neo.cortec

User Guide

Doc Status:	**Release**
Doc version:	**2.5**
Date:	**December 2022**

Table of Contents

1 Document revisions 5
2 Introduction 5
3 NeoMesh Wireless Ad-Hoc MESH-networking 6
3.1 Overview 6
3.2 NeoMesh – 2nd generation Wireless Mesh Network 6
3.2.1 Legacy mesh network characteristics 6
3.2.2 NeoMesh network characteristics 8
3.3 NeoMesh Core functionality 8
3.3.1 Time Synchronized Operation 8
3.3.2 Network Initialization 9
3.3.2.1 Beacon Transmissions 9
3.3.2.2 Scheduled Data Transmissions 10
3.3.3 Network maintenance 12
3.3.3.1 Lost neighbour 12
3.3.3.2 The moving node 12
3.3.4 Sending Payload Data 13
3.3.4.1 Speed Routing 13
3.3.4.2 Acknowledged Transmissions 14
3.3.4.3 Non-Acknowledged Transmissions 14
3.3.4.3.1 Sending to non-Sink Nodes 15
3.3.4.4 Broadcast Transmissions 15
3.3.4.5 Payload transmission method 16
3.3.4.6 Latency 18
3.3.4.7 Payload size & throughput 18
3.3.4.7.1 Payload size 19
3.3.4.7.2 Throughput 19
3.3.4.7.3 Payload Aggregation 20
3.3.5 Security 21
3.3.6 Reliability 21
3.3.6.1 Node-to-node Ack/Nack 21
3.3.6.2 FHSS – Frequency Hopping Spread Spectrum 22
3.3.7 Wireless Encrypted Setup 23
3.3.7.1 Remote WES Client activation 23
3.3.8 Power Consumption 24
3.3.9 Hibernation 24
3.3.9.1 Wake-Up Burst types 25
3.3.9.2 Local Wake-Up 25
3.3.9.3 Global Wake-Up 26
3.3.10 Alternate Configuration Mode 26
3.4 Configurable Parameters 28

3.4.1 Node Access and Application settings 28
3.4.1.1 Password Level 1..3 28
3.4.1.2 Trace Output 28
3.4.1.3 Generic Application – Normal Mode 29
3.4.1.4 Generic Application – Alternate Mode 29
3.4.1.5 Application Type ID 29
3.4.1.6 AAPI Aggregation 29
3.4.1.7 AAPI CTS Interleave 29
3.4.1.8 AAPI CTS Timeout 30
3.4.1.9 AAPI Wakeup time 30
3.4.1.10 nWES pin enable 30
3.4.2 RF settings 30
3.4.2.1 TX Output Power 30
3.4.2.2 FHSS Channel Mapping 30
3.4.3 Network access and address settings 31
3.4.3.1 Node ID 31
3.4.3.2 Sink ID 31
3.4.3.3 Network ID 32
3.4.3.4 WES Key 32
3.4.4 Network performance settings 32
3.4.4.1 Scheduled Data Rate Normal Mode 32
3.4.4.2 Scheduled Data Rate Alternate Mode 32
3.4.4.3 Scheduled Data Receive Retry Limit 32
3.4.4.4 Maximum local retries of Payload data 33
3.4.4.5 Beacon Rate 33
3.4.4.6 Beacon Full Scan Rate 33
3.4.4.7 Beacon Full Scan Rate Initial 33
3.4.4.8 Beacon Full Scan Count Initial 34
3.4.4.9 Beacon TX Initial Hold Down Time 34
3.4.5 Neighbour acquisition settings 34
3.4.5.1 Maximum number of neighbours 34
3.4.5.2 Node Inclusion Increment Speed 34
3.4.5.3 Node Inclusion Hysteresis 35
3.4.6 Payload messaging & routing settings 35
3.4.6.1 Payload Aggregation Limit 35
3.4.6.2 High Accuracy Package Age – HAPA 35
3.4.6.3 Network Maximum Roundtrip Time 36
3.4.6.4 Payload Data Time to Live – TTL for Acknowledged Payload 36
3.4.6.5 Payload Data Time to Live – TTL for unacknowledged Payload 37
3.4.6.6 Routing Field Strength Threshold 37

3.4.7 Hibernation and Alternate mode settings....37
3.4.7.1 ACM ID....37
3.4.7.2 Alternate Mode Config....38
3.4.7.3 ACM Presence Time Out to Hibernation....39
3.4.7.4 Wake Up Burst count on Start Up....39
3.4.7.5 Wake Up Burst length....39
3.4.7.6 Max Wake Up Burst period....39
4 Application....40
4.1 Overview....40
4.2 UART....40
4.3 GPIO....41
4.4 HTU21D....43
4.5 RSSI....43
4.6 Payload Package format....44
4.6.1 GPIO....44
4.6.1.1 Internal Temperature Sensor....44
4.6.2 HTU21D....46
4.6.3 RSSI....46
4.7 Finding more information and support....46

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
1.8	Added Settings ID to identify settings from raw list.
2.0	Major update including better system description as well as parameter descriptions for protocol rel.1.3.
2.1	Added missing details for a number of parameters
2.3	Major update for protocol rel1.4
2.4	Corrected error in section 4
2.5	Major update for protocol rel1.5

2 Introduction

This document introduces the reader to the NeoCortec Wireless Ad-Hoc Mesh Networking Technology - NeoMesh. It introduces the core features of the technology. The documents also detail the configuration parameters, and the tools available.
The document is relevant for protocol rel.1.5. Please refer to earlier document versions if using earlier protocol releases.

3 NeoMesh Wireless Ad-Hoc MESH-networking

3.1 Overview

NeoMesh is a wireless system designed with versatility in mind allowing users to build products in many different application areas. It is optimised for ultra-low power operation and allows operation on small batteries for several years. NeoMesh is ideally suited for Wireless Sensor Networks where each sensor/device does not need to send data very often, and where the payload size is small. Contrary to most other Mesh Network Technologies, NeoMesh allows flexible and dynamic network topologies with almost no limit to network size or network depth.
The NeoMesh wireless communication protocol is available in a series of fully integrated and pre-certified modules. Several versions of the modules are available; they all integrate the same core NeoMesh protocol stack across different frequency bands.

Data transmission through the network is done sequentially from node to node, until the data reaches its destination. The following sections provides a detailed description of the network protocol functionality. It is essential to understand these details prior to planning and engineering a product based on the NeoMesh technology.

3.2 NeoMesh – 2nd generation Wireless Mesh Network

To better understand the advantages of NeoMesh, it is useful to compare it with legacy mesh network technologies. NeoMesh is a 2nd generation mesh network which builds on the experience from legacy mesh networks. Improvements are made in the areas which were challenging. In particular, legacy mesh networks had limitations in terms of scalability and power efficiency.

3.2.1 Legacy mesh network characteristics

Legacy mesh networks are organized in a hierarchical system, where different nodes have different capabilities as well as different “responsibilities” in the network.
A typical topology is illustrated below:

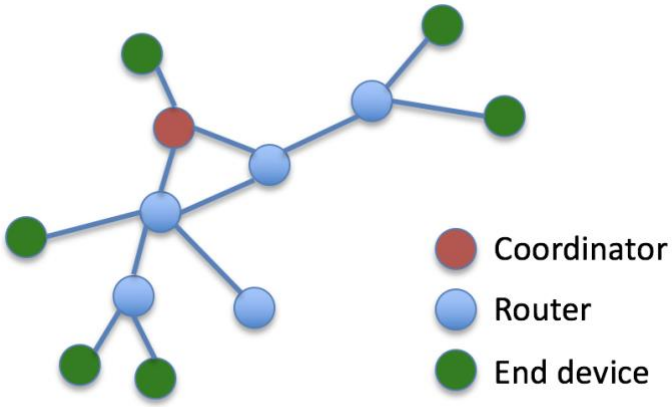


Figure 1 - Legacy Mesh topology

As can be seen, there are 3 types of nodes:

Node Type	Description	Power consumption
Coordinator	Is required to form and maintain the network. Organizes who gets to be neighbours with who, and as such is a single point of failure.	Cannot be battery powered.
Router	Can route messages through the network on behalf of other nodes.	Can normally not be battery powered, as they must continuously listen for messages from end devices which may transmit asynchronously.
End device	Optimised for low power operation and can only send (and receive) messages on its own behalf. It cannot route messages in the network, and as such cannot be used to build the mesh network infrastructure.	Low power operation designed for battery usage.

