neo.cortec

User Guide

Doc Status:	Released
Doc version:	1.7
Date:	May 2016

Table of Contents

1 Document revisions

Document version	Details
1.2	Initial release
1.3	HAPA details added
1.4	Generic Application added
1.5	Wireless Encrypted Setup added
1.6	Minor typos fixed
1.7	FHSS description updated

2 Introduction

This document introduces the reader to the NEOCORTEC Wireless Ad-Hoc Mesh Networking Technology. It introduces the core features of the technology.
The documents also detail the configuration parameters, and the tools available.

3 NEOCORTEC Wireless Ad-Hoc MESH-networking

3.1 Overview

The NEOCORTEC wireless mesh network protocol is capable of building large networks with thousands and thousand of nodes. Each individual node is a fully capable autonomous unit, which can act as a source for data, a router of data or a destination for data. Network creation is done without the involvement of a central coordinator device. Neighbour relationships are formed locally. This means that networks are created at the same pace regardless of size.
Payload data is routed through the network along the fastest path from source to sink. Routes are created *while* the payload data is travelling through the network, and not *before* as with other technologies. This means that nodes can move around, the topology can change without impacting routing of data. This is a patented technology exclusively used by NEOCORTEC.

Apart from the core functionality of network creation and routing, the NEOCORTEC wireless mesh network technology boasts a number of other features related to security and reliability.

3.2 Wireless Encrypted Setup

A NEOCORTEC node is pre-configured for a particular network. The configuration parameters are written to the module through the System API.
To ease the deployment of new nodes, and to allow end-users to add new nodes to an existing network without having to write settings to the module manually, Wireless Encrypted Setup (WES) is provided as a functionality which allows unconfigured modules/nodes to be setup for an existing network wirelessly by having another node who is already part of the existing network to announce the network through a Encrypted WES channel.
The Encryption Key for the WES Channel is separate from the Network ID (also an encryption key), and is usually the same for a group or a category of modules/nodes.
The WES process requires one node in an already existing network to be setup as a WES Server. This is achieved through sending a set of commands through the application API (see integration manual). Once the node is setup as a WES Server, it begins to send out WES Beacons on the WES Channel, which is encrypted with the WES key. This means that nodes that do not have the correct WES key cannot be setup for the particular network.
If an unconfigured node/module is put into WES Client mode (see integration manual), it will start looking for a WES beacon. If it receives the WES Beacon, it will send a setup request, along with its UID to the WES server node.

The WES server node will then send a message to the application layer (through the application API) with the UID such that the application can decide if the node who wants to be setup is allowed or not. If the application decides to allow the node to join, it shall provide the Node ID, which the joining node is supposed to have. This means that the application layer needs to make sure the nodes inside the network has unique Node ID's.
Once the application sends the Node ID to the WES server, the WES server will send the full set of configuration parameters to the joining node. Hereafter the unconfigured node will write the parameters to the non-volatile memory, reboot and become part of the network.

3.3 Network Initialization

The wireless mesh network is created automatically when the participating nodes are powered up. Each node will go through the same sequence after power up: Initially it will scan for an already active network to which it is allowed to join. If it detects an operating network, it will join it by start listening to the Schedule Data transmissions from the neighbouring nodes.
As the join process happens locally, and not via a network manager several hops away, the network will initiate fast – typically within 5 to 20 seconds depending on the density, and on the settings of the network parameters (see section 3.8 Configurable Parameters). The overall network size does not impact the pace of which the network is created.

3.4 Routing of Payload Data

Payload data can be sent to sink nodes present in the network. A sink node is characterized by having a Node ID between 1 and 127. As a background activity, routes to the sink nodes are being generated all the time without any involvement from the Application layer. This means that when the application layer on any node inside the network wants to transmit payload data to a given sink node, all it has to do is to issue a Send command to the API, and the NEOCORTEC module will make sure the payload data is delivered at the intended destination.

Routes are being created with the unique Speed Routing algorithm. The algorithm identifies the fastest path to the destination, taking into account paths with radio noise which would potentially slow down the delivery of the payload data.
Because the routes are being generated all the time as a background task, changes in topology (ie. nodes moving around – either nodes on-route or even the destination node) do not impact the delivery of the payload data. Because the routes are being maintained all the time, there is no such thing as a "bad route" which needs to be healed.

3.5 Security

All communication, not only payload data but also network communication, is encrypted using AES128. The key for the encryption is the same as the Network ID (see section 3.8 Configurable Parameters). This means that nodes that do not have the correct Network ID, will not be able to decode transmissions from a network for which it is not approved.
In order to keep the network secure, the Network ID must be kept secret for a particular installation.

In addition to the encryption, all payload data exchange between a originating node, and a sink node is governed by an end-to-end challenge/response mechanism. This means that each new message exchange will be subject to a new challenge with a new associated response. Since the challenge/response messaging is included in the encrypted communication, the data exchanged will look completely random from transmission to transmission, even if it is the same payload that is being exchanged. This prevents playback attacks[1] and improves the overall security level of the communication system dramatically.

NOTE: The synchronisation of the challenge/response mechanism between any source and a particular sink happens at the first payload data exchange. This means that the first payload data transmission from a source to a sink will fail due to the wrong response included in the datagram. However, the protocol will automatically retry the transmission with the correct challenge. This happens transparently from the application layer. The only impact is that the first payload data exchange will take twice as long time as normally due to the retransmission.

3.6 Reliability

The delivery of payload data to the destination is very reliable. There are a number of features build into the core of the protocol, which will help to increase the likelihood of payload data being successfully delivered to the destination:

End-to-end Ack/Nack: When the application sends payload data to a sink node, all it has to do is specify the Node ID of the intended sink node along with the actual payload data. The protocol will then automatically route the data to the destination. Once the data has been delivered, an acknowledge packet will be routed back to the source and will be delivered at the application layer.

