UNIVERSIDADE DA CORUÑA

ESCUELA POLITÉCNICA SUPERIOR

TRABAJO FIN DE MÁSTER

CURSO 2016/2017

DISEÑO Y DESARROLLO DEL SISTEMA DE COMUNICACIONES INALÁMBRICO DE UN ENJAMBRE DE UAV'S COLABORATIVOS

MÁSTER EN INGENIERÍA INDUSTRIAL

ALUMNO

Cristian Méndez Sanmartín

TUTORES

Francisco Javier Bellas Bouza

Félix Orjales Saavedra

FECHA

Junio 2017

TÍTULO	Diseño y Desarrollo del Sistema de Comunicaciones Inalámbrico de un Enjambre de UAV's Colaborativos.
AUTOR	Cristian Méndez Sanmartín
TUTORES	Francisco Javier Bellas Bouza
	Félix Orjales Saavedra
GRUPO DE INVESTIGACIÓN	Grupo Integrado de Ingeniería (GII)
LÍNEA DE INVESTIGACIÓN	Inteligencia Computacional (IC)
ÁREA DE TRABAJO	Inteligencia Colectiva y Sistemas de Comunicaciones.
TITULACIÓN	Máster en Ingeniería Industrial
FECHA DE ENTREGA	19 de Junio de 2017
IDIOMA	Castellano

RESUMEN DEL TRABAJO:

La Inteligencia Colectiva, dentro del campo de la Inteligencia Computacional, aborda temas de estudio como el comportamiento de los agentes individuales dentro de un sistema global y la asociación de la información para la resolución conjunta de problemas complejos.

Dentro de la línea de investigación de Inteligencia Computacional e Inteligencia Colectiva, el Grupo Integrado de Ingeniería (GII) ha propuesto como proyecto de estudio el abordar un análisis exhaustivo de las tecnologías IoT de comunicación existentes, con el fin de adoptar la más adecuada a los requerimientos e incorporarla a un modelo de enjambre de UAV's colaborativos.

Las características más valoradas a la hora de la adopción del sistema serán las inherentes a los UAV's, como son el alcance o la eficiencia energética, además de obtener las mejores prestaciones a la hora de establecer el radioenlace, como obtener el máximo ancho de banda, optimización de latencias y ciclos de CPU y la mayor inmunidad posible frente a interferencias.

Una vez elegido el sistema, se tratará de optimizar para su uso en UAV's y se realizarán pruebas de vuelo en escenarios reales para obtener datos específicos sobre consumo energético, rendimiento y alcance máximo.

RESUMO DO TRABALLO:

A Intelixencia Colectiva, dentro do campo do campo da Intelixencia Computacional, aborda temas de estudo como o comportamento dos axentes individuais dentro dun sistema global e a asociación da información para a resolución conxunta de problemas complexos.

Dentro da liña de Investigación da Intelixencia Computacional e Intelixencia Colectiva, o Grupo Integrado de Enxeñaría (GII) propuxo como proxecto de estudo abordar un análise exhaustivo das tecnoloxías IoT de comunicacións existentes, coa fin de adoptar a máis adecuada ós requirimentos e incorporala a un modelo de enxame de UAV's cooperativos.

As características mais valoradas á hora da adopción do sistema serán as inherentes ós UAV's, como son o alcance ou a eficiencia enerxética, ademais de obter as mellores prestacións á hora de establecer o radioenlace, como obter o máximo ancho de banda, optimización de latencias e ciclos de CPU e a maior inmunidade posible fronte as interferencias.

Unha vez elixido o sistema, tratarase de optimizar para o seu uso en UAV's e realizaranse probas de voo en escenarios reais para obter datos específicos sobre consumo enerxético, rendemento e alcance máximo.

ABSTRACT:

Collective Intelligence, into the field of Computational Intelligence, approach topics of study such as the behaviour of individual agents in global systems and the association of information for common resolution of complex issues.

In the research line of Computational Intelligence and Collective Intelligence, the Integrated Engineering Group (GII) has proposed a study project to approach a comprehensive analysis of existing IoT communication technologies to adopt the most appropriate to the requirements and incorporate it into a swarm model of collaborative UAV's.

The most valued characteristics for the adoption of the system will be those inherent to the UAV's, such as range or energy efficiency, moreover to obtain the best benefits when stablishing the radio link, as obtain the maximum bandwidth, optimization of latencies and CPU cycles and highest immunity possible to interference.

Once the system is chosen, it will be optimized for use in UAV's and flight test will be carried out in real scenarios to obtain specific data on energy consumption, performance and maximum range.

UNIVERSIDADE DA CORUÑA

ESCUELA POLITÉCNICA SUPERIOR

TRABAJO FIN DE MÁSTER

CURSO 2016/2017

DISEÑO Y DESARROLLO DEL SISTEMA DE COMUNICACIONES INALÁMBRICO DE UN ENJAMBRE DE UAV'S COLABORATIVOS

MÁSTER EN INGENIERÍA INDUSTRIAL

DOCUMENTO DEL PROYECTO

MEMORIA

ÍNDICE DE CONTENIDOS

ÍNDICE DE FIGURAS

ÍNDICE DE DIAGRAMAS

ÍNDICE DE ECUACIONES

ÍNDICE DE GRÁFICAS

ÍNDICE DE ILUSTRACIONES

ÍNDICE DE TABLAS

1 INTRODUCCIÓN

La **Inteligencia Artificial** (IA) está enfocada a comprender la lógica del pensamiento humano, en cómo pensamos o decidimos y en la construcción de entidades inteligentes. Este proceso resulta muy complicado debido que en la actualidad no se tiene un entendimiento completo de cómo funciona la mente humana y los procesos que esta conlleva. El campo de estudio abarca una gran variedad de aplicaciones, desde la demostración de teoremas matemáticos, implementación de algoritmos de búsqueda, toma de decisiones y tareas de carácter intelectual.

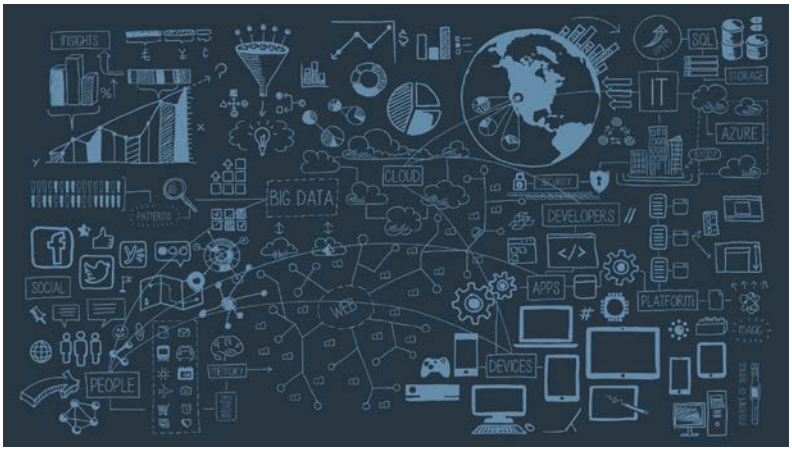


Ilustración 1.1: Representación Abstracta de IA.

La **Inteligencia Computacional** (IC) es adoptada en el estudio de ciertas disciplinas que pertenecen a la Inteligencia Artificial. Se introducen las redes de neuronas artificiales (RNA's), la computación evolutiva, la lógica difusa, la inteligencia en enjambre y los sistemas inmunes artificiales.

Las metodologías y aproximaciones de la Inteligencia Computacional permiten el desarrollo de sistemas adaptativos que combinan ciertas técnicas de aprendizaje, generalización, abstracción y asociación de información para resolución de problemas. En el campo de Inteligencia Colectiva se pretende obtener sistemas donde la inteligencia no resida en el comportamiento de un agente individual, sino en el comportamiento del sistema de forma global.

Una de las bases de la Inteligencia Artificial consiste en el desarrollo de avanzados **Sistemas de Comunicaciones**. El Grupo Integrado de Ingeniería ha propuesto como proyecto el análisis de las tecnologías de comunicación existentes, con el fin de adoptar la más adecuada e incorporarla a un modelo de enjambre de UAV's colaborativos.

A lo largo de esta memoria veremos el proceso de análisis y optimización para finalmente mostrar su aplicación con pruebas de vuelo en escenarios reales.

2 OBJETIVOS

El presente TFM se enmarca dentro de la línea de investigación sobre los UAV (Unmanned Aerial Vehicles) del GII (Grupo Integrado de Ingeniería). El objetivo global de la línea es el desarrollo de un enjambre consistente en varios UAV que puedan volar de forma autónoma y colaborar entre sí para realizar una tarea en común.

Las comunicaciones entre los UAV del enjambre, la estación de tierra y el PC de supervisión se realizarán de forma inalámbrica. Para las labores de supervisión, se enviarán periódicamente los datos de la telemetría de todos los UAV del equipo a la estación de tierra.

Los objetivos marcados para este proyecto serán los siguientes:

- Análisis de la capacidad del sistema de comunicaciones empleado en un enjambre de UAV's Colaborativos.
- Optimización de la configuración del sistema para obtener el máximo ancho de banda posible, minimizando el uso de CPU, los paquetes perdidos y su latencia.
- Utilización en medida de lo posible de software libre durante el desarrollo del proyecto.

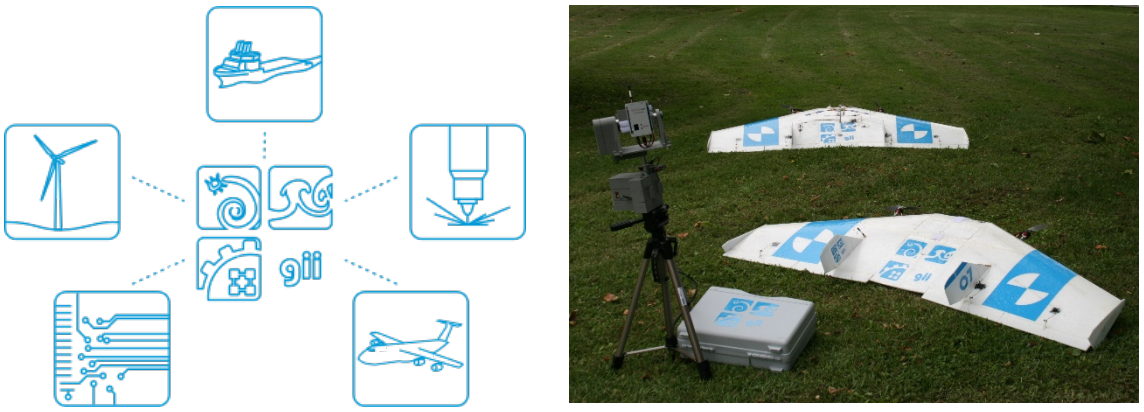


Ilustración 2.1: Grupo Integrado de Ingeniería (GII).

3 ESTADO DEL ARTE

3.1 INTRODUCCIÓN

La comunicación es la transferencia de información con un sentido desde un lugar (origen, fuente, transmisor) a otro lugar (destino, receptor). La información transferida (mensaje), es un patrón físico al cuál se le ha asignado un significado acordado y puede presentarse de diferentes formas:

- Secuencias de símbolos.
- Impulsos eléctricos.
- Presión.

El mensaje debe ser único, capaz de ser enviado por el transmisor y ser detectado y entendido por el receptor. Los sistemas de comunicación nos brindan los medios necesarios para que la información codificada en forma de señal, se transmita o intercambie.

En el siguiente diagrama se muestra un modelo básico de un sistema de comunicaciones, en el que se muestran los principales componentes que permiten la comunicación:

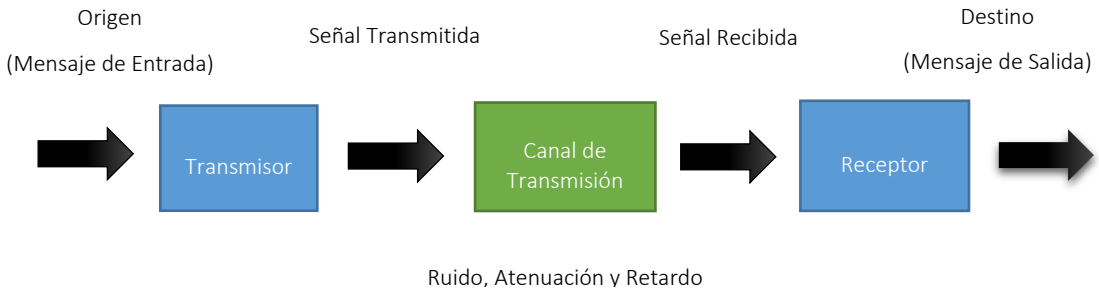


Diagrama 3.1.1: Sistema de Comunicaciones.

El mensaje original, producido por la fuente, no es eléctrico, y este debe ser convertido en señales eléctricas a través de un transductor de entrada. En el destino, otro transductor de salida cumple la función de transformar nuevamente la señal para que llegue al receptor del modo en el que fue emitido el mensaje.

Pese a las diferentes vías en las que es posible la transmisión de la información, la electricidad es la encargada de trasladar el mensaje, en forma de señal. Las ondas electromagnéticas permiten la transmisión de mensajes o señales.

3.2 TEORÍA DE ONDAS ELECTROMAGNÉTICAS

El físico alemán Heinrich Rudolf Hertz, en 1887, demostró que la electricidad puede transmitirse en forma de ondas electromagnéticas. Se basó en la teoría de James Clerk Maxwell quien afirmó que las oscilaciones eléctricas pueden propagarse por el espacio.

Ilustración 3.2.1: Heinrich Rudolf Hertz – James Clerk Maxwell.

Estas se difunden en el espacio de modo similar al movimiento del agua en un estanque, tal como puede observarse al arrojar en él una piedra y se desplazan a 311.000.000 m/s en el vacío. Pero cuando atraviesan materiales de diferente densidad, su velocidad se verá afectada, decreciendo en función de la densidad de los materiales interpuestos.

Los campos eléctricos y magnéticos de estas ondas vibran en un plano generalmente horizontal o vertical. Estos campos son perpendiculares entre sí y su dirección de propagación es también perpendicular a los mismos.

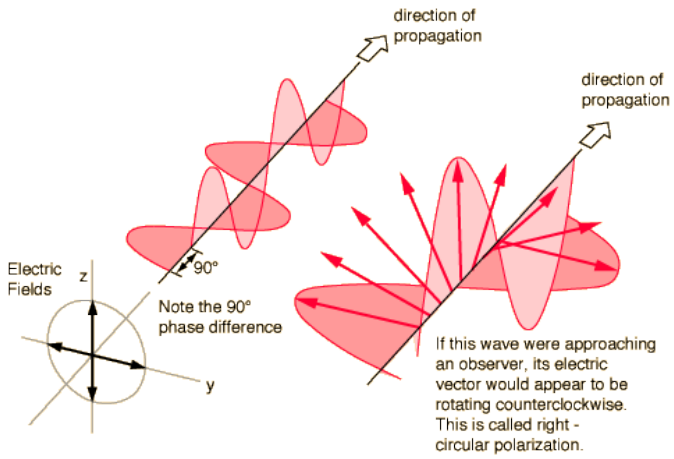


Ilustración 3.2.2: Ondas Electromagnéticas.

Las ondas electromagnéticas se caracterizan por las siguientes magnitudes:

- Amplitud: es la magnitud que mide la variación máxima que presenta una onda periódica a lo largo del tiempo.
- Frecuencia: es la magnitud que mide el número de oscilaciones o ciclos por unidad de tiempo. La medida adoptada por el Sistema Internacional es el Hertzio (Hz).
- Longitud de Onda (λ): es la magnitud que mide la distancia que recorre una onda en un tiempo igual al período. La medida adoptada por el Sistema Internacional para ondas electromagnéticas es el metro (m).

El espectro electromagnético abarca el rango de todas las ondas electromagnéticas, ahora bien, no todas las ondas electromagnéticas son propicias para usarse como medios de transmisión, de forma que sólo las que se encuentran en determinado rango serán susceptibles de ser empleadas.

El espectro radioeléctrico será la subdivisión donde operan los servicios de comunicaciones.

Ilustración 3.2.3: Espectro Electromagnético.

Debido a la cantidad de servicios que se pueden prestar por medio del espectro radioeléctrico, su organización y regulación resulta imprescindible para permitir el desarrollo del mismo. Para la correcta administración del espectro radioeléctrico, este se subdivide en bandas de frecuencia, que designan porciones del espectro radioeléctrico atendiendo a criterios técnicos relacionados con los servicios que, por las características propias, resultan viables en determinada banda.

El Cuadro Nacional de Atribución de Frecuencias (CNAF) es el instrumento legal, dependiente del Ministerio de Industria, Energía y Turismo de España, utilizado para asignar a distintos servicios de radiocomunicaciones las diferentes bandas de frecuencias, estas bandas se extienden desde aproximadamente 8,30 kHz hasta 3.000 GHz (véase en Anexo de Normativa).

3.3 TRANSMISIÓN DE ONDAS

Las ondas radioeléctricas pueden transmitirse sin necesidad del uso de cables de un lugar a otro y dependiendo de la frecuencia o longitud de onda, éstas viajan de diferentes maneras. Las de baja frecuencia, por ejemplo, no siguen el mismo curso que las que tiene altas o muy altas frecuencias. Podemos dividir las ondas en tres tipos dependientes de las formas de propagación:

- Ondas Terrestres o de Superficie:

Son ondas que en parte se desplazan pegadas a la corteza terrestre. Al ir tan cerca del suelo, las características de éste influyen bastante en su forma de propagación. Viajan incómodas sobre suelos secos, como el desierto, y recorren mayores distancias si el terreno es húmedo, porque les ofrece mejor conductividad. Las ondas que se propagan de esta forma no se despegan de la tierra. Presentan como ventaja que no le afectan mucho los obstáculos, pero al mismo tiempo esto es un inconveniente ya que este roce las va atenuando o desgastando.

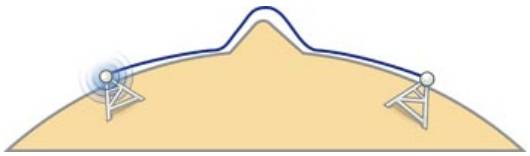


Ilustración 3.3.1: Emisión de Ondas Terrestres o de Superficie.

- Ondas Reflejadas o Ionosféricas:

Son ondas que se proyectan contra la atmósfera, pero se encuentran con un escudo, una capa de la misma, llamada ionosfera que, por sus características, actúa como un espejo y las rebota devolviéndolas a la tierra, tras esto, las ondas rebotan contra la corteza terrestre y vuelven a subir repitiéndose el proceso sucesivamente.

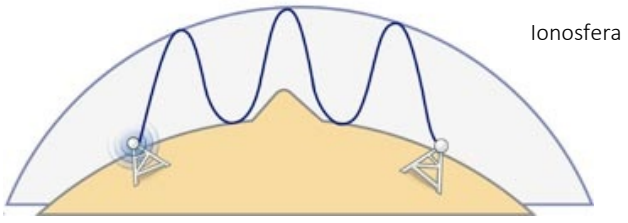


Ilustración 3.3.2: Emisión de Ondas Reflejadas o Ionosféricas.

La ionosfera está situada entre 60 y 400 km de la corteza terrestre. Dependiendo de la hora del día y las condiciones de la atmósfera, sus características pueden cambian drásticamente. Esto hace que las radiocomunicaciones de esta clase varíen mucho en función de la estación del año o del momento del día. El invierno y las horas nocturnas son más beneficiosas cuando, por la falta de rayos solares, la capa se vuelve más densa y se aleja de la tierra permitiendo a las ondas llegar más lejos. La mayoría de las ondas no eligen un solo camino para propagarse, sino que estas realizan una combinación de este tipo de transmisión y la anterior.

- Ondas Directas o Espaciales:

Son ondas de alta frecuencia que tienen longitudes de onda muy pequeñas. Se realiza su transmisión en línea recta, hasta donde alcanza la vista. Son muy vulnerables a los obstáculos, incluso la misma curvatura de la tierra hace que se pierda la señal. Por esta limitación, las antenas transmisoras siempre se colocan en lugares elevados para no perder la línea de visión.

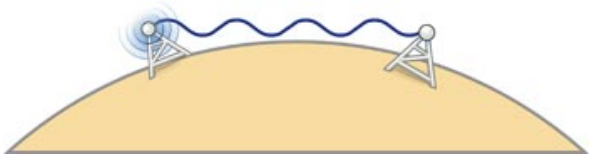


Ilustración 3.3.3: Emisión de Ondas Directas.

Si aumentamos la frecuencia y la potencia dirigiendo las antenas hacia lugares donde no haya nada, como el espacio, alcanzamos distancias muy superiores. Estas ondas no son reflejadas en la ionosfera y la traspasan, viajando miles de kilómetros. Son las encargadas de mandar señales a los satélites para transmisiones de largo alcance que luego regresan rebotadas.

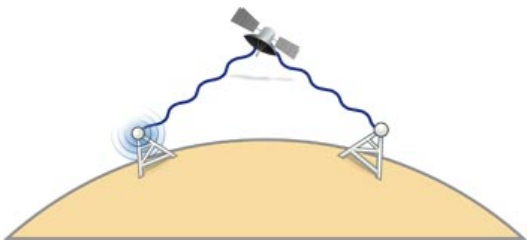


Ilustración 3.3.4: Emisión de Ondas Espaciales.

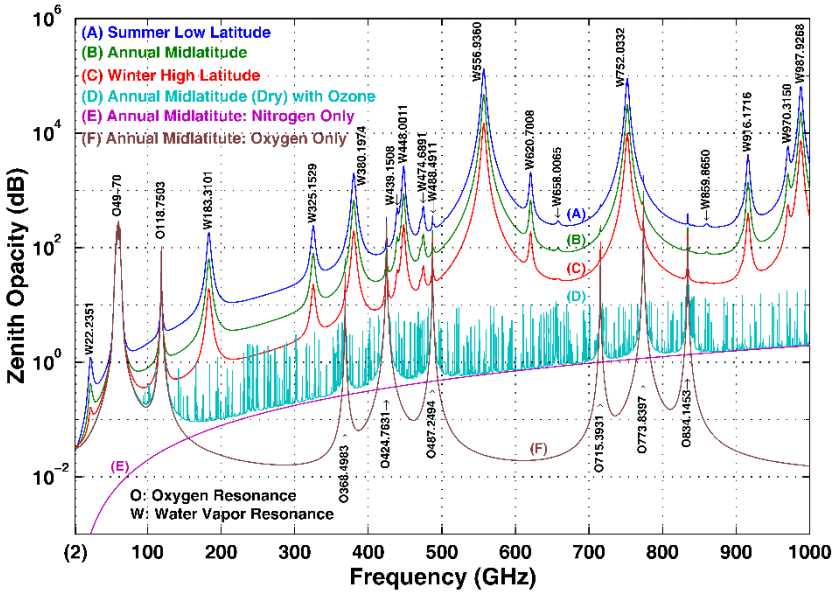
Estos satélites están situados en la "Órbita de Clarke" (Órbita Geoestacionaria), de excentridad nula y movimiento de Oeste a Este, siendo vistos desde la tierra como un objeto inmóvil en el cielo, situándose a una distancia sobre el nivel del mar de 35.786 kilómetros.

Como se ha comentado, la mayoría de las ondas no eligen un solo camino para propagarse, esto está provocado por las interacciones con objetos en el ambiente que, a su vez, están asociados con el método de propagación. Esto es provocado por los siguientes fenómenos:

- Absorción:

Cuando las ondas radioeléctricas atraviesan materiales, se debilitan y se atenúan. La cantidad de potencia perdida dependerá de la frecuencia y del material. La potencia decrecerá de manera exponencial y la energía absorbida se transformará en calor.

En redes de comunicación, los metales y el agua son considerados absorbentes perfectos, no van a poder ser atravesados (salvo que estén presentes como capas finas, lo cual permitiría que una pequeña porción de potencia los traspase). Cuando se habla del agua, es importante tener en cuenta que se puede encontrar en diferentes formas como lluvia, niebla, vapor o nubes bajas.



Gráfica 3.3.1: Pérdidas de por Absorción en Oxígeno o Vapor de Agua.

(Recomendación UIT -R P676)

Además, existen otros materiales que tienen efectos más complejos en la absorción de radiación, como la madera, dependiente de la cantidad de agua y resto de compuestos químicos.

- Reflexión:

La reflexión de las ondas radioeléctricas ocurre cuando una onda incidente choca con una barrera existente y parte de la potencia incidente no penetra en el mismo. Debido a que todas las ondas reflejadas permanecen en el mismo medio que las ondas incidentes, sus velocidades son iguales y por lo tanto el ángulo de reflexión es el mismo que el de incidencia.

- Difracción:

La difracción ocurre cuando la trayectoria que siguen las ondas radioeléctricas entre el emisor y el receptor se encuentra obstruida por una superficie con irregularidades. Las ondas que resultan tras la superficie de obstrucción se desvían en el espacio y flectan alrededor del obstáculo.

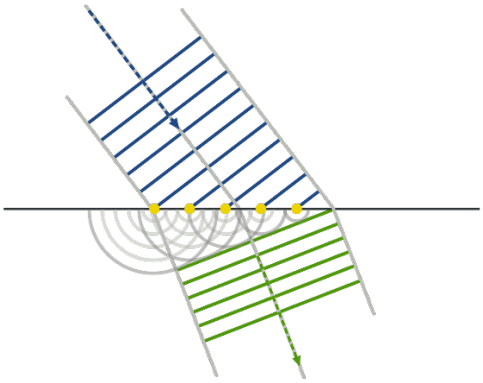


Ilustración 3.3.5: Fenómeno de Difracción. Principio de Fresnel - Hüygens.

El "Principio de Fresnel - Hüygens" es enunciado para comprender este comportamiento:

> "Todo punto de un frente de onda inicial puede considerarse como una fuente de ondas esféricas secundarias que se extienden en todas las direcciones con la misma velocidad, frecuencia y longitud de onda que el frente de onda del que proceden."

En la difracción se genera una pérdida de potencia de transmisión, siendo la potencia de la onda difractado mucho menor que el frente de onda que la provoca. Este fenómeno se da en todas las frecuencias, a menor frecuencia mayor difracción, pero en contraposición, a mayor frecuencia mayor pérdida.

- Dispersión:

La dispersión ocurre cuando el medio utilizado para la transmisión de las ondas radioeléctricas está ocupado por una gran cantidad de obstáculos de dimensiones pequeñas comparables a la longitud de onda (λ) o donde existe un gran número de obstáculos por unidad de volumen. Las ondas dispersadas son irradiadas en un gran número de direcciones diferentes, siendo estas añadidas a interferencias destructivas de la señal.

- Refracción:

La refracción es manifestada como el cambio de dirección de una onda radioeléctrica conforme pasa oblicuamente de un medio a otro con diferentes velocidades de propagación.

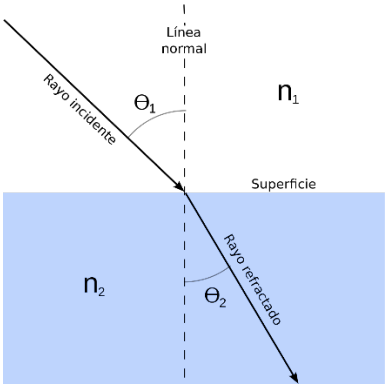


Ilustración 3.3.6: Fenómeno de Refracción.

Por lo tanto, la refracción ocurre siempre que la onda pasa de un medio a otro que presente diferente densidad. El ángulo de incidencia es el formado entre la onda incidente y la normal y el ángulo de refracción el formado entre la onda refractada y la normal.

- Multitrayectoria (Multipath):

El efecto multipath ocurre debido a que las ondas radioeléctricas llegan al receptor a través de diferentes caminos y a diferentes tiempos. Esto provoca retardos e interferencia en los sistemas de comunicaciones, acrecentándose en transmisiones interiores. Las ondas recibidas pueden ser directas desde la línea de visión, ondas reflejadas, ondas dispersadas u ondas difractadas.

El sistema de comunicación debe ser diseñado para que este efecto sea reducido al mínimo. Existen dos clases de efecto multitrayectoria:

- Multitrayectoria Especular: provoca un nivel de ruido de fondo de interferencia.
- Multitrayectoria Difusa: provoca interrupciones de la señal o puntos muertos.

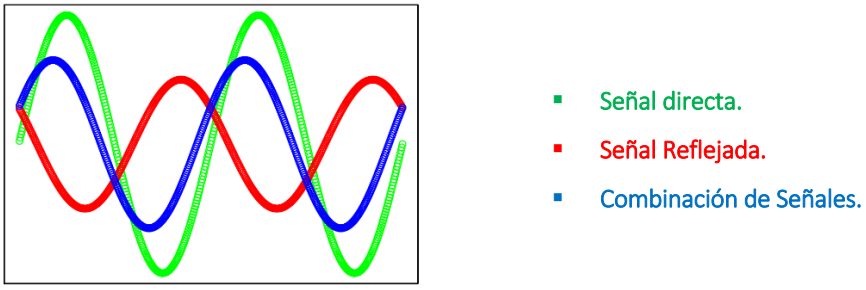


Ilustración 3.3.7: Efecto Multipath.

3.4 FACTORES QUE AFECTAN LA CALIDAD DE LA SEÑAL

Además de los efectos explicados en el subcapítulo 3.3 durante la transmisión de señal existen una serie de factores que afectan a la calidad de las ondas radioeléctricas transmitidas por lo que nunca serán iguales a las recibidas. Las principales son:

- Atenuación: efecto de desgaste por el cual la señal presenta una pérdida de potencia ocasionada por la distancia entre emisor y receptor. Esta aumenta con la frecuencia, la temperatura y el tiempo. Es medida en decibelios por unidad de distancia (dB/m).

$$\alpha = 10log\frac{P_1}{P_2}$$

Ecuación 3.4.1: Atenuación de Potencia.

- Distorsión: efecto de deformación que experimenta una señal al ser transmitida por un canal por la respuesta imperfecta del sistema, desaparece al dejar de aplicar la señal.
- Interferencia: efecto de contaminación de señal causado por señales extrañas al sistema originadas de forma artificial. Esta es conocida como EMI (ElectroMagnetic Interference) o RFI (Radio Frequency Interference).
- Ruido: efecto de contaminación de señal causado por señales aleatorias e impredecibles originadas de forma natural. Estas señales pueden agregarse a la señal original portadora alterándola y quedando ocultas, no pudiendo ser eliminado completamente. Existen numerosos tipos de ruido, siendo algunos de ellos los siguientes:
    - Ruido Atmosférico: es producido por las descargas atmosféricas que se propagan en todas las direcciones por reflexión ionosférica.
    - Ruido Galáctico: es producido por disturbios o perturbaciones producidas fuera de la atmósfera. La fuente primaria es el sol y este tipo de ruido tiene su base en la teoría de la "Emitancia Espectral de Planck".

$$M(\lambda, \mathrm{T}) = (a/\lambda^5)(e^{b/\lambda \mathrm{T}} - 1)$$

Ecuación 3.4.2: Emitancia Espectral de Planck.

    - Ruido Térmico: es producido por la agitación térmica de los electrones presente en las resistencias y se manifiesta en toda la naturaleza. Solo en condiciones de Zero Absoluto (0 K) cesaría, por tanto, siempre va a existir y es inevitable.

3.5 TECNOLOGÍAS DE RADIOFRECUENCIA

Existen dos tipos de tecnología de radiofrecuencia para transmitir la información:

- Tecnología de Banda Estrecha (Narrowband):

El fundamento básico es una modulación de la señal a transmitir en un rango muy reducido de frecuencias. Debido a ese reducido ancho de banda, la velocidad de transmisión se ve disminuida.

Ilustración 3.5.1: Operadores Narrowband.

$$R_b(\text{bit/s}) = BW\ log(1 + {}^{S}/_{R})$$

Ecuación 3.5.1: Teorema Shannon - Hartley.

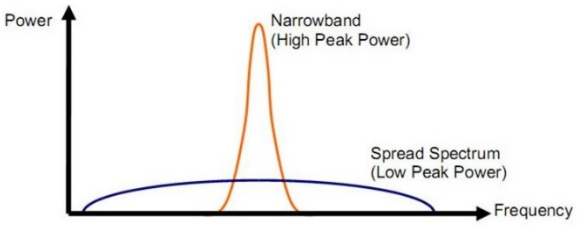
- Tecnología de Banda Ancha (Spread Spectrum):

El fundamento básico es una modulación por ensanchamiento de la señal a transmitir a lo largo de una banda muy ancha de frecuencias. La señal de espectro ensanchado, puede coexistir con señales en banda estrecha, ya que sólo les aportan un pequeño incremento en el ruido.

Ventajas	Desventajas
Resistente a gran parte de interferencias.	Ineficiencia del uso del ancho de banda.
Eliminación o atenuación del efecto multipath.	Complejidad en implementación.
Posibilidad de compartir banda.	

Tabla 3.5.1 Características Spread Spectrum.

Ilustración 3.5.2: Operadores Spread Spectrum.



Gráfica 3.5.1: Narrowband vs Spread Spectrum.

3.6 DISEÑO DE UN SISTEMA DE COMUNICACIONES

En sistemas de comunicación, las características del diseño de instalación se pueden transformar en obstáculos capaces de arruinar nuestra comunicación, por ello debemos abordar con criterio los siguientes aspectos fundamentales:

- Potencia de Radiofrecuencia:

Es medida por lo general en Watts (W) o miliwatts (mW) aunque en muchas aplicaciones, también se encuentran referencias a esta magnitud expresada en decibelios por metro (dBm). La relación que existe entre ambas formas sigue la siguiente fórmula:

$$dBm = 10logP$$

Ecuación 3.6.1: Relación P/dBm.

Una cuestión a tener en cuenta es el número de antenas, estas pueden sumarse en un número progresivo en base 2, por lo tanto, al duplicar el número de antenas duplicamos la potencia.

Ilustración 3.6.1: Antena de Radiofrecuencia.

- Relación Frecuencia-Propagación:

La gran mayoría de las aplicaciones industriales, domésticas, científicas y de investigación, suelen trabajar con sistemas de transmisión que operan en las Bandas Libres.

Las organizaciones que administran el uso racional del espectro radioeléctrico de cada país asignan porciones o bandas de frecuencias donde no se hace necesario solicitar una autorización, abonar cánones o impuestos para trabajar de forma libre dentro de sus límites.

Estas bandas son conocidas también como Bandas ISM (Industrial, Scientific and Medical) y los valores más populares son:

- Bandas de 2,40/ 5,80 GHz (Global).
- Bandas de 433/868 MHz (Europa).
- Banda de 915 MHz (América).

La gran ventaja de poder trabajar en frecuencias cada vez más altas es que el ancho del canal a utilizar aumenta. Esto significa tener la posibilidad de transmitir una mayor cantidad de datos o información dentro de un canal único.

La desventaja de esto es que la distancia a enlazar y la capacidad de la señal para superar obstáculos decrece de manera notable al aumentar la frecuencia de transmisión. A su vez, el trabajar a frecuencias bajas requieren de antenas de mayor tamaño, mientras que, a frecuencias mayores, se pueden colocar sistemas de antenas de mayor ganancia para suplir las pérdidas mencionadas anteriormente.

- Optimización de las Antenas:

La optimización constructiva de las antenas permitirá obtener unas ganancias significativas para garantizar una transmisión óptima.

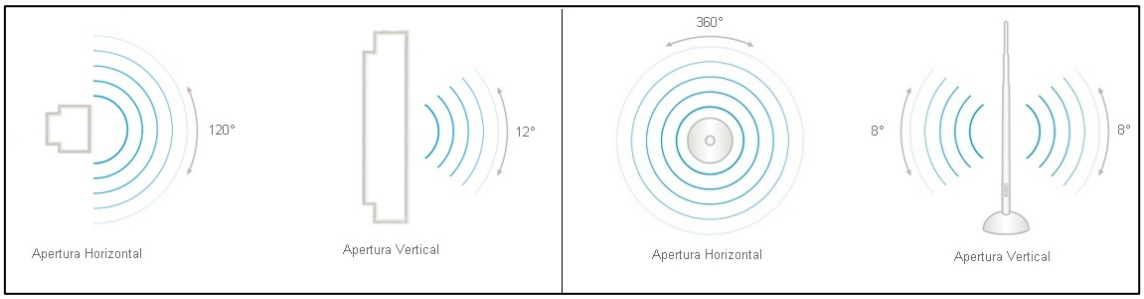


Ilustración 3.6.2: Antena Sectorial – Antena Omnidireccional.

Las antenas unidireccionales son capaces de enfocar toda la potencia que llega a ellas en una única dirección. A su vez, una antena omnidireccional emitirá hacia todos los sentidos, perdiendo la concentración de señal. Cuando colocamos las antenas de los sistemas en un espacio abierto y libre de obstáculos entre ellas, la atenuación provocada por el medio se incrementa con una relación del cuadrado de la distancia. La atenuación provocada por otros obstáculos incrementa también notablemente las pérdidas.

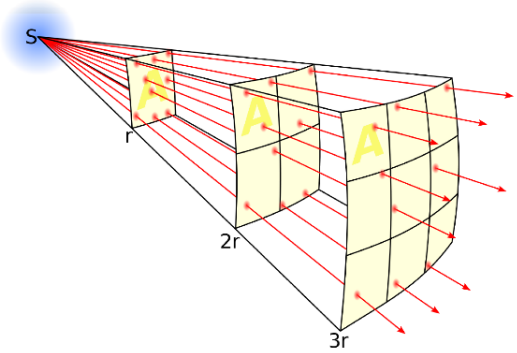


Ilustración 3.6.3: Pérdidas en el Espacio Libre.

$$FSPL = \left(\frac{4\pi d}{\lambda}\right)^2 = \left(\frac{4\pi df}{c}\right)^2$$

$$FSPL\ (dB) = 20\log(d) + 20\log(f) - 147{,}55$$

Ecuación 3.6.2: Pérdidas en el Espacio Libre.

(Modelo de Friis – Región de Fraunhöfer)

- Sensibilidad en Receptores:

La sensibilidad en un receptor viene dada por la capacidad que pueda tener de recuperar señales muy débiles. Esto significa que cuanto mayor sea la cifra expresada en dBm (en valores negativos), más sensible será el receptor y mayor posibilidad de recuperar datos correctos y realizar una transmisión exitosa.

$$Sensitivity\ (dB) = -174 + 10\log(BW) + NF + SNR$$

Ecuación 3.6.3: Sensibilidad del Receptor.

En la actualidad los receptores utilizan transistores de entrada con un nivel de figura de ruido (NF) tan bajo como esto sea posible. Algunos suplen estos defectos reduciendo la velocidad de transmisión de datos o mejorando la calidad de las antenas de recepción, cuando en realidad lo que se debe mejorar es la etapa de entrada de la señal de RF en el receptor. La figura de ruido en dispositivos electrónicos es generada por la agitación electrónica dentro del semiconductor, provocada por la temperatura ambiente.

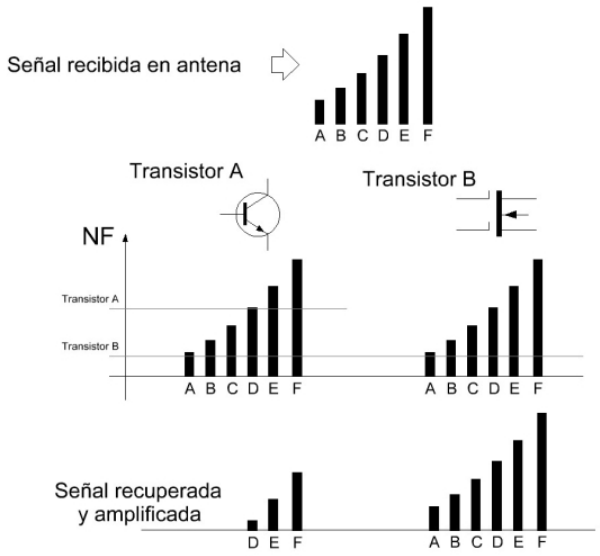


Ilustración 3.6.4: Comparativa de Sensibilidad en Receptores.

- Determinación del Origen de Ruido e Interferencias:

Como se ha visto en el subcapítulo 3.4 existe una gran variedad de fuentes generadoras de ruido e interferencias que son capaces de ensordecer al receptor. Una forma de contabilizar el ruido es por medio de la relación señal/ruido (SNR) siendo la proporción existente entre la potencia de la señal transmitida y la potencia del ruido que la corrompe.

- Margen de Desvanecimiento (Fading):

Una transmisión debe ser capaz de soportar las pérdidas de señal que pueden acarrear cambios climáticos severos. El valor que se toma como norma es de 10 dB de pérdida de señal y aun así seguir manteniendo la transmisión activa. Es muy importante tener prevista esta condición ya que, si tenemos una recepción en el límite de las posibilidades, esto se transforma en un vínculo muy frágil que deja de funcionar con apenas un poco de lluvia.

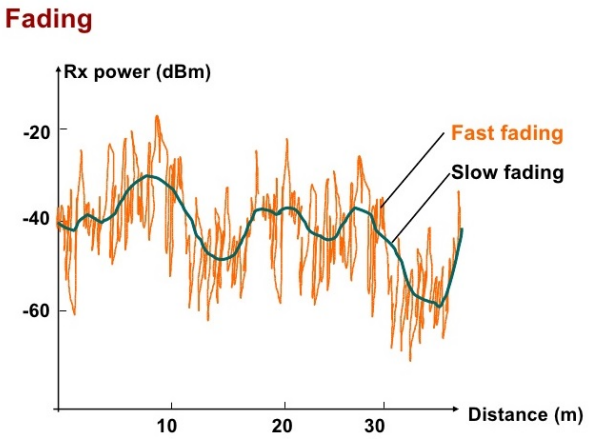


Ilustración 3.6.5: Desvanecimiento (Fading).

- Pérdidas en Medios de Transmisión:

La elección del cable y los conectores apropiados para lograr una transmisión exitosa dependen de muchos factores y todos varían de un escenario a otro. Esta situación algo difícil de equilibrar se basa en la propiedad de atenuación que poseen todos los cables coaxiales utilizados para enlazar los equipos con la antena. Un estudio preliminar de las distancias a cubrir, junto a la potencia del equipo transmisor y la ganancia de las antenas ayuda a seleccionar la mejor opción.

En cuanto al medio de transmisión inalámbrico, debe admitir una tasa de error y corrección de datos en la transmisión o recepción superior a lo que podría ser mediante una conexión por cable. El protocolo de comunicación de los datos a enlazar, debe ser capaz de sortear las deficiencias lógicas que posee el sistema inalámbrico. Las terminales de transmisión y recepción deben sincronizar su velocidad de proceso y adaptarse al límite que imponga la transmisión de acuerdo a su frecuencia de trabajo y ancho de banda.

En sistemas operados a altas frecuencias, es importante mantener la línea de visión para evitar pérdidas importantes. Para modelar las pérdidas que se producen por la obstrucción del enlace radioeléctrico se utiliza el concepto de las llamadas "Zonas de Fresnel".

Estas zonas consisten en unos elipsoides concéntricos que rodean a las ondas radioeléctricas directas de un enlace radioeléctrico y que quedan definidos a partir de las posiciones de las antenas transmisora y receptora.

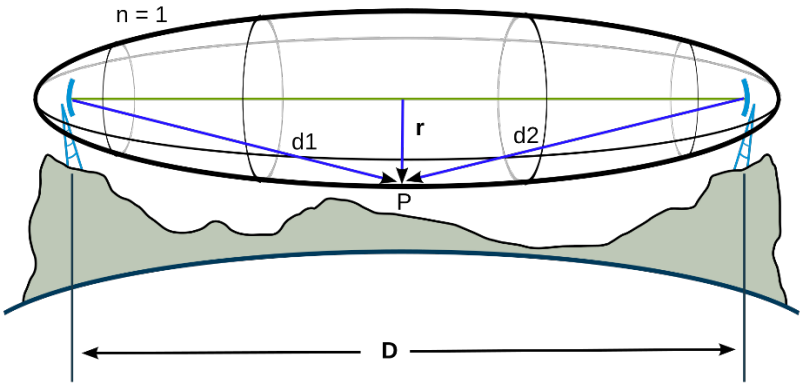


Ilustración 3.6.6: Elipsoide de Fresnel.

Las Zonas de Fresnel tienen la propiedad de que una onda derivada de la antena transmisora se puede reflejar sobre la superficie del elipsoide y después incidir sobre la antena receptora habiendo recorrido una distancia superior a la recorrida por la onda radioeléctrica directa en múltiplos de media longitud de onda.

$$r_n = \sqrt{n\frac{\lambda d_1 d_2}{d_1+d_2}}$$

Ecuación 3.6.4: Cálculo del Radio del Elipsoide de Fresnel.

La obstrucción permitida depende del factor K (curvatura de la tierra), por tanto, durante la fase de planificación del radioenlace, debe asegurarse que la primera zona de Fresnel se encuentre libre de obstáculos considerando un factor K = 4/3 y despejada en un 60% considerando un factor K = 2/3.

Para estimar las pérdidas por obstáculos cercanos a un enlace radioeléctrico suele emplearse la Recomendación UIT-R P530 (Apartado 2.2.):

$$A\ (dB) = -20\frac{h}{r_1} + 10$$

Ecuación 3.6.5: Cálculo de Atenuación por Obstáculos en el Elipsoide de Fresnel.

- Presupuesto de Potencia del Enlace (Link Budget):

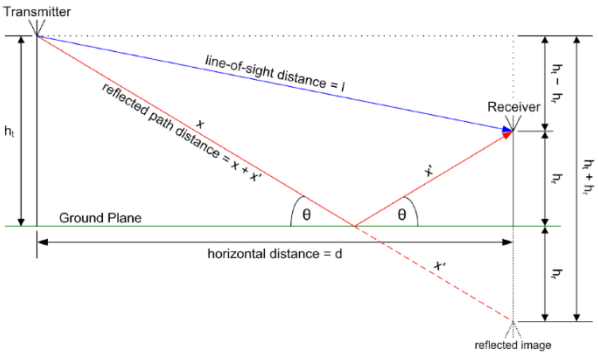
El presupuesto de potencia para un enlace de transmisión de datos es el balance de ganancias y pérdidas desde el transmisor, a través de cables, conectores y espacio libre hacia el receptor. La estimación del valor de potencia en diferentes partes del radioenlace es necesaria para hacer el mejor diseño y elegir el equipamiento adecuado.

$$Link\ Budget\ (dB) = TX\ (dB) - RX(dB) + Gain\ (dB) - Losses\ (dB)$$

Ecuación 3.6.6: Presupuesto de Potencia del Enlace (Link Budget).

Los elementos pueden ser divididos en 3 partes principales:

- Lado de Transmisión: la potencia de transmisión, las pérdidas en conectores y la ganancia de antena.
- Espacio de Transmisión: pérdidas por propagación en el espacio, (como las pérdidas en el espacio libre siguen un modelo ideal, se suele utilizar el Modelo de los 2 Rayos).



$$MPPL\ (dB) = 40\log(d) - 10\log(G_T G_R h_t^2 h_r^2)$$

Ecuación 3.6.7: Modelo de los dos rayos.

Ilustración 3.6.7: Modelo de los dos rayos.

- Lado de Recepción: la ganancia de antena, las pérdidas en conectores y la sensibilidad del dispositivo receptor.

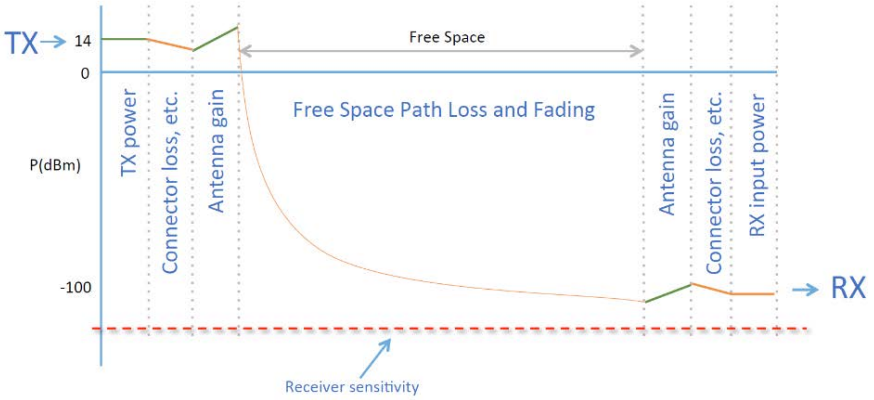


Ilustración 3.6.8: Representación del Presupuesto de Enlace (Link Budget).

3.7 PROTOCOLOS DE COMUNICACIÓN

El primer paso hacia la estandarización internacional de los protocolos para la comunicación de red fue introducido por la International Standards Organization (ISO) en 1978 con el modelo ISO para sistemas abiertos de interconexión (OSI).

Ilustración 3.7.1: International Standards Organization (ISO).

El modelo de referencia OSI es el modelo principal para las comunicaciones por red. Aunque existen otros modelos, en la actualidad la mayoría de los fabricantes de redes relacionan sus productos con el modelo de referencia OSI.

Este modelo es utilizado para comprender el cómo viaja la información a través de una red y visualizar cómo la información o los paquetes de datos viajan desde los programas de aplicación a través de un medio de red, hasta otra aplicación ubicada en otro dispositivo de la red, aun cuando el transmisor y el receptor tengan distintos tipos de medios de red.