Table 1 - Legacy mesh network node roles

The layered structure of the legacy mesh networks puts limitation on scalability as well as flexibility.

Another limiting characteristic of the legacy mesh networks is the message routing principles. There are severe limitations on how well the routing protocols handle deep network structures (e.g. more than 10 hops) as well as dynamic topologies. These

limitations reduce the deployment options, and effectively reduces the practical network sizes.

3.2.2 NeoMesh network characteristics

NeoMesh is based on a decentralized approach, where all nodes are self-governing, and not dependent on a coordinator. Similarly, NeoMesh does not implement individual node types, and all nodes share the same low power characteristics. This provides significant advantages over existing technologies in terms of power consumption for the entire network. Moreover, it allows for dynamic network topologies and has no real limitations on scalability or network topologies.

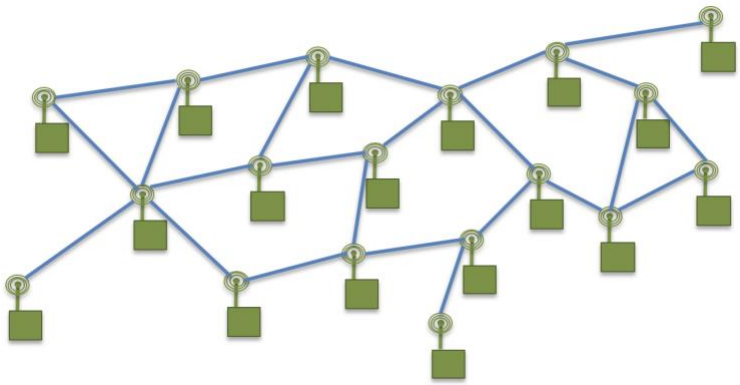


Figure 2 - NeoMesh topology

As can be seen from the figure above, there are no "special" nodes. All nodes act as a router, and at the same time maintain ultra-low power consumption capability.
These capabilities allow for full flexibility when deploying the network and allows for a network which scales with the application requirements.

3.3 NeoMesh Core functionality

3.3.1 Time Synchronized Operation

The NeoMesh is a Time Synchronized network, where the nodes during normal operation enter sleep mode as often as possible to conserve energy. They wake up at pre-determined intervals to communicate with their neighbours. Each node in the network is aware of when other nodes within range will transmit, and as such they can schedule to wake up when that occurs.
The synchronized operation of the nodes ensures that all nodes are capable of low power operation. This is contrary to most other mesh network technologies where asynchronous operation is being used, and as a result limits the low power capabilities for nodes which acts as a router.

The scheduled communication is being used to maintain the network in terms of neighbour relations, and to maintain routing information. As such, this communication happens periodically regardless of payload data being send or not.

3.3.2 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. If no network is detected, the node will initiate the network on its own, allowing other new nodes to detect it.

3.3.2.1 Beacon Transmissions

A node which is participating in the network, or is trying to initialize a new network, will periodically transmit a beacon on a dedicated beacon channel.
Nodes not part of a network, or nodes already in a network, can detect new neighbouring nodes through the reception of Beacons. Nodes who are part of the same network will synchronize their Beacon timing such that the Beacon Event occurs at the same time throughout the entire network.
Let us consider the situation where only two nodes are present, and they are in their initial state where they are not yet synchronized to each other:

Node A transmits its beacon periodically. Node B does the same, but they are not synchronized yet:

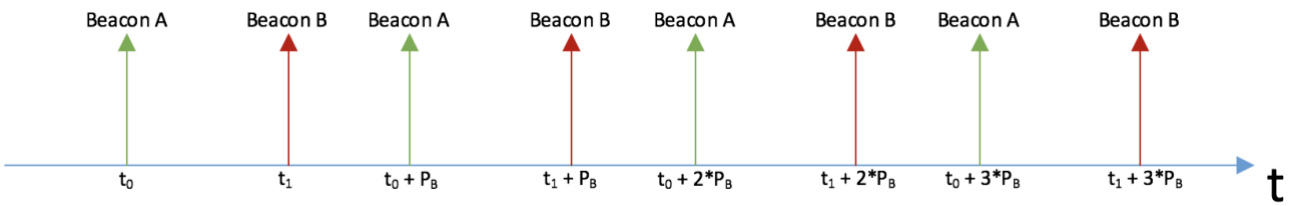


Figure 3 – Beacon timing, unsynchronized

To synchronize, the nodes perform a Full Beacon Scan. This is done periodically and entails that the scanning node listens at least for a full beacon period ("P_B" in the figure above).
When either Node A or Node B receives a beacon from its neighbouring node, it will - based on certain rules, decide if it will follow the beacon timing of the other node, or if it keeps its own timing. In the following illustration, Node B performs a Full Beacon Scan, where it receives the beacon from Node A, and decides to follow the beacon timing of Node A, after which the beacon transmissions of both Node A and Node B occurs at the same time:

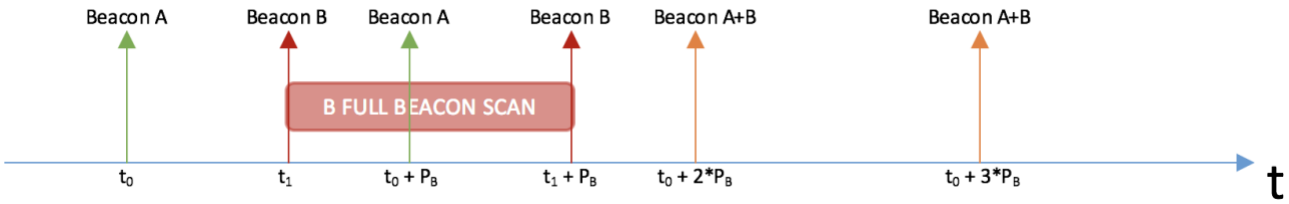


Figure 4 – beacon timing, synchronized

The Beacon Event is divided into timeslots. When a node transmits its Beacon, it randomly selects between the available timeslots. During the Beacon Event each node will listen for the full duration of the Beacon Event, only interrupted by the time when it transmits its own beacon. In dense networks with many nodes within radio range, the nodes will skip beacon transmissions randomly to further decrease the likelihood of collisions. This approach ensures that 100's of devices can operate within radio range of each other.

Once the network is created, the beacon transmissions are in theory no longer needed provided that the network is static, and no new nodes join the network. Beacon Events do however continue to occur in order to ensure that when the topology changes – either because nodes physically change location, or because radio links are broken due to noise or obstructions – the nodes in the network can discover the nodes within radio range. This also ensures that if two sections of the network become disjoined, they can re-join once within range of each other again.

The period of the Beacon Event is user configurable and can be tuned to the desired rate of which the network shall be able to detect new devices. In most configurations where static topologies are used, a rather slow setting can be used to conserve energy.

3.3.2.2 Scheduled Data Transmissions

In addition to the Beacon transmissions, each node also transmits Scheduled Data. Scheduled Data is the backbone in the NeoMesh network architecture, and serves a number of purposes; to keep neighbouring nodes in synch, transmit payload data when needed, create and maintain routes in the network and to keep nodes logically separated in super dense networks.
When a node transmits its Beacon, it includes information about the time until it plans to transmit Scheduled Data next time. This allows neighbouring nodes to allocate a wake-up timer to wake the node from a low power mode and listen to the Scheduled Data Transmission of the node, and by doing so becoming a neighbour to the node.

