Serial Protocol for use with *SimpleMesh*

Colorado Micro Devices, Open Source Mesh Networking Stack

Command Code Summary

Commands in blue text show related command pairs. The command issued on the left side of the column elicits the command on the right in the following row. See the text for more complete discussion.

Command	Command ID (code)
ACK	0x00
Test Request	0x01
Test Response	0x02
Reset Request	0x03
Settings Request	0x04
Set UART Mode	0x05
Sleep Request	0x06
Wakeup Indication	0x07
Data Request	0x20
Data Confirmation	0x21
Data Indication	0x22
Set Address Request	0x23
Get Address Request	0x24
Get Address Response	0x25
Set PANID Request	0x26
Get PANID Request	0x27
Get PANID Response	0x28
Set Channel Request	0x29
Get Channel Request	0x2A
Get Channel Response	0x2B
Set Receiver State Request	0x2C
Get Receiver State Request	0x2D
Get Receiver State Response	0x2E
Set Transmit Power Request	0x2F
Get Transmit Power Request	0x30
Get Transmit Power Response	0x31
Set Security Key Request	0x32
Set ACK State Request	0x35
Get ACK State Request	0x36
Set ACK State Response	0x37
Set LED State Request	0x80

Table 1 - Command Summary

IEEEE802.15.4 Notes

The SimpleMesh network uses *intra-pan* addressing. That means that the MAC header contains only one PANID. SimpleMesh also uses short (16 bit) addressing for all addresses. Therefore the MAC header is always 9 bytes long.

General command format:

Start Byte	Size	Payload	CRC
1	1	Variable	2

- Start Byte – constant 0xAB
- Size – size of the *command id + options + payload* field
- Payload – command payload
- CRC – 16 bit CRC calculated over payload with initial value 0x1234 (i.e. it is calculated over the *command id, options and payload* if any.)

Every command is confirmed with an acknowledgment command even if it is impossible to immediately execute the command. There is no particular order in which responses are sent, so for example Data Indication Command might be sent before Acknowledgment Command.

Acknowledgment Command format:

Command Id	Status
1	1

- Command Id – constant 0x00
- Status – command execution status, one of:

Value	Description
0x00	Success
0x01	Unknown error
0x02	Out Of Memory
0x11	No Acknowledgment Was Received
0x40	Channel Access Failure
0x41	No Physical Acknowledgment Was Received
0x80	Invalid Command Size
0x81	Invalid CRC
0x82	Timeout
0x83	Unknown Command
0x84	Malformed Command
0x85	Internal Flash Error
0x86	Invalid Data Request payload size

Commands:

Test Request

This command is used to check the communication channel and performs no other actions. A Test Response command is sent as the result of execution of this command.

Command format:

Command Id
1

- Command Id – constant 0x01

Test Response

This command is a response to the Test Request command.

Command format:

Command Id
1

- Command Id – constant 0x02

This command is used to reset the module.

Command format:

Command Id
1

- Command Id – constant 0x03

This command is used to store the current settings to the persistent memory or restore the current settings to a default value.

Command format:

Command Id	Operation
1	1

- Command Id – constant 0x04
- Operation – operation to be performed, one of:

Value	Description
0x10	Save current settings
0x15	Restore default settings

This command is used to change the UART settings. New settings are applied after an Acknowledgment Command is sent.

Command format:

Command Id	Data Bits	Parity	Stop Bits	Baudrate
1	1	1	1	1

- Command Id – constant 0x05
- Data Bits – UART data bits, one of:

Value	Description
0x00	5
0x01	6
0x02	7
0x03	8

- Parity – UART parity, one of:

Value	Description
0x00	None
0x01	Odd
0x02	Even
0x03	Force 1
0x04	Force 0

- Stop Bits – UART stop bits, one of:

Value	Description
0x00	1
0x01	2 (1.5 for 5 data bits)

- Baudrate – UART baudrate, one of:

Value	Description

0x00	Reserved (Auto)
0x01	50
0x02	75
0x03	110
0x04	150
0x05	300
0x06	1200
0x07	2400
0x08	4800
0x09	9600
0x0a	19200
0x0b	38400
0x0c	57600
0x0d	115200
0x0e	230400
0x0f	460800
0x10	2000
0x11	4000
0x12	8000
0x13	10000
0x14	20000
0x15	30000
0x16	40000
0x17	50000
0x18	60000
0x19	70000
0x1a	80000
0x1b	90000
0x1c	100000
0x1d	200000
0x1e	300000
0x1f	400000

This command is used to sleep the module. The Wakeup Indication command is sent after the sleep interval has passed.

Command format:

Command Id	Interval
1	4

- Command Id – constant 0x06
- Interval – sleep interval (ms)

This command is sent in response to the Sleep Request command after the sleep time is over.

Command format:

Command Id
1

- Command Id – constant 0x07

This command is used to send data over the network.

Command format:

Command Id	Destination	Options	Handle
1	2	1	1

- Command Id – constant 0x20
- Destination – destination address
- Options – request options, any combination of:

Value	Description
0x00	None
0x01	Request an Acknowledgment
0x02	Enable Security

- Handle – request handle; it is used to identify this request with the following Data Confirmation command

This command is sent in response to the Data Request command.

Command format:

Command Id	Status	Handle
1	1	1

- Command Id – constant 0x21
- Status – data request status, one of:

Value	Description
0x00	Success
0x01	Unknown error
0x02	Out Of Memory
0x11	No Acknowledgment Was Received
0x40	Channel Access Failure
0x41	No Physical Acknowledgment Was Received

- Handle – request handle; it contains value used in corresponding Data Request command

This command is sent when a data frame is received.