En el modelo de referencia OSI, hay siete capas numeradas, cada una de las cuales ilustra una función de red específica. Esta división de las funciones de networking se denomina división en capas. Si la red se divide en estas siete capas, se obtienen las siguientes ventajas:

- Divide la comunicación de red en partes más pequeñas y sencillas.
- Normaliza los componentes de red para permitir el desarrollo y el soporte de los productos de diferentes fabricantes.
- Permite a los distintos tipos de hardware y software de red comunicarse entre sí.
- Impide que los cambios en una capa puedan afectar las demás capas, para que se puedan desarrollar con más rapidez.

El problema de trasladar información entre dispositivos se divide problemas más pequeños y simples representados en el modelo de referencia OSI en capas, siendo estas:

1. Capa Física: esta capa define las especificaciones eléctricas, mecánicas, de procedimiento y funcionales para activar, mantener y desactivar el enlace físico entre sistemas finales.
2. Capa de Enlace: esta capa proporciona el tránsito de los datos confiables a través de un enlace físico. Al hacerlo, la capa de enlace de datos se ocupa del direccionamiento físico, la topología y acceso a red, la notificación de errores, entrega de tramas y control de flujo.
3. Capa de Red: esta capa proporciona conectividad y selección de ruta entre dos sistemas que pueden estar ubicados en redes geográficamente distintas.
4. Capa de Transporte: esta capa efectúa el transporte de los datos originados por el emisor y recibidos por el receptor, independizando el tipo de red física utilizada.
5. Capa de Sesión: esta capa establece, administra y finaliza el enlace entre dos dispositivos que están transmitiendo datos. Dada una sesión establecida entre dos dispositivos, la misma efectúa las operaciones definidas de principio a fin reanudándolas en caso de interrupción.
6. Capa de Presentación: esta capa garantiza la presentación de información de manera que, aunque los dispositivos tengan distintos formatos, los datos obtenidos sean reconocibles. Incluso de ser necesario, esta capa traduce estos formatos a uno en común.
7. Capa de Aplicación: esta capa es la más cercana al usuario del modelo OSI, ofrece a las aplicaciones externas al modelo la posibilidad de acceder a los servicios del resto de capas y define protocolos para intercambio de datos.

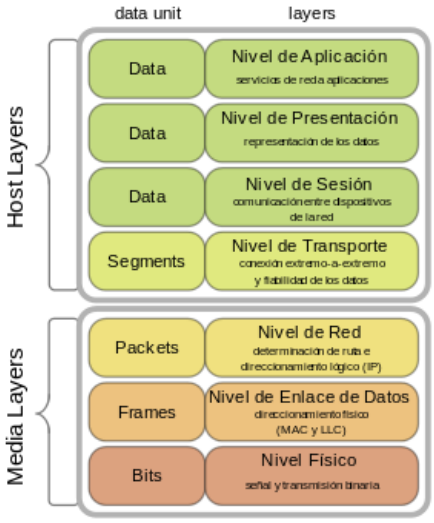


Ilustración 3.7.2: Capas del Modelo de Referencia ISO-OSI

3.8 INTERNET DE LAS COSAS Y REDES DE BAJA POTENCIA

El Internet de las Cosas o Internet of Things (IoT) es un concepto un concepto que nació en el Instituto de Tecnología de Massachusetts (MIT). Se trata una revolución en las relaciones entre los objetos y las personas, que se conectarán entre ellos y la red ofreciendo datos en tiempo real, dicho de otro modo, acerca la digitalización del mundo físico.

Diagrama 3.8.1: Bases de la Implementación del IOT.

> "Si una persona se conecta a la red, le cambia la vida. Pero si todas las cosas y objetos se conectan, es el mundo el que cambia." Hans Vestberg (CEO Ericsson)

Ilustración 3.8.1: Redes de Comunicación del Internet de las Cosas.

Las tecnologías LPWAN están diseñadas para conectar una gran cantidad de dispositivos de forma bidireccional, con un bajo coste, con un consumo de energía reducido y con un uso bajo de ancho de banda. Actualmente no existe un standard preestablecido para las LPWAN, pues las principales tecnologías en cuanto a desarrollo son las siguientes:

sigfox LoRa™ NB - IoT

Ilustración 3.8.2: Principales Tecnologías LPWAN.

4 METODOLOGÍA

4.1 ANÁLISIS COMPARATIVO ENTRE TECNOLOGÍAS LPWAN

Como primer paso, debemos realizar un análisis comparativo entre tecnologías LPWAN existentes con el fin de conseguir una solución óptima acorde a los requisitos de diseño de UAV's. Como primer approach realizaremos una introducción estas tecnologías:

- SIGFOX:

Sigfox es una empresa francesa fundada en 2009, basa su actividad principalmente en desarrollo de redes inalámbricas independientes para su implementación en el internet de las cosas.
Para su uso, es necesario la incorporación de chips compatibles, para ello trabaja con fabricantes como Texas Instruments, Atmel, Silicon Labs, entre otros, para poder ofrecer los distintos tipos de SOC, transceptores y componentes de conexión a su red.

Sigfox fue inspirado en el sistema de comunicaciones existente en los submarinos de la II Guerra Mundial, que eran capaces de transmitir mensajes cortos a todo el mundo por medio de transmisores bidireccionales de baja potencia.

Ilustración 4.1.1: Sigfox.

Esta tecnología se presenta como un complemento o alternativa al uso de las redes actuales. La red actualmente se encuentra disponible en 32 países, entre ellos España y presenta previsiones de una expansión mayor en un corto plazo de tiempo.

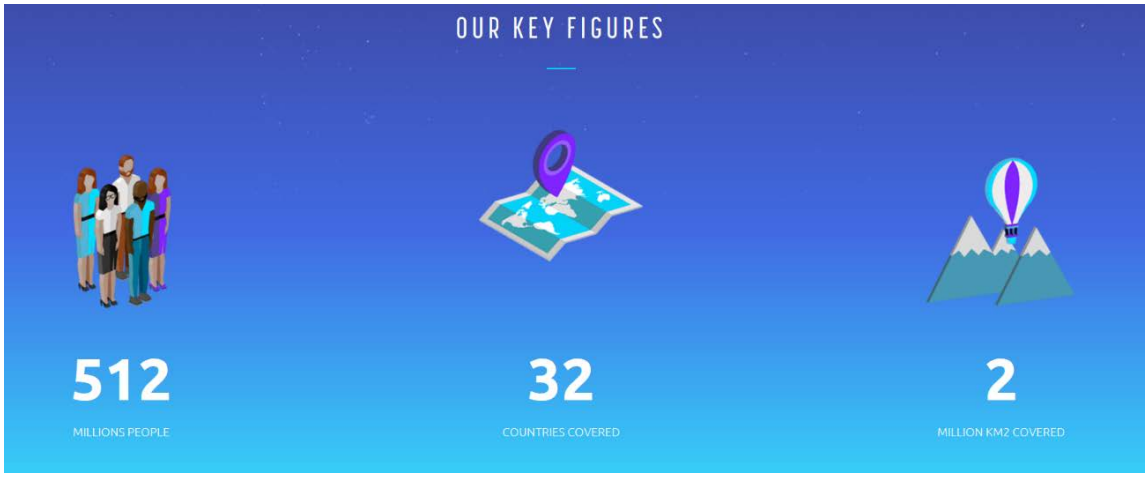


Ilustración 4.1.2: Key Figures de Sigfox.

- LoRa y LoRaWAN:

Una serie de líderes de la industria del internet como ARM, Cisco, IBM, Microchip, Semtech, entre otros, fundaron la LoRa Alliance con el objetivo de desarrollar un nuevo tipo de tecnología de comunicación más eficiente y conseguir un standard de tecnología para el internet de las cosas.

Estas comunicaciones están enfocadas para pequeños dispositivos que desean transmitir poca información y, por tanto, no tener la necesidad de una gran velocidad, pero si con la clara intención de emplear el menor consumo posible para mantener el servicio de estos dispositivos.

Ilustración 4.1.3: LoRa.

Este tipo de tecnología es la evolución de las tecnologías de espectro ensanchado usadas en la II Guerra Mundial donde los esfuerzos de investigación y desarrollo se centraban en contramedidas de radares, interceptación de señales y balizas de navegación.

Actualmente existen redes abiertas a usuarios finales en Francia, Bélgica, Suiza, Países Bajos y Sudáfrica, desplegadas por grandes operadores de telecomunicaciones como Orange. En España, no exista actualmente una red operativa, aunque parece que Orange, al igual que en Francia, tiene intención de desplegar una en España.

- NB-IoT:

NarrowBand IoT es una tecnología standard evolución de la tecnología LTE/4G para su adaptación al internet de las cosas. Es desarrollado por Vodafone, Huawei y U-Blox, presentando la gran apuesta de las operadoras por la implementación de versiones reducidas de tecnología móvil 4G.

El Grupo Aguas de Valencia y Vodafone España han presentado recientemente una prueba piloto que demuestra la viabilidad de esta tecnología con resultados muy positivos.

Ilustración 4.1.4: NB-IoT.

Vodafone anunció en octubre de 2016 que implementará las primeras redes NB-IoT comerciales en Alemania, Irlanda, Países Bajos y España en los primeros meses de 2017.

Vodafone contará precisamente en España con más de 1.000 estaciones de red actualizadas con esta tecnología, permitiendo cada una de ellas, conectar más de 100.000 dispositivos a internet.

Tras esta introducción, exponemos las principales características de estas tecnologías LPWAN:

Características	sigfox	LoRa	NB - IoT
Tecnología	Ultra Narrowband	Spread Spectrum	Ultra Narrowband
Modulación	BPSK / GFSK	CSS / FSK	OFDMA
Topología	Estrella	Estrella	Estrella
Bandas de Frecuencia (EU)	868 MHz	433/868 MHz	800 MHz
Tipo de Banda de Frecuencia	Libre (ISM)	Libre (ISM)	Licenciada (4G)
Ancho de Banda	100 Hz	125-500 kHz	200 kHz
Rango de Alcance Urbano	3-10 km	2-5 km	Red Móvil
Rango de Alcance Suburbano	30-50 km	15 km	Red Móvil
Velocidad de Transmisión	100 bps	290 bps - 50 kbps	20 kbps
Tamaño del Mensaje	12 bytes	242 bytes	-
Limitación de Mensajes	140 mensajes/día	Ilimitado	Ilimitado
Eficiencia Energética	Alta	Alta	Alta
Inmunidad Interferencias	Baja	Alta	Baja
Desarrolladores	Sigfox	Lora Alliance	3GPP
Disponibilidad en España	Si	Si	2017

* NB-IoT es el nombre dado a LTE Cat. 13 NB1, desarrollado por Vodafone, Huawei y U-Blox.

Tabla 4.1.1: Análisis Comparativo de Características de las Tecnologías LPWAN.

Una vez establecidas las características de las tecnologías a valorar, debemos establecer el criterio de valoración, para ello expondremos las características demandadas para este proyecto.

Ilustración 4.1.5: UAV's del Grupo Integrado de Ingeniería.

- Alcance:

La OACI define siete tipos de espacio aéreo, nombrando a cada uno de ellos con una letra desde la A hasta la G. De la A a la E nos encontramos ante espacios aéreos controlados. Los requisitos y grado de control requeridos para volar en estos espacios son mayores en la zona A y se van reduciendo hacia la E. Los espacios F y G son espacios aéreos no controlados y son este tipo de espacios aéreos en los que los UAV's pueden volar.

Para los vuelos se seguirán las Reglas de Vuelo Visual (VFR), vuelo realizado de día y condiciones visuales favorables, siempre a la vista y con la limitación de no superar los 120 metros de altura, además de realizar el vuelo en zonas adecuadas, en nuestro caso, en el Club de Aeromodelismo A Pombiña, situado en Narón.

- Eficiencia Energética:

La eficiencia energética será otro factor imprescindible, debemos asegurar cargar al dron con el menor el menor consumo posible con el fin de afectar lo mínimo la autonomía de vuelo.

- Transmisión de Información:

Debemos asegurar el máximo ancho de banda y las menores interferencias con el fin de mantener un enlace estable en el cuál se pueda transmitir la mayor cantidad de datos evitando la pérdida de paquetes.

Tras esto, exponemos las valoraciones en cuanto a afinidad, siendo elegida la tecnología LoRa:

Requisitos de Diseño	sigfox	LoRa	NB - IoT
Alcance	✓	✓	✓
Eficiencia Energética	✓	✓	-
Ancho de Banda	X	✓	✓
Inmunidad a Interferencias	X	✓	X
Disponibilidad	✓	✓	X

Tabla 4.1.2: Análisis Comparativo de Afinidad a Requisitos de Diseño.

4.2 LORA Y LORAWAN

LoRa es una técnica de modulación basada en tecnología de espectro ensanchado y es a su vez una variación de la modulación Chirp Spread Spectrum (CSS). Este tipo de modulación mejora de manera notable la sensibilidad del receptor, utilizando todo el ancho de banda para la transmisión de la señal y dándole robustez frente a ruido o interferencias. Se define como la Capa Física (véase subcapítulo 3.7) dentro del sistema de comunicaciones. Fue desarrollada por Cycleo SAS en 2010 y posteriormente adquirida por Semtech.

Ilustración 4.2.1: Semtech.

La modulación de espectro ensanchado LoRa se realiza representando cada bit del área de datos como múltiples pedazos" de información. La velocidad con la que la información ensanchada es enviada se denomina velocidad de símbolo (Symbol Rate o Rs) y la relación entre velocidad de símbolo y la velocidad de pedazos de información se denomina factor de ensanchamiento (SF), representando el número de símbolos que son enviados por cada bit de información.

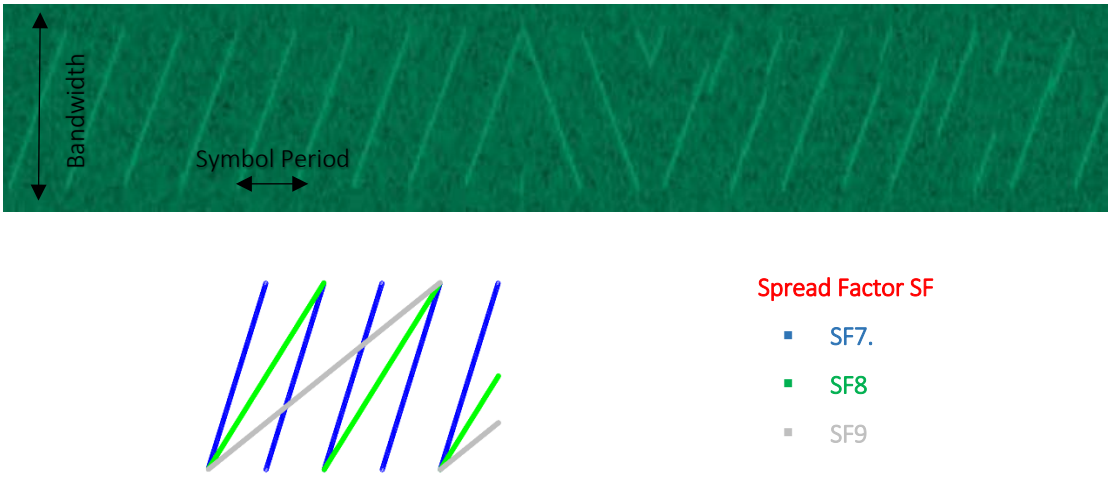


Ilustración 4.2.2: Modulación LoRa y Factores de Ensanchamiento.

Para aumentar la robustez del enlace, LoRa emplea una codificación por redundancia cíclica (CRC) para realizar detección y corrección de errores hacia adelante (FEC), siendo particularmente eficiente mejorando la fiabilidad del enlace en la presencia de interferencias.

$$T_S = {}^{2^{SF}}/_{BW} \qquad R_S(\text{sym/s}) = {}^{1}/_{T_S}$$

Ecuaciones 4.2.1: Cálculo del Período y el Ratio de Símbolos.

$$R_b(\text{bit/s}) = SF\ x\ R_S\ x\ \frac{4}{4+CR}$$

Ecuación 4.2.2: Cálculo de la Velocidad de Modulación.

LoRaWAN™ es una especificación de red de área amplia de baja potencia (LPWAN) propuesta por LoRa Alliance, ofreciendo la Capa de Enlace o MAC (véase subcapítulo 3.7). LoRaWAN™ se centra en los requisitos clave de Internet de las cosas, como la comunicación bidireccional segura, la movilidad y servicios de localización, siguiendo la LoRaWAN™ Specification.

Ilustración 4.2.3: LoRa Alliance.

La arquitectura de red se presenta típicamente en una topología de estrella y su funcionamiento se basa en establecer una red de dispositivos conectados llamados nodos, que establecerán un enlace inalámbrico con modulación LoRa con un elemento más potente capaz de comunicarse con todos estos elementos de manera gestionada llamado Gateway. A su vez, este gateway será capaz de comunicarse por otro método de red, con mayor ancho de banda y transmitir toda la información de estos dispositivos a aquellos elementos que la soliciten.

La LoRaWAN™ Specification distingue en una red LoRa las siguientes clases de dispositivos:

- Dispositivos Finales – End Nodes (Clase A):

Los dispositivos de clase A permiten las comunicaciones bidireccionales con la limitación de solo recibir datos (canal downlink) siempre y cuando se haya enviado antes un paquete (canal uplink). El paquete de vuelta contiene la confirmación o acknowledgement (ACK) del paquete enviado, así como los datos de la aplicación si esto fuese necesario. Esta clase presenta menor consumo energético y es implementada como la clase básica compatible con LoRaWAN™.

- Dispositivos Finales con Ranuras de Recepción Programadas – Gateway (Clase B):

Los dispositivos de clase B eliminan la limitación de recepción de datos con la necesidad de enviar previamente un paquete, de esta forma, se permite el envío de datos a los dispositivos de forma programada. Los gateway deben enviar periódicamente beacons, que permiten mantener los dispositivos sincronizados. Este tipo de dispositivos presentan un mayor consumo de energía que los dispositivos de clase A.

- Dispositivos Finales con Ranuras de Recepción Máximas- Base Station (Clase C):

Los dispositivos de clase C están casi continuamente escuchando (modo de recepción). Esta clase proporciona la menor latencia y capacidad de envío entre el servidor a los dispositivos. Por la contra, presenta un consumo energético mucho mayor que las clases A y B.

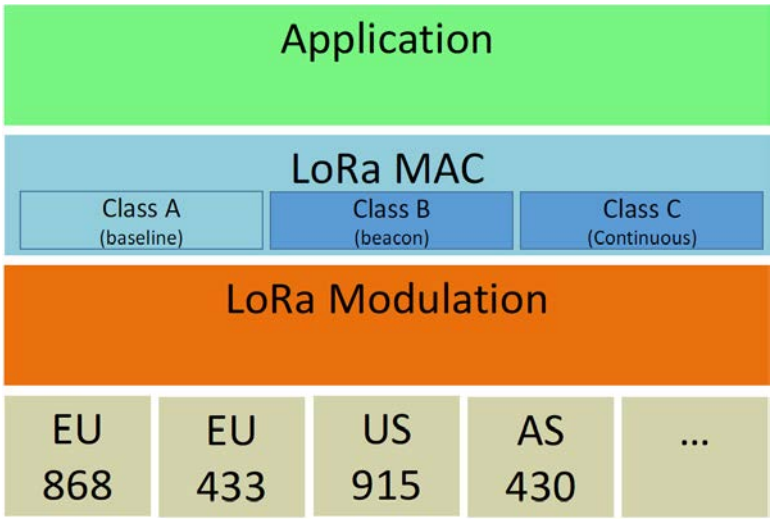


Ilustración 4.2.4: Clases de Dispositivos LoRa.

Los módulos LoRaWAN™ permiten el envío de datos directamente a cualquier Gateway que sea compatible con LoRaWAN™. Esta es la principal conexión dentro de las diferentes configuraciones de red. Las principales empresas que nos ofrecen este tipo de soluciones actualmente son:

- Cisco.
- Kerlink.
- Link-Labs.
- Multitech

Ilustración 4.2.5: Principales Empresas Proveedoras de Gateway LoRaWAN™.

Para visualizar la información también se necesitará una plataforma en la nube o un servidor local, donde los datos sean enviados. Normalmente, al adquirir el gateway, este ya cuenta con una la licencia para plataformas en la nube o software compatible.

Ilustración 4.2.6: Configuración de Red por Conexión Directa a Gateway LoRaWAN™.

Debido a que estos gateway y/o licencias presentan actualmente un precio prohibitivo, existen modos de conexión alternativos dentro de las configuraciones de red, podemos destacar el modo P2P y el modo Híbrido.

En Modo P2P, los dispositivos finales pueden conectarse directamente entre ellos sin costes, ya que no utilizan infraestructura de red. Este modo funciona sin la necesidad de una estación base, ya que se usará un dispositivo final como gateway, además tampoco se necesitará una cuenta en la nube, no requiriendo la compra de una licencia.

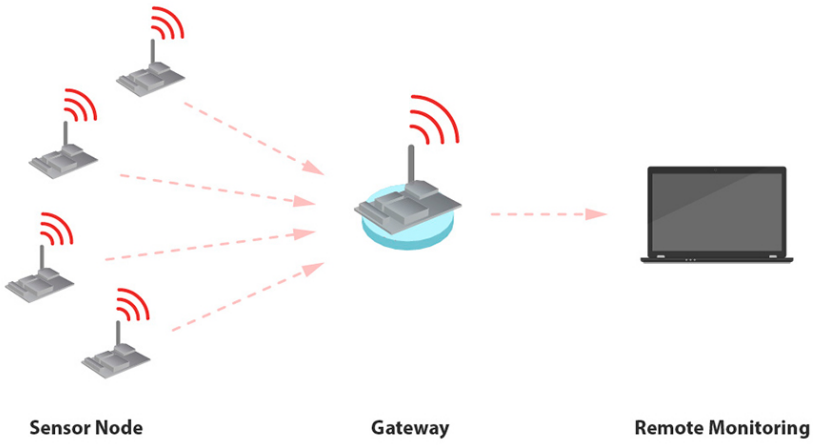


Ilustración 4.2.7: Configuración de Red por Conexión en Modo P2P.

En Modo Híbrido, se utiliza una combinación de los modos LoRaWAN™ y P2P que permite enviar mensajes utilizando redes LoRaWAN™. Los nodos utilizan una topología de estrella en modo P2P para llegar al dispositivo final que actúa como gateway y hacer que se acceda a la red LoRaWAN™. Este modo requiere licencia o acceso a plataformas libres en la nube como The Things Network™.

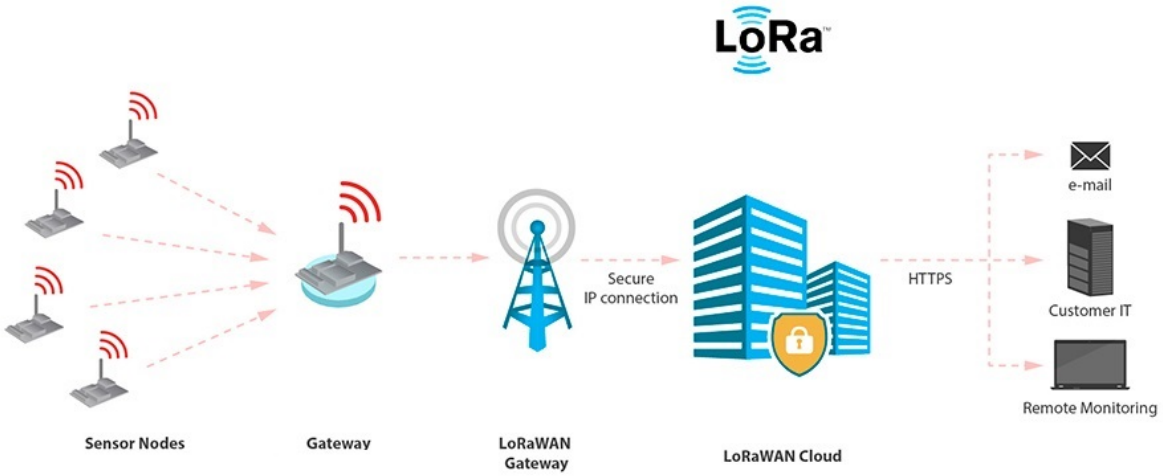


Ilustración 4.2.8: Configuración de Red por Conexión en Modo Híbrido.

Las características de la transmisión vienen especificadas para las diferentes bandas de frecuencia en la LoRaWAN™ Specification:

7.1.2. Frecuencias de Canales en la Banda ISM EU863-870:

En Europa, la asignación del espectro radioeléctrico es la banda ISM definida por ETSI [EN300.220].
Los operadores o desarrolladores pueden atribuir libremente los canales de red. Sin embargo, los tres primeros canales especificados, por defecto, deben ser implementados en cada dispositivo final EU868. Estos canales deben ser los mínimos que todos los gateway deben estar siempre escuchando.

MODULACIÓN	LoRa
ANCHO DE BANDA (kHz)	125
FRECUENCIA DE CANALES (MHz)	868,10 / 868,30 / 868,50
VELOCIDAD DE TRANSMISIÓN (FSK / LoRa)	DR0 – DR5 / 0,30 -5 kbps
NÚMERO DE CANALES	3
CICLO DE TRABAJO	< 1%

Tabla 4.2.1: Características de Canales EU863-870.

Para acceder al medio físico, la normativa ETSI imponen algunas restricciones, como el tiempo máximo que el transmisor puede estar encendido o el tiempo que puede estar transmitiendo por hora. A su vez, estas permiten la elección del uso de limitaciones por Ciclo de Trabajo o gestión de transmisiones por agilidad de frecuencia para Escuchar antes de Hablar (LBT AFA).

LoRaWAN impone una limitación del ciclo de trabajo por sub-banda. Tras la transmisión, la sub-banda no podrá ser utilizada durante T_{OFF} segundos:

$$T_{off} = \frac{TimeOnAir}{Duty\ Cycle} - TimeOnAir$$

Ecuación 4.2.3: Cálculo del Toff en función del Ciclo de Trabajo y el Tiempo de Transmisión.

Durante el tiempo de indisponibilidad de la sub-banda, el dispositivo podrá transmitir por otras sub-bandas. Si estas estuvieran ocupadas el dispositivo se pondría en modo de espera.

Los dispositivos que utilicen la banda ISM de EU868MHz deben utilizar los siguientes parámetros por defecto:

- Potencia de transmisión radiada: 14 dBm. → Ecuación 3.6.1 → (25 mW).

Cumple con la Normativa Unión Europea para Dispositivos de Corto Alcance (2006/771/EC).

Los dispositivos deben poder ser utilizados a una frecuencia de 863/870 MHz y debe de disponer de una estructura de datos de canal para al menos almacenar los parámetros de 16 canales.

7.4.2. Frecuencias de Canales en la Banda ISM EU433:

En Europa, LoRaWAN puede utilizarse en la banda ETSI 433-434 MHz siempre y cuando la potencia de transmisión radiada del dispositivo de radio sea inferior a los 10 dBm → Ecuación 3.6.1 → (10 mW).

Cumple con la Normativa Unión Europea para Dispositivos de Corto Alcance (2006/771/EC).

El ciclo de trabajo de los dispositivos debe ser también inferior al 1%. La frecuencia central de los canales LoRaWAN puede estar en el siguiente rango:

- Frecuencia Mínima: 433,175 MHz.
- Frecuencia Máxima: 434,665 MHz.

Los dispositivos deben poder ser utilizados a una frecuencia de 433,05/434,79 MHz y debe de disponer de una estructura de datos de canal para al menos almacenar los parámetros de 16 canales.

MODULACIÓN	LoRa
ANCHO DE BANDA (kHz)	125
FRECUENCIA DE CANALES (MHz)	433,175/433,375/433,575
VELOCIDAD DE TRANSMISIÓN (FSK / LoRa)	DR0 – DR5 / 0,30 -5 kbps
NÚMERO DE CANALES	3
CICLO DE TRABAJO	< 1%

Tabla 4.2.2: Características del Canal EU433.

* Una estructura de datos de canal corresponde a una frecuencia y un conjunto de velocidades de datos utilizable en esta frecuencia.

La comunicación entre los dispositivos finales y los gateway se extiende en diferentes canales de frecuencia y velocidad. Debido a la tecnología de espectro ensanchado, las comunicaciones con diferentes velocidades de datos no interfieren entre sí y crean un conjunto de canales "virtuales" que aumentan la capacidad de los gateway. Para conseguir maximizar la duración de las baterías de dispositivos finales como la capacidad total de la red, el servidor de red LoRaWAN™ gestiona la velocidad de datos y salida de RF de cada dispositivo final individualmente mediante un esquema de Velocidad de datos Adaptativa (ADR).

Para participar en una red LoRaWAN™, cada dispositivo final debe ser personalizado y activado. Como indica la LoRaWAN™ Specification, la siguiente información ha de ser almacenada en los dispositivos finales:

- Dirección de Dispositivo (DevAddr):

El DevAddr consiste en 32 bits que identifican al dispositivo final dentro de la red actual.

Bit	[31 … 25]	[24 … 0]
DevAddr Bits	NwkID	NwkAddr

Tabla 4.2.3: Estructura de Dirección de Dispositivo.

Los 7 bits más significativos se utilizan como identificador de red (NwkID) para separar direcciones en redes territorialmente superpuestas de operadores de redes y para remediar problemas de roaming. Los menos significativos 25 bits, la dirección de red (NwkAddr) del dispositivo final, pueden ser asignado arbitrariamente por el administrador de la red.

- Identificador de Aplicación Global (AppEUI):

El AppEUI es un identificador de aplicación global en las direcciones IEEE EUI64 que identificará de forma única al proveedor de la aplicación del dispositivo final (propietario). Esta se almacenará en el dispositivo antes de ejecutar el procedimiento de activación.

- Clave de Sesión de Red (NwkSKey):

El NwkSKey es una clave de sesión de red específica para el dispositivo final. Es utilizado tanto por el servidor de red y el dispositivo final para calcular y verificar el código de integridad del mensaje (MIC) para garantizar la integridad de los datos. También se utiliza para cifrar y descifrar el campo de carga útil en mensajes de carácter de protocolo de red (MAC).

- Clave de Sesión de Aplicación (AppSKey):

La AppSKey es una clave de sesión de aplicación específica para el dispositivo final. Es utilizado tanto por servidor de red y el dispositivo final para cifrar y descifrar el campo de mensajes de datos específicos de la aplicación. También se utiliza para calcular y verificar una un nivel de aplicación que pueda incluirse en la carga útil de mensajes de datos específicos de la aplicación.

Según la LoRaWAN™ Specification, existen dos métodos para la conexión de dispositivos a una red LoRaWAN™:

- Activación sobre el Aire (OTTA):

Para la Activación sobre el Aire, los dispositivos finales deben seguir un procedimiento previo de conexión para poder participar en intercambios de datos con el servidor. Además de esto, cada dispositivo que pierda la información de la sesión debe volver a conectarse nuevamente.

El procedimiento de conexión requiere que el dispositivo final haya sido personalizado con la siguiente información:

- Identificador de Dispositivo Final Global (DevEUI): identificación de dispositivo final global en las direcciones IEEE EUI64 que identifica al dispositivo de forma única.
- Identificador de Aplicación Global (AppEUI).
- Llave de Aplicación (AppKey): clave de aplicación con cifrado AES-128 para dispositivo final asignada por el propietario de la aplicación y llave de administración específica bajo control del proveedor de la aplicación. Se utiliza para derivar las claves de sesión NwkSKey y AppSKey específicas para el cifrado y verificación del dispositivo final.

- Activación por Personalización (ABP):

Bajo ciertas circunstancias, los dispositivos finales se pueden activar mediante la personalización. Para la Activación por Personalización, se vinculará directamente un dispositivo final a una red específica omitiendo la petición de conexión y el procedimiento de aceptación.

La activación por personalización implica que la DevAddr, NwkSKey y AppSKey se almacenarán directamente en el dispositivo final en lugar delos DevEUI, AppEUI y AppKey como se hacía con el procedimiento de Activación sobre el Aire (OTTA).

El dispositivo LoRa estará equipado con la información necesaria para su participación en una red específica cuando se inicia. Cada dispositivo tendrá sus únicos respectivos NwSKey y AppSKey, no comprometiendo la seguridad de comunicación con el resto de dispositivos.

El proceso de construcción de las claves debe ser hecho de tal forma que la información de las claves no se puede derivar de ninguna manera.

4.3 DISEÑO DE LA RED

Para este proyecto se llevará a cabo un diseño de red en Modo P2P por motivos de seguridad y privacidad. De igual manera, con vistas a una posible migración, el código ha sido desarrollado para permitir la activación del Modo Híbrido modificando el servidor y la codificación de mensaje, combinando el modo LoRaWAN™, P2P y el soporte de la plataforma libre The Things Network™.

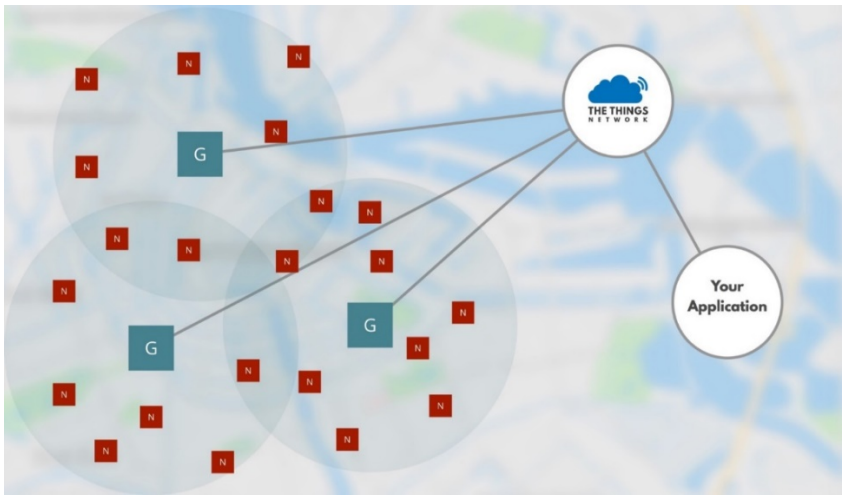


Ilustración 4.3.1: Configuración de Red por Conexión en Modo Híbrido – The Things Network™.

The Things Network™ utiliza tecnología LoRaWAN™ para la conexión de una red descentralizada de código abierto para intercambio de datos con aplicaciones y otras plataformas.

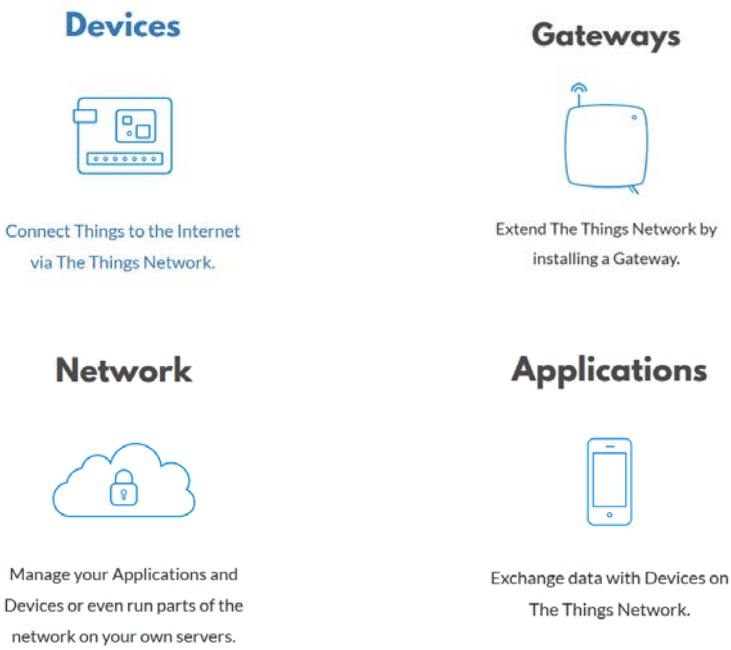


Ilustración 4.3.2: Componentes de Red LoRaWAN™ con soporte de The Things Network™.

Semtech presenta soluciones de conectividad integral que cubren el espectro de frecuencia de radio de la banda industrial, científica y médica (ISM) hasta los 2,40 GHz. La gama de productos para construcción de redes de comunicación LoRa, está compuesta de los siguientes productos:

Producto	Descripción
SX1272	Transceptor de RF 860/1.000 MHz con Tecnología LoRa®.
SX1273	Transceptor de RF 860/1.000 MHz con Tecnología LoRa®.
SX1276	Transceptor de Dual-RF 137/960 MHz con Tecnología LoRa®.
SX1277	Transceptor de RF 137/1.020 MHz con Tecnología LoRa®.
SX1278	Transceptor de RF 137/525 MHz con Tecnología LoRa®.
SX1301	Procesador de Banda Base para Concentrador con Tecnología LoRa®.

Ilustración 4.3.1: Gama de Productos Semtech.

Se han elegido los siguientes componentes para la red LoRaWAN™ a desarrollar:

- Dispositivos Finales o Nudos (End Nodes):

El RN2483 de Microchip es un módulo 433/868 MHz completamente certificado basado en la tecnología inalámbrica LoRa®, utilizando modulación de espectro ensanchado única dentro de la banda Sub-GHz para permitir capacidad de red de largo alcance y baja potencia.

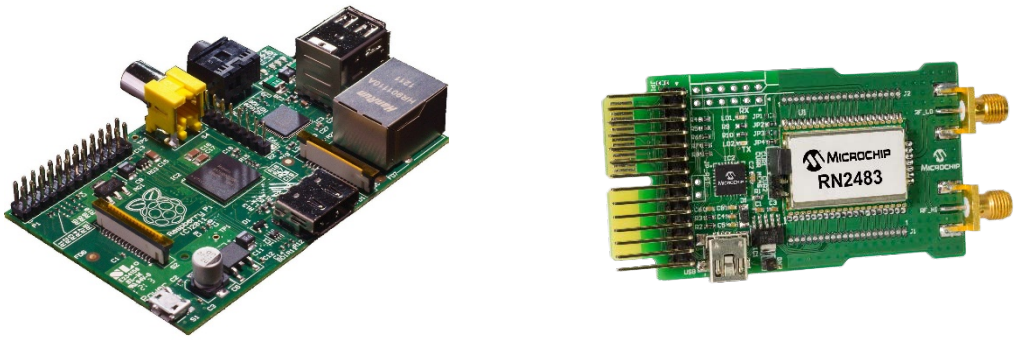


Ilustración 4.3.3: Desarrollo de End-Nodes con Raspberry Pi Modelo B y Microchip RN2483.

Este módulo fue el primero en pasar las pruebas de certificación de LoRa Alliance e incorporar el protocolo LoRaWAN™ Clase A, permitiendo conectividad con infraestructuras de red compatibles con LoRaWAN™, ya sean de uso público o privado. El módulo ha sido escogido debido a que está diseñado para desarrolladores, facilitando su uso y acortando el tiempo de desarrollo, se usará en conjunto con una Raspberry Pi Modelo B desde donde se realizará la lógica del End-Node.

(Véanse Características en Anexo de Datasheets)

- Gateway:

Como se ha comentado en el subcapítulo 4.2 la compra de un gateway comercial actualmente, dadas las dimensiones del proyecto, se hace PROHIBITIVO. Por ello se ha decidido desarrollar un prototipo de gateway por medio de una Raspberry Pi Compute Module y un SX1276 de Semtech.

Ilustración 4.3.4: Desarrollo del Gateway con Raspberry Pi Compute Module y SX1276MB1xAS.

El diseño original del gateway ha sido hecho por Thomas Telkamp, desarrollador de arquitecturas en red para The Things Network™. Este desarrollo consiste en un gateway de canal único utilizado exclusivamente para desarrollo, ya que este, no mantendría la filosofía de gateway planteado por la LoRaWAN™ Specification no siendo multicanal.

Para este proyecto, se ha desarrollado un código más adecuado a los requisitos de diseño y escrito en el lenguaje de programación Python, basado en la solución anteriormente presentada.

Ilustración 4.3.5: Logo de Python

Desarrollador Original	Thomas Telkamp
GitHub	https://github.com/tftelkamp
Dominio Web	https://www.thethingsnetwork.org

Ilustración 4.3.2: Características del Desarrollo Original del Gateway.

4.4 PUESTA EN MARCHA DE DISPOSITIVOS

Este apartado se dividirá en dos partes. En primer lugar, dispondremos la configuración y montaje del Gateway y por último de la configuración de los End-Nodes.

Como primer paso en la configuración de nuestro Gateway debemos realizar la instalación del sistema operativo, en nuestro caso será Raspbian Jessie Lite.

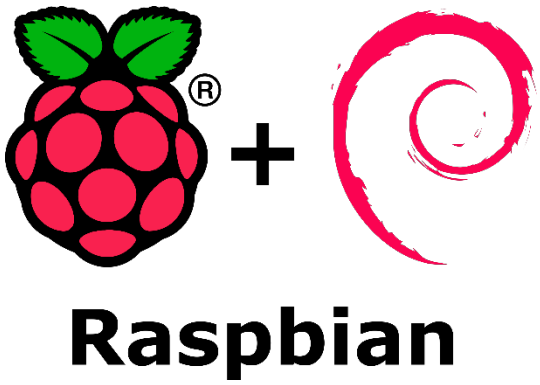


Ilustración 4.4.1: Raspbian Jessie Lite.

La Raspberry Compute Module dispone de una interfaz eMMC conectada a la tarjeta SD, que se encuentra integrada en el módulo, por tanto, para instalar el sistema operativo, es necesario acceder al mismo por medio de las conexiones que nos ofrece la Compute Module I/O Board.

Debemos asegurarnos del correcto acoplamiento de la Raspberry Compute Module a la Compute Module I/O Board por medio de los clips de acoplamiento disponibles y que la conexión del puerto J4 (Usb Slave Boot Enable) esté en posición "EN".

Debemos disponer de otro sistema operativo desde donde realizar la instalación en remoto (se recomiendan principalmente sistemas Linux como Ubuntu o un sistema Windows 7) e instalar RPIBoot (en este caso Kubuntu) siguiendo los siguientes pasos:

1. Utilizamos Git para obtener el código fuente RPIBoot, por tanto, aseguramos de que Git esté instalado.

```
sudo apt-get install git
```

2. Al utilizar Git, se puede producir un error por fecha incorrecta, por tanto, ejecutaremos lo siguiente:

```
sudo date MMDDhhmm
```

Siendo MM es el mes, DD el día, hh las horas y mm los minutos respectivamente

3. Debemos asegurarnos de que libusb esté instalado:

```
sudo apt-get install libusb-1.0-0-dev
```

4. Clonamos por medio de Git el repositorio de herramientas usbboot:

```
git clone --depth = 1 https://github.com/raspberrypi/usbboot
```

5. Nos situamos en la carpeta del repositorio descargado y lo compilamos:

```
cd usbboot
make
```

6. Ejecutamos la herramienta usbboot y esperamos la conexión.

```
sudo ./rpiboot
```

7. Ahora conectamos el ordenador a la placa por medio de un cable micro USB a través del puerto J15 y encendemos la Compute Module I/O Board. La herramienta RPIBoot accederá al Compute Module y permitirá el acceso a la eMMC.

8. En este paso se escribirá la imagen previamente descargada de Raspbian en el dispositivo.

```
sudo dd if=raw 2017-04-10-raspbian-jessie-lite.img of=/dev/sdc bs=4MiB
```

9. Cuando termine, debemos asegurarnos de que el puerto J4 esté desactivado y no hay nada enchufado al puerto J15. Al alimentar la Compute Module I/O Board debería iniciar el sistema desde la eMMC.

Una vez realizada la instalación del sistema operativo, en la primera puesta en marcha, debemos realizar la conexión de periféricos. En nuestro caso conectaremos al puerto hdmi un monitor y en el puerto usb un hub-usb externo para conectar un adaptador wifi y un teclado.

Lo primero que haremos será configurar el módulo wifi y activar el acceso por Servidor Shell (SSH) para tener un funcionamiento normal en remoto. Para ello debemos realizar lo siguiente:

1. Habilitar el módulo wifi modificando la configuración de los archivos interfaces y wpa_supplicant.

```
sudo nano /etc/network/interfaces

sudo nano/etc/wpa_supplicant/wpa_supplicant.conf
```

1.1. Editamos el archivo interfaces para realizar la conexión por IP Dinámica (DHCP).

```
source-directory /etc/network/interfaces.d
auto lo

iface lo inet loopback
iface eth0 inet dhcp

allow-hotplug wlan0
iface wlan0 inet dhcp
        wpa-conf /etc/wpa_supplicant/wpa_supplicant.conf

allow-hotplug wlan1
iface wlan1 inet dhcp
        wpa-conf /etc/wpa_supplicant/wpa_supplicant.conf
```

1.2. Editamos el archivo wpa_supplicant para configurar las redes a las que se tiene acceso.

```
ctrl_interface=DIR=/var/run/wpa_supplicant GROUP=netdev
update_config = 1
network={
        ssid = "Nombre de Red"
        psk = "Contraseña de Red"
}
```

2. Habilitar la opción de Acceso por Servidor Shell (SSH).

```
sudo raspi-config
```

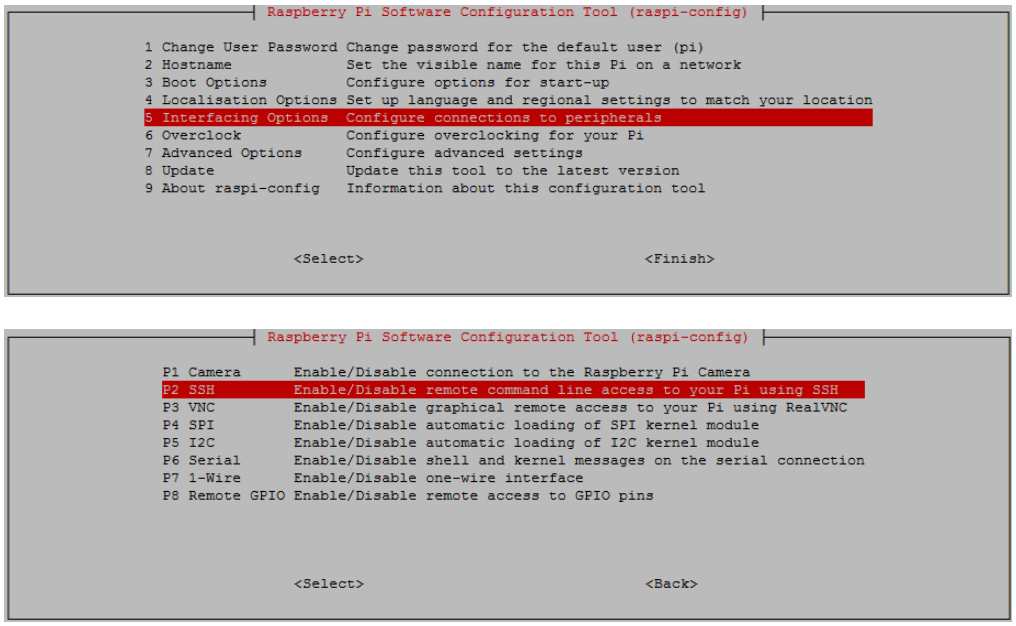


Ilustración 4.4.2: Activación de Servidor SSH y SPI.

* Activaremos a su vez el bus de comunicaciones SPI necesario para la aplicación desarrollada del Gateway.

Tras este proceso, podremos comunicarnos en remoto de forma inalámbrica, eliminando el uso de periféricos salvo el módulo wifi.

Los programas para la gestión en remoto serán PuTTY (servidor SSH) para realizar un control desde terminal y WinSCP (cliente SFTP) para el intercambio de archivos.

Ilustración 4.4.3: Programas PuTTy y WinSCP.

Una vez finalizada la instalación de herramientas para funcionamiento en remoto, nos quedará la instalación de la IDE Python, la librería WiringPi para el control de la interfaz SPI y los GPIO's, la librería PyCrypto para la encriptación AES-128, la librería NTPLib para sincronización con un servidor NTP en las pruebas de latencia y el propio cableado físico.

1. Instalación de la Interfaz de Desarrollo Python 3.

```
sudo apt-get install python3
```

2. Instalación de librería **WiringPi.**

```
sudo apt-get install wiringpi
```

3. Instalación de librería PyCrypto.

```
git clone https://github.com/dlitz/pycrypto

cd pycrypto

python3 setup.py install
```

4. Instalación de librería NTPLib.

```
git clone https://github.com/Tipoca/ntplib

cd ntplib

python3 setup.py install
```

5. Cableado entre Raspberry Pi y Semtech SX1276.

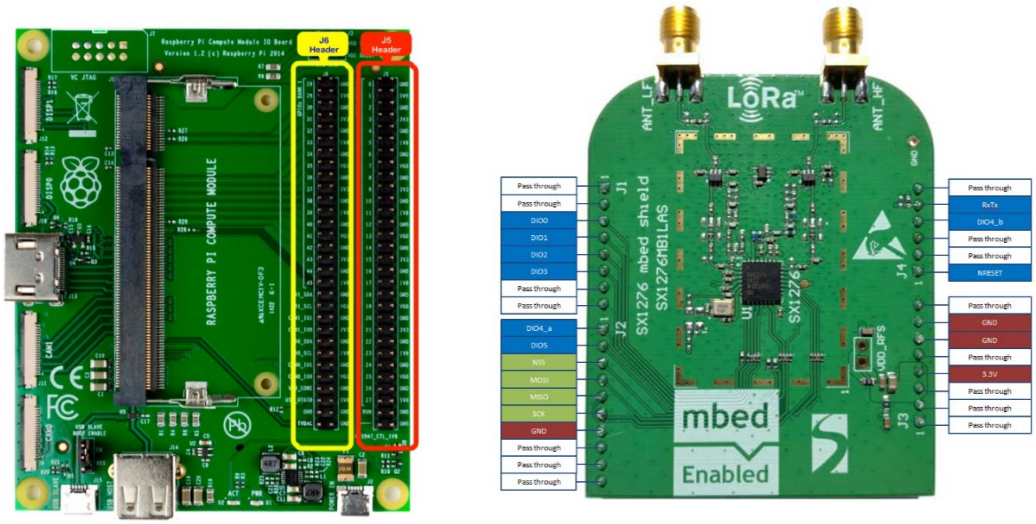


Ilustración 4.4.4: Compute Module I/O Board – Semtech SX1276.