[1] See http://encyclopedia2.thefreedictionary.com/Playback+attack for explanation

If no acknowledgement has been received within the configured timeout window, the protocol will automatically retry the packet a configurable number of times, say three times for this discussion (see Configuration Parameters). If the acknowledge is not received during the retries, a non-acknowledge (Nack) message will be delivered to the Application layer. This could happen, for example if the destination node was removed from the network, and so, this mechanism allows the application to take action, according to the needs of the application.

Node-to-node Ack/Nack: When neighbouring nodes communicate, the receiver of payload data responds with an Acknowledge message if the CRC calculation on the received data is successful. The sending node will retry the packet until either an Acknowledge has been received, or until the packet is too old according to the TTL (Time To Live) parameter. The sending node will always retry the packet to the neighbour, which is currently the one closest to the destination. Should the missing Acknowledgements be a result of the given neighbour no longer being within radio range, the neighbour will automatically be removed from the neighbour list after a configurable number of times, say three times for this discussion (see Configuration Parameters) missing scheduled data transmissions. Once this has happened, retries of the packet will be done with the neighbour now closest to the destination.

Theoretically there can be a situation where the data was received correctly at the neighbour, but the Acknowledge was not received by the sending node. In this situation, the neighbour will route the received packet onwards to the next node on route to the destination. At the same time, the sending node will retry the packet. This retry is now a duplicate of the previous packet. The receiving node will also route this duplicate onwards to the next node on route to the destination. Once the packets reach the destination, the first packet will be acknowledged, and a Ack will be routed back to the Originator. The second packet will be non-acknowledged due to the challenge-response authentication failing. The Nack will be routed back to the Originator, however the Nack will be ignored, because an Ack for the same packet was already received.

FHSS – Frequency Hopping Spread Spectrum: The protocol uses FHSS to avoid noise and to ensure that the radio communication is not blocked by problems which are typically seen when operating on a fixed frequency such as fading and blocking due to other radio communication equipment operating at the same frequency.
Each time a node is sending its Scheduled Data Transmission (either with or without payload) it does so at the next channel in the hopping list. The hopping list consists of 15 logical channels, each of which can be assigned to a physical radio frequency. This means that the network can be configured to use an arbitrary range of radio frequency

channels, either evenly spread in the frequency band, or grouped in sections to avoid certain parts of the band. Channel 16 is the beacon channel, and is not used as part of the hopping scheme.

3.7 Power Consumption

The NEOCORTEC protocol has been designed with overall power consumption as a key parameter. The operating mode of the nodes, is such that they are spending most of the time in sleep mode, where the current consumption is ultra low. The module wakes up in these situations:

- Transmit of own Scheduled Data – either with or without payload data
- Receive of Scheduled Data from each of its neighbours
- Transmit/receive of Beacon
- Beacon Scan
- At least every 2 seconds to maintain timers internally

The exchange of Scheduled Data is done in a very short time – typically less than 2ms.
The timer maintenance is done in less than 0.5ms.
Beacon transmit/receive event is typically done in less than 5ms.
Beacon scans are the most power hungry, and are limited to occur only initially, and then very rarely once the network is formed to allow dislocated nets to join. The rate of beacon scans during normal operation is configurable. A Beacon scan runs for a full Beacon period.
Since the rate of the above mentions events can be configured, the overall average power consumption of the module depends on the theses settings.
The NEOCORTEC Config Application can be used to calculate the expected average current consumption based on the actual configurations for the node.

3.8 Configurable Parameters

In the following, all parameters, which affect the performance of the network, is listed and explained.
NOTE: All parameters must be set to the same setting for all nodes in a given network. If parameters are not set to the same for all nodes, network performance/functionality will be impacted, and in general will be unpredictable.

3.8.1 Password Level 1..3

Specifies the passwords for logging into the System API. Each level has different privileges for viewing/modifying various settings.

Default passwords are as follows:

- Level 1: Lvl10
- Level 2: Lvl20
- Level 3: Lvl30

The passwords can be modified by logging on with the same or higher level. Ie. logging on with Level 2 allows modification of the password for Level 2 and Level 3, but not Level1.

3.8.2 Node ID

Specifies the Node ID of the node. Legal values are 0x0001 to 0xFFFE. Nodes in the range 0x0001 … 0x007F are sink nodes, and can be directly addressed by other nodes in the network.
The value 0xFFFF is special, and is can be used for an unconfigured Node. A node with this value, will automatically enter into WES Client mode, and start scanning for a network which it can be set up for using the WES feature.

3.8.3 Network ID

Specifies the ID of the network, which the node is configured for. The Network ID is also the key for the AES128 encryption used for all radio communication.
The Network ID is 128bit long.

3.8.4 WES Key

Specifies the AES 128 encryption key for the WES Channel.

3.8.5 Scheduled Data Rate

The rate in seconds for transmission of Scheduled Data. The setting controls the overall pace of the network; It affects how fast networks are created, and it affects how fast routes are generated, and it affects how fast payload data can move through the network[2].
The setting also directly affects the average current consumption of the module.

[2] This is because payload data is transmitted as part of the normal Scheduled Data transmission, if payload data is enqueued for transmission.

The parameter can be set to the following values: 1..30 [s].

3.8.6 Beacon Rate

The rate in seconds for transmission of Beacons. The setting controls how fast new nodes can detect a network which is already running, and it controls how fast two (or more) disjoined networks can discover each other.
The setting also directly affects the average current consumption of the module.

The parameter can be set to the following values: 1..30 [s].

