neo.cortec

Integration Manual for NCxxxx series Modules

Doc Status:	Release
Doc version:	3.0
Date:	Mar 2019

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.
2.1	Added details for Wireless Encrypted Setup – WES
2.2	Fixed error in WES doc.
2.3	Unacknowledged messages have been introduced together with some other unimplemented messages, and the document has been through a general review and update.
2.4	Added missing command detail.
2.5	Fixed error in Wake commands
3.0	Major update for NeoMesh Release 1.4

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 (see User Guide for details). The duration of the individual Beacon Scans is dependent on configuration settings.
The power supply should stay within the specified 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 significantly 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 module and the external controller 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. The module will signal to the external controller whenever the module has data to send to the controller. The wakeup signal (nWU[1]) can not be used to detect the wake/sleep state of the module, as the module will not wake up the external controller unless it has data to send.

The external controller will have to be in active state whenever the wakeup signal 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 7 Serial UART Interface.

6 WES – Wireless Encrypted Setup

The nodes can be configured for a particular network using WES (see user guide document). The client mode of WES, can be started in two ways:

6.1 Auto WES after power up

If the node ID is configured to 0xFFFF, the node will start up in WES client mode after power up. See the user guide for further details.

6.2 Force WES client mode

Any node can be forced into WES client mode by pulling the WES Client Enable Pin[2] low for at least 2 seconds. See the user guide for further details.
When integrating the module in an embedded application, care must be taken not to connect the WES Client Enable Pin directly to a GPIO which may not be controlled fully during sleep or power up. This to ensure the module does not unintentionally enter WES Client Mode.

[1] See NCxxxx datasheets for details on the WakeUp signal.
[2] See NCxxxx datasheets for details on the WES Client Enable Pin.

7 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 NCxxxx Module. Provides functions for sending and receiving payload data through the NEOCORTEC wireless network as well as other local application layer related functions.
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. This interface is also used for firmware update of the module.

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

7.1 Physical interface

Figure 1 provides an overview of the physical serial interface.

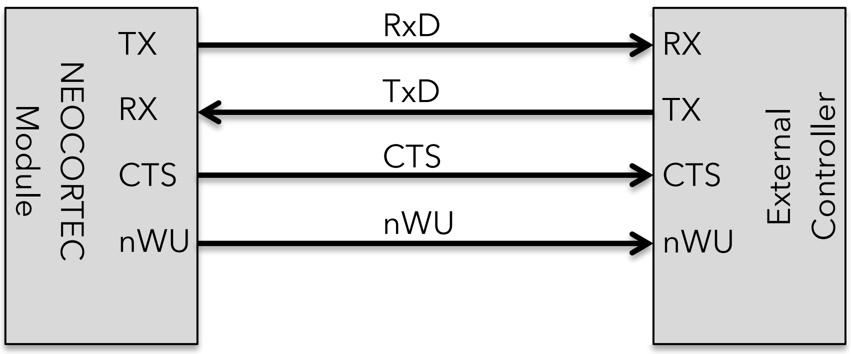

Figure 1 – Generic UART Interface overview

Description of the signals:

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	NEO	Indicates that the NEOCORTEC node is ready to accept commands (see below)
nWU	Wake Up	13	n/a	NEO	Indicates the activity mode of the NEOCORTEC node (sleep/active). Active low

nWU is used as a wake-up signal (as described earlier).

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

7.2 Easy Integration with PC

For easy integration with a PC, it is recommended to use a UART to USB converter. The TTL-232R-3V3[3] 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 receive data.

[3] 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.

8 Communication Specification – Application UART

8.1 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.

8.2 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 1.1.1 List of Application data.

8.2.1 Parameters used in the Commands and Application data

Destination ID and **Originator ID** are addresses of a NEOCORTEC module in the network for which the command data is intended or from which the application data originates.
An 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 (see User Guide for details). 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 0x0000 and 0xFFFF 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).

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 2^{-19} seconds.

See the USER GUIDE for more information about how to enable HAPA.

