

ZigBee PRO Network Processor

Accelerate your ZigBee Development

Applications

- ZigBee™ systems
- Home/Building automation
- Industrial control and monitoring
- Low power wireless sensor networks
- Set-top boxes and remote controls
- Automated Meter Reading

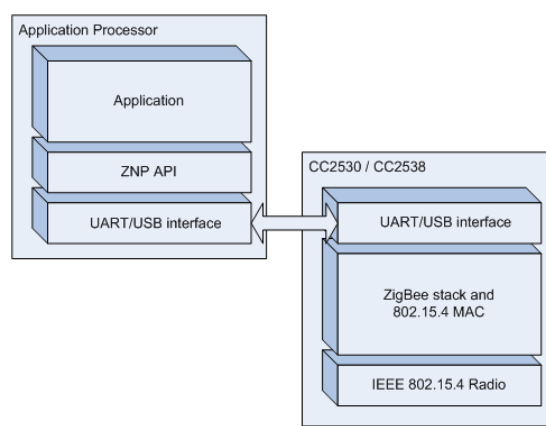
Description

The **Z-Stack ZNP** is a cost-effective, low power, ZigBee Network Processor that provides full ZigBee functionality with a minimal development effort.

In this solution, the ZigBee PRO stack runs on a SoC and the application runs on an external microcontroller. The **Z-Stack ZNP** handles all the ZigBee protocol tasks, and leaves the resources of the application microcontroller free to handle the application.

This makes it easy for users to add ZigBee to new or existing products at the same time as it provides great flexibility in choice of microcontroller.

Z-Stack ZNP interfaces to any microcontroller through a range of serial interfaces.



Key Features

- All the powerful features of the ZigBee PRO system-on-chip with a simplified application interface.
- UART¹ or USB² interface to application processor.
- Access to 12-bit analog-to-digital converter, GPIO pins, non-volatile memory

¹ UART interface is supported on CC2530 and CC2538.

² USB interface is supported on CC2538 and CC2531.

APPLICATIONS	1
DESCRIPTION	1
KEY FEATURES	1
REFERENCES	3
ACRONYMS.....	3
1 INTRODUCTION	4
2 PHYSICAL INTERFACE	5
2.1 CC2538.....	5
2.1.1 <i>Network processor signals</i>	5
2.1.1.1 Pin Configurations	5
2.1.2 <i>Interface Configuration</i>	6
2.1.2.1 IAR project configuration	6
2.1.3 <i>UART Transport</i>	7
2.1.3.1 Configuration	7
2.1.3.2 Frame Format.....	7
2.1.3.3 Sample FCS Calculation	8
2.1.3.4 Signal Description.....	8
2.1.3.5 Signal Operation	9
2.1.4 <i>General Frame Format</i>	9
2.1.5 <i>Initialization Procedures</i>	11
2.1.5.1 CC2538-ZNP power-up procedure	11
2.2 CC2530.....	11
2.2.1 <i>Network processor signals</i>	11
2.2.1.1 Pin Configurations	12
2.2.2 <i>Interface Configuration</i>	14
2.2.2.1 IAR project configuration	14
2.2.3 <i>UART Transport</i>	14
2.2.3.1 Configuration	14
2.2.3.2 Frame Format.....	14
2.2.3.3 Sample FCS Calculation	14
2.2.3.4 Signal Description.....	14
2.2.3.5 Signal Operation	15
2.2.4 <i>General Frame Format</i>	15
2.2.5 <i>Initialization Procedures</i>	15
2.2.5.1 CC2530-ZNP power-up procedure	15
3 ZNP SOFTWARE COMMAND INTERFACE.....	16
3.1 CONFIGURATION INTERFACE	16
3.1.1 <i>Device specific configuration parameters</i>	16
3.1.1.1 ZCD_NV_STARTUP_OPTION.....	16
3.1.1.2 ZCD_NV_LOGICAL_TYPE.....	17
3.1.1.3 ZCD_NV_ZDO_DIRECT_CB	17
3.1.2 <i>Network specific configuration parameters</i>	17
3.1.2.1 ZCD_NV_PANID.....	17
3.2 Z-STACK 3.0 ZNP CONSIDERATIONS	18
3.2.1 <i>Backward compatibility</i>	18
3.2.2 <i>ZNP for Z3.0</i>	18
3.2.3 <i>ZNP startup procedure for Z3.0 implementation</i>	18
3.3 RETURN VALUES	20
3.4 ADDITIONAL CONSIDERATIONS FOR ZNP DEVICE IN Z-STACK 3.0.....	21
3.5 ADDITIONAL INFORMATION	21
4 GENERAL INFORMATION.....	22
4.1 DOCUMENT HISTORY	22

References

- [R1] Z-Stack Monitor and Test API
- [R2] CC2538 Online Documentation : <http://www.ti.com/product/cc2538>
- [R3] CC2530 Online Documentation : <http://www.ti.com/product/cc2530>
- [R4] CC2531 Online Documentation : <http://www.ti.com/product/cc2531>
- [R5] CC2591 Online Documentation : <http://www.ti.com/product/cc2591>
- [R6] CC2592 Online Documentation : <http://www.ti.com/product/cc2592>