3.8.7 Beacon Full Scan Rate

The rate given in a multiple of Beacon Periods, for which the protocol will perform a full beacon scan while the node is already participating in a network. The setting controls how fast two (or more) disjoined nets can discover each other. It also controls how fast a node which has been out of range, or is not part of a network in general, can discover networks which are already in operation.

The parameter can be set to the following values: 200..60.000. To get the rate in seconds, the values should be multiplied by the Beacon Rate.

3.8.8 Beacon Full Scan Rate Initial

The rate of which full beacon scans are performed after power up. The setting is similar to Beacon Full Scan Rate, however this initial value is only used until the number of full beacon scans exceeds the Beacon Full Scan Count Initial parameter.

The parameter can be set to the following values: 2..20. To get the rate in seconds, the values should be multiplied by the Beacon Rate.

3.8.9 Beacon Full Scan Count Initial

Upon power up of the node, the protocol will start searching for already active networks. It will do so by performing a series of full beacon scans. The Beacon Full Scan Count Initial parameter controls how many full scans are performed before the protocol enters into normal operation.
The parameter can be anything between 3 and 255. However, typical values would be from 4 to 16.

3.8.10 Beacon TX Initial Hold Down Time

While doing initial beacon full scans, the beacon transmission is suppressed, until receiving, and synchronized to another nodes beacon, or until the beacon hold-down time is over. This to avoid collisions of beacon transmission in dense networks.
The parameter is a multiple of the Beacon Rate. Valid values are 2..4.

3.8.11 Scheduled Data Receive Retry Limit

If a receive of scheduled data from a neighbour fails, a number of retries on subsequent transmissions are done. If the number of retries exceeds the limit set with this parameter, the neighbour is removed from the neighbour list.
Valid values for this parameter is: 1..15. Typical values are in the range 2..10.

3.8.12 Network Maximum Roundtrip Time

Maximum network roundtrip time, or network roundtrip for short, is the time it takes a packet to get to a destination and the destination to return an ACK / NAK to originator. The roundtrip also accounts for queue delays, and local retries.

Network roundtrip time is highly dependent on network topology, and may change for networks with the same n number of nodes but different topologies by a factor of 10 or even 100 or more for large n.
As a rule of thunb, the max. network roundtrip time should be greater than about 20% of the average roundtrip time from a given topology.

Roundtrip time resolution is 32 seconds, and valid values are [32..992] seconds, parameter setting [1..31].

Internally, originator application data is time stamped with a retry or resend timestamp, which is used to control retries. After each successful (locally acknowledged) retry, the resend timestamp is reset.

A packet will be resent when the time since last transmission is in the interval of [network roundtrip] sec. to [network roundtrip]+64 sec.
A packet will only be resent when the packet age is less than TTL - [network roundtrip]

If for example the network roundtrip is 32 seconds (network roundtrip value is 1), and TTL is 96 seconds (TTL value is 3), the packet will perhaps be retried once, if this retry happens to fall between 32 to 64 seconds.
The retry period is between 32 and 96 seconds, so to make sure retry happens, the TTL must have a value at least 3 units (96 seconds) higher than network roundtrip.

See section 3.8.13 for more details on the Payload Data Time To Live – TTL parameter.

3.8.13 Payload Data Time to Live – TTL

TTL is the maximum age a globally retried packet can have. When a packet reaches this age, it is removed.
Other data types, such as ACK and NAK are removed when their age reaches [Network Maxìmum Roundtrip Rime] / 2.
In general; TTL = [max. network roundtrip time] * R + 3, where R is the number of desired global retries.

TTL time resolution is 32 seconds, and valid values are [32..992] seconds, parameter setting [1..31].

3.8.14 Routing Field Strength Threshold

For a route to be accepted, the field strength has to be above a certain limit, otherwise, the node will choose a router (node) closer by, as it is not known if there is sufficient transmitter power to reliably send to the station. It is thus best to keep the station in the neighbour list, but avoid routing. Especially true for mobile stations, where the signal may change to worse after routing info has been updated, perhaps rejecting routers with stronger signal in the process.

3.8.15 TX Output Power

The RF transmitter output power. Possible values depend on the module version.

3.8.16 Node Inclusion Increment Speed

Node inclusion speed determines how fast a node will try to acquire new neighbours. A high setting will make the node connect to neighbours more readily, but note that if this setting is too high, neighbour connections may happen too quickly: A node will connect to neighbours based on a contention system, and if many neighbours are present, this setting must be suitably low, not to cause too much contention. Settings can be tuned by experiment for a given network, but most likely the default settings will be appropriate.

3.8.17 Node Inclusion Hysteresis

Node Inclusion Hysteresis determines how many neighbours a node keeps track of as a minimum before adding more nodes as neighbours. The parameter can be used in tuning of the performance of a network wherein the nodes are mobile. This parameter works in conjunction with the parameter of "Node Inclusion Increment Speed" above, and is considered an advanced subject not discussed further here. For most networks the default value will perform well.

3.8.18 FHSS Channel Mapping

The frequency hopping channels can be configured freely to a range of RF Frequencies. 16 logical channels can be mapped to a range of 251 RF Frequencies.
Resolution as well as highest and lowest frequencies depends on the frequency band and module version. The possible frequencies are also limited by regulatory requirements. The configuration tool limits the possible frequencies in accordance with this.
Channel 1 though Channel 15 is used for Scheduled Data transmissions. The hopping sequence is 1,2,3,4,5,6,7,8,9,10,11,12,13,14,15. If a pseudo random hopping sequence is desired, the mapping must be done accordingly.
Channel 16 is used for Beacon transmissions.

3.8.19 High Accuracy Package Age - HAPA

When enabled, the package age is given with a greater accuracy than normal. This is useful in applications where network wide synchronisation is needed with millisecond accuracy.
When the HAPA setting is 0, HAPA is disabled, and package age is given with a 0.125 second resolution. In this configuration package age is 2 bytes.
When the HAPA setting is 1, HAPA is enabled, and package age is given with a $1/(2^{19})$ second resolution. In this configuration package age is 4 bytes.

