neo.cortec

Integration Manual for NCxxxx series Modules

Doc Status:	**Release**
Doc version:	**2.0**
Date:	**Sept 2015**

Table of Contents

1 Document revisions

Document version	Details
1.4	Initial release
1.5	Added details for HAPA & corrected application data 0x50 & 0x51
2.0	Document updated to include details for NC1000 & NC0400 as well. Also added documentation regarding the API for embedded controllers.

2 Introduction

This document describes how to integrate the NEOCORTEC NCxxxx series modules with a Host Controller both from a SW and HW perspective. The Host Controller can be an embedded micro controller or a PC.

3 Abbreviations

- HW – Hardware
- SW – Software
- UART - Universal Asynchronous Receiver/Transmitter
- RX – Receive
- TX - Transmit

4 Definitions

- Host – A system consisting of HW and SW which is interfacing to the NCxxxx module in order to use the module as a wireless transmission system for data such as sensing data or control parameters.

5 System Considerations

When designing a product that contains the NCxxxx module, there are a few items from a System point of view that need to be considered. This section of the document lists these items, and where applicable provides recommendations on solutions.

5.1 Power Supply

5.1.1 Voltage capabilities

The NCxxxx module series is designed such that it can be powered directly from a single cell battery in the voltage range of 2.0V to 3.6V DC. However, the module contains an internal linear voltage regulator, which regulates the voltage down to 1.8V for some parts of the system. Supplying the module at higher voltages (above 2.0V DC) leads to some level of inefficiency due to the nature of the linear converter.
Before considering the addition of a Switching Mode Regulator in the supply though, it should be noted that the extreme low power capabilities of the NCxxxx module is achieved through very low duty cycle. Therefore most of the average current consumption stems from the actual sleep mode current of the module. This means that the leakage current of the potential Switching Mode DC/DC regulator needs to be very low in order not to jeopardize the low Power Consumption.

5.1.2 I/O Voltage levels

The NCxxxx modules I/O voltage levels follow the supply voltage. This means that the external controller needs to use the same logical voltage range.

5.1.3 Current sourcing capabilities

Even though the average current consumption is ultra low, the module does draw current at a higher level in short bursts. This happens when the module is transmitting or receiving data or control information (see data sheet for active mode RX and TX current consumption).
The worst-case situation is when the module is performing its Beacon Scan. The duration of the individual Beacon Scans is dependent on configuration settings.
The power supply should stay within the supply voltage range of the NEOCORTEC module during the current bursts.
When operating the module from a battery supply, it should be noted that some batteries will greatly reduce their lifetime when their max rated current sourcing capabilities are violated.
Be careful to select batteries that match both the average current consumption as well as the peak current consumption.

5.2 Sleep

As the external controller, as well as the NEOCORTEC Module, is expected to spend a lot of time in sleep mode to save power, it is important that wake-up is synchronized, so that the modules can communicate with each other.

The external controller can enter sleep mode when it has finished processing of e.g. sensor data on its inputs, or data received from the NEOCORTEC module.

The NEOCORTEC node controls wake-up exclusively. When the NEOCORTEC node is active, the WakeUp signal is active. Likewise, when the NEOCORTEC node is in sleep, the WakeUp signal is inactive[1].

The external controller will have to be in active state whenever the NEOCORTEC node is in active state. This to ensure that the external controller is ready to receive UART data whenever the NEOCORTEC node potentially can send such data.

For more details, see chapter 6 Serial UART Interface.

6 Serial UART Interface

The NEOCORTEC NCxxxx module provides two serial communications ports, which are, used for each their own individual purpose:

Name	Desciption
Application API	Used to interface the Host Controller Application layer with the NC2400 Module. Provides functions for sending and receiving payload data through the NEOCORTEC wireless network
System API	Used for configuration and debugging purposes. Will typically be connected to a PC when the user either configures the module, or during the development process to provide trace outputs.

The Host Processor can communicate with the NCxxxx module through a standard UART serial interface (see module data sheet for electrical details and Pin locations).

[1] See NCxxxx datasheets for details on the WakeUp signal.