Pines Raspberry Pi	Pines Semtech SX1276
GPIO 0 *	NReset
GPIO 6 *	NSS
GPIO 7 *	DIO0
SPI-MISO (9)	MISO
SPI-MOSI (10)	MOSI
SPI-SCLK (11)	SCK
GND	GND (J3)
3.3V	3.3V (J3)

* Determinados por el software desarrollado. Son configurables modificando líneas de código en la aplicación.

Tabla 4.4.1: Cableado de Pines Raspberry Pi – Semtech SX1276.

Una vez realizados todos estos pasos, procederemos a subir el código realizado y ejecutarlo. Para ello realizaremos los siguientes pasos:

1. Subimos el código a la Raspberry Pi por medio de WinSCP.

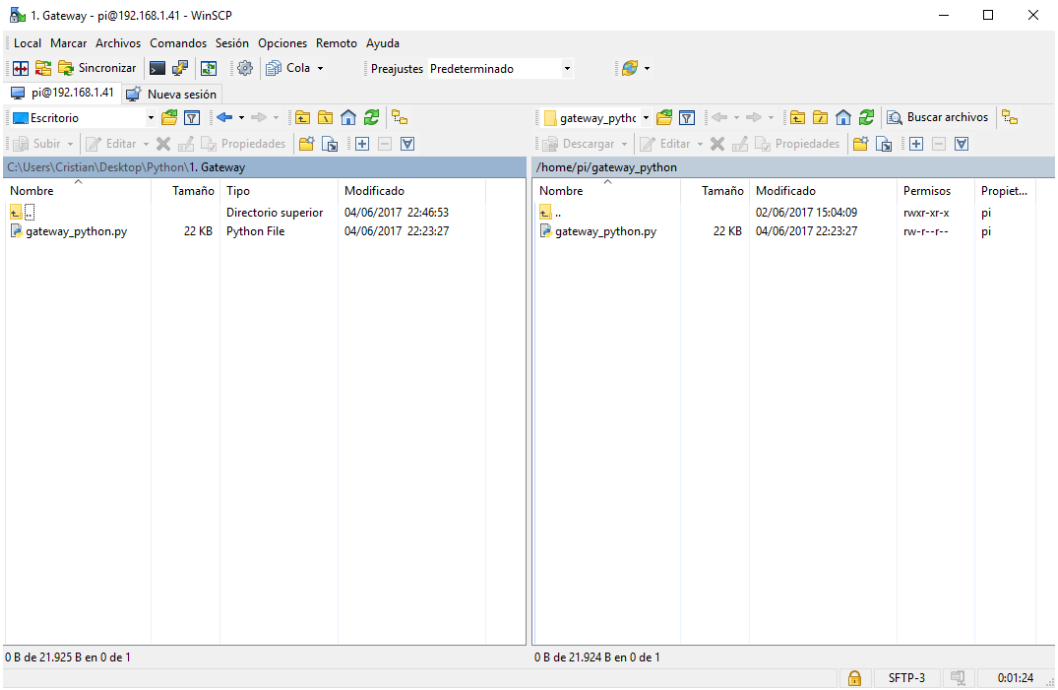


Ilustración 4.4.5: Subida de Código desde WinSCP.

2. Iniciamos Sesión en PuTTY y entramos en la carpeta:

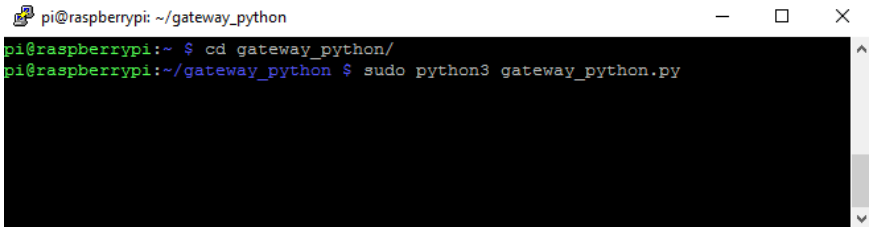


Ilustración 4.4.6: Ejecución de Código desde PuTTy.

3. Ejecutamos el código para poner en funcionamiento el Gateway:

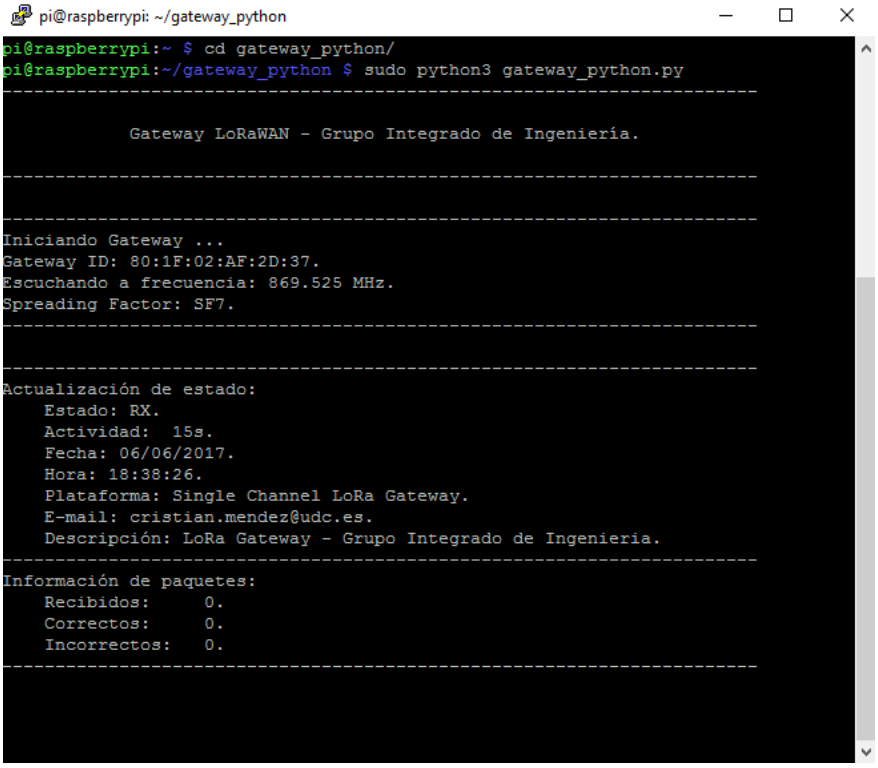


Ilustración 4.4.7: Funcionamiento Normal del Gateway.

La aplicación realizada para el Gateway está expuesta en el Anexo IV (Aplicaciones Desarrolladas) y consistirá en un bucle de lectura de mensajes entrantes. Si se recibe una IRQ desde el dio0, tendremos un paquete recibido en el buffer FIFO del transceptor SX1276, al cual se accederá por medio del bus de comunicaciones SPI. Si el mensaje que se ha recibido es correcto, se realiza la desencriptación de la carga útil y se convierte de hexadecimal a código ASCII.

Como podemos ver en esta simulación, se inicia el Gateway mostrando su dirección MAC y sus parámetros de trabajo, frecuencia de 433 MHz con SF7BW125. Tras esto se recibe un mensaje en hexadecimal desde un end-node y el Gateway mostrará como resultado los parámetros del envío recibido como el RSSI (nivel de potencia de la señal recibida), la SNR (relación señal/ruido), la longitud del paquete y la transcripción del mensaje de hexadecimal a código ASCII.

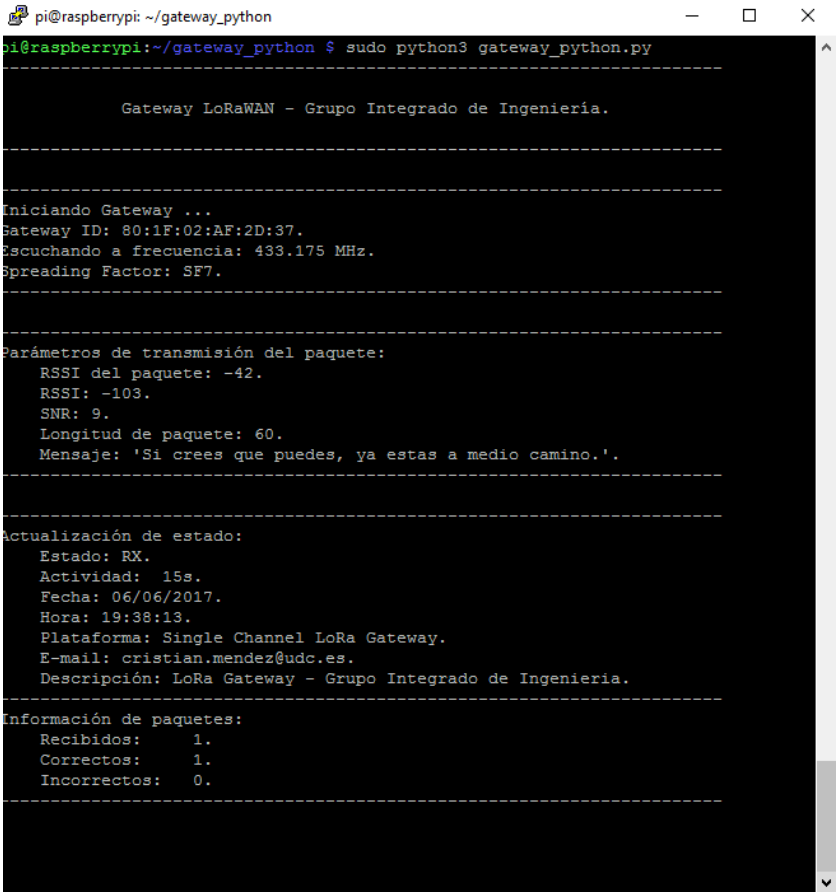


Ilustración 4.4.8: Mensaje Recibido por el Gateway.

Como podemos observar, el código desarrollado nos permite recibir la actualización del estado del Gateway y el estado de paquetes en el intervalo de tiempo pre-configurado. Además de esto, si se activa el parámetro Statistics por código, se recibirán también los tiempos en los cuales se ha realizado una recepción de mensaje (Timestamp) y las medias del RSSI y el SNR.

Como función adicional, se ha desarrollado también un Servidor en Python para la monitorización del Gateway desde diferentes puntos de la topología de red desarrollada. Este servidor se inicia en modo de espera y en el momento que el Gateway se activa, establece comunicación por medio del protocolo de comunicación User Datagram Protocol (UDP) perteneciente a la Capa de Transporte (véase subcapítulo 3.7).

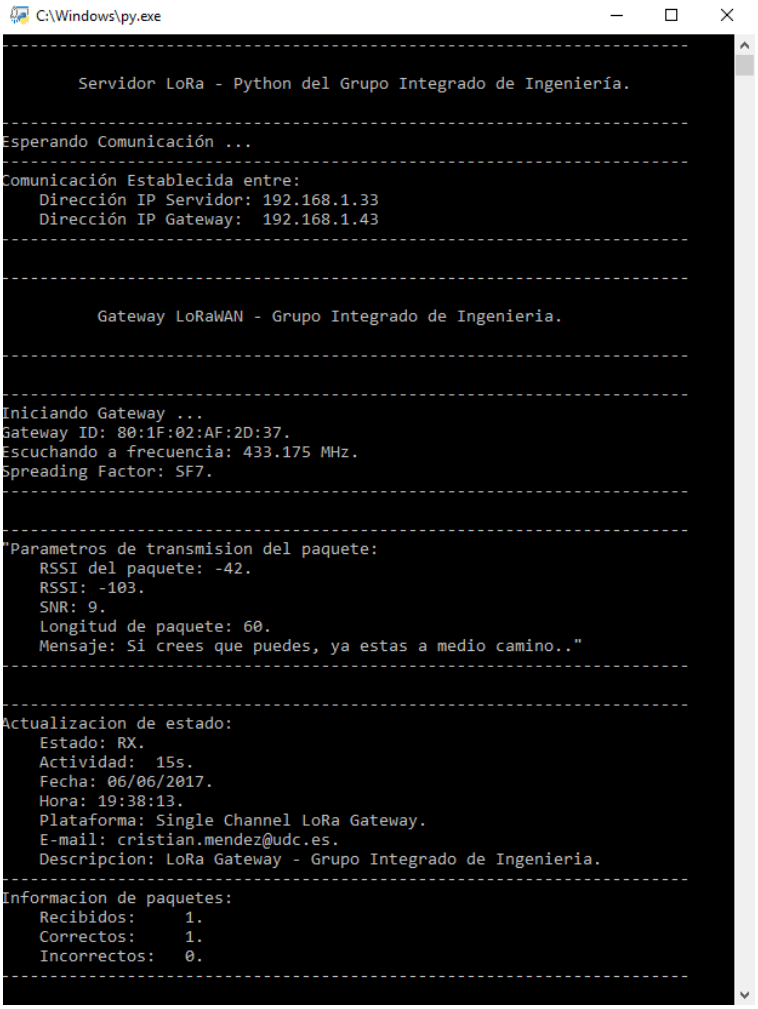


Ilustración 4.4.9: Mensaje Recibido por el Servidor Python.

Con vistas a una posible migración, como se ha planteado previamente en el subcapítulo 4.3 el código ha sido desarrollado para permitir la activación del Modo Híbrido modificando el servidor y la codificación de mensaje, combinando el modo LoRaWAN™, P2P y el soporte de la plataforma libre The Things Network™.

Como último paso, se realizará la configuración de los End-Nodes. Debido a la mayor simplicidad de la Raspberry Pi Modelo B será más sencillo instalar Raspbian Jessie Lite. Para ello debemos descomprimir la imagen previamente descargada para el Gateway y cargarla en la SD externa que irá en la Raspberry. En este caso, se ha llevado a cabo por medio del programa Etcher.

Tras esto se debe activar el módulo wifi, el servidor SSH e instalar la IDE de Python junto con la librería NTPLib, como previamente se ha realizado para el Gateway.

Una vez realizado esto, nos queda cargar el programa desarrollado en Python para establecer su funcionamiento normal. Primero se creará una comunicación serie entre la Raspberry y el módulo RN2483 de Microchip y, tras ello, se cargará la configuración del End-Node y se iniciará el bucle de envío de paquetes.

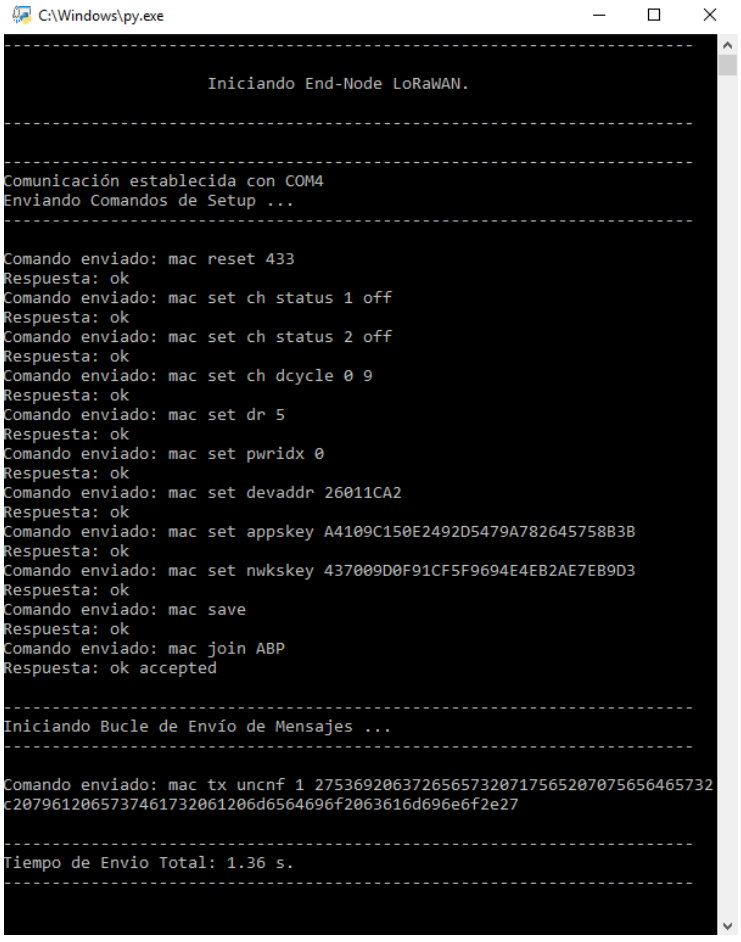


Ilustración 4.4.10: Funcionamiento Normal del End-Node.

Como podemos observar, el código también nos permite activar el parámetro Statistics por código, por el cual se obtendrán los tiempos en los cuales se ha realizado el envío del paquete.

4.5 PRUEBAS DE LA TECNOLOGÍA EN ESCENARIOS REALES

En los apartados anteriores se presentan las principales características de la tecnología LoRa y LoRaWAN junto con los principales componentes de nuestra red. En este apartado, se llevará a cabo el desarrollo de pruebas en las que analizar los resultados de su aplicación en un sistema de UAV's. Estas pruebas consistirán en analizar el consumo energético, el throughput o rendimiento, la latencia y el alcance máximo, en los diferentes estados de transmisión.

- Estudio de Consumo Energético:

Uno de los parámetros más importantes será el análisis del consumo de los end-nodes debido a que esto nos afectará directamente a la autonomía de vuelo. Para analizar esto, debemos analizar el consumo en el estado de funcionamiento normal del UAV, que en nuestro caso será el modo que nos asegure el mayor ancho de banda (DR5).

DR	Configuración	Tamaño Máx. Mensaje	Bitrate Capa Física
DR0	LoRa: SF12 / 125 kHz	64 bytes	250 bits/s
DR1	LoRa: SF11 / 125 kHz	64 bytes	440 bits/s
DR2	LoRa: SF10 / 125 kHz	64 bytes	980 bits/s
DR3	LoRa: SF9 / 125 kHz	128 bytes	1.760 bits/s
DR4	LoRa: SF8 / 125 kHz	255 bytes	3.125 bits/s
DR5	LoRa: SF7 / 125 kHz	255 bytes	5.470 bits/s

Tabla 4.5.1: Características de los distintos modos de funcionamiento.

Durante el vuelo, el end-node experimentará diferentes estados, como pudiera ser el estado de inactividad o de transmisión y recepción de datos, mostrando diferentes consumos.

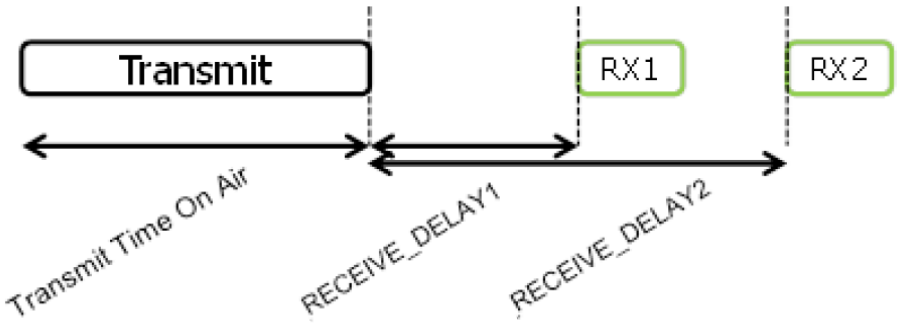


Ilustración 4.5.1: Fases de Transmisión de Mensajes.

Tras realizar el envío de un mensaje, el transceptor abrirá dos ventanas de recepción como indica la LoRaWAN™ Specification, por tanto, para estimar el consumo medio del transceptor debemos aplicar la siguiente fórmula:

$$I_{media} = t_{trans.}\, x\, I_{trans.} + t_{ventana\,1}\, x\, I_{ventana\,1} + t_{ventana\,2}\, x\, I_{ventana\,2} + t_{sleep}\, x\, I_{sleep}$$

Ecuación 4.5.1: Consumo del Transceptor RN2483.

Como en nuestro caso el end-node está compuesto de una Raspberry Pi con un Módulo Wi-Fi y el Módulo Transceptor RN2483, la medición de la intensidad se hace más complicada, por tanto, realizaremos mediciones por medio de un Osciloscopio Tektronix y un circuito con una resistencia de 1 Ω que nos convierte la intensidad a voltaje (1:1), pudiendo ser medido por el osciloscopio. Las mediciones obtenidas han sido las siguientes:

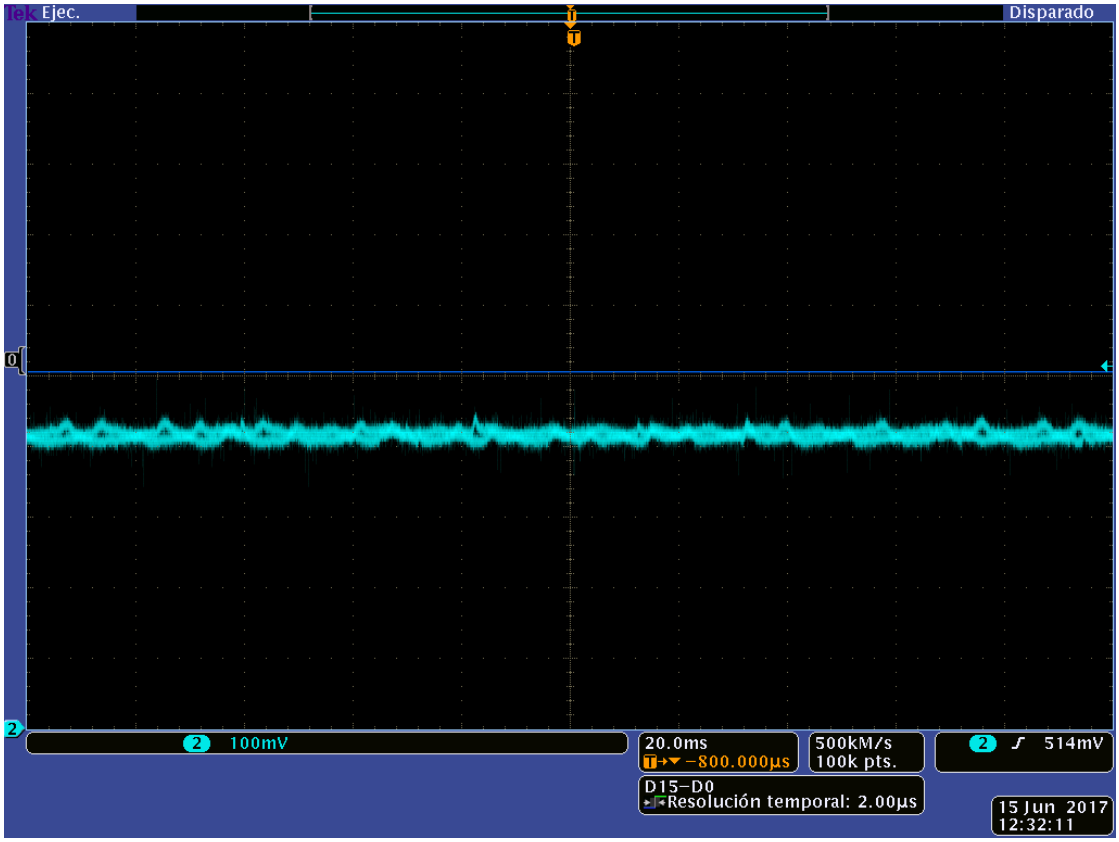


Ilustración 4.5.2: Consumo Medio del End-Node.

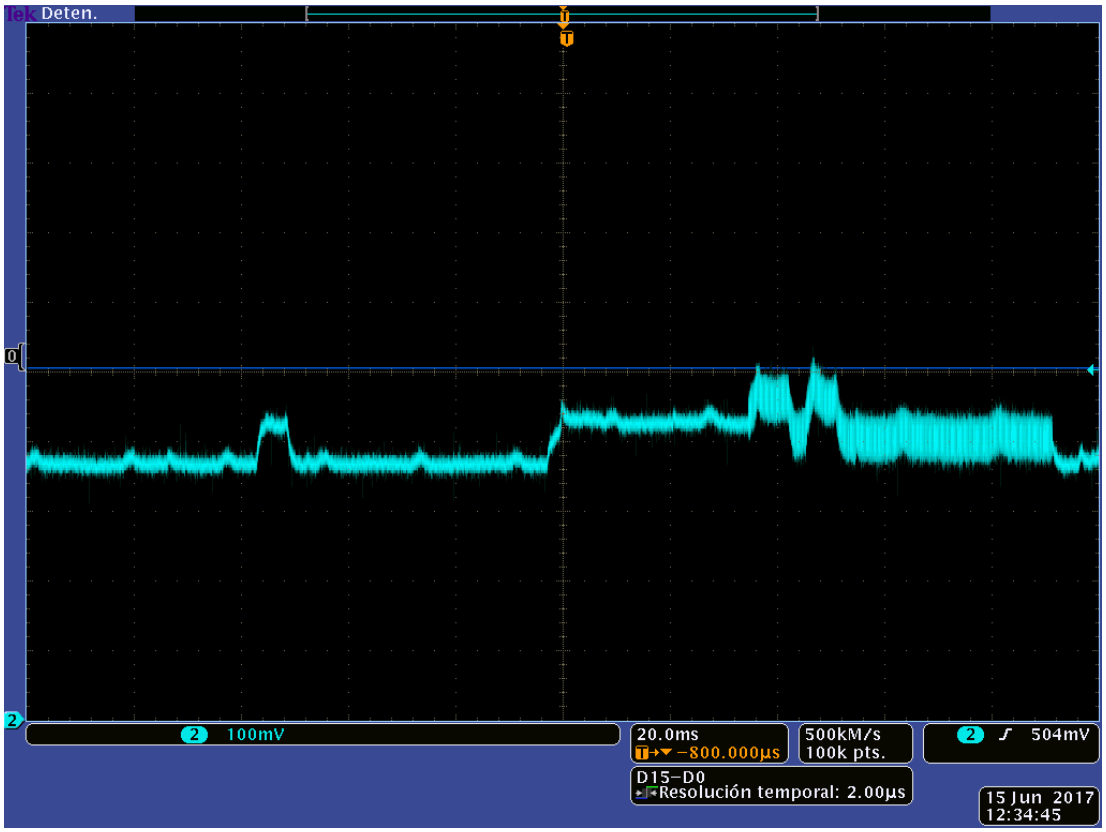


Ilustración 4.5.3: Consumo en Transmisión del End-Node.

El consumo de la Raspberry Pi 1 Modelo B se estima entre 360 mA en reposo y 480 mA en pleno funcionamiento, mientras que el módulo RN2483 se estima en 40 mA en transmisión, 14 mA en recepción y 100-150 µA en reposo.

Nota: el consumo puede variar en función de la versión de firmware de los módulos RN2483 de Microchip respecto al mostrado en el datasheet. http://ww1.microchip.com/downloads/en/DeviceDoc/80000689A.pdf

En la primera foto, en estado de inactividad, observamos que el consumo medio del sistema será aproximadamente de 400 mA, mientras que, en la segunda, ya a pleno rendimiento, podemos destacar que el dispositivo pasa por distintas fases. En una primera podemos observar que el consumo aumenta en el momento que se recibe y ejecuta una orden recibida desde el servidor SSH del módulo Wi-Fi y tras ella, se observa un fuerte pico de consumo, donde se estima que se inicia la comunicación LoRa, finalizando nuevamente en el estado de inactividad.

Los resultados obtenidos y estimaciones realizadas están bastante próximos. Dado que durante el tiempo que se ejecuta la orden recibida del servidor SSH y se realiza la transmisión del mensaje es de aproximadamente de 100 ms, en conjunto a ser utilizado el modo de mayor consumo (DR5), presentando un ritmo mínimo entre mensajes de 2,25s (como se puede observar en la prueba de throughput), se estima un consumo medio total aproximado de 405 mA.

El UAV sobre el que se realizan las pruebas presenta 2 Baterías Turnigy de 5.000 mAh y se estima una autonomía de vuelo de 45 – 60 min según el modo de vuelo.

Ilustración 4.5.4: Batería Turnigy 5.000 mAh.

Estos datos nos permiten estimar la repercusión global de la instalación de este dispositivo sobre la autonomía de vuelo, siendo determinada por la siguiente fórmula:

$$C = I\ x\ t$$

Ecuación 4.5.2: Capacidad de Batería.

Aumentando el consumo previsto (405 mA) al consumo actual (10.000 mA - 13.333 mA), nos dará una autonomía de entre 43,67 – 57,67 min. De esta forma veremos una reducción de autonomía de vuelo en un rango de 1,31 - 2,33 min, lo que se nos presenta como un consumo asumible.

Debe además destacarse que el consumo de LoRa representa un porcentaje muy pequeño en relación al consumo total del dispositivo end-node.

- Estudio Throughput o Rendimiento:

Otro parámetro fundamental a analizar es el throughput o ancho de banda efectivo, siendo este la velocidad real de transporte de la información a través de la red e inferior al ancho de banda. Esta prueba se ha dividido en dos fases, en la primera se ha propuesto el desarrollo de un bucle de envío de mensajes, en concreto 2.400 mensajes con una diferencia de tiempos de 40 (espera) + 10 (timeout) ms, con diferentes longitudes. Los resultados obtenidos han sido los siguientes:

Frecuencia	DR	Recibidos	Tiempo Envío (s)	Ritmo Envío (s)	Bitrate (bits/s)
433 MHz	DR0	11	125.71	11.43	9.80
	DR1	19	126.07	6.64	16.88
	DR2	43	125.48	2.92	38.38
	DR3	52	125.62	2.42	46.36
	DR4	54	125.36	2.32	48.25
	DR5	55	125.41	2.28	49.12
868 MHz	DR0	11	124.96	11.36	9.86
	DR1	19	125.98	6.63	16.89
	DR2	43	125.42	2.92	38.40
	DR3	52	125.43	2.41	46.43
	DR4	54	125.28	2.32	48.28
	DR5	56	125.34	2.24	50.04

Tabla 4.5.2: Throughput con Mínimo Payload.

Frecuencia	DR	Recibidos	Tiempo Envío (s)	Ritmo Envío (s)	Bitrate (bits/s)
433 MHz	DR0	9	149.85	16.65	15.38
	DR1	15	149.31	9.95	25.72
	DR2	33	149.47	4.53	56.52
	DR3	59	149.08	2.53	101.31
	DR4	62	149.05	2.40	106.49
	DR5	64	149.02	2.33	109.94
868 MHz	DR0	9	149.85	16.65	15.38
	DR1	15	149.31	9.95	25.72
	DR2	32	149.51	4.67	54.79
	DR3	58	149.08	2.57	99.60
	DR4	62	149.05	2.40	106.49
	DR5	64	149.02	2.33	109.94

Tabla 4.5.3: Throughput con Mensajes de 32 Bytes.

Frecuencia	DR	Recibidos	Tiempo Envío (s)	Ritmo Envío (s)	Bitrate (bits/s)
433 MHz	DR0	6	149.88	24.98	20.50
	DR1	10	149.53	14.95	34.24
	DR2	22	149.05	6.78	75.57
	DR3	37	148.30	4.01	127.74
	DR4	59	149.10	2.53	202.60
	DR5	62	150.00	2.42	211.63
868 MHz	DR0	6	149.88	24.98	20.50
	DR1	10	149.53	14.95	34.24
	DR2	22	149.02	6.77	75.59
	DR3	37	148.30	4.01	127.74
	DR4	59	149.10	2.53	202.60
	DR5	62	150.00	2.42	211.63

Tabla 4.5.4: Throughput con Mensajes de 64 Bytes.

Frecuencia	DR	Recibidos	Tiempo Envío (s)	Ritmo Envío (s)	Bitrate (bits/s)
433 MHz	DR0	5	126.65	25.33	20.21
	DR1	9	126.35	14.04	36.47
	DR2	18	125.99	7.00	73.15
	DR3	27	186.30	6.90	148.41
	DR4	39	291.38	7.47	273.05
	DR5	69	293.35	4.25	479.84
868 MHz	DR0	5	126.89	25.38	20.17
	DR1	9	126.53	14.06	36.42
	DR2	18	126.06	7.00	73.11
	DR3	26	186.71	7.18	142.60
	DR4	39	193.56	7.53	271.02
	DR5	69	293.21	4.25	480.07

Tabla 4.5.5: Throughput con el Máximo Payload.

Como se observa, el máximo ancho de banda se obtiene al aumentar al máximo la longitud de mensaje, por otra parte, también se ve claramente afectado el tiempo entre mensajes.

De esto obtenemos que sería útil mantener un buffer de memoria para realizar envíos grandes de datos, ya que de por sí, los envíos de paquetes que se llevarán a cabo desde el avión tendrán una longitud aproximada de 50 bytes.

Cabe destacar que el Ciclo de Trabajo utilizado en ambas frecuencias ha sido del 10 %, debido a la limitación impuesta por la Normativa Europea. Esta limitación nos arrastrará el bitrate a valores próximos al 10% del ancho de banda.

El canal de frecuencia usado en 433 MHz corresponde al canal 0 especificado en la LoRaWAN™ Specification (433,175 MHz) y posee una limitación de ciclo de trabajo del 10%, pero los canales en frecuencia 868 MHz disponen de una limitación del 1%, debido a esto, se ha utilizado el canal de frecuencia de 869,525 MHz, que dispone de la limitación del 10%.

La segunda fase de esta prueba, será ajustar los tiempos para asegurar el envío y recepción completa de los paquetes. Para ello, se han ajustado los tiempos y se han enviado 10 mensajes con la diferencia de tiempos de la primera fase para el caso de longitud máxima, que será de donde obtengamos el máximo ancho de banda. Los resultados obtenidos han sido:

Frecuencia	DR	Recepción	Ritmo Ajustado (s)	Tiempo Envío (s)	Bitrate (bits/s)
433 MHz	DR0	10/10	28.25	282.52	18.12
	DR1	10/10	15.75	157.52	32.50
	DR2	10/10	7.25	72.52	70.60
	DR3	10/10	7.00	70.27	145.72
	DR4	10/10	7.50	75.77	269.24
	DR5	10/10	4.50	45.76	445.80
868 MHz	DR0	10/10	28.25	282.52	18.12
	DR1	10/10	15.75	157.52	32.50
	DR2	10/10	7.25	72.52	70.60
	DR3	10/10	7.00	70.27	145.72
	DR4	10/10	7.50	75.77	269.24
	DR5	10/10	4.50	45.77	445.71

Tabla 4.5.6: Throughput Ajustado con el Máximo Payload.

Como resultado de esta prueba, podemos decir que el protocolo LoRa dispone de un throughput limitado, que nos permitirá el envío de pequeños volúmenes de datos para usos que no requieran de transmisión en continuo. En nuestro caso, para transmitir datos de telemetría periódicamente.

- Estudio de Latencia:

Otro punto a tratar será la latencia, resultando esta la suma de tiempos desde que se realiza el envío del paquete hasta obtener el mensaje decodificado. La latencia está relacionada con la ecuación 4.2.3, a mayor tiempo de envío (Time on Air), más tiempo deberá permanecer el End-Node en espera para realizar el siguiente, afectando directamente al ancho de banda.

Para llevar a cabo esta prueba, se ha sincronizado el End-Node y el Gateway con un servidor NTP (protocolo para sincronización de relojes) para obtener la misma referencia de reloj, tras esto, se han enviado 10 mensajes y por medio de la diferencia entre el timestamp en envío y recepción, se calculará la latencia. Los resultados obtenidos han sido los siguientes:

Frecuencia	DR	Media Tiempo Envío (s)	Media Tiempo Recepción (s)	Latencia (s)
433 MHz	DR0	1496573953	1496573956	3.111
	DR1	1496571963	1496571965	1.874
	DR2	1496570403	1496570404	0.997
	DR3	1496568843	1496568844	1.092
	DR4	1496567308	1496567310	1.270
	DR5	1496564793	1496564794	0.969
868 MHz	DR0	1496605447	1496605450	3.174
	DR1	1496602369	1496602371	1.912
	DR2	1496600716	1496600714	0.995
	DR3	1496599204	1496599205	1.089
	DR4	1496577022	1496577023	1.274
	DR5	1496575465	1496575466	0.937

Tabla 4.5.7: Latencia con Mensajes de Máximo Payload.

Como resultado de esta prueba se puede destacar que el protocolo LoRa incrementará la latencia desde el modo DR5 hacia el modo DR0. Esto se debe a que el espectro de radiofrecuencia será más esparcido, mayor factor de ensanchamiento (SF), permitiéndonos un alcance muy superior. Por la contra, como se ha dicho, para obtener un mayor alcance, se verá gravemente afectado el ancho de banda, siendo este muy inferior al obtenido en el modo DR5.

- Estudio de Alcance:

Como último, se procederá a realizar la prueba del alcance máximo. Para ello, compararemos los niveles de RSSI (nivel de potencia de la señal recibida), la SNR (relación señal/ruido) y Porcentaje de Paquetes Perdidos, en puntos situados a diferentes distancias.

Para realizar la prueba se enviarán 100 mensajes de 17 bytes, en modo DR5, con el fin de saber el máximo alcance que tenemos con el modo que nos permite el máximo ancho de banda.

Esta prueba se realizará en el La Malata (Ferrol), sobre tierra, debido a las malas condiciones por viento presentadas durante los días a realizar las pruebas, lo que nos impide realizarlas en vuelo. El alcance máximo que nos interesa será marcado por las Reglas de Vuelo Visual (VFR), realizando pruebas de alcance hasta una distancia máxima de 1 km. El Gateway y el End-Node activarán su funcionamiento por medio de dos Access-Points de forma independiente.

Los puntos sobre los que se harán las mediciones y las distancias entre el Gateway y el End-Node serán marcados en la siguiente imagen:

Ilustración 4.5.5: Puntos de Medición de Alcance.

Una vez establecidos los puntos desde donde se realizarán las pruebas de alcance, se comenzará la prueba, resultando para ambas frecuencias:

Puntos	Distancia	RSSI Paquete (dB)	RSSI (dB)	SNR (dB)	Paquetes Recibidos
1	252 m	- 110	- 105	4	98 %
2	510 m	- 115	- 106	8	42 %
3	756 m	- 113	- 110	9	63 %
4	1.040 m	- 114	- 108	10	25 %

Tabla 4.5.8: Resultados de Alcance en Banda de Frecuencia de 433 MHz.

Puntos	Distancia	RSSI Paquete (dB)	RSSI (dB)	SNR (dB)	Paquetes Recibidos
1	252 m	- 104	- 109	4	98 %
2	510 m	- 107	- 107	4	84 %
3	756 m	- 113	- 113	6	81 %
4	1.040 m	- 113	- 116	7	55 %

Tabla 4.5.9: Resultados de Alcance en Banda de Frecuencia de 868 MHz.

Como observamos, la potencia de la señal de la banda de frecuencia de 868 MHz nos afecta en la calidad de señal, permitiendo un mejor ratio de paquetes recibidos respecto a 433 MHz. A su vez, al mantener línea de visión, se mantendrá un alto nivel de paquetes recibidos hasta distancias cercanas a los 750 m, a partir de donde podemos considerar que distancia comienza a afectar la recepción de paquetes, reflejándose en el parámetro SNR.

Ilustración 4.5.6: Pruebas de Alcance en el Punto 2.

Como último punto, cabe destacar la anomalía detectada en la medición del Punto 2 en la banda de frecuencia de 433 MHz con un valor anómalo que no sigue la tendencia lógica. Se ha asumido este valor como el resultado de algún tipo de interferencia o apantallamiento.

5 CONCLUSIONES

A lo largo del presente proyecto, se ha realizado un análisis de las tecnologías de comunicaciones LPWAM con el fin de obtener una idoneidad de adopción a un enjambre de UAV's colaborativos.

Las tecnologías más interesantes respecto a las características de diseño del proyecto han sido LoRa, Sigfox y NB-IoT, siendo finalmente elegida LoRa debido a sus numerosas ventajas como la versatilidad de sus modos de transmisión, el alcance, la eficiencia energética, la mayor inmunidad a interferencias y su disponibilidad actual.

Como principal hándicap, se presenta la limitación del ciclo de trabajo, no permitiendo explotar el máximo ancho de banda posible e impidiéndonos la posibilidad de transmisión en continuo.

En cuanto a la configuración del sistema, se ha realizado una topología en estrella, disponiendo los end-nodes en los UAV's y disponiendo un Gateway o estación central, en tierra, desde donde se realizará la recepción y tratamiento de los datos.

Para llevar a cabo la optimización del sistema, se han realizado pruebas de consumo energético, throughput, latencia y alcance bajo las Reglas de Vuelo Visual (VFR). Las conclusiones han sido que LoRa ha cumplido los objetivos propuestos, suponiendo al UAV un consumo energético que no influye de manera elevada sobre su autonomía de vuelo y permitiendo obtener el mayor ancho de banda en línea de visión directa con radioenlace estable a grandes distancias.

La programación de los componentes ha sido la parte más compleja del proyecto debido a que LoRa se trata de una tecnología emergente y existe relativamente poca información en la red, de hecho, los desarrollos encontrados no implementan todas las características o se encuentran bajo fase de desarrollo actualmente.

La programación de componentes ha sido realizada íntegramente en lenguaje de programación Python 3 debido a ser multiplataforma y software libre, disponer de fácil acceso a documentación y ser muy intuitivo a la hora de programar.

Como líneas futuras, surge la necesidad de sincronizar la lectura de los sensores al programa de end-nodes en Python y generar un caudal de datos que pueda ofrecernos una visión real del vuelo. Además, el Gateway se encuentra en fase de desarrollo, pudiendo ser programadas nuevas características como Channel Activity Detection (CAD) para la detección automática del Spreading Factor o Frequency Hopping para la detección de mensajes desde distintos canales.

6 BIBLIOGRAFÍA

[1] Gemalto, Actility and Semtech, "LoRaWAN Security. Full end-to-end encryption for IoT Application Providers", *LoRa Alliance White Papers*, February 2017.

Available in: www.lora-alliance.org

[2] George Margelis, Robert Piechocki, Dritan Kaleshi, Paul Thomas, "Low throughput networks for the IoT: Lessons learned from industrial implementations", *IEEE, Internet of Things (WF-IoT), 2015 IEEE 2nd World Forum On*, January 2016.

Available in: www.ieeexplore.ieee.org

[3] Hardy Schmidbauer, "NB-IoT vs LoRa Technology. Which could take gold?", *LoRa Alliance White Papers*, September 2016.

Available in: www.lora-alliance.org

[4] Jean-Paul Bardyn, Thierry Melly, Olivier Seller, Nicolas Sornin, "IoT: The era of LPWAN is starting now", *IEEE, European Solid-State Circuits Conference, ESSCIRC Conference 2016: 42nd*, October 2016.

Available in: www.ieeexplore.ieee.org

[5] Juha Petäjäjärvi, Konstantin Mikhaylov, Antti Roivainen, Tuomo Hänninen, "Range evaluation and channel attenuation model for LoRa Technology", *IEEE, 14th International Conference on ITS Telecommunications (ITST)*, December 2015.

Available in: www.ieeexplore.ieee.org

[6] Juha Petäjäjärvi, Konstantin Mikhaylov, Marko Pettissalo, Janne Janhunen and Jari Iinatti, "Performance of a low power wide area network based on LoRa Technology: Doppler Robustness, Scalability and Coverage", *Sage Journals,* volume 13, issue 3, March 2017.

Available in: www.journals.sagepub.com

[7] Konstantin Mikhaylov, Juha Petäjäjärvi, Janne Janhunen, "On LoRaWAN Scalability: Empirical evaluation of susceptibility to inter-network interference", *pre-print of article to be presented at the European Conference on Networking and Communications (EUCNC)*, June 2017.

Available in: www.arxiv.org

[8] Konstantin Mikhaylov, Juha Petäjäjärvi, Tuomo Hänninen, "Analysis of capacity and scalability of the LoRa low power wide area network technology", *VDE*, *European Wireless 2016, 22th European Wireless Conference*, June 2016.

Available in: www.ieeexplore.ieee.org

[9] LoRa Alliance Technical Marketing Workgroup, "LoRaWAN. What is it? A technical overview of LoRa and LoRaWAN", *LoRa Alliance White Papers*, November 2015.

Available in: www.lora-alliance.org

[10] Machina Research, "LPWA Technologies. Unlock new IoT market potential", *LoRa Alliance White Papers*, November 2015.

Available in: www.lora-alliance.org

[11] N.Sornin (Semtech), M.Luis (Semtech), T. Eirich (IBM), T. Kramp (IBM) , O.Hersent (Actility), "LoRaWAN Specification V1.0", *LoRa Alliance*, January 2015.

Available in: www.lora-alliance.org

[12] Rashmi Sharan Sinha, Yiqiao Wei and Seung-Hoon Hwang, "A survey on LPWA Technology: LoRa and NB-IoT", *ICT Express*, volume 3 (issue 1), pages 14-21, March 2017.

Available in: www.sciencedirect.com

[13] Thomas Telkamp, "LoRa crash Course", *The Things Network*, Webinar, November 2016.

Available in: www.thethingsnetwork.org

UNIVERSIDADE DA CORUÑA

ESCUELA POLITÉCNICA SUPERIOR

TRABAJO FIN DE MÁSTER

CURSO 2016/2017

DISEÑO Y DESARROLLO DEL SISTEMA DE COMUNICACIONES INALÁMBRICO DE UN ENJAMBRE DE UAV'S COLABORATIVOS

MÁSTER EN INGENIERÍA INDUSTRIAL

DOCUMENTO DEL PROYECTO

ANEXOS

ÍNDICE DE ANEXOS

UNIVERSIDADE DA CORUÑA

ESCUELA POLITÉCNICA SUPERIOR

TRABAJO FIN DE MÁSTER

CURSO 2016/2017

DISEÑO Y DESARROLLO DEL SISTEMA DE COMUNICACIONES INALÁMBRICO DE UN ENJAMBRE DE UAV'S COLABORATIVOS

MÁSTER EN INGENIERÍA INDUSTRIAL

DOCUMENTO DEL PROYECTO

ANEXO I - NORMATIVA

NORMATIVA EUROPEA PARA DISPOSITIVOS DE CORTO ALCANCE (2016/771/EC)

DECISIÓN DE LA COMISIÓN

de 9 de noviembre de 2006

sobre la armonización del espectro radioeléctrico para su uso por dispositivos de corto alcance

[notificada con el número C(2006) 5304]

(Texto pertinente a efectos del EEE)

(2006/771/CE)

LA COMISIÓN DE LAS COMUNIDADES EUROPEAS,

Visto el Tratado constitutivo de la Comunidad Europea,

Vista la Decisión nº 676/2002/CE del Parlamento Europeo y del Consejo, de 7 de marzo de 2002, sobre un marco regulador de la política del espectro radioeléctrico en la Comunidad Europea (Decisión del espectro radioeléctrico) [1], y, en particular, su artículo 4, apartado 3,

Considerando lo siguiente:

(1) Dada su amplia utilización en la Comunidad Europea y en el mundo, los dispositivos de corto alcance están desempeñando un papel cada vez más importante en la economía y en la vida de los ciudadanos, teniendo diferentes tipos de aplicaciones como alarmas, equipos de comunicaciones locales, aparatos de apertura de puertas o implantes sanitarios. El desarrollo de aplicaciones basadas en los dispositivos de corto alcance en la Comunidad Europea podría contribuir también a la consecución de objetivos concretos de las políticas comunitarias, como la plena realización del mercado interior, la promoción de la innovación y la investigación, y el desarrollo de la sociedad de la información.

(2) Los dispositivos de corto alcance son generalmente productos portátiles o con un mercado de masas que pueden llevarse y utilizarse a través de las fronteras con facilidad; por ello, las diferencias en las condiciones de acceso al espectro impiden su libre circulación, aumentan sus costes de producción y crean riesgos de interferencias perjudiciales con otras aplicaciones y servicios radioeléctricos. A fin de recoger los beneficios que aporta el mercado interior para este tipo de dispositivos, impulsar la competitividad de la industria manufacturera comunitaria al aumentar las economías de escala, y rebajar los costes a los consumidores, el espectro radioeléctrico debe ofrecerse en la Comunidad con arreglo a unas condiciones técnicas armonizadas.