HAPA is a global setting, which means that all nodes inside the network must have the same setting for HAPA.

When HAPA is enabled, it steals 4 bytes from the payload data field, such that 15 bytes are available for payload data, as opposed to 19 bytes when HAPA is disabled.

4 Application

4.1 Overview

The NEOCORTEC protocol is designed such that zero involvement at the Application layer is needed in order for the network to function. The network will be created, and routes will be generated in the background as soon as the node is powered up. The application layer only has to interface with the node when payload data is being transmitted or received. As such, there are no requirements to have an external host controller connected to the module in order for the network to operate.

The module can be configured for a range of different application options. The configuration is controlled through a setting, which can be set up with the config tool. The parameter is named "Generic Application" and can be accessed by logging on the module with level 1 password. When the parameter is selected, a series of application types are available to choose from:

Application Type	Description
UART	UART based interface for sending and receiving payload data using an external controller
GPIO	Highly configurable application for Digital Inputs and Output, Digital counter, Analog inputs.
HTU21D	Direct support for the integrated I2C Temperature & Humidity sensor HTU21D from Measurement Specialties Inc.
RSSI	Application which sends the list of neighbours along with the RSSI level for each neighbour

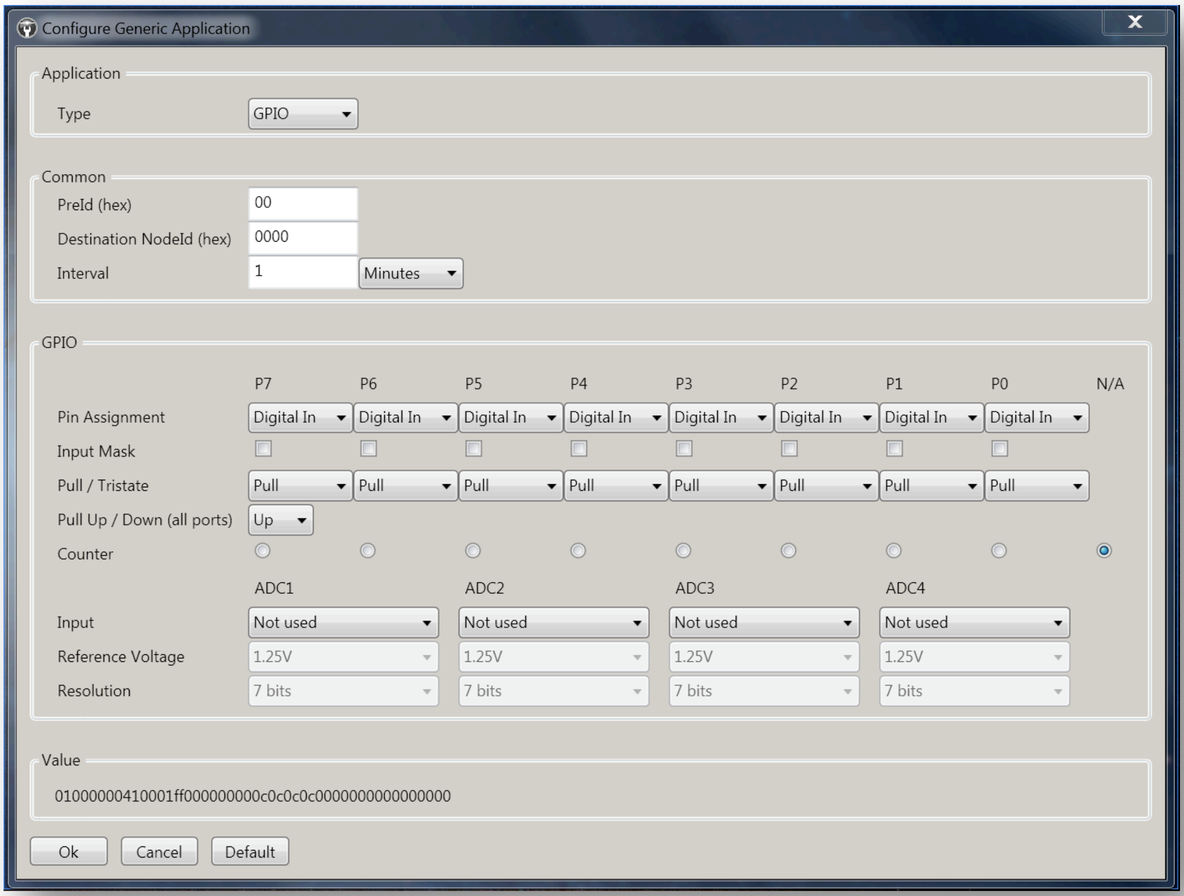
The details of each application type are described in the following sections.

4.2 UART

The NEOCORTEC NCxxxx has a Serial API that enables an external host controller to easily interface with the module. Please refer to the "Integration Manual for NCxxxx series Modules" document for further details.

4.3 GPIO

When selecting this application type, a settings window is brought up which allows the user to configure a range of parameters specific to this application:


Configure Generic Application

Application

Type: GPIO

Common

PreId (hex): 00

Destination NodeId (hex): 0000

Interval: 1 Minutes

GPIO

	P7	P6	P5	P4	P3	P2	P1	P0	N/A
Pin Assignment	Digital In	Digital In	Digital In	Digital In	Digital In	Digital In	Digital In	Digital In	
Input Mask	☐	☐	☐	☐	☐	☐	☐	☐	
Pull / Tristate	Pull	Pull	Pull	Pull	Pull	Pull	Pull	Pull	
Pull Up / Down (all ports)	Up								
Counter	○	○	○	○	○	○	○	○	●

	ADC1	ADC2	ADC3	ADC4
Input	Not used	Not used	Not used	Not used
Reference Voltage	1.25V	1.25V	1.25V	1.25V
Resolution	7 bits	7 bits	7 bits	7 bits

Value

01000000410001ff000000000c0c0c0c0000000000000000

In the Common section of the window, the user can input 3 parameters:

PreId: This value can be freely selected, and will be prepended to the payload package. This allows the receiver of the payload package to identify what configuration is used for this particular package, and as such be able to interpret the data.