Let us consider a situation with 2 nodes (Node A & B) within radio range of each other, where their Beacon Timing is already synchronized, and they are about to become neighbours to each other:

First, let us look at the transmission activity of Node A:

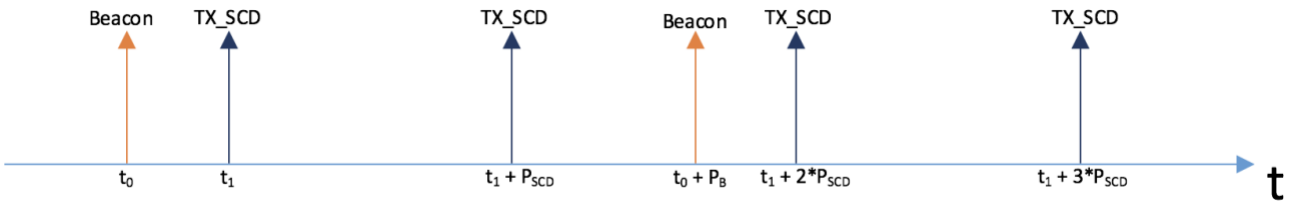


Figure 5 – Single node activity pattern

In this example, the Beacon Period (P_B) is twice as long as the Scheduled Data Period (P_{SCD}).
As mentioned above, the beacon transmissions include information about the time until next Scheduled Data transmission. When the neighbouring nodes receive the beacon, they can assign a wake-up event to listen for the Scheduled Data transmission of Node A. Let us look at Node B's activity (both TX and RX) compared to A:

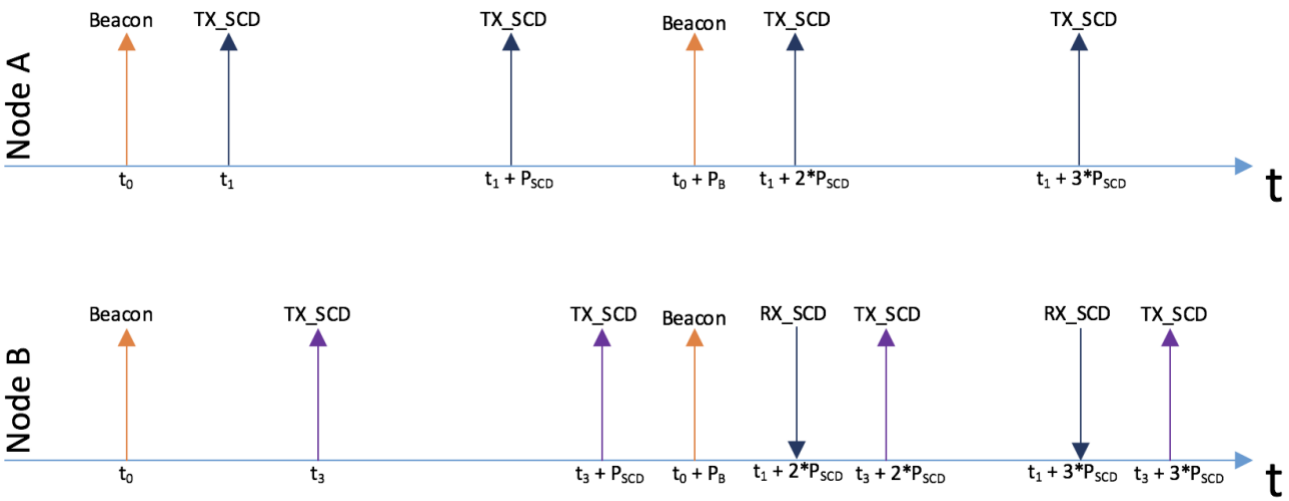


Figure 6 – Activity pattern of two nodes

In the example above, Node B does not pick-up the Beacon transmission from Node A at time t_0, but receives the beacon transmission a time $t_0 + P_B$. Node B wakes up hereafter to receive the Scheduled Data transmissions of Node A – the first time at $t_1 + 2*P_{SCD}$.
Now Node B is neighbour to Node A.

Node A will in a similar fashion receive the Beacon transmission from Node B, and then wake up to receive the Scheduled Data transmissions from Node B, and as such they are now neighbours.

When more nodes join, they do so in the same way as described above, and the network will initialize autonomously.
The current consumption of a node is proportional to the number of neighbours it is connected to and therefore it is desired to reduce the number of neighbours to a minimum. On the other hand, multiple neighbours increase the redundancy of the network and provides higher stability. It is recommended to plan the network such that each node has at least 4 neighbours. In very dense networks, we want to limit how many nodes are accepted as neighbours to conserve energy. There is however a risk that reducing the number of neighbours too much will result in "routing islands" where data cannot be routed across the dense network sections. To control the neighbour acquisition, a setting is provided which can limit the maximum number of neighbours a node will accept. A setting of maximum 6 neighbours eliminate the problem of routing islands in dense networks. In summary an ideal NeoMesh network is deployed such that each node has minimum 4 neighbours in range, and a setting that limits the maximum neighbours to 6.

3.3.3 Network maintenance

After the network has been established, and the nodes has been linked to each other by means of listening to the Scheduled Data transmissions of their neighbouring nodes, the network will have to cope with changes during normal mode of operation. The changes that are likely to occur is that nodes move position, or some external event influences the radio link quality between two or more nodes – for instance radio noise from other sources impact the link between two nodes, or an obstacle moves in between two nodes, blocking the radio link.

3.3.3.1 Lost neighbour

The criterion for when a neighbour is lost, is when a certain number of Scheduled Data transmissions are not received (the number is configurable).
If the configured number of Scheduled Data transmissions are not received, the wake up event for the particular node will be removed, and the neighbour relationship is no longer there.

3.3.3.2 The moving node

A node which moves through the network such that it moves out of range for some nodes, and moves within range of other nodes, will try to update its neighbour connections while it moves.

As already mentioned in section 3.3.3.1, when Scheduled Data transmissions are no longer heard from a neighbour, it will be removed from the neighbour list. Similarly, as explained in section 3.3.2.1and section 3.3.2.2, the moving node will pick up beacon transmissions from nodes which comes within range automatically and as such become neighbours.
The pace with which the node(s) can move, depends on the setting of the Beacon Period and the Scheduled Data period.

3.3.4 Sending Payload Data

NeoMesh provides multiple ways of sending data through the mesh network. Overall, Acknowledged Transmissions, Non-Acknowledged Transmissions and Broadcast Transmissions are supported. Both Acknowledged and Non-Acknowledged transmission types is using the unique Speed Routing algorithm. Whereas Broadcast Transmissions are flooded into the network.
Depending on the transmission method used, payload data can be sent to any device in the network.

3.3.4.1 Speed Routing

Both Acknowledged and Non-Acknowledged transmission types are using Speed Routing to route the payload data from source to destination. Speed Routing is a patented technology used exclusively in NeoMesh.
The Speed Routing algorithm identifies the fastest path from source 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 like seen in other Mesh Network implementations.
Speed Routing is based on a “next-best-hop” principle, where each node has a routing table containing information on which one of its current neighbours is the best candidate to route a message to a given sink node (destination).
Speed Routing can handle up to 128 sink nodes. This means that in each network, up to 128 nodes can be configured to be destinations for routed payload data. All nodes in the network acts as routers.

3.3.4.2 Acknowledged Transmissions

This is the most reliable type of payload data transmission. It corresponds to TCP transport known from IP networks, where the originating node will receive an acknowledge message once the payload has been delivered at the destination.
When the application wants to send payload data to a sink node, all it must 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.
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 act, according to the needs of the application.
This type of messaging is useful in applications where each message is important – for instance in metering and sub-metering applications where the data contained in the message is vital.

Note: If the originating node is a non-sink node (i.e. Node ID > 128), the protocol cannot use Speed Routing to route the ACK message back to the originator. In this case, the protocol will route the ACK message back along the same route as the payload message was routed. Since this backward route is only valid while the topology stays static, it is not recommended to use Acknowledged Transmissions in dynamic topology networks with more than 128 nodes. Similarly, only sink node Node IDs shall be used (Node ID < 128). In static networks, it is ok to use Acknowledged Transmission for any number of nodes in the network.