8.2.2 List of Commands

Overview of the Command message types that may be sent:

Type	Data (no of bytes)	Return	Description
0x02	Destination ID (2) Port (1) Sequence n (2) Payload (n)	0x56, 0x57	**Unacknowledged Packet.** Send a data packet that does not require an acknowledge. The maximum payload size is 21 bytes.
0x03	Destination ID (2) Port (1) Payload (n)	0x50, 0x51	**Acknowledged Packet.** Send a data packet that does require an acknowledge. The maximum payload size is 21 bytes.
0x08	-	0x58	**Node Info Request.** Send a request to the local NEOCORTEC node to return various information about the node.
0x09	-	0x59	**Neighbour List Request.**

			Send a request to the local NEOCORTEC node to return information about its neighbours.
0x0A	Destination ID (2) Command (1) Payload (n)	0x5A	**Network Command.** Send a network command to another node (Destination)
0x0B	-	n/a	**Enable SAPI on AAPI UART** This command stops the protocol, and enables the SAPI on the AAPI port. This can be used if the host controller needs to modify configuration parameters, but does not have access to the actual SAPI UART.
0x0C	-	0x5C	**Route info request** Queries the routing table of the module. This can be used to investigate if a certain Sink Node is present inside the network.
0x10	Command (1)	0x60[4]	**WES command.** Send a command to control the Wireless Encrypted Setup.
0x11	Unique ID (5) Node ID (2) Application Settings (24)	n/a	**WES Setup Response.** If a node setup request is accepted, the application can send this command with Unique ID, Node ID, and Application Settings as arguments.
0x20	Mode (1)	0x70[5]	**ALT Mode**. Sets the mode of the network, if the local NeoCortec Module is a ACM. Mode = 1 sets the network in Alternate Mode. Mode = 0 sets the network in Normal Mode.

[4] If the command is "Request Status"
[5] If the command is "Request Status"

			If Mode = 2, the module will respond with a status message indicating the current mode of the network.

0x02: Unacknowledged Packet
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. In addition a sequence number shall be given which is unique for the payload message. This is used to allow the destination node to filter our duplicate messages.
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.
0x03: Acknowledged Packet
Similar to the unacknowledged command, this command initiates the transmission of application payload data to another NEOCORTEC module inside the NEOCORTEC 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 section List of Application data).

0x0A: Network command
Network commands are commands which can be sent to another node in the mesh network. The following Network Commands are possible:

Command	Payload	Meaning
0x02	-	**Hibernate** Forces the destination node to hibernation state
0x03	type(1), n(1), [UID(5)]	**Wake / Unconfigure** n specifies the number of Wake Bursts. Type specifies the Wake Burst type: 0x00: Will wake certain UID 0x40: Will wake node with the same WES key 0x80: Will wake nodes with the same Network ID 0xC0: Will wake and unconfigure (set NodeID to 0xFFFF) a node with a certain UID. Note this message is transmitted through the WES channel, and requires the receiving node to be within radio range. Similarly, the receiving node does not have to be associated with the same NetworkID as the sending node.
0x05	-	**Force WES Mode**

		The command forces the destination node into WES mode, meaning that the NodeID of the receiving node will be set to 0xFFFF.
0x20	Mode (1)	**ALT Mode**. Sets the mode of the network, if the remote NeoCortec Module is a ACM. Mode = 1 sets the network in Alternate Mode. Mode = 0 sets the network in Normal Mode. If Mode = 2, the module will respond with a status message indicating the current mode of the network.

0x10: WES Command

This command controls the Wireless Encrypted Setup functionality in a Node that is announcing a network, or a node that wants to start looking for a new network to join. There are these arguments in the command, which has different functionality:

Argument	Meaning
0x00	Stop WES
0x01	Start WES server
0x02	Request WES status

0x11: WES Setup Response

When a “0x61: WES Setup Request” has been received from a Node that is trying to join the network, the application layer can add the node to the network using this command. The arguments to the command is the Node ID, which the joining Node shall be configured with, and the UID of the joining Node (as given by the “0x61: WES Setup Request”).