Destination NodeId: The 4 byte NEOCORTEC address of the destination node for the payload packages.

Interval: The interval with which the application transmits the payload. Can be selected in units of seconds, minutes and hours, with the fastest interval being 10 seconds, and the slowest 24 hours. The interval should be set in consideration for the Scheduled Data Rate setting.

In the GPIO section of the window, the user can select what data gets transmitted in the payload package.
The following parameters can be set:

Pin assignment: For each pin, it can be configured to be either; Digital Input (default), Digital Output or Analog Input.

Input Mask: If enabled for a particular pin, the digital state of the pin will be transmitted in the payload package.

Pull / Tri-State: Each pin can be either Tri-Stated, or pulled up or down. If Pull is selected, the parameter Pull Up / Pull Down will decide how the pin is pulled.

Pull Up / Pull Down: A setting common for all pins (which are configured to be Digital Input), where it is decided if the Pull is up or down when Pull is selected as described in the previous item.

Counter: One pin can be configured as a Digital counter. It will trigger on a falling edge, and the returned value will be number of falling edge transitions since the previous payload package. Only one pin can be configured as a Counter.

The remaining parameters control the Analog to Digital Converters. Overall there are 4 ADCs (ADC1 through ADC4) that can be configured individually. Here are the settings that can be set for each ADC:

Input: Select the input to the ADC. It can be any of the pins, which are configured as an Analog Input either single ended, or differential. The input can also be the Internal Temperature Sensor, or the supply voltage (VDD/3 – meaning the supply voltage divided by 3).

Reference Voltage: The reference voltage used for the ADC. Options are: Internal 1.25V reference, external voltage from pin7, Vdd and external differential from Pin6-Pin7.

Resolution: The ADC resolution can be selected to be 7,9, 10 or 12 bits.

4.4 HTU21D

This application is capable of reading temperature and humidity using the HTU21D sensor.
The parameters that can be configured are as follows:

PreId: This value can be freely selected, and will be pre-pended the payload package. This allows the receiver of the payload package to identify what configuration is used for this particular package, and as such be able to interpret the data.

Destination NodeId: The 4 byte NEOCORTEC address of the destination node for the payload packages.

Interval: The interval with which the application transmits the payload. Can be selected in units of seconds, minutes and hours, with the fastest interval being 10 seconds, and the slowest 24 hours.

4.5 RSSI

This application sends payload data that include a list of neighboring nodes along with the RSSI level for each neighbor.

The parameters that can be configured are as follows:

PreId: This value can be freely selected, and will be pre-pended the payload package. This allows the receiver of the payload package to identify what configuration is used for this particular package, and as such be able to interpret the data.

Destination NodeId: The 4 byte NEOCORTEC address of the destination node for the payload packages.

Interval: The interval with which the application transmits the payload. Can be selected in units of seconds, minutes and hours, with the fastest interval being 10 seconds, and the slowest 24 hours.

4.6 *Payload Package format*

When using the applications that generate data on the module, and not through the UART interface, the Payload package is structured in different ways depending on the specific application:

4.6.1 GPIO

Field	PreId	Pin Changed	Pin State	Counter	ADC1	ADC2	ADC3	ADC4
Length	1 byte	1 byte	1 byte	2 bytes	2 bytes	2 bytes	2 bytes	2 bytes

PreId: The byte configured as part of the setting

Pin Changed: A byte where each bit corresponds to a pin (MSB = P7, LSB = P0), and indicates if the pin has changed state during the sample period. It is trigged by a falling edge. A 1 means that the pin has changed state, and 0 means no change (or that the pin has changed from Low to High).

Pin State: A byte where each bit corresponds to a pin (MSB = P7, LSB = P0), and indicates the state of the particular pin when the payload package was generated. A 1 means that the pin is High, and a 0 means the pin is Low.

Counter: Number of falling edge transitions on the pin selected as Counter during the sample interval (10s to 24h).

ADCx: A 2s complement value, being either 7,9,10 or 12 bits depending on the configuration. The value is aligned towards MSB of the ADCx bytes. For example a 12bit ADC reading will look like this:

ADC BYTES	Byte 0								Byte 1							
	msb							lsb	msb							lsb
ADC Bits	msb	-	-	-	-	-	-	-	-	-	-	lsb	X	X	X	X

'X' means don't care.

4.6.1.1 Internal Temperature Sensor

If the internal temperature sensor is used, the conversion from ADC value to Temperature should be done according to this formula:

$$T = \frac{(Output\ Voltage[mV] - offset[mV])}{coefficient[\frac{mV}{°C}]}$$

Where offset and coefficient can be found from the table below:

	Offset	Coefficient
NC2400	2.43mV/°C	750mV
NC1000/NC0400	2.54 mV/°C	755mV

4.6.2 HTU21D

Field	PreId	Temperature	Humidity
Length	1 byte	2 bytes	2 bytes

PreId: The byte configured as part of the setting

Temperature: is reported as a 14-bit value, aligned towards msb. Before converting the value to a temperature reading, the 2 least significant bits should be set to 0.

Humidity: is reported as a 12-bit value, aligned towards msb. Before converting the value to a humidity reading, the 4 least significant bits should be set to 0.