3.3.4.3 Non-Acknowledged Transmissions

Non-Acknowledged Transmission corresponds to UDP transport known from IP networks. Messages are routed from source to destination, and while travelling through the network, the local package exchange between neighbouring nodes is done using a local ack/nack handshaking. This means that even though an acknowledge message is not routed back from the destination to the originator, the payload data is still being transmitted in a reliable fashion while on route from originator to destination.
This type of transmission is useful in networks where data is exchanged frequently, or in networks with a highly dynamic topology (because backward routes are not used). Since the total number of messages routed in the network when using Non-Acknowledged transmission is typically half of that when using Acknowledged transmission, the pace of

the network can typically be configured lower and thereby the average power consumption can typically be lower.

3.3.4.3.1 Sending to non-Sink Nodes

As previously mentioned, Speed Routing maintains routing information for nodes with Node Id < 128 (sink nodes), and as such not all nodes in a network are directly addressable.
There is however a method for sending Non-Acknowledged messages to non-sink nodes:
Non-Acknowledged messages are being routed in the same way as Acknowledged messages, and when these messages travel through the network, a backward route is being created. This means that once a non-sink node has transmitted either an Acknowledged or a Non-Acknowledged message to a given sink node (for instance a gateway node), it is possible to send a Non-Acknowledged message from the sink to the non-sink.

As an example: A network with 1000's of wireless NeoMesh enabled sensors, is configured such that the sensors send measurement samples periodically to a gateway node in the network. All the sensor devices are non-sink nodes. The gateway occasionally has to send messages to the individual sensors. It can do so by waiting until a message has been received from the sensor to which it needs to send data. At this time, a backward route exists and can be used for sending the message to the non-sink sensor.

Note: This method of sending messages to non-sink nodes is only useful in static networks, as the backward routes are only valid while the network stays stationary. Similarly, there is a limit for the maximum number of concurrent backward routes, and old routes will be exchanged for new routes. This means, that the backward route will disappear after a while. The validity period of a backward route is determined by the total number of devices in the network, and the frequency of payload transmissions. In general, backward routes shall only be used within a few multiples of 10 Scheduled Data periods after they have been created.

3.3.4.4 Broadcast Transmissions

Broadcast Transmissions can either be unicast or groupcast. There is however no filtering for groupcast messages in the protocol layer, and as such the application layer will have to filter out groupcast messages which are non-relevant based on the group ID.

Broadcast Transmissions is a special type of Non-Acknowledged transmissions where Speed Routing is not being used. Instead, the messages are flooded into the network and as such all devices will receive them. Notice that this is not limited to sink nodes – non-sink nodes will receive these messages as well.
Broadcast Transmissions are useful in situations where the same information needs to be transmitted to all – or a group of devices.

Broadcast messages are stored in all nodes until they expire (a setting controls the Time To Live). If new nodes join the network while broadcast messages are still present (not yet expired), they will be delivered to the newly joined node.

The method for sending broadcast or groupcast messages is exactly the same as sending Non-Acknowledged messages. The difference is that the destination Node ID is in the broadcast or groupcast Node ID range.
Node ID's 0xFFF0 to 0xFFFF are broadcast/groupcast ID's. When sending a message to a Node ID in that range, all devices will receive it. Upon reception, the application layer can decide to ignore or use it depending on what destination Node ID was used for the broadcast transmission. See the Integration Manual for further details on how to send and receive broadcast messages.

3.3.4.5 Payload transmission method

Regardless of which transmission type is being used, payload data is being transmitted from node to node through the network, as part of the normal Scheduled Data transmissions.
To illustrate how payload data is being transmitted through a network, let us consider a very simple network consisting of 3 Nodes:

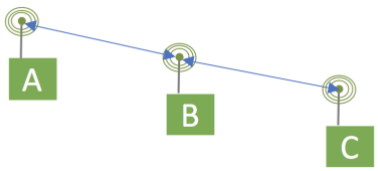


Figure 7

In the example, Node A is neighbour to Node B, and Node B is neighbour to both Node A & Node C. The application layer at Node A wants to transmit a payload package to the application layer at Node C using Acknowledged Transmission. In the following figure, the Scheduled data events between the nodes are illustrated as well as how the Payload Package travels from node to node, and how the acknowledge travels back to the originator:

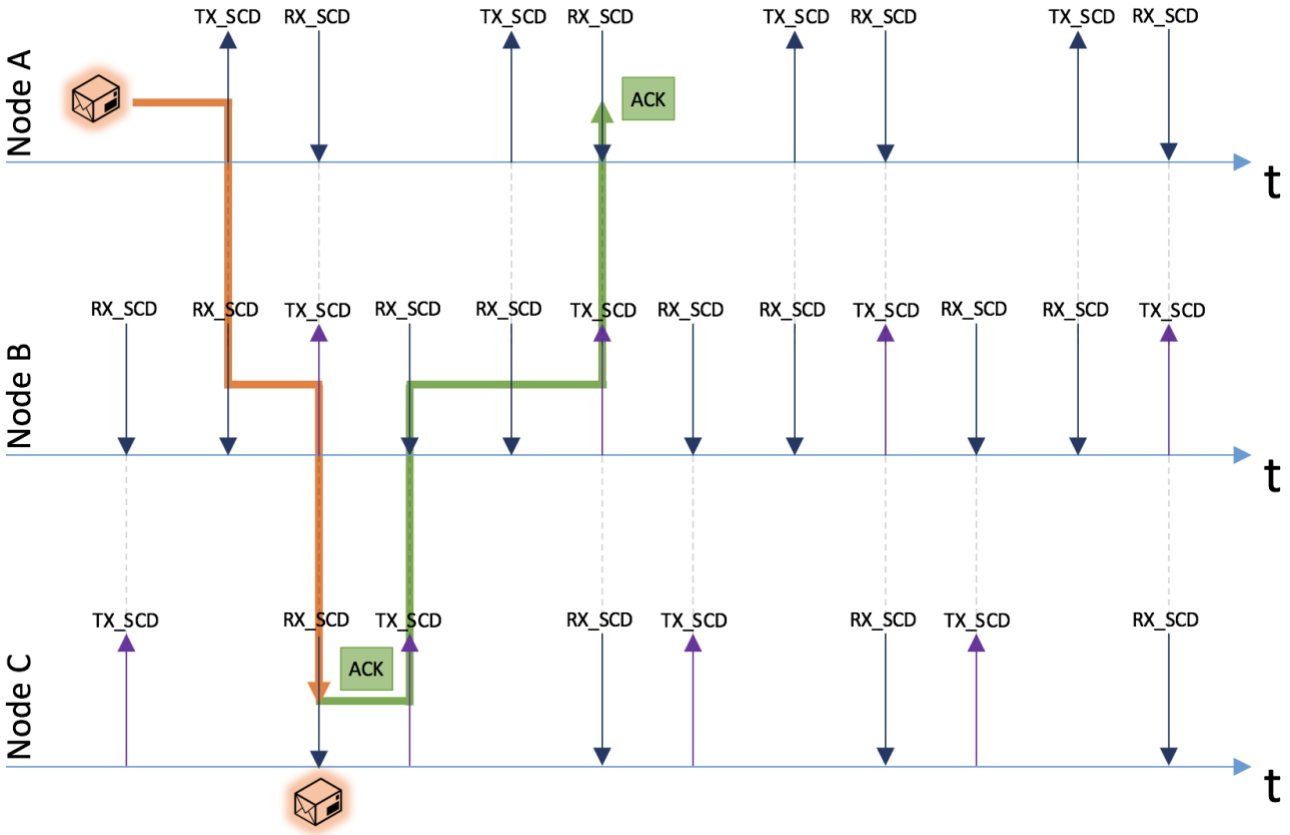


Figure 8

Initially the application layer enqueues the payload package in Node A. When Node A transmits Scheduled Data next time, it will also transmit the payload package. Node B will receive the payload package and hold on to it until it is time to transmit scheduled data. When Node B transmits Scheduled Data, the payload is included, and Node C receives it. Note that the Scheduled Data transmissions of Node B is received by both Node A and Node C, but the routing protocol ensures that the payload is only received by Node C. Once the payload package has reached Node C, and if the transmission type is "Acknowledged", an acknowledge will be routed back to the originator in the same way as the payload data was routed – this is shown by the green line in the illustration.