Acronyms

AF	ZigBee Application Framework
API	Application Programming Interface
AREQ	Asynchronous Request
BDB	Base Device Behavior
CTS	Clear To Send
FCS	Frame Check Sequence
GP	Green Power
GPIO	General Purpose I/O
NPI	Network Processor Interface
NV	Non-Volatile
PA/LNA	Power Amplifier / Low Noise Amplifier (CC259x)
RTS	Ready To Send
SoC	System on Chip
SREQ	Synchronous request
SRSP	Synchronous response
UART	Universal Asynchronous Receiver Transmitter
ZDO	ZigBee Device Object
ZNP	ZigBee Network Processor

1 Introduction

This document consolidates the ZNP Interface Specifications for the following devices of the ZigBee® SimpleLink family: CC2530, CC2531 and CC2538.

To reduce duplicated and redundant information, simple summaries are provided here and references to external documents are provided where necessary.

2 Physical Interface

The following sections describe the physical interface for the ZNP for each of the supported platforms.

2.1 CC2538

2.1.1 Network processor signals

The figure below shows how an application processor interfaces with the CC2538.

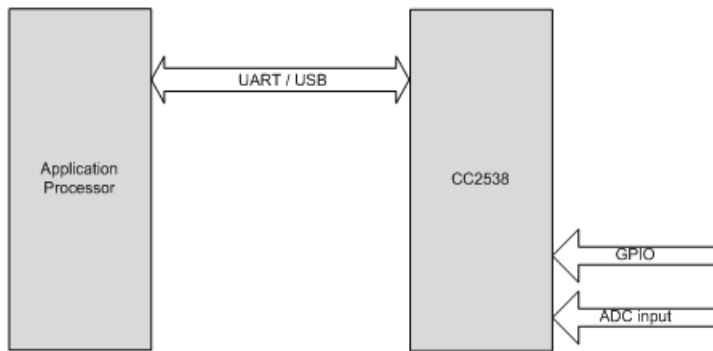


Figure 1 CC2538 Interface

The CC2538-ZNP uses the following signals for the hardware interface

- **RX/TX/RTS/CTS for UART:** UART communication. See section 2.1.3.4 for details.

2.1.1.1 Pin Configurations

The CC2538-ZNP Pin configurations are described in the following sections.

2.1.1.1.1 Default pin configuration

By default, the Pin Configurations are the following:

Transport	CC2538-ZNP signal	CC2538 PIN	CC2538 NAME	Direction (on C2538)
UART	TX	P1_09	PA1	Out
UART	RX	P1_07	PA0	In
UART	CTS	P1_03	PB0	In
UART	RTS	P2_18	PD3	Out
UART	GND	GND	GND	In/Out

2.1.1.1.2 Alternate pin configuration

Go to Project-> Options-> C/C++Compiler-> Preprocessor-> DefinedSymbols and add ZNP_ALT. The ZNP_ALT supports RESET pin. The Pin Configurations is the following:

Transport	CC2538-ZNP signal	CC2538 PIN	CC2538 NAME	Direction (on C2538)
UART	TX	P1_09	PA1	Out
UART	RX	P1_07	PA0	In
UART	CTS	N/A	N/A	In
UART	RTS	N/A	N/A	Out
UART	GND	GND	GND	In/Out

2.1.1.1.3 USB pin configuration

In this configuration, the CC2538-ZNP will use the USB transport. The pin-out of the CC2538 can be found in the datasheet. The USB transport exposes the CDC (communication device class) USB interface and exposes a virtual COM port to the host. The host processor would then access this device as a regular COM port device and communicate with the ZNP using the UART transport. For more information on the USB driver go to CC2538 foundation firmware package [R2]. For the USB to survive system reset, Soft Reset has been introduced. For more information on SYS_RESET_REQ please refer to [R1].

2.1.2 Interface Configuration

The CC2538-ZNP supports UART or USB interface to the application processor.

2.1.2.1 IAR project configuration

The CC2538-ZNP IAR project that is included in the ZStack software package can be configured to UART or USB by using preprocessor defines. Go to Project->Options->C/C++ Compiler->Preprocessor->Defined Symbols and configure as following:-

For USB, add "HAL_UART_USB" and "USB_SETUP_MAX_NUMBER_OF_INTERFACES=5". This is out-of-the box configuration.

For UART, delete "HAL_UART_USB" and "USB_SETUP_MAX_NUMBER_OF_INTERFACES=5". No addition of preprocessor defines is needed to setup as UART.