Please refer to the datasheet of the HTU21D for details on how to convert the values into temperature and humidity.

4.6.3 RSSI

Field	PreId	Neighbor 1		Neighbor 2		Neighbor 3		Neighbor 4		Neighbor 5	
		NodeId	RSSI	NodeId	RSSI	NodeId	RSSI	NodeId	RSSI	NodeId	RSSI
Length	1 byte	2 bytes	1 byte	2 bytes	1 byte	2 bytes	1 byte	2 bytes	1 byte	2 bytes	1 byte

In each transmission, there will be up to 5 neighbors being reported. In situations where there are more than 5 neighbors in the list, the neighbors will be reported cyclically in chunks of 5 neighbors.

The RSSI value is given in –dBm.

5 NEOCORTEC Modules

5.1 Common Information

The NEOCORTEC modules are designed to be easily integrated into the final device, and to ensure fast time to market.
The modules are certified for the major markets (CE, FCC, IC), which means that the final product can use the modular certification. This is a significant cost saver in the product development of the final device, and reduces the overall time for developing the product.
The modules are generally available with u.fl. connector for external antenna, and are an LGA type.

5.2 NC2400c

The NC2400c module is designed for the 2,4GHz ISM band, and is certified as a module for compliance with the requirements for FCC, CE and IC.
Please refer to the datasheet for further details.

5.3 NC1000c

The NC1000c module is designed for the 868MHz SRD band (EU and others) and the 915MHz ISM band (US and others). It is certified as a module for compliance with the requirements for FCC, CE and IC.
Please refer to the datasheet for further details.

5.4 NC0400c

The NC0400c module is designed for the 433MHz SRD band, and is certified as a module for compliance with the requirements for CE.
Please refer to the datasheet for further details.

6 NEOCORTEC PC Tools

6.1 System Requirements

The NEOCORTEC PC tools are designed for Microsoft Windows 7 or newer operating systems. The tools can be used to communicate with the NEOCORTEC module either stand alone, or as part of the Evaluation Kit. The communication interface is Serial UART, which can be either a native COM port, or (more likely) a USB-UART converter.

6.2 Getting Started

The NEOCORTEC PC Tools can be obtained either from the NEOCORTEC website (www.neocortec.com) or as part of the Evaluation Kit, where they are available on the included USB memory stick.
On the USB stick there is a launcher application, which can be used to easily start using the tools:

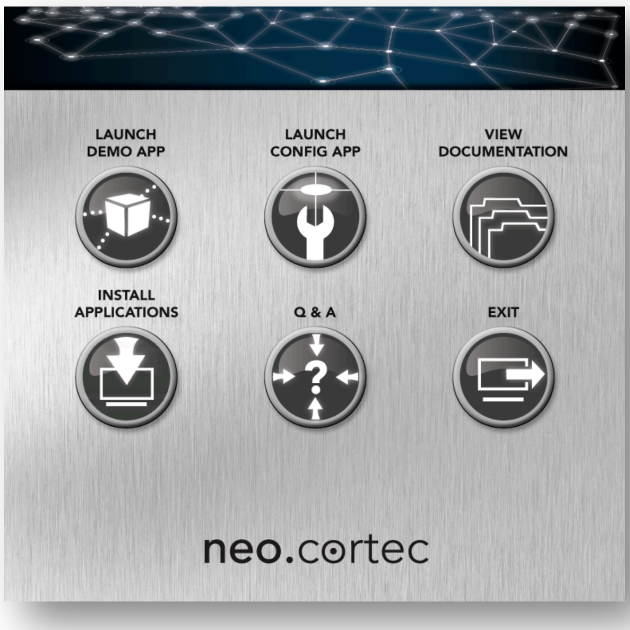


Figure 1 - Launcher application

6.3 Config App

6.3.1 Introduction

The config app can be used for several things. It has 3 basic features (parameter change, FW upload & developer functions). These functions will be explained in detail in the following sections.

6.3.2 Configuration Values

The pane for modifying the configurations values look like this:

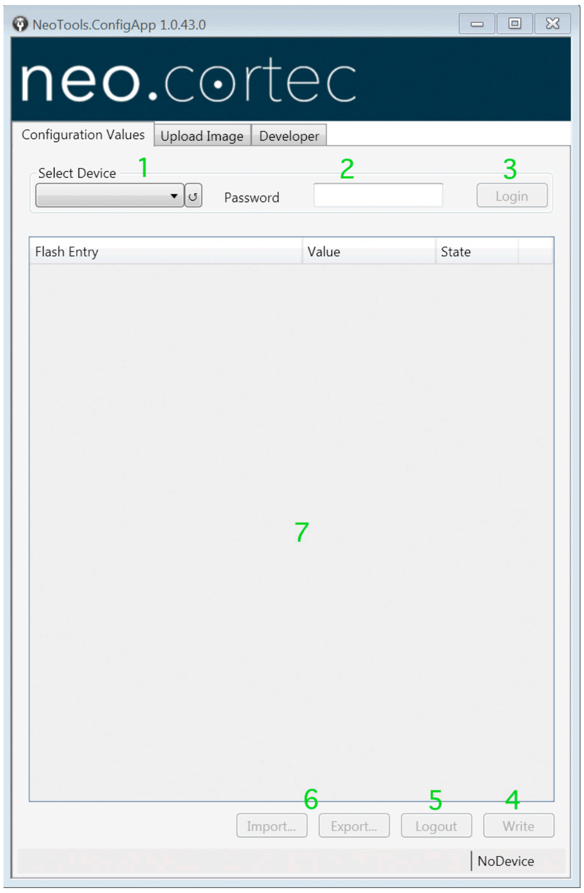


Figure 2 - Configuration values

The areas identified with green numbers, will be explained in the following:

1. Select the com-port to which the System API of the modules is connected. When using the evaluation kit, the tool will automatically detect System API ports.
2. Input the password for the required access level. See section 3.8.1 - Password Level 1..3 for more details.
3. Click this button once com port and password has been entered. The tool will now connect to the module, and read the current configuration values according to the specified access level. The values will be displayed in the section marked with "7".
4. When configuration values have been modified, they can be written to the module by clicking the "Write" button.
5. Once done with modifying configuration values, the Logout button should be clicked.
6. Export and Import can be used to save the current setting to a file on the PC, and to later read it back into a module.
7. Contains the current parameter values.

