RemoTI
Network Processor
Interface Specification

Document Number: SWRA271F

Table of Contents

List of Figures

List of Tables

Acronyms and Definitions

EM	Evaluation module
IO	Input Output
I2C	Inter-Integrated Circuit – a 2-wire synchronous communications bus.
MRDY	Master Ready
MOSI	Master Output, Slave Input
MISO	Master Input, Slave Output
NP	Network Processor
NPI	Network Processor Interface
NSDU	Network Service Data Unit
RPC	Remote Procedure Call
RTI	RemoTI application framework layer
RTIS	RTI Surrogate
SCLK	SPI clock
SOC	System on Chip
SOF	Start Of Frame byte
SPI	Serial Peripheral Interface
SRDY	Slave Ready
SS	Slave Select
UART	Universal Asynchronous Receiver-Transmitter

References

[1] RemoTI API, SWRA268, Texas Instruments Inc.

[2] “HAL Drivers, Application Programming Interface”, SWRA193, Texas Instruments Inc

1. Introduction

RemoTI development kit includes reference support for either UART, SPI or I2C[1] interface for a host processor to control RemoTI network processor via remote procedure calls through the selected interface medium. This document describes specification of such interface.

2. Architecture

The diagram below shows how a host processor interfaces with RemoTI network processor.

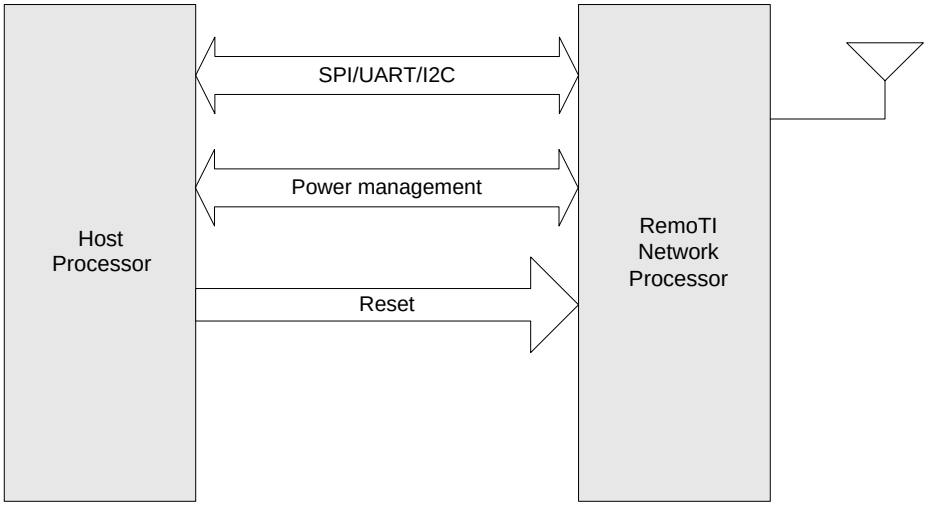

Figure 1: RNP Architecture

The SPI, UART or I2C physical interface is used to communicate between the two processors. The other interfaces are described below.

- Power Management: This interface consists of two signals (SRDY and MRDY) and is used to communicate the power management status and to wake up sleeping devices. This interface is only required if SPI or I2C transport is used.
- Reset: The host processor can reset RemoTI network processor through the RESET_N pin. In addition, a software reset interface is provided.

[1] I2C HW module is available only for CC2533 devices. Throughout this document we refer to I2C interface with specific scope to 2533 devices only.

3. Default pin configuration

Default pin configuration to support RemoTI network processor interface is found in

Table 1: Default Pin Configurations

PIN NAME	CC253x PIN	SPI configuration	I2C configuration	UART configuration
P0_2	17	-		UART RX
P0_3[2]	16	MRDY	MRDY	UART TX
P0_4[3]	15	SRDY	SRDY	-
P1_4	6	SS		-
P1_5	5	SCLK		-
P1_6	38	MOSI		-
P1_7	37	MISO		-
SCL	2	-	I2C clock	-
SDA	3	-	I2C data	-
RESET_N	20	Reset RNP	Reset RNP	Reset RNP

4. Physical Interface

The RemoTI network processor for the CC253x family supports either SPI or UART transport interface to the host processor; the CC2533 supports I2C as well.

4.1 SPI Transport

4.1.1 Configuration

The following SPI configuration is supported by RemoTI network processor:

- SPI slave.
- Clock speed up to 4 MHz on CC253x.
- Clock polarity 0 and clock phase 0 on CC253x.
- Bit order MSB first on CC253x.

4.1.2 Frame Format

SPI transport uses the general frame format described in 4.3.

Note that there are four frame types frequently quoted in the subsequent sections. They are as follows:

- POLL command frame: A POLL command is used to retrieve queued data. This command is only applicable to SPI transport. For a POLL command the subsystem and ID are set to zero and data length is zero.
- SREQ command frame: A synchronous request that requires an immediate response. For example, a function call with a return value would use an SREQ command.
- AREQ command frame: An asynchronous request. For example, a callback event or a function call with no return value would use an AREQ command.
- SRSP command frame: 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.

4.1.3 Signal Description

The following standard SPI signals are used:

[2] SPI and I2C drivers also support MRDY signal mapped on IO 1.3, pin 7.
[3] SPI and I2C drivers also support SRDY signal mapped on IO 1.2, pin 8.

- SCLK: Serial clock.
- SS: Slave select.
- MOSI: Master-output slave-input data.
- MISO: Master-input slave-output data.

The following additional signals are required for SPI transaction handling and power management:

- MRDY: Master ready. This signal is set by the host processor when it has data ready to send to the RemoTI network processor. It is active low. This signal can either be controlled independently or it can be hardwired to the slave select signal. The scenarios in this document assume MRDY is hardwired to SS.
- SRDY: Slave ready. This signal is set by the RemoTI network processor when it is ready to receive or send data. When set low, it indicates that the RemoTI network processor is ready to receive data. When set high during an SPI POLL command or SREQ command transaction it indicates that the RemoTI network processor is ready to send data. When set high during an SPI AREQ command transaction it indicates that the RemoTI network processor is done receiving data.

4.1.4 Signal Operation

The signals operate according to the following rules:

1. The host processor initiates a transaction by setting MRDY low and then waits for SRDY to go low.
2. The host processor shall never set MRDY high to end a transaction before all bytes of the frame have been transferred.
3. When receiving a POLL command or an SREQ command, the RemoTI network processor shall set SRDY high when it has data ready for the host processor.
4. When receiving an AREQ command, the RemoTI network processor shall set SRDY high when all bytes of the frame have been received.

4.1.5 Protocol Scenarios

4.1.5.1 Synchronization of Host and Network Processors

Before SPI communication can be performed between the host and network processors, a synchronization sequence must occur. The following figure shows this process.

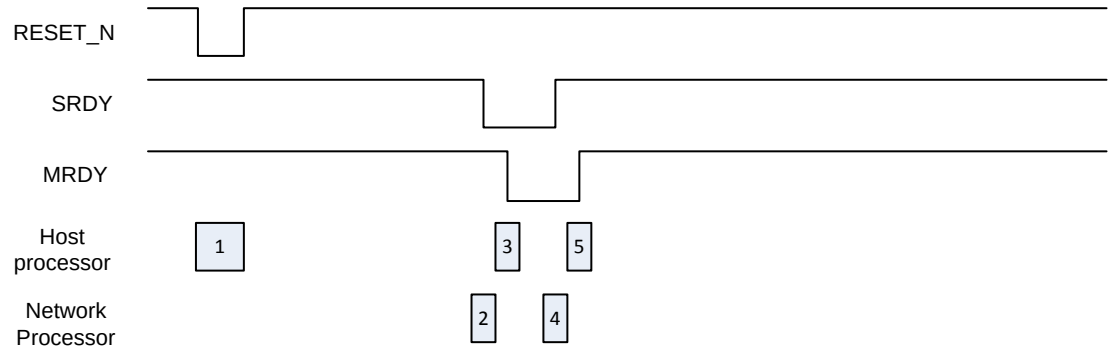

Figure 2: Host and Network Processor Synchronization

The following sequence of events occurs on the host processor and RemoTI network processor:

1. Host processor issues a reset to the network processor.
2. Network processor completes its initialization and indicates it is ready for SPI synchronization by setting SRDY low. Note that if the host processor polls for SRDY,

it is valid approximately 5ms after RESET_N is released, and will be set low up to 500ms after RESET_N is released.

3. Host processor reads SRDY low and confirms by setting MRDY low.
4. Network processor reads MRDY low and sets SRDY high to confirm.
5. Host processor reads SRDY high and sets MRDY high to complete synchronization.

4.1.5.2 AREQ Command

The following figure shows an AREQ command sent from the host processor to the RemoTI network processor.