6.1 Physical interface

The following signals are used:

Abbreviation	Name	Module pin number		Origin	Purpose
		Application	System		
RxD	Received Data	17	5	NEO	Carries data from the NEOCORTEC node to the external controller.
TxD	Transmitted Data	16	3	EXT	Carries data from the external controller to the NEOCORTEC node
CTS	Clear to Send	19	4 (nWU used as CTS)	NEO	Indicates that the NEOCORTEC node is ready to accept commands (see below)
nWU	Wake Up	4	n/a	NEO	Indicates the activity mode of the NEOCORTEC node (sleep/active). Active low

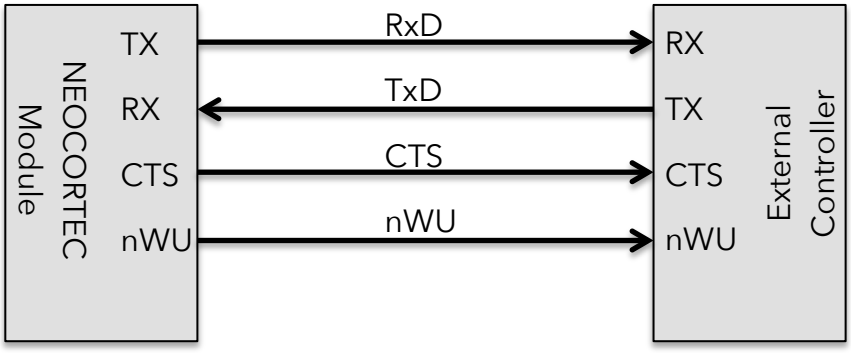

Figure 1 - Generic UART Interface drawing

nWU is used as a wake-up signal, to indicate that the NEOCORTEC module is waking up, and that it *may* have data to send. The external controller must not ignore this wake-up, and must start the UART receiver at latest at the time specified by WU setup time, after wake-up. This parameter can be changed by changing the module configuration, but it is default set to 100us.

The CTS is used to signal the controller that the NEOCORTEC module can receive data.

The UART should be configured with the following settings:

- Transfer Speed: 115.2 kbps
- Data bits: 8
- Stop bits: 1
- Parity Check: None

6.2 Easy Integration with PC

For easy integration with a PC, it is recommended to use a UART to USB converter. The TTL-232R-3V3[2] cable from FTDI (www.ftdichip.com) has been proven to work with the NC2400 module directly.
Connect the pins according to this table:

Name	Module pin number		FTDI Pin number
	Application	System	
RxD	17	5	5
TxD	16	3	4
CTS	19	4	2
nWU	13	n/a	n/a
GND	1,6,9,15, 26		1

Note: nWU is not connected, as it is assumed the PC will not be entering sleep mode, and will always be ready to send and receive data.

7 Communication Specification – Application UART

7.1 Logical data exchange

Data is exchanged over the interface in Big Endian byte order.

The general format of a data frame is:

Section	Code	Length	UART Payload
Length	1 byte	1 byte	(Length) bytes

The Length field indicates the length of the "UART Payload" in bytes.

[2] Note that the FTDI cable expects logic levels at 3.3v, and as such the NEOCORTEC Module will have to be supplied from a 3.3v supply. Pin 3 of the FTDI cable is a 5V output, and if regulated properly, it can be used as a supply for the NEOCORTEC Module.

7.2 Commands and Application data

There are two categories of frames exchanged, Commands and Application data:

Commands are frames accepted by the NEOCORTEC node.

Application data is data sent from the NEOCORTEC node to the External Host.

Some commands will cause values to be returned. These values are sent as application data (frames).

Some data frames come from information received from another node in the network, and as such it can be subject to long delays. The principle is that the application should continue execution, and react on the return value based on the Application Data Code, which associates it with the command previously issued.

If the command listed hereunder has a corresponding return frame, the Application Data code is listed under the “Return” column. For a definition of these codes, please refer to 7.2.2

7.2.1 List of commands

Command	Code	Length	Data	Return	Description
API_CMD_TX_UNACKNOWLEDGED [3]	0x02	n	NEOCORTEC ID + 1 byte port + payload	n/a	Write data to device / port. Maximum payload size is 19 byte Refer to text below table for more details.
API_CMD_TX_ACKNOWLEDGED	0x03	n	NEOCORTEC ID + 1 byte port + payload	0x50, 0x51	Write data to device / port. Maximum payload size is 19 byte Refer to text below table for more details.