(3) Como este tipo de dispositivo utiliza el espectro radioeléctrico con una baja potencia de emisión y una capacidad de emisión de corto alcance, sus posibilidades de causar interferencias con otros usuarios del espectro es normalmente muy limitada. Por lo tanto, estos aparatos pueden compartir bandas de frecuencias con otros servicios, sujetos, o no, a autorización particular, sin causar interferencias perjudiciales, y pueden coexistir con otros dispositivos de corto alcance. Por consiguiente, su uso no debe estar sujeto a autorización con arreglo a la Directiva 2002/20/CE del Parlamento Europeo y del Consejo [2]. Además, los servicios de radiocomunicaciones, definidos en el Reglamento de Radiocomunicaciones de la Unión Internacional de Telecomunicaciones, tienen preferencia sobre los dispositivos de corto alcance y no están obligados a asegurar la protección contra interferencias de determinados tipos de dispositivos de corto alcance. Por lo tanto, al no poder asegurarse a los usuarios de dispositivos de corto alcance una protección contra interferencias, es responsabilidad de los fabricantes de estos aparatos protegerlos contra interferencias perjudiciales de los servicios de radiocomunicaciones, así como de otros dispositivos de corto alcance que funcionen de acuerdo con las disposiciones nacionales o comunitarias aplicables. En virtud de la Directiva 1999/5/CE del Parlamento Europeo y del Consejo, de 9 de marzo de 1999, sobre equipos radioeléctricos y equipos terminales de telecomunicación y reconocimiento mutuo de su conformidad [3], los fabricantes deben asegurar que los dispositivos de corto alcance utilicen de forma eficaz el espectro radioléctrico para evitar las interferencias perjudiciales con otros dispositivos del mismo tipo.

(4) Un buen número de estos dispositivos ya están clasificados, o es probable que lo estén en el futuro, como equipos de la «categoría 1» en virtud de la Decisión 2000/299/CE de la Comisión, de 6 de abril de 2000, relativa al establecimiento de la clasificación inicial de los equipos radioeléctricos y equipos terminales de telecomunicación y los identificadores asociados [4], adoptada de conformidad con el artículo 4, apartado 1, de la Directiva sobre equipos radioeléctricos y equipos terminales de telecomunicación. La Decisión 2000/299/CE reconoce la equivalencia de las interfaces radioeléctricas que cumplan las condiciones de la «categoría 1», de manera que estos equipos pueden comercializarse y ponerse en servicio sin restricciones en toda la Comunidad.

(5) Dado que la disponibilidad de espectro armonizado y las condiciones de uso consiguientes determinan la clasificación como «categoría 1», la presente Decisión reforzará la continuidad de esta clasificación, una vez conseguida.

[1] DO L 108 de 24.4.2002, p. 1.

[2] DO L 108 de 24.4.2002, p. 21.

[3] DO L 91 de 7.4.1999, p. 10.

[4] DO L 97 de 19.4.2000, p. 13.

(6) Por ello, el 11 de marzo de 2004, la Comisión dio un mandato ([5]) a la CEPT, en virtud del artículo 4, apartado 2, de la Decisión sobre el espectro radioeléctrico, para armonizar el uso de frecuencias de los dispositivos de corto alcance. Respondiendo a ese mandato, en su informe ([6]) de 15 de noviembre de 2004, la CEPT estableció la lista de medidas voluntarias de armonización existentes en la Comunidad Europea para los dispositivos de corto alcance y declaró que se requería una compromiso más vinculante por parte de los Estados miembros a fin de asegurar la estabilidad jurídica de la armonización de frecuencias conseguida en la CEPT. Por tanto, es necesario establecer un mecanismo para hacer que estas medidas de armonización sean jurídicamente vinculantes en la Comunidad Europea.

(7) Los Estados miembros pueden permitir, a nivel nacional, el funcionamiento de equipos en condiciones más permisivas que las especificadas en la presente Decisión. Sin embargo, en este caso estos equipos no podrían funcionar en toda la Comunidad sin restricciones y, por tanto, se considerarían equipos de la «categoría 2» según la clasificación de la Directiva sobre equipos radioeléctricos y equipos terminales de telecomunicación.

(8) La armonización con arreglo a la presente Decisión no excluye la posibilidad de que un Estado miembro aplique, cuando esté justificado, períodos de transición o acuerdos de uso compartido del espectro radioeléctrico. Estos períodos o medidas deben reducirse al mínimo, ya que limitarían los beneficios que reporta la clasificación en la «categoría 1».

(9) La presente Decisión general de armonización técnica se entiende sin perjuicio de las medidas comunitarias de armonización técnica que se aplican a determinados tipos de dispositivos y bandas, como la Decisión 2004/545/CE de la Comisión, de 8 de julio de 2004, relativa a la armonización del espectro radioeléctrico en la gama de 79 GHz para el uso de equipos de radar de corto alcance para automóviles en la Comunidad ([7]), la Decisión 2005/50/CE de la Comisión, de 17 de enero de 2005, relativa a la armonización del espectro radioeléctrico en la banda de 24 GHz para el uso temporal por equipos de radar de corto alcance para automóviles en la Comunidad ([8]), la Decisión 2005/513/CE de la Comisión, de 11 de julio de 2005, por la que se armoniza la utilización del espectro radioeléctrico en la banda de frecuencias de 5 GHz con vistas a la aplicación de los sistemas de acceso inalámbrico, incluidas las redes radioeléctricas de área local (WAS/RLAN) ([9]), o la Decisión 2005/928/CE de la Comisión, de 20 de diciembre de 2005, sobre la armonización de la banda de frecuencias de 169,4 a 169,8125 MHz en la Comunidad ([10]).

([5]) Mandato a la CEPT para analizar la mayor armonización de las bandas de frecuencia utilizadas para dispositivos de corto alcance (DCA).

([6]) Informe final del Comité de Comunicaciones Electrónicas (EEC, en su sigla inglesa) en respuesta al mandato de la CE a la CEPT sobre la armonización del espectro radioeléctrico de los dispositivos de corto alcance.

([7]) DO L 241 de 13.7.2004, p. 66.

([8]) DO L 21 de 25.1.2005, p. 15.

([9]) DO L 187 de 19.7.2005, p. 22.

([10]) DO L 344 de 27.12.2005, p. 47.

(10) El uso del espectro está sujeto a las obligaciones que impone la legislación comunitaria sobre protección de la salud pública, en particular la Directiva 2004/40/CE del Parlamento Europeo y del Consejo ([11]) y la Recomendación 1999/519/CE del Consejo ([12]). La protección de la salud en lo que respecta a los equipos radioeléctricos está asegurada por la conformidad de estos equipos con los requisitos esenciales que establece la Directiva sobre equipos radioeléctricos y equipos terminales de telecomunicación.

(11) Debido a los rápidos cambios experimentados en la tecnología y la demanda social, surgirán nuevas aplicaciones de los dispositivos de corto alcance que requerirán un examen constante de las condiciones de armonización del espectro, teniendo en cuenta los beneficios económicos de las nuevas aplicaciones y las necesidades de la industria y los usuarios. Los Estados miembros tendrán que estar atentos a esta evolución. Por tanto, serán necesarias actualizaciones periódicas de la presente Decisión para responder a la evolución del mercado y la tecnología. Por ello, el anexo se revisará, al menos, una vez al año basándose en la información recogida por los Estados miembros y facilitada a la Comisión. También podrá iniciarse una revisión cuando algún Estado miembro haya tomado las medidas adecuadas a las que se refiere el artículo 9 de la Directiva sobre equipos radioeléctricos y equipos terminales de telecomunicación. Cuando una revisión muestre la necesidad de adaptar la Decisión, las modificaciones a que haya lugar se decidirán ateniéndose a los procedimientos especificados en la Decisión sobre el espectro radioeléctrico para la adopción de las medidas de aplicación. Las actualizaciones podrían incluir períodos de transición, teniendo en cuenta la existencia de situaciones heredadas.

(12) Las medidas previstas en la presente Decisión se ajustan al dictamen del Comité del Espectro Radioeléctrico.

HA ADOPTADO LA PRESENTE DECISIÓN:

Artículo 1

La finalidad de la presente Decisión es armonizar las bandas de frecuencias y los parámetros técnicos conexos a fin de poder disponer de espectro radioeléctrico para los dispositivos de corto alcance y de conseguir un mejor aprovechamiento de este, de manera que estos dispositivos puedan beneficiarse de la clasificación en la «categoría 1» de la Decisión 2000/299/CE de la Comisión.

Artículo 2

A efectos de la presente Decisión:

1) «dispositivos de corto alcance» significa los transmisores de radio que proporcionan comunicación unidireccional o bidireccional y transmiten a corta distancia y baja potencia;

([11]) DO L 159 de 30.4.2004, p. 1. Versión corregida en el DO L 184 de 24.5.2004, p. 1.

([12]) DO L 199 de 30.7.1999, p. 59.

2) «sobre una base de ausencia de interferencia y de protección» significa que no puede causarse interferencia perjudicial a ningún servicio de radiocomunicaciones y que no puede solicitarse la protección de estos dispositivos frente a las interferencias perjudiciales producidas por servicios de radiocomunicaciones.

Artículo 3

1. Los Estados miembros designarán y facilitarán, sobre una base no exclusiva de ausencia de interferencia y de protección, las bandas de frecuencias para los tipos de dispositivos de corto alcance con sujeción a las condiciones específicas y al plazo de aplicación que establece el anexo de la presente Decisión.

2. No obstante lo dispuesto en el apartado 1, los Estados miembros podrán solicitar unos períodos de transición o acuerdos de reparto del espectro radioeléctrico con arreglo al artículo 4, apartado 5, de la Decisión del espectro radioeléctrico.

3. La presente Decisión se entenderá sin perjuicio del derecho de los Estados miembros a permitir el uso de las bandas de frecuencias en condiciones menos restrictivas que las especificadas en el anexo de la presente Decisión.

Artículo 4

Los Estados miembros mantendrán bajo escrutinio el uso de las bandas correspondientes e informarán a la Comisión al respecto para permitir una revisión de la Decisión de manera periódica y en el momento oportuno.

Artículo 5

Los destinatarios de la presente Decisión serán los Estados miembros.

Hecho en Bruselas, el 9 de noviembre de 2006.

Por la Comisión

Viviane REDING

Miembro de la Comisión

ANEXO

Bandas de frecuencias y parámetros técnicos armonizados para dispositivos de corto alcance

Tipo de dispositivo de corto alcance	Banda(s) de frecuencias/ Frecuencias únicas	Potencia máxima/ Fuerza del campo	Parámetros reglamentarios adicionales/Requisitos de mitigación	Otras restricciones	Plazo de aplicación
Dispositivos de corto alcance no específicos [1]	26,957-27,283 MHz	10 mW de potencia radiada efectiva (p.r.e.), que corresponde a 42 dBµA/m a 10 metros		Se excluyen las aplicaciones de vídeo	1 de junio de 2007
	40,660-40,700 MHz	10 mW p.r.e.		Se excluyen las aplicaciones de vídeo	1 de junio de 2007
	433,05-434,79 MHz	10 mW p.r.e.	Ciclo de ocupación [2]: hasta el 10 %	Se excluyen las señales de audio y vídeo y las aplicaciones de vídeo	1 de junio de 2007
	868,0-868,6 MHz	25 mW p.r.e.	Ciclo de ocupación [2]: hasta el 1 %	Se excluyen las aplicaciones de vídeo	1 de junio de 2007
	868,7-869,2 MHz	25 mW p.r.e.	Ciclo de ocupación [2]: hasta el 0,1 %	Se excluyen las aplicaciones de vídeo	1 de junio de 2007
	869,4-869,65 MHz	500 mW p.r.e.	Ciclo de ocupación [2]: hasta el 10 % Separación entre canales: 25 kHz, teniendo en cuenta como excepción que también puede usarse toda la banda como canal único para la transmisión de datos a alta velocidad	Se excluyen las aplicaciones de vídeo	1 de junio de 2007
	869,7-870 MHz	5 mW p.r.e.	Se permiten aplicaciones de voz con técnicas de mitigación avanzadas	Se excluyen las aplicaciones de vídeo y audio	1 de junio de 2007
	2 400-2 483,5 MHz	10 mW de potencia radiada isotrópica equivalente (p.r.i.e.)			1 de junio de 2007
	5 725-5 875 MHz	25 mW e.i.r.p.			1 de junio de 2007
Sistemas de alarma	868,6-868,7 MHz	10 mW p.r.e.	Separación entre canales: 25 kHz También puede usarse toda la banda como canal único para la transmisión de datos a alta velocidad Ciclo de ocupación [2]: hasta el 0,1 %		1 de junio de 2007
	869,25-869,3 MHz	10 mW p.r.e.	Separación entre canales: 25 kHz Ciclo de ocupación [2]: por debajo del 0,1 %		1 de junio de 2007
	869,65-869,7 MHz	25 mW p.r.e.	Separación entre canales: 25 kHz Ciclo de ocupación [2]: por debajo del 10 %		1 de junio de 2007

Tipo de dispositivo de corto alcance	Banda(s) de frecuencias/ Frecuencias únicas	Potencia máxima/ Fuerza del campo	Parámetros reglamentarios adicionales/Requisitos de mitigación	Otras restricciones	Plazo de aplicación
Alarmas de teleasistencia ([3])	869,20-869,25 MHz	10 mW p.r.e.	Separación entre canales: 25 kHz Ciclo de ocupación ([2]): por debajo del 0,1 %		1 de junio de 2007
Aplicaciones inductivas ([4])	20,05-59,75 kHz	72 dBµA/m a 10 metros			1 de junio de 2007
	59,75-60,25 kHz	42 dBµA/m a 10 metros			1 de junio de 2007
	60,25-70 kHz	69 dBµA/m a 10 metros			1 de junio de 2007
	70-119 kHz	42 dBµA/m a 10 metros			1 de junio de 2007
	119-127 kHz	66 dBµA/m a 10 metros			1 de junio de 2007
	127-135 kHz	42 dBµA/m a 10 metros			1 de junio de 2007
	6 765-6 795 kHz	42 dBµA/m a 10 metros			1 de junio de 2007
	13,553-13,567 MHz	42 dBµA/m a 10 metros			1 de junio de 2007
Implantes sanitarios activos ([5])	402-405 MHz	25 mW p.r.e.	Separación entre canales: 25 kHz Otras restricciones sobre canales: distintos transmisores pueden combinar canales adyacentes para aumentar el ancho de banda con técnicas de mitigación avanzadas		1 de junio de 2007
Aplicaciones de audio inalámbricas ([6])	863-865 MHz	10 mW p.r.e.			1 de junio de 2007

([1]) Esta categoría está disponible para cualquier tipo de aplicaciones que cumplan las condiciones técnicas (los usos habituales son la telemetría, los mandos a distancia, las alarmas, los datos en general y otras aplicaciones semejantes).

([2]) Ciclo de ocupación: proporción del tiempo en el que un equipo está transmitiendo de forma activa dentro de un período de una hora.

([3]) Los dispositivos de teleasistencia se emplean para asistir a las personas mayores o discapacitadas que viven en sus hogares, en caso de que necesiten socorro.

([4]) Esta categoría cubre, por ejemplo, los dispositivos para la inmovilización de vehículos, la identificación de animales, los sistemas de alarma, la detección de cables, la gestión de residuos, la identificación de personas, los enlaces de voz inalámbricos, el control de acceso, los sensores de proximidad, los sistemas antirrobo —incluidos los sistemas de inducción antirrobo RF—, la transferencia de datos a dispositivos manuales, la identificación automática de artículos, los sistemas de control inalámbricos y el pago de peajes automático.

([5]) Esta categoría cubre la parte radioléctrica de un producto sanitario implantable activo en el sentido del artículo 1 de la Directiva 90/385/CEE del Consejo, de 20 de junio de 1990, relativa a la aproximación de las legislaciones de los Estados miembros sobre los productos sanitarios implantables activos y sus periféricos.

([6]) Aplicaciones para sistemas de audio inalámbricos, incluyendo: altavoces inalámbricos; auriculares inalámbricos; auriculares inalámbricos portátiles, por ejemplo, para aparatos portátiles de lectura de CD o casetes o de radio; auriculares inalámbricos para uso en vehículos, por ejemplo, para una radio o teléfono móvil, etc.; control mediante aparatos en la oreja para uso en conciertos o representaciones.

CUADRO NACIONAL DE ATRIBUCIÓN DE FRECUENCIA (CNAF)

UN - 30 Aplicaciones de baja potencia en banda ICM de 433 MHz

En la banda 433,050-434,790 MHz (Frecuencia central 433,920 MHz), de conformidad con la Decisión de la Comisión 2011/829/UE por la que se modifica la Decisión 2006/771/CE sobre la armonización del espectro radioeléctrico para su uso por dispositivos de corto alcance, así como la Recomendación 70-03 (anexo 1) de la CEPT, se permite con la consideración de uso común, la utilización de dispositivos no específicos de corto alcance (SRD), bajo las siguientes características:

Banda de frecuencias	Potencia	Canalización	Notas
433,050 - 434,790 MHz	1 mW p.r.a. -13 dBm /10 kHz	No se define	Se excluyen aplicaciones de audio y vídeo. Se permiten comunicaciones de voz con técnicas de mitigación avanzada.
433,050 - 434,040 MHz	10 mW p.r.a.	No se define	Ciclo de trabajo ≤ 10%. Se excluyen aplicaciones de audio y vídeo.
434,040 – 434,790 MHz	10 mW p.r.a.	No se define	Ciclo de trabajo ≤ 10%. Se permiten técnicas de mitigación avanzadas
434,040 – 434,790 MHz	10 mW p.r.a.	25 kHz	Se excluyen aplicaciones de audio y vídeo. Se permiten comunicaciones de voz con técnicas de mitigación avanzada.

Tabla AI.2.1: Características de Transmisión en Banda de Frecuencia 433 MHz.

A las utilizaciones descritas en esta nota les es de aplicación el contenido de la nota UN-32 relativa a aplicaciones industriales, científicas y médicas (ICM).

Las instalaciones de este tipo deben de aceptar la interferencia perjudicial que pudiera resultar de aplicaciones ICM u otros usos de radiocomunicaciones en estas frecuencias o en bandas adyacentes.

UN - 32 Aplicaciones ICM en 433 MHz

Banda de frecuencias designada para aplicaciones industriales, científicas y médicas (aplicaciones ICM, no servicios de radiocomunicaciones): 433,050 a 434,790 MHz (frecuencia central 433,920 MHz).

Los servicios de radiocomunicaciones que funcionen en esta banda bajo la consideración de uso común deberán aceptar la interferencia perjudicial resultante de dichas aplicaciones.

UN - 39 Banda de 868-870 MHz

Esta banda de se destina para aplicaciones de baja potencia y de datos en general de conformidad con la Decisión de la Comisión 2011/829/UE por la que se modifica la Decisión 2006/771/CE sobre la armonización del espectro radioeléctrico para su uso por dispositivos de corto alcance, así como la Recomendación 70-03 (anexos 1 y 7) de la CEPT, conforme a la siguiente clasificación de dispositivos:

- Dispositivos de baja potencia no especificados para aplicaciones genéricas:
  - 868 - 868,600 MHz con 25 mW (p.r.a.) de potencia radiada aparente máxima. Estos dispositivos deberán utilizar técnicas de acceso y mitigación de interferencias con rendimiento al menos equivalente a las técnicas descritas en las normas armonizadas según la Directiva 1995/5/CE, o alternativamente no sobrepasar el 1% de ciclo de trabajo.

    Se excluyen las aplicaciones analógicas de vídeo.
  - 868 ,700 – 869,200 MHz con 25 mW (p.r.a.) de potencia radiada aparente máxima. Estos dispositivos deberán utilizar técnicas de acceso y mitigación de interferencias con rendimiento al menos equivalente a las técnicas descritas en las normas armonizadas según la Directiva 1995/5/CE, o alternativamente no sobrepasar el 0,1% de ciclo de trabajo.

    Se excluyen las aplicaciones analógicas de vídeo.
  - 869,400 – 869,650 MHz con 500 mW (p.r.a.) de potencia radiada aparente máxima y 25 kHz de canalización, si bien pudiera utilizarse toda la banda como canal único de datos de alta velocidad. Estos dispositivos deberán utilizar técnicas de acceso y mitigación de interferencias con rendimiento al menos equivalente a las técnicas descritas en las normas armonizadas según la Directiva 1995/5/CE, o alternativamente no sobrepasar el 10% de ciclo de trabajo.

    Se excluyen las aplicaciones analógicas de vídeo.

    Con potencia igual o inferior a 25 mW se permiten aplicaciones de voz utilizando técnicas de voz, utilizando técnicas de acceso y mitigación de interferencias con rendimiento al menos equivalente a las técnicas descritas en las normal armonizadas según la Directiva 1995/5/CE, o alternativamente no sobrepasar el 0,1% de ciclo de trabajo.

- 869,700 – 870 MHz con 5 mW (p.r.a.) de potencia radiada aparente máxima. Se permiten aplicaciones de voz con técnicas de mitigación avanzadas, excluyéndose aplicaciones de audio y de vídeo.
- 869,700 – 870 MHz con 25 mW (p.r.a.) de potencia radiada aparente máxima. Estos dispositivos deberán utilizar técnicas de acceso y mitigación de interferencias con rendimiento al menos equivalente a las técnicas descritas en las normas armonizadas según la Directiva 1995/5/CE, o alternativamente no sobrepasar el 1% de ciclo de trabajo.

  Se excluyen las aplicaciones analógicas de vídeo.

- Alarmas:
  - 868,600 – 868,700 MHz con 10 mW (p.r.a.) de potencia radiada aparente máxima y 25 kHz de canalización, si bien pudiera utilizarse toda la banda como canal único de datos de alta velocidad. Ciclo de trabajo máximo del 1%.
  - 869,250 – 869,300 MHz con 10 mW (p.r.a.) de potencia radiada aparente máxima y 25 kHz de canalización. Ciclo de trabajo máximo del 0,1%.
  - 869,300 – 869,400 MHz con 10 mW (p.r.a.) de potencia radiada aparente máxima y 25 kHz de canalización. Ciclo de trabajo máximo del 1%.
  - 869,650 – 869,700 MHz con 25 mW (p.r.a.) de potencia radiada aparente máxima y 25 kHz de canalización. Ciclo de trabajo máximo del 10%.
- Alarmas de teleasistencia:
  - 869,200 – 869,700 MHz con 10 mW (p.r.a.) de potencia radiada aparente máxima y 25 kHz de canalización. Ciclo de trabajo máximo del 0,1%.

    La norma técnica de referencia para estos dispositivos es la EN 300 220.

    Los títulos habilitantes existentes en estas frecuencias, habrán de ajustarse a las características indicadas en esta nota a más tardar la caducidad de los mismos, pasando a uso común.

UNIVERSIDADE DA CORUÑA

ESCUELA POLITÉCNICA SUPERIOR

TRABAJO FIN DE MÁSTER

CURSO 2016/2017

DISEÑO Y DESARROLLO DEL SISTEMA DE COMUNICACIONES INALÁMBRICO DE UN ENJAMBRE DE UAV'S COLABORATIVOS

MÁSTER EN INGENIERÍA INDUSTRIAL

DOCUMENTO DEL PROYECTO

ANEXO II – LORAWAN SPECIFICATION

LORAWAN SPECIFICATION

LoRa Specification

NOTICE OF USE AND DISCLOSURE

LoRa Alliance, Inc.
2400 Camino Ramon, Suite 375
San Ramon, CA 94583

LoRaWAN™ Specification

Authors:
N. Sornin (Semtech), M. Luis (Semtech), T. Eirich (IBM), T. Kramp (IBM),
O.Hersent (Actility)

Version: V1.0
Date: 2015 January
Status: Released

Important note: This is a candidate specification for the LoRa™ Alliance protocol named LoRaWAN™.

Contents

Tables

Figures

1 Introduction

This document describes the LoRaWAN™ network protocol which is optimized for battery-powered end-devices that may be either mobile or mounted at a fixed location.

LoRaWAN networks typically are laid out in a star-of-stars topology in which **gateways**[1] relay messages between **end-devices**[2] and a central **network server** at the backend. Gateways are connected to the network server via standard IP connections while end-devices use single-hop LoRa™ or FSK communication to one or many gateways.[3] All communication is generally bi-directional, although uplink communication from an end-device to the network server is expected to be the predominant traffic.

Communication between end-devices and gateways is spread out on different **frequency channels** and **data rates**. The selection of the data rate is a trade-off between communication range and message duration, communications with different data rates do not interfere with each other. LoRa data rates range from 0.3 kbps to 50 kbps. To maximize both battery life of the end-devices and overall network capacity, the LoRa network infrastructure can manage the data rate and RF output for each end-device individually by means of an **adaptive data rate** (ADR) scheme.

End-devices may transmit on any channel available at any time, using any available data rate, as long as the following rules are respected:

- The end-device changes channel in a pseudo-random fashion for every transmission. The resulting frequency diversity makes the system more robust to interferences.
- The end-device respects the maximum transmit duty cycle relative to the sub-band used and local regulations.
- The end-device respects the maximum transmit duration (or dwell time) relative to the sub-band used and local regulations.

> **Note:** Maximum transmit duty-cycle and dwell time per sub-band are region specific and are defined in the Chapter 6.

1.1 LoRaWAN Classes

All LoRaWAN devices implement at least the Class A functionality as described in this document. In addition they may implement options named Class B, Class C as also described in this document or others to be defined. In all cases, they must remain compatible with Class A.

[1] Gateways are also known as **concentrators** or **base stations**.
[2] End-devices are also known as **motes**.
[3] Support for intermediate elements – repeaters – is not described in the document, however payload restrictions for encapsulation overhead are included in this specification. A repeater is defined as using LoRaWAN as its backhaul mechanism.

1.2 Conventions

MAC commands are written ***LinkCheckReq***, bits and bit fields are written **FRMPayload**, constants are written RECEIVE_DELAY1, variables are written *N*.

In this document,

- The octet order for all multi-octet fields is little endian and
- EUI are 64 bits integers and are transmitted as little endian.

2 Introduction on LoRaWAN options

LoRa™ is a wireless modulation for long-range low-power low-data-rate applications developed by Semtech. Devices implementing more than Class A are generally named "higher Class end-devices" in this document.

2.1 LoRaWAN Classes

A LoRa network distinguishes between a basic LoRaWAN (named Class A) and optional features (Class B, Class C …):

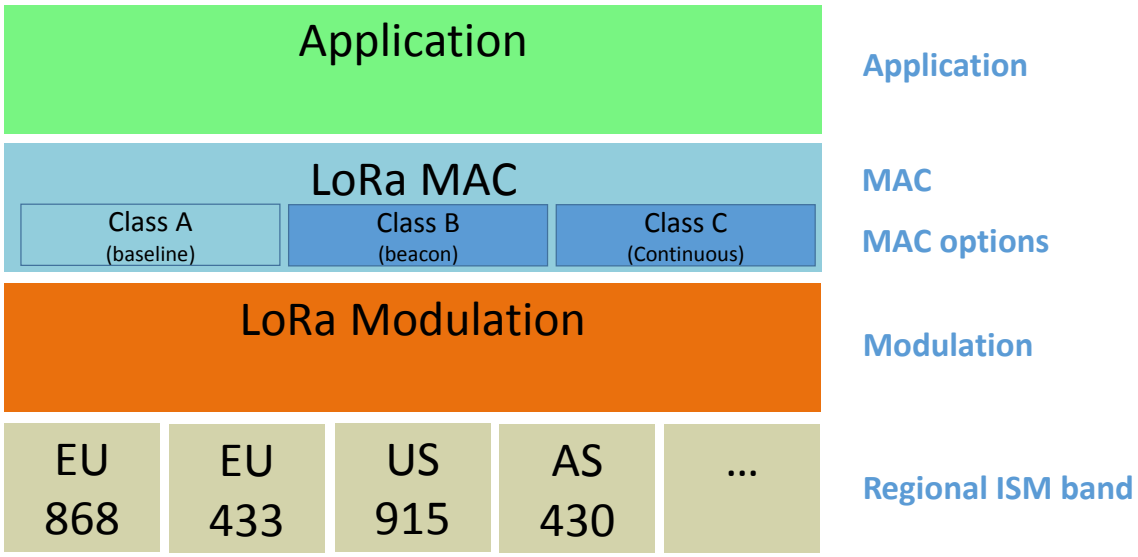


Figure 1: LoRaWAN Classes

- **Bi-directional end-devices (Class A):** End-devices of Class A allow for bi-directional communications whereby each end-device's uplink transmission is followed by two short downlink receive windows. The transmission slot scheduled by the end-device is based on its own communication needs with a small variation based on a random time basis (ALOHA-type of protocol). This Class A operation is the lowest power end-device system for applications that only require downlink communication from the server shortly after the end-device has sent an uplink transmission. Downlink communications from the server at any other time will have to wait until the next scheduled uplink.
- **Bi-directional end-devices with scheduled receive slots (Class B):** End-devices of Class B allow for more receive slots. In addition to the Class A random receive windows, Class B devices open extra receive windows at scheduled times. In order for the End-device to open it receive window at the scheduled time it receives a time synchronized Beacon from the gateway. This allows the server to know when the end-device is listening.
- **Bi-directional end-devices with maximal receive slots (Class C):** End-devices of Class C have nearly continuously open receive windows, only closed when transmitting. Class C end-device will use more power to operate than Class A or Class B but they offer the lowest latency for server to end-device communication.

2.2 Specification scope

This LoRaWAN specification describes the additional functions differentiating an end-device higher Class from one of Class A. A higher Class end-device shall also implement all the functionality described in the LoRaWAN Class A specification.

> **NOTE:** Physical message format, MAC message format, and other parts of this specification that are common to both end-devices of Class A and higher Classes are described only in the LoRaWAN Class A specification to avoid redundancy.

CLASS A – ALL END-DEVICES

All LoRaWAN end-devices must implement Class A features.

3 Physical Message Formats

The LoRa terminology distinguishes between uplink and downlink messages.

3.1 Uplink Messages

Uplink messages are sent by end-devices to the network server relayed by one or many gateways.

Uplink messages use the LoRa radio packet explicit mode in which the LoRa physical header (**PHDR**) plus a header CRC (**PHDR_CRC**) are included.[1] The integrity of the payload is protected by a CRC.

The **PHDR**, **PHDR_CRC** and payload **CRC** fields are inserted by the radio transceiver.

Uplink PHY:

Figure 2: Uplink PHY structure

3.2 Downlink Messages

Each **downlink message** is sent by the network server to only one end-device and is relayed by a single gateway.[2]

Downlink messages use the radio packet explicit mode in which the LoRa physical header (**PHDR**) and a header CRC (**PHDR_CRC**) are included.[3]

Downlink PHY:

Figure 3: Downlink PHY structure

3.3 Receive Windows

Following each uplink transmission the end-device opens two short receive windows. The receive window start times is a configured periods are the end of the transmission of the last uplink bit.

[1] See the LoRa radio transceiver datasheet for a description of LoRa radio packet implicit/explicit modes.
[2] This specification does not describe the transmission of multicast messages from a network server to many end-devices.
[3] No payload integrity check is done at this level to keep messages as short as possible with minimum impact on any duty-cycle limitations of the ISM bands used.

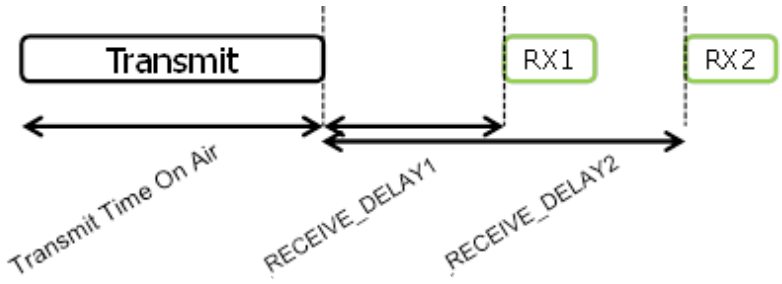


Figure 4: End-device receive slot timing.

3.3.1 First receive window channel, data rate, and start

The first receive window RX1 uses the same frequency channel as the uplink and a data rate that is a function of the data rate used for the uplink. RX1 opens RECEIVE_DELAY1[1] seconds (+/- 20 microseconds) after the end of the uplink modulation. The relationship between uplink and RX1 slot downlink data rate is region specific and detailed in the Section 7. By default the first receive window datarate is identical to the datarate of the last uplink.

3.3.2 Second receive window channel, data rate, and start

The second receive window RX2 uses a fixed configurable frequency and data rate and opens RECEIVE_DELAY2[1] seconds (+/- 20 microseconds) after the end of the uplink modulation. The frequency and data rate used can be modified through MAC commands (see Section 5).The default frequency and data rate to use are region specific and detailed in the Section 7 .

3.3.3 Receive window duration

The length of a receive window must be at least the time required by the end-device's radio transceiver to effectively detect a downlink preamble.

3.3.4 Receiver activity during the receive windows

If a preamble is detected during one of the receive windows, the radio receiver stays active until the downlink frame is demodulated. If a frame was detected and subsequently demodulated during the first receive window and the frame was intended for this end-device after address and MIC (message integrity code) checks, the end-device does not open the second receive window.

3.3.5 Network sending a message to an end-device

If the network intends to transmit a downlink to an end-device, it will always initiate the transmission precisely at the beginning of one of those two receive windows.

[1] RECEIVE_DELAY1 and RECEIVE_DELAY2 are described in Chapter 6.

3.3.6 Important notice on receive windows

An end-device shall not transmit another uplink message before it either has received a downlink message in the first or second receive window of the previous transmission, or the second receive window of the previous transmission is expired.

3.3.7 Receiving or transmitting other protocols

The node may listen or transmit other protocols or do any transactions between the LoRaWAN transmission and reception windows, as long as the end-device remains compatible with the local regulation and compliant with the LoRaWAN specification.

4 MAC Message Formats

All LoRa uplink and downlink messages carry a PHY payload (**Payload**) starting with a single-octet MAC header (**MHDR**), followed by a MAC payload (**MACPayload**)[1], and ending with a 4-octet message integrity code (**MIC**).

Radio PHY layer:

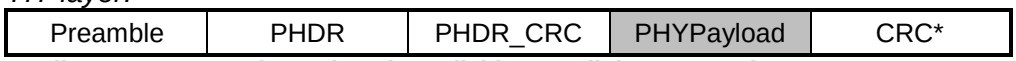


Figure 5: Radio PHY structure (CRC* is only available on uplink messages)

PHYPayload:

Figure 6: PHY payload structure

MACPayload:

Figure 7: MAC payload structure

FHDR:

Figure 8: Frame header structure

Figure 9: LoRa message format elements

4.1 MAC Layer (PHYPayload)

Size (bytes)	1	1..*M*	4
PHYPayload	MHDR	MACPayload	MIC

The maximum length (*M*) of the **MACPayload** field is region specific and is specified in Chapter 6.

4.2 MAC Header (MHDR field)

Bit#	7..5	4..2	1..0
MHDR bits	MType	RFU	Major

The MAC header specifies the message type (**MType**) and according to which major version (**Major**) of the frame format of the LoRaWAN layer specification the frame has been encoded.

[1] Maximum payload size is detailed in the Chapter 6.

4.2.1 Message type (MType bit field)

The LoRaWAN distinguishes between six different MAC message types: **join request**, **join accept**, **unconfirmed data up/down**, and **confirmed data up/down**.

MType	Description
000	Join Request
001	Join Accept
010	Unconfirmed Data Up
011	Unconfirmed Data Down
100	Confirmed Data Up
101	Confirmed Data Down
110	RFU
111	Proprietary

Table 1: MAC message types

4.2.1.1 Join-request and join-accept messages

The join-request and join-accept messages are used by the over-the-air activation procedure described in Chapter 6.2.

4.2.1.2 Data messages

Data messages are used to transfer both MAC commands and application data, which can be combined together in a single message. A **confirmed-data message** has to be acknowledged by the receiver, whereas an **unconfirmed-data message** does not require an acknowledgment.[1] **Proprietary messages** can be used to implement non-standard message formats that are not interoperable with standard messages but must only be used among devices that have a common understanding of the proprietary extensions.

Message integrity is ensured in different ways for different message types and is described per message type below.

4.2.2 Major version of data message (Major bit field)

Major bits	Description
00	LoRaWAN R1
01..11	RFU

Table 2: Major list

> **Note:** The Major version specifies the format of the messages exchanged in the join procedure (see Chapter 6.2) and the first four bytes of the MAC Payload as described in Chapter 4. For each major version, end-devices may implement different minor versions of the frame format. The minor version used by an end-device must be made known to the network server beforehand using out of band messages (e.g., as part of the device personalization information).

[1] A detailed timing diagram of the acknowledge mechanism is given in Section 18.

4.3 MAC Payload of Data Messages (MACPayload)

The MAC payload of the data messages, a so-called “data frame”, contains a frame header (**FHDR**) followed by an optional port field (**FPort**) and an optional frame payload field (**FRMPayload**).

4.3.1 Frame header (FHDR)

The **FHDR** contains the short device address of the end-device (**DevAddr**), a frame control octet (**FCtrl**), a 2-octets frame counter (**FCnt**), and up to 15 octets of frame options (**FOpts**) used to transport MAC commands.

Size (bytes)	4	1	2	0..15
FHDR	DevAddr	FCtrl	FCnt	FOpts

For downlink frames the FCtrl content of the frame header is:

Bit#	7	6	5	4	[3..0]
FCtrl bits	ADR	ADRACKReq	ACK	FPending	FOptsLen

For uplink frames the FCtrl content of the frame header is:

Bit#	7	6	5	4	[3..0]
FCtrl bits	ADR	ADRACKReq	ACK	RFU	FOptsLen

4.3.1.1 Adaptive data rate control in frame header (ADR, ADRACKReq in FCtrl)

LoRa network allows the end-devices to individually use any of the possible data rates. This feature is used by the LoRaWAN to adapt and optimize the data rate of static end-devices. This is referred to as Adaptive Data Rate (ADR) and when this is enabled the network will be optimized to use the fastest data rate possible.

Mobile end-devices should use their fixed default data rate as data rate management is not practical when the moving end-device causes fast changes in the radio environment.

If the **ADR** bit is set, the network will control the data rate of the end-device through the appropriate MAC commands. If the **ADR** bit is not set, the network will not attempt to control the data rate of the end-device regardless of the received signal quality. The **ADR** bit may be set and unset by the end-device or the Network on demand. However, whenever possible, the ADR scheme should be enabled to increase the battery life of the end-device and maximize the network capacity.

> **Note:** Even mobile end-devices are actually immobile most of the time. So depending on its state of mobility, an end-device can request the network to optimize its data rate using ADR.

If an end-device whose data rate is optimized by the network to use a data rate higher than its default data rate, it periodically needs to validate that the network still receives the uplink frames. Each time the uplink frame counter is incremented (for each new uplink, repeated transmissions do not increase the counter), the device increments an ADR_ACK_CNT counter. After ADR_ACK_LIMIT uplinks (ADR_ACK_CNT >= ADR_ACK_LIMIT) without any downlink response, it sets the ADR acknowledgment request bit (**ADRACKReq**). The network is required to respond with a downlink frame within the time set by the ADR_ACK_DELAY, any received downlink frame following an uplink frame resets the ADR_ACK_CNT counter. The downlink **ACK** bit does not need to be set as any response during the receive slot of the end-device indicates that the gateway has still received the uplinks from this device. If no reply is received within the next ADR_ACK_DELAY uplinks

(i.e., after a total of ADR_ACK_LIMIT + ADR_ACK_DELAY), the end-device may try to regain connectivity by switching to the next lower data rate that provides a longer radio range. The end-device will further lower its data rate step by step every time ADR_ACK_LIMIT is reached. The **ADRACKReq** shall not be set if the device uses its default data rate because in that case no action can be taken to improve the link range.

> **Note:** Not requesting an immediate response to an ADR acknowledgement request provides flexibility to the network to optimally schedule its downlinks.

> **Note:** In uplink transmissions the **ADRACKReq** bit is set if ADR_ACK_CNT >= ADR_ACK_LIMIT and the current data-rate is greater than the device defined minimum data rate, it is cleared in other conditions.

4.3.1.2 Message acknowledge bit and acknowledgement procedure (ACK in FCtrl)

When receiving a *confirmed data* message, the receiver shall respond with a data frame that has the acknowledgment bit (**ACK**) set. If the sender is an end-device, the network will send the acknowledgement using one of the receive windows opened by the end-device after the send operation. If the sender is a gateway, the end-device transmits an acknowledgment at its own discretion.

Acknowledgements are only sent in response to the latest message received and are never retransmitted.

> **Note:** To allow the end-devices to be as simple as possible and have as few states as possible it may transmit an explicit (possibly empty) acknowledgement data message immediately after the reception of a data message requiring a confirmation. Alternatively the end-device may defer the transmission of an acknowledgement to piggyback it with its next data message.

4.3.1.3 Retransmission procedure

The number of retransmissions (and their timing) for the same message where an acknowledgment is requested but not received is at the discretion of the end device and may be different for each end-device, it can also be set or adjusted from the network server.

> **Note:** Some example timing diagrams of the acknowledge mechanism are given in Chapter 18.

> **Note:** If an end-device has reached its maximum number of retransmissions without receiving an acknowledgment, it can try to re-gain connectivity by moving to a lower data rate with longer reach. It is up to the end-device to retransmit the message again or to forfeit that message and move on.

> **Note:** If the network server has reached its maximum number of retransmissions without receiving an acknowledgment, it will generally consider the end-device as unreachable until it receives further

messages from the end-device. It is up to the network server to retransmit the message once connectivity to the end-device in question is regained or to forfeit that message and move on.

Note: The recommended data rate back-off strategy during re-transmissions is described in Chapter 18.4

4.3.1.4 Frame pending bit (FPending in FCtrl, downlink only)

The frame pending bit (**FPending**) is only used in downlink communication, indicating that the gateway has more data pending to be sent and therefore asking the end-device to open another receive window as soon as possible by sending another uplink message.

The exact use of **FPending** bit is described in Chapter 18.3.

4.3.1.5 Frame counter (FCnt)

Each end-device has two frame counters to keep track of the number of data frames sent uplink to the network server (FCntUp), incremented by the end-device and received by the end-device downlink from the network server (FCntDown), which is incremented by the network server. The network server tracks the uplink frame counter and generates the downlink counter for each end-device. After a join accept, the frame counters on the end-device and the frame counters on the network server for that end-device are reset to 0. Subsequently FCntUp and FCntDown are incremented at the sender side by 1 for each data frame sent in the respective direction. At the receiver side, the corresponding counter is kept in sync with the value received provided the value received has incremented compared to the current counter value and is less than the value specified by MAX_FCNT_GAP[1] after considering counter rollovers. If this difference is greater than the value of MAX_FCNY_GAP then too many data frames have been lost then subsequent will be discarded.

The LoRaWAN allows the use of either 16-bits or 32-bits frame counters. The network side needs to be informed out-of-band about the width of the frame counter implemented in a given end-device. If a 16-bits frame counter is used, the **FCnt** field can be used directly as the counter value, possibly extended by leading zero octets if required. If a 32-bits frame counter is used, the **FCnt** field corresponds to the least-significant 16 bits of the 32-bits frame counter (i.e., FCntUp for data frames sent uplink and FCntDown for data frames sent downlink).

The end-device shall not reuse the same FCntUp value, except for retransmission, with the same application and network session keys.

Note: Since the **FCnt** field carries only the least-significant 16 bits of the 32-bits frame counter, the server must infer the 16 most-significant bits of the frame counter from the observation of the traffic.

4.3.1.6 Frame options (FOptsLen in FCtrl, FOpts)

The frame-options length field (**FOptsLen**) in **FCtrl** byte denotes the actual length of the frame options field (**FOpts**) included in the frame.

[1] Actual value for MAX_FCNT_GAP, RECEIVE_DELAY1 and RECEIVE_DELAY2 can be found at 7.1.7 for EU863-870 or 7.2.7 for US902-928.

FOpts transport MAC commands of a maximum length of 15 octets that are piggybacked onto data frames; see Chapter 4.4 for a list of valid MAC commands.

If **FOptsLen** is 0, the **FOpts** field is absent. If **FOptsLen** is different from 0, i.e. if MAC commands are present in the **FOpts** field, the port 0 cannot be used (**FPort** must be either not present or different from 0).

MAC commands cannot be simultaneously present in the payload field and the frame options field.

4.3.2 Port field (FPort)

If the frame payload field is not empty, the port field must be present. If present, an **FPort** value of 0 indicates that the **FRMPayload** contains MAC commands only; see Chapter 4.4 for a list of valid MAC commands. **FPort** values 1..223 (0x01..0xDF) are application-specific. **FPort** values 224..255 (0xE0..0xFF) are reserved for future standardized application extensions.

Size (bytes)	7..23	0..1	0..*N*
MACPayload	FHDR	FPort	FRMPayload

N is the number of octets of the application payload. The valid range for *N* is region specific and is defined in Section 7

N should be equal or smaller than:

$$N \leq M - 1 - (\text{length of } \textbf{FHDR} \text{ in octets})$$

where *M* is the maximum MAC payload length.

4.3.3 MAC Frame Payload Encryption (FRMPayload)

If a data frame carries a payload, **FRMPayload** must be encrypted before the message integrity code (**MIC**) is calculated.

The encryption scheme used is based on the generic algorithm described in IEEE 802.15.4/2006 Annex B [IEEE802154] using AES with a key length of 128 bits.

As a default, the encryption/decryption is done by the LoRaWAN layer for all FPort. The encryption/decryption may be done above the LoRaWAN layer for specific FPorts except 0, if this is more convenient for the application. The information concerning which FPort from which node is encrypted/decrypted outside of the LoRaWAN layer must be communicated to the server using an out-of-band channel (see Section 19).

4.3.3.1 Encryption in LoRaWAN

The key *K* used depends on the FPort of the data message:

FPort	K
0	NwkSKey
1..255	AppSKey

Table 3: FPort list

The fields encrypted are:

pld = **FRMPayload**

For each data message, the algorithm defines a sequence of Blocks A_i for $i = 1..k$ with k = ceil(len(*pld*) / 16):

Size (bytes)	1	4	1	4	4	1	1
A_i	0x01	4 x 0x00	Dir	DevAddr	FCntUp or FCntDown	0x00	*i*

The direction field (**Dir**) is 0 for uplink frames and 1 for downlink frames.

The blocks A_i are encrypted to get a sequence S of blocks S_i:

$$S_i = \text{aes128\_encrypt}(K, A_i) \text{ for } i = 1..k$$
$$S = S_1 \mid S_2 \mid .. \mid S_k$$

Encryption and decryption of the payload is done by truncating

$$(pld \mid \text{pad}_{16}) \text{ xor } S$$

to the first len(*pld*) octets.

4.3.3.2 Encryption above the LoRaWAN layer

If the layers above LoRaWAN provide pre-encrypted FRMPayload to LoRaWAN on selected ports (but not on FPort 0 which is reserved for MAC commands), LoRaWAN transfers FRMPayload from MACPayload to the application and FRMPayload from the application to MACPayload without any modification of FRMPayload.

4.4 Message Integrity Code (MIC)

The message integrity code (**MIC**) is calculated over all the fields in the message.

$$msg = \textbf{MHDR} \mid \textbf{FHDR} \mid \textbf{FPort} \mid \textbf{FRMPayload}$$

whereby len(*msg*) denotes the length of the message in octets.

The **MIC** is calculated as follows [RFC4493]:

$$cmac = \text{aes128\_cmac}(\textbf{NwkSKey}, B_0 \mid msg)$$
$$\textbf{MIC} = cmac[0..3]$$

whereby the block B_0 is defined as follows:

Size (bytes)	1	4	1	4	4	1	1
B_0	0x49	4 x 0x00	Dir	DevAddr	FCntUp or FCntDown	0x00	len(*msg*)

The direction field (**Dir**) is 0 for uplink frames and 1 for downlink frames.

5 MAC Commands

For network administration, a set of MAC commands may be exchanged exclusively between the network server and the MAC layer on an end-device. MAC layer commands are never visible to the application or the application server or the application running on the end-device.

A single data frame can contain any sequence of MAC commands, either piggybacked in the **FOpts** field or, when sent as a separate data frame, in the **FRMPayload** field with the **FPort** field being set to 0. Piggybacked MAC commands are always sent without encryption and must not exceed 15 octets. MAC commands sent as **FRMPayload** are always encrypted and must not exceed the maximum **FRMPayload** length.

> **Note:** MAC commands whose content shall not be disclosed to an eavesdropper must be sent in the **FRMPayload** of a separate data message.

A MAC command consists of a command identifier (**CID**) of 1 octet followed by a possibly empty command-specific sequence of octets.

CID	Command	Transmitted by		Short Description
		End-device	Gateway	
0x02	***LinkCheckReq***	x		Used by an end-device to validate its connectivity to a network.
0x02	***LinkCheckAns***		x	Answer to LinkCheckReq command. Contains the received signal power estimation indicating to the end-device the quality of reception (link margin).
0x03	***LinkADRReq***		x	Requests the end-device to change data rate, transmit power, repetition rate or channel.
0x03	***LinkADRAns***	x		Acknowledges the LinkRateReq.
0x04	***DutyCycleReq***		x	Sets the maximum aggregated transmit duty-cycle of a device
0x04	***DutyCycleAns***	x		Acknowledges a DutyCycleReq command
0x05	***RXParamSetupReq***		x	Sets the reception slots parameters
0x05	***RXParamSetupAns***	x		Acknowledges a RXSetupReq command
0x06	***DevStatusReq***		x	Requests the status of the end-device
0x06	***DevStatusAns***	x		Returns the status of the end-device, namely its battery level and its demodulation margin
0x07	***NewChannelReq***		x	Creates or modifies the definition of a radio channel
0x07	***NewChannelAns***	x		Acknowledges a NewChannelReq command
0x08	***RXTimingSetupReq***		x	Sets the timing of the of the reception slots
0x08	***RXTimingSetupAns***	x		Acknowledge RXTimingSetupReq command
0x80 to 0xFF	Proprietary	x	x	Reserved for proprietary network command extensions

Table 4: MAC commands

> **Note:** The length of a MAC command is not explicitly given and must be implicitly known by the MAC implementation. Therefore unknown MAC commands cannot be skipped and the first unknown MAC command terminates the processing of the MAC command sequence. It is therefore advisable to order MAC commands according to the

version of the LoRaWAN specification which has introduced a MAC command for the first time. This way all MAC commands up to the version of the LoRaWAN specification implemented can be processed even in the presence of MAC commands specified only in a version of the LoRaWAN specification newer than that implemented.