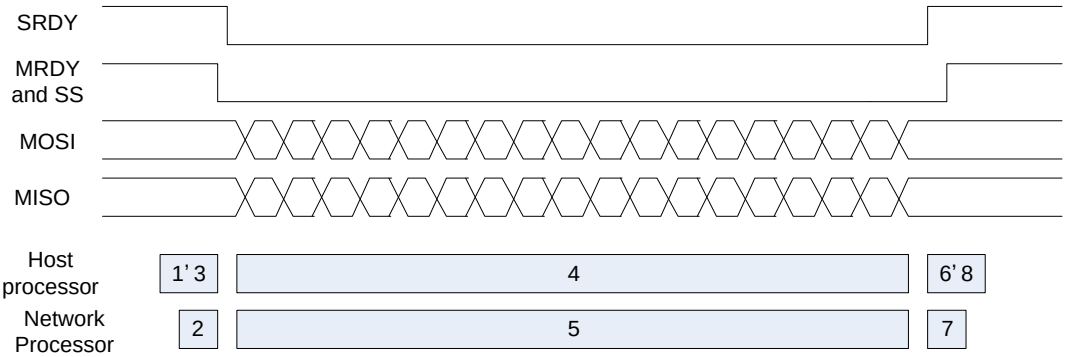

Figure 3: SPI AREQ Command Signaling

The following sequence of events occurs on the host processor and RemoTI network processor:

6. Host processor has an AREQ frame to send. Set MRDY low and wait for SRDY to go low.
7. RemoTI network processor receives falling edge of MRDY. When ready to receive data set SRDY low.
8. Host processor reads SRDY low. Start data transmission.
9. Host processor transmits data until frame is complete.
10. Network processor receives data until frame is complete.
11. Host processor waits for SRDY to go high.
12. Network processor receives complete frame and sets SRDY high.
13. Host processor reads SRDY high. Set MRDY high.

4.1.5.3 POLL Command

The following figure shows a POLL command sent from the host processor to the RemoTI network processor.

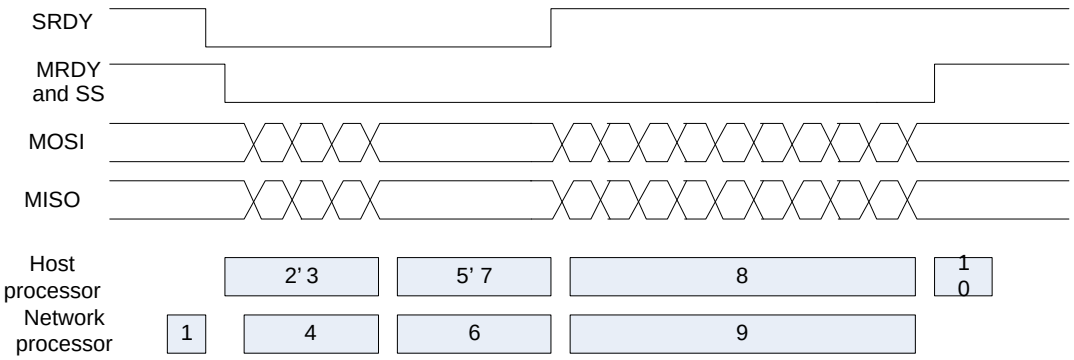

Figure 4: SPI POLL Command Signaling

The following sequence of events occurs on the host processor and RemoTI network processor:

1. Network processor has an AREQ frame to send. When ready to receive POLL command, set SRDY low.
2. Host processor detects SRDY low and sets MRDY low. Prepare POLL command and start data transmission.
3. Host processor transmits data until frame is complete.
4. Network processor receives data until frame is complete.
5. Host processor waits for SRDY to go high.
6. Network processor prepares AREQ frame for transmission. When ready to transmit set SRDY high.
7. Host processor reads SRDY high. Start data reception.
8. Host processor receives data until frame is complete.
9. Network processor transmits data until frame is complete.
10. Host processor receives complete frame. Set MRDY high.

4.1.5.4 SREQ Command/SRSP Response

The following figure shows a SREQ command sent from the host processor to the RemoTI network processor.

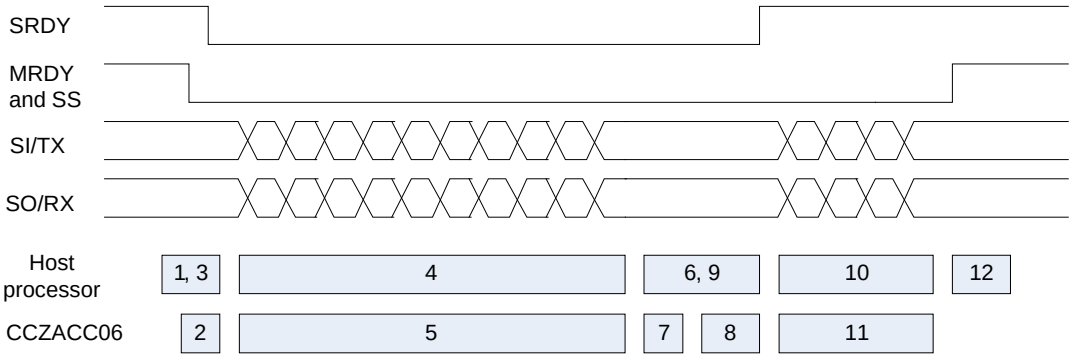

Figure 5: SPI SREQ Command Signaling

The following sequence of events occurs on the host processor and RemoTI network processor:

1. Host processor has an SREQ frame to send. Set MRDY low and wait for SRDY to go low.
2. Network processor receives falling edge of MRDY. When ready to receive data set SRDY low.
3. Host processor reads SRDY low. Start data transmission.
4. Host processor transmits data until frame is complete.
5. Network processor receives data until frame is complete.
6. Host processor waits for SRDY to go high.
7. Network processor processes SREQ command and executes function
8. Network processor prepares SRSP frame. When ready to transmit data set SRDY high.
9. Host processor reads SRDY high. Start data reception.
10. Host processor receives data until frame is complete.
11. Network processor transmits data until frame is complete.
12. Host processor receives complete frame. Set MRDY high.

4.2 UART Transport

4.2.1 Configuration

The following UART configuration is supported:

- Baud rate: 115200.
- 8N1 byte format.

4.2.2 Frame Format

UART transport frame format is shown in the following figure. The left-most field is transmitted first over the wire.

Bytes: 1	3-126	1
SOF	General format frame	FCS

Figure 6: UART 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 4.3.

