes of BitCloud internal tables and buffers in the `configuration.h` file of the ZAppSINP application (see Section 3.2).

3. Applications

This chapter describes how to configure and run host applications and network processor's application.

3.1 Host configuration

For a ZAppSI host, a BitCloud application should be compiled under a special ZAppSI host project. For example, PC as the host platform there are Code Blocks projects located in `HADevice\codeBlocksPrj\` directory. Within such project, core stack components (APS, ZDO, etc.) are not included in the build. The application is built with ZAppSI, System Environment, and ZCL which are provided in source code, and linked with the HAL library for the host platform.

HAL is also provided in source code, but it should be compiled separately from the application. To compile HAL, use Makefile located in the SDK's `\BitCloud\Components\HAL\` directory: from the command line in this directory execute `make clean all`. As a result of the compilation, HAL library files will appear in the `\BitCloud\lib\` directory.

How to set Configuration Server parameters for a host is described in Section 3.3.

3.2 Network processor configuration

The embedded application ZAppSINP is expected to run on the network processor device. The application itself contains no logic, which is concentrated in ZAppSI and BitCloud components. Generally, there is no need to modify the source code of ZAppSINP application, but the user may need to change network processor's configuration in the `configuration.h` file located in the `.\Applications\ZAppSINP\` directory. For details see Section 3.3.

Note that routines related to network security also reside on the network processor. The level of security applied for network authentication: no security, standard security, standard security with link keys depends on BitCloud libraries the network processor's application is built with.

3.3 Stack parameters configuration

When ZAppSI is used a part of the ConfigServer parameters are set on the host and a part on the network processor. The parameters are set in `configuration.h` files located in the root directory of the host application, for the host, and the ZAppSINP application, for the network processor.

The network processor's parameters that can be changed at runtime are configured on the host's side. On application's startup these parameters are transferred over the serial interface to the network processor by ZAppSI and set in the ConfigServer.

Compile-time parameters related to components residing on the network processor (APS, ZDO, NWK, and MAC) are set on the network processor. These parameters are mostly sizes of internal tables (such as neighbor table and routing table) and buffers employed by the components to store incoming commands. Values provided for these parameters in the host application's `configuration.h` file will be ignored.

Refer to [2] and [4] and to find out which ConfigServer parameters can be changed at runtime and which can be not.

3.4 Running an application on ZAppSI

To run a BitCloud application on two processors:

Host application:

1. In the `configuration.h` file of the host application configure the serial interface used for communication with network processor:

a. Specify the interface type in the `APP_ZAPPSI_INTERFACE` parameter, setting either to `APP_INTERFACE_USART`, `APP_INTERFACE_SPI`, or `APP_INTERFACE_USBFIFO`. The interface type should match the interface set on the network processor (see interfaces used by default in Table 1-1) unless there is a kind of a bridge device connecting two interfaces between the host and the network processor.
b. Specify the channel to be used by the interface (if applicable) in the `APP_ZAPPSI_MEDIUM_CHANNEL` parameter. For example for a PC host it is likely to be a COM port:

```
#define APP_ZAPPSI_MEDIUM_CHANNEL COM1
```

2. In the `configuration.h` file of the ZAppSINP application set values for application and ZCL parameters if their default values are not appropriate.
3. Compile the host application.

ZAppSINP application:

4. In the `configuration.h` file of the ZAppSINP application, configure the serial interface used for communication with the host:
   a. Specify the interface type in the `APP_ZAPPSI_INTERFACE` parameter, setting either to `APP_INTERFACE_USART`, `APP_INTERFACE_SPI`, or `APP_INTERFACE_USBFIFO`.
   b. Specify the channel to be used by the interface (if applicable) in the `APP_ZAPPSI_MEDIUM_CHANNEL` parameter. For example:

```
#define APP_ZAPPSI_MEDIUM_CHANNEL USART_CHANNEL_1
```

5. In the `configuration.h` file of the ZAppSINP application set compile-time parameters if their default values are not appropriate.
6. If any changes have been made in the `configuration.h` file of the ZAppSINP application recompile ZAppSINP with the required security level.

7. Connect the host and the network processor via a serial cable.
8. Program the network processor device with the ZAppSINP's firmware.
9. Program the host device with firmware generated for it.
10. Restart devices (if a PC is the host then simply run the corresponding `.exe` file on it).

4. Porting the host application

When the application logic is executed separately from network functions on the host, it is assumed that the host device may be of any platform. The solution provided with BitCloud demonstrates out-of-the-box support for a limited number of platforms (see Chapter 1). However, ZAppSI may be ported to an unsupported platform without much effort, by implementing Hardware Abstraction Layer (HAL) for this platform and creating application project.

Though the standard version of HAL supports a wide range of interfaces, a version for a host platform needs to implement only a few interfaces: those that are used by ZAppSI, ZCL, and System Environment. As an example, the user may take implementation of HAL for one of the supported host platforms (for instance, for SAM3S), which are also provided with SDKs for all network processor platforms. A good example is HAL for PC, since it implements only the intefaces required by the host environment.