Note: Any values adjusted by the network server (e.g., RX2, new or adjusted channels definitions) remain only valid until the next join of the end-device. Therefore after each successful join procedure the end-device uses the default parameters again and it is up to the network server to re-adjust the values again as needed.

5.1 Link Check commands (*LinkCheckReq, LinkCheckAns*)

With the ***LinkCheckReq*** command, an end-device may validate its connectivity with the network. The command has no payload.

When a ***LinkCheckReq*** is received by the network server via one or multiple gateways, it responds with a ***LinkCheckAns*** command.

Size (bytes)	1	1
LinkCheckAns Payload	Margin	GwCnt

The demodulation margin (**Margin**) is an 8-bit unsigned integer in the range of 0..254 indicating the link margin in dB of the last successfully received ***LinkCheckReq*** command. A value of “0” means that the frame was received at the demodulation floor (0 dB or no margin) while a value of “20”, for example, means that the frame reached the gateway 20 dB above the demodulation floor. Value “255” is reserved.

The gateway count (**GwCnt**) is the number of gateways that successfully received the last ***LinkCheckReq*** command.

5.2 Link ADR commands (*LinkADRReq, LinkADRAns*)

With the ***LinkADRReq*** command, the network server requests an end-device to perform a rate adaptation.

Size (bytes)	1	2	1
LinkADRReq Payload	DataRate_TXPower	ChMask	Redundancy

Bits	[7:4]	[3:0]
DataRate_TXPower	DataRate	TXPower

The requested date rate (**DataRate**) and TX output power (**TXPower**) are region-specific and are encoded as indicated in Chapter 7.

The channel mask (**ChMask**) encodes the channels usable for uplink access as follows with bit 0 corresponding to the LSB:

Bit#	Usable channels
0	Channel 1
1	Channel 2
..	..
15	Channel 16

Table 5: Channel state table

A bit in the **ChMask** field set to 1 means that the corresponding channel can be used for uplink transmissions if this channel allows the data rate currently used by the end-device. A bit set to 0 means the corresponding channels should be avoided.

Bits	7	[6:4]	[3:0]
Redundancy bits	RFU	ChMaskCntl	NbRep

In the Redundancy bits the number of repetitions (**NbRep**) field is the number of repetition for each uplink message. This applies only to “unconfirmed” uplink frames. The default value is 1. The valid range is [1:15]. If **NbRep**==0 is received the end-device should use the default value. This field can be used by the network manager to control the redundancy of the node uplinks to obtain a given Quality of Service. The end-device performs frequency hopping as usual between repeated transmissions, it does wait after each repetition until the receive windows have expired.

The channel mask control (**ChMaskCntl**) field controls the interpretation of the previously defined **ChMask** bit mask. This field will only be non-zero values in networks where more than 16 channels are implemented. It controls the block of 16 channels to which the **ChMask** applies. It can also be used to globally turn on or off all channels using specific modulation. This field usage is region specific and is defined in Chapter 7.

The channel frequencies are region-specific and they are defined in Chapter 6. An end-device answers to a ***LinkADRReq*** with a ***LinkADRAns*** command.

Size (bytes)	1
LinkADRAns Payload	Status

Bits	[7:3]	2	1	0
Status bits	RFU	Power ACK	Data rate ACK	Channel mask ACK

The ***LinkADRAns* Status** bits have the following meaning:

	Bit = 0	**Bit = 1**
Channel mask ACK	The channel mask sent enables a yet undefined channel. The command was discarded and the end-device state was not changed.	The channel mask sent was successfully interpreted. All currently defined channel states were set according to the mask.
Data rate ACK	The data rate requested is unknown to the end-device or is not possible given the channel mask provided (not supported by any of the enabled channels). The command was discarded and the end-device state was not changed.	The data rate was successfully set.
Power ACK	The requested power level is not implemented in the device. The command was discarded and the end-device state was not changed.	The power level was successfully set.

Table 6: LinkADRAns status bits signification

If any of those three bits equals 0, the command did not succeed and the node has kept the previous state.

5.3 End-Device Transmit Duty Cycle (*DutyCycleReq*, *DutyCycleAns*)

The ***DutyCycleReq*** command is used by the network coordinator to limit the maximum aggregated transmit duty cycle of an end-device. The aggregated transmit duty cycle corresponds to the transmit duty cycle over all sub-bands.

Size (bytes)	1
DutyCycleReq Payload	MaxDCycle

The maximum end-device transmit duty cycle allowed is:

$$aggregated\ duty\ cycle = \frac{1}{2^{\mathrm{MaxDCycle}}}$$

The valid range for **MaxDutyCycle** is [0 : 15]. A value of 0 corresponds to “no duty cycle limitation” except the one set by the regional regulation.

A value of 255 means that the end-device shall become silent immediately. It is equivalent to remotely switching off the end-device.

An end-device answers to a ***DutyCycleReq*** with a ***DutyCycleAns*** command. The ***DutyCycleAns*** MAC reply does not contain any payload.

5.4 Receive Windows Parameters (*RXParamSetupReq*, *RXParamSetupAns*)

The ***RXParamSetupReq*** command allows a change to the frequency and the data rate set for the second receive window (RX2) following each uplink. The command also allows to program an offset between the uplink and the RX1 slot downlink data rates.

Size (bytes)	1	3
RX2SetupReq Payload	DLsettings	Frequency

Bits	7	6:4	3:0
DLsettings	RFU	RX1DRoffset	RX2DataRate

The RX1DRoffset field sets the offset between the uplink data rate and the downlink data rate used to communicate with the end-device on the first reception slot (RX1). As a default this offset is 0.. The offset is used to take into account maximum power density constraints for base stations in some regions and to balance the uplink and downlink radio link margins.

The data rate (RX2**DataRate**) field defines the data rate of a downlink using the second receive window following the same convention as the ***LinkADRReq*** command (0 means DR0/125kHz for example). The frequency (**Frequency**) field corresponds to the frequency of the channel used for the second receive window, whereby the frequency is coded following the convention defined in the ***NewChannelReq*** command.

The ***RXParamSetupAns*** command is used by the end-device to acknowledge the reception of ***RXParamSetupReq*** command. The payload contains a single status byte.

Size (bytes)	1
RX2SetupAns Payload	Status

The status (**Status**) bits have the following meaning.

Bits	7:3	2	1	0
Status bits	RFU	RX1DRoffset ACK	RX2 Data rate ACK	Channel ACK

	Bit = 0	**Bit = 1**
Channel ACK	The frequency requested is not usable by the end-device.	RX2 slot channel was successfully set
RX2 Data rate ACK	The data rate requested is unknown to the end-device.	RX2 slot data rate was successfully set
RX1DRoffset ACK	the uplink/downlink data rate offset for RX1 slot is not in the allowed range	RX1DRoffset was successfully set

Table 7: RX2SetupAns status bits signification

If either of the 3 bits is equal to 0, the command did not succeed and the previous parameters are kept.

5.5 End-Device Status (*DevStatusReq*, *DevStatusAns*)

With the ***DevStatusReq*** command a network server may request status information from an end-device. The command has no payload. If a ***DevStatusReq*** is received by an end-device, it responds with a ***DevStatusAns*** command.

Size (bytes)	1	1
DevStatusAns Payload	Battery	Margin

The battery level (**Battery**) reported is encoded as follows:

Battery	**Description**
0	The end-device is connected to an external power source.
1..254	The battery level, 1 being at minimum and 254 being at maximum
255	The end-device was not able to measure the battery level.

Table 8: Battery level decoding

The margin (**Margin**) is the demodulation signal-to-noise ratio in dB rounded to the nearest integer value for the last successfully received ***DevStatusReq*** command. It is a signed integer of 6 bits with a minimum value of -32 and a maximum value of 31.

Bits	7:6	5:0
Status	RFU	Margin

5.6 Creation / Modification of a Channel (*NewChannelReq*, *NewChannelAns*)

The ***NewChannelReq*** command can be used to either modify the parameters of an existing channel or to create a new one. The command sets the center frequency of the new channel and the range of data rates usable on this channel:

Size (bytes)	1	3	1
NewChannelReq Payload	ChIndex	Freq	DrRange

The channel index (**ChIndex**) is the index of the channel being created or modified. Depending on the region and frequency band used, the LoRaWAN specification imposes default channels which must be common to all devices and cannot be modified by the ***NewChannelReq*** command (cf. Chapter 6). If the number of default channels is *N*, the default channels go from 0 to *N*-1, and the acceptable range for **ChIndex** is *N* to 15. A device must be able to handle at least 16 different channel definitions. In certain region the device may have to store more than 16 channel definitions.

The frequency (**Freq**) field is a 24 bits unsigned integer. The actual channel frequency in Hz is 100 x **Freq** whereby values representing frequencies below 100 MHz are reserved for future use. This allows setting the frequency of a channel anywhere between 100 MHz to 1.67 GHz in 100 Hz steps. A **Freq** value of 0 disables the channel. The end-device has to check that the frequency is actually allowed by its radio hardware and return an error otherwise.

The data-rate range (**DrRange**) field specifies the data-rate range allowed for this channel. The field is split in two 4-bit indexes:

Bits	7:4	3:0
DrRange	MaxDR	MinDR

Following the convention defined in Section 5.2 the minimum data rate (**MinDR**) subfield designate the lowest data rate allowed on this channel. For example 0 designates DR0 / 125 kHz. Similarly, the maximum data rate (**MaxDR**) designates the highest data rate. For example, DrRange = 0x77 means that only 50 kbps GFSK is allowed on a channel and DrRange = 0x50 means that DR0 / 125 kHz to DR5 / 125 kHz are supported.

The newly defined channel is enabled and can immediately be used for communication.

The end-device acknowledges the reception of a ***NewChannelReq*** by sending back a ***NewChannelAns*** command. The payload of this message contains the following information:

Size (bytes)	1
NewChannelAns Payload	Status

The status (**Status**) bits have the following meaning:

Bits	7:2	1	0
Status	RFU	Data rate range ok	Channel frequency ok

	Bit = 0	**Bit = 1**
Data rate range ok	The designated data rate range exceeds the ones currently defined for this end-device	The data rate range is compatible with the possibilities of the end-device
Channel frequency ok	The device cannot use this frequency	The device is able to use this frequency.

Table 9: NewChannelAns status bits signification

If either of those 2 bits equals 0, the command did not succeed and the new channel has not been created.

5.7 Setting delay between TX and RX (*RXTimingSetupReq*, *RXTimingSetupAns*)

The ***RXTimingSetupReq*** command allows configuring the delay between the end of the TX uplink and the opening of the first reception slot. The second reception slot opens one second after the first reception slot.

Size (bytes)	1
RXTimingSetupReq Payload	Settings

The delay (**Delay**) field specifies the delay. The field is split in two 4-bit indexes:

Bits	7:4	3:0
Settings	RFU	Del

The delay is expressed in seconds. **Del** 0 is mapped on 1 s.

Del	**Delay [s]**
0	1
1	1
2	2
3	3
..	..

15	15

Table 10: Del mapping table

An end device answers ***RXTimingSetupReq*** with ***RXTimingSetupAns*** with no payload.

6 End-Device Activation

To participate in a LoRaWAN network, each end-device has to be personalized and activated.

Activation of an end-device can be achieved in two ways, either via **Over-The-Air Activation** (OTAA) when an end-device is deployed or reset, or via **Activation By Personalization** (ABP) in which the two steps of end-device personalization and activation are done as one step.

6.1 Data Stored in the End-device after Activation

After activation, the following information is stored in the end-device: a device address (**DevAddr**), an application identifier (**AppEUI**), a network session key (**NwkSKey**), and an application session key (**AppSKey**).

6.1.1 End-device address (DevAddr)

The **DevAddr** consists of 32 bits identifies the end-device within the current network. Its format is as follows:

Bit#	[31..25]	[24..0]
DevAddr bits	NwkID	NwkAddr

The most significant 7 bits are used as network identifier (**NwkID**) to separate addresses of territorially overlapping networks of different network operators and to remedy roaming issues. The least significant 25 bits, the network address (**NwkAddr**) of the end-device, can be arbitrarily assigned by the network manager.

6.1.2 Application identifier (AppEUI)

The **AppEUI** is a global application ID in IEEE EUI64 address space that uniquely identifies the application provider (i.e., owner) of the end-device.

The **AppEUI** is stored in the end-device before the activation procedure is executed.

6.1.3 Network session key (NwkSKey)

The **NwkSKey** is a **network session key** specific for the end-device. It is used by both the network server and the end-device to calculate and verify the **MIC** (message integrity code) of all data messages to ensure data integrity. It is further used to encrypt and decrypt the payload field of a MAC-only data messages.

6.1.4 Application session key (AppSKey)

The **AppSKey** is an **application session key** specific for the end-device. It is used by both the network server and the end-device to encrypt and decrypt the payload field of application-specific data messages. It is also used to calculate and verify an application-level **MIC** that may be included in the payload of application-specific data messages.

6.2 Over-the-Air Activation

For over-the-air activation, end-devices must follow a join procedure prior to participating in data exchanges with the network server. An end-device has to go through a new join procedure every time it has lost the session context information.

The join procedure requires the end-device to be personalized with the following information before its starts the join procedure: a globally unique end-device identifier (**DevEUI**), the application identifier (**AppEUI**), and an AES-128 key (**AppKey**).

The **AppEUI** is described above in 6.1.2.

> **Note:** For over-the-air-activation, end-devices are not personalized with any kind of network key. Instead, whenever an end-device joins a network, a network session key specific for that end-device is derived to encrypt and verify transmissions at the network level. This way, roaming of end-devices between networks of different providers is facilitated. Using both a network session key and an application session key further allows federated network servers in which application data cannot be read or tampered with by the network provider.

6.2.1 End-device identifier (DevEUI)

The **DevEUI** is a global end-device ID in IEEE EUI64 address space that uniquely identifies the end-device.

6.2.2 Application key (AppKey)

The **AppKey** is an AES-128 application key specific for the end-device that is assigned by the application owner to the end-device and most likely derived from an application-specific root key exclusively known to and under the control of the application provider.[1] Whenever an end-device joins a network via over-the-air activation, the AppKey is used to derive the session keys NwkSKey and AppSKey specific for that end-device to encrypt and verify network communication and application data.

6.2.3 Join procedure

From an end-device's point of view, the join procedure consists of two MAC messages exchanged with the server, namely a **join request** and a **join accept**.

6.2.4 Join-request message

The join procedure is always initiated from the end-device by sending a join-request message.

Size (bytes)	8	8	2
Join Request	AppEUI	DevEUI	DevNonce

The join-request message contains the **AppEUI** and **DevEUI** of the end-device followed by a **nonce** of 2 octets (**DevNonce**).

DevNonce is a random value.[2] For each end-device, the network server keeps track of a certain number of **DevNonce** values used by the end-device in the past, and ignores join requests with any of these **DevNonce** values from that end-device.

1. Since all end-devices end up with unrelated application keys specific for each end-device, extracting the AppKey from an end-device only compromises this one end-device.
[2] The DevNonce can be extracted by issuing a sequence of RSSI measurements under the assumption that the quality of randomness fulfills the criteria of true randomness

> **Note:** This mechanism prevents replay attacks by sending previously recorded join-request messages with the intention of disconnecting the respective end-device from the network.

The message integrity code (**MIC**) value (see Chapter 4 for MAC message description) for a join-request message is calculated as follows:[1]

cmac = aes128_cmac(AppKey, MHDR | AppEUI | DevEUI | DevNonce)
MIC = *cmac*[0..3]

The join-request message is not encrypted.

6.2.5 Join-accept message

The network server will respond to the **join-request** message with a **join-accept** message if the end-device is permitted to join a network. The join-accept message is sent like a normal downlink but uses delays JOIN_ACCEPT_DELAY1 or JOIN_ACCEPT_DELAY2 (instead of RECEIVE_DELAY1 and RECEIVE_DELAY2, respectively). The channel frequency and data rate used for these two receive windows are identical to the one used for the RX1 and RX2 receive windows described in the “receive windows” section of the “Physical Layer” chapter

No response is given to the end-device if the join request is not accepted.

The join-accept message contains an application nonce (**AppNonce**) of 3 octets, a network identifier (**NetID**), an end-device address (**DevAddr**), a delay between TX and RX (**RxDelay**) and an optional list of channel frequencies (**CFList**) for the network the end-device is joining. The CFList option is region specific and is defined in Section 7.

Size (bytes)	3	3	4	1	1	(16) Optional
Join Accept	AppNonce	NetID	DevAddr	DLSettings	RxDelay	CFList

The **AppNonce** is a random value or some form of unique ID provided by the network server and used by the end-device to derive the two session keys **NwkSKey** and **AppSKey** as follows:[3]
NwkSKey = aes128_encrypt(AppKey, 0x01 | AppNonce | NetID | DevNonce | pad_{16})
AppSKey = aes128_encrypt(AppKey, 0x02 | AppNonce | NetID | DevNonce | pad_{16})

The MIC value for a join-accept message is calculated as follows:[4]
cmac = aes128_cmac(AppKey,
MHDR | AppNonce | NetID | DevAddr | RFU | RxDelay | CFList)
MIC = *cmac*[0..3]

The join-accept message itself is encrypted with the **AppKey** as follows:
aes128_decrypt(AppKey, AppNonce | NetID | DevAddr | RFU | RxDelay | CFList | MIC)

> **Note:** The network server uses an AES decrypt operation in ECB mode to encrypt the join-accept message so that the end-device can use an AES encrypt operation to decrypt the message. This way an end-device only has to implement AES encrypt but not AES decrypt.

[1] [RFC4493]
[3] The pad_{16} function appends zero octets so that the length of the data is a multiple of 16.
[4] [RFC4493]

> **Note:** Establishing these two session keys allows for a federated network server infrastructure in which network operators are not able to eavesdrop on application data. In such a setting, the application provider must support the network operator in the process of an end-device actually joining the network and establishing the NwkSKey for the end-device. At the same time the application provider commits to the network operator that it will take the charges for any traffic incurred by the end-device and retains full control over the AppSKey used for protecting its application data.

The format of the **NetID** is as follows: The seven LSB of the **NetID** are called **NwkID** and match the seven MSB of the short address of an end-device as described before. Neighboring or overlapping networks must have different **NwkID**s. The remaining 17 MSB can be freely chosen by the network operator.

The DLsettings field contains the downlink configuration:

Bits	7	6:4	3:0
DLsettings	RFU	RX1DRoffset	RX2 Data rate

The RX1DRoffset field sets the offset between the uplink data rate and the downlink data rate used to communicate with the end-device on the first reception slot (RX1). By default this offset is 0. The downlink data rate is always lower or equal than the uplink data rate. The offset is used to take into account maximum power density constraints for base stations in some regions and to balance the uplink and downlink radio link margins.

The actual relationship between the uplink and downlink data rate is region specific and detailed in the “Physical Layer” section

The delay **RxDelay** follows the same convention as the **Delay** field in the ***RXTimingSetupReq*** command.

6.3 Activation by Personalization

Under certain circumstances, end-devices can be activated by personalization. Activation by personalization directly ties an end-device to a specific network by-passing the **join request - join accept** procedure.

Activating an end-device by personalization means that the **DevAddr** and the two session keys **NwkSKey** and **AppSKey** are directly stored into the end-device instead of the **DevEUI, AppEUI** and the **AppKey**. The end-device is equipped with the required information for participating in a specific LoRa network when started.

Each device should have a unique set of NwkSKey and AppSKey. Compromising the keys of one device shouldn’t compromise the security of the communications of other devices. The process to build those keys should be such that the keys cannot be derived in any way from publicly available information (like the node address for example).

7 Physical Layer

7.1 EU 863-870MHz ISM Band

7.1.1 EU863-870 Preamble Format

The following synchronization words should be used:

Modulation	Sync word	Preamble length
LORA	0x34	8 symbols
GFSK	0xC194C1	5 bytes

Table 11: EU863-870 synch words

7.1.2 EU863-870 ISM Band channel frequencies

In Europe, radio spectrum allocation is the ISM band is defined by ETSI [EN300.220].

The network channels can be freely attributed by the network operator. However the three following default channels must be implemented in every EU868MHz end-device. Those channels are the minimum set that all network gateways should always be listening on.

Modulation	Bandwidth [kHz]	Channel Frequency [MHz]	FSK Bitrate or LoRa DR / Bitrate	Nb Channels	Duty cycle
LoRa	125	868.10 868.30 868.50	DR0 to DR5 / 0.3-5 kbps	3	<1%

Table 12: EU863-870 default channels

In order to access the physical medium the ETSI regulations impose some restrictions such maximum time the transmitter can be on or the maximum time a transmitter can transmit per hour. The ETSI regulations allow the choice of using either a duty-cycle limitation or a so-called **Listen Before Talk Adaptive Frequency Agility** (LBT AFA) transmissions management. The current LoRaWAN specification exclusively uses duty-cycled limited transmissions to comply with the ETSI regulations.

The LoRaWAN enforces a per sub-band duty-cycle limitation. Each time a frame is transmitted in a given sub-band, the time of emission and the on-air duration of the frame are recorded for this sub-band. The same sub-band cannot be used again during the next *Toff* seconds where:

$$\mathrm{Toff}_{\mathrm{subband}} = \frac{\mathrm{TimeOnAir}}{\mathrm{DutyCycle}_{\mathrm{subband}}} - \mathrm{TimeOnAir}$$

During the unavailable time of a given sub-band, the device may still be able to transmit on another sub-band. If all sub-bands are unavailable, the device has to wait before any further transmission. The device adapts its channel hoping sequence according to the sub-band availability.

Example: A device just transmitted a 0.5 s long frame on one default channel. This channel is in a sub-band allowing 1% duty-cycle. Therefore this whole sub-band (868 – 868.6) will be unavailable for 49.5 s.

EU868MHz ISM band end-devices should use the following default parameters

- Default radiated transmit output power: 14 dBm

EU868Mhz end-devices should be capable of operating in the 863 to 870 MHz frequency band and should feature a channel data structure to store the parameters of at least 16 channels. A channel data structure corresponds to a frequency and a set of data rates usable on this frequency.

The first three channels correspond to 868.1, 868.3, and 868.5 MHz / DR0 to DR5 and must be implemented in every end-device. Those default channels cannot be modified through the ***NewChannelReq*** command and guarantee a minimal common channel set between end-devices and network gateways.

The following table gives the list of frequencies that should be used by end-devices to broadcast the JoinReq message. The JoinReq message transmit duty-cycle should never exceed 0.1%

Modulation	Bandwidth [kHz]	Channel Frequency [MHz]	FSK Bitrate or LoRa DR / Bitrate	Nb Channels	Duty cycle
LoRa	125	864.10 864.30 864.50 868.10 868.30 868.50	DR0 – DR5 / 0.3-5 kbps	6	<0.1%

Table 13: EU863-870 JoinReq Channel List

7.1.3 EU863-870 Data Rate and End-point Output Power encoding

The following encoding is used for Data Rate (DR) and End-point Output Power (TXPower) in the EU863-870 band:

DataRate	Configuration	Indicative physical bit rate [bit/s]
0	LoRa: SF12 / 125 kHz	250
1	LoRa: SF11 / 125 kHz	440
2	LoRa: SF10 / 125 kHz	980
3	LoRa: SF9 / 125 kHz	1760
4	LoRa: SF8 / 125 kHz	3125
5	LoRa: SF7 / 125 kHz	5470
6	LoRa: SF7 / 250 kHz	11000
7	FSK: 50 kbps	50000
8..15	RFU	

TXPower	Configuration
0	20 dBm (if supported)
1	14 dBm
2	11 dBm
3	8 dBm
4	5 dBm
5	2 dBm
6..15	RFU

Table 14: Data rate and TX power table

7.1.4 EU863-870 JoinAccept CFList

The EU 863-870 ISM band LoRaWAN implements an optional **channel frequency list** (CFlist) of 16 octets in the JoinAccept message.

In this case the CFList is a list of five channel frequencies for the channels four to eight whereby each frequency is encoded as a 24 bits unsigned integer (three octets). All these channels are usable for DR0 to DR5 125kHz LoRa modulation. The list of frequencies is followed by a single RFU octet for a total of 16 octets.

Size (bytes)	3	3	3	3	3	1
CFList	Freq Ch4	Freq Ch5	Freq Ch6	Freq CN7	Freq Ch8	RFU

The actual channel frequency in Hz is 100 x frequency whereby values representing frequencies below 100 Mhz are reserved for future use. This allows setting the frequency of a channel anywhere between 100 MHz to 1.67 GHz in 100 Hz steps. Unused channels have a frequency value of 0. The **CFList** is optional and its presence can be detected by the length of the join-accept message. If present, the **CFList** replaces all the previous channels stored in the end-device apart from the three default channels as defined in Chapter 6. The newly defined channels are immediately enabled and usable by the end-device for communication.

7.1.5 EU863-870 LinkAdrReq command

The EU863-870 LoRaWAN only supports a maximum of 16 channels. When **ChMaskCntl** field is 0 the ChMask field individually enables/disables each of the 16 channels.

ChMaskCntl	**ChMask applies to**
0	Channels 1 to 16
1	RFU
..	..
4	RFU
5	RFU
6	All channels ON The device should enable all currently defined channels independently of the ChMask field value.
7	RFU

Table 15: ChMaskCntl value table

If the ChMask field value is one of values meaning RFU, the end-device should reject the command and unset the "**Channel mask ACK**" bit in its response.

7.1.6 EU863-870 Maximum payload size

The maximum **MACPayload** size length (*M)* is given by the following table. It is derived from limitation of the PHY layer depending on the effective modulation rate used taking into account a possible repeater encapsulation layer. The maximum application payload length in the absence of the optional **FOpt** control field (*N*) is also given for information only. The value of N might be smaller if the **FOpt** field is not empty:

DataRate	***M***	***N***
0	59	51
1	59	51
2	59	51
3	123	115
4	230	222

5	230	222
6	230	222
7	230	222
8:15	Not defined	

Table 16: EU863-870 maximum payload size

If the end-device will never operate with a repeater then the maximum application payload length in the absence of the optional **FOpt** control field should be:

DataRate	M	N
0	59	51
1	59	51
2	59	51
3	123	115
4	250	242
5	250	242
6	250	242
7	250	242
8:15	Not defined	

Table 17 : EU863-870 maximum payload size (not repeater compatible)

7.1.7 EU863-870 Receive windows

The RX1 receive window uses the same channel than the preceding uplink. The data rate is a function of the uplink data rate and the RX1DROffset as given by the following table. The allowed values for RX1DROffset are in the [0:5] range. Values in the [6:7] range are reserved for future use.

RX1DROffset **Upstream data rate**	**0**	**1**	**2**	**3**	**4**	**5**
	Downstream data rate in RX1 slot					
DR0	DR0	DR0	DR0	DR0	DR0	DR0
DR1	DR1	DR0	DR0	DR0	DR0	DR0
DR2	DR2	DR1	DR0	DR0	DR0	DR0
DR3	DR3	DR2	DR1	DR0	DR0	DR0
DR4	DR4	DR3	DR2	DR1	DR0	DR0
DR5	DR5	DR4	DR3	DR2	DR1	DR0
DR6	DR6	DR5	DR4	DR3	DR2	DR1
DR7	DR7	DR6	DR5	DR4	DR3	DR2

The RX2 receive window uses a fixed frequency and data rate. The default parameters are 869.525 MHz / DR0 (SF12, 125 kHz)

7.1.8 EU863-870 Default Settings

The following parameters are recommended values for the EU863-870Mhz band.

RECEIVE_DELAY1	1 s
RECEIVE_DELAY2	2 s (must be RECEIVE_DELAY1 + 1s)
JOIN_ACCEPT_DELAY1	5 s

JOIN_ACCEPT_DELAY2	6 s
MAX_FCNT_GAP	16384
ADR_ACK_LIMIT	64
ADR_ACK_DELAY	32
ACK_TIMEOUT	2 +/- 1 s (random delay between 1 and 3 seconds)

If the actual parameter values implemented in the end-device are different from those default values (for example the end-device uses a longer RECEIVE_DELAY1 and RECEIVE_DELAY2 latency), those parameters must be communicated to the network server using an out-of-band channel during the end-device commissioning process. The network server may not accept parameters different from those default values.

7.2 US 902-928MHz ISM Band

7.2.1 US902-928 Preamble Format

The following synchronization words should be used:

Modulation	Sync word	Preamble length
LORA	0x34	8 symbols

LoRaWAN does not make use of GFSK modulation in the US902-928 ISM band.

7.2.2 US902-928 Channel Frequencies

The 915 MHz ISM Band shall be divided into the following channel plans.

- Upstream – 64 channels numbered 0 to 63 utilizing LoRa 125 kHz BW varying from DR0 toDR3 starting at 902.3 MHz and incrementing linearly by 200 kHz to 914.9 MHz
- Upstream – 8 channels numbered 64 to 71 utilizing LoRa 500 kHz BW at DR4 starting at 903.0 MHz and incrementing linearly by 1.6 MHz to 914.2 MHz
- Downstream – 8 channels numbered 0 to 7 utilizing LoRa 500 kHz BW at DR10 to DR13) starting at 923.3 MHz and incrementing linearly by 600 kHz to 927.5 MHz

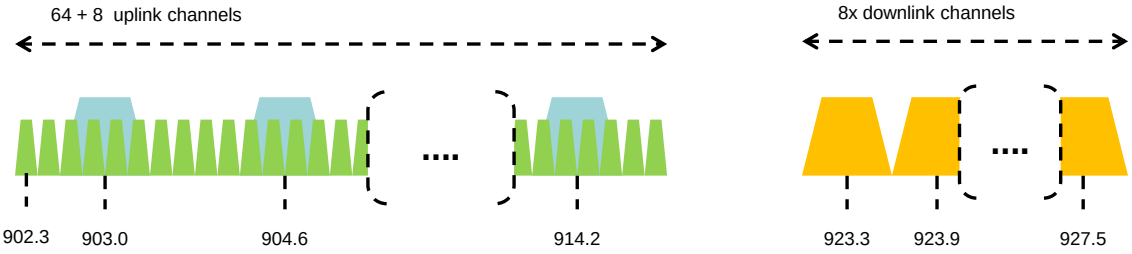


Figure 10: US902-928 channel frequencies

915 MHz ISM band end-devices should use the following default parameters:

- Default radiated transmit output power: 20 dBm
  - Devices, when transmitting with 125 kHz BW may use a maximum of +30 dBm. The transmission should never last more than 400 ms.
  - Devices, when transmitting with 500 kHz BW may use a maximum of +26 dBm

US902-928 end-devices should be capable of operating in the 902 to 928 MHz frequency band and should feature a channel data structure to store the parameters of 72 channels. A channel data structure corresponds to a frequency and a set of data rates usable on this frequency.

If using the over-the-air activation procedure, the end-device should broadcast the JoinReq message alternatively on a random 125 kHz channel amongst the 64 channels defined using **DR0** and a random 500 kHz channel amongst the 8 channels defined using **DR4**. The end-device should change channel for every transmission.

7.2.3 US902-928 Data Rate and End-point Output Power encoding

The following encoding is used for Data Rate (**DR**) and End-point Output Power (**TXPower**) in the US902-928 band:

DataRate	Configuration	Indicative physical bit rate [bit/sec]	TXPower	Configuration
0	LoRa: SF10 / 125 kHz	980	0	30 dBm – 2*TXpower
1	LoRa: SF9 / 125 kHz	1760	1	28 dBm
2	LoRa: SF8 / 125 kHz	3125	2	26 dBm
3	LoRa: SF7 / 125 kHz	5470	3 : 9	….
4	LoRa: SF8 / 500 kHz	12500	10	10 dBm
5:7	RFU		11:15	RFU
8	LoRa: SF12 / 500 kHz	980		
9	LoRa: SF11 / 500 kHz	1760		
10	LoRa: SF10 / 500 kHz	3900		
11	LoRa: SF9 / 500 kHz	7000		
12	LoRa: SF8 / 500 kHz	12500		
13	LoRa: SF7 / 500 kHz	21900		
14:15	RFU			

Table 18: Data rate and TX power table (Rem: DR4 is identical to DR12, DR8..13 must be implemented in end-devices and are reserved for future applications)

7.2.4 US902-928 JoinResp CFList

The US902-928 LoRaWAN does not support the use of the optional **CFlist** appended to the JoinResp message. If the **CFlist** is not empty it is ignored by the end-device.

7.2.5 US902-928 LinkAdrReq command

For the US902-928 version the **ChMaskCntl** field of the ***LinkADRReq*** command has the following meaning:

ChMaskCntl	ChMask applies to
0	Channels 0 to 15
1	Channels 16 to 31
..	..
4	Channels 64 to 71
5	RFU
6	All 125 kHz ON ChMask applies to channels 65 to 72
7	All 125 kHz OFF ChMask applies to channels 65 to 72

Table 19: ChMaskCntl value table

If **ChMaskCntl** = 6 (resp 7) then 125 kHz channels are enabled (resp disabled). Simultaneously the channels 64 to 71 are set according to the **ChMask** bit mask.

> **Note:** FCC regulation requires hopping over at least 50 channels when using maximum output power. It is possible to have end-devices with less channels (at least six 125 kHz channels) when limiting the end-device transmit power to 21 dBm.

7.2.6 US902-928 Maximum payload size

The maximum **MACPayload** size length (*M*) is given by the following table. It is derived from the maximum allowed transmission time at the PHY layer taking into account a possible repeater encapsulation. The maximum application payload length in the absence of the optional **FOpt** MAC control field (*N*) is also given for information only. The value of *N* might be smaller if the **FOpt** field is not empty:

DataRate	*M*	*N*
0	19	11
1	61	53
2	137	129
3	250	242
4	250	242
5:7	Not defined	
8	41	33
9	117	109
10	230	222
11	230	222
12	230	222
13	230	222
14:15	Not defined	

Table 20: US902-928 maximum payload size (repeater compatible)

The greyed lines correspond to the data rates that may be used by an end-device behind a repeater.

If the end-device will never operate under a repeater then the maximum application payload length in the absence of the optional **FOpt** control field should be:

DataRate	*M*	*N*
0	19	11
1	61	53
2	137	129
3	250	242
4	250	242
5:7	Not defined	
8	61	53
9	137	129
10	250	242
11	250	242
12	250	242
13	250	242
14:15	Not defined	

Table 21 : US902-928 maximum payload size (not repeater compatible)

7.2.7 US902-928 Receive windows

- The RX1 receive channel is a function of the upstream channel used to initiate the data exchange. The RX1 receive channel can be determined as follows.
  - RX1 Channel Number = Transmit Channel Number modulo 8
- The RX1 window data rate depends on the transmit data rate (see Table 22 below).
- The RX2 (second receive window) settings uses a fixed data rate and frequency. Default parameters are 923.3Mhz / DR8

Upstream data rate	**Downstream data rate**			
RX1DROffset	**0**	**1**	**2**	**3**
DR0	DR10	DR9	DR8	DR8
DR1	DR11	DR10	DR9	DR8
DR2	DR12	DR11	DR10	DR9
DR3	DR13	DR12	DR11	DR10
DR4	DR13	DR13	DR12	DR11
DR8	DR8	DR8	DR8	DR8
DR9	DR9	DR8	DR8	DR8
DR10	DR10	DR9	DR8	DR8
DR11	DR11	DR10	DR9	DR8
DR12	DR12	DR11	DR10	DR9
DR13	DR13	DR12	DR11	DR10

Table 22: Data rate mapping

The allowed values for RX1DROffset are in the [0:3] range. Values in the range [4:7] are reserved for future use.

7.2.8 US902-928 Default Settings

The following parameters are recommended values for the US902-928 band.

RECEIVE_DELAY1	1 s
RECEIVE_DELAY2	2 s (must be RECEIVE_DELAY1 + 1s)
JOIN_ACCEPT_DELAY1	5 s
JOIN_ACCEPT_DELAY2	6 s
MAX_FCNT_GAP	16384
ADR_ACK_LIMIT	64
ADR_ACK_DELAY	32
ACK_TIMEOUT	2 +/- 1 s (random delay between 1 and 3 seconds)

If the actual parameter values implemented in the end-device are different from those default values (for example the end-device uses a longer RECEIVE_DELAY1 & 2 latency), those parameters must be communicated to the network server using an out-of-band channel during the end-device commissioning process. The network server may not accept parameters different from those default values.

7.3 China 779-787MHz ISM Band

7.3.1 CN779-787 Preamble Format

The following synchronization words should be used :

Modulation	Sync word	Preamble length
LORA	0x34	8 symbols
GFSK	0xC194C1	5 bytes

Table 23: CN779-787 synch words

7.3.2 CN779-787 ISM Band channel frequencies

The LoRaWAN can be used in the Chinese 779-787MHz band as long as the radio device EIRP is less than 10mW (or 10dBm).

The end-device transmit duty-cycle should be lower than 1%.

The LoRaWAN channels center frequency can be in the following range:

- Minimum frequency : 779.5Mhz
- Maximum frequency : 786.5 MHz

CN780Mhz end-devices should be capable of operating in the 779 to 787 MHz frequency band and should feature a channel data structure to store the parameters of at least 16 channels. A channel data structure corresponds to a frequency and a set of data rates usable on this frequency.

The first three channels correspond to 779.5, 779.7 and 779.9 MHz with DR0 to DR5 and must be implemented in every end-device. Those default channels cannot be modified through the ***NewChannelReq*** command and guarantee a minimal common channel set between end-devices and gateways of all networks. Other channels can be freely distributed across the allowed frequency range on a network per network basis.

The following table gives the list of frequencies that should be used by end-devices to broadcast the JoinReq message. The JoinReq message transmit duty-cycle should never exceed 0.1%

Modulation	Bandwidth [kHz]	Channel Frequency [MHz]	FSK Bitrate or LoRa DR / Bitrate	Nb Channels	Duty cycle
LoRa	125	779.5 779.7 779.9 780.5 780.7 780.9	DR0 – DR5 / 0.3-5 kbps	6	<0.1%

Table 24: CN780 JoinReq Channel List

7.3.3 CN779-787 Data Rate and End-point Output Power encoding

The following encoding is used for Data Rate (DR) and End-point Output Power (TXPower) in the CN780 band:

DataRate	Configuration	Indicative physical bit rate [bit/s]
0	LoRa: SF12 / 125 kHz	250
1	LoRa: SF11 / 125 kHz	440
2	LoRa: SF10 / 125 kHz	980
3	LoRa: SF9 / 125 kHz	1760
4	LoRa: SF8 / 125 kHz	3125
5	LoRa: SF7 / 125 kHz	5470
6	LoRa: SF7 / 250 kHz	11000
7	FSK: 50 kbps	50000
8..15	RFU	

TXPower	Configuration
0	10 dBm
1	7 dBm
2	4 dBm
3	1 dBm
4	-2 dBm
5	-5 dBm
6..15	RFU

Table 25: Data rate and TX power table

7.3.4 CN779-787 JoinAccept CFList

The CN780 ISM band LoRaWAN implements an optional **channel frequency list** (CFlist) of 16 octets in the JoinAccept message.

In this case the CFList is a list of five channel frequencies for the channels four to eight whereby each frequency is encoded as a 24 bits unsigned integer (three octets). All these channels are usable for DR0 to DR5 125kHz LoRa modulation. The list of frequencies is followed by a single RFU octet for a total of 16 octets.

Size (bytes)	3	3	3	3	3	1
CFList	Freq Ch4	Freq Ch5	Freq Ch6	Freq CN7	Freq Ch8	RFU

The actual channel frequency in Hz is 100 x frequency whereby values representing frequencies below 100 Mhz are reserved for future use. This allows setting the frequency of a channel anywhere between 100 MHz to 1.67 GHz in 100 Hz steps. Unused channels have a frequency value of 0. The **CFList** is optional and its presence can be detected by the length of the join-accept message. If present, the **CFList** replaces all the previous channels stored in the end-device apart from the three default channels as defined in Chapter 6.

The newly defined channels are immediately enabled and usable by the end-device for communication.

7.3.5 CN779-787 LinkAdrReq command

The CN780 LoRaWAN only supports a maximum of 16 channels. When **ChMaskCntl** field is 0 the ChMask field individually enables/disables each of the 16 channels.

ChMaskCntl	ChMask applies to
0	Channels 1 to 16
1	RFU
..	..
4	RFU
5	RFU
6	All channels ON The device should enable all currently defined channels independently of the ChMask field value.
7	RFU

Table 26: ChMaskCntl value table

If the ChMask field value is one of values meaning RFU, then end-device should reject the command and unset the "**Channel mask ACK**" bit in its response.

7.3.6 CN779-787 Maximum payload size

The maximum **MACPayload** size length (*M*) is given by the following table. It is derived from limitation of the PHY layer depending on the effective modulation rate used taking into account a possible repeater encapsulation layer. The maximum application payload length in the absence of the optional **FOpt** control field (*N*) is also given for information only. The value of *N* might be smaller if the **FOpt** field is not empty:

DataRate	*M*	*N*
0	59	51
1	59	51
2	59	51
3	123	115
4	230	222
5	230	222
6	250	242
7	230	222
8:15	Not defined	

Table 27: CN780 maximum payload size

If the end-device will never operate with a repeater then the maximum application payload length in the absence of the optional **FOpt** control field should be:

DataRate	*M*	*N*
0	59	51
1	59	51
2	59	51
3	123	115
4	250	242
5	250	242
6	250	242
7	250	242

8:15	Not defined

Table 28 : CN780 maximum payload size (not repeater compatible)

7.3.7 CN779-787 Receive windows

The RX1 receive window uses the same channel than the preceding uplink. The data rate is a function of the uplink data rate and the RX1DROffset as given by the following table. The allowed values for RX1DROffset are in the [0:5] range. Values in the range [6:7] are reserved for future use

RX1DROffset **Upstream data rate**	**0**	**1**	**2**	**3**	**4**	**5**
	Downstream data rate in RX1 slot					
DR0	DR0	DR0	DR0	DR0	DR0	DR0
DR1	DR1	DR0	DR0	DR0	DR0	DR0
DR2	DR2	DR1	DR0	DR0	DR0	DR0
DR3	DR3	DR2	DR1	DR0	DR0	DR0
DR4	DR4	DR3	DR2	DR1	DR0	DR0
DR5	DR5	DR4	DR3	DR2	DR1	DR0
DR6	DR6	DR5	DR4	DR3	DR2	DR1
DR7	DR7	DR6	DR5	DR4	DR3	DR2

The RX2 receive window uses a fixed frequency and data rate. The default parameters are 786 MHz / DR0.

7.3.8 CN779-787 Default Settings

The following parameters are recommended values for the CN779-787MHz band.

RECEIVE_DELAY1	1 s
RECEIVE_DELAY2	2 s (must be RECEIVE_DELAY1 + 1s)
JOIN_ACCEPT_DELAY1	5 s
JOIN_ACCEPT_DELAY2	6 s
MAX_FCNT_GAP	16384
ADR_ACK_LIMIT	64
ADR_ACK_DELAY	32
ACK_TIMEOUT	2 +/- 1 s (random delay between 1 and 3 seconds)

If the actual parameter values implemented in the end-device are different from those default values (for example the end-device uses a longer RECEIVE_DELAY1 and RECEIVE_DELAY2 latency), those parameters must be communicated to the network server using an out-of-band channel during the end-device commissioning process. The network server may not accept parameters different from those default values.

7.4 EU 433MHz ISM Band

7.4.1 EU433 Preamble Format

The following synchronization words should be used :

Modulation	Sync word	Preamble length
LORA	0x34	8 symbols
GFSK	0xC194C1	5 bytes

Table 29: EU433 synch words

7.4.2 EU433 ISM Band channel frequencies

The LoRaWAN can be used in the ETSI 433-434 MHz band as long as the radio device EIRP is less than 10 mW (or 10 dBm).

The end-device transmit duty-cycle should be lower than 1%[1].

The LoRaWAN channels center frequency can be in the following range:

- Minimum frequency : 433.175 MHz
- Maximum frequency : 434.665 MHz

EU433 end-devices should be capable of operating in the 433.05 to 434.79 MHz frequency band and should feature a channel data structure to store the parameters of at least 16 channels. A channel data structure corresponds to a frequency and a set of data rates usable on this frequency.

The first three channels correspond to 433.175, 433.375 and 433.575 MHz with DR0 to DR5 and must be implemented in every end-device. Those default channels cannot be modified through the ***NewChannelReq*** command and guarantee a minimal common channel set between end-devices and gateways of all networks. Other channels can be freely distributed across the allowed frequency range on a network per network basis.

The following table gives the list of frequencies that should be used by end-devices to broadcast the JoinReq message. The JoinReq message transmit duty-cycle should never exceed 0.1%

Modulation	Bandwidth [kHz]	Channel Frequency [MHz]	FSK Bitrate or LoRa DR / Bitrate	Nb Channels	Duty cycle
LoRa	125	433.175 433.375 433.575	DR0 – DR5 / 0.3-5 kbps	3	<1%

Table 30: EU433 JoinReq Channel List

[1] The EN300220 ETSI standard limits to 10% the maximum transmit duty-cycle in the 433MHz ISM band. The LoRaWAN requires a 1% transmit duty-cycle lower than the legal limit to avoid network congestion.

7.4.3 EU433 Data Rate and End-point Output Power encoding

The following encoding is used for Data Rate (DR) and End-point Output Power (TXPower) in the EU433 band:

DataRate	Configuration	Indicative physical bit rate [bit/s]
0	LoRa: SF12 / 125 kHz	250
1	LoRa: SF11 / 125 kHz	440
2	LoRa: SF10 / 125 kHz	980
3	LoRa: SF9 / 125 kHz	1760
4	LoRa: SF8 / 125 kHz	3125
5	LoRa: SF7 / 125 kHz	5470
6	LoRa: SF7 / 250 kHz	11000
7	FSK: 50 kbps	50000
8..15	RFU	

TXPower	Configuration
0	10 dBm
1	7 dBm
2	4 dBm
3	1 dBm
4	-2 dBm
5	-5 dBm
6..15	RFU

Table 31: Data rate and TX power table

7.4.4 EU433 JoinAccept CFList

The EU433 ISM band LoRaWAN implements an optional **channel frequency list** (CFlist) of 16 octets in the JoinAccept message.

In this case the CFList is a list of five channel frequencies for the channels four to eight whereby each frequency is encoded as a 24 bits unsigned integer (three octets). All these channels are usable for DR0 to DR5 125 kHz LoRa modulation. The list of frequencies is followed by a single RFU octet for a total of 16 octets.

Size (bytes)	3	3	3	3	3	1
CFList	Freq Ch4	Freq Ch5	Freq Ch6	Freq CN7	Freq Ch8	RFU

The actual channel frequency in Hz is 100 x frequency whereby values representing frequencies below 100 Mhz are reserved for future use. This allows setting the frequency of a channel anywhere between 100 MHz to 1.67 GHz in 100 Hz steps. Unused channels have a frequency value of 0. The **CFList** is optional and its presence can be detected by the length of the join-accept message. If present, the **CFList** replaces all the previous channels stored in the end-device apart from the three default channels as defined in Chapter 6.

The newly defined channels are immediately enabled and usable by the end-device for communication.

7.4.5 EU433 LinkAdrReq command

The CN780 LoRaWAN only supports a maximum of 16 channels. When **ChMaskCntl** field is 0 the ChMask field individually enables/disables each of the 16 channels.

ChMaskCntl	ChMask applies to
0	Channels 1 to 16
1	RFU
..	..
4	RFU
5	RFU
6	All channels ON The device should enable all currently defined channels independently of the ChMask field value.
7	RFU

Table 32: ChMaskCntl value table

If the ChMask field value is one of the values meaning RFU, then end-device should reject the command and unset the "**Channel mask ACK**" bit in its response.

7.4.6 EU433 Maximum payload size

The maximum **MACPayload** size length (*M*) is given by the following table. It is derived from limitation of the PHY layer depending on the effective modulation rate used taking into account a possible repeater encapsulation layer. The maximum application payload length in the absence of the optional **FOpt** control field (*N*) is also given for information only. The value of N might be smaller if the **FOpt** field is not empty:

DataRate	*M*	*N*
0	59	51
1	59	51
2	59	51
3	123	115
4	230	222
5	230	222
6	230	222
7	230	222
8:15	Not defined	

Table 33: EU433 maximum payload size

If the end-device will never operate with a repeater then the maximum application payload length in the absence of the optional **FOpt** control field should be:

DataRate	M	N
0	59	51
1	59	51
2	59	51
3	123	115
4	250	242
5	250	242
6	250	242
7	250	242

8:15	Not defined

Table 34 : EU433 maximum payload size (not repeater compatible)

7.4.7 EU433 Receive windows

The RX1 receive window uses the same channel than the preceding uplink. The data rate is a function of the uplink data rate and the RX1DROffset as given by the following table. The allowed values for RX1DROffset are in the [0:5] range. Values in the range [6:7] are reserved for future use.