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

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;
}
```

4.2.3 Signal Description

The following standard UART signals are used:

- TX: Transmit data.
- RX: Receive data.
- The additional MRDY and SRDY signals are not used with UART transport. Instead, RemoTI network processor will wake up on UART RX data, and power operation will be controlled by software interface.

4.2.4 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 host processor or the RemoTI network processor.

4.2.5 Network Processor UART Sleep and Wakeup Operation

Network processor application is responsible to put UART receiver of the network processor to sleep. For example, network processor application could put UART receiver into sleep when an application command such as RTI_ENABLE_SLEEP_REQ described in section 5.1.22 is received.

Once network processor UART receiver is asleep, host processor shall wake up network processor UART receiver by sending a null character (value 0x00). Once network processor successfully wake up its UART receiver, the network processor shall send a null character (value 0x00) back to the host processor.

Such handshaking for waking up network processor UART RX has to happen before host processor sending any valid frame to network processor.

This section does not apply to wakeup and sleep of host processor UART receiver. The network processor interface specification assumes that host processor can autonomously sleep and it could wakeup on an UART frame sent by network processor and correctly decode the frame.

4.3 I2C Transport (2533 devices only)

4.3.1 Configuration

The following I2C configuration is supported by RemoTI network processor:

- I2C slave[4], compliance with v2.1 of I2C specification.
- Standard mode up to 100 Khz and fast-mode up to 400 Khz
- 7 bit slave device address SW-configurable (HAL drivers set default slave address 0x41). See [2] to understand how the slave address can be configured.
- Support of REPEATED start transactions
- Clock stretching

4.3.2 Frame Format

I2C physical transport uses the general serial logical frame format as described in 4.1.2, valid for the SPI physical transport.

4.3.3 Signal Description

The following standard SPI signals are used:

- SCL: I2C clock signal, generated by the master, up to 400 Khz (fast-mode).
- SDA: I2C data signal, bi-directional.

As per I2C protocol, while the master is responsible to generate START/STOP/RESTART and DIR bit to open, close and manage the direction of the I2C transaction, the ACK bit (active low for positive acknowledge) is set by the receiver and hence depends on the data direction flow. As usual for I2C protocol, the address and data bytes are sent MSB transmitted first.

The following additional signals are required for I2C transaction handling and power management:

- MRDY: Master ready. This signal is set by the host processor when it has data ready to send to the RemoTI network processor. It is active low.
- SRDY: Slave ready. This signal is set by the RemoTI network processor when it is ready to receive or send data. When set low, it indicates that the RemoTI network processor is ready to receive data. When set high during a POLL command or SREQ command transaction it indicates that the RemoTI network processor is ready to send data. When set high during an SPI AREQ command transaction it indicates that the RemoTI network processor is done receiving data.

4.3.4 Signal Operation

The signals operate according to the following rules:

1. The host processor initiates a transaction by setting MRDY low and then waits for SRDY to go low (HW handshake acknowledge).
2. The host processor shall never set MRDY high to end a transaction before all bytes of the frame have been transferred.
3. When receiving a POLL command or an SREQ command, the RemoTI network processor shall set SRDY high when it has data ready for the host processor.
4. When receiving an AREQ command, the RemoTI network processor shall set SRDY high when all bytes of the frame have been received.

[4] I2C module in CC2533 can also operate as a master for non-RNP configuration

4.3.5 Protocol Scenarios

4.3.5.1 AREQ Command

The following figure shows an AREQ command sent from the host processor to the RemoTI network processor.

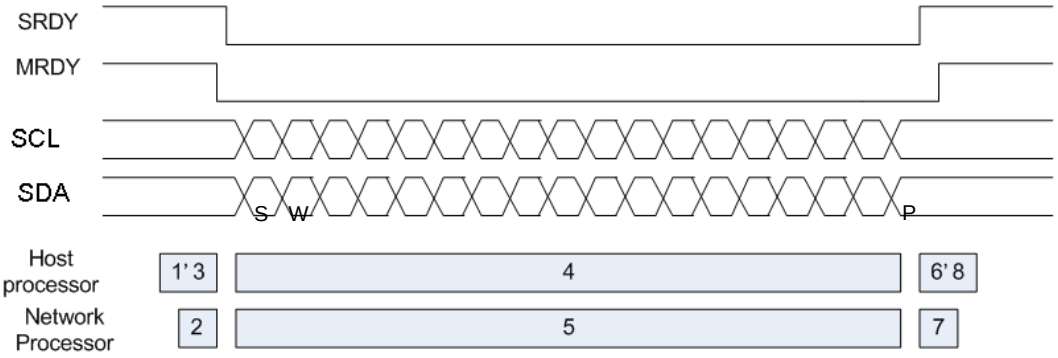

Figure 7: I2C AREQ Command Signaling

The following sequence of events occurs on the host processor and RemoTI network processor:

1. Host processor has an AREQ frame to send. Set MRDY low and wait for SRDY to go low.
2. RemoTI network processor receives falling edge of MRDY. When ready to receive data set SRDY low.
3. Host processor reads SRDY low. Start data transmission on the I2C bus through the start bit (S) and address the slave address through a write (W).
4. Host processor transmits data until frame is complete and closes the I2C transaction sending the stop bit (P).
5. Network processor receives data until frame is complete.
6. Host processor waits for SRDY to go high.
7. Network processor receives complete frame and sets SRDY high.
8. Host processor reads SRDY high. Set MRDY high.

4.3.5.2 POLL Command

The following figure shows a POLL command sent from the host processor to the RemoTI network processor.

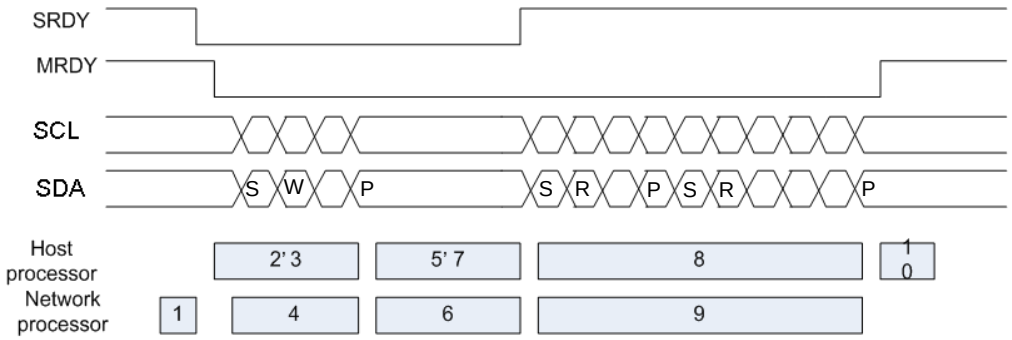

Figure 8: I2C POLL Command Signaling

The following sequence of events occurs on the host processor and RemoTI network processor:

1. Network processor has an AREQ frame to send. When ready to receive POLL command, set SRDY low.
2. Host processor detects SRDY low and sets MRDY low. Prepare POLL command and start data transmission. I2C transaction is initiated with Start bit (S) set and addressing the slave in Write (W) mode.
3. Host processor transmits data until frame is complete. When data is completed host terminates the I2C transaction by sending a stop bit (P).
4. Network processor receives data until frame is complete.
5. Host processor waits for SRDY to go high.
6. Network processor prepares AREQ frame for transmission. When ready to transmit set SRDY high.
7. Host processor reads SRDY high. Start data reception by initiating the I2C bus by setting the I2C bus through a Start bit (S), addressing the slave in Read mode (R). First it performs a 3 bytes read (receives the frame header) then, knowing the size of the message which is being received, it performs a sub-sequent read and then issue a stop bit (P) to close I2C transaction
8. Host processor receives data until frame is complete. Host processor, knowing the size of the incoming data, knows when a stop
9. Network processor transmits data until frame is complete.
10. Host processor receives complete frame. Set MRDY high.

4.3.5.3 SREQ Command

The following figure shows a SREQ command sent from the host processor to the RemoTI network processor.

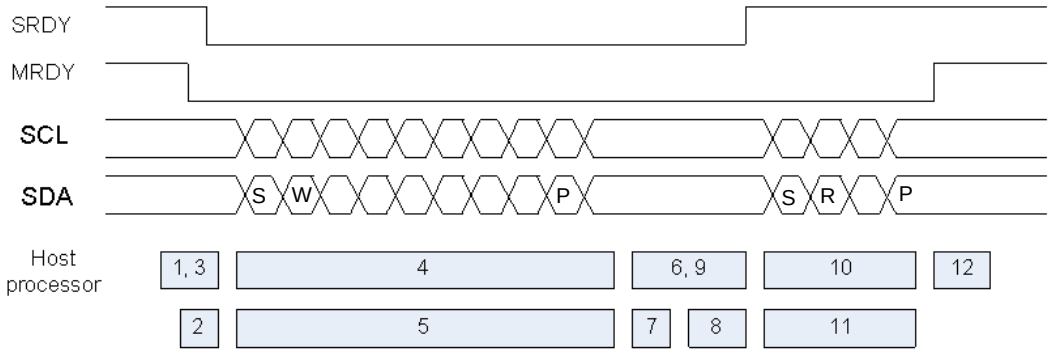

Figure 9: I2C SREQ Command Signaling

The following sequence of events occurs on the host processor and RemoTI network processor:

1. Host processor has an SREQ frame to send. Set MRDY low and wait for SRDY to go low.
2. Network processor receives falling edge of MRDY. When ready to receive data set SRDY low.
3. Host processor reads SRDY low. Start data transmission by sending the synchronous request over the I2C bus, setting the start bit (S) and addressing the RNP slave through its address and a write command (W).
4. Host processor transmits data until frame is complete. At the end of data sending, it sends the stop bit (P) to close the I2C transaction
5. Network processor receives data until frame is complete.
6. Host processor waits for SRDY to go high.
7. Network processor processes SREQ command and executes function
8. Network processor prepares SRSP frame. When ready to transmit data set SRDY high.