NEOCORTEC ID is the address of the NEOCORTEC Module, for which the payload data is intended.
The NEOCORTEC ID is two bytes. The first 127d addresses should be used for the sink nodes in the network. The sink nodes are the ones that can be addressed directly as a destination for payload data. Addresses over 127d should be used for nodes which are only sources of payload data or routers of payload data. If the total number of nodes are less than 127d, then it is recommended to use addresses in the range 1d…127d. Addresses 0x00 00 and 0xFF FF are not allowed.

The port field of the data, can be used at the application layer, to direct the data to different applications. One can for instance be a lower level control application, used to set parameters of the application, while other ports is ordinary application data.
IMPORTANT: Only the last two bits (LSB) in the port argument can be used:
For the application layer, only 2 bit port numbers are available (i.e. 4 ports total).

Send Unacknowledged Packet:(currently not supported)
The command is used to initiate a transmission of application payload data to another NEOCORTEC Module inside the NEOCORTEC network. The NEOCORTEC ID should be given as well as a port number. The port number is used to direct the payload data to a given handler/service at the receiving NEOCORTEC application layer.
Once the packet has been enqueued in the NEOCORTEC module, there will be no messaging back to the application layer indicating if the transmission to the destination was successful.

Send Acknowledged Packet:
Similar to the above mentioned command, this command initiates the transmission of application payload data to another NEOCORTEC module inside the NEOCORTEC

[3] Unacknowledged transmission of payload data is currently not supported by the protocol stack.

network. The difference is that the payload data is transmitted with end-to-end acknowledge enabled. This means that once the application payload data is received at the destination, an acknowledge message is transmitted back. When the Ack or NAK is received, a corresponding Application code will be transmitted from the NEOCORTEC module to the external controller (see next chapter).

7.2.2 List of application data

Code	Length	Data	Description
0x50	2	ID(2)	Acknowledge for previously sent packet
0x51	2	ID(2)	Non-Acknowledge for previously sent packet
0x52	n	Header + Payload	Data received from another device (see below)
0x53	n	Header + Payload	Data received from another device with HAPA (see below)

0x50:

Receiving an Acknowledge is a guarantee that the recipient (with address = ID) has received the packet sent previously in acknowledged mode.

Acknowledges from a recipient node is received for packets in the order they were enqueued.

0x51:

Receiving a Non-acknowledge only happens if the maximum global retries were exceeded. This can happen if, for example, the destination node has been destroyed, or otherwise has been removed from the network.

0x52:

Header + Payload data is: Originator ID(2) + Package Age(2) + Port(1) + Payload Data(max. 19)

0x53:

Header + Payload data is: Originator ID(2) + High Accuracy Package Age(4) + Port(1) + Payload Data(max. 15)

In general:

Packet Age is a measure from the point in time where the data was enqueued at the originator, to the time when the packet was delivered to the host application at the destination node.

For the UART interface, enqueue happens at the time the external controller has send the package through the UART interface, and dequeue happens at the time when the data is transmitted over the UART interface to the external controller. Packet Age has a resolution of 0.125 seconds, and as such the value received should be multiplied with 0.125 seconds to get a time indicator in seconds.

HAPA (High Accuracy Package Age) is used when higher resolution is required on the Package Age. When enabled, the Package Age field is replaced with a HAPA field. The HAPA has a resolution of $1/(2^{19})$ seconds.
See the USER GUIDE for more information about how to enable HAPA.

7.3 API for embedded controllers

A C90 compliant API is provided which simplifies the integration of the NCxxxx series modules, with an embedded controller.
When using the API, the user does not need to worry about how to generate UART frames, nor how to decode them.
The API is structured according to the illustration below:

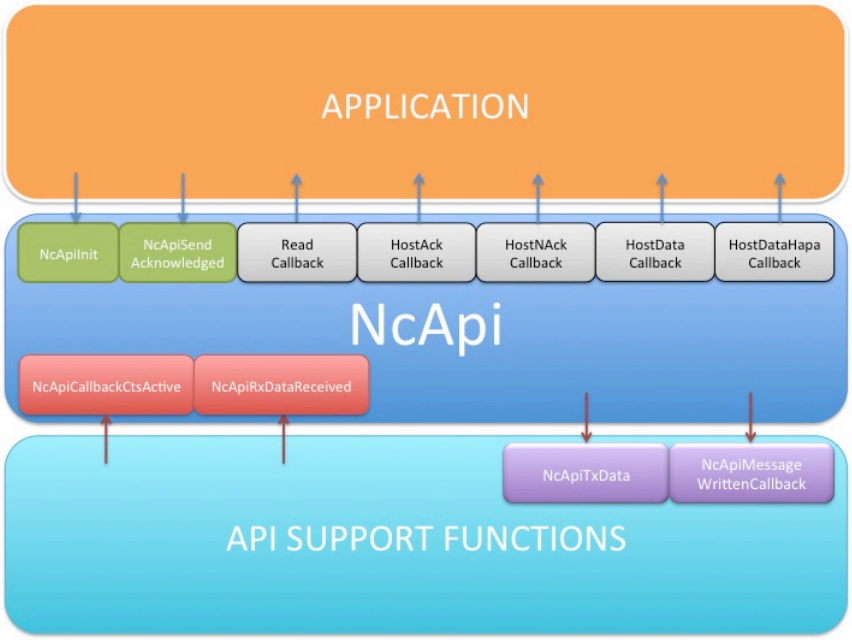

Figure 2 - NcApi structure

The API serves as layer, which implements high-level functions for sending and receiving payload data through the mesh network.
The API provides two basic functions to the application layer: NcApiInit & NcApiSendAcknowledged.

NcApiInit will have to be called once to initialize the API.

The **NcApiSendAcknowledged** function can be used to send payload data to another node inside the mesh network.

When data is received from the mesh network, a series of call-back functions will call the Application with the received data. Depending on what type of data is received, a type specific call-back is issued. It is optional for the application layer to register for the call-backs.

ReadCallback will deliver a byte array containing the raw UART frame received following the formats in section 7.2.2 "List of application data". It is normally not necessary to register for this callback, as there are other callbacks which are specific to the various types of application data. These are described here:

The application layer can register for **HostAckCallback**. The callback is issued when the module receives an ACK message for a previously send payload package. The callback function delivers the NEOCORTEC ID for the node, which has send, the ACK.

The application layer can register for **HostNAckCallback**. The callback is issued when the module receives an NAC message for a previously send payload package. The callback function delivers the NEOCORTEC ID for the node, which has send, the NACK.

The application layer can register for **HostDataCallback**. The callback is issued when the modules receives payload data from another module in the NEOCORTEC mesh network. The callback function delivers a pointer to a struct containing the following information: OriginatorID, Port number, PayloadLength, Packet Age & Payload.

The application layer can register for **HostDataHapaCallback**. The callback is issued when the modules receives payload data from another module in the NEOCORTEC mesh network which has been configured to use the High Precision Packet Age feature. The callback function delivers a pointer to a struct containing the following information: OriginatorID, Port number, PayloadLength, Packet Age & Payload.

To enable the API on a particular embedded controller, certain support functions are required, which are specific to the controller:

The API will call **NcApiTxData** every time data is to be written to the UART. The function is called with a pointer to the actual data, which is to be written. The function shall implement the necessary code required to output the data to the UART.

The API will call **NcApiMessageWrittenCallback** once a message has successfully been written in full to the module. This can be used by the application layer to check that a previous call to NcApiSendAcknowledged was completed successfully.

Similarly, there are functions inside the API, which has to be called when certain events occur:

NcApiCallbackCtsActive is a function inside the API, which shall be called each time the CTS line on the UART transitions to active (low) state. As such, an interrupt handler for the particular pin on the controller must be implemented, such that the API is "aware" of when CTS is active. There is no need to do anything when the CTS line transitions to passive (high).

NcApiRxDataReceived is a function inside the API, which shall be called each time a byte is received on the UART. The byte shall be included as an argument when the function is called.

Further details of the data structures as well as the functions, can be found in the Documentation folder, which is part of the API delivery. Open the "index.html" file in your preferred browser.

8 Communication Specification – System UART

The System UART serves three purposes; trace messages[4] for debugging purposes are being transmitted over the port while the Protocol stack is running. The second purpose is for setting various parameters, which configures the operating mode of the module. Finally it can be used to update the firmware of the module.

NEOCORTEC provides a PC tool that can be used to interface with the System UART port. The tool – NeoTools.ConfigApp – can be downloaded from www.neocortec.com. The tool is available both as a command line version and with a Windows GUI. The command line version is well suited for production testing/configuration.
Please review the documentation for the tool for more details on usage and options.

If it is desired to interface directly with the System API UART, more information about the messaging format etc. can be given upon request.