8.2.3 List of Application data

Overview of the Application message types that may be received:

Type	Data (no of bytes)	Description
0x50	Originator ID (2)	**Acknowledge** for previously sent packet 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	Originator ID (2)	**Non-Acknowledge** for previously sent packet 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	Originator ID (2) Package Age (2) Port (1) Payload (n)	**Host Data** Data received from another device, where acknowledge is required. The maximum payload size is 19 bytes.
0x53	Originator ID (2) HAPA (4) Port (1) Payload Data (n)	**Host Data HAPA** Data received from another device with HAPA (High Accuracy Package Age), where acknowledge is required. The maximum payload size is 15 bytes.
0x54	Originator ID (2) Package Age (2) Port (1) Application Sequence No (2) Payload (n)	**Host Data** Data received from another device, where acknowledge is NOT required. The maximum payload size is 19 bytes.
0x55	Originator ID (2) HAPA (4) Port (1) Application Sequence No (2) Payload Data (n)	**Host Data HAPA** Data received from another device with HAPA (High Accuracy Package Age), where acknowledge is NOT required. The maximum payload size is 15 bytes.

0x56	Destination ID (2) Sequence n (2)	**Unacknowledge Send** The unacknowledged package which was intended for the Destination ID with the Sequency n was successfully routed on to the next node in the network. Is does not guarantee delivery at destination.
0x57	Destination ID (2) Sequence n (2)	**Unacknowledge Dropped** The unacknowledged package which was intended for the Destination ID with the Sequency n was dropped due to TTL timeout. This happens if there is not route to the destination.
0x58	Node ID (2) Unique ID (5) Type (1)	**Node Info Reply** Returned information about the Unique ID and Type of the local NEOCORTEC node.
0x59	List of: Node ID (2) RSSI (1)	**Neighbour List Reply** A status list of up to 12 neighbours. Unused locations in the list are marked by a Node ID of 0xFFFF. The RSSI value is given in –dBm
0x5A	Node ID (2) Command (1) Payload Data (n)	**Network Command Reply** The reply from the NEOCORTEC node that previously has received a Network Command. The reply repeats the command in question and includes any related payload.
0x5C	Bit Mask (16)	**Route info response** The bit mask indicates if there is a route to a certain sink, which can be used to investigate if a certain sink node is present in the network. The bit mask shall be interpreted as follows: LSB in the first Byte corresponds to NodeID 0x0000 MSB in the final Byte corresponds to NodeID 0x007F
0x60	Status (1)	**WES Status** The response to a previously sent status request. Status can be either 0x00 (WES stopped) or 0x01 (WES server running)

0x61	Unique ID (5) Application Function Type (1)	**WES Setup Request.** When an unconfigured node is requesting to be set up for the network, this message is received. The UID of the node requesting the setup, is included with the request. The application layer must decide if the node is allowed to join the network or not. If the node is not allowed, this message can be ignored. If the node is allowed to join, the application layer must use the "0x11: WES Setup Response" command to provide the Node ID to the joining node. The protocol stack handles the rest of the process, and delivers all necessary setup information to the joining node, such that it can write the configuration parameters to the nonvolatile memory and reboot to start joining the network.
0x70	Status (1)	**ALT Mode status** The response to a previously sent status request. Status can be either 0x00 (Network in Normal mode) or 0x01 (Network in Alternate mode)

8.3 API for embedded controllers

A C90 standard C programming language compliant API is provided, which simplifies the integration of the NCxxxx series modules with an embedded controller.
It can be downloaded from www.neocortec.com.

The API is structured according to the illustration below:

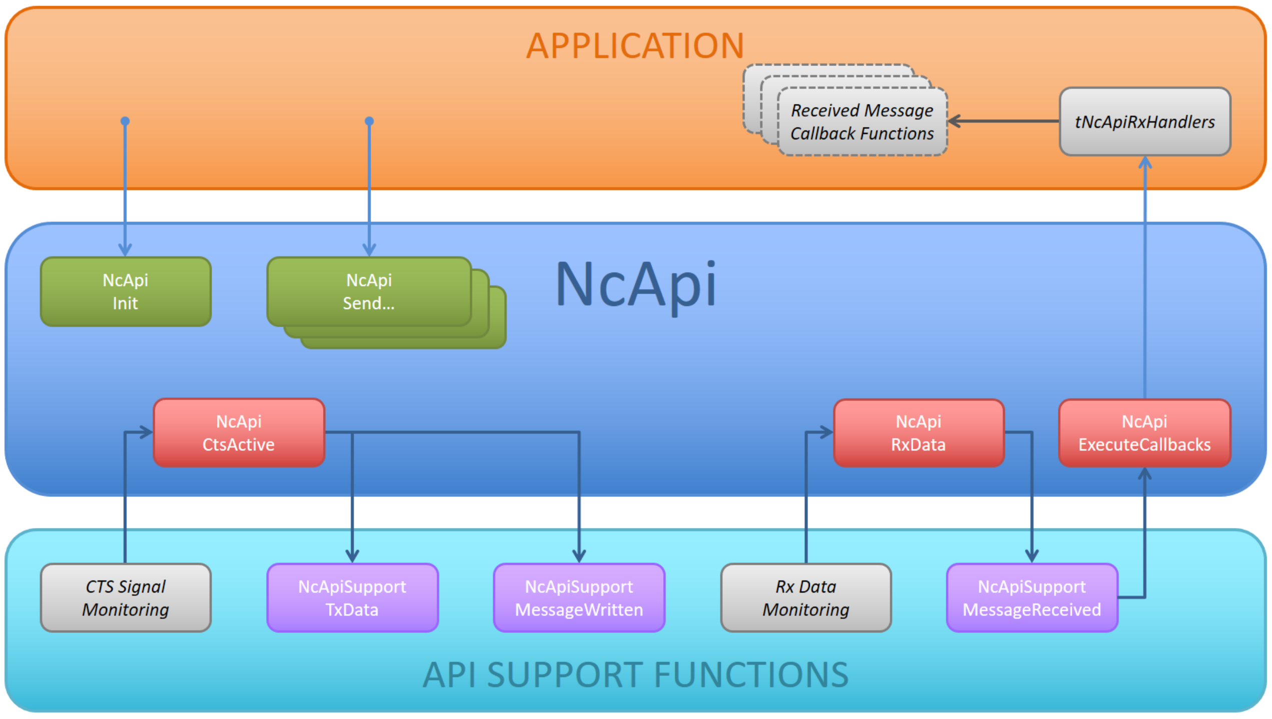

Figure 2 - NcApi structure

The API serves as a layer that implements high-level functions for sending and receiving payload data through the mesh network. When using the API, the user does not need to worry about how to generate UART frames, nor how to decode them.

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

For further information, please refer to the documentation that is embedded in the source code of the API, and to the documentation that comes with the downloaded API, which can be found in the “Documentation” folder. Open the “index.html” file in your preferred browser, or the “refman.pdf” using a PDF viewer.

9 Communication Specification – System UART

The System UART serves three purposes:

- Trace messages[6] for debugging purposes are being transmitted over the port while the Protocol stack is running.
- Setting various parameters, which configures the operating mode of the module.
- 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.

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

10 Examples

10.1 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.