9. Host processor reads SRDY high. Start data reception on the I2C bus by setting the start bit (S) and addresses the RNP slave in read mode (R) through its slave device address.
10. Host processor receives data until frame is complete.
11. Network processor transmits data until frame is complete.
12. Host processor receives complete frame. Knowing the size of the response to the synchronous request, closes the I2C transaction by sending the stop bit (P). It then sets MRDY high.

4.4 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.

Bytes: 1	2	0-123
Length	Command	Data

Figure 10: I2C and SPI General Frame Format

Length: The length of the data field of the frame. The length can range from 0-123.

Command: The command of the frame.

Data: The frame data. This depends on the command field and is described for each command in Section 5.

4.4.1 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 in a frame.

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

Figure 11: Command Field of General Frame Format

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

- 0: POLL. A POLL command is used to retrieve queued data. This command is only applicable to SPI transport. For a POLL command the subsystem and ID are set to zero and data length is zero.
- 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 in Table 2.

Table 2: RPC Subsystem

Subsystem Value	Subsystem Name
0	Reserved
1	SYS interface
2-9	Reserved
10	RemoTI Application Framework interface
11	Reserved (RemoTI network layer interface)
12	Reserved (RemoTI network layer client interface)
13-32	Reserved

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

5. RemoTI Network Processor Application Interface

The following subsections describe the RemoTI network processor application command interface.

In all the message formats shown below, the left-most field is transmitted first over the wire. For multi-byte fields, the lowest order byte is transmitted first.

5.1 RemoTI Application Framework interface

5.1.1 RTI_READ_ITEM – Deprecated.

5.1.1.1 Description

This command has been deprecated in favor of the extended API defined in 5.1.29.

This command is issued by the application processor to remotely call RTI_ReadItem() function of network processor (See [1]).

5.1.1.2 Usage

Table 3: RTI_ReadItem() SREQ

1	1	1	1	*1*
Length = 0x02	*Cmd0 = 0x2A*	*Cmd1 = 0x01*	*itemId*	*len*

Table 4: RTI_ReadItem() SRSP

1	1	1	1	***variable***
Length = 1 + value length	*Cmd0 = 0x6A*	*Cmd1 = 0x01*	*status**	*value***

*status corresponds to return value of `RTI_ReadItem()` call

**value corresponds to data payload to be returned as pointed by `pValue` argument of `RTI_ReadItem()` function.

5.1.2 RTI_WRITE_ITEM – Deprecated.

5.1.2.1 Description

This command has been deprecated in favor of the extended API defined in 5.1.30.

This command is issued by the application processor to remotely call `RTI_WriteItem()` function of network processor (See [1]).

5.1.2.2 Usage

Table 5: RTI_WriteItem() SREQ

1	1	1	1	1	***variable***
Length = 0x02 + value length	*Cmd0 = 0x2A*	*Cmd1 = 0x02*	*itemId*	*len*	*value**

Table 6: RTI_WriteItem() SRSP

1	1	1	*1*
Length = 0x01	*Cmd0 = 0x6A*	*Cmd1 = 0x02*	*status***

*value corresponds to data payload as pointed by `pValue` argument of `RTI_WriteItem()` function.

**status corresponds to return value of `RTI_WriteItem()` function.

5.1.3 RTI_INIT_REQ

5.1.3.1 Description

This command is issued by the host processor to remotely call `RTI_InitReq()` function of network processor (See [1]).

5.1.3.2 Usage

Table 7: RTI_InitReq() AREQ

1	1	*1*
Length = 0x00	*Cmd0 = 0x4A*	*Cmd1 = 0x03*

5.1.4 RTI_INIT_CNF

5.1.4.1 Description

This command is issued by the network processor to notify occurrence of `RTI_InitCnf()` callback in response to a prior `RTI_INIT_REQ` command (See [1]).

5.1.4.2 Usage

Table 8: RTI_InitCnf() AREQ

1	1	1	*1*
Length = 0x01	*Cmd0 = 0x4A*	*Cmd1 = 0x01*	*status*

5.1.5 RTI_PAIR_REQ

5.1.5.1 Description

This command is issued by the host processor to remotely call `RTI_PairReq()` function of network processor (See [1]).

5.1.5.2 Usage

Table 9: RTI_PairReq() AREQ

1	1	*1*
Length = 0x00	*Cmd0 = 0x4A*	*Cmd1 = 0x04*

5.1.6 RTI_PAIR_CNF

5.1.6.1 Description

This command is issued by the network processor to notify occurrence of `RTI_PairCnf()` callback in response to a prior `RTI_PAIR_REQ` command (See [1]).

5.1.6.2 Usage

Table 10: RTI_PairCnf() AREQ

1	1	1	1	1	*1*
Length = 0x03	*Cmd0 = 0x4A*	*Cmd1 = 0x02*	*status*	*dstIndex*	*devType*

5.1.7 RTI_PAIR_ABORT_REQ

5.1.7.1 Description

This command is issued by the host processor to remotely call `RTI_PairAbortReq()` function of network processor (See [1]).

5.1.7.2 Usage

Table 11: RTI_PairAbortReq() AREQ

1	1	*1*
Length = 0x00	*Cmd0 = 0x4A*	*Cmd1 = 0x0C*

5.1.8 RTI_PAIR_ABORT_CNF

5.1.8.1 Description

This command is issued by the network processor to notify occurrence of `RTI_PairAbortCnf()` callback in response to a prior `RTI_PAIR_ABORT_REQ` command (See [1]).

5.1.8.2 Usage

Table 12: RTI_PairAbortCnf() AREQ

1	1	1	*1*
Length = 0x01	*Cmd0 = 0x4A*	*Cmd1 = 0x0C*	*status*

5.1.9 RTI_ALLOW_PAIR_REQ

5.1.9.1 Description

This command is issued by the host processor to remotely call `RTI_AllowPairReq()` function of network processor (See [1]).

5.1.9.2 Usage

Table 13: RTI_AllowPairReq() AREQ

1	1	*1*
Length = 0x00	*Cmd0 = 0x4A*	*Cmd1 = 0x06*

5.1.10 RTI_ALLOW_PAIR_CNF

5.1.10.1 Description

This command is issued by the network processor to notify occurrence of `RTI_AllowPairCnf()` callback in response to a prior `RTI_ALLOW_PAIR_REQ` command (See [1]).

5.1.10.2 Usage

Table 14: RTI_AllowPairCnf() AREQ

1	1	1	1	1	*1*
Length = 0x03	*Cmd0 = 0x4A*	*Cmd1 = 0x04*	*status*	*dstIndex*	*devType*

5.1.11 RTI_ALLOW_PAIR_ABORT_REQ

5.1.11.1 Description

This command is issued by the host processor to remotely call `RTI_AllowPairAbortReq()` function of network processor (See [1]).

5.1.11.2 Usage

Table 15: RTI_AllowPairAbortReq() AREQ

1	1	*1*
Length = 0x00	*Cmd0 = 0x4A*	*Cmd1 = 0x0D*

5.1.12 RTI_UNPAIR_REQ

5.1.12.1 Description

This command is issued by the host processor to remotely call `RTI_UnpairReq()` function of network processor (See [1]).

5.1.12.2 Usage

Table 16: RTI_UnpairReq() AREQ

1	1	1	*1*

Length = 0x01	*Cmd0 = 0x4A*	*Cmd1 = 0x0B*	*dstIndex*

5.1.13 RTI_UNPAIR_CNF

5.1.13.1 Description

This command is issued by the network processor to notify occurrence of `RTI_UnpairCnf()` callback in response to a prior RTI_UNPAIR_REQ command (See [1]).

5.1.13.2 Usage

Table 17: RTI_UnpairCnf() AREQ

1	1	1	1	*1*
Length = 0x02	*Cmd0 = 0x4A*	*Cmd1 = 0x0A*	*Status*	*dstIndex*

5.1.14 RTI_UNPAIR_IND

5.1.14.1 Description

This command is issued by the network processor to notify occurrence of `RTI_UnpairInd()` callback upon receipt of unpair command over the air (See [1]).

5.1.14.2 Usage

Table 18: RTI_UnpairInd() AREQ

1	1	1	*1*
Length = 0x01	*Cmd0 = 0x4A*	*Cmd1 = 0x0B*	*dstIndex*

5.1.15 RTI_SEND_DATA_REQ

5.1.15.1 Description

This command is issued by the host processor to remotely call `RTI_SendDataReq()` function of network processor (See [1]).

5.1.15.2 Usage

Table 19: RTI_SendDataReq() AREQ

1	1	1	1	*1*
Length = 6 + len	*Cmd0 = 0x4A*	*Cmd1 = 0x05*	*dstIndex*	*profileId*

2	1	1	*variable*
vendorId	*txOptions*	*len*	*data**

*data field corresponds to data payload referenced by pData argument of `RTI_SendDataReq()` function.

5.1.16 RTI_SEND_DATA_CNF

5.1.16.1 Description

This command is issued by the network processor to notify occurrence of `RTI_SendDataCnf()` callback in response to a prior `RTI_SEND_DATA_REQ` command (See [1]).

5.1.16.2 Usage

Table 20: RTI_SendDataCnf() AREQ

1	1	1	*1*
Length = 0x01	*Cmd0 = 0x4A*	*Cmd1 = 0x03*	*status*

5.1.17 RTI_RECEIVE_DATA_IND

5.1.17.1 Description