6.3.3 Upload Image

The firmware inside the NEOCORTEC module can be updated to a newer version. New FW versions can be downloaded from the NEOCORTEC website. The files have a .neo extension.

The upload image pane of the Config App looks like this:

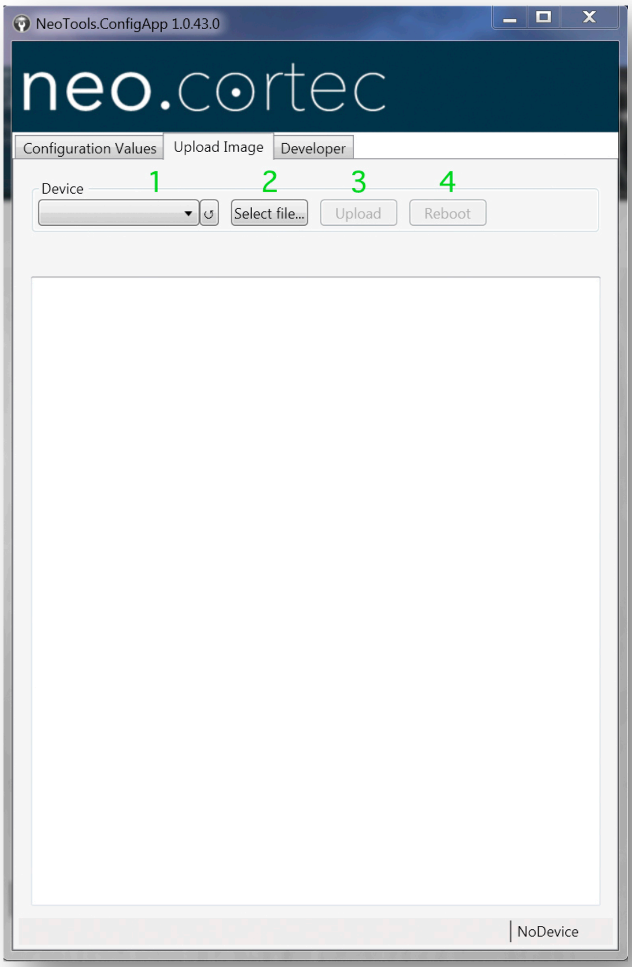


Figure 3 - Upload image

1. Select the com-port which the System API of the modules is connected. When using the evaluation kit, the tool will automatically detect System API ports.
2. Click to select the .neo file to use for the FW update.
3. Once the .neo file has been selected, click the Upload button to start updating the FW in the module.
4. After the FW has been updated, click the Reboot button to force a reboot of the module, and to start with the new FW.

6.3.4 Developer

The developer pane will display real-time data about a node and about the network. This "trace" data is intended for debugging and for displaying general node statistics. The developer pane can also be used to send a default payload message to the detected neighbours, for testing purposes (requires connection to the Application API).

The developer pane looks like this:

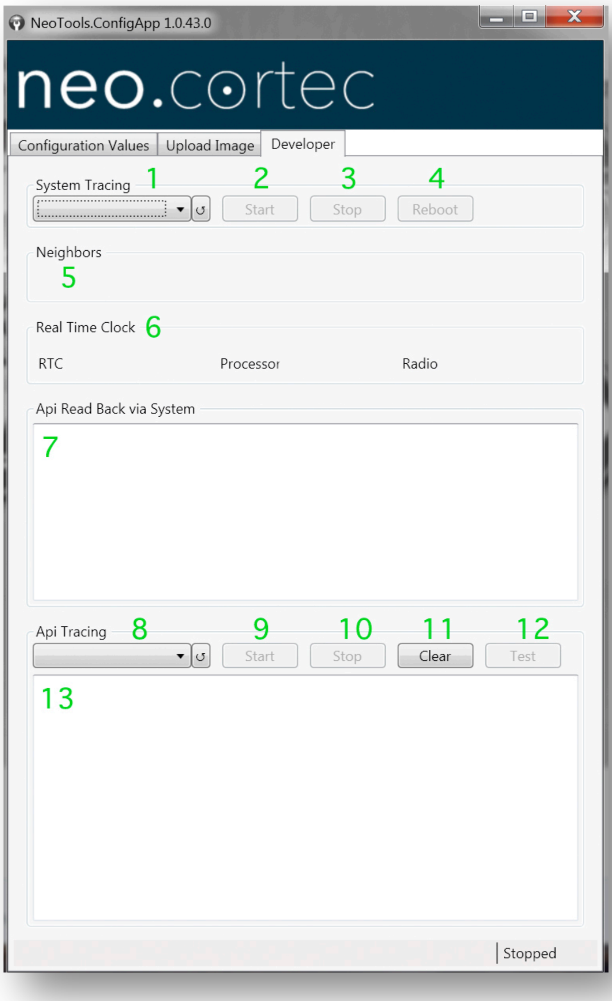


Figure 4 - Developer pane

1. Select the com-port which the System API of the modules is connected. When using the evaluation kit, the tool will automatically detect System API ports
2. Click this button to start monitoring the trace output from the NEOCORTEC module.
3. Click this button to stop monitoring the trace output from the NEOCORTEC module.
4. Click this button to reset the NEOCORTEC module. (corresponds to power cycling the device)