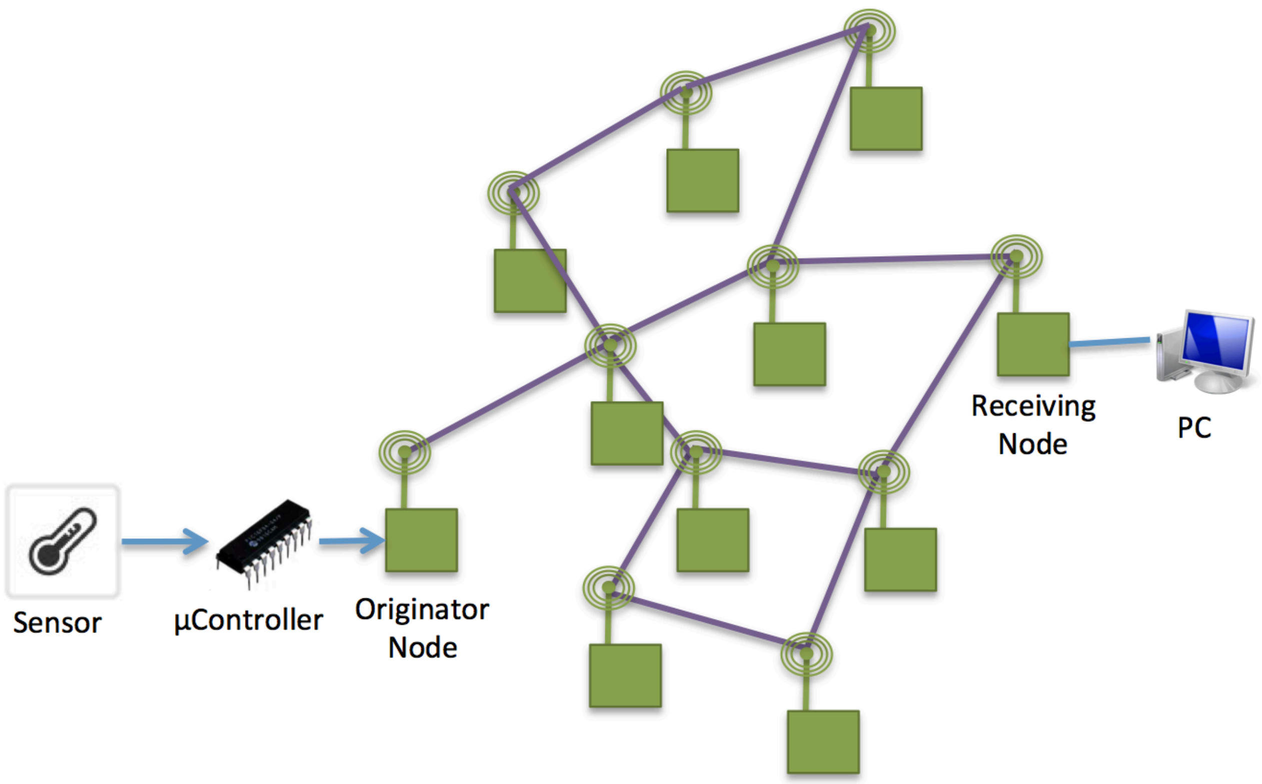

Figure 3 - Network example

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

10.2 Wireless Encrypted Setup

In the following example, a network consisting of a mesh of nodes, has one node which is announcing the network through the Wireless Encrypted Setup Channel. Another node, which is unconfigured, discovers the network, and sends a request to the announcing node to receive setup information for the particular network.

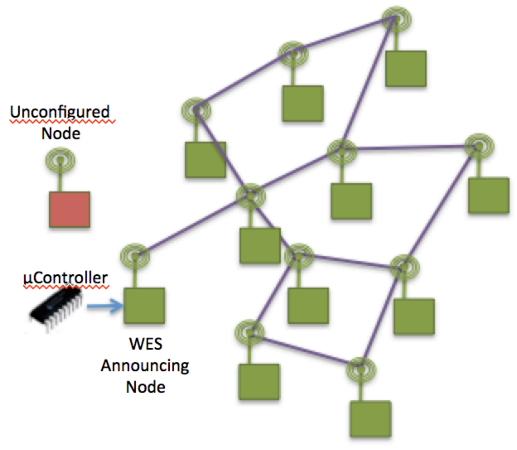

Figure 4 - WES Example

1) The unconfigured node is in WES client mode. Either the node comes directly from the factory, with the Node ID set to 0xFFFF, or the host controller on the device has started the WES client with the following command:

Section	Code	Length	UART Payload
Content	0x10	0x01	0x02

2) The WES Announcing Node is now put into WES server mode by issuing the following command from the uController (or Host in general) to the node:

Section	Code	Length	UART Payload
Content	0x10	0x01	0x01

3) When the unconfigured node discovers the announced network, it will send a setup request to the announcing node. The announcing node will send the following message to the uController:

Section	Code	Length	UART Payload
Content	0x61	0x05	0xnnnnnnnnnn

Where “nnnnnnnnnn” is the UID of the unconfigured node.

4) The application layer on the uController will now decide to accept the request, and it will assign the Node ID 0x002A to the unconfigured node by sending this command to the announcing node:

Section	Code	Length	UART Payload
Content	0x11	0x07	0xnnnnnnnnnn002A

Where “nnnnnnnnnn” is the UID of the unconfigured node.

5) The WES Announcing Node will now send the full configuration to the unconfigured node. The unconfigured node will now write the setting to its non volatile memory, and reboot in order for the changes to take effect.

6) The application layer at the WES Announcing Node can decide to stop the WES server by sending this command:

Section	Code	Length	UART Payload
Content	0x10	0x01	0x00