RX1DROffset **Upstream data rate**	**0**	**1**	**2**	**3**	**4**	**5**
	Downstream data rate in RX1 slot					
DR0	DR0	DR0	DR0	DR0	DR0	DR0
DR1	DR1	DR0	DR0	DR0	DR0	DR0
DR2	DR2	DR1	DR0	DR0	DR0	DR0
DR3	DR3	DR2	DR1	DR0	DR0	DR0
DR4	DR4	DR3	DR2	DR1	DR0	DR0
DR5	DR5	DR4	DR3	DR2	DR1	DR0
DR6	DR6	DR5	DR4	DR3	DR2	DR1
DR7	DR7	DR6	DR5	DR4	DR3	DR2

The RX2 receive window uses a fixed frequency and data rate. The default parameters are 434.665MHz / DR0 (SF12 , 125kHz)

7.4.8 EU433 Default Settings

The following parameters are recommended values for the EU863-870Mhz band.

RECEIVE_DELAY1	1 s
RECEIVE_DELAY2	2 s (must be RECEIVE_DELAY1 + 1s)
JOIN_ACCEPT_DELAY1	5 s
JOIN_ACCEPT_DELAY2	6 s
MAX_FCNT_GAP	16384
ADR_ACK_LIMIT	64
ADR_ACK_DELAY	32
ACK_TIMEOUT	2 +/- 1 s (random delay between 1 and 3 seconds)

If the actual parameter values implemented in the end-device are different from those default values (for example the end-device uses a longer RECEIVE_DELAY1 & 2 latency) , those parameters must be communicated to the network server using an out-of-band channel during the end-device commissioning process. The network server may not accept parameters different from those default values.

CLASS B – BEACON

8 Introduction to Class B

This section describes the LoRaWAN Class B layer which is optimized for battery-powered end-devices that may be either mobile or mounted at a fixed location.

End-devices should implement Class B operation when there is a requirement to open receive windows at fixed time intervals for the purpose of enabling server initiated downlink messages.

LoRaWAN Class B option adds a synchronized reception window on the end-device.

One of the limitations of LoRaWAN Class A is the Aloha method of sending data from the end-device; it does not allow for a known reaction time when the customer application or the server wants to address the end-device. The purpose of Class B is to have an end-device available for reception on a predictable time, in addition to the reception windows that follows the random uplink transmission from the end-device of Class A. Class B is achieved by having the gateway sending a beacon on a regular basis to synchronize the all the end-devices in the network so that the end-device can opening a short extra reception window (called “ping slot”) at a predictable time during a periodic time slot.

> **Note:** The decision to switch from Class A to Class B comes from the application layer of the end-device. If this class A to Class B switch needs to be controlled from the network side, the customer application must use one of the end-device’s Class A uplinks to send back a downlink to the application layer, and it needs the application layer on the end-device to recognize this request – this process is not managed at the LoRaWAN level.

9 Principle of synchronous network initiated downlink (Class-B option)

For a network to support end-devices of Class B, all gateways must synchronously broadcast a beacon providing a timing reference to the end-devices. Based on this timing reference the end-devices can periodically open receive windows, hereafter called “ping slots”, which can be used by the network infrastructure to initiate a downlink communication. A network initiated downlink using one of these ping slots is called a “ping”. The gateway chosen to initiate this downlink communication is selected by the network server based on the signal quality indicators of the last uplink of the end-device. For this reason, if an end-device moves and detects a change in the identity advertised in the received beacon, it must send an uplink to the network server so that the server can update the downlink routing path database.

All end-devices start and join the network as end-devices of Class A. The end-device application can then decide to switch to Class B. This is done through the following process:

- The end-device application requests the LoRaWAN layer to switch to Class B mode. The LoRaWAN layer in the end-device searches for a beacon and returns either a BEACON_LOCKED service primitive to the application if a network beacon was found and locked or a BEACON_NOT_FOUND service primitive. To accelerate the beacon discovery the LoRaWAN layer may use the “BeaconTimingReq” message described later.
- Based on the beacon strength and the battery life constraints, the end-device application selects a ping slot data rate and periodicity, this is then requested them from the end-device LoRaWAN layer.
- Once in Class B mode, the MAC layer sets to 1 the *Class B* bit of the FCTRL field of every uplink frame transmitted. This bit signals to the server that the device has switched to Class B. The MAC layer will autonomously schedule a reception slot for each beacon and each ping slot. When the beacon reception is successful the end-device LoRaWAN layer forwards the beacon content to the application together with the measured radio signal strength. The end-device LoRaWAN layer takes into account the maximum possible clock drift in the scheduling of the beacon reception slot and ping slots. When a downlink is successfully demodulated during a ping slot, it is processed similarly to a downlink as described in the LoRaWAN Class A specification.
- A mobile end-device must periodically inform the network server of its location to update the downlink route. This is done by transmitting a normal (possibly empty) “unconfirmed” or “confirmed” uplink. The end-device LoRaWAN layer will appropriately set the *Class B* bit to 1. Optimally this can be done more efficiently if the application detects that the node is moving by analyzing the beacon content. In that case the end-device must apply a random delay (as defined in Section 15.5 between the beacon reception and the uplink transmission to avoid systematic uplink collisions.
- If no beacon has been received for a given period (as defined in Section 12.2), the synchronization with the network is lost. The MAC layer must inform the application layer that it has switched back to Class A. As a consequence the end-device LoRaWAN layer stops setting the *Class B* bit in all uplinks and this informs the network server that the end-device is no longer in Class B mode. The end-device application can try to switch back to Class B periodically. This will restart this process starting with a beacon search.

The following diagram illustrates the concept of beacon reception slots and ping slots.

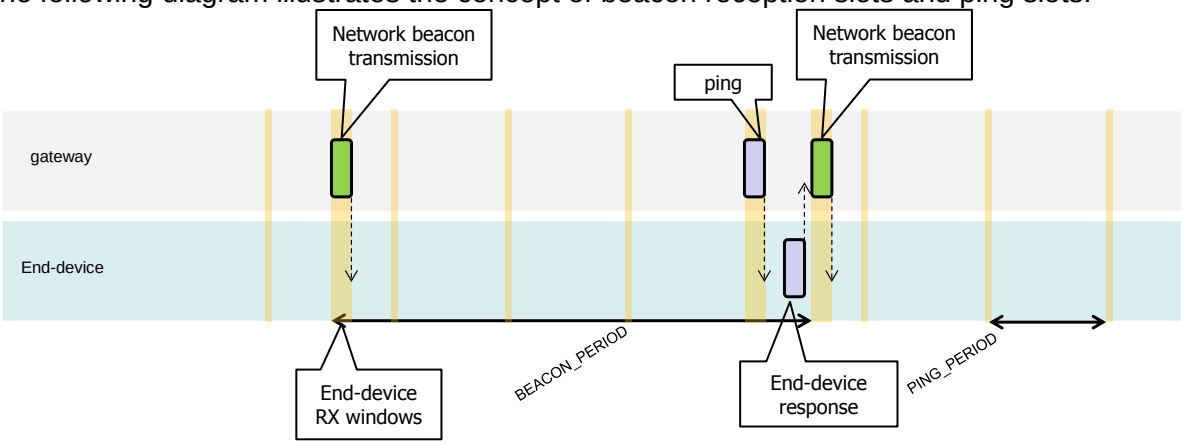


Figure 11: Beacon reception slot and ping slots

In this example, given the beacon period is 128 s, the end-device also opens a ping reception slot every 32 s. Most of the time this ping slot is not used by the server and therefore the end-device reception window is closed as soon as the radio transceiver has assessed that no preamble is present on the radio channel. If a preamble is detected the radio transceiver will stay on until the downlink frame is demodulated. The MAC layer will then process the frame, check that its address field matches the end-device address and that the Message Integrity Check is valid before forwarding it to the application layer.

10 Uplink frame in Class B mode

The uplink frames in Class B mode are same as the Class A uplinks with the exception of the RFU bit in the FCtrl field in the Frame header. In the Class A uplink this bit is unused (RFU). This bit is used for Class B uplinks.

Bit#	7	6	5	4	3..0
FCtrl	ADR	ADRACKReq	ACK	Class B	FOptsLen

The *Class B* bit set to 1 in an uplink signals the network server that the device as switched to Class B mode and is now ready to receive scheduled downlink pings.

The signification of the FPending bit for downlink is unaltered and still signals that one or more downlink frames are queued for this device in the server and that the device should keep is receiver on as described in the Class A specification.

11 Downlink Ping frame format (Class B option)

11.1 Physical frame format

A downlink Ping uses the same format as a Class A downlink frame but might follow a different channel frequency plan.

11.2 Unicast & Multicast MAC messages

Messages can be “unicast” or “multicast”. Unicast messages are sent to a single end-device and multicast messages are sent to multiple end-devices. All devices of a multicast group must share the same multicast address and associated encryption keys. The LoRaWAN Class B specification does not specify means to remotely setup such a multicast group or securely distribute the required multicast key material. This must either be performed during the node personalization or through the application layer.

11.2.1 Unicast MAC message format

The MAC payload of a unicast downlink **Ping** uses the format defined in the Class A specification. It is processed by the end-device in exactly the same way. The same frame counter is used and incremented whether the downlink uses a Class B ping slot or a Class A “piggy-back” slot.

11.2.2 Multicast MAC message format

The Multicast frames share most of the unicast frame format with a few exceptions:

- They are not allowed to carry MAC commands, neither in the **FOpt** field, nor in the payload on port 0 because a multicast downlink does not have the same authentication robustness as a unicast frame.
- The **ACK** and **ADRACKReq** bits must be zero. The **MType** field must carry the value for Unconfirmed Data Down.
- The **FPending** bit indicates there is more multicast data to be sent. If it is set the next multicast receive slot will carry a data frame. If it is not set the next slot may or may not carry data. This bit can be used by end-devices to evaluate priorities for conflicting reception slots.

12 Beacon acquisition and tracking

Before switching from Class A to Class B, the end-device must first receive one of the network beacons to align his internal timing reference with the network.

Once in Class B, the end-device must periodically search and receive a network beacon to cancel any drift of its internal clock time base, relative to the network timing.

A Class B device may be temporarily unable to receive beacons (out of range from the network gateways, presence of interference, ..). In this event, the end-device has to gradually widen its beacon and ping slots reception windows to take into account a possible drift of its internal clock.

> **Note:** For example, a device which internal clock is defined with a +/- 10ppm precision may drift by +/-1.3mSec every beacon period.

12.1 Minimal beacon-less operation time

In the event of beacon loss, a device shall be capable of maintaining Class B operation for 2 hours (120 minutes) after it received the last beacon. This temporary Class B operation without beacon is called “beacon-less” operation. It relies on the end-device’s own clock to keep timing.

During beacon-less operation, unicast, multicast and beacon reception slots must all be progressively expanded to accommodate the end-device’s possible clock drift.

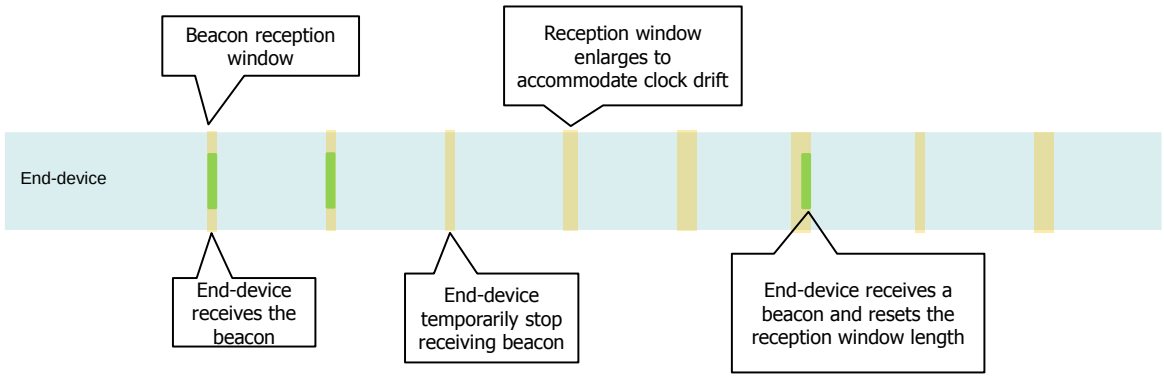


Figure 12 : beacon-less temporary operation

12.2 Extension of beacon-less operation upon reception

During this 120 minutes time interval the reception of any beacon directed to the end-device, should extend the Class B beacon-less operation further by another 120 minutes as it allows to correct any timing drift and reset the receive slots duration.

12.3 Minimizing timing drift

The end-devices may use the beacon’s (when available) precise periodicity to calibrate their internal clock and therefore reduce the initial clock frequency imprecision. As the timing oscillator’s exhibit a predictable temperature frequency shift, the use of a temperature sensor could enable further minimization of the timing drift.

13 Class B Downlink slot timing

13.1 Definitions

To operate successfully in Class B the end-device must open reception slots at precise instants relative to the infrastructure beacon. This section defines the required timing.

The interval between the start of two successive beacons is called the beacon period. The beacon frame transmission is aligned with the beginning of the BEACON_RESERVED interval. Each beacon is preceded by a guard time interval where no ping slot can be placed. The length of the guard interval corresponds to the time on air of the longest allowed frame. This is to insure that a downlink initiated during a ping slot just before the guard time will always have time to complete without colliding with the beacon transmission. The usable time interval for ping slot therefore spans from the end of the beacon reserved time interval to the beginning of the next beacon guard interval.

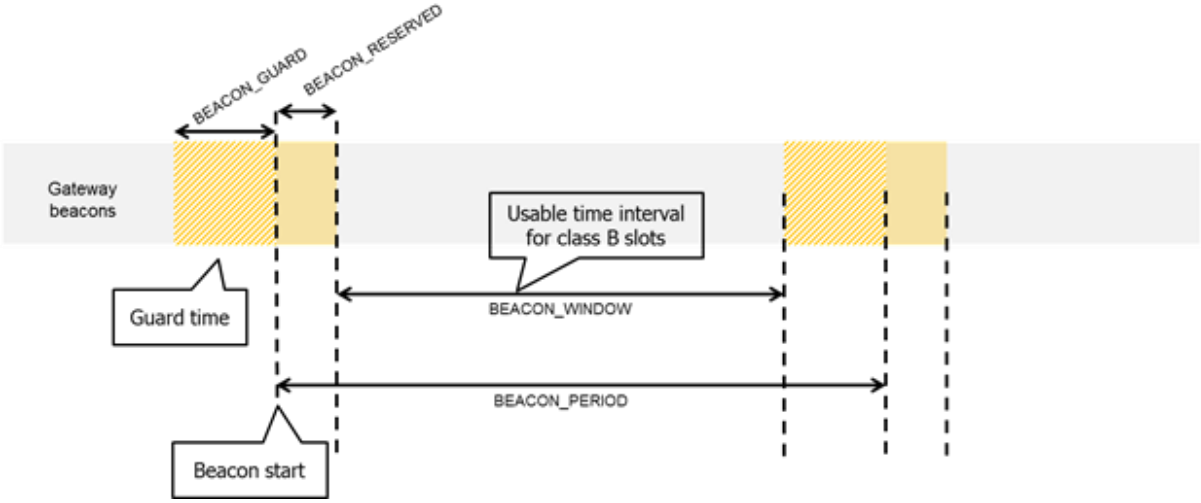


Figure 13: Beacon timing

Beacon_period	128 s
Beacon_reserved	2.120 s
Beacon_guard	3.000 s
Beacon-window	122.880 s

Table 35: Beacon timing

The beacon frame time on air is actually much shorter than the beacon reserved time interval to allow appending network management broadcast frames in the future.

The beacon window interval is divided into 2^{12} = 4096 ping slots of 30 ms each numbered from 0 to 4095.

An end-device using the slot number N must turn on its receiver exactly *Ton* seconds after the start of the beacon where:

$$Ton = beacon\_reserved + N * 30 \text{ ms}$$

N is called the *slot index.*

The latest ping slot starts at *beacon_reserved* + 4095 * 30 ms = 124 970 ms after the beacon start or 3030 ms before the beginning of the next beacon.

13.2 Slot randomization

To avoid systematic collisions or over-hearing problems the slot index is randomized and changed at every beacon period.

The following parameters are used:

DevAddr	Device 32 bit network unicast or multicast address
pingNb	Number of ping slots per beacon period. This must be a power of 2 integer: *pingNb* = 2^k where 1 <= k <=7
pingPeriod	Period of the device receiver wake-up expressed in number of slots: *pingPeriod* = 2^{12} / *pingNb*
pingOffset	Randomized offset computed at each beacon period start. Values can range from 0 to (pingPeriod-1)
beaconTime	The time carried in the field **BCNPayload**.Time of the immediately preceding beacon frame
slotLen	Length of a unit ping slot = 30 ms

At each beacon period the end-device and the server compute a new pseudo-random offset to align the reception slots. An AES encryption with a fixed key of all zeros is used to randomize:

Key = 16 x 0x00
Rand = aes128_encrypt(Key, beaconTime | DevAddr | pad16)
pingOffset = (*Rand*[0] + *Rand*[1]x 256) modulo *pingPeriod*

The slots used for this beacon period will be:
pingOffset + *N* x *pingPeriod* with *N*=[0:*pingNb*-1]

The node therefore opens receive slots starting at :

First slot	Beacon_reserved + pingOffset x slotLen
Slot 2	Beacon_reserved + (pingOffset + pingPeriod) x slotLen
Slot 3	Beacon_reserved + (pingOffset + 2 x pingPeriod) x slotLen
…	…

If the end-device serves simultaneously a unicast and one or more multicast slots this computation is performed multiple times at the beginning of a new beacon period. Once for the unicast address (the node network address) and once for each multicast group address.

In the case where a multicast ping slot and a unicast ping slot collide and cannot be served by the end-device receiver then the end-device should preferentially listen to the multicast slot. If there is a collision between multicast reception slots the FPending bit of the previous multicast frame can be used to set a preference.

The randomization scheme prevents a systematic collision between unicast and multicast slots. If collisions happen during a beacon period then it is unlikely to occur again during the next beacon period.

14 Class B MAC commands

All commands described in the Class A specification shall be implemented in Class B devices. The Class B specification adds the following MAC commands.

CID	Command	Transmitted by		Short Description
		End-device	Gateway	
0x10	***PingSlotInfoReq***	x		Used by the end-device to communicate the ping unicast slot data rate and periodicity to the network server
0x10	***PingSlotInfoAns***		x	Used by the network to acknowledge a PingInfoSlotReq command
0x11	***PingSlotChannelReq***		x	Used by the network server to set the unicast ping channel of an end-device
0x11	***PingSlotFreqAns***	x		Used by the end-device to acknowledge a ***PingSlotChannelReq***command
0x12	***BeaconTimingReq***	x		Used by end-device to request next beacon timing & channel to network
0x12	***BeaconTimingAns***		x	Used by network to answer a ***BeaconTimingReq*** uplink
0x13	***BeaconFreqReq***		x	Command used by the network server to modify the frequency at which the end-device expects to receive beacon broadcast
0x13	***BeaconFreqAns***	x		Used by the end-device to acknowledge a BeaconFreqReq command

14.1 PingSlotInfoReq

With the ***PingSlotInfoReq*** command an end-device informs the server of its unicast ping slot periodicity and expected data rate. This command must only be used to inform the server of the parameters of a UNICAST ping slot. A multicast slot is entirely defined by the application and should not use this command.

Size (bytes)	1
PingSlotInfoReq Payload	Periodicity & data rate

Bit#	7	[6:4]	[3:0]
Periodicity & data rate	RFU	Periodicity	Data rate

The **Periodicity** subfield is an unsigned 3 bits integer encoding the ping slot period currently used by the end-device using the following equation.

$$pingSlotPeriod = 2^{Periodicity} \text{ in seconds}$$

- **Periodicity** = 0 means that the end-device opens a ping slot every second
- **Periodicity** = 7 , every 128 seconds which is the maximum ping period supported by the LoRaWAN Class B specification.

The **Data rate** subfield encodes the data rate at which the end-point expects any ping. This uses the same encoding scheme that the ***LinkAdrReq*** command described in the Class A specification.

The server needs to be aware of the end-device ping slot periodicity or expected data rate else Class B downlinks will not happen successfully. For that purpose the ***PingSlotInfoReq*** MAC command **must be acknowledged** with a ***PingSlotInfoAns*** before the device can switch from class A to Class B**.** To change its ping slot scheduling or data rate a device should first revert to Class A , send the new parameters through a ***PingSlotInfoReq*** command and get an acknowledge from the server through a ***PinSlotInfoAns*** . It can then switch back to Class B with the new parameters.

This command can be concatenated with any other MAC command in the **FHDRFOpt** field as described in the Class A specification frame format.

14.2 BeaconFreqReq

This command is sent by the server to the end-device to modify the frequency on which this end-device expects the beacon.

Octets	3
PingSlotChannelReqPayload	Frequency

The Frequency coding is identical to the ***NewChannelReq*** MAC command defined in the Class A.

Frequency is a 24bits unsigned integer. The actual beacon channel frequency in Hz is 100 x frequ. This allows defining the beacon channel anywhere between 100 MHz to 1.67 GHz by 100 Hz step. The end-device has to check that the frequency is actually allowed by its radio hardware and return an error otherwise.

A valid non-zero Frequency will force the device to listen to the beacon on a fixed frequency channel even if the default behavior specifies a frequency hopping beacon (i.e US ISM band).

A value of 0 instructs the end-device to use the default beacon frequency plan as defined in the “Beacon physical layer” section. Where applicable the device resumes frequency hopping beacon search.

14.3 PingSlotChannelReq

This command is sent by the server to the end-device to modify the frequency on which this end-device expects the downlink pings.

Octets	3	1
PingSlotChannelReq Payload	Frequency	DrRange

The Frequency coding is identical to the ***NewChannelReq*** MAC command defined in the Class A.

Frequency is a 24bits unsigned integer. The actual ping channel frequency in Hz is 100 x frequ. This allows defining the ping channel anywhere between 100MHz to 1.67GHz by 100Hz step. The end-device has to check that the frequency is actually allowed by its radio hardware and return an error otherwise.

A value of 0 instructs the end-device to use the default frequency plan.

DrRange is the data rate range allowed on this channel. This byte is split in two 4-bit indexes.

Bits	7:4	3:0
DrRange	Max data rate	Min data rate

Following the convention defined in the “Physical layer” section of the Class A specification, the “Min data rate” subfield designates the lowest data rate allowed on this channel. For example 0 designates DR0 / 125 kHz in the EU physical layer. Similarly “Max data rate” designates the highest data rate. For example in the EU spec, DrRange = 0x77 means that only 50 kbps GFSK is allowed on a channel and DrRange = 0x50 means that DR0 / 125 kHz to DR5 / 125 kHz are supported.

Upon reception of this command the end-device answers with a ***PingSlotFreqAns*** message. The MAC payload of this message contains the following information:

Size (bytes)	1
pingSlotFreqAns Payload	Status

The **Status** bits have the following meaning:

Bits	7:2	1	0
Status	RFU	Data rate range ok	Channel frequency ok

	Bit = 0	**Bit = 1**
Data rate range ok	The designated data rate range exceeds the ones currently defined for this end device, the previous range is kept	The data rate range is compatible with the possibilities of the end device
Channel frequency ok	The device cannot use this frequency, the previous ping frequency is kept	The device is able to use this frequency.

14.4 BeaconTimingReq

This command is sent by the end-device to request the next beacon timing and channel. This MAC command has no payload. The ***BeaconTimingReq*** & ***BeaconTimingAns*** mechanism is only meant to accelerate the initial beacon search to lower the end-device energy requirement.

The network may answer only a limited number of requests per a given time period. An end-device must not expect that ***BeaconTimingReq*** is answered immediately with a ***BeaconTimingAns***. Class A end-devices wanting to switch to Class B should not transmit more than one ***BeaconTimingReq*** per hour.

End-devices requiring a fast beacon lock must implement an autonomous beacon finding algorithm.

14.5 BeaconTimingAns

This command is sent by the network to answer a ***BeaconInfoReq*** request.

Size (bytes)	2	1
BeaconInfoReqPayload	Delay	Channel

The "**Delay**" field is a 16bits unsigned integer. If the remaining time between the end of the current downlink frame and the start of the next beacon frame is noted *RTime* then:

*30 ms x (**Delay**+1) > RTime* >= 30 ms x **Delay**

In networks where the beacon uses alternatively several channels, the "**Channel**" field is the index of the beaconing channel on which the next beacon will be broadcasted. For networks where the beacon broadcast frequency is fixed then this field content is 0.

15 Beaconing (Class B option)

15.1 Beacon physical layer

Besides relaying messages between end-devices and network servers, all gateways participate in providing a time-synchronization mechanisms by sending beacons at regular fixed intervals configurable per network (BEACON_INTERVAL). All beacons are transmitted in radio packet implicit mode, that is, without a LoRa physical header and with no CRC being appended by the radio.

PHY	Preamble	BCNPayload

The beacon Preamble begins with (a longer than default) 10 unmodulated symbols. This allows end-devices to implement a low power duty-cycled beacon search.

The beacon frame length is tightly coupled to the operation of the radio Physical layer. Therefore the actual frame length might change from one region implementation to another. The changing fields are highlighted in **Bold** in the following sections.

15.1.1 EU 863-870MHz ISM Band

The beacons are transmitted using the following settings

DR	3	Corresponds to SF9 spreading factor with 125 kHz BW
CR	1	Coding rate = 4/5
frequency	869.525 MHz	This is the recommended frequency allowing +27 dBm EIRP. Network operators may use a different frequency as long as ETSI compliance is achieved

The beacon frame content is:

Size (bytes)	3	4	**1**	7	**2**
BCNPayload	NetID	Time	**CRC**	GwSpecific	**CRC**

15.1.2 US 902-928MHz ISM Band

The beacons are transmitted using the following settings:

DR	10	Corresponds to SF10 spreading factor with 500kHz bw
CR	1	Coding rate = 4/5
frequencies	923.3 to 927.5MHz with 600kHz steps	Beaconing is performed on the same channel that normal downstream traffic as defined in the Class A specification

The downstream channel used for a given beacon is:

$$\text{Channel} = \left[floor\left(\frac{beacon\_time}{beacon\_period}\right)\right] \ modulo\ 8$$

- whereby beacon_time is the integer value of the 4 bytes “Time” field of the beacon frame
- whereby beacon_period is the periodicity of beacons , 128 seconds
- whereby *floor(x)* designates rounding to the integer immediately inferior to x

Example: the first beacon will be transmitted on 923.3Mhz , the second on 923.9MHz, the 9th beacon will be on 923.3Mhz again.

Beacon channel nb	Frequency [MHz]
0	923.3
1	923.9
2	924.5
3	925.1
4	925.7
5	926.3
6	926.9
7	927.5

The beacon frame content is:

Size (bytes)	3	4	**2**	7	**1**	**2**
BCNPayload	NetID	Time	**CRC**	GwSpecific	**RFU**	**CRC**

15.2 Beacon frame content

The beacon payload **BCNPayload** consists of a network common part and a gateway-specific part.

Size (bytes)	3	4	1/2	7	0/1	2
BCNPayload	NetID	Time	CRC	GwSpecific	RFU	CRC

The network common part contains a network identifier **NetID** to uniquely identify the network for which the beacon is sent, and a timestamp **Time** in seconds since 00:00:00 Coordinated Universal Time (UTC), 1 January 1970. The integrity of the beacon's network common part is protected by an 8 or 16 bits CRC depending on PHY layer parameters. The CRC-16 is computed on the NetID+Time fields as defined in the IEEE 802.15.4-2003 section 7.2.1.8. When an 8 bits CRC is required then the 8 LSBs of the computed CRC-16 are used.

For example: This is a valid EU868 beacon frame:

AA BB CC | 00 00 02 CC | 7E | 00 | 01 20 00 | 00 81 03 | DE 55

Bytes are transmitted left to right. The corresponding field values are:

Field	NetID	Time	**CRC**	InfoDesc	lat	long	**CRC**
Value Hex	CCBBAA	CC020000	**7E**	0	002001	038100	**55DE**

The CRC-16 of the NetID+Time fields is 0xC87E but only the 8LSBs are used in that case

The seven LSB of the **NetID** are called **NwkID** and match the seven MSB of the short address of an end-device. Neighboring or overlapping networks **must have** different **NwkIDs**.

The gateway specific part provides additional information regarding the gateway sending a beacon and therefore may differ for each gateway. The RFU field when applicable (region specific) should be equal to 0. The optional part is protected by a CRC-16 computed on the GwSpecific+RFU fields. The CRC-16 definition is the same as for the mandatory part.

For example: This is a valid US900 beacon:

Field	NetID	Time	**CRC**	InfoDesc	lat	long	**RFU**	**CRC**
Value Hex	CCBBAA	CC020000	**C87E**	0	002001	038100	**00**	**D450**

Over the air the bytes are sent in the following order:

AA BB CC | 00 00 02 CC | 7E C8 | 00 | 01 20 00 | 00 81 03 |00 | 50 D4

Listening and synchronizing to the network common part is sufficient to operate a stationary end-device in Class B mode. A mobile end-device should also demodulate the gateway specific part of the beacon to be able to signal to the network server whenever he is moving from one cell to another.

> **Note:** As mentioned before, all gateways send their beacon at exactly the same point in time (i.e., time-synchronously) so that for network common part there are no visible on-air collisions for a listening end-device even if the end-device simultaneously receives beacons from several gateways. With respect to the gateway specific part, collision occurs but an end-device within the proximity of more than one gateway will still be able to decode the strongest beacon with high probability.

15.3 Beacon GwSpecific field format

The content of the **GwSpecific** field is as follow:

Size (bytes)	1	6
GwSpecific	InfoDesc	Info

The information descriptor**InfoDesc** describes how the information field **Info** shall be interpreted.

InfoDesc	**Meaning**
0	GPS coordinate of the gateway first antenna
1	GPS coordinate of the gateway second antenna
2	GPS coordinate of the gateway third antenna
3:127	RFU
128:255	Reserved for custom network specific broadcasts

For a single omnidirectional antenna gateway the **InfoDesc** value is 0 when broadcasting GPS coordinates. For a site featuring 3 sectored antennas for example, the first antenna broadcasts the beacon with **InfoDesc** equals 0, the second antenna with **InfoDesc** field equals 1, etc …

15.3.1 Gateway GPS coordinate: InfoDesc = 0, 1 or 2

For **InfoDesc** = 0 ,1 or 2, the content of the **Info** field encodes the GPS coordinates of the antenna broadcasting the beacon

Size (bytes)	3	3
Info	Lat	Lng

The latitude and longitude fields (**Lat** and **Lng**, respectively) encode the geographical location of the gateway as follows:

- The north-south latitude is encoded using a signed 24 bit word where -2^{23} corresponds to 90° south (the South Pole) and 2^{23} corresponds to 90° north (the North Pole). The equator corresponds to 0.
- The east-west longitude is encoded using a signed 24 bit word where -2^{23}corresponds to 180° west and 2^{23} corresponds to 180° east. The Greenwich meridian corresponds to 0.

15.4 Beaconing precise timing

The beacon is sent every 128 seconds starting at 00:00:00 Coordinated Universal Time (UTC), 1 January 1970 plus **NwkID plus** TBeaconDelay. Therefore the beacon is sent at

$B_T = k * 128 +$ **NwkID** + TBeaconDelay

seconds after 00:00:00 Coordinated Universal Time (UTC), 1 January 1970

where*k* is the smallest integer for which

$k * 128 +$ **NwkID**$> T$

whereby

T = seconds since 00:00:00 Coordinated Universal Time (UTC), 1 January 1970.

> **Note:** *T* is not (!) Unix time. Similar to GPS time and unlike Unix time, *T* is strictly monotonically increasing and is not influenced by leap seconds.

Whereby TBeaconDelay is a network specific delay in the [0:50] ms range. TBeaconDelay may vary from one network to another and is meant to allow a slight transmission delay of the gateways. TBeaconDelay must be the same for all gateways of a given network. TBeaconDelay must be smaller than 50 ms. All end-devices ping slots use the beacon transmission time as a timing reference, therefore the network server as to take TBeaconDelay into account when scheduling the class B downlinks.

15.5 Network downlink route update requirements

When the network attempts to communicate with an end-device using a Class B downlink slot, it transmits the downlink from the gateway which was closest to the end-device when the last uplink was received. Therefore the network server needs to keep track of the rough position of every Class B device.

Whenever a Class B device moves and changes cell, it needs to communicate with the network server in order to update its downlink route. This update can be performed simply by sending a “confirmed” or “unconfirmed” uplink, possibly without applicative payload.

The end-device has the choice between 2 basic strategies:

- Systematic periodic uplink: simplest method that doesn't require demodulation of the "gateway specific" field of the beacon. Only applicable to slowly moving or stationery end-devices. There are no requirements on those periodic uplinks.
- Uplink on cell change: The end-device demodulates the "gateway specific" field of the beacon, detects that the ID of the gateway broadcasting the beacon it demodulates has changed, and sends an uplink. In that case the device should respect a pseudo random delay in the [0:120] seconds range between the beacon demodulation and the uplink transmission. This is required to insure that the uplinks of multiple Class B devices entering or leaving a cell during the same beacon period will not systematically occur at the same time immediately after the beacon broadcast.

Failure to report cell change will result in Class B downlink being temporary not operational. The network server may have to wait for the next end-device uplink to transmit downlink traffic.

16 Class B unicast & multicast downlink channel frequencies

16.1 EU 863-870MHz ISM Band

All unicast&multicastClass B downlinks use a single frequency channel defined by the "***PingSlotChannelReq"*** MAC command. The default frequency is 869.525MHz

16.2 US 902-928MHz ISM Band

By default Class B downlinks use a channel function of the Time field of the last beacon (see Beacon Frame content) and the DevAddr.

$$\text{Class B downlink channel} = \left[\text{DevAddr} + \text{floor}\left(\frac{\text{Beacon\_Time}}{\text{Beacon\_period}}\right)\right] \text{ modulo } 8$$

- Whereby Beacon_Time is the 32 bit Time field of the current beacon period
- Beacon_period is the length of the beacon period (defined as 128sec in the specification)
- Floor designates rounding to the immediately lower integer value
- DevAddr is the 32 bits network address of the device

Class B downlinks therefore hop across 8 channels in the ISM band and all Class B end-devices are equally spread amongst the 8 downlink channels.

If the "***PingSlotChannelReq"*** command with a valid non-zero argument is used to set the Class B downlink frequency then all subsequent ping slots should be opened on this single frequency independently of the last beacon frequency.

If the "***PingSlotChannelReq"*** command with a zero argument is sent, the end-device should resume the default frequency plan, id Class B ping slots hoping across 8 channels.

The underlying idea is to allow network operators to configure end-devices to use a single proprietary dedicated frequency band for the Class B downlinks if available, and to keep as much frequency diversity as possible when the ISM band is used.

CLASS C – CONTINUOUSLY LISTENING

17 Class C: Continuously listening end-device

The end-devices implanting the Class C option are used for applications that have sufficient power available and thus do not need to minimize reception time.

Class C end-devices cannot implement Class B option.

The Class C end-device will listen with RX2 windows parameters as often as possible. The end-device listens on RX2 when it is not either (a) sending or (b) receiving on RX1, according to Class A definition. To do so, it will open a short window on RX2 parameters between the end of the uplink transmission and the beginning of the RX1 reception window and it will switch to RX2 reception parameters as soon as the RX1 reception window is closed; the RX2 reception window will remain open until the end-device has to send another message.

> **Note:** There is not specific message for a node to tell the server that it is a Class C node. It is up to the application on server side to know that it manages Class C nodes based on the contract passed during the join procedure.

17.1 Second receive window duration for Class C

Class C devices implement the same two receive windows as Class A devices, but they do not close RX2 window until they need to send again. Therefore they may receive a downlink in the RX2 window at nearly any time. A short listening window on RX2 frequency and data rate is also opened between the end of the transmission and the beginning of the RX1 receive window.

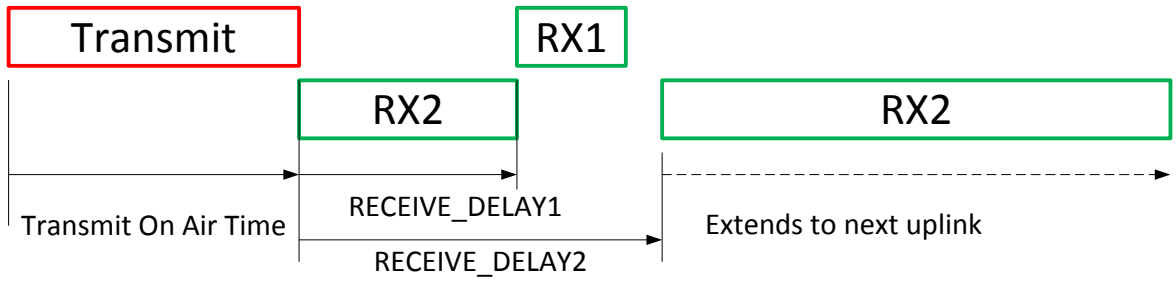


Figure 14: Class C end-device receive slot timing.

17.2 Class C Multicast downlinks

Similarly to Class B, Class C devices may receive multicast downlink frames. The multicast address and associated network session key and application session key must come from the application layer. The same limitations apply for Class C multicast downlink frames:

- They are not allowed to carry MAC commands, neither in the **FOpt** field, nor in the payload on port 0 because a multicast downlink does not have the same authentication robustness as a unicast frame.
- The **ACK** and **ADRACKReq** bits must be zero. The **MType** field must carry the value for Unconfirmed Data Down.
- The **FPending** bit indicates there is more multicast data to be sent. Given that a Class C device keeps its receiver active most of the time, the **FPending** bit does not trigger any specific behavior of the end-device.

SUPPORT INFORMATION

This sub-section is only a recommendation.

18 Examples and Application Information

Examples are illustrations of the LoRaWAN spec for information, but they are not part of the formal specification.

18.1 Uplink Timing Diagram for Confirmed Data Messages

The following diagram illustrates the steps followed by an end-device trying to transmit two confirmed data frames (Data0 and Data1):

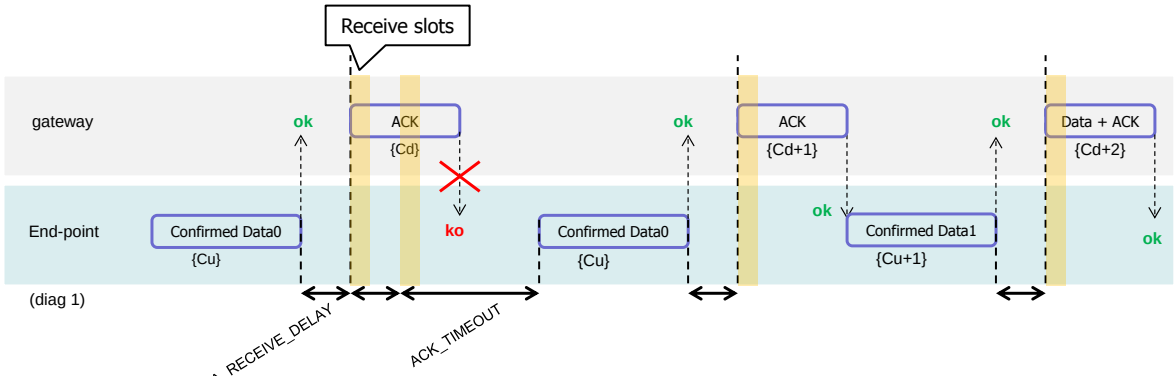


Figure 15: Uplink timing diagram for confirmed data messages

The end-device first transmits a confirmed data frame containing the Data0 payload at an arbitrary instant and on an arbitrary channel. The frame counter Cu is simply derived by adding 1 to the previous uplink frame counter. The network receives the frame and generates a downlink frame with the ACK bit set exactly RECEIVE_DELAY1 seconds later, using the first receive window of the end-device. This downlink frame uses the same data rate and the same channel as the Data0 uplink. The downlink frame counter Cd is also derived by adding 1 to the last downlink towards that specific end-device. If there is no downlink payload pending the network shall generate a frame without a payload. In this example the frame carrying the ACK bit is not received.

If an end-point does not receive a frame with the ACK bit set in one of the two receive windows immediately following the uplink transmission it may resend the same frame with the same payload and frame counter again at least ACK_TIMEOUT seconds after the second reception window. This resend must be done on another channel and must obey the duty cycle limitation as any other normal transmission. If this time the network receives the ACK downlink during its first receive window, as soon as the ACK frame is demodulated, the end-device is free to transmit a new frame on a new channel.

18.2 The third ACK frame in this example also carries an application payload. A downlink frame can carry any combination of ACK, MAC control commands and payload. Downlink Diagram for Confirmed Data Messages

The following diagram illustrates the basic sequence of a “confirmed” downlink.

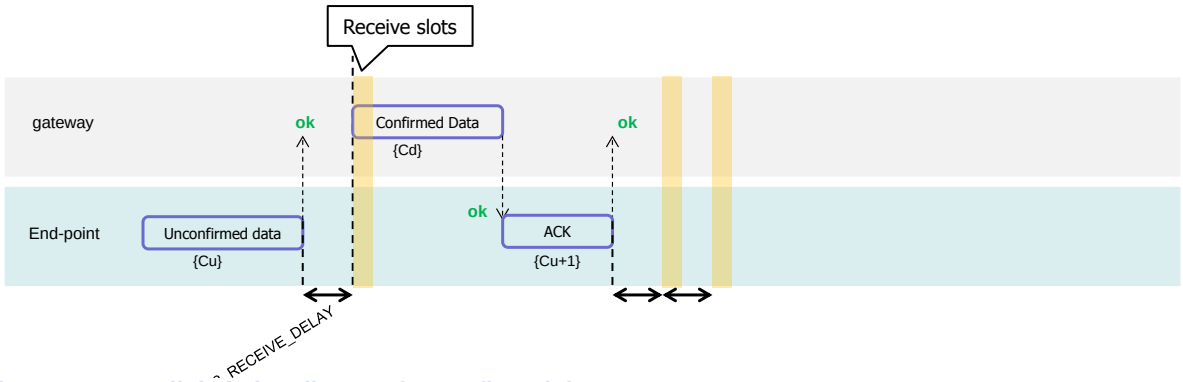


Figure 16: Downlink timing diagram for confirmed data messages

The frame exchange is initiated by the end-device transmitting an “unconfirmed” application payload or any other frame on channel A. The network uses the downlink receive window to transmit a “confirmed” data frame towards the end-device on the same channel A. Upon reception of this data frame requiring an acknowledgement, the end-device transmits a frame with the ACK bit set at its own discretion. This frame might also contain piggybacked data or MAC commands as its payload. This ACK uplink is treated like any standard uplink, and as such is transmitted on a random channel that might be different from channel A.

> **Note:** To allow the end-devices to be as simple as possible and have keep as few states as possible it may transmit an explicit (possibly empty) acknowledgement data message immediately after the reception of a data message requiring an acknowledgment. Alternatively the end-device may defer the transmission of an acknowledgement to piggyback it with its next data message.

18.3 Downlink Timing for Frame-Pending Messages

The next diagram illustrates the use of the **frame pending** (FPending) bit on a downlink. The FPending bit can only be set on a downlink frame and informs the end-device that the network has several frames pending for him; the bit is ignored for all uplink frames.

If a frame with the FPending bit set requires an acknowledgement, the end-device shall do so as described before. If no acknowledgment is required, the end-device may send an empty data message to open additional receive windows at its own discretion, or wait until it has some data to transmit itself and open receive windows as usual.

> **Note:** The FPending bit is independent to the acknowledgment scheme.

(*) F_P means ‘frame pending’ bit set

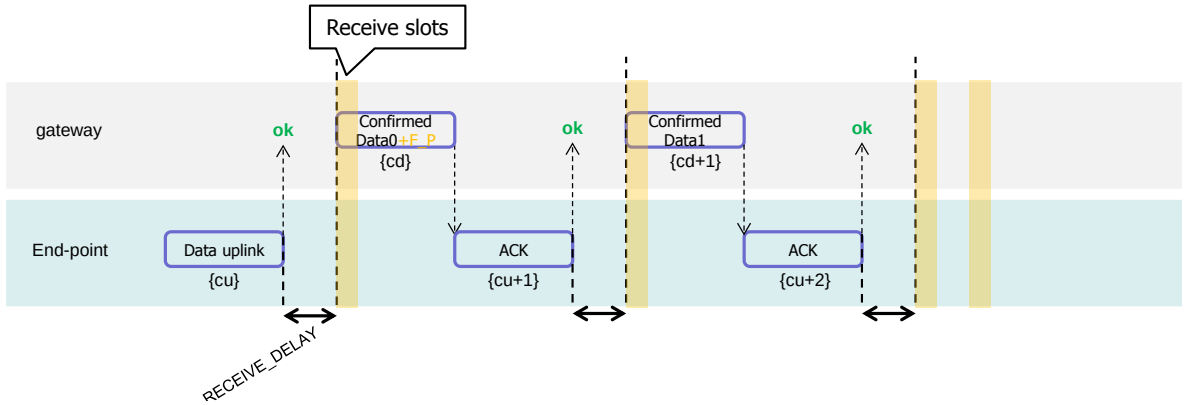


Figure 17: Downlink timing diagram for frame-pending messages, example 1

In this example the network has two confirmed data frames to transmit to the end-device. The frame exchange is initiated by the end-device via a normal “unconfirmed” uplink message on channel A. The network uses the first receive window to transmit the Data0 with the bit FPending set as a confirmed data message. The device acknowledges the reception of the frame by transmitting back an empty frame with the ACK bit set on a new channel B. RECEIVE_DELAY1 seconds later, the network transmits the second frame Data1 on channel B, again using a confirmed data message but with the FPending bit cleared. The end-device acknowledges on channel C.

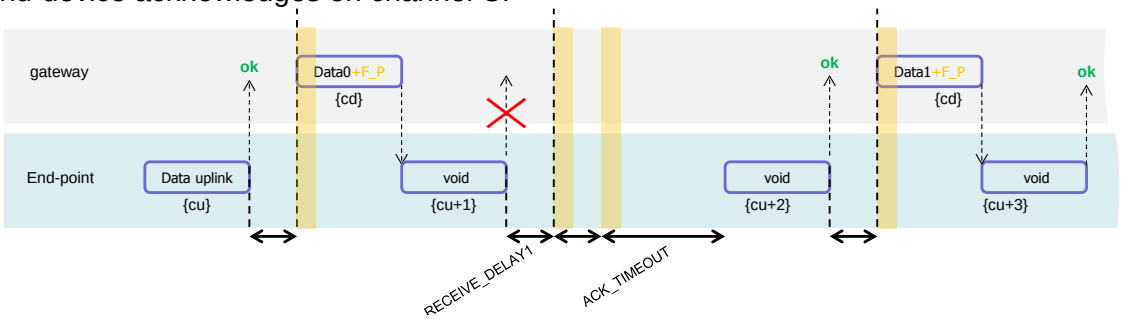


Figure 18: Downlink timing diagram for frame-pending messages, example 2

In this example, the downlink frames are “unconfirmed” frames, the end-device does not need to send back and acknowledge. Receiving the Data0 unconfirmed frame with the FPending bit set the end-device sends an empty data frame. This first uplink is not received by the network. If no downlink is received during the two receive windows, the network has to wait for the next spontaneous uplink of the end-device to retry the transfer. The end-device can speed up the procedure by sending a new empty data frame.

> **Note:** An acknowledgement is never sent twice.

The FPending bit, the ACK bit, and payload data can all be present in the same downlink. For example, the following frame exchange is perfectly valid.

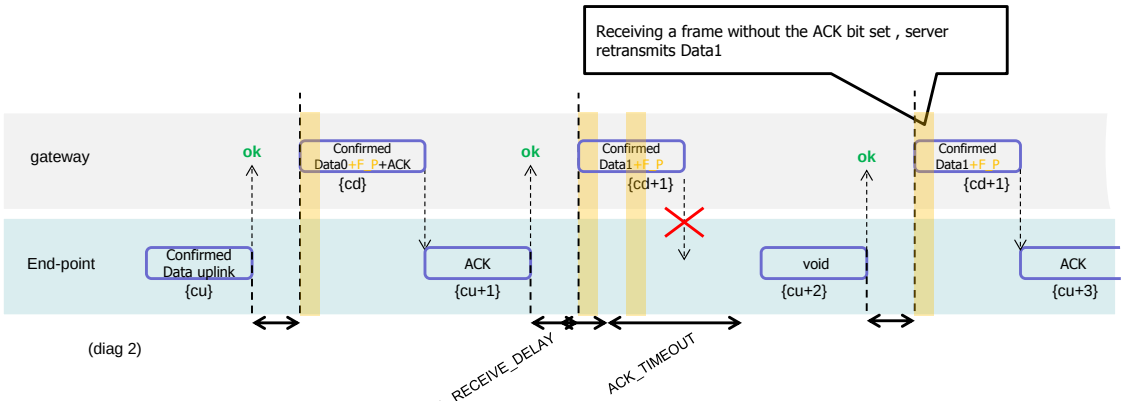


Figure 19: Downlink timing diagram for frame-pending messages, example 3

The end-device sends a “confirmed data” uplink. The network can answer with a confirmed downlink containing Data + ACK + “Frame pending” then the exchange continues as previously described.