This command is issued by the network processor to notify occurrence of `RTI_ReceiveDataInd()` callback upon receipt of data over the air (See [1]).

5.1.17.2 Usage

Table 21: RTI_ReceiveDataInd() AREQ

1	1	1	1	*1*
Length = 7 + len	*Cmd0 = 0x4A*	*Cmd1 = 0x05*	*srcIndex*	*profileId*

2	1	1	1	***variable***
vendorId	*rxLQI*	*rxFlags*	*len*	*data**

*data field corresponds to data payload referenced by pData argument of `RTI_ReceiveDataInd()` function.

5.1.18 RTI_STANDBY_REQ

5.1.18.1 Description

This command is issued by the host processor to remotely call `RTI_StandbyReq()` function of network processor (See [1]).

5.1.18.2 Usage

Table 22: RTI_StandbyReq() AREQ

1	1	1	*1*
Length = 0x01	*Cmd0 = 0x4A*	*Cmd1 = 0x07*	*mode*

5.1.19 RTI_STANDBY_CNF

5.1.19.1 Description

This command is issued by the network processor to notify occurrence of `RTI_StandbyCnf()` callback in response to a prior `RTI_STANDBY_REQ` command (See [1]).

5.1.19.2 Usage

Table 23: RTI_StandbyCnf() AREQ

1	1	1	*1*
Length = 0x01	*Cmd0 = 0x4A*	*Cmd1 = 0x06*	*status*

5.1.20 RTI_RX_ENABLE_REQ

5.1.20.1 Description

This command is issued by the host processor to remotely call `RTI_RxEnableReq()` function of network processor (See [1]).

5.1.20.2 Usage

Table 24: RTI_RxEnableReq() AREQ

1	1	1	*2*
Length = 0x02	*Cmd0 = 0x4A*	*Cmd1 = 0x08*	*duration*

5.1.21 RTI_RX_ENABLE_CNF

5.1.21.1 Description

This command is issued by the network processor to notify occurrence of `RTI_RxEnableCnf()` callback in response to a prior `RTI_RX_ENABLE_REQ` command (See [1]).

5.1.21.2 Usage

Table 25: RTI_RxEnableCnf() AREQ

1	1	1	*1*
Length = 0x01	*Cmd0 = 0x4A*	*Cmd1 = 0x07*	*status*

5.1.22 RTI_ENABLE_SLEEP_REQ

5.1.22.1 Description

This command is issued by the host processor to remotely call `RTI_EnableSleepReq()` function of network processor (See [1]).

5.1.22.2 Usage

Table 26: RTI_EnableSleepReq() AREQ

1	1	*1*
Length = 0x00	*Cmd0 = 0x4A*	*Cmd1 = 0x09*

5.1.23 RTI_ENABLE_SLEEP_CNF

5.1.23.1 Description

This command is issued by the network processor to notify occurrence of `RTI_EnableSleepCnf()` callback in response to a prior `RTI_ENABLE_SLEEP_REQ` command (See [1]).

5.1.23.2 Usage

Table 27: RTI_EnableSleepCnf() AREQ

1	1	1	*1*
Length = 0x01	*Cmd0 = 0x4A*	*Cmd1 = 0x08*	*status*

5.1.24 RTI_DISABLE_SLEEP_REQ

5.1.24.1 Description

Note that this command is not used by UART transport protocol (See section 4.2.5 for UART transport protocol).

For all other transport protocols, this command is issued by the host processor to remotely call `RTI_DisableSleepReq()` function of network processor (See [1]).

5.1.24.2 Usage

Table 28: RTI_DisableSleepReq()

1	1	*1*
Length = 0x00	*Cmd0 = 0x4A*	*Cmd1 = 0x0A*

5.1.25 RTI_DISABLE_SLEEP_CNF

5.1.25.1 Description

Note that this command is not used by UART transport protocol (See section 4.2.5 for UART transport protocol).

For all other transport protocols, this command is issued by the network processor to notify occurrence of `RTI_DisableSleepCnf()` callback in response to a prior `RTI_DISABLE_SLEEP_REQ` command (See [1]).

5.1.25.2 Usage

Table 29: RTI_DisableSleepCnf() AREQ

1	1	1	*1*
Length = 0x01	*Cmd0 = 0x4A*	*Cmd1 = 0x09*	*status*

5.1.26 RTI_TEST_MODE_REQ

5.1.26.1 Description

This command is issued by the host processor to remotely call `RTI_TestModeReq()` function of network processor (See [1]).

5.1.26.2 Usage

Table 30: RTI_TestModeReq() AREQ

1	1	1	1	1	*1*
Length = 0x03	*Cmd0 = 0x4A*	*Cmd1 = 0x11*	*mode*	*txPower*	*channel*

5.1.27 RTI_TEST_RX_COUNTER_GET_REQ

5.1.27.1 Description

This command is issued by the application processor to remotely call `RTI_TestRxCounterGetReq()` function of network processor (See [1]).

5.1.27.2 Usage

Table 31: RTI_TestRxCounterGetReq() SREQ

1	1	1	*1*
Length = 0x01	*Cmd0 = 0x2A*	*Cmd1 = 0x12*	*resetFlag*

Table 32: RTI_TestRxCounterGetReq() SRSP

1	1	1	*2*
Length = 0x02	*Cmd0 = 0x6A*	*Cmd1 = 0x12*	*value**

*value corresponds to return value of `RTI_TestRxCounterGetReq()` function.

5.1.28 RTI_SW_RESET_REQ

5.1.28.1 Description

This command is issued by the host processor to remotely call `RTI_SwResetReq()` function of network processor (See [1]).

5.1.28.2 Usage

Table 33: RTI_SwResetReq() AREQ

1	1	*1*
Length = 0x00	*Cmd0 = 0x4A*	*Cmd1 = 0x13*

5.1.29 RTI_READ_ITEM_EX

5.1.29.1 Description

This command is issued by the application processor to remotely call `RTI_ReadItemEx()` function of network processor (See [1]).

5.1.29.2 Usage

Table 34: RTI_ReadItemEx() AREQ

1	1	1	1	1	*1*
Length = 0x03	*Cmd0 = 0x2A*	*Cmd1 = 0x21*	*profileId*	*itemId*	*len*

Table 35: RTI_ReadItemEx() SRSP

1	1	1	1	***variable***
Length = 1 + value length	*Cmd0 = 0x6A*	*Cmd1 = 0x21*	*status**	*value***

*status corresponds to return value of `RTI_ReadItemEx()` call

**value corresponds to data payload to be returned as pointed by `pValue` argument of `RTI_ReadItemEx()` function.

5.1.30 RTI_WRITE_ITEM_EX

5.1.30.1 Description

This command is issued by the application processor to remotely call `RTI_WriteItemEx()` function of network processor (See [1]).

5.1.30.2 Usage

Table 36: RTI_WriteItemEx() AREQ

1	1	1	1	1	1	***variable***
Length = 0x02 + value length	*Cmd0 = 0x2A*	*Cmd1 = 0x22*	*profileId*	*itemId*	*len*	*value**

Table 37: RTI_WriteItemEx() SRSP

1	1	1	*1*
Length = 0x01	*Cmd0 = 0x6A*	*Cmd1 = 0x22*	*status***

*value corresponds to data payload as pointed by pValue argument of `RTI_WriteItemEx()` function.

**status corresponds to return value of `RTI_WriteItemEx()` function.

5.2 RemoTI Network Layer Interface

In addition to application framework interface in section 5.1, RemoTI provides direct command interface to network layer through network processor interface. Note that with the exception of `RCN_NLME_GET_REQ` command and `RCN_NLME_SET_REQ` command, the network layer interface commands must be used exclusively to the application framework interface commands. That is, if the host processor chooses to use network layer interface, it must not use application framework interface at all. If the host processor chooses to use the application framework interface, it must not use network layer interface except for `RCN_NLME_GET_REQ` command and `RCN_NLME_SET_REQ` command.

The following subsections describe format of individual commands. See [1] for details of remote functions referred in each subsection.

5.2.1 RCN_NLDE_DATA_REQ

5.2.1.1 Description

This command is issued by the host processor to remotely call `RCN_NldeDataAlloc()` function followed by `RCN_NldeDataReq()` function of network processor.

5.2.1.2 Usage

Table 38: RCN_NldeDataReq() AREQ

1	1	1	1	*1*
Length = nn	*Cmd0 = 0x4B*	*Cmd1 = 0x01*	*pairingRef*	*profileId*

2	1	1	*variable*
vendorId	*nsduLength*	*txOptions*	*nsdu*

5.2.2 RCN_NLDE_DATA_IND

5.2.2.1 Description

This command is issued by the network processor to remotely call `RCN_CbackEvent()` function for `RCN_NLDE_DATA_IND` event.

5.2.2.2 Usage

Table 39: RCN_NLDE_DATA_IND AREQ

1	1	1	1	*1*
Length = nn	*Cmd0 = 0x4C*	*Cmd1 = 0x01*	*pairingRef*	*profileId*

2	1	1	1	*variable*
vendorId	*nsduLength*	*rxLinkQuality*	*rxFlags*	*nsdu*

5.2.3 RCN_NLDE_DATA_CNF

5.2.3.1 Description

This command is issued by the network processor to remotely call `RCN_CbackEvent()` function for `RCN_NLDE_DATA_CNF` event.

5.2.3.2 Usage

Table 40: RCN_NLDE_DATA_CNF AREQ

1	1	1	1	*1*
Length = 0x02	*Cmd0 = 0x4C*	*Cmd1 = 0x02*	*status*	*pairingRef*

5.2.4 RCN_NLME_COMM_STATUS_IND

5.2.4.1 Description

This command is issued by the network processor to remotely call `RCN_CbackEvent()` function for `RCN_NLME_COMM_STATUS_IND` event.

5.2.4.2 Usage

Table 41: RCN_NLME_COMM_STATUS_IND AREQ

1	1	1	1	*1*
Length = 0x0D	*Cmd0 = 0x4C*	*Cmd1 = 0x03*	*status*	*pairingRef*

2	1	*8*
dstPanId	*dstAddrMode*	*dstAddr**