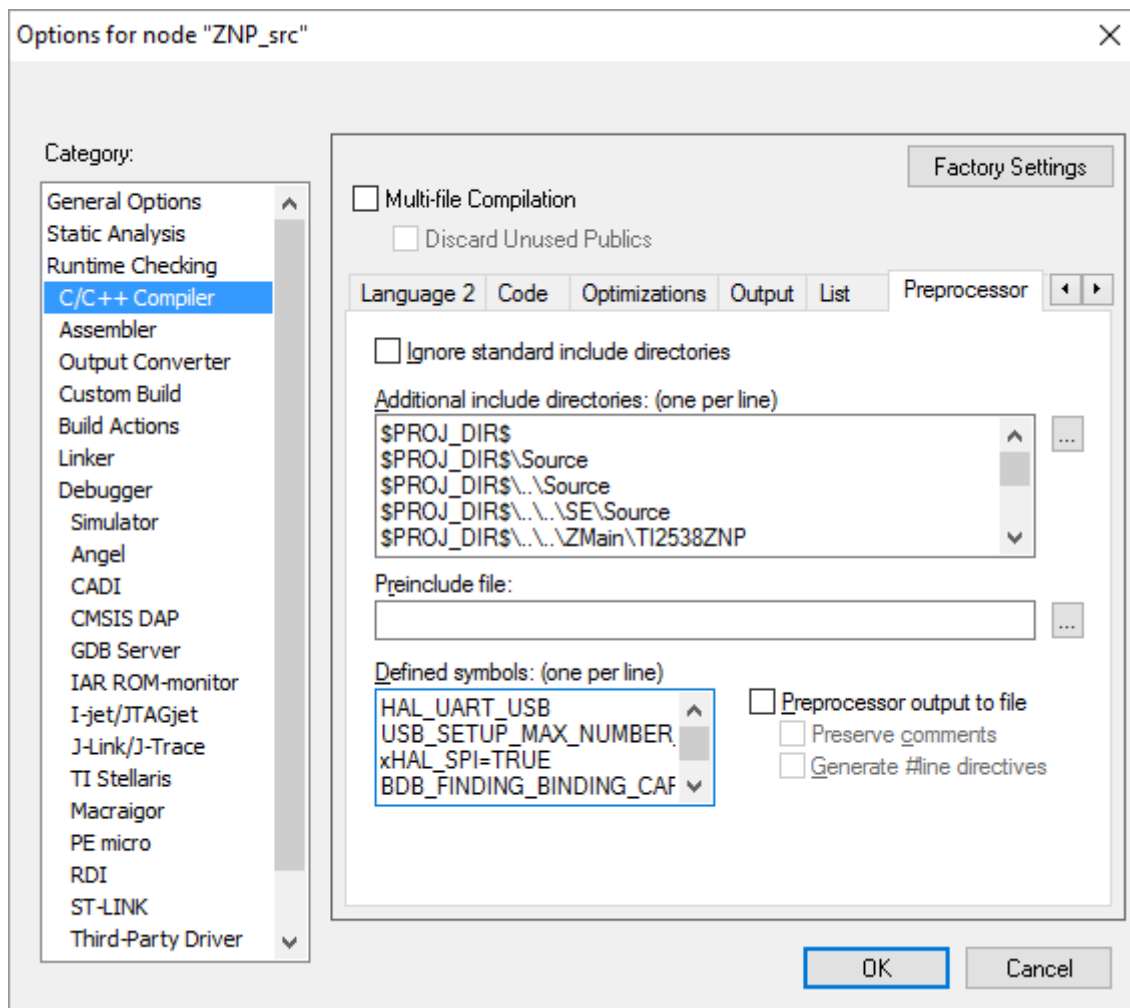


Figure 2 IAR Setup for CC2538 ZNP

2.1.3 UART Transport

2.1.3.1 Configuration

The following UART configuration is supported:

- Baud rate: 115200
- Hardware (RTS/CTS) flow control.
- 8-N-1 byte format.

2.1.3.2 Frame Format

UART transport frame format is shown in the following figure. The left-most field is transmitted first over the wire. This is the same General Serial Packet defined by the Monitor and Test (MT) specification [R1].

Bytes: 1	3-253	1
SOF	General format frame	FCS

Figure 3 UART Transport Frame Format

SOF: Start of frame indicator. This is always set to 0xFE.

General frame format: This is the general frame format as described in **Error! Reference source not found..**

FCS: Frame-check sequence. This field is computed as an XOR of all the bytes in the general format frame fields.

2.1.3.3 Sample FCS Calculation

Shown below is a C example for the FCS calculation:

```
unsigned char calcFCS(unsigned char *pMsg, unsigned char len)
{
    unsigned char result = 0;
    while (len--)
    {
        result ^= *pMsg++;
    }
    return result;
}
```

2.1.3.4 Signal Description

The following standard UART signals are used:

- TX: Transmit data.
- RX: Receive data.
- CTS: Clear to send.
- RTS: Ready to send.

Figure below shows the RTS/CTS flow control connections to the host processor. On the CC2538, RTS and CTS are active-low signals. The RT output is driven low when the receive register is empty and reception is enabled. Transmission of a byte does not occur before the CTS input is low.

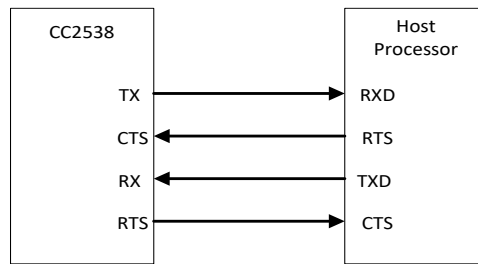


Figure 4 RTS/CTS Flow Control Connections

2.1.3.5 Signal Operation

UART transport sends and receives data asynchronously. Data can be sent and received simultaneously and the transfer of a frame can be initiated at any time by either the application processor or the ZNP SoC.