18.4 Data-Rate Adaptation during Message Retransmissions

When an end-device attempts the transmission of a “confirmed’ frame toward the network it expects to receive an acknowledgement in one of the subsequent reception slot. In the absence of the acknowledgement it will try to re-transmit the same data again. This re-transmission happens on a new frequency channel, but can also happen at a different data rate (preferable lower) than the previous one. It is strongly recommended to adopt the following re-transmission strategy.

The first transmission of the “confirmed” frame happens with a data rate DR.

Transmission nb	Data Rate
1 (first)	DR
2	DR
3	max(DR-1,0)
4	max(DR-1,0)
5	max(DR-2,0)
6	max(DR-2,0)
7	max(DR-3,0)
8	max(DR-3,0)

The Data Rate max(a,b) stands for maximum of a and b values.

If after a recommended 8 transmissions, the frame has not been acknowledged the MAC layer should return an error code to the application layer.

> **Note:** For each re-transmission, the frequency channel is selected randomly as for normal transmissions.

Any further transmission uses the last data rate used.

For example if an end-device sends a “confirmed” frame first using DR5 and has to retransmit 3 times (twice at DR5 then twice at DR4), the next frame transmitted will use DR4

Other example, if an end-device sends a “confirmed” frame first using DR5 and does not receive an acknowledge after 8 transmissions (2 at DR5, 2 at DR4, … , 2 at DR2), and the application of this end-device re-initiates a “confirmed” transmission a little later, the first two transmission will be tentatively at DR2, then switch to DR1, then to DR0.

19 Recommendation on contract to be provided to the network server by the end-device provider at the time of provisioning

Configuration data related to the end-device and its characteristics must be known by the network server at the time of provisioning. –This provisioned data is called the “contract”. This contract cannot be provided by the end-device and must be supplied by the end-device provider using another channel (out-of-band communication).

This end-device contract is stored in the network server. It can be used by the application server and the network controller to adapt the algorithms.

This data will include:

- End-device specific radio parameters (device frequency range, device maximal output power, device communication settings - RECEIVE_DELAY1, RECEIVE_DELAY2)
- Application type (Alarm, Metering, Asset Tracking, Supervision, Network Control)

20 Recommendation on finding the locally used channels

End-devices that can be activated in territories that are using different frequencies for LoRaWAN will have to identify what frequencies are supported for join message at their current location before they send any message. The following methods are proposed:

- A GPS enabled end-device can use its GPS location to identify which frequency band to use.
- End-device can search for a beacon and use its frequency to identify its region
- End-device can search for a beacon and if this one is sending the antenna GPS coordinate, it can use this to identify its region
- End-device can search for a beacon and if this one is sending a list of join frequencies, it can use this to send its join message

21 Revisions

21.1 Revision 1.0

- Draft version of LoRaWAN

22 Glossary

ADR	Adaptive Data Rate
AES	Advanced Encryption Standard
AFA	Adaptive Frequency Agility
AR	Acknowledgement Request
CBC	Cipher Block Chaining
CMAC	Cipher-based Message Authentication Code
CR	Coding Rate
CRC	Cyclic Redundancy Check
DR	Data Rate
ECB	Electronic Code Book
ETSI	European Telecommunications Standards Institute
EIRP	Equivalent Isotropically Radiated Power
FSK	Frequency Shift Keying modulation technique
GPRS	General Packet Radio Service
HAL	Hardware Abstraction Layer
IP	Internet Protocol
LBT	Listen Before Talk
LoRa™	Long Range modulation technique
LoRaWAN™	Long Range Network protocol
MAC	Medium Access Control
MIC	Message Integrity Code
RF	Radio Frequency
RFU	Reserved for Future Usage
Rx	Receiver
RSSI	Received Signal Strength Indicator
SF	Spreading Factor
SNR	Signal Noise Ratio
SPI	Serial Peripheral Interface
SSL	Secure Socket Layer
Tx	Transmitter
USB	Universal Serial Bus

23 Bibliography

23.1 References

[IEEE802154]: IEEE Standard for Local and Metropolitan Area Networks—Part 15.4: Low-Rate Wireless Personal Area Networks (LR-WPANs), IEEE Std 802.15.4TM-2011 (Revision of IEEE Std 802.15.4-2006), September 2011.

[RFC4493]: The AES-CMAC Algorithm, June 2006.

24 NOTICE OF USE AND DISCLOSURE

LoRa Alliance, Inc.

2400 Camino Ramon, Suite 375

San Ramon, CA 94583

UNIVERSIDADE DA CORUÑA

ESCUELA POLITÉCNICA SUPERIOR

TRABAJO FIN DE MÁSTER

CURSO 2016/2017

DISEÑO Y DESARROLLO DEL SISTEMA DE COMUNICACIONES INALÁMBRICO DE UN ENJAMBRE DE UAV'S COLABORATIVOS

MÁSTER EN INGENIERÍA INDUSTRIAL

DOCUMENTO DEL PROYECTO

ANEXO III – TRANSMISIÓN DE MENSAJES

ESTRUCTURA DE MENSAJES

Los paquetes a transmitir en las redes LoRaWAN™ siguen la estructura definida por la LoRaWAN™ Specification, separando la estructura de Capa Física (PHY) y de Capa de Enlace (MAC).

La terminología de LoRa distingue entre dos tipos de mensajes:

- Los Mensajes Uplink son enviados por dispositivos finales al servidor de red retransmitido por uno o varios gateway.
- Los Mensajes Downlink son enviados por desde el servidor de red a un único dispositivo final y es retransmitido a su vez por un único gateway.

Los paquetes en una comunicación en red LoRaWAN™ contarán con la siguiente estructura:

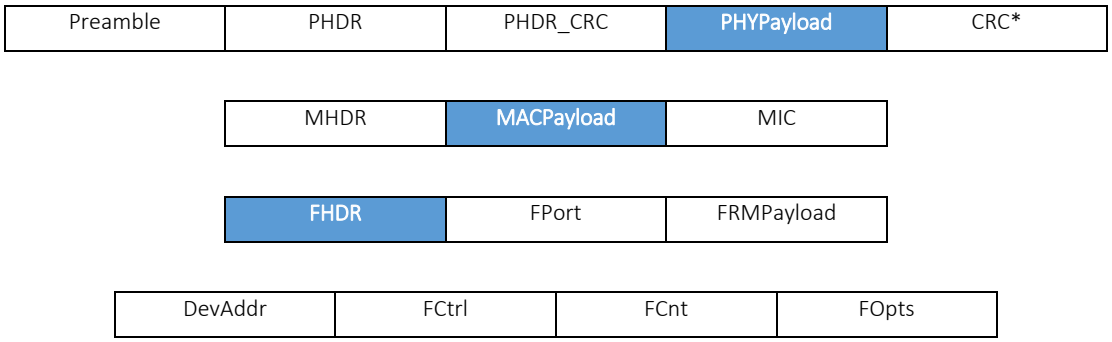
- Preamble o sincronización (LoRA: 0x34, (G)FSK: 0xC194C1).
- LoRa Physical Header (PHDR), establecido por el transceptor de radio.
- Header CRC (PHDR_CRC), establecido por el transceptor de radio.
- Physical Payload (PHYPayload).
- Codificación de Redundancia Cíclica (CRC), establecido por el transceptor de radio.

Preamble	PHDR	PHDR_CRC	PHYPayload	CRC*

* En el caso de mensajes downlink no estaría disponible la estructura CRC.

Estructura AIII.1.1: Estructura de Mensajes.

Todos los paquetes en LoRa presentan una estructura de PHYPayload que se descompone en las siguientes estructuras:



Estructura AIII.1.2: Descomposición de la PHYPayload.

La descomposición de PHYPayload nos deriva en las siguientes estructuras:

- MAC Header (MHDR):

La MAC Header especificará el tipo de mensaje (MType) y la versión principal (Major) del formato de la trama de la especificación de la capa LoRaWAN™ con la que ha sido codificada.

Bit	[7 … 5]	[4 … 2]	[1 … 0]
MHDR bits	MType	RFU	Major

Estructura AIII.1.3: Estructura del MAC Header (MHDR).

La LoRaWAN™ Specification distingue entre seis diferentes tipos de mensajes MAC (MType):

MType	Description
000	Join Request
001	Join Accept
010	Unconfirmed Data Up
011	Unconfirmed Data Down
100	Confirmed Data Up
101	Confirmed Data Down
110	RFU

Tabla AIII.1.1: Tipos de Mensajes MAC.

La versión principal (Major) especificará el formato de los mensajes intercambiados en el proceso de conexión a red y los primeros cuatro bytes del MAC Payload. La versión secundaria utilizada por el dispositivo final debe ser previamente conocida por el servidor de red, utilizando mensajes fuera de banda (como parte de la información de personalización del dispositivo).

- MAC Payload (MACPayload):

La MAC Payload es denominada como "marco de datos". Se compone del Frame Header (FHDR), un Port Field (FPort) opcional y el Frame Payload (FRMPayload).

Como podemos observar, el Frame Header (FHDR) se descompondrá de la siguiente manera:

Bytes	[4]	[1]	[2]	[0 … 15]
FHDR	DevAddr	FCtrl	FCnt	FOpts

Estructura AIII.1.4: Estructura del Frame Header (FHDR).

El FHDR contiene la dirección del dispositivo final (DevAddr), un Frame Control (FCtrl) que nos permitirá activar el Velocidad de Datos Adaptativa (ADR) y la Recepción de ACK en paquetes con confirmación, un Frame Control (FCnt) que es usado como un contador en Seguimiento de Envíos y Recepciones y un Frame Options (FOpts) utilizado para el transporte de comandos MAC.

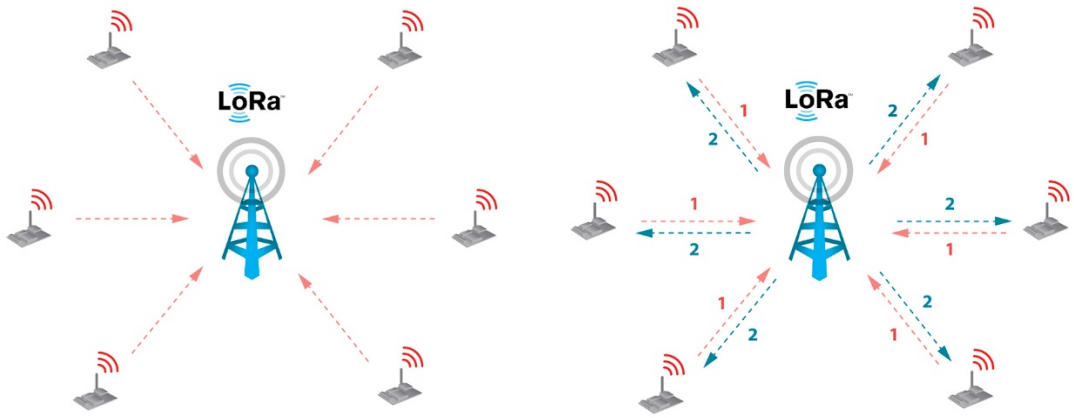


Ilustración AIII.1.1: Comunicación LoRa sin y con Acknowledgement (ACK).

Como se ha visto antes, el Port Field (FPort) presenta un carácter opcional, esto es debido a que si el FRMPayload estuviera vacío no sería necesario. Puede presentar diferentes valores, un valor de 0 indica que el FRMPayload contiene comandos MAC, si este valor fuera entre [1 … 223] contendría aplicaciones específicas y entre [224 … 255] sería espacio reservado para futuras extensiones.

Si el FRMPayload contiene cierta carga útil, este debe ser cifrado antes de que se calcule el Código de Integridad del Mensaje (MIC). El esquema de cifrado utilizado se basa en el algoritmo genérico descrito en IEEE 802.15.4 /2006 Anexo B, utilizando AES-128.

Como valor predeterminado, el cifrado/descifrado se realiza mediante la capa LoRaWAN™ para todos los Port Fields (FPorts). El cifrado/descifrado se puede hacer encima de la capa LoRaWAN™ para Port Fields (FPorts) específicos excepto el puerto 0, si esto fuera más conveniente.

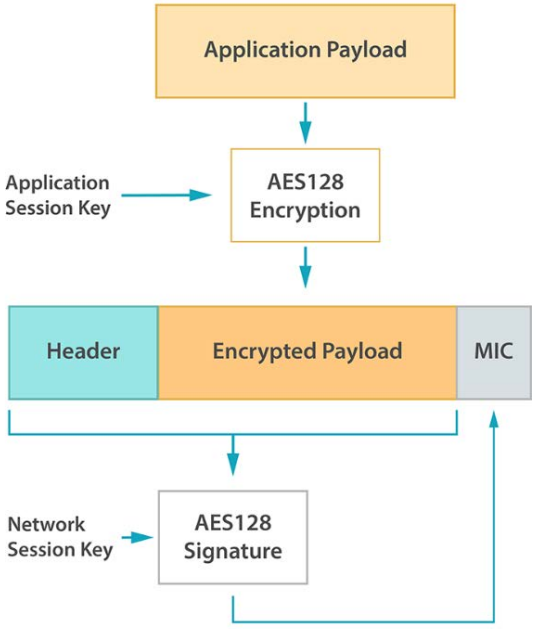


Ilustración AIII.1.2: Cifrado AES-128 de la Carga Útil del Mensaje.

- Message Integrity Code (MIC):

El Message Integrity Code es un código de seguridad que es calculado sobre todos los campos del mensaje.

MIC	MHDR	FHDR	FPort	FRMPayload

Estructura AIII.1.5: Estructura del Message Integrity Code (MIC).

DECODIFICACIÓN DE MENSAJES

En este apartado, se llevará a cabo un ejemplo de Decodificación de Mensaje, se explicará el tratamiento de datos por medio de un ejemplo hecho durante las pruebas de diseño del Gateway.

La AppSKey yNwkSKey estarán previamente establecidas y serán utilizadas en función del FPort utilizado para la transmisión del mensaje:

FPort	Key
0	NwkSKey
[1, …, 255]	AppSKey

Tabla AIII.2.1: Clave utilizada en función de FPort.

El mensaje transmitido por el end-node será sin confirmación (unconfirmed), contendrá la carga útil "LoRa" (hexadecimal), y será enviado a través del FPort 1. Por tanto, la orden transmitida al emisor será la siguiente:

mac tx uncnf 1 4c6f5261

Una vez transmitido el mensaje, el gateway recibirá el mensaje, que deberá descomponerse de la siguiente manera:

MHDR	FHDR	FPort	FRMPayload	MIC
40	a2 1c 01 26 00 01 00	01	79 36 7c 14	e3 85 ce 96

Estructura AIII.2.1: Descomposición del Mensaje Recibido.

Una vez descompuesto, nos queda descifrar la carga útil. Para ello, haremos uso de la LoRaWAN™ Specification, que nos dice que, para el descifrado de los mensajes, se debe trabajar en bloques en 16 bytes.

En primera instancia debemos realizar la creación de bloques de 16 bytes con la carga útil y, en segundo lugar, llevar a cabo la creación de los Bloques A, de mismo número que el resultante de bloques creados con la carga útil.

La estructura de los Bloques A estará compuesta de la siguiente manera:

Size (bytes)	1	4	1	4	4	1	1
Ai	0x01	0x00	Dir	DevAddr	FCnt	0x00	i

Estructura AIII.2.2: Descomposición del Mensaje Recibido.

Por tanto, nuestro mensaje al disponer solo de 4 bytes, dispondrá de un único bloque de Payload y Bloque A:

Payload	[0x79, 0x36, 0x7c, 0x14, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00]
Bloque A	[0x01, 0x00, 0x00, 0x00, 0x00, 0x00, 0xa2, 0x1c, 0x01, 0x26, 0x04, 0x00, 0x00, 0x00, 0x00, 0x01]

Tabla AIII.2.2: Composición de Payload y Bloque A (16 bytes).

Una vez realizado esto, por medio de la AppSKey debemos encriptar los Bloques A generados con cifrado AES-128. El resultado será recogido con el nombre de Bloques S, resultando en este caso:

AppSKey	[0xa4, 0x10, 0x9c, 0x15, 0x0e, 0x24, 0x92, 0xd5, 0x47, 0x9a, 0x78, 0x26, 0x45, 0x75, 0x8b, 0x3b]
Bloque S	[0x35, 0x59, 0x2e, 0x75, 0x88, 0xbc, 0x39, 0x87, 0x14, 0x36, 0xdb, 0xfb, 0xd8, 0xf7, 0x88, 0x7b]

Tabla AIII.2.3: Cifrado AES-128 de Bloque A - Bloque B (16 bytes).

Como último paso, solo nos quedará realizar la operación lógica XOR entre los Bloques Payload y el Bloques S, y al resultado obtenido, eliminando la parte agregada sobre la longitud original del payload, realizar la conversión de hexadecimal a Código ASCII. El resultado obtenido será:

XOR	[0x4c, 0x6f, 0x52, 0x61, 0x88, 0xbc, 0x39, 0x87, 0x14, 0x36, 0xdb, 0xfb, 0xd8, 0xf7, 0x88, 0x7b]
Resultado Final	[0x4c, 0x6f, 0x52, 0x61] -- LoRa

Tabla AIII.2.3: Resultado de Operación Lógica XOR y Resultado Final.

UNIVERSIDADE DA CORUÑA

ESCUELA POLITÉCNICA SUPERIOR

TRABAJO FIN DE MÁSTER

CURSO 2016/2017

DISEÑO Y DESARROLLO DEL SISTEMA DE COMUNICACIONES INALÁMBRICO DE UN ENJAMBRE DE UAV'S COLABORATIVOS

MÁSTER EN INGENIERÍA INDUSTRIAL

DOCUMENTO DEL PROYECTO

ANEXO IV – APLICACIONES DESARROLLADAS

APLICACIÓN DESARROLLADA PARA EL GATEWAY

```
# encoding: utf-8
'''

    Proyecto: Diseño y Desarrollo del Sistema de Comunicaciones de un Enjambre
    de UAV's Colaborativos.

    Integración: Desarrollo de Gateway Single Channel LoRaWAN.
    Lenguaje de Programación: Python 3.

    Alumno: Cristian Méndez Sanmartín.
    Titulación: Máster en Ingeniería Industrial.

    Grupo de Investigación: Grupo Integrado de Ingeniería.
    Universidad: Universidade da Coruña.

    Código Original: Thomas Telkamp.
    GitHub: https://github.com/tftelkamp/single_chan_pkt_fwd

    Características:
      - Gateway LoRa de Canal Único.
      - SF Configurable (SF7-SF12).
      - Frecuencia Configurable (433-868 MHz).
      - Actualización de Estado Automática en Intervalo Configurable.
      - Recepción y Decodificación de Mensajes con Cifrado AES-128.

'''

import sys
from time import localtime, strftime, time, sleep
from socket import socket, AF_INET, SOCK_DGRAM
from ntplib import NTPClient
from re import findall
from uuid import getnode
from wiringpi import wiringPiSetup, wiringPiSPISetup, wiringPiSPIDataRW, pinMode, INPUT, OUTPUT,
digitalRead, digitalWrite, LOW, HIGH
from Crypto.Cipher import AES
```

```
#-----------------------------------------------------------------------------
#
#                          CONFIGURACIÓN DEL GATEWAY LoRAWAN
#
#-----------------------------------------------------------------------------

#  1. Descripción de la red LoRaWAN

gateway_name = 'Single Channel LoRa Gateway'
student_email = 'cristian.mendez@udc.es'
description = 'LoRa Gateway - Grupo Integrado de Ingenieria'

#  2. Configuración de Pines y bus SPI

SPIConfig = {'spi_channel': 0, 'spi_frequency': 500000}  #  2 Channels | Range: 500 kHz - 32 MHz  (1 MHz Default)
pines = {'NReset': 0, 'NSS': 6, 'dio0': 7}

#  3. Configuración de Frecuencia y Detección de SF

configFrequency = [['LF', 433175000], ['HF', 868100000], ['HF', 869525000]]
configSF = [['SF7', 7], ['SF8', 8], ['SF9', 9], ['SF10', 10], ['SF11', 11], ['SF12', 12]]

frequency = configFrequency[2]
sf = configSF[0]

#  4. Configuración de Conexión a Servidor Remoto

host = '192.168.1.33'
port = 1700

#  5. Monitorización de Paquetes y Estado del Gateway

listStatus = ['Init', 'RX', 'RX Done']
status = listStatus[0]

statistics = False
```

```
if statistics is True:
    NTPServer = 'es.pool.ntp.org'
    timestamp = []
    midPacketRSSI =[]
    midRSSI = []
    midSNR = []

receivedPackets = 0
correctPackets = 0
updateInterval = 15

#-----------------------------------------------------------------------
#
#                         CONFIGURACIÓN DE REGISTROS LoRa
#
#-----------------------------------------------------------------------

#   1. LoRa Mode Register Map

REG_FIFO                        = 0x00
REG_OP_MODE                     = 0x01
REG_FRF_MSB                     = 0x06
REG_FRF_MID                     = 0x07
REG_FRF_LSB                     = 0x08
REG_PA_CONFIG                   = 0x09
REG_PA_RAMP                     = 0x0A
REG_OCP                         = 0x0B
REG_LNA                         = 0x0C
REG_FIFO_ADDR_PTR               = 0x0D
REG_FIFO_TX_BASE_ADDR           = 0x0E
REG_FIFO_RX_BASE_ADDR           = 0x0F
REG_FIFO_RX_CURRENT_ADDR        = 0x10
REG_IRQ_FLAGS_MASK              = 0x11
REG_IRQ_FLAGS                   = 0x12
REG_RX_NB_BYTES                 = 0x13
REG_RX_HEADER_CNT_VALUE_MSB     = 0x14
REG_RX_HEADER_CNT_VALUE_LSB     = 0x15
```

```
REG_RX_PACKET_CNT_VALUE_MSB     = 0x16
REG_RX_PACKET_CNT_VALUE_LSB     = 0x17
REG_MODEM_STAT                  = 0x18
REG_PACKET_SNR_VALUE            = 0x19
REG_PACKET_RSSI_VALUE           = 0x1A
REG_RSSI_VALUE                  = 0x1B
REG_HOP_CHANNEL                 = 0x1C
REG_MODEM_CONFIG_1              = 0x1D
REG_MODEM_CONFIG_2              = 0x1E
REG_SYMB_TIMEOUT_LSB            = 0x1F
REG_PREAMBLE_MSB                = 0x20
REG_PREAMBLE_LSB                = 0x21
REG_PAYLOAD_LENGTH              = 0x22
REG_MAX_PAYLOAD_LENGTH          = 0x23
REG_HOP_PERIOD                  = 0x24
REG_FIFO_RX_BYTE_ADDR           = 0x25
REG_MODEM_CONFIG_3              = 0x26
REG_FEI_MSB                     = 0x28
REG_FEI_MID                     = 0x29
REG_FEI_LSB                     = 0x2A
REG_RSSI_WIDEBAND               = 0x2C
REG_DETEC_OPTIMIZE              = 0x31
REG_INVERT_IQ                   = 0x33
REG_DETECTION_THRESHOLD         = 0x37
REG_SYNC_WORD                   = 0x39


# 2. Modos de Operación


SLEEP_MODE              = 0x00
STANDBY_MODE            = 0x01
FSTX_MODE               = 0x02
TX_MODE                 = 0x03
FSRX_MODE               = 0x04
RX_CONTINUOUS_MODE      = 0x05
RX_SINGLE_MODE          = 0x06
CAD_MODE                = 0x07
```

```
#   3. IRQ's del Transceptor SX1276


CAD_DETECTED_MASK             = 0x01
FHHS_CHANGE_CHANNEL_MASK      = 0x02
CAD_DONE_MASK                 = 0x04
TX_DONE_MASK                  = 0x08
VALID_HEADER_MASK             = 0x10
PAYLOAD_CRC_ERROR_MASK        = 0x20
RX_DONE_MASK                  = 0x40
RX_TIMEOUT_MASK               = 0x80


#- - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - -


def readRegister(address):
   digitalWrite(pines['NSS'], LOW) # Select Chip Through SPI
   SPIReading = wiringPiSPIDataRW(SPIConfig['spi_channel'], bytes([address & 0x7F, 0x00])) # 1 byte: 1 bit: read access (0) | 7 bits: address
   digitalWrite(pines['NSS'], HIGH) # Unselect Chip Through SPI


   return SPIReading[1][1] # Return of Read Value


def writeRegister(address, data):
   digitalWrite(pines['NSS'], LOW) # Select Chip Through SPI
   wiringPiSPIDataRW(SPIConfig['spi_channel'], bytes([ 0x80 | address, data])) # 1 byte: 1 bit: write access (1) | 7 bits: address | 1 byte: data
   digitalWrite(pines['NSS'], HIGH) # Unselect Chip Through SPI


#- - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - -


def eventHandler():
   global status


   if digitalRead(pines['dio0']) is HIGH:
      dio0Interrupt() # Set Input polling dio0 interrupt


   if status is listStatus[2]:
      receivePacket() # Reception of Packets
      status = listStatus[1] # Reset to RX Status
```

```
    writeRegister(REG_IRQ_FLAGS, 0xFF) # Clear all IRQ Flags
    writeRegister(REG_IRQ_FLAGS_MASK, 0xFF ^ (RX_DONE_MASK)) # Mask to Accept Only RX Interruptions

def dio0Interrupt():
  global status
  global sf

  IRQFlags = readRegister(REG_IRQ_FLAGS) # Read IRQ Flag Register
  IRQMask = readRegister(REG_IRQ_FLAGS_MASK) # Read IRQ Mask Register

  interruptions = IRQFlags & (0xFF ^ IRQMask) # Return Non Masked Interrupts

  if interruptions is 0x00:
    pass
  else:
    if status is listStatus[1]:
      status = listStatus[2] # Set LoRa RX Done Status
      writeRegister(REG_IRQ_FLAGS, 0xFF) # Clear all IRQ Flags

#-------------------------------------------------------------------------

def readPacket():
  global timestamp

  writeRegister(REG_IRQ_FLAGS_MASK, 0x40)  # Clear RX Done Flag
  IRQFlags = readRegister(REG_IRQ_FLAGS) # Read IRQ Flag Register

  if (IRQFlags & 0x20) is 0x20:
    writeRegister(REG_IRQ_FLAGS, 0x20) # Set CRC Error Flag
    message = ['Failed Reception', 0] # Set Message Value to Fail
  else:
    numBytes = readRegister(REG_RX_NB_BYTES) # Read Message Number Bytes Received
    payload = [] # Payload Buffer for Reading

    for i in range(numBytes):
      payload.append(readRegister(REG_FIFO)) # Read FIFO Register to get the message
```

```
        message = [payload, numBytes] # Compose the message with Payload and Number Bytes Received


    return message # Return Payload and Number Bytes Received


def decodePacket(message):
    decodedMessage = '' # Decoded Message Buffer


    blockLength = 16 # Standard Block Length (Could vary in the last block)
    counter = 9 # Counter for Payload Read (Byte 0: MHDR | Bytes 1-7: FHDR | Byte 8: FPort)


    dataLength = message[1] - 13 # Payload Length: Message Bytes - (Byte 0: MHDR | Bytes 1-7: FHDR | Byte 8: FPort | Bytes n-3 - n: MIC)
    numBlocks = int(dataLength / 16) # Number of Block A
    restLength = dataLength % 16 # Rest May Needed for the Last Block A


    if restLength > 0:
        numBlocks += 1 # Aditional Block A


    devAddr = [message[0][1], message[0][2], message[0][3], message[0][4]] # Get Device Address from Message
    frameCounter = [message[0][6], message[0][7]] # Get Frame Counter from Message


    for i in range(numBlocks):
        AppSKey = bytes([0xA4, 0x10, 0x9C, 0x15, 0x0E, 0x24, 0x92, 0xD5, 0x47, 0x9A, 0x78, 0x26, 0x45, 0x75, 0x8B, 0x3B]) # Application Session Key for AES-128 Encrypt
        mode = AES.MODE_CBC # Set AES CBC Encrypt Mode
        IV = bytes(16 * [0x00]) # Set IV Null


        Crypto = AES.new(AppSKey, mode, IV) # Define new AES Encrypt Object


        BlockA = bytes([0x01, 0x00, 0x00, 0x00, 0x00, 0x00, devAddr[0], devAddr[1], devAddr[2], devAddr[3], frameCounter[0], frameCounter[1], 0x00, 0x00, 0x00, i + 1]) # Composing Block A as LoRaWAN Specification
        BlockS = Crypto.encrypt(BlockA) # Get Block S in Cypher Library Format from Encpypted Block A


        if i is (numBlocks - 1) and restLength > 0:
            blockLength = restLength # Set Last Block Length if vary from 16 bytes
```

```
    for j in range(blockLength):
        decodedMessage += str(chr(message[0][counter] ^ BlockS[j])) # Get Decoded Message XORing Payload and Blocks S
        counter += 1 # Increase Count of Payload

  return decodedMessage # Return Decoded Message

def receivePacket():
  global receivedPackets
  global correctPackets

  message = readPacket() # Read Message Received (CRC Error | Correct Message)
  receivedPackets += 1 # Increase Received Packet Counter

  if message[1] > 0:
    decodedMessage = decodePacket(message) # Function call for decoding

    if statistics:
      timestamp.append(NTP.request(NTPServer).tx_time) # Latency medition

    correctPackets += 1 # Increase Correct Packet Counter

    if readRegister(REG_PACKET_SNR_VALUE) & 0x80:
      SNR = -((( 0xFF ^ readRegister(REG_PACKET_SNR_VALUE)) + 1) & 0xFF) >> 2) # Get SNR Value
    else:
      SNR = (readRegister(REG_PACKET_SNR_VALUE) & 0xFF) >> 2 # Get SNR Value

    if frequency[0] is 'LF':
      if SNR >= 0:
        RSSI = -164 + (16/15) * readRegister(REG_RSSI_VALUE) # Get RSSI from Adjusted Standard Formula
        packetRSSI =  -164 + (16/15) * readRegister(REG_PACKET_RSSI_VALUE) # Get Packet Strenght from Standard Formula
      else:
        RSSI = -164 + readRegister(REG_RSSI_VALUE) # Get RSSI from Standard Formula
        packetRSSI =  -164 + readRegister(REG_PACKET_RSSI_VALUE) + 0.25 * SNR # Get Packet Strenght from Adjusted Formula
```

```
    else:
        if SNR >= 0:
            RSSI = -157 + (16/15) * readRegister(REG_RSSI_VALUE) # Get RSSI from Adjusted Standard Formula
            packetRSSI =  -157 + (16/15) * readRegister(REG_PACKET_RSSI_VALUE) # Get Packet Strenght from Standard Formula
        else:
            RSSI = -157 + readRegister(REG_RSSI_VALUE) # Get RSSI from Standard Formula
            packetRSSI =  -157 + readRegister(REG_PACKET_RSSI_VALUE) + 0.25 * SNR # Get Packet Strenght from Adjusted Formula

    if statistics:
        midPacketRSSI.append(packetRSSI) # RSSI Packet medition
        midRSSI.append(RSSI) # RSSI Packet medition
        midSNR.append(SNR) # RSSI Packet medition

    print('-----------------------------------------------------------------')
    print('Parámetros de transmisión del paquete:\n    RSSI del paquete: ' + str(int(packetRSSI)) + '.\n RSSI: ' + str(int(RSSI)) + '.\n    SNR: ' + str(int(SNR)) + '.\n    Longitud de paquete: ' + str(message[1]) + '.\n    Mensaje: ' + decodedMessage + '.')
    print('-----------------------------------------------------------------\n')

    sendUDP('Parametros de transmision del paquete:|    RSSI del paquete: ' + str(int(packetRSSI)) + '.|    RSSI: ' + str(int(RSSI)) + '.|    SNR: ' + str(int(SNR)) + '.|    Longitud de paquete: ' + str(message[1]) + '.|    Mensaje: ' + decodedMessage + '.') # Send by UDP Correct Message Report

  else:
    print('-----------------------------------------------------------------')
    print('Parámetros de transmisión del paquete:\n    Error en Recepción: CRC.')
    print('-----------------------------------------------------------------\n')

    sendUDP('Parametros de transmision del paquete:|    Error en Recepcion: CRC.') # Send by UDP Message Error Report

#- - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - -

def sendStatus(time):
  print('-----------------------------------------------------------------')
  print ('Actualización de estado:\n    Estado: ' + status + '.\n    Actividad:  ' + str(int(time)) + 's.\n Fecha: ' + str(strftime('%d/%m/%Y')) +'.\n    Hora: ' + str(strftime('%H:%M:%S')) + '.\n    Plataforma: ' + gateway_name + '.\n    E-mail: ' + student_email + '.\n    Descripción: ' + description + '.')
```

```
    print('------------------------------------------------------------------------')

    print ('Información de paquetes:\n    Recibidos:     ' + str(receivedPackets) + '.\n    Correctos:     ' + str(correctPackets) + '.\n    Incorrectos:   ' + str(receivedPackets - correctPackets) + '.')


    sendUDP('Actualizacion de estado:|    Estado: ' + status + '.|    Actividad:  ' + str(int(time)) + 's.|    Fecha: ' + str(strftime('%d/%m/%Y')) +'.|    Hora: ' + str(strftime('%H:%M:%S')) + '.|    Plataforma: ' + gateway_name + '.|    E-mail: ' + student_email + '.|    Descripcion: ' + description + '.') # Send by UDP Status Report

    sendUDP('Informacion de paquetes:|    Recibidos:     ' + str(receivedPackets) + '.|    Correctos:     ' + str(correctPackets) + '.|    Incorrectos:   ' + str(receivedPackets - correctPackets) + '.') # Send by UDP Packet Report


    if statistics and correctPackets > 0:
        statPacketRSSI = 0
        statRSSI = 0
        statSNR = 0

        for i in range(len(midPacketRSSI)):
            statPacketRSSI += midPacketRSSI[i] # Acumulate value from Packet RSSI list
            statRSSI += midRSSI[i] # Acumulate value from RSSI list
            statSNR += midSNR[i] # Acumulate value from SNR list
        print('------------------------------------------------------------------------')
        print('Informacion de estadisticas:\n    Timestamp: ' + str(timestamp) + '.\n    RSSI de paquetes (media): ' + str(int(statPacketRSSI/len(midPacketRSSI))) + '.\n    RSSI (media): ' + str(int(statRSSI/len(midRSSI))) + '.\n    SNR (media): ' + str(int(statSNR/len(midSNR))) + '.')


        sendUDP('Informacion de estadisticas:|    Timestamp: '+ str(timestamp) + '.|    RSSI de paquetes (media): ' + str(int(statPacketRSSI/len(midPacketRSSI))) + '.|    RSSI (media): ' + str(int(statRSSI/len(midRSSI))) + '.|    SNR (media): ' + str(int(statSNR/len(midSNR))) + '.')


    print('------------------------------------------------------------------------\n')


def sendUDP(message):
    socket_udp.sendto(message.encode(), (host, port)) # Send UDP Message


#- - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - -


def setSPI():
    wiringPiSetup() # Set up WiringPi Library
    wiringPiSPISetup(SPIConfig['spi_channel'], SPIConfig['spi_frequency']) # Set Up Serial Peripherical Interface (SPI)
```

```
    pinMode(pines['NReset'], OUTPUT) # Set Reset Pin as Output
    pinMode(pines['NSS'], OUTPUT) # Set Pin Chip Select as Output
    pinMode(pines['dio0'], INPUT) # Set Pin Dio0 as Input

    digitalWrite(pines['NReset'], LOW) # Low NReset for Manual Reset of the Chip
    sleep(0.01) # 10 ms Wait in Reset Process
    digitalWrite(pines['NReset'], HIGH) # High NReset for Manual Reset of the Chip
    sleep(0.01) # 10 ms Wait in Reset Process

def setMode(mode):
    if frequency[0] is 'LF':
        newMode = 0x80 | 0x08 # LoRa LF Mode
    else:
        newMode = 0x80 # LoRa HF Mode

    newMode |= mode # Compose New Operation Mode
    writeRegister(REG_OP_MODE, newMode) # Set New Operation Mode

def setFrequency(frequency):
    frf = int((frequency[1] << 19) / 32000000) # FRF Stablished
    MSB = frf >> 16
    MID = frf >> 8 ^ (MSB << 8)
    LSB = frf ^ (MSB << 16 ^ MID << 8)
    writeRegister(REG_FRF_MSB, MSB) # Set RFR MSB Register
    writeRegister(REG_FRF_MID, MID) # Set FRF MID Register
    writeRegister(REG_FRF_LSB, LSB) # Set FRF LSB Register

def setRate(sf):
    writeRegister(REG_MODEM_CONFIG_1, 0x72) # BW 125 | CRC 4/5
    writeRegister(REG_MODEM_CONFIG_2, sf[1] << 4 | 0x04) #SF | CRC ON

    if sf[0] is 'SF11' or sf[0] is 'SF12':
        writeRegister(REG_MODEM_CONFIG_3, 0x0C) # Symbol Length Exceed 16 ms | LNA internal AGC
Loop
    else:
        writeRegister(REG_MODEM_CONFIG_3, 0x04) # LNA internal AGC Loop
```

```
    if sf[0] is 'SF10' or sf[0] is 'SF11' or sf[0] is 'SF12':
        writeRegister(REG_SYMB_TIMEOUT_LSB, 0x05) # Length Receiver Window in Symbols (SF10 - SF12)
    else:
        writeRegister(REG_SYMB_TIMEOUT_LSB, 0x08) # Length Receiver Window in Symbols (SF7 - SF9)

#------------------------------------------------------------------------------------------

def setLoRaModem():
    setSPI() # Set up SPI
    setMode(SLEEP_MODE) # Set LoRa in Sleep Mode
    setFrequency(frequency) # Set Frequency Configuration
    setRate(sf) # Set SF and CRC Configuration

    if frequency[0] is 'LF':
        writeRegister(REG_LNA, 0x20) # Maximum Gain | LNA LF | Default LNA Current
    else:
        writeRegister(REG_LNA, 0x23) # Maximum Gain | LNA HF | 150% LNA Current

    writeRegister(REG_SYNC_WORD, 0x34) # Set LoRa Synchronization Word
    writeRegister(REG_INVERT_IQ, 0x27) # Set up default InvertIQ
    writeRegister(REG_PAYLOAD_LENGTH, 0x40) # Set up Payload Length | Packet Format Variable = Máx. Length
    writeRegister(REG_MAX_PAYLOAD_LENGTH, 0xFF) # Set up Máx. Payload Length (255 Bytes)
    writeRegister(REG_FIFO_ADDR_PTR, readRegister(REG_FIFO_RX_BASE_ADDR)) # Read FIFO through SPI for RX Demodulator
    writeRegister(REG_HOP_PERIOD, 0x00) # Hopping Period Disabled
    writeRegister(REG_PA_RAMP, (readRegister(REG_PA_RAMP) & 0xF0) | 0x08) # Set Power Amplifier Ramp to 50 us
    writeRegister(REG_IRQ_FLAGS, 0xFF) # Clear all IRQ Flags

def rxLoRaModem():
    setMode(SLEEP_MODE) # Set LoRa in Sleep Mode
    setFrequency(frequency) # Set Frequency Configuration
    setRate(sf) # Set SF and CRC Configuration

    writeRegister(REG_IRQ_FLAGS, 0xFF) # Clear all IRQ Flags
    writeRegister(REG_IRQ_FLAGS_MASK, 0xFF ^ (RX_DONE_MASK)) # Mask to Accept Only RX Interruptions
    setMode(RX_CONTINUOUS_MODE) # Set LoRa in RX Continuous Mode
```

```
def setupLoRa():
    global status

    setLoRaModem() # Set LoRa Initial Configuration
    status = listStatus[1] # Set LoRa RX Status
    rxLoRaModem() # Set LoRa RX Configuration

#--------------------------------------------------------------------------------

if __name__ == '__main__':
    print('-------------------------------------------------------------------\n')
    print('          Gateway LoRaWAN - Grupo Integrado de Ingeniería.\n')
    print('-------------------------------------------------------------------\n')

    print('-------------------------------------------------------------------')
    print('Iniciando Gateway ...\nGateway ID: ' + str(':'.join(findall('..', '%02x' % getnode())).upper()) +
'.\nEscuchando a frecuencia: ' + str(frequency[1]/1000000) + ' MHz.\nSpreading Factor: ' + str(sf[0]) +
'.')
    print('-------------------------------------------------------------------\n')

    socket_udp = socket(AF_INET, SOCK_DGRAM) # Set Socket for UDP Transmissions to Server

    if statistics:
        NTP = NTPClient() # Set NTP Server for Latency Statistics

    setupLoRa() # Set up LoRa Configuration

    sendUDP('         Gateway LoRaWAN - Grupo Integrado de Ingenieria.') # Send by UDP Gateway
Header
    sendUDP('Iniciando Gateway ...|Gateway ID: ' + str(':'.join(findall('..', '%02x' % getnode())).upper()) +
'.|Escuchando a frecuencia: ' + str(frequency[1]/1000000) + ' MHz.|Spreading Factor: ' + str(sf[0]) + '.')
#Send by UDP Initial Status and Configuration

    firstTime = time() # Set first time reference for Activity
    pastTime = firstTime # Set past time reference for Status Update

    while(1):
        try:
            eventHandler() # Set LoRa funcion for Incoming Events
```

```
        nowTime = time() # Set actual time for Status Update

        if (nowTime - pastTime) >= updateInterval:
            pastTime = nowTime # Refresh time reference
            sendStatus(nowTime - firstTime) # Send Status Update

    except KeyboardInterrupt:
        sys.exit()
```

APLICACIONES DESARROLLADAS PARA LOS END-NODES

Envío de Mensajes en Bucle Finito – Frecuencia 433 MHz (DR5)

```
# encoding: utf-8

'''
        Proyecto: Diseño y Desarrollo del Sistema de Comunicaciones de un Enjambre
        de UAV's Colaborativos.

        Integración: Desarrollo de End-Node LoRaWAN.
        Lenguaje de Programación: Python 3.

        Alumno: Cristian Méndez Sanmartín.
        Titulación: Máster en Ingeniería Industrial.

        Grupo de Investigación: Grupo Integrado de Ingeniería.
        Universidad: Universidade da Coruña.

        Características:
        - Modelo para Pruebas.
        - Tiempos Configurables.
        - Número de Mensajes Configurable.

'''

import sys
from ntplib import NTPClient
from serial import Serial, EIGHTBITS, PARITY_NONE, STOPBITS_ONE
from time import time, sleep

if __name__ == '__main__':
    #   1. Configuración de Tiempos y Repeticiones
    commandPause = 0.10
    savePause = 2.00
    messagePause = 2.50
    messages = 100
```

```
#   2. Configuración de Puerto Serie
mySerialPort = Serial('/dev/ttyACM0',
            baudrate = 57600,
            bytesize = EIGHTBITS,
            parity = PARITY_NONE,
            stopbits = STOPBITS_ONE,
            timeout = 0.010
            )

mySerialPort.flushInput()
mySerialPort.flushOutput()

#   3. Configuración del Servidor NTP
statistics = False
NTPServer = 'es.pool.ntp.org'
timestamp = []

if statistics:
      NTP = NTPClient()

#- - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - -

print('-----------------------------------------------------------------\n')
print('                Iniciando End-Node LoRaWAN.\n                 ')
print('-----------------------------------------------------------------\n')

print('-----------------------------------------------------------------\n')
if mySerialPort.isOpen():
      print('Comunicación establecida con ' + mySerialPort.name)
print ('Enviando Comandos de Setup ...')
print ('-----------------------------------------------------------------\n')

 #- - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - -
 try:
      print('Comando enviado: mac reset 433')
      mySerialPort.write('mac reset 433'.encode('ascii') + '\r\n'.encode('ascii'))
      sleep(commandPause)
```

```
            print('Respuesta: ' + str(mySerialPort.read(15)).lstrip ("b'").replace ("'", "").replace("\\r\\n", "
"))

            print('Comando enviado: mac set ch status 1 off')
            mySerialPort.write('mac set ch status 1 off'.encode('ascii') + '\r\n'.encode('ascii'))
            sleep(commandPause)
            print('Respuesta: ' + str(mySerialPort.read(15)).lstrip ("b'").replace ("'", "").replace("\\r\\n", "
"))

           print('Comando enviado: mac set ch status 2 off')
            mySerialPort.write('mac set ch status 2 off'.encode('ascii') + '\r\n'.encode('ascii'))
            sleep(commandPause)
            print('Respuesta: ' + str(mySerialPort.read(15)).lstrip ("b'").replace ("'", "").replace("\\r\\n", "
"))

            print('Comando enviado: mac set ch dcycle 0 9')
            mySerialPort.write('mac set ch dcycle 0 9'.encode('ascii') + '\r\n'.encode('ascii'))
            sleep(commandPause)
            print('Respuesta: ' + str(mySerialPort.read(15)).lstrip ("b'").replace ("'", "").replace("\\r\\n", "
"))

            print('Comando enviado: mac set dr 5')
            mySerialPort.write('mac set dr 5'.encode('ascii') + '\r\n'.encode('ascii'))
            sleep(commandPause)
            print('Respuesta: ' + str(mySerialPort.read(15)).lstrip ("b'").replace ("'", "").replace("\\r\\n", "
"))

            print('Comando enviado: mac set pwridx 0')
            mySerialPort.write('mac set pwridx 0'.encode('ascii') + '\r\n'.encode('ascii'))
            sleep(commandPause)
            print('Respuesta: ' + str(mySerialPort.read(15)).lstrip ("b'").replace ("'", "").replace("\\r\\n", "
"))

            print('Comando enviado: mac set devaddr 26011CA2')
            mySerialPort.write('mac set devaddr 26011CA2'.encode('ascii') + '\r\n'.encode('ascii'))
            sleep(commandPause)
            print('Respuesta: ' + str(mySerialPort.read(15)).lstrip ("b'").replace ("'", "").replace("\\r\\n", "
"))
```

```
            print('Comando enviado: mac set appskey A4109C150E2492D5479A782645758B3B')
            mySerialPort.write('mac set appskey A4109C150E2492D5479A782645758B3B'.encode('ascii')
+ '\r\n'.encode('ascii'))
            sleep(commandPause)
            print('Respuesta: ' + str(mySerialPort.read(15)).lstrip ("b'").replace ("'", "").replace("\\r\\n", "
"))


            print('Comando enviado: mac set nwkskey 437009D0F91CF5F9694E4EB2AE7EB9D3')
            mySerialPort.write('mac set nwkskey 437009D0F91CF5F9694E4EB2AE7EB9D3'.encode('ascii')
+ '\r\n'.encode('ascii'))
            sleep(commandPause)
            print('Respuesta: ' + str(mySerialPort.read(15)).lstrip ("b'").replace ("'", "").replace("\\r\\n", "
"))


            print('Comando enviado: mac save')
            mySerialPort.write('mac save'.encode('ascii') + '\r\n'.encode('ascii'))
            sleep(savePause)
            print('Respuesta: ' + str(mySerialPort.read(15)).lstrip ("b'").replace ("'", "").replace("\\r\\n", "
"))


            print('Comando enviado: mac join ABP')
            mySerialPort.write('mac join ABP'.encode('ascii') + '\r\n'.encode('ascii'))
            sleep(savePause)
            print('Respuesta: ' + str(mySerialPort.read(15)).lstrip ("b'").replace ("'", "").replace("\\r\\n", "
"))


            print('\n----------------------------------------------------------------------')
            print('Iniciando Bucle de Envío de Mensajes ...                    ')
            print('----------------------------------------------------------------------\n')


            init = time() # Comenzamos a contar el tiempo de envío
            for i in range(messages):
                  print('Comando enviado: mac tx uncnf 1
2753692063726565732071756520707565646573 2c20796120657374617320612 06d6564696f2063616
d696e6f2e27')


                  if statistics:
                        timestamp.append(NTP.request(NTPServer).tx_time)
```

```
                mySerialPort.write('mac tx uncnf 1
27536920637265657320717565207075656465732c207961206573746173206120 6d6564696f2063616
d696e6f2e27'.encode('ascii') + '\r\n'.encode('ascii'))
                sleep(messagePause)
                print('Respuesta: ' + str(mySerialPort.read(15)).lstrip ("b'").replace ("'",
"").replace("\\r\\n", " "))


        print('\n----------------------------------------------------------------------')


        if statistics:
            print('Timestamp: ' + str(timestamp))
            print('----------------------------------------------------------------------')


        print('Tiempo de Envio Total: ' + str(round(time() - init, 2)) + ' s.')
        print('----------------------------------------------------------------------')
        sleep(180)


    except KeyboardInterrupt:
        mySerialPort.close() # Cerramos el Puerto Serie
        sys.exit() # Salida de Programa


    mySerialPort.close() # Cerramos el Puerto Serie
```