- `dstAddr` could be either IEEE address or 16 bit short address depending on the `dstAddrMode` field value.

5.2.5 RCN_NLME_DISCOVERY_REQ

5.2.5.1 Description

This command is issued by the host processor to remotely call `RCN_NlmeDiscoveryReq()` function.

5.2.5.2 Usage

Table 42: RCN_NlmeDiscoveryReq() AREQ

1	1	1	2
Length = 0x1A	*Cmd0 = 0x4B*	*Cmd1 = 0x02*	*dstPanId*

2	1	3
dstNwkAddr	*appCapabilities*	*devTypeList*

7		*1*	1
profileIdList		*searchDevType*	*discProfileIdList-Size*

7		2
discProfileIdList		*discDurationInMs*

5.2.6 RCN_NLME_DISCOVERY_IND

5.2.6.1 Description

This command is issued by the network processor to remotely call `RCN_CbackEvent()` function for `RCN_NLME_DISCOVERY_IND` event.

5.2.6.2 Usage

Table 43: RCN_NLME_DISCOVERY_IND AREQ

1	1	1	1		8
Length = 0x2F	*Cmd0 = 0x4C*	*Cmd1 = 0x04*	*status*		*orgIeeeAddress*

1	2	7		1
nodeCapabilities	*vendorId*	*vendorString*		*appCapabilities*

15		3
userString		*devTypeList*

	7	1	1
	ProfileIdList	*searchDevType*	*rxLinkQuality*

5.2.7 RCN_NLME_DISCOVERY_RSP

5.2.7.1 Description

This command is issued by the host processor to call `RCN_NlmeDiscoveryRsp()` function.

5.2.7.2 Usage

Table 44: RCN_NlmeDiscoveryRsp() AREQ

1	1	1	1	8	
Length = 0x15	*Cmd0 = 0x4B*	*Cmd1 = 0x03*	*status*	*dstIeeeAddress*	

1	3	7	
appCapabilities	*devTypeList*	*ProfileIdList*	

1
discReqLqi

5.2.8 RCN_NLME_DISCOVERED_EVENT

5.2.8.1 Description

This command is issued by the network processor to remotely call `RCN_CbackEvent()` function for `RCN_NLME_DISCOVERED_EVENT` event.

5.2.8.2 Usage

Table 45: RCN_NLME_DISCOVERED_EVENT AREQ

1	1	1	1	1	1
Length = 0x30	*Cmd0 = 0x4C*	*Cmd1 = 0x05*	*status*	*logicalChannel*	*panId*

	8	1	2
	IeeeAddress	*nodeCapabilities*	*vendorId*

7		1	15	
vendorString		*appCapabililities*	*userString*	

3	7		1
devTypeList	*ProfileIdList*		*discReqLqi*

5.2.9 RCN_NLME_DISCOVERY_CNF

5.2.9.1 Description

This command is issued by the network processor to remotely call `RCN_CbackEvent()` function for `RCN_NLME_DISCOVERY_CNF` event.

5.2.9.2 Usage

Table 46: RCN_NLME_DISCOVERY_CNF AREQ

1	1	1	1	*1*
Length = 0x02	*Cmd0 = 0x4C*	*Cmd1 = 0x06*	*status*	*numNodes*

5.2.10 RCN_NLME_DISCOVERY_ABORT_REQ

5.2.10.1 Description

This command is issued by the host processor to remotely call `RCN_NlmeDiscoveryAbortReq()` function.

5.2.10.2 Usage

Table 47: RCN_NlmeDiscoveryAbortReq() AREQ

1	1	*1*
Length = 0x00	*Cmd0 = 0x4B*	*Cmd1 = 0x0F*

5.2.11 RCN_NLME_DISCOVERY_ABORT_CNF

5.2.11.1 Description

This command is issued by the network processor to remotely call `RCN_CbackEvent()` function for `RCN_NLME_DISCOVERY_ABORT_CNF` event.

5.2.11.2 Usage

Table 48: RCN_NLME_DISCOVERY_ABORT_CNF AREQ

1	1	*1*
Length = 0x00	*Cmd0 = 0x4C*	*Cmd1 = 0x11*

5.2.12 RCN_NLME_GET_REQ

5.2.12.1 Description

This command is issued by the host processor to remotely call `RCN_NlmeGetReq()` function.

5.2.12.2 Usage

Table 49: RCN_NlmeGetReq() AREQ or SREQ

1	1	1	1	*1*
Length = 0x02	*Cmd0 = 0x4B or 0x2B*	*Cmd1 = 0x04*	*attribute*	*attributeIndex*

This command can be transported as either a synchronous request message or an asynchronous request message. The corresponding `RCN_NLME_GET_CNF` command is generated as either a synchronous response message or an asynchronous request message matching the message type of the request.

5.2.13 RCN_NLME_GET_CNF

5.2.13.1 Description

This command is issued by the network processor to pass the result of the `RCN_NlmeGetReq()` function call triggered by `RCN_NLME_GET_REQ` command.

5.2.13.2 Usage

Table 50: RCN_NlmeGetCnf() AREQ or SRSP

1	1	1	1	1	1
Length = nn	*Cmd0 = 0x4C or 0x6C*	*Cmd1 = 0x07*	*status*	*attribute*	*attributeIndex*

1	Variable
length	*value*

5.2.14 RCN_NLME_PAIR_REQ

5.2.14.1 Description

This command is issued by the host processor to remotely call `RCN_NlmePairReq()` function.

5.2.14.2 Usage

Table 51: RCN_NlmePairReq() AREQ

1	1	1	1	8
Length = 0x17	*Cmd0 = 0x4B*	*Cmd1 = 0x05*	*logicalChannel*	*dstIeeeAddress*

2	1	3
dstPanId	*appCapabilities*	*devTypeList*

7	1
profileIdList	*keyExTransfer-Count*

5.2.15 RCN_NLME_PAIR_IND

5.2.15.1 Description

This command is issued by the network processor to remotely call `RCN_CbackEvent()` function for `RCN_NLME_PAIR_IND` event.

5.2.15.2 Usage

Table 52: RCN_NLME_PAIR_IND AREQ

1	1	1	1	2
Length = 0x30	*Cmd0 = 0x4C*	*Cmd1 = 0x08*	*status*	*srcPanId*

8
orgIeeeAddress

1	2	7	1
NodeCapabilities	*vendorId*	*vendorString*	*AppCapabilities*

15	3
userString	*devTypeList*

7	1	1
ProfileIdList	*provPairingRef*	*keyExTransfer-Count*

5.2.16 RCN_NLME_PAIR_RSP

5.2.16.1 Description

This command is issued by the host processor to remotely call `RCN_NlmePairRsp()` function.

5.2.16.2 Usage

Table 53: RCN_NlmePairRsp() AREQ

1	1	1	1	2
Length = 0x17	*Cmd0 = 0x4B*	*Cmd1 = 0x06*	*status*	*dstPanId*

8		1
dstIeeeAddress		*appCapabilities*

3	7		1
devTypeList	*ProfileIdList*		*provPairingRef*

5.2.17 RCN_NLME_PAIR_CNF

5.2.17.1 Description

This command is issued by the network processor to remotely call `RCN_CbackEvent()` function for `RCN_NLME_PAIR_CNF` event.

5.2.17.2 Usage

Table 54: RCN_NLME_PAIR_CNF AREQ

1	1	1	1	1	1
Length = 0x26	*Cmd0 = 0x4C*	*Cmd1 = 0x09*	*status*	*pairingRef*	*nodeCapabilities*

2	7	
vendorId	*vendorString*	

1	15	
appCapabililities	*userString*	

3	7	
devTypeList	*ProfileIdList*	

5.2.18 RCN_NLME_RESET_REQ

5.2.18.1 Description

This command is issued by the host processor to remotely call `RCN_NlmeResetReq()` function.

5.2.18.2 Usage

Table 55: RCN_NlmeResetReq() AREQ or SREQ

1	1	1	1
Length = 0x01	*Cmd0 = 0x4B or 0x2B*	*Cmd1 = 0x07*	*setDefaultNib*

5.2.19 RCN_NLME_RESET_CNF

5.2.19.1 Description

This command is issued by the network processor to pass the result of `RCN_NlmeResetReq()` call triggered by the `RCN_NLME_RESET_REQ` command.

5.2.19.2 Usage

Table 56: RCN_NlmeResetCnf() AREQ or SRSP

1	1	1	*1*
Length = 0x01	*Cmd0 = 0x4C or 0x6C*	*Cmd1 = 0x0A*	*status**

status field is always set to SUCCESS (0x00).

5.2.20 RCN_NLME_RX_ENABLE_REQ

5.2.20.1 Description

This command is issued by the host processor to remotely call `RCN_NlmeRxEnableReq()` function.

5.2.20.2 Usage

Table 57: RCN_NlmeRxEnableReq() AREQ or SREQ

1	1	1	*2*
Length = 0x02	*Cmd0 = 0x4B or 0x2B*	*Cmd1 = 0x08*	*rxOnDuration-InMs*

5.2.21 RCN_NLME_RX_ENABLE_CNF

5.2.21.1 Description

This command is issued by the network processor to pass the result of `RCN_NlmeRxEnableReq()` call triggered by `RCN_NLME_RX_ENABLE_REQ` command.

5.2.21.2 Usage

Table 58: RCN_NlmeRxEnableCnf() AREQ or SRSP

1	1	1	*1*
Length = 0x01	*Cmd0 = 0x4C or 0x6C*	*Cmd1 = 0x0B*	*status*

5.2.22 RCN_NLME_SET_REQ

This command is issued by the host processor to remotely call `RCN_NlmeSetReq()` function.

5.2.22.1 Usage

Table 59: RCN_NlmeSetReq() AREQ or SREQ

1	1	1	1	*1*
Length = 0xnn	*Cmd0 = 0x4B or 0x2B*	*Cmd1 = 0x09*	*nibAttribute*	*nibAttribute-Index*

1	*Variable*
length	*value*

5.2.23 RCN_NLME_SET_CNF

5.2.23.1 Description

This command is issued by the network processor to pass the result of the `RCN_NlmeSetReq()` function call triggered by `RCN_NLME_SET_REQ` command.

5.2.23.2 Usage

Table 60: RCN_NlmeSetCnf() AREQ or SRSP

1	1	1	1	1	*1*
Length = 0x03	*Cmd0 = 0x4C or 0x6C*	*Cmd1 = 0x0C*	*status*	*nibAttribute*	*nibAttribute-Index*

5.2.24 RCN_NLME_START_REQ

5.2.24.1 Description

This command is issued by the host processor to remotely call `RCN_NlmeStartReq()` function.

5.2.24.2 Usage

Table 61: RCN_NlmeStartReq() AREQ

1	1	*1*
Length = 0x00	*Cmd0 = 0x4B*	*Cmd1 = 0x0A*

5.2.25 RCN_NLME_START_CNF

5.2.25.1 Description

This command is issued by the network processor to remotely call `RCN_CbackEvent()` function for `RCN_NLME_START_CNF` event.

5.2.25.2 Usage

Table 62: RCN_NLME_START_CNF AREQ

1	1	1	*1*
Length = 0x01	*Cmd0 = 0x4C*	*Cmd1 = 0x0D*	*status*

5.2.26 RCN_NLME_UNPAIR_REQ