HAL sources are located in the `\BitCloud\Components\HAL\` directory inside the SDK. The `\include\` directory from this location contains header files for all supported interfaces. Other directories contain implementations of HAL interfaces for specific platforms. HAL for the host platform shall implement the following items:

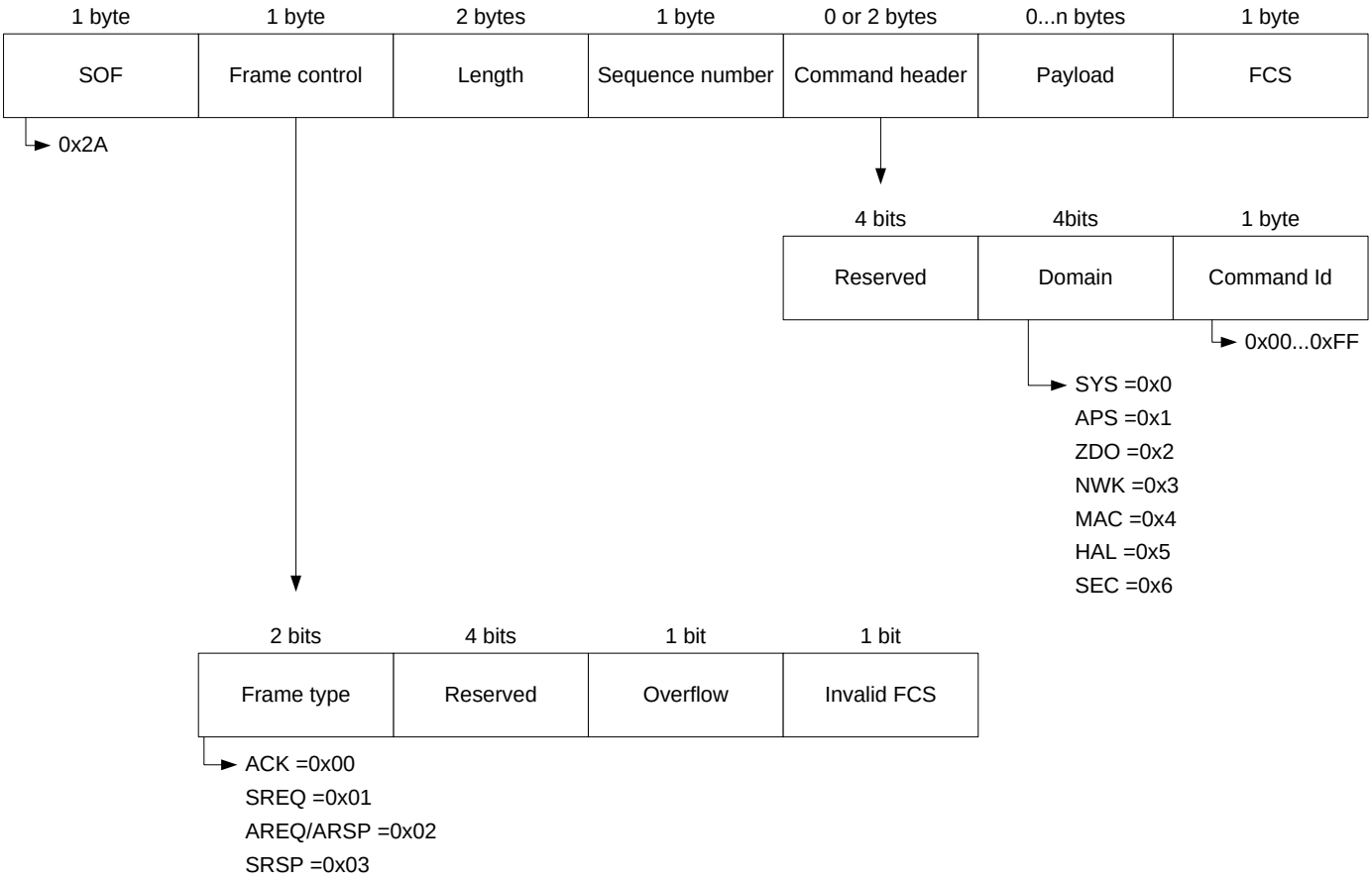
- The `HAL_TaskHandler()` function (called by System Environment to process HAL tasks): refer to `\BitCloud\Components\HAL\pc\common\src\halTaskManager.c` file for example implementation
- Hardware reset functions: the `resetReason.h` file
- Application timer: the `appTimer.h` file.
- Serial interface for communication with the network processor: USART (`usart.h`), SPI (`spi.h`), or USBFIFO (`usb.h`)

HAL compilation options are set in `Configuration`, `Makerules`, and `Makefile` files located in the HAL's directory. These files shall be also modified if a new hardware platform is added.

Appendix A. Serial frame format

Figure A-1 shows the structure of ZAppSI frames exchanged by the host and the network processor over the serial interface connecting them.

Figure A-1. The structure of serial frames exchanged by the host and the network processor.



5. Reference

[1] BitCloud SDK

[2] BitCloud API Reference (available in BitCloud SDK)

[3] AVR2052: Atmel BitCloud Quick Start Guide

[4] AVR2050: BitCloud Developer's Guide

6. Document revision history

Doc. Rev.	Date	Comment
A	15.02.2014	First release version.

Atmel | Enabling Unlimited Possibilities®

Atmel Corporation
1600 Technology Drive
San Jose, CA 95110
USA
Tel: (+1)(408) 441-0311
Fax: (+1)(408) 487-2600
www.atmel.com

Atmel Asia Limited
Unit 01-5 & 16, 19F
BEA Tower, Millennium City 5
418 Kwun Tong Road
Kwun Tong, Kowloon
HONG KONG
Tel: (+852) 2245-6100
Fax: (+852) 2722-1369

Atmel Munich GmbH
Business Campus
Parkring 4
D-85748 Garching b. Munich
GERMANY
Tel: (+49) 89-31970-0
Fax: (+49) 89-3194621

Atmel Japan G.K.
16F Shin-Osaki Kangyo Bldg.
1-6-4 Osaki, Shinagawa-ku
Tokyo 141-0032
JAPAN
Tel: (+81)(3) 6417-0300
Fax: (+81)(3) 6417-0370