Envío de Mensajes en Bucle Infinito – Frecuencia 433 MHz (DR5)

```
# encoding: utf-8

'''
        Proyecto: Diseño y Desarrollo del Sistema de Comunicaciones de un Enjambre
        de UAV's Colaborativos.

        Integración: Desarrollo de End-Node LoRaWAN.
        Lenguaje de Programación: Python 3.

        Alumno: Cristian Méndez Sanmartín.
        Titulación: Máster en Ingeniería Industrial.

        Grupo de Investigación: Grupo Integrado de Ingeniería.
        Universidad: Universidade da Coruña.

        Características:
        - Modelo Final.
        - Tiempos Configurables.
        - Pendiente bucle de lectura de sensores.

'''

import sys
from ntplib import NTPClient
from serial import Serial, EIGHTBITS, PARITY_NONE, STOPBITS_ONE
from time import time, sleep

if __name__ == '__main__':
    #   1. Configuración de Tiempos
    commandPause = 0.10
    savePause = 2.00
    messagePause = 2.50
```

```
#   2. Configuración de Puerto Serie
mySerialPort = Serial('/dev/ttyACM0',
            baudrate = 57600,
            bytesize = EIGHTBITS,
            parity = PARITY_NONE,
            stopbits = STOPBITS_ONE,
            timeout = 0.010
            )

mySerialPort.flushInput()
mySerialPort.flushOutput()

#   3. Configuración del Servidor NTP
statistics = False
NTPServer = 'es.pool.ntp.org'
timestamp = []

if statistics:
      NTP = NTPClient()

#- - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - -

print('-----------------------------------------------------------------\n')
print('                Iniciando End-Node LoRaWAN.\n                 ')
print('-----------------------------------------------------------------\n')

print('-----------------------------------------------------------------\n')
if mySerialPort.isOpen():
      print('Comunicación establecida con ' + mySerialPort.name)
print ('Enviando Comandos de Setup ...')
print ('-----------------------------------------------------------------\n')

 #- - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - -
 try:
      print('Comando enviado: mac reset 433')
      mySerialPort.write('mac reset 433'.encode('ascii') + '\r\n'.encode('ascii'))
      sleep(commandPause)
```

```
            print('Respuesta: ' + str(mySerialPort.read(15)).lstrip ("b'").replace ("'", "").replace("\\r\\n", "
"))

            print('Comando enviado: mac set ch status 1 off')
            mySerialPort.write('mac set ch status 1 off'.encode('ascii') + '\r\n'.encode('ascii'))
            sleep(commandPause)
            print('Respuesta: ' + str(mySerialPort.read(15)).lstrip ("b'").replace ("'", "").replace("\\r\\n", "
"))

           print('Comando enviado: mac set ch status 2 off')
            mySerialPort.write('mac set ch status 2 off'.encode('ascii') + '\r\n'.encode('ascii'))
            sleep(commandPause)
            print('Respuesta: ' + str(mySerialPort.read(15)).lstrip ("b'").replace ("'", "").replace("\\r\\n", "
"))

            print('Comando enviado: mac set ch dcycle 0 9')
            mySerialPort.write('mac set ch dcycle 0 9'.encode('ascii') + '\r\n'.encode('ascii'))
            sleep(commandPause)
            print('Respuesta: ' + str(mySerialPort.read(15)).lstrip ("b'").replace ("'", "").replace("\\r\\n", "
"))

            print('Comando enviado: mac set dr 5')
            mySerialPort.write('mac set dr 5'.encode('ascii') + '\r\n'.encode('ascii'))
            sleep(commandPause)
            print('Respuesta: ' + str(mySerialPort.read(15)).lstrip ("b'").replace ("'", "").replace("\\r\\n", "
"))

            print('Comando enviado: mac set pwridx 0')
            mySerialPort.write('mac set pwridx 0'.encode('ascii') + '\r\n'.encode('ascii'))
            sleep(commandPause)
            print('Respuesta: ' + str(mySerialPort.read(15)).lstrip ("b'").replace ("'", "").replace("\\r\\n", "
"))

            print('Comando enviado: mac set devaddr 26011CA2')
            mySerialPort.write('mac set devaddr 26011CA2'.encode('ascii') + '\r\n'.encode('ascii'))
            sleep(commandPause)
            print('Respuesta: ' + str(mySerialPort.read(15)).lstrip ("b'").replace ("'", "").replace("\\r\\n", "
"))
```

```
            print('Comando enviado: mac set appskey A4109C150E2492D5479A782645758B3B')
            mySerialPort.write('mac set appskey A4109C150E2492D5479A782645758B3B'.encode('ascii')
+ '\r\n'.encode('ascii'))
            sleep(commandPause)
            print('Respuesta: ' + str(mySerialPort.read(15)).lstrip ("b'").replace ("'", "").replace("\\r\\n", "
"))


            print('Comando enviado: mac set nwkskey 437009D0F91CF5F9694E4EB2AE7EB9D3')
            mySerialPort.write('mac set nwkskey 437009D0F91CF5F9694E4EB2AE7EB9D3'.encode('ascii')
+ '\r\n'.encode('ascii'))
            sleep(commandPause)
            print('Respuesta: ' + str(mySerialPort.read(15)).lstrip ("b'").replace ("'", "").replace("\\r\\n", "
"))


            print('Comando enviado: mac save')
            mySerialPort.write('mac save'.encode('ascii') + '\r\n'.encode('ascii'))
            sleep(savePause)
            print('Respuesta: ' + str(mySerialPort.read(15)).lstrip ("b'").replace ("'", "").replace("\\r\\n", "
"))


            print('Comando enviado: mac join ABP')
            mySerialPort.write('mac join ABP'.encode('ascii') + '\r\n'.encode('ascii'))
            sleep(savePause)
            print('Respuesta: ' + str(mySerialPort.read(15)).lstrip ("b'").replace ("'", "").replace("\\r\\n", "
"))


            print('\n------------------------------------------------------------------------')
            print('Iniciando Bucle de Envío de Mensajes ...                    ')
            print('------------------------------------------------------------------------\n')


            init = time() # Comenzamos a contar el tiempo de envío
            try:
                  while 1:
                        print('Comando enviado: mac tx uncnf 1
2753692063726565732071756520707565646573 2c20796120657374617320612 06d6564696f2063616d696e6f2e27')


                        if statistics:
                              timestamp.append(NTP.request(NTPServer).tx_time)
```

```
                    mySerialPort.write('mac tx uncnf 1
2753692063726565732071756520707565646573 2c20796120657374617320612 06d6564696f2063616
d696e6f2e27'.encode('ascii') + '\r\n'.encode('ascii'))
                    sleep(messagePause)
                    print('Respuesta: ' + str(mySerialPort.read(15)).lstrip ("b'").replace ("'",
"").replace("\\r\\n", " "))

        except KeyboardInterrupt:
            pass

        print('\n----------------------------------------------------------------------')

        if statistics:
            print('Timestamp: ' + str(timestamp))
            print('----------------------------------------------------------------------')

        print('Tiempo de Envio Total: ' + str(round(time() - init, 2)) + ' s.')
        print('----------------------------------------------------------------------')
        sleep(180)

    except KeyboardInterrupt:
        mySerialPort.close() # Cerramos el Puerto Serie
        sys.exit() # Salida de Programa

    mySerialPort.close() # Cerramos el Puerto Serie
```

Envío de Mensajes en Bucle Finito – Frecuencia 868 MHz (DR5)

```
# encoding: utf-8

'''
        Proyecto: Diseño y Desarrollo del Sistema de Comunicaciones de un Enjambre
        de UAV's Colaborativos.

        Integración: Desarrollo de End-Node LoRaWAN.
        Lenguaje de Programación: Python 3.

        Alumno: Cristian Méndez Sanmartín.
        Titulación: Máster en Ingeniería Industrial.

        Grupo de Investigación: Grupo Integrado de Ingeniería.
        Universidad: Universidade da Coruña.

        Características:
        - Modelo para Pruebas.
        - Tiempos Configurables.
        - Número de Mensajes Configurable.

'''

import sys
from ntplib import NTPClient
from serial import Serial, EIGHTBITS, PARITY_NONE, STOPBITS_ONE
from time import time, sleep

if __name__ == '__main__':
    #   1. Configuración de Tiempos y Repeticiones
    commandPause = 0.10
    savePause = 2.00
    messagePause = 2.50
    messages = 100
```

```
#   2. Configuración de Puerto Serie
mySerialPort = Serial('/dev/ttyACM0',
            baudrate = 57600,
            bytesize = EIGHTBITS,
            parity = PARITY_NONE,
            stopbits = STOPBITS_ONE,
            timeout = 0.010
            )

mySerialPort.flushInput()
mySerialPort.flushOutput()

#   3. Configuración del Servidor NTP
statistics = False
NTPServer = 'es.pool.ntp.org'
timestamp = []

if statistics:
      NTP = NTPClient()

#- - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - -

print('-----------------------------------------------------------------\n')
print('                Iniciando End-Node LoRaWAN.\n                 ')
print('-----------------------------------------------------------------\n')

print('-----------------------------------------------------------------\n')
if mySerialPort.isOpen():
      print('Comunicación establecida con ' + mySerialPort.name)
print ('Enviando Comandos de Setup ...')
print ('-----------------------------------------------------------------\n')

 #- - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - -
 try:
      print('Comando enviado: mac reset 868')
      mySerialPort.write('mac reset 868'.encode('ascii') + '\r\n'.encode('ascii'))
      sleep(commandPause)
```

```
            print('Respuesta: ' + str(mySerialPort.read(15)).lstrip ("b'").replace ("'", "").replace("\\r\\n", "
"))

            print('Comando enviado: mac set ch status 0 off')
            mySerialPort.write('mac set ch status 0 off'.encode('ascii') + '\r\n'.encode('ascii'))
            sleep(commandPause)
            print('Respuesta: ' + str(mySerialPort.read(15)).lstrip ("b'").replace ("'", "").replace("\\r\\n", "
"))

            print('Comando enviado: mac set ch status 1 off')
            mySerialPort.write('mac set ch status 1 off'.encode('ascii') + '\r\n'.encode('ascii'))
            sleep(commandPause)
            print('Respuesta: ' + str(mySerialPort.read(15)).lstrip ("b'").replace ("'", "").replace("\\r\\n", "
"))

           print('Comando enviado: mac set ch status 2 off')
            mySerialPort.write('mac set ch status 2 off'.encode('ascii') + '\r\n'.encode('ascii'))
            sleep(commandPause)
            print('Respuesta: ' + str(mySerialPort.read(15)).lstrip ("b'").replace ("'", "").replace("\\r\\n", "
"))

            print('Comando enviado: mac set ch freq 3 869525000')
            mySerialPort.write('mac set ch freq 3 869525000'.encode('ascii') + '\r\n'.encode('ascii'))
            sleep(commandPause)
            print('Respuesta: ' + str(mySerialPort.read(15)).lstrip ("b'").replace ("'", "").replace("\\r\\n", "
"))

            print('Comando enviado: mac set ch dcycle 3 9')
            mySerialPort.write('mac set ch dcycle 3 9'.encode('ascii') + '\r\n'.encode('ascii'))
            sleep(commandPause)
            print('Respuesta: ' + str(mySerialPort.read(15)).lstrip ("b'").replace ("'", "").replace("\\r\\n", "
"))

            print('Comando enviado: mac set ch drrange 3 0 5')
            mySerialPort.write('mac set ch drrange 3 0 5'.encode('ascii') + '\r\n'.encode('ascii'))
            sleep(commandPause)
            print('Respuesta: ' + str(mySerialPort.read(15)).lstrip ("b'").replace ("'", "").replace("\\r\\n", "
"))
```

```
            print('Comando enviado: mac set ch status 3 on')
            mySerialPort.write('mac set ch status 3 on'.encode('ascii') + '\r\n'.encode('ascii'))
            sleep(commandPause)
            print('Respuesta: ' + str(mySerialPort.read(15)).lstrip ("b'").replace ("'", "").replace("\\r\\n", "
"))


            print('Comando enviado: mac set dr 5')
            mySerialPort.write('mac set dr 5'.encode('ascii') + '\r\n'.encode('ascii'))
            sleep(commandPause)
            print('Respuesta: ' + str(mySerialPort.read(15)).lstrip ("b'").replace ("'", "").replace("\\r\\n", "
"))


            print('Comando enviado: mac set pwridx 1')
            mySerialPort.write('mac set pwridx 1'.encode('ascii') + '\r\n'.encode('ascii'))
            sleep(commandPause)
            print('Respuesta: ' + str(mySerialPort.read(15)).lstrip ("b'").replace ("'", "").replace("\\r\\n", "
"))


            print('Comando enviado: mac set devaddr 26011CA2')
            mySerialPort.write('mac set devaddr 26011CA2'.encode('ascii') + '\r\n'.encode('ascii'))
            sleep(commandPause)
            print('Respuesta: ' + str(mySerialPort.read(15)).lstrip ("b'").replace ("'", "").replace("\\r\\n", "
"))


            print('Comando enviado: mac set appskey A4109C150E2492D5479A782645758B3B')
            mySerialPort.write('mac set appskey A4109C150E2492D5479A782645758B3B'.encode('ascii')
+ '\r\n'.encode('ascii'))
            sleep(commandPause)
            print('Respuesta: ' + str(mySerialPort.read(15)).lstrip ("b'").replace ("'", "").replace("\\r\\n", "
"))


            print('Comando enviado: mac set nwkskey 437009D0F91CF5F9694E4EB2AE7EB9D3')
            mySerialPort.write('mac set nwkskey 437009D0F91CF5F9694E4EB2AE7EB9D3'.encode('ascii')
+ '\r\n'.encode('ascii'))
            sleep(commandPause)
            print('Respuesta: ' + str(mySerialPort.read(15)).lstrip ("b'").replace ("'", "").replace("\\r\\n", "
"))


            print('Comando enviado: mac save')
            mySerialPort.write('mac save'.encode('ascii') + '\r\n'.encode('ascii'))
            sleep(savePause)
```

```
        print('Respuesta: ' + str(mySerialPort.read(15)).lstrip ("b'").replace ("'", "").replace("\\r\\n", "
"))


        print('Comando enviado: mac join ABP')
        mySerialPort.write('mac join ABP'.encode('ascii') + '\r\n'.encode('ascii'))
        sleep(savePause)
        print('Respuesta: ' + str(mySerialPort.read(15)).lstrip ("b'").replace ("'", "").replace("\\r\\n", "
"))


        print('\n------------------------------------------------------------------------')
        print('Iniciando Bucle de Envío de Mensajes ...                    ')
        print('------------------------------------------------------------------------\n')


        init = time() # Comenzamos a contar el tiempo de envío
        for i in range(messages):
            print('Comando enviado: mac tx uncnf 1
2753692063726565732071756520707565646573 2c207961206573746173206 1206d6564696f2063616d696e6f2e27')


            if statistics:
                timestamp.append(NTP.request(NTPServer).tx_time)


            mySerialPort.write('mac tx uncnf 1
2753692063726565732071756520707565646573 2c207961206573746173206 1206d6564696f2063616d696e6f2e27'.encode('ascii') + '\r\n'.encode('ascii'))
            sleep(messagePause)
            print('Respuesta: ' + str(mySerialPort.read(15)).lstrip ("b'").replace ("'",
"").replace("\\r\\n", " "))


        print('\n------------------------------------------------------------------------')


        if statistics:
            print('Timestamp: ' + str(timestamp))
            print('------------------------------------------------------------------------')


        print('Tiempo de Envio Total: ' + str(round(time() - init, 2)) + ' s.')
        print('------------------------------------------------------------------------')
        sleep(180)
```

```
except KeyboardInterrupt:
    mySerialPort.close() # Cerramos el Puerto Serie
    sys.exit() # Salida de Programa

mySerialPort.close() # Cerramos el Puerto Serie
```

Envío de Mensajes en Bucle Infinito – Frecuencia 868 MHz (DR5)

```
# encoding: utf-8

'''
        Proyecto: Diseño y Desarrollo del Sistema de Comunicaciones de un Enjambre
        de UAV's Colaborativos.

        Integración: Desarrollo de End-Node LoRaWAN.
        Lenguaje de Programación: Python 3.

        Alumno: Cristian Méndez Sanmartín.
        Titulación: Máster en Ingeniería Industrial.

        Grupo de Investigación: Grupo Integrado de Ingeniería.
        Universidad: Universidade da Coruña.

        Características:
        - Modelo Final.
        - Tiempos Configurables.
        - Pendiente bucle de lectura de sensores.

'''

import sys
from ntplib import NTPClient
from serial import Serial, EIGHTBITS, PARITY_NONE, STOPBITS_ONE
from time import time, sleep

if __name__ == '__main__':
    #   1. Configuración de Tiempos
    commandPause = 0.10
    savePause = 2.00
    messagePause = 2.50
```

```
#   2. Configuración de Puerto Serie
mySerialPort = Serial('/dev/ttyACM0',
            baudrate = 57600,
            bytesize = EIGHTBITS,
            parity = PARITY_NONE,
            stopbits = STOPBITS_ONE,
            timeout = 0.010
            )

mySerialPort.flushInput()
mySerialPort.flushOutput()

#   3. Configuración del Servidor NTP
statistics = False
NTPServer = 'es.pool.ntp.org'
timestamp = []

if statistics:
      NTP = NTPClient()

#- - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - -

print('-------------------------------------------------------------------\n')
print('                Iniciando End-Node LoRaWAN.\n                ')
print('-------------------------------------------------------------------\n')

print('-------------------------------------------------------------------\n')
if mySerialPort.isOpen():
      print('Comunicación establecida con ' + mySerialPort.name)
print ('Enviando Comandos de Setup ...')
print ('-------------------------------------------------------------------\n')

 #- - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - -
 try:
      print('Comando enviado: mac reset 868')
      mySerialPort.write('mac reset 868'.encode('ascii') + '\r\n'.encode('ascii'))
      sleep(commandPause)
```

```
        print('Respuesta: ' + str(mySerialPort.read(15)).lstrip ("b'").replace ("'", "").replace("\\r\\n", "
"))

        print('Comando enviado: mac set ch status 0 off')
        mySerialPort.write('mac set ch status 0 off'.encode('ascii') + '\r\n'.encode('ascii'))
        sleep(commandPause)
        print('Respuesta: ' + str(mySerialPort.read(15)).lstrip ("b'").replace ("'", "").replace("\\r\\n", "
"))

        print('Comando enviado: mac set ch status 1 off')
        mySerialPort.write('mac set ch status 1 off'.encode('ascii') + '\r\n'.encode('ascii'))
        sleep(commandPause)
        print('Respuesta: ' + str(mySerialPort.read(15)).lstrip ("b'").replace ("'", "").replace("\\r\\n", "
"))

       print('Comando enviado: mac set ch status 2 off')
        mySerialPort.write('mac set ch status 2 off'.encode('ascii') + '\r\n'.encode('ascii'))
        sleep(commandPause)
        print('Respuesta: ' + str(mySerialPort.read(15)).lstrip ("b'").replace ("'", "").replace("\\r\\n", "
"))

        print('Comando enviado: mac set ch freq 3 869525000')
        mySerialPort.write('mac set ch freq 3 869525000'.encode('ascii') + '\r\n'.encode('ascii'))
        sleep(commandPause)
        print('Respuesta: ' + str(mySerialPort.read(15)).lstrip ("b'").replace ("'", "").replace("\\r\\n", "
"))

        print('Comando enviado: mac set ch dcycle 3 9')
        mySerialPort.write('mac set ch dcycle 3 9'.encode('ascii') + '\r\n'.encode('ascii'))
        sleep(commandPause)
        print('Respuesta: ' + str(mySerialPort.read(15)).lstrip ("b'").replace ("'", "").replace("\\r\\n", "
"))

        print('Comando enviado: mac set ch drrange 3 0 5')
        mySerialPort.write('mac set ch drrange 3 0 5'.encode('ascii') + '\r\n'.encode('ascii'))
        sleep(commandPause)
        print('Respuesta: ' + str(mySerialPort.read(15)).lstrip ("b'").replace ("'", "").replace("\\r\\n", "
"))
```

```
            print('Comando enviado: mac set ch status 3 on')
            mySerialPort.write('mac set ch status 3 on'.encode('ascii') + '\r\n'.encode('ascii'))
            sleep(commandPause)
            print('Respuesta: ' + str(mySerialPort.read(15)).lstrip ("b'").replace ("'", "").replace("\\r\\n", "
"))


            print('Comando enviado: mac set dr 5')
            mySerialPort.write('mac set dr 5'.encode('ascii') + '\r\n'.encode('ascii'))
            sleep(commandPause)
            print('Respuesta: ' + str(mySerialPort.read(15)).lstrip ("b'").replace ("'", "").replace("\\r\\n", "
"))


            print('Comando enviado: mac set pwridx 1')
            mySerialPort.write('mac set pwridx 1'.encode('ascii') + '\r\n'.encode('ascii'))
            sleep(commandPause)
            print('Respuesta: ' + str(mySerialPort.read(15)).lstrip ("b'").replace ("'", "").replace("\\r\\n", "
"))


            print('Comando enviado: mac set devaddr 26011CA2')
            mySerialPort.write('mac set devaddr 26011CA2'.encode('ascii') + '\r\n'.encode('ascii'))
            sleep(commandPause)
            print('Respuesta: ' + str(mySerialPort.read(15)).lstrip ("b'").replace ("'", "").replace("\\r\\n", "
"))


            print('Comando enviado: mac set appskey A4109C150E2492D5479A782645758B3B')
            mySerialPort.write('mac set appskey A4109C150E2492D5479A782645758B3B'.encode('ascii')
+ '\r\n'.encode('ascii'))
            sleep(commandPause)
            print('Respuesta: ' + str(mySerialPort.read(15)).lstrip ("b'").replace ("'", "").replace("\\r\\n", "
"))


            print('Comando enviado: mac set nwkskey 437009D0F91CF5F9694E4EB2AE7EB9D3')
            mySerialPort.write('mac set nwkskey 437009D0F91CF5F9694E4EB2AE7EB9D3'.encode('ascii')
+ '\r\n'.encode('ascii'))
            sleep(commandPause)
            print('Respuesta: ' + str(mySerialPort.read(15)).lstrip ("b'").replace ("'", "").replace("\\r\\n", "
"))


            print('Comando enviado: mac save')
            mySerialPort.write('mac save'.encode('ascii') + '\r\n'.encode('ascii'))
            sleep(savePause)
```

```
        print('Respuesta: ' + str(mySerialPort.read(15)).lstrip ("b'").replace ("'", "").replace("\\r\\n", "
"))


        print('Comando enviado: mac join ABP')
        mySerialPort.write('mac join ABP'.encode('ascii') + '\r\n'.encode('ascii'))
        sleep(savePause)
        print('Respuesta: ' + str(mySerialPort.read(15)).lstrip ("b'").replace ("'", "").replace("\\r\\n", "
"))


        print('\n-----------------------------------------------------------------------')
        print('Iniciando Bucle de Envío de Mensajes ...                    ')
        print('-----------------------------------------------------------------------\n')


        init = time() # Comenzamos a contar el tiempo de envío
        try:
            while 1:
                print('Comando enviado: mac tx uncnf 1
2753692063726565732071756520707565646573 2c20796120657374617320612 06d6564696f2063616d696e6f2e27')


                if statistics:
                    timestamp.append(NTP.request(NTPServer).tx_time)


                mySerialPort.write('mac tx uncnf 1
2753692063726565732071756520707565646573 2c20796120657374617320612 06d6564696f2063616d696e6f2e27'.encode('ascii') + '\r\n'.encode('ascii'))
                sleep(messagePause)
                print('Respuesta: ' + str(mySerialPort.read(15)).lstrip ("b'").replace ("'",
"").replace("\\r\\n", " "))


        except KeyboardInterrupt:
            pass


        print('\n-----------------------------------------------------------------------')


        if statistics:
            print('Timestamp: ' + str(timestamp))
            print('-----------------------------------------------------------------------')
```

```
        print('Tiempo de Envio Total: ' + str(round(time() - init, 2)) + ' s.')
        print('-----------------------------------------------------------------')
        sleep(180)

    except KeyboardInterrupt:
        mySerialPort.close() # Cerramos el Puerto Serie
        sys.exit() # Salida de Programa

    mySerialPort.close() # Cerramos el Puerto Serie
```

APLICACIÓN DESARROLLADA PARA EL SERVIDOR DE MONITORIZACIÓN REMOTA

```
# encoding: utf-8

'''

    Proyecto: Diseño y Desarrollo del Sistema de Comunicaciones de un Enjambre
    de UAV's Colaborativos.

    Integración: Desarrollo de Servidor LoRa - Python.
    Lenguaje de Programación: Python 3.

    Alumno: Cristian Méndez Sanmartín.
    Titulación: Máster en Ingeniería Industrial.

    Grupo de Investigación: Grupo Integrado de Ingeniería.
    Universidad: Universidade da Coruña.

'''

import sys
from socket import socket, AF_INET, SOCK_DGRAM

if __name__ == '__main__':
    # Definimos el Host y el Puerto
    host = '192.168.0.3'
    port = 1700

    # Instanciamos un Objeto para trabajar con el Socket Server
    socket_udp = socket(AF_INET, SOCK_DGRAM)

    # Asignamos Host y Puerto al Objeto con el Método Bind
    socket_udp.bind((host, port))

    print('------------------------------------------------------------------------\n')
    print('       Servidor LoRa - Python del Grupo Integrado de Ingeniería.\n')
    print('------------------------------------------------------------------------')
    print('Esperando Comunicación ...')
```

```
# Iniciamos Contador para Impresión de Direcciones IP
counter = 0

# Iniciamos Bucle de Comunicación con el Cliente
try:
   while 1:

      # Recibimos el Mensaje con el Método Recvfrom
      message = socket_udp.recvfrom(1024)
      data = str(message[0])

      # Imprimimos Direcciones IP
      if counter is 0:
         counter += 1
         address = message [1]

         print('-----------------------------------------------------------------')
         print('Comunicación Establecida entre: ')
         print('    Dirección IP Servidor: ' + host)
         print('    Dirección IP Gateway:  ' + str(address[0]))
         print('-----------------------------------------------------------------')

      # Eliminamos Caracteres Incluídos Durante la Transmisión
      data = data.lstrip("b'")
      data = data.replace("'", "")

      # División del Mensaje Recibido
      divisions = data.split("|")

      # Impresión de Separadores Iniciales de Mensaje
      if 'Gateway LoRaWAN' in divisions[0]:
         print('\n-----------------------------------------------------------------\n')
      if 'Iniciando' in divisions[0]:
         print ('\n-----------------------------------------------------------------')
      if 'Actualizacion' in divisions[0]:
         print ('\n-----------------------------------------------------------------')
```

```
        if 'Informacion de paquetes' in divisions[0]:
            print ('-----------------------------------------------------------------')
        if 'Parametros' in divisions[0]:
            print ('\n-----------------------------------------------------------------')

        # Impresión del Mensaje Recibido
        for i in range(len(divisions)):
            print (divisions[i])

        # Impresión de Separadores Finales de Mensaje
        if 'Gateway LoRaWAN' in divisions[0]:
            print('\n-----------------------------------------------------------------')
        if 'Iniciando' in divisions[0]:
            print('-----------------------------------------------------------------')
        if 'Informacion' in divisions[0]:
            print('-----------------------------------------------------------------')
        if 'Parametros' in divisions[0]:
            print('-----------------------------------------------------------------')

    except KeyboardInterrupt:
        socket_udp.close() # Cerramos el Socket Server
        sys.exit() # Salida de Programa
```

UNIVERSIDADE DA CORUÑA

ESCUELA POLITÉCNICA SUPERIOR

TRABAJO FIN DE MÁSTER

CURSO 2016/2017

DISEÑO Y DESARROLLO DEL SISTEMA DE COMUNICACIONES INALÁMBRICO DE UN ENJAMBRE DE UAV'S COLABORATIVOS

MÁSTER EN INGENIERÍA INDUSTRIAL

DOCUMENTO DEL PROYECTO

ANEXO V – DATASHEETS

DATASHEET MICROCHIP RN2483

RN2483

Low-Power Long Range LoRa™ Technology Transceiver Module

General Features

- On-board LoRaWAN™ Class A protocol stack
- ASCII command interface over UART
- Compact form factor: 17.8 x 26.7 x 3 mm
- Castellated SMT pads for easy and reliable PCB mounting
- Environmentally friendly, RoHS compliant
- European R&TTE Directive Assessed Radio Module
- Device Firmware Upgrade (DFU) over UART, see *"RN2483 LoRa™ Technology Module Command Reference User's Guide"* (DS40001784A)

Operational

- Single operating voltage: 2.1V to 3.6V (3.3V typical)
- Temperature range: -40°C to +85°C
- Low-power consumption
- Programmable RF Communication Bit Rate up to 300 kbps with FSK modulation, 5468 bps with LoRa™ Technology modulation
- Integrated MCU, Crystal, EUI-64 Node Identity Serial EEPROM, Radio Transceiver with Analog Front End, Matching Circuitry
- 14 GPIOs for control and status

RF/Analog Features

- Low-Power Long Range Transceiver operating in the 433 MHz and 868 MHz frequency bands
- High Receiver Sensitivity: down to -148 dBm
- TX Power: adjustable up to +14 dBm high efficiency PA
- FSK, GFSK, and LoRa Technology modulation
- IIP3 = -11 dBm
- >15 km coverage at suburban and >5 km coverage at urban area

Description

Microchip's RN2483 Low-Power Long Range LoRa Technology Transceiver module provides an easy to use, low-power solution for long range wireless data transmission. The advanced command interface offers rapid time to market.

The RN2483 module complies with the LoRaWAN Class A protocol specifications. It integrates RF, a baseband controller, command Application Programming Interface (API) processor, making it a complete long range Solution.

The RN2483 module is suitable for simple long range sensor applications with external host MCU.

Applications

- Automated Meter Reading
- Home and Building Automation
- Wireless Alarm and Security Systems
- Industrial Monitoring and Control
- Machine to Machine
- Internet of Things (IoT)

Table of Contents

TO OUR VALUED CUSTOMERS

It is our intention to provide our valued customers with the best documentation possible to ensure successful use of your Microchip products. To this end, we will continue to improve our publications to better suit your needs. Our publications will be refined and enhanced as new volumes and updates are introduced.

If you have any questions or comments regarding this publication, please contact the Marketing Communications Department via E-mail at **docerrors@microchip.com**. We welcome your feedback.

Most Current Data Sheet

To obtain the most up-to-date version of this data sheet, please register at our Worldwide Web site at:

http://www.microchip.com

You can determine the version of a data sheet by examining its literature number found on the bottom outside corner of any page. The last character of the literature number is the version number, (e.g., DS30000000A is version A of document DS30000000).

Errata

An errata sheet, describing minor operational differences from the data sheet and recommended workarounds, may exist for current devices. As device/documentation issues become known to us, we will publish an errata sheet. The errata will specify the revision of silicon and revision of document to which it applies.

To determine if an errata sheet exists for a particular device, please check with one of the following:

- Microchip's Worldwide Web site; **http://www.microchip.com**
- Your local Microchip sales office (see last page)

When contacting a sales office, please specify which device, revision of silicon and data sheet (include literature number) you are using.

Customer Notification System

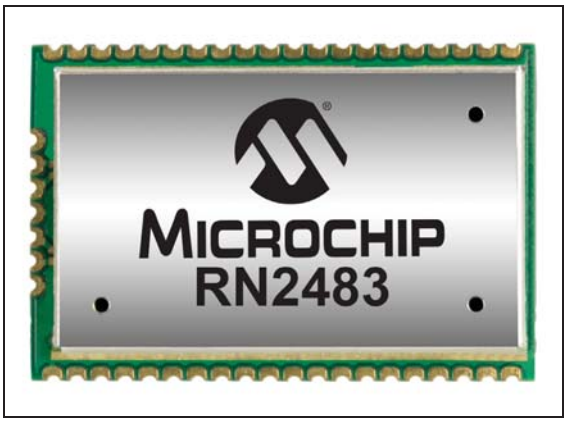
Register on our web site at **www.microchip.com** to receive the most current information on all of our products.

1.0 DEVICE OVERVIEW

The RN2483 Transceiver module features LoRa Technology RF modulation, which provides long range spread spectrum communication with high interference immunity.

Using LoRa Technology modulation technique, RN2483 can achieve a receiver sensitivity of -148 dBm. The high sensitivity combined with the integrated +14 dBm power amplifier yields industry leading link budget, which makes it optimal for applications requiring extended range and robustness.

FIGURE 1-1: RN2483 TOP VIEW



LoRa Technology modulation also provides significant advantages in both blocking and selectivity compared to the conventional modulation techniques, solving the traditional design compromise between extended range, interference immunity, and low-power consumption.

The RN2483 module delivers exceptional phase noise, selectivity, receiver linearity, and IIP3 for significantly lower power consumption. Figure 1-1, Figure 1-2, and Figure 1-3 show the module's top view, the pinout, and the block diagram.

FIGURE 1-2: RN2483 PIN DIAGRAM

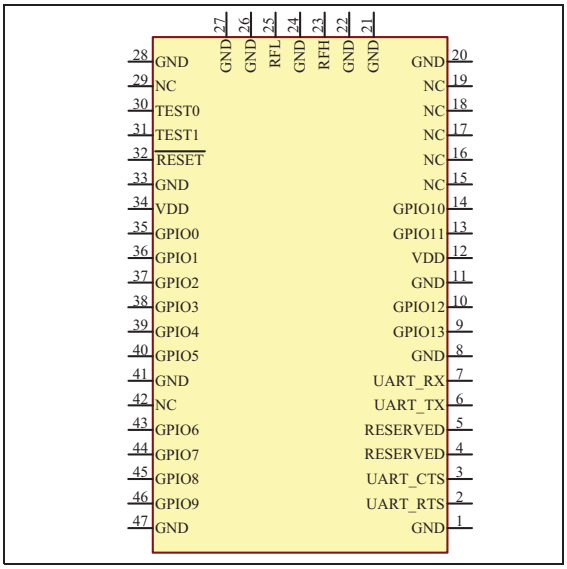


FIGURE 1-3: RN2483 BLOCK DIAGRAM

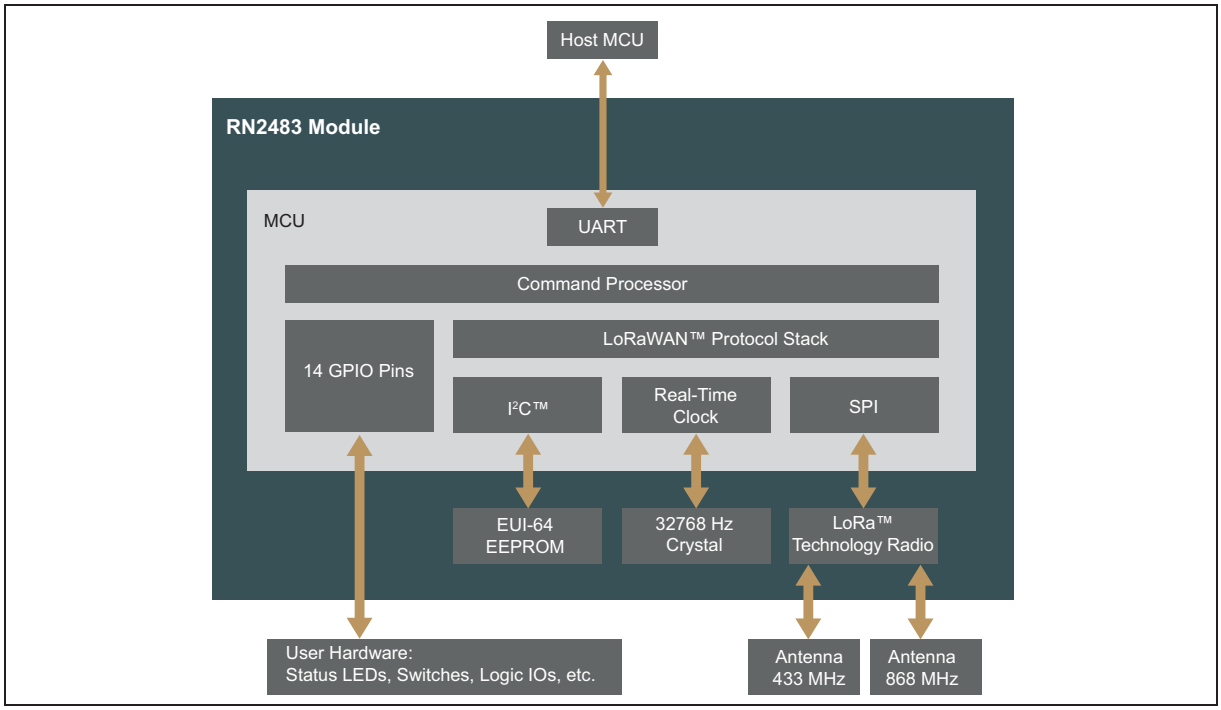


Table 1-1 describes the module's pins.

TABLE 1-1: PIN DESCRIPTION

Pin	Symbol	Type	Description
1	GND	Power	Ground supply terminal
2	UART_RTS	Output	Communication UART RTS signal(1)
3	UART_CTS	Input	Communication UART CTS signal(1)
4	RESERVED	—	Do not connect
5	RESERVED	—	Do not connect
6	UART_TX	Output	Communication UART Transmit (TX)
7	UART_RX	Input	Communication UART Receive (RX)
8	GND	Power	Ground supply terminal
9	GPIO13	Input/Output	General purpose I/O pin
10	GPIO12	Input/Output	General purpose I/O pin
11	GND	Power	Ground supply terminal
12	VDD	Power	Positive supply terminal
13	GPIO11	Input/Output	General purpose I/O pin
14	GPIO10	Input/Output	General purpose I/O pin
15	NC	—	Not connected
16	NC	—	Not connected
17	NC	—	Not connected
18	NC	—	Not connected
19	NC	—	Not connected
20	GND	Power	Ground supply terminal
21	GND	Power	Ground supply terminal
22	GND	Power	Ground supply terminal
23	RFH	RF analog	RF signal pin for high band
24	GND	Power	Ground supply terminal
25	RFL	RF analog	RF signal pin for low band
26	GND	Power	Ground supply terminal
27	GND	Power	Ground supply terminal
28	GND	Power	Ground supply terminal
29	NC	—	Not connected
30	TEST0	—	Do not connect
31	TEST1	—	Do not connect
32	$\overline{\text{RESET}}$	Input	Active-low device Reset input
33	GND	Power	Ground supply terminal
34	VDD	Power	Positive supply terminal
35	GPIO0	Input/Output	General purpose I/O pin
36	GPIO1	Input/Output	General purpose I/O pin
37	GPIO2	Input/Output	General purpose I/O pin
38	GPIO3	Input/Output	General purpose I/O pin
39	GPIO4	Input/Output	General purpose I/O pin
40	GPIO5	Input/Output	General purpose I/O pin
41	GND	Power	Ground supply terminal
42	NC	—	Not connected
43	GPIO6	Input/Output	General purpose I/O pin

TABLE 1-1: PIN DESCRIPTION (CONTINUED)

Pin	Symbol	Type	Description
44	GPIO7	Input/Output	General purpose I/O pin
45	GPIO8	Input/Output	General purpose I/O pin
46	GPIO9	Input/Output	General purpose I/O pin
47	GND	Power	Ground supply terminal

Note 1: Optional handshake lines are supported in future firmware releases.

2.0 GENERAL SPECIFICATIONS

Table 2-1 provides the general specifications for the module. Table 2-2 and Table 2-3 provide the module's electrical characteristics and current consumption. Table 2-4 and Table 2-5 show the module's dimensions and the RF output power calibration data.

TABLE 2-1: GENERAL SPECIFICATIONS

Specification	Description
Frequency Band	863.000 MHz to 870.000 MHz; 433.050 MHz to 434.790 MHz
Modulation Method	FSK, GFSK, and LoRa™ Technology modulation
Maximum Over-the-Air Data Rate	300 kbps with FSK modulation; 5468 bps with LoRa Technology modulation
RF connection	Board edge connection
Interface	UART
Operation Range	>15 km coverage at suburban; >5 km coverage at urban area
Sensitivity at 0.1% BER	-148 dBm(1)
RF TX Power	Adjustable up to max. 10 dBm on 433 MHz band (limited to meet regulations); max. 14 dBm on the 868 MHz band(2)
Temperature (operating)	-40°C to +85°C
Temperature (storage)	-40°C to +115°C
Humidity	10% ~ 90% non-condensing

Note 1: Depends on modulation. Expand Spreading Factor (SF).

2: TX power is adjustable. For more information, refer to the *"RN2483 LoRa™ Technology Module Command Reference User's Guide"* (DS40001784A).

TABLE 2-2: ELECTRICAL CHARACTERISTICS

Parameter	Min.	Typ.	Max.	Units
Supply Voltage	2.1	—	3.6	V
Voltage on any pin with respect to VSS (except VDD)	-0.3	—	VDD + 0.3	V
Voltage on VDD with respect to VSS	-0.3	—	3.9	V
Input Clamp Current (IIK) (VI < 0 or VI > VDD)	—	—	+/-20	mA
Output Camp Current (IOK) (VO < 0 or VO > VDD)	—	—	+/-20	mA
GPIO sink/source current each	—	—	25/25	mA
Total GPIO sink/source current	—	—	200/185	mA
RAM Data Retention Voltage (in Sleep mode or Reset state)	1.5	—	—	V
VDD Start Voltage to ensure internal Power-on Reset signal	—	—	0.7	V
VDD Rise Rate to ensure internal Power-on Reset signal	0.05	—	—	V/ms
Brown-out Reset Voltage	1.75	1.9	2.05	V
Logic Input Low Voltage	—	—	0.15 x VDD	V
Logic Input High Voltage	0.8 x VDD	—	—	V
Input Leakage at <25°C (VSS<VPIN<VDD, Pin at high-impedance)	—	0.1	50	nA
Input Leakage at +60°C (VSS<VPIN<VDD, Pin at high-impedance)	—	0.7	100	nA
Input Leakage at +85°C (VSS<VPIN<VDD, Pin at high-impedance)	—	4	200	nA
RF Input Level	—	—	+10	dBm

TABLE 2-3: CURRENT CONSUMPTION

Mode	Typical Current at 3V (mA)
Idle	2.8
RX	14.2
Deep Sleep	0.0099

TABLE 2-4: MODULE DIMENSIONS

Parameter	Value
Dimensions	17.8 x 26.7 x 3 mm
Weight	2.05g

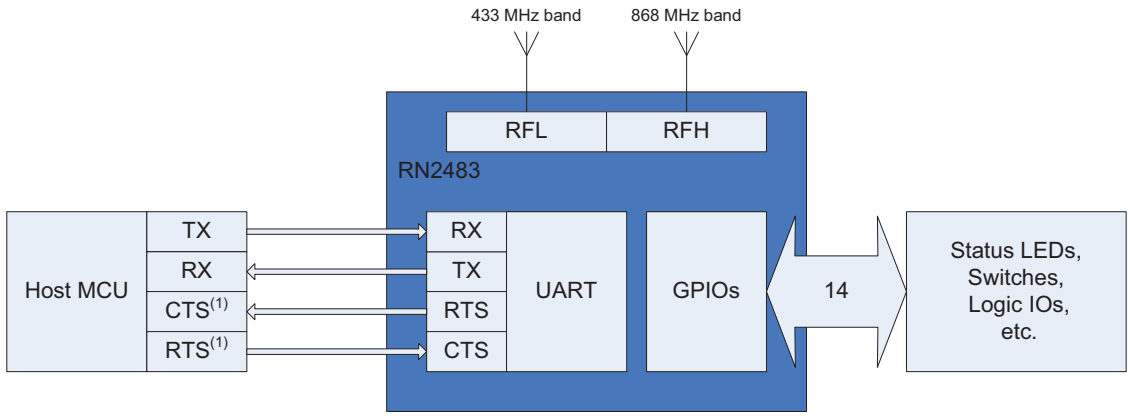
TABLE 2-5: OUTPUT POWER OF TX POWER SETTING

Band	TX Power Setting	Output Power (dBm)	Typical Supply Current at 3V (mA)
868 MHz	-3	-4.0	17.3
	-2	-2.9	18.0
	-1	-1.9	18.7
	0	-1.7	20.2
	1	-0.6	21.2
	2	0.4	22.3
	3	1.4	23.5
	4	2.5	24.7
	5	3.6	26.1
	6	4.7	27.5
	7	5.8	28.8
	8	6.9	30.0
	9	8.1	31.2
	10	9.3	32.4
	11	10.4	33.7
	12	11.6	35.1
	13	12.5	36.5
	14	13.5	38.0
	15	14.1	38.9
433 MHz	-3	-3.5	14.7
	-2	-2.3	15.1
	-1	-1.3	15.6
	0	-2.3	15.8
	1	-1.2	16.4
	2	-0.1	17.0
	3	1.0	17.7
	4	2.1	18.5
	5	3.2	19.4
	6	4.3	20.3
	7	5.4	21.4
	8	6.5	22.3
	9	7.6	23.3
	10	8.8	24.5
	11	9.9	25.8
	12	10.9	27.3
	13	11.9	28.8
	14	12.9	30.7
	15	13.6	32.9

3.0 TYPICAL HARDWARE CONNECTIONS

Figure 3-1 shows the typical hardware connections.

FIGURE 3-1: HARDWARE CONNECTIONS



Note 1: Optional handshake lines are supported in future firmware releases.

3.1 INTERFACE TO HOST MCU

The RN2483 module has a dedicated UART interface to communicate with a host controller. Optional handshake lines are supported in future firmware releases. The *"RN2483 LoRa™ Technology Module Command Reference User's Guide"* (DS40001784A) provides a detailed UART command description. Table 3-1 shows the default settings for the UART communication.

TABLE 3-1: DEFAULT UART SETTINGS

Specification	Description
Baud Rate	57600 bps
Packet Length	8 bit
Parity Bit	No
Stop Bits	1 bit
Hardware Flow Control	No

3.2 GPIO PINS (GPIO0-GPIO13)

The module has 14 GPIO pins. These lines can be connected to switches, LEDs, and relay outputs. The pins are either logic inputs or outputs that can be accessed via the module firmware. These pins have limited sink and source capabilities. Electrical characteristics are described in Table 2-2.

3.3 RF CONNECTIONS (RFL, RFH)

RFL is the RF analog port for the lower frequency band (433 MHz) while RFH is for the higher frequency band (868 MHz). When routing RF paths, use proper strip lines with an impedance of 50 Ohm.

3.4 RESET PIN

The module's reset pin is an active-low logic input.

3.5 POWER PINS

It is recommended to connect power pins (Pin 12 and 34) to a stable supply voltage with sufficient source current. Table 2-2 shows the current consumption.

Additional filtering capacitors are not required but can be used to ensure stable supply voltage in noisy environment.

4.0 PHYSICAL DIMENSIONS

Figure 4-1 and Figure 4-2 illustrate the physical dimensions and the recommended PCB layout for the RN2483 module.

FIGURE 4-1: RN2483 PHYSICAL DIMENSIONS

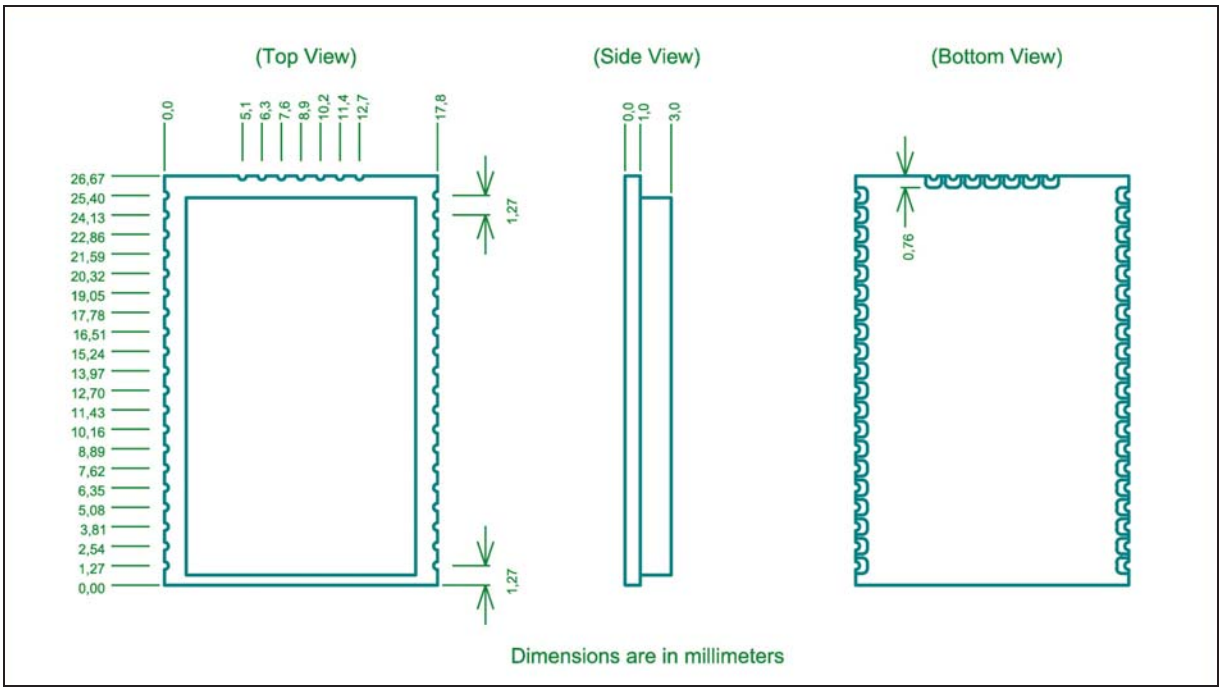