When the payload travels from node to node through the network, a local handshake between the sending and receiving node is made; upon successful receive of the payload package, a local Acknowledge is send back to the sending node. This ensures that a package is not removed from the outbound queue of the sending node until it has received confirmation that the receiving node has received the payload package, and the CRC was successful.
If the local acknowledgement is not received within a timeout window, the payload package will be retried next time the Scheduled Data transmission is due.

3.3.4.6 Latency

Since the payload is being transmitted with the scheduled data transmissions, the end-to-end latency is directly coupled to the Scheduled Data rate.
As can be seen from Figure 8, the payload package waits at each node on the route. The wait time is the time from the package was received as part of a Scheduled Data from a neighbouring node, until it is time for the node to transmit its own Scheduled Data. Depending on the offset between the Scheduled Data Events from the two nodes, the wait time can be anything between 0 and the Scheduled Data Period. In a real network, where there typically are multiple hops between the originator and the destination, the wait time at each node will in average be one half of the Scheduled Data Period.
Generalising this, enables us to calculate the average end-to-end latency like this:

$$t = 0.5 * T_{SCD} * n$$

Where T_{SCD} is the Scheduled Data period, and n is the number of hops between source and destination.

3.3.4.7 Payload size & throughput

The NeoMesh network is optimized for applications where the amount of data being transmitted is relatively low. Wireless Sensors, where the measurements may only be a few bytes is a good example.
When planning the network and the application a few considerations have to be made. The following two sections explains this in detail.

3.3.4.7.1 Payload size

A payload message can consist of up to 21 bytes of raw payload data[1]. The raw payload is allocated exclusively to application data. In addition to the raw payload, the data, which is being send, includes a header with the following information:

Field	Description
Destination Node ID	The Node ID of the Sink Node to where the package is intended
Source Node ID	The Node ID of the node which is the originator of the package
TTL	Time To Live value – if the package is older than the TTL value, it will be removed from the network
CRC	Cyclic Redundancy Checksum value used to validate a package locally
Age	The accumulated age of the package, calculated from the time it was enqueued in the source node

3.3.4.7.2 Throughput

When calculating throughput, it does not make sense to use bytes/sec or similar, as the data is not being streamed but rather sent as individual packages with a fairly long latency (see section 3.3.4.6). In a real network with 100's of nodes, and a Scheduled Data Rate optimised for low average current consumption, the throughput from each node may only be a few payload packages per minute.
When a network is planned, the total number of nodes and the reporting interval shall be balanced with the throughput at the destination for the data traffic.

Let us consider a network with 100 sensors who is sending sensor data to a single gateway device in the network:
Since all payload data is intended for the same destination (the gateway), the destination node will form the main bottleneck of the network. This means that the receive capability of the destination node will limit how much payload data the entire network can generate pr time unit.
As mentioned in section 3.3.4, payload data is sent when Scheduled Data is being transmitted. This means that the destination node can receive one payload message from each of its direct neighbours once every Scheduled Data Period.
In other words, if the Scheduled Data Period is 1 sec, the destination node can receive 1 payload message from each of its neighbours every second.

[1] The size depends on the HAPA setting – see section 3.4.6.1 for further details.

In our example network, the destination node has 4 direct neighbours, and can therefor receive 4 payload message every second. This throughput is shared between the 100 nodes in the network, which means that each node cannot send payload data more frequent than once every 25 seconds[2]. To account for retransmissions etc in the network, a minimum of 20% back off shall be considered.

3.3.4.7.3 Payload Aggregation

In the calculations above, it is assumed that only a single payload message will be transmitted at each Scheduled Data transmit event. With release 1.5 of the NeoMesh protocol stack, payload aggregation has been added which allow for multiple packages to be transmitted at each Scheduled Data transmit event. This increases the throughput at a given Scheduled Data Rate, or alternatively allow for a reduced Scheduled Data Rate at a given payload data reporting interval – thus lowering the average current consumption of the nodes in the network.

Let us consider a network section where a node has 6 neighbours. During the last Scheduled Data period, 4 of its neighbours has transmitted 1 payload message each which is now in the outbound queue of the node. Now it is time for the node to transmit Scheduled Data. It will scan the local outbound queue to see if there are payload messages to transmit. Normally, without aggregation, it will transmit the first message in the queue (FIFO) to the neighbour to whom it is intended (based on the routing table). With aggregation, the node will scan the outbound queue to see if more messages are enqueued for the same neighbour. In our example, all 4 messages in queue are to be routed through the same neighbour, and therefore the node will aggregate all 4 messages and transmit them at once during the same Scheduled Data transmit event.

The maximum number of packages which can be transmitted during one Scheduled Data transmit event is 5. The actual number of packages transmitted can be lower. This will be the case if for instance the timing of events does not allow enough time to complete the transmission of all packages, or if the inbound queue at the receiving node is full.

With aggregation, multiple aggregated payload packages will remain individual packages when they are routed onwards in the network, and when delivered at the application layer at the final destination. This also implies that the criteria for aggregating packages at a node, is not based on the end-destination for a payload package, but only the next routing hop along the way. This entails that aggregation helps to increase throughput not only at the destination node, but also at any choke point in the network enroute towards the destination.

[2] 1 / (4 messages/sec) * 100 nodes

3.3.5 Security

All communication, not only payload data, but the full Scheduled Data transmission as well as the Beacon transmission, is encrypted using AES128. The key for the encryption is the same as the Network ID (see section 3.4 Configurable Parameters). This means that nodes which 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.

Acknowledged Transmission is using a challenge/response authentication between the originator and the destination. This means that each message exchange will be subject to a unique challenge with a unique associated response. Since the challenge/response messaging is included in the encrypted communication, the data exchanged will appear completely random from transmission to transmission, even if it is the same payload that is being exchanged. This prevents playback attacks[3] and improves the overall security level of the communication system drastically.

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 (acknowledged) 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.

3.3.6 Reliability

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

3.3.6.1 Node-to-node Ack/Nack

When neighbouring nodes communicate, the receiver of payload data responds with an Local Acknowledge message if the CRC calculation on the received data is successful. The sending node will retry the packet until either a Local 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 best next hop towards the destination. Should the missing Local Acknowledgements be a result of the given neighbour no longer being within radio range, the neighbour will

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

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 being the best next hop towards the destination.

Theoretically there can be a situation where the data was received correctly at the neighbour, but the Local 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. For Acknowledged Transmission Type, once the packet reaches the destination, the first packet will be acknowledged, and an Ack will be routed back to the Originator. The second packet will be Nack'd 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. For Non-Acknowledged Transmission types, the duplicate payload messages will be delivered at the application layer, and the application layer will have to filter them out based on the included serial number which is supposed to be unique for each payload datagram. See the Integration manual for further details on how to send payload data, and which parameters are required for each type.

3.3.6.2 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. The range of physical channels varies from frequency band to frequency band, and thereby module variant to module variant. The modules are pre-certified for compliance to EU and US rules, and the available physical channels are limited to comply with rules in the different regions.

3.3.7 Wireless Encrypted Setup

A NeoMesh node is pre-configured for a particular network. The configuration parameters can be written to the module through the System API UART.

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 announce the network through an 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 which do not have the correct WES key cannot be setup for the particular network because they will not be able to decrypt the WES beacon.
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[4] 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.