2.1.4 General Frame Format

The general frame format is shown in the following figure. The left-most field is transmitted first over the wire. For multi-byte fields, the lowest order byte is transmitted first. This is the same General Frame Format defined by the Monitor and Test (MT) specification [R1].

Bytes: 1	2	0-250
Length	Command	Data

Figure 5 General Frame Format

Length: The length of the data field of the frame. The length can range from 0-250.

Command: The command of the frame.

Data: The frame data. This depends on the command field and is described for each command in Section 3.

2.1.4.1.1 Command Field

The command field is constructed of two bytes. The bytes are formatted as shown in the following figure. The Cmd0 byte is transmitted first.

Cmd0		Cmd1	
Bits: 7-5	4-0	7-0	
Type	Subsystem	ID	

Figure 6 Command Field

Type: The command type has one of the following values:

- **0: POLL.** For CC2630, this command type is not supported.
- **1: SREQ:** A synchronous request that requires an immediate response. For example, a function call with a return value would use an SREQ command.
- **2: AREQ:** An asynchronous request. For example, a callback event or a function call with no return value would use an AREQ command.
- **3: SRSP:** A synchronous response. This type of command is only sent in response to a SREQ command. For an SRSP command the subsystem and ID are set to the same values as the corresponding SREQ. The length of an SRSP is generally nonzero, so an SRSP with length=0 can be used to indicate an error.
- **4-7: Reserved.**

Subsystem: The subsystem of the command. Values are shown below:

Subsystem Value	Subsystem Name
0	RPC Error interface
1	SYS interface
2	Reserved
3	Reserved
4	AF interface
5	ZDO interface
6	Simple API interface
7	UTIL interface
8	Reserved
9	APP Interface
10-31	Reserved

ID: The command ID. The ID maps to a particular interface message. Value range: 0-255.

When the ZNP cannot recognize an SREQ command from the host processor, the following SRSP is returned:

SRSP:

1	1	1	1	1	1
Length = 0x03	Cmd0 = 0x60	Cmd1 = 0x00	ErrorCode	ReqCmd0	ReqCmd1

Attributes:

Attribute	Length (byte)	Description	
ErrorCode	1	The error code maps to one of the following enumerated values.	
		Value	Description
		0x01	Invalid subsystem
		0x02	Invalid command ID
		0x03	Invalid parameter
		0x04	Invalid length
ReqCmd0	1	The Cmd0 value of the processed SREQ	
ReqCmd1	1	The Cmd1 value of the processed SREQ	

2.1.5 Initialization Procedures

2.1.5.1 CC2538-ZNP power-up procedure

The recommended power-up procedure is as follows:

1. Application processor and CC2538 power up.
2. For ZNP_ALT configuration, the application processor sets CC2538 EM_RESET pin low, holding CC2538 in reset. Please note that step 2 is only for ZNP Alternate configuration.
3. The application processor initializes its UART interface.
4. For ZNP_ALT configuration, the application processor sets CC2538 EM_RESET pin high and CC2538 starts operation. Please ignore this step if not using ZNP_ALT preprocessor define.
5. The application processor receives the `SYS_RESET_IND` message.

The CC2538-ZNP can be reset when the application processor sends a `SYS_RESET_REQ` message.

2.2 CC2530

2.2.1 Network processor signals

The figure below shows how an application processor interfaces with the CC2530.

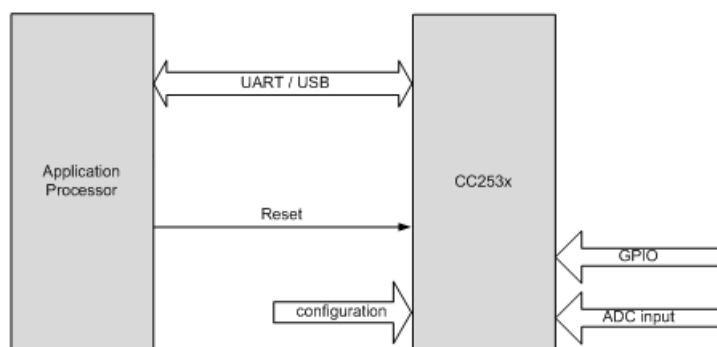


Figure 7 CC2530 Interface

The CC2530-ZNP uses the following signals for the hardware interface

- **RX/TX/RT/CT for UART:** These are the standard signals used for UART communication. See section 2.2.3.4 (for UART) for details.
- **RESET:** This signal is used by the application processor to reset the CC2530.
- **CFG0:** This signal is used to configure the CC2530-ZNP. The CC2530-ZNP reads this signal at power up and configures its operation accordingly. See section 2.2.1.1.1 for details.

2.2.1.1 Pin Configurations

2.2.1.1.1 Configuration pins

The CC2530-ZNP project reads the two hardware configuration pins at powerup and configures itself accordingly.