5. This section displays the Node ID's of the Neighbours currently connected with the NEOCORTEC module.
6. This section display information about how much time has been spend in the various modes since the module was powered up.
7. This section is intended to debug host processor integration. It displays the received commands from the Application API.
8. The API tracing section is used to listen to the Application API, and to show received payload data from other nodes. It can also be used for sending payload data to connected neighbours. Select the COM port which the Application API is connected to. When using the evaluation kit, the tool will automatically detect the available Application API ports.
9. Click this button to start monitoring the Application API port.
10. Click this button to stop monitoring the Application API port.
11. Clear the display section (see item 13)
12. Click this button to send a test-payload message to the connected neighbours.
13. Displays messages received on the Application API.

6.4 *Demo App*

6.4.1 Introduction

The Demo Application is intended to be used for simulating the Application Layer in a host processor. It can be used for sending specific payload data, as well as receiving payload data.

6.4.2 Settings

The following screen shot shows the Demo Application:

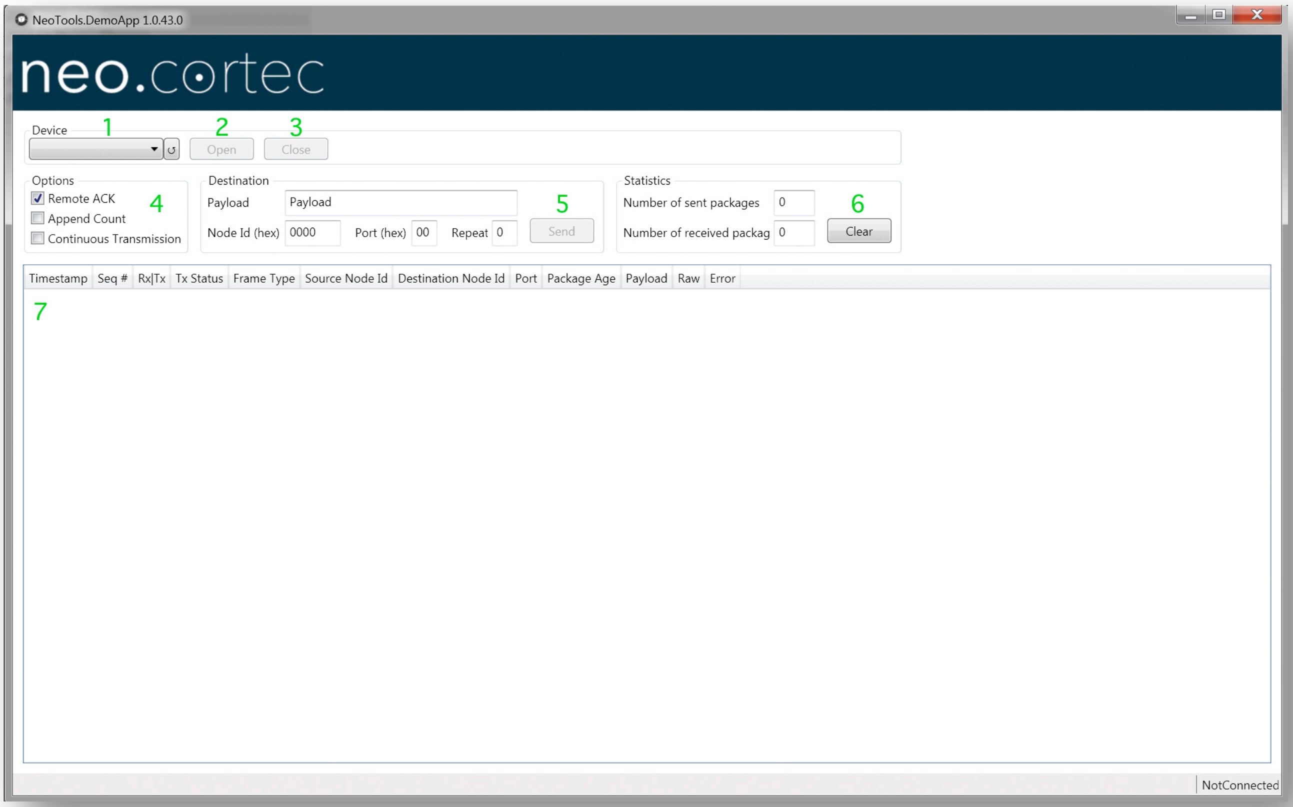


Figure 5 - Demo application

1. Select the COM port which the Application API is connected to. If using the Evaluation Kit, the tool will automatically detect the COM ports available which are connected to a Application API.
2. Click this button to open the selected COM port.
3. Click this button to close the selected COM port.
4. Select parameters for payload data transmission:

   a. Remote ACK: Default enabled. When disabled, the payload data will be send unacknowledged.[3]
   b. Append Count: Default disabled: When enabled, a serial counter number is appended to the payload data. This is useful to see if packet are received in sequence, and if all packets are accounted for.
   c. Continuous transmission: When enabled, a single click on the Send button will start a ongoing transmission of payload data to the specified Node ID. The continuous transmission can be stopped again by disabling this setting.
5. This section is used for defining and sending payload:
   a. The "Payload" text input field can be used for entering the desired payload data to be transmitted.
   b. Node ID: Is the address of the destination node.
   c. Port: Is the port for which the payload data is intended.
   d. Repeat: If set to a number, the send will result in Repeat number of transmissions.
   e. Click the Send button to send the payload specified.
6. Statistics for payload data transmission.
7. A list of activities on the Application API port. Both transmit and receive.

[3] Currently not supported