3.3.7.1 Remote WES Client activation

The Wireless Encrypted Setup client mode is enabled when the node has 0xFFFF as its Node ID. This can be programmed directly through the system API, or by activating the nWES pin on the module.

[4] Each NeoMesh module is equipped with a 5 byte unique ID pre-programmed at factory level, and non-changeable

Remote WES activation allows WES mode to be enabled remotely, by sending a package to the node through the mesh network. When the target node receives the package, it will correspond to activating the nWES pin.
This is particularly useful in situations where it is required that nodes can be removed from a network and reassigned to a new network.

The remote WES activation is done through the Application API. See Integration Manual for further details.

3.3.8 Power Consumption

As mentioned previously, the NeoMesh wireless network operates as a time synchronized network, where all nodes are in sleep mode most of the time and wakes up periodically to communicate with the neighbouring nodes. Depending on the settings, this allows for very low average current consumption for all the nodes in the network.
The NEOCORTEC Config Application can be used to calculate the expected average current consumption based on the actual configuration for the node.

3.3.9 Hibernation

In some applications of the NeoMesh network technology, it is desirable to configure a node such that it can hibernate until a user wakes the node up. In hibernation mode, it is understood that the node will consume current at a level lower than any active mode allowing extended battery life in situations where the active network operation is not required.
This is particularly useful in applications where for instance the node and the battery in its entirety will be moulded into for instance a plastic cavity to ensure efficient shielding from the environment. It could also be used in situations where the network is not required to be operational in longer periods of time, such that it would be beneficial to bring the entire network into hibernation mode.
Hibernation is an additional state which the node can be in. The following diagram illustrates the various states and how they are entered and exited:

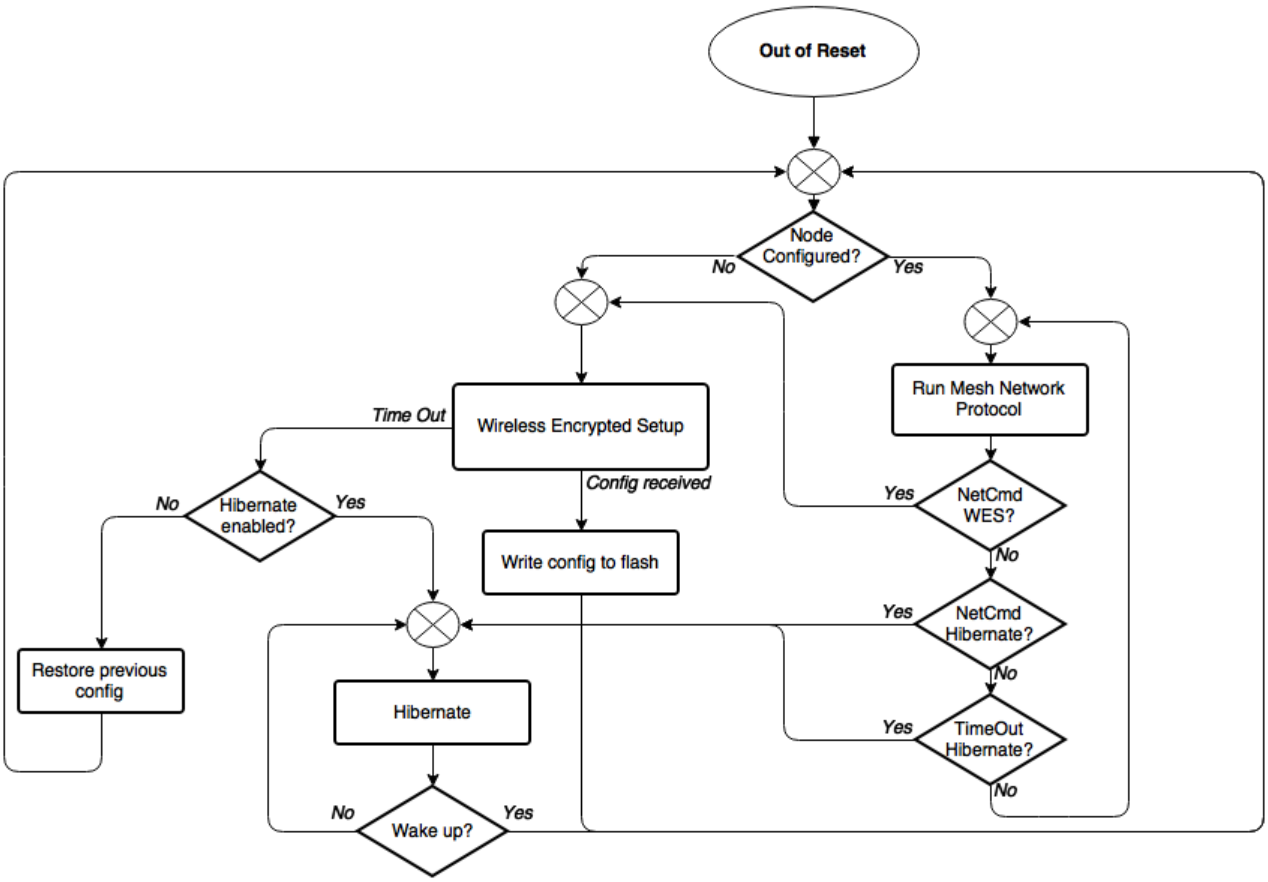


Figure 9

A hibernating node can be brought out of hibernation when another node transmits a wake-up burst.

3.3.9.1 Wake-Up Burst types

To wake up a hibernating node, a Wake-Up Burst needs to be transmitted. All nodes can send Wake-Up Burst. There are 3 kinds of Wake-Up Burst which will wake 3 different groups of nodes:

1) All Nodes with a specified Network ID
2) All Nodes with a specified WES ID
3) The Node with the specified UID

The node(s) which is targeted for the wake up, shall be within radio range of the node who sends the Wake-Up Burst.

3.3.9.2 Local Wake-Up

The application layer at a Node can command the node to transmit a Wake-Up Burst. This will only wake nodes which are within radio range. See integration manual for details on how to send Wake-Up Bursts.

3.3.9.3 Global Wake-Up

A node can be configured to automatically transmit Wake-Up Burst depending on a set of configurable parameters.
Automatic Wake-Up Bursts are always of type 1 according to section 3.3.9.1, where the Network ID is same as the Node who sends the Wake-Up Burst.

Automatic Wake-Up Bursts can be sent always when the node is in active mode. Alternatively, it can send the bursts when the ACM (see section 3.3.10) node is present in the network.
This will allow for applications where it is required to hibernate the entire network for longer durations to reduce energy consumption. With automatic Wake-Up burst enabled when the ACM is present, the network will wake up when the ACM is powered up, and it wakes the nodes within range of the ACM. The nodes who have been waken up by the ACM will now wake up nodes within their range, and as such the entire network will wake up sequentially.

3.3.10 Alternate Configuration Mode

In some applications it is desirable to operate the network in two different modes depending on an external situation. This could for instance be in a setup or debug situation, where a technician needs rapid information about the network health and integrity.

The mode of the network (alternate vs. normal) is controlled from a single node – Alternative Configuration Master (ACM) Node. This node will as part of the normal operation, broadcast the mode setting into the entire network. This can be done at no additional overhead in terms of network throughput or current consumption. The ACM node will typically be a gateway node but does not need to be.
The application layer at the ACM node, can control the mode through the UART API interface (see Integration Manual for details).
Each node in the network needs to be configured with the Node ID of the Alternative Configuration Master Node (see section 3.4.7.1 for details on configuration)

In the alternate configuration, it is possible to define the following parameters:

1. Scheduled Data Rate
2. Generic application settings
3. Hibernation behaviour

The Scheduled data can be configured to be two different values in Normal and Alternate mode.

The Generic Application can be fully configured to perform two different tasks, depending on the mode. It could for instance be:

Normal	Alternate
HTU21D, temperature and humidity measurements send to the gateway node once every 10 minutes	RSSI application sends list of neighbours including the RSSI levels for each neighbour, to the gateway node once every 10 seconds.