5.2.26.1 Description

This command is issued by the host processor to remotely call `RCN_NlmeUnpairReq()` function.

5.2.26.2 Usage

Table 63: RCN_NlmeUnpairReq() AREQ

1	1	1	*1*
Length = 0x01	*Cmd0 = 0x4B*	*Cmd1 = 0x0B*	*pairingRef*

5.2.27 RCN_NLME_UNPAIR_IND

5.2.27.1 Description

This command is issued by the network processor to remotely call `RCN_CbackEvent()` function for `RCN_NLME_UNPAIR_IND` event.

5.2.27.2 Usage

Table 64: RCN_NLME_UNPAIR_IND AREQ

1	1	1	*1*
Length = 0x01	*Cmd0 = 0x4C*	*Cmd1 = 0x0F*	*pairingRef*

5.2.28 RCN_NLME_UNPAIR_RSP

5.2.28.1 Description

This command is issued by the host processor to remotely call `RCN_NlmeUnpairRsp()` function.

5.2.28.2 Usage

Table 65: RCN_NlmeUnpairRsp() AREQ

1	1	1	1
Length = 0x01	*Cmd0 = 0x4B*	*Cmd1 = 0x0C*	*pairingRef*

5.2.29 RCN_NLME_UNPAIR_CNF

5.2.29.1 Description

This command is issued by the network processor to remotely call `RCN_CbackEvent()` function for `RCN_NLME_UNPAIR_CNF` event.

5.2.29.2 Usage

Table 66: RCN_NLME_UNPAIR_CNF AREQ

1	1	1	1	1
Length = 0x02	*Cmd0 = 0x4C*	*Cmd1 = 0x0E*	*status*	*pairingRef*

5.2.30 RCN_NLME_AUTO_DISCOVERY_REQ

5.2.30.1 Description

This command is issued by the host processor to remotely call `RCN_NlmeAutoDiscoveryReq()` function.

5.2.30.2 Usage

Table 67: RCN_NlmeAutoDiscoveryReq() AREQ

1	1	1	2
Length = 0x0D	*Cmd0 = 0x4B*	*Cmd1 = 0x0D*	*autoDiscDurationInMs*

1	3	7
appCapabilities	*devTypeList*	*profileIdList*

5.2.31 RCN_NLME_AUTO_DISCOVERY_CNF

5.2.31.1 Description

This command is issued by the network processor to remotely call `RCN_CbackEvent()` function for `RCN_NLME_AUTO_DISCOVERY_CNF` event.

5.2.31.2 Usage

Table 68: RCN_NLME_AUTO_DISCOVERY_CNF AREQ

1	1	1	1	8
Length = 0x09	*Cmd0 = 0x4C*	*Cmd1 = 0x10*	*status*	*srcIeeeAddr*

5.2.32 RCN_NLME_AUTO_DISCOVERY_ABORT_REQ

5.2.32.1 Description

This command is issued by the host processor to remotely call `RCN_NlmeAutoDiscoveryAbortReq()` function.

5.2.32.2 Usage

Table 69: RCN_NlmeAutoDiscoveryAbortReq() AREQ

1	1	1
Length = 0x00	*Cmd0 = 0x4B*	*Cmd1 = 0x0E*

6. Build Configuration

CC253x RemoTI network processor project of RemoTI software is configured with UART interface, by default.

The selection of UART vs. SPI is done by preprocessor definition, which is captured in `np_main.cfg` file, which can be found under Application group under rnp_cc2530 workspace. The preprocessor definitions must be set as Table 70, out of the box.

Table 70. Preprocessor Definitions for UART

Definition	Value	Description
`HAL_SPI`	`FALSE`	SPI interface selection. It must not be selected `TRUE` when `HAL_UART` is selected `TRUE`.
`HAL_UART`	`TRUE`	UART interface selection
`INT_HEAP_LEN`	Greater than or equal to 1280	Heap size. It is not relevant to UART or SPI configuration. The value is optimized for the network layer heap usage and static memory usage. For CC253x, greater value is recommended as far as such fits the memory usage for application.

To configure the software for SPI interface the preprocessor definitions have to be modified as in Table 71.

Table 71. Preprocessor Definitions for SPI

Definition	Value	Description
`HAL_SPI`	`TRUE`	SPI interface selection.
`HAL_UART`	`FALSE`	UART interface selection. It must not be selected `TRUE` when `HAL_SPI` is selected `TRUE`.
`INT_HEAP_LEN`	Greater than or equal to 1280	Heap size. It is not relevant to UART or SPI configuration. The value is optimized for the network layer heap usage and static memory usage. For CC253x, greater value is recommended as far as such fits the memory usage for application.

7. General Information

7.1 Document History

Revision	Date	Description/Changes
1.0	2009.04.15	Initial release.
1.1	2009.07.28	Added network layer interface command section and new application framework interfaces such as `RTI_UnpairReq`, `RTI_UnpairCnf`, `RTI_UnpairInd`, `RTI_PairAbortReq`, `RTI_PairAbortCnf` and `RTI_AllowPairAbortReq`. Corrected `RTI_WRITE_ITEM` command frame length value. Clarification of `RTI_DISABLE_SLEEP_REQ` and `RTI_DISABLE_SLEEP_CNF` usage.
swra271a	2009.09.18	Corrected revision numbering scheme.
swra271b	2009.10.27	Corrected section 6.1.24.
swra271c	2010.07.01	Added references to CC2533.
swra271d	2011.03.02	Added extended `RTI_Read/WriteEx()` and deprecated the old `RTI_Read/Write()`.
swra271e	2011.10.01	Add I2C interface description chapter
swra271f	2012.10.02	Add additional SPI interface timing information

Address Information

Texas Instruments Norway AS
Gaustadalléen 21
N-0349 Oslo
NORWAY
Phone: +47 22 95 85 44
Fax: +47 22 95 85 46
Web site: http://www.ti.com/lpw

Texas Instruments Incorporated
Low-Power RF Software Development
9276 Scranton Road, Suite 450
San Diego CA 92121
United States of America
Phone: +1 858 638 4294
Fax: +1 858 638 4202
Web site: http://www.ti.com/lpw

IMPORTANT NOTICE

Texas Instruments Incorporated and its subsidiaries (TI) reserve the right to make corrections, modifications, enhancements, improvements, and other changes to its products and services at any time and to discontinue any product or service without notice. Customers should obtain the latest relevant information before placing orders and should verify that such information is current and complete. All products are sold subject to TI's terms and conditions of sale supplied at the time of order acknowledgment.

TI warrants performance of its hardware products to the specifications applicable at the time of sale in accordance with TI's standard warranty. Testing and other quality control techniques are used to the extent TI deems necessary to support this warranty. Except where mandated by government requirements, testing of all parameters of each product is not necessarily performed.

TI assumes no liability for applications assistance or customer product design. Customers are responsible for their products and applications using TI components. To minimize the risks associated with customer products and applications, customers should provide adequate design and operating safeguards.

TI does not warrant or represent that any license, either express or implied, is granted under any TI patent right, copyright, mask work right, or other TI intellectual property right relating to any combination, machine, or process in which TI products or services are used. Information published by TI regarding third-party products or services does not constitute a license from TI to use such products or services or a warranty or endorsement thereof. Use of such information may require a license from a third party under the patents or other intellectual property of the third party, or a license from TI under the patents or other intellectual property of TI.

Reproduction of TI information in TI data books or data sheets is permissible only if reproduction is without alteration and is accompanied by all associated warranties, conditions, limitations, and notices. Reproduction of this information with alteration is an unfair and deceptive business practice. TI is not responsible or liable for such altered documentation. Information of third parties may be subject to additional restrictions.

Resale of TI products or services with statements different from or beyond the parameters stated by TI for that product or service voids all express and any implied warranties for the associated TI product or service and is an unfair and deceptive business practice. TI is not responsible or liable for any such statements.