The CFG0 pin is used to indicate the presence (if pin is high) or absence of the 32kHz crystal connected to the CC2530-ZNP. This is the sleep crystal that is used to maintain accurate timing when the device is in sleep mode. The advantage of using this instead of the internal 32kHz oscillator is that it typically provides faster wakeup time for sleep and a lower power consumption during this time. If this crystal is not populated, then the CC2530 can use the internal RC oscillator.

2.2.1.1.2 Main pin configuration

CC2530-ZNP signal	CC2530 PIN	CC2530 NAME	Direction (on C2530)
CT	6	P1_4	In
RT	5	P1_5	In / Out
TX	38	P1_6	In / Out
RX	37	P1_7	Out / In
RESET	20	RESET_N	In
PAEN	9	P1_1	Out
EN	7	P1_3	Out
HGM	12	P0_7	Out
CFG0	8	P1_2	In
CFG1	36	P2_0	In
GPIO0/AIN0	19	P0_0	Configurable
GPIO1/AIN1	18	P0_1	Configurable

GPIO2	13	P0_6	<i>Configurable</i>
GPIO3	11	P1_0	<i>Configurable</i>

2.2.1.1.3 Alternate pin configuration

CC2530-ZNP signal	CC2530 PIN	CC2530 NAME	Direction (on C2530)
CT	15	P0_4	<i>In</i>
RT	14	P0_5	<i>In / Out</i>
TX	16	P0_3	<i>In / Out</i>
RX	17	P0_2	<i>Out / In</i>
RESET	20	RESET_N	<i>In</i>
PAEN	9	P1_1	<i>Out</i>
EN	6	P1_4	<i>Out</i>
HGM	12	P0_7	<i>Out</i>
CFG0	8	P1_2	<i>In</i>
CFG1	36	P2_0	<i>In</i>
GPIO0/AIN0	19	P0_0	<i>Configurable</i>
GPIO1/AIN1	18	P0_1	<i>Configurable</i>
GPIO2	13	P0_6	<i>Configurable</i>
GPIO3	11	P1_0	<i>Configurable</i>

2.2.1.1.4 ZNP Kit pin configuration

CC2530-ZNP signal	CC2530 PIN	CC2530 NAME	Direction (on C2530)
CT	15	P0_4	<i>In</i>
RT	14	P0_5	<i>In / Out</i>
TX	16	P0_3	<i>In / Out</i>
RX	17	P0_2	<i>Out / In</i>
RESET	20	RESET_N	<i>In</i>
PAEN	9	P1_1	<i>Out</i>
EN	6	P1_4	<i>Out</i>
HGM	12	P0_7	<i>Out</i>
CFG0	19	P0_0	<i>In</i>
CFG1	18	P0_1	<i>In</i>
GPIO0	13	P0_6	<i>Configurable</i>
GPIO1	12	P0_7	<i>Configurable</i>
GPIO2	38	P1_6	<i>Configurable</i>
GPIO3	37	P1_7	<i>Configurable</i>

2.2.1.1.5 USB pin configuration

This is only available when used with the CC2531 chip. In this configuration, the CC2531-ZNP will use the USB transport with the alternate pin configuration. The pin-out of the CC2531 can be found in the datasheet [R4]. The USB transport exposes the CDC (communication device class)

USB interface and exposes a virtual COM port to the host. The host processor would then access this device as a regular COM port device and communicate with the ZNP using the UART transport.

2.2.2 *Interface Configuration*

The CC2530-ZNP supports UART or USB interface to the application processor.

2.2.2.1 *IAR project configuration*

The CC2530-ZNP IAR project that is included in the ZStack software package has two project configurations – CC2530-ZNP and CC2531-ZNP. As the name indicates, the configurations are intended for use with the CC2530 and CC2531 (USB) chips.

2.2.3 *UART Transport*

2.2.3.1 *Configuration*

The following UART configuration is supported:

- Baud rate: 115200
- Hardware (RTS/CTS) flow control.
- 8-N-1 byte format.

2.2.3.2 *Frame Format*

Please refer to section 2.1.3.2.

2.2.3.3 *Sample FCS Calculation*

Please refer to section 2.1.3.3 for the sample code.

2.2.3.4 *Signal Description*

The following standard UART signals are used:

- TX: Transmit data.
- RX: Receive data.
- CTS: Clear to send.
- RTS: Ready to send.

Figure 8 shows the RTS/CTS flow control connections to the host processor. On the CC2530, RT and CT are active-low signals. The RT output is driven low when the receive register is empty and reception is enabled. Transmission of a byte does not occur before the CT input goes low.

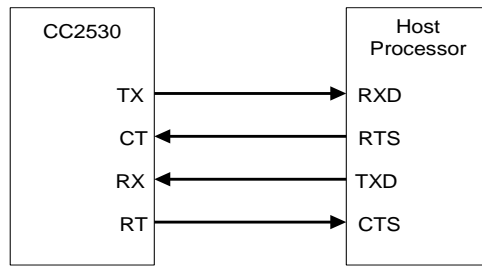


Figure 8 RTS/CTS Flow Control Connections

2.2.3.5 Signal Operation

UART transport sends and receives data asynchronously. Data can be sent and received simultaneously and the transfer of a frame can be initiated at any time by either the application processor or the CC2530.

2.2.4 General Frame Format

Please refer to section 2.1.4.

2.2.5 Initialization Procedures

2.2.5.1 CC2530-ZNP power-up procedure

The recommended power-up procedure is as follows:

1. Application processor and CC2530 power up.
2. Application processor sets CC2530 RESET_N pin low, holding CC2530 in reset.
3. The application processor sets the optional CC2530 CFG0 and CFG1 pins (if these pins are controlled by the application processor).
4. The application processor initializes its UART interface.
5. The application processor sets CC2530 RESET_N pin high and CC2530 starts operation.
6. The application processor receives the `SYS_RESET_IND` message.

The CC2530-ZNP can also be reset when the application processor sends a `SYS_RESET_REQ` message. However, resetting CC2530 with the RESET_N pin is recommended because it is faster and more reliable.

3 ZNP software command interface

The ZNP software command interface is sub-divided into the following categories

- The SYS interface (MT_SYS) provides the application processor with a low level interface to the ZNP hardware and software.
- The AF (MT_AF) and ZDO (MT_ZDO) interfaces feature the complete ZigBee interface and can be used to create a full range of ZigBee compliant applications. The AF (Application Framework) interface allows the application processor to register its application with the ZNP and send and receive data. The ZDO (ZigBee Device Object) interface provides various ZigBee management functions like device and service discovery.
- The UTIL (MT_UTIL) interface provides support functionalities such as setting PAN-ID, getting device info, getting NV info, subscribing callbacks...etc.
- The APP CONF (MT_APP_CNF) interface provides support for BDB functionality such as set Install Codes, Primary or Secondary Channel, trigger different commissioning methods and other Trust Center configurations.

For further details on the MT interface, refer to [R1]

3.1 Configuration interface

The ZNP device has numerous parameters that can be configured by the application processor. These configuration parameters are stored in non volatile memory on the ZNP device and their values persist across a device reset.

The configuration parameters are divided into “network-specific” and “device-specific” parameters. The “network-specific” configuration parameters should be set to the same value for all ZNP devices in a ZigBee network to ensure proper network operation. The “device-specific” parameters can be set to different values on each device. These parameters are listed in detail in sections 3.1.1 and 3.1.2. These configuration parameters must be written in Nv for which the host processor must use the MT interface to write Nv parameters into the ZNP device. Refer to [R1] for further details on how to write into Nv.

When the ZNP device powers up, it reads two of the configuration parameters immediately. These are the `STARTOPT_CLEAR_CONFIG` bit (part of the `ZCD_NV_STARTUP_OPTION` parameter) and the `ZCD_NV_LOGICAL_TYPE` parameters. Any modification of these parameters will require a CC2530-ZNP device reset before they can take effect.

3.1.1 Device specific configuration parameters

3.1.1.1 ZCD_NV_STARTUP_OPTION

Configuration ID: 0x0003; Size: 1 byte; Default value: 0

This parameter controls the device startup options. This is a bit mask of the following values

Bit position	7	6-2	1	0
Description	ZCD_STARTOPT_CLEAR_NWK_FRAME_COUNTER	Reserved	STARTOPT_CLEAR_STATE	STARTOPT_CLEAR_CONFIG

- ZCD_STARTOPT_CLEAR_NWK_FRAME_COUNTER – If this option is set, then the network frame counter is cleared for all networks.

Note: This should be use only for debug purposes as the network frame counters must be persistent, even after Factory New resets. The usage of this option during the operation in the networks may lead to undesaried behaviour, such as get the ZNP device ignored by other devices in the network.

- **STARTOPT_CLEAR_CONFIG** – If this option is set, the device will overwrite all the configuration parameters (except this one) with the “default” values that it is programmed with. This is used to erase the existing configuration and bring the device into a known state.

*Note: The **STARTOPT_CLEAR_CONFIG** bit is read by the ZNP device immediately when it powers up after a reset.*

*When the configuration parameters are restored to defaults, the **ZCD_NV_STARTUP_OPTION** itself is not restored except for clearing the **STARTOPT_CLEAR_CONFIG** bit.*

- **STARTOPT_CLEAR_STATE** – If this option is set, the device will clear its previous network state (which would exist if the device had been operating on a network prior to the reset). This is typically used during application development. During regular device operation, this flag is typically not set, so that an accidental device reset will not cause loss of network state.

Notes: The CC2530-ZNP device has two kinds of information stored in non-volatile memory. The configuration parameters (listed in this section) and network state information.

The configuration parameters are configured by the user before start of ZigBee operation.

The network state information is collected by the device after it joins a network and creates bindings etc. (at runtime). This is not set by the application processor. This information is stored so that if the device were to reset accidentally, it can restore itself without going through all the network joining and binding process again.

*If the application processor does not wish to continue operating in the previous ZigBee network, it needs to instruct the CC2530-ZNP device to clear the network state information and start again based on the configuration parameters. This is done by setting the **STARTOPT_CLEAR_STATE** bit in the startup option.*

3.1.1.2 ZCD_NV_LOGICAL_TYPE

Configuration ID: 0x0087; Size: 1 byte; Default value: 0x00

This is the logical type of the device in the ZigBee network. This can be set to a COORDINATOR (0x00), ROUTER (0x01) or ENDDEVICE (0x02).

Note:

This parameter is read by the ZNP device immediately when it powers up after a reset.

3.1.1.3 ZCD_NV_ZDO_DIRECT_CB

Configuration ID: 0x008F; Size: 1 byte; Default value: FALSE

This configures the manner in which ZDO responses (hereby referred to as callbacks) are issued to the host processor. By default, this item is set to FALSE, which means that the host processor must use the ZDO_MSG_CB_REGISTER command to subscribe to a specific ZDO callback in order to receive it. The ZDO callback is then conveyed as part of the ZDO_MSG_CB_INCOMING command. If ZCD_NV_ZDO_DIRECT_CB is set TRUE, then the host processor will receive the “verbose” response. For example, the host processor would receive the ZDO_IEEE_ADDR_RSP command in response to ZDO_IEEE_ADDR_REQ.

3.1.2 Network specific configuration parameters

3.1.2.1 ZCD_NV_PANID

Configuration ID: 0x0083; Size: 2 bytes; Default value: 0xFFFF

This parameter identifies the ZigBee network. This should be set to a value between 0 and 0x3FFF. Networks that exist in the same vicinity must have different values for this parameter. It can be set to a special value of 0xFFFF to indicate “don’t care”.

3.2 Z-Stack 3.0 ZNP considerations

3.2.1 Backward compatibility

ZNP is backward compatible with non Z3.0 devices by using the same API that already existed in previous releases of the stack, or by using Base Device Behavior commissioning MT interface with exception of the new security schemas for Z3.0 such as Distributed networks or Install Codes.

3.2.2 ZNP for Z3.0

While the ZNP implementation provides a compatible baseline for ZigBee 3.0 devices, a full implementation of a ZigBee 3.0 device involves additional layers on top of the ZNP. These layers shall be implemented by the user on the host-side of the stack, since they are outside the scope of the network processor. The ZNP provides several new interfaces to enable the required functionality on the host.

In order to update a legacy ZNP-based device to support ZigBee 3.0, the following main updates need to be implemented on the host:

Base Device Behavior Specification:

1. Finding and Binding: Host processor is required to implement Finding and Binding commissioning method (either as Initiator or as Target) according to the cluster supported by the host application. Touchlink (optional): proximity-based commissioning method.

Green Power Basic proxy:

1. ZigBee 3.0 coordinator and router devices must implement Green Power Basic proxy functionality. ZNP includes the necessary GP Stub interfaces which are available to the application and allow it to implement GP basic proxy functionality on the host processor.

3.2.3 ZNP startup procedure for Z3.0 implementation

After executing the power-up procedure, the host processor must call some mandatory APIs before executing any APIs that invoke ZigBee over-the-air messaging. Not following this sequence could result in unexpected behaviour. The recommended startup procedure is as follows:

1. The host processor must use the ZB_WRITE_CONFIGURATION command to configure at the minimum the ZCD_NV_LOGICAL_TYPE
2. If logical device is defined as ZC or ZR, GP basic proxy must be initialized in the host processor (No ZNP commands are required until interaction with GP devices are needed).
3. Optional configurations to commission the device are:
 - a. Set the Primary and/or Secondary channel mask to perform Formation or Network Steering.
 - b. Set the PAN ID to create or join by setting ZCD_NV_PAN_ID.
 - c. Set Install codes for networks which require it.
4. AF_REGISTER command should be sent by the host processor to register the application endpoint.
5. Host should use BDB commissioning API to create or join the network via standard network formation or joining.
6. The host processor should wait for BDB notifications on the different commissioning methods used by the host. Also host processor can rely on the supported ZDO states reported.

Example Message Exchange

The following sequence chart is provided as a simple example of a message exchange between a Host and ZNP. In this example the following (generalized) events take place:

1. The ZNP is reset.
2. The host writes some configuration data to the ZNP into Nv (ZCD_NV_LOGICAL_TYPE) which will define the logical device role.
3. An endpoint in the host is registered with the ZNP.
4. ZNP device is instructed to start a commissioning method according to the device role defined in step 2 (e.g. Commissioning Formation for ZC or ZR, or Commissioning Network Steering for ZR or ZED).
5. BDB reports the result of the process of the commissioning method execution.
6. Another device joins the network, indicated by the ZDO Device indications.
7. Data is exchanged between the Host+ZNP and joining device through AF Data Req's and AF Incoming Messages.

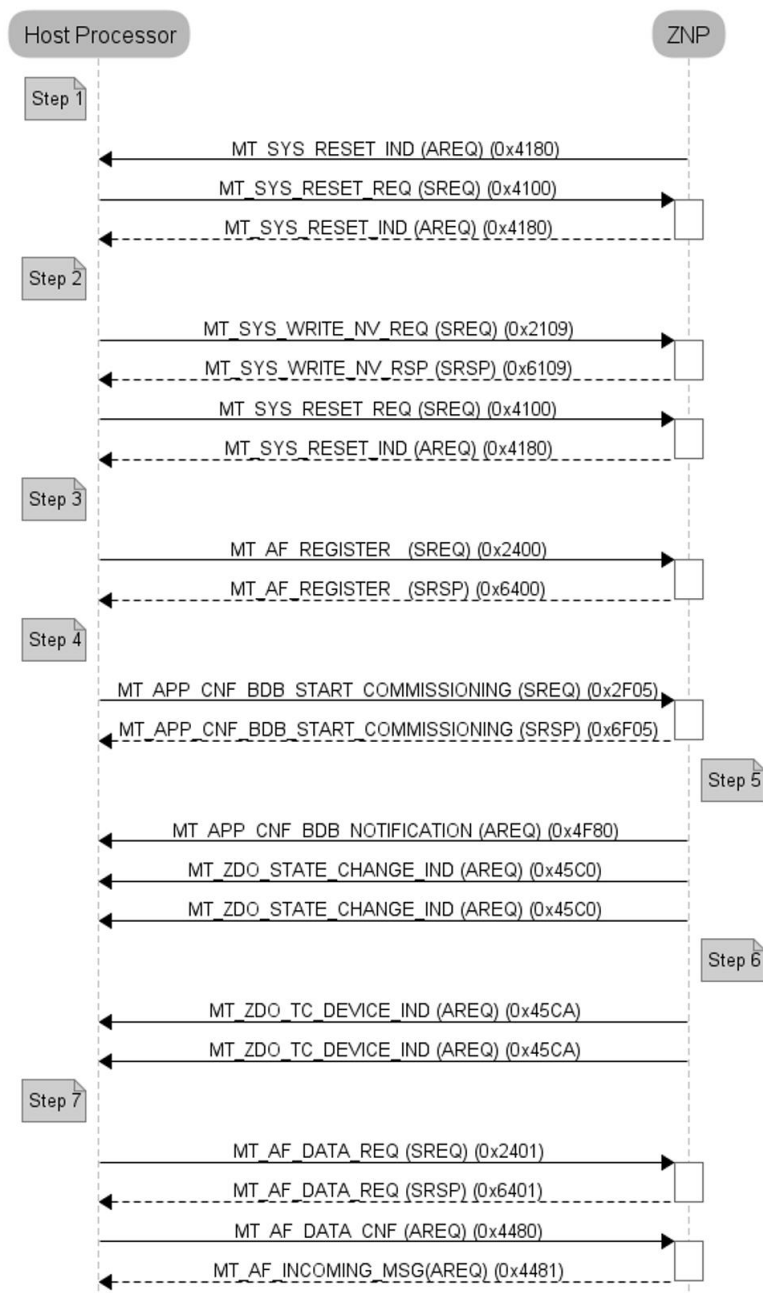


Figure 9 Example Message Sequence Chart

3.3 Return Values

The status parameter that is returned from the ZNP device may take one of the following values:

Name	Value
ZSuccess	0x00
Zfailure	0x01
ZinvalidParameter	0x02
NV_ITEM_UNINIT	0x09
NV_OPER_FAILED	0x0a
NV_BAD_ITEM_LEN	0x0c
ZmemError	0x10
ZbufferFull	0x11
ZunsupportedMode	0x12
ZmacMemError	0x13
zdoInvalidRequestType	0x80
zdoInvalidEndpoint	0x82
zdoUnsupported	0x84
zdoTimeout	0x85
zdoNoMatch	0x86
zdoTableFull	0x87
zdoNoBindEntry	0x88
ZsecNoKey	0xa1
ZsecMaxFrmCount	0xa3
ZapsFail	0xb1
ZapsTableFull	0xb2
ZapsIllegalRequest	0xb3
ZapsInvalidBinding	0xb4
ZapsUnsupportedAttrib	0xb5
ZapsNotSupported	0xb6
ZapsNoAck	0xb7
ZapsDuplicateEntry	0xb8
ZapsNoBoundDevice	0xb9
ZnwkInvalidParam	0xc1
ZnwkInvalidRequest	0xc2
ZnwkNotPermitted	0xc3
ZnwkStartupFailure	0xc4
ZnwkTableFull	0xc7
ZnwkUnknownDevice	0xc8
ZnwkUnsupportedAttribute	0xc9
ZnwkNoNetworks	0xca
ZnwkLeaveUnconfirmed	0xcb
ZnwkNoAck	0xcc
ZnwkNoRoute	0xcd
ZMacNoACK	0xe9

3.4 Additional considerations for ZNP device in Z-Stack 3.0

1. CC2530 has limitations on RAM, due to this all ZNP projects for this target are upper bounded in the number of neighbours and APS TCLK that these devices can support. Refer to the compilation definitions in the project options for `NWK_MAX_DEVICE_LIST` and `ZDSECMGR_TC_DEVICE_MAX`.
2. The current version of ZNP device does not support commissioning GP devices in the network if these devices require the basic proxy device to switch channel during this commissioning process. Other commissioning methods require that a Host process drives the commissioning process in the application level.

3.5 Additional Information

For additional details of the individual commands, please refer to the Z-Stack Monitor and Test API [R1].

4 General Information

4.1 Document History

Table 1: Document History

Revision	Date	Description/Changes
1.0	02/20/2015	Initial version
1.1	11/11/2016	Update ZNP usage and definitions to Z3.0 stack. Removed support for SPI