The Hibernation behaviour can be configured such that hibernation state is supressed in alternate mode. This is useful in situations where the presence of the ACM node controls if the nodes enter hibernation. If Alternate Mode is configured to supress hibernation, hibernation is not entered even if the ACM node disappears from the network.

3.4 Configurable Parameters

In the following, all parameters, which affect the performance of the network as well as the individual node, are listed and explained. Each setting has an ID which is noted in its decimal value. This ID is used when configuring the module directly without the NeoCortec tools.
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.

3.4.1 Node Access and Application settings

3.4.1.1 Password Level 1..3

ID: 5 – 7 (Level 1, Level 2, Level 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.

3.4.1.2 Trace Output

ID: 44
Enables or disables trace messaging output on the SAPI UART interface. A value of 1 enables the trace output, whereas a setting of 0 disables the output.
By default, trace output includes Neighbour lists and AAPI messaging debug information which can be viewed in the Config Application on the "Developer" pane. It is useful when debugging network connection issues etc.
Note: Enabling trace output causes an increase in current consumption, and should only be done in debugging situations.

3.4.1.3 Generic Application – Normal Mode

ID: 25
This parameter controls the application layer when the network is operating in normal mode (see section 3.3.10 for details on Normal vs. Alternate mode).
The setting value is best created using the Config Application – see section 4 for more details.

3.4.1.4 Generic Application – Alternate Mode

ID: 58
This parameter controls the application layer when the network is operating in alternate mode (see section 3.3.10 for details on Normal vs. Alternate mode).
The setting value is best created using the Config Application – see section 4 for more details.

3.4.1.5 Application Type ID

ID: 15
When the node is configured through WES, it transmits this value as part of its WES Setup Request. It can be set to an arbitrary value (0…255) and shall be used to identify a type of node, which the WES Server can use to decide which application setting to use for the WES configuration.

3.4.1.6 AAPI Aggregation

ID: 74
This setting controls how many AAPI packages the module is allowed to transmit to the host controller at each nWU event.
Allowed values are 1 … 5 corresponding to 1 … 5 AAPI packages at each nWU event.

3.4.1.7 AAPI CTS Interleave

ID: 50
The CTS pin is active (low) every time there is a TX Scheduled Data event. If it is desired to conserve energy and payload data is not send very often, the CTS Interleave setting can be used to limit the CTS event.
Default value is 1, which results in one CTS event every TX Scheduled Data event. Legal values are 0 to 255.
0 is a special value which trigger a CTS event on every Wake Up event (TX SD, RX SD, Beacon events etc).

3.4.1.8 AAPI CTS Timeout

ID: 51
Specifies the duration which the CTS pin is active (low) before it transitions In-Active (high) if no data is received.
Possible values are 1…255, where 16 is default. The Timeout parameter is in steps of 30.5uS. The default value (16) corresponds to 488uS.

3.4.1.9 AAPI Wakeup time

ID: 52
Specifies the time from which the nWU pin transitions active (low) until the module starts to send data to the host controller. This value shall be set such that the external host has enough time to receive data, when waking up from the nWU active event.
Possible values are 1…255, where 16 is default. The Wakeup time parameter is in steps of 30.5uS. The default value (16) corresponds to 488uS.

3.4.1.10 nWES pin enable

ID: 47
Controls if the nWES pin on the module is enabled or disabled. When enabled, the module will enter WES client mode when the nWES pin is pulled low. If disabled, the nWES pin will not affect the module behaviour.
Enable = 0x01
Disable = 0x00

3.4.2 RF settings

3.4.2.1 TX Output Power

ID: 33
The RF transmitter output power. Possible values depend on the module version.

3.4.2.2 FHSS Channel Mapping

ID: 41
The frequency hopping channels can be configured freely to a range of RF Frequencies. 16 logical channels can be mapped to a range of 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 the ETSI (2.4GHz & 868MHz) and FCC (2.4GHz & 915MHz) rules. For other regions, there may be other rules which needs to be observed when allocating the channels.

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. Channel 17 is used for WES. Channel 18 is used for Wake beacons.

3.4.3 Network access and address settings

3.4.3.1 Node ID

ID: 10
Specifies the Node ID of the node. Legal values are 0x0000 to 0xFFEF. Nodes in the range 0x0000 … 0x007F are sink nodes, and can be directly addressed by other nodes in the network.
Values 0xFFF0 to 0xFFFF are broadcast/groupcast ID's and cannot be assigned to a particular node.
The value 0xFFFF is special, and 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. 0xFFFF is also a broadcast/groupcast ID.

3.4.3.2 Sink ID

ID: 69
Normally sink nodes are defined as nodes with Node ID in the range 0x0000 … 0x007F (see section 3.4.3.1 for more details on Node ID). However, a node can also have a Node ID in the non-sink node range and then be assigned a Sink ID. This is useful in applications where it is required to have dynamic assignment of Sink nodes, or if it is desired to have multiple devices sharing the same Sink ID – for instance in networks with multiple gateways. When using this setting to assign Sink IDs, the normal Node IDs shall be assigned in the non-sink node range.
Allowed values are 0x00 … 0x7F.

This setting can be overwritten through the AAPI interface while the protocol stack is running, allowing for dynamic Sink ID assignment. See Integration Manual for further details.

3.4.3.3 Network ID

ID: 42
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.

3.4.3.4 WES Key

ID: 46
Specifies the AES 128 encryption key for the WES Channel.

3.4.4 Network performance settings

3.4.4.1 Scheduled Data Rate Normal Mode

ID: 17
The rate in seconds for transmission of Scheduled Data when the network is in normal mode. 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[5].

The setting also directly affects the average current consumption of the module.

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

3.4.4.2 Scheduled Data Rate Alternate Mode

ID: 64
Same as 3.4.4.1 but in Alternate Mode.

3.4.4.3 Scheduled Data Receive Retry Limit

ID: 23
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.

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

3.4.4.4 Maximum local retries of Payload data

ID:27

Deprecated in release 1.5. Now retries will continue until TTL expires for the package.

When payload data is transmitted from one node to the next, a local Ack/Nack between the nodes increases the likelihood for the transmitted payload to be received. This parameter specifies how many local retries are done if an Ack is not received by the transmitting node. Default value is 10, and possible values are 0…30.

3.4.4.5 Beacon Rate

ID:18
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].

3.4.4.6 Beacon Full Scan Rate

ID: 19
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.

3.4.4.7 Beacon Full Scan Rate Initial

ID: 20
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.

3.4.4.8 Beacon Full Scan Count Initial

ID: 21
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.

3.4.4.9 Beacon TX Initial Hold Down Time

ID: 22
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.

3.4.5 Neighbour acquisition settings

3.4.5.1 Maximum number of neighbours

ID: 49
As the number of neighbours directly influences the average current consumption, it is advisable to limit the number of neighbours a node will accept. The setting can be anything between 2 and 12. The recommended setting is 6. This ensures that a certain section of the network does not close around itself, and does not create islands in the network.

3.4.5.2 Node Inclusion Increment Speed

ID: 28
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.

3.4.5.3 Node Inclusion Hysteresis

ID: 29
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.

3.4.6 Payload messaging & routing settings

3.4.6.1 Payload Aggregation Limit

ID: 73
Aggregation allows more than one payload message to be transmitted at the Transmit Scheduled Data event (see section 3.3.4.7.3 for further details).
This setting limit how many payload messages which will be aggregated.
Allowed values are 1 … 5.

3.4.6.2 High Accuracy Package Age – HAPA

ID: 45
When enabled (value = 1), the package age is given with a greater accuracy than normal (value = 0). 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 uses 4 bytes from the payload data field, such that 17 bytes are available for payload data, as opposed to 21 bytes when HAPA is disabled.

3.4.6.3 Network Maximum Roundtrip Time

ID: 65
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 125 milli seconds and defined by a 2 byte value. Valid values are [32..992] seconds, parameter setting [256..7.936].