TI products are not authorized for use in safety-critical applications (such as life support) where a failure of the TI product would reasonably be expected to cause severe personal injury or death, unless officers of the parties have executed an agreement specifically governing such use. Buyers represent that they have all necessary expertise in the safety and regulatory ramifications of their applications, and acknowledge and agree that they are solely responsible for all legal, regulatory and safety-related requirements concerning their products and any use of TI products in such safety-critical applications, notwithstanding any applications-related information or support that may be provided by TI. Further, Buyers must fully indemnify TI and its representatives against any damages arising out of the use of TI products in such safety-critical applications.

TI products are neither designed nor intended for use in military/aerospace applications or environments unless the TI products are specifically designated by TI as military-grade or "enhanced plastic." Only products designated by TI as military-grade meet military specifications. Buyers acknowledge and agree that any such use of TI products which TI has not designated as military-grade is solely at the Buyer's risk, and that they are solely responsible for compliance with all legal and regulatory requirements in connection with such use.

TI products are neither designed nor intended for use in automotive applications or environments unless the specific TI products are designated by TI as compliant with ISO/TS 16949 requirements. Buyers acknowledge and agree that, if they use any non-designated products in automotive applications, TI will not be responsible for any failure to meet such requirements.

Following are URLs where you can obtain information on other Texas Instruments products and application solutions:

Products		Applications	
Amplifiers	amplifier.ti.com	Audio	www.ti.com/audio
Data Converters	dataconverter.ti.com	Automotive	www.ti.com/automotive
DSP	dsp.ti.com	Broadband	www.ti.com/broadband
Interface	interface.ti.com	Digital Control	www.ti.com/digitalcontrol
Logic	logic.ti.com	Military	www.ti.com/military
Power Mgmt	power.ti.com	Optical Networking	www.ti.com/opticalnetwork
Microcontrollers	microcontroller.ti.com	Security	www.ti.com/security
		Telephony	www.ti.com/telephony
		Video & Imaging	www.ti.com/video
		Wireless	www.ti.com/wireless

Mailing Address: Texas Instruments, Post Office Box 655303, Dallas, Texas 75265

IMPORTANT NOTICE

Texas Instruments Incorporated and its subsidiaries (TI) reserve the right to make corrections, enhancements, improvements and other changes to its semiconductor products and services per JESD46, latest issue, and to discontinue any product or service per JESD48, latest issue. Buyers should obtain the latest relevant information before placing orders and should verify that such information is current and complete. All semiconductor products (also referred to herein as “components”) are sold subject to TI’s terms and conditions of sale supplied at the time of order acknowledgment.

TI warrants performance of its components to the specifications applicable at the time of sale, in accordance with the warranty in TI’s terms and conditions of sale of semiconductor products. Testing and other quality control techniques are used to the extent TI deems necessary to support this warranty. Except where mandated by applicable law, testing of all parameters of each component is not necessarily performed.

TI assumes no liability for applications assistance or the design of Buyers’ products. Buyers are responsible for their products and applications using TI components. To minimize the risks associated with Buyers’ products and applications, Buyers should provide adequate design and operating safeguards.

TI does not warrant or represent that any license, either express or implied, is granted under any patent right, copyright, mask work right, or other intellectual property right relating to any combination, machine, or process in which TI components or services are used. Information published by TI regarding third-party products or services does not constitute a license to use such products or services or a warranty or endorsement thereof. Use of such information may require a license from a third party under the patents or other intellectual property of the third party, or a license from TI under the patents or other intellectual property of TI.

Reproduction of significant portions of TI information in TI data books or data sheets is permissible only if reproduction is without alteration and is accompanied by all associated warranties, conditions, limitations, and notices. TI is not responsible or liable for such altered documentation. Information of third parties may be subject to additional restrictions.

Resale of TI components or services with statements different from or beyond the parameters stated by TI for that component or service voids all express and any implied warranties for the associated TI component or service and is an unfair and deceptive business practice. TI is not responsible or liable for any such statements.

Buyer acknowledges and agrees that it is solely responsible for compliance with all legal, regulatory and safety-related requirements concerning its products, and any use of TI components in its applications, notwithstanding any applications-related information or support that may be provided by TI. Buyer represents and agrees that it has all the necessary expertise to create and implement safeguards which anticipate dangerous consequences of failures, monitor failures and their consequences, lessen the likelihood of failures that might cause harm and take appropriate remedial actions. Buyer will fully indemnify TI and its representatives against any damages arising out of the use of any TI components in safety-critical applications.

In some cases, TI components may be promoted specifically to facilitate safety-related applications. With such components, TI’s goal is to help enable customers to design and create their own end-product solutions that meet applicable functional safety standards and requirements. Nonetheless, such components are subject to these terms.

No TI components are authorized for use in FDA Class III (or similar life-critical medical equipment) unless authorized officers of the parties have executed a special agreement specifically governing such use.

Only those TI components which TI has specifically designated as military grade or “enhanced plastic” are designed and intended for use in military/aerospace applications or environments. Buyer acknowledges and agrees that any military or aerospace use of TI components which have ***not*** been so designated is solely at the Buyer's risk, and that Buyer is solely responsible for compliance with all legal and regulatory requirements in connection with such use.

TI has specifically designated certain components as meeting ISO/TS16949 requirements, mainly for automotive use. In any case of use of non-designated products, TI will not be responsible for any failure to meet ISO/TS16949.

Products		**Applications**	
Audio	www.ti.com/audio	Automotive and Transportation	www.ti.com/automotive
Amplifiers	amplifier.ti.com	Communications and Telecom	www.ti.com/communications
Data Converters	dataconverter.ti.com	Computers and Peripherals	www.ti.com/computers
DLP® Products	www.dlp.com	Consumer Electronics	www.ti.com/consumer-apps
DSP	dsp.ti.com	Energy and Lighting	www.ti.com/energy
Clocks and Timers	www.ti.com/clocks	Industrial	www.ti.com/industrial
Interface	interface.ti.com	Medical	www.ti.com/medical
Logic	logic.ti.com	Security	www.ti.com/security
Power Mgmt	power.ti.com	Space, Avionics and Defense	www.ti.com/space-avionics-defense
Microcontrollers	microcontroller.ti.com	Video and Imaging	www.ti.com/video
RFID	www.ti-rfid.com		
OMAP Applications Processors	www.ti.com/omap	**TI E2E Community**	e2e.ti.com
Wireless Connectivity	www.ti.com/wirelessconnectivity		

Mailing Address: Texas Instruments, Post Office Box 655303, Dallas, Texas 75265