Command format:

Command Id	Source Address	Options	LQI	RSSI	Payload
1	2	1	1	1	Variable

- Command Id – constant 0x22
- Source Address – address of the sending device
- Options – frame options, any combination of:

Value	Description
0x00	None
0x01	Acknowledgment was requested
0x02	Security was used

- LQI – Link Quality Indicator
- RSSI – Received Signal Strength Indicator
- Payload – received data

Set Address Request

This command is used to set the address of the device.

Command format:

Command Id	Address
1	2

- Command Id – constant 0x23
- Address – new device address

Get Address Request

This command is used to get the address of the device.

Command format:

Command Id
1

- Command Id – constant 0x24

Get Address Response

This command is sent in response to the Get Address Request command.

Command format:

Command Id	Address
1	2

- Command Id – constant 0x25
- Address – device address

This command is used to set the PAN Id of the device.

Command format:

Command Id	PAN Id
1	2

- Command Id – constant 0x26
- PAN Id – new device PAN Id

This command is used to get the PAN Id of the device.

Command format:

Command Id
1

- Command Id – constant 0x27

Get PAN Id Response

This command is sent in response to the Get PAN Id Request command.

Command format:

Command Id	PAN Id
1	2

- Command Id – constant 0x28
- PAN Id – device PAN Id

This command is used to set the channel of the device.

Command format:

Command Id	Channel
1	1

- Command Id – constant 0x29
- Channel – new channel (valid range 11-25)

Get Channel Request

This command is used to get the channel of the device.

Command format:

Command Id
1

- Command Id – constant 0x2a

Get Channel Response

This command is sent in response to the Get Channel Request command.

Command format:

Command Id	Channel
1	1

- Command Id – constant 0x2b
- Channel – device channel

This command is used to set the receiver state of the device.

Command format:

Command Id	Receiver State
1	1

- Command Id – constant 0x2c
- Receiver State – new receiver state (0 – Off, 1 – On)

Get Receiver State Request

This command is used to get the receiver state of the device.

Command format:

Command Id
1

- Command Id – constant 0x2d

Get Receiver State Response

This command is sent in response to the Get Receiver State Request command.

Command format:

Command Id	Receiver State
1	1

- Command Id – constant 0x2e
- Receiver State – state of the receiver

This command is used to set the transmit power of the device.

Command format:

Command Id	Transmit Power
1	1

- Command Id – constant 0x2f
- Transmit Power – new transmit power, one of:

Value	Description
0x00	+3.0 dBm
0x01	+2.8 dBm
0x02	+2.3 dBm
0x03	+1.8 dBm
0x04	+1.3 dBm
0x05	+0.7 dBm
0x06	0 dBm
0x07	-1.0 dBm
0x08	-2.0 dBm
0x09	-3.0 dBm
0x0a	-4.0 dBm
0x0b	-5.0 dBm
0x0c	-7.0 dBm
0x0d	-9.0 dBm
0x0e	-12.0 dBm
0x0f	-17.0 dBm

Get Transmit Power Request

This command is used to get the transmit power of the device.

Command format:

Command Id
1

- Command Id – constant 0x30

Get Transmit Power Response

This command is sent in response to the Get Transmit Power Request command.

Command format:

Command Id	Transmit Power
1	1

- Command Id – constant 0x31
- Transmit Power – transmitter output power

This command is used to set the security key.

Command format:

Command Id	Security Key
1	16

- Command Id – constant 0x32
- Security Key – new security key

This command is used to enable or disable acknowledgments for incoming frames.

Command format:

Command Id	Acknowledgment State
1	1

- Command Id – constant 0x35
- Acknowledgment State – 0 – Disable, 1 – Enable.

Get Acknowledgment State Request

This command is used to get the acknowledgment state of the device.

Command format:

Command Id
1

- Command Id – constant 0x36

Set Acknowledgment State Response

This command is sent in response to the Get Acknowledgment State Request command.

Command format:

Command Id	Acknowledgment State
1	1

- Command Id – constant 0x37
- Acknowledgment State – acknowledgment state

This command is used to set the LED state.

Command format:

Command Id	State
1	1

- Command Id – constant 0x80
- State – LED state (0 – Off, 1 – On, 2 – Toggle)

Example of Sending Data

Here's an example of how to "package" *data* to be sent.

Given the general frame format:

Start Byte	Size	Payload	CRC
1	1	Variable	2

And a data request:

Command Id	Destination	Options	Handle
1	2	1	1

The data request is encapsulated inside the payload of the general frame format:

Start Byte	Size	Command Id	Destination	Options	Handle	Payload	CRC
1	1	1	2	1	1	Variable	2

CRC Calculation Details & Example

The CRC used is a 16-bit CRC with an initial value of 0x1234. It is calculated over the command id, options and payload. Note some of those fields may be non-existent in certain message types.

The following example C code shows an implementation of the CRC algorithm. To generate a CRC for adding to an outgoing message, first call `crc_start()`. Then call `crc_addByte(data)` with each byte of the command id, options, and payload. Finally call `crc_done()` to get the final CRC.

```c
static uint16_t crc;

void crc_start(void)
{
        crc = 0x1234;
}

void crc_addByte(uint8_t data)
{
        data ^= crc & 0xff;
        data ^= data << 4;
        crc = (((uint16_t)data << 8) | ((crc >> 8) & 0xff)) ^ (uint8_t)(data >> 4) ^
                ((uint16_t)data << 3);
}

uint16_t crc_done(void)
{
        return crc;
}
```

The following is an example message with CRC showing the proper CRC calculation & byte order. The resulting CRC would be 0xE251 in this example.

Start Byte	Size	Command Id	Status	CRC LSB	CRC MSB
0xAB	0x02	0x00	0x00	0x51	0xE2