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 3.4.6.4 for more details on the Payload Data Time To Live – TTL parameter.

3.4.6.4 Payload Data Time to Live – TTL for Acknowledged Payload

ID: 66
TTL is the maximum age a globally retried acknowledged 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 125 milli seconds, and valid values are [32..992] seconds, parameter setting [256..7.936].

3.4.6.5 Payload Data Time to Live – TTL for unacknowledged Payload

ID: 67
Similar to 3.4.6.4, but applicable for unacknowledged payload transmission including broadcast messages. TTL time resolution is 125 milli seconds, and valid values are [32..992] seconds, parameter setting [256..7.936]. When a unacknowledged payload package reaches TTL it will be removed from the network.

3.4.6.6 Routing Field Strength Threshold

ID: 30
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. The value is in -dBm and should be set such that there is a margin to the receiver sensitivity level for the particular module (see module data sheet). The default value is optimal in most cases.

3.4.7 Hibernation and Alternate mode settings

3.4.7.1 ACM ID

ID: 56
The Node ID of the node which is controlling the mode of the network (Alternate Configuration Master). Value 0xFFFF is default and corresponds to disabling the ACM functionality. See section 3.3.10 for further details.

3.4.7.2 Alternate Mode Config

ID: 57

This parameter controls the behaviour of the Alternate Mode feature. The value is 1 byte, with the following meaning:

Bit	Function
7	N/A
6	**Hibernation behaviour in Alternate Mode:** 1: Enter hibernation state when there is no Route to the ACM. 0: Do not hibernate even if there no Route to the ACM.
5	**Generic Application behaviour in Alternate Mode** 1: Generic Application will use the setting for Alternate Mode 0: Generic Application will use the setting for Normal Mode
4,3	**ACM Presence Behaviour** Depending on the whether the ACM node is present in the network or not, the node is behaving according to the table below: 00: ACM presence does not affect Alternate Mode setting 01: Use "Default Alternate Configuration Mode" (Bit 1 setting) if ACM is not present. 10: N/A 11: N/A
2	**Force Hibernation.** 1: Hibernation mode will be entered on startup 0: Normal operation
1	**Default Alternate Configuration Mode.** 1: Node is in Alternate Mode when starting up. 0: Node is in Normal Mode when starting up. Mode will be overwritten by Alternate Configuration Master once a network connection is established.
0	**Hibernation enabled** 1: The node will hibernate if the condition for hibernation is met. 0: The node will not hibernate even of the condition for hibernations is met.

NOTE: Currently the node will transmit Wake Up bursts forever when in Alternate Mode.

3.4.7.3 ACM Presence Time Out to Hibernation

ID: 60
A multiple of Scheduled Data periods defining how long time it takes before the node enters hibernation mode after ACM is no longer present in the network, if 0 timeout is disabled.

3.4.7.4 Wake Up Burst count on Start Up

ID: 61
Defines how many wake up bursts the node will transmit after start up.
Possible values are 0 to 255:

Value	Behaviour
0	No Wake Up burst are transmitted
255	Depends on setting 3.4.7.1 ACM ID. If ACM ID is 0xFFFF, no bursts are send. If ACM ID is not 0xFFFF, bursts are send as long as the ACM is present in the network
1-254	A number of wake up bursts will be send corresponding to the value set.

3.4.7.5 Wake Up Burst length

ID: 62
Defines the length of the wake up burst transmitted to wake up hibernating nodes.
The values define the maximum length in seconds. The actual duration depends on the number of neighbours. Default value is 20, and should be appropriate in most networks.
Possible values are 1 to 255.

3.4.7.6 Max Wake Up Burst period

ID: 63
Defines the maximum period between two Wake Up bursts. The actual period will be a randomized number between 3 and the setting value. Value is in steps of 2 seconds.

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

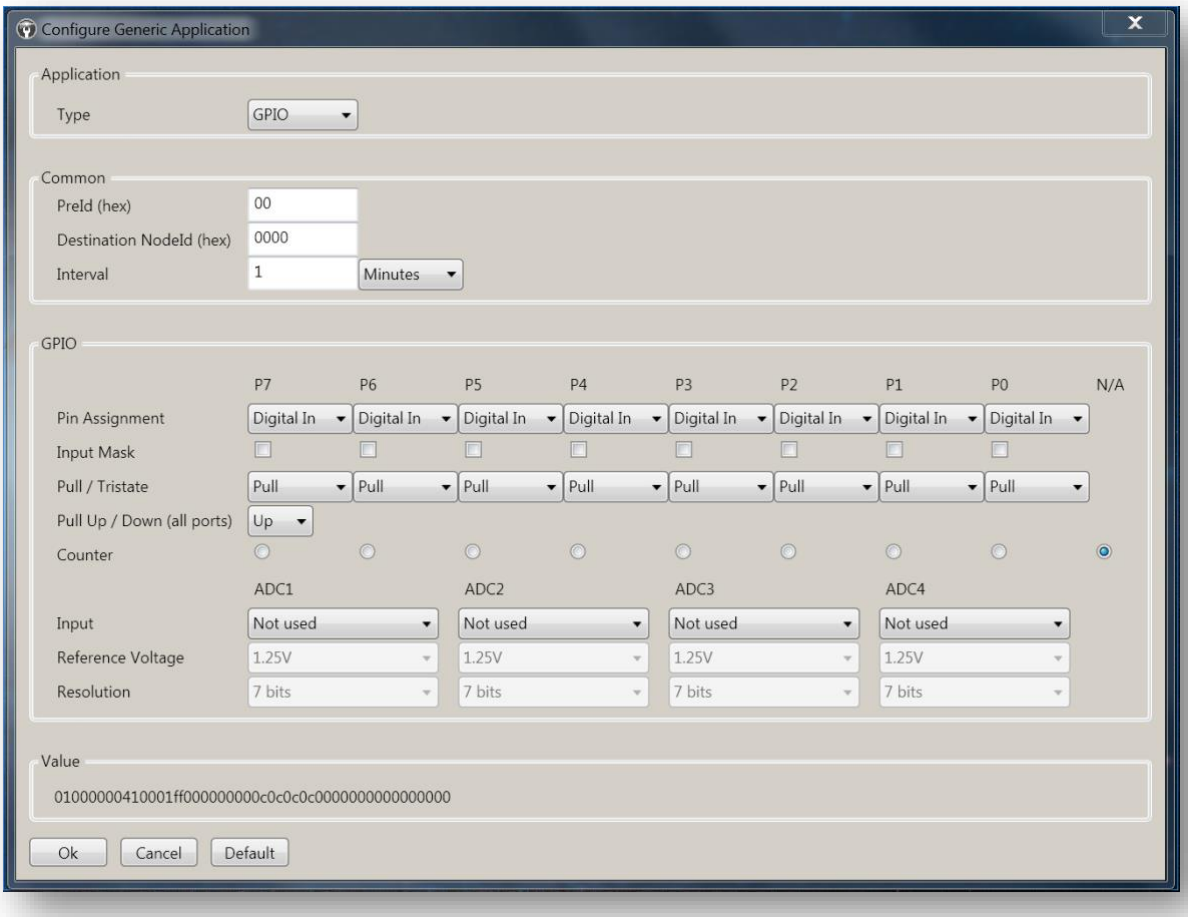
The details of each application type are described in the following sections.

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:



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 2 byte NeoMesh 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.
If Digital Output is selected, it is possible to set the state of the pins by sending a payload message to the node where the first byte of the payload data will be used as P0..7.

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 2 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 2 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.

4.7 Finding more information and support

In addition to this section on how the application layer can be implemented, the Neocortec website neocortec.com is a great resource of information. In particular the Quick Start Guide and the Video Tutorials offer simple access to advanced information on this topic.

Should you have questions which are not covered in this document or other places, please feel free to use the support channel at support@neocortec.com .

-- End of document --

NeoCortec A/S
Tonsbakken 16-18
2740 Skovlunde
Denmark
www.neocortec.com