[4] Trace messages can be switched on or off by controlling the settings of the module.

9 Examples

9.1 Acknowledged Payload data transmission through the network

In the following example, payload data is send from a sensor node to a PC which is attached to another node in the NEOCORTEC network.

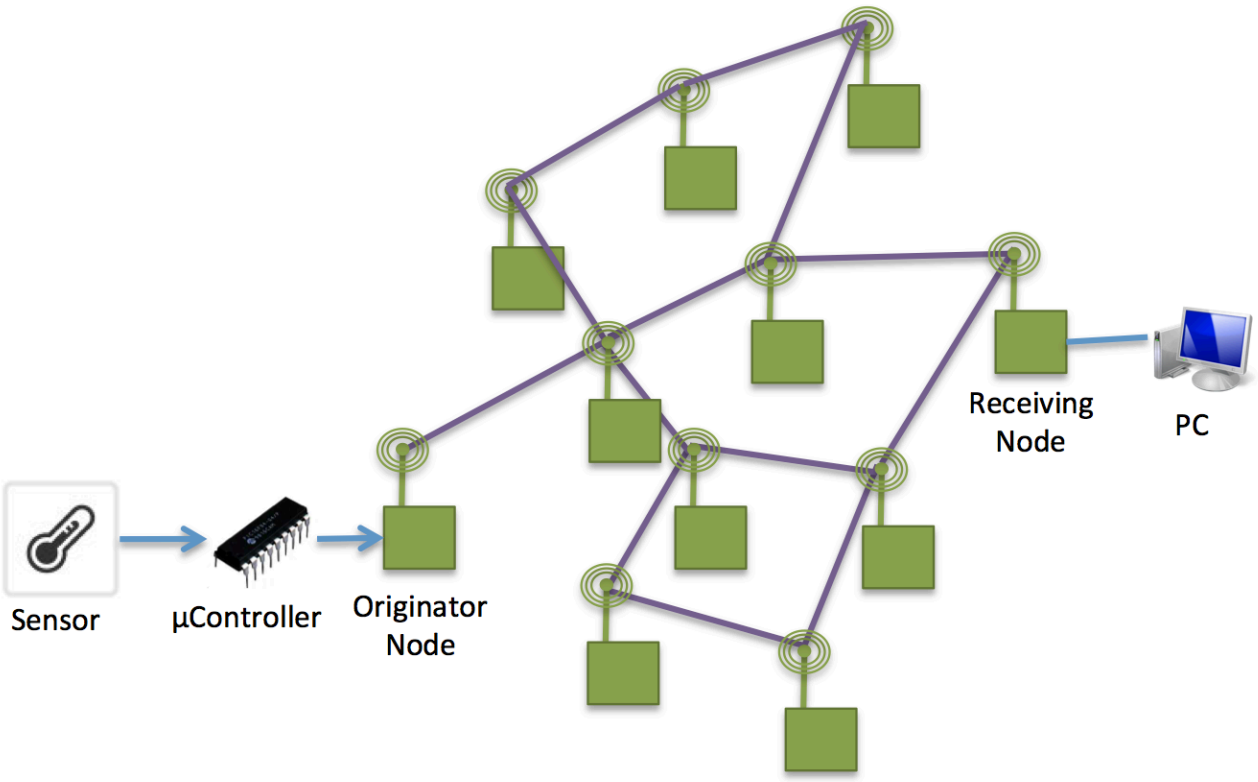

Figure 3 - Network example

In the example, the Originator node has the NEOCORTEC ID 0x00 20 and the Receiving node has the NEOCORTEC ID 0x00 2A. Port 0x00 is used throughout the example.
This example only uses the Application API.
When the uController has a sensor measurement that it wants to send to the PC, this is what happens:

1) uController sends a frame to the Originator node. The frame instructs the Originator node to send the sensor data to the Receiving node. The sensor data is 1 byte and has the value 0x23:

Section	Code	Length	UART Payload
Content	0x03	0x04	0x002A0023

2) The payload data arrives at the Receiving node and is delivered to the PC via the UART interface:

Section	Code	Length	UART Payload
Content	0x52	0x06	0x002000500023

At the same time, an acknowledge message is send from the Receiving node back to the Originator node

3) The acknowledge message arrives as the Originating node:

Section	Code	Length	UART Payload
Content	0x50	0x04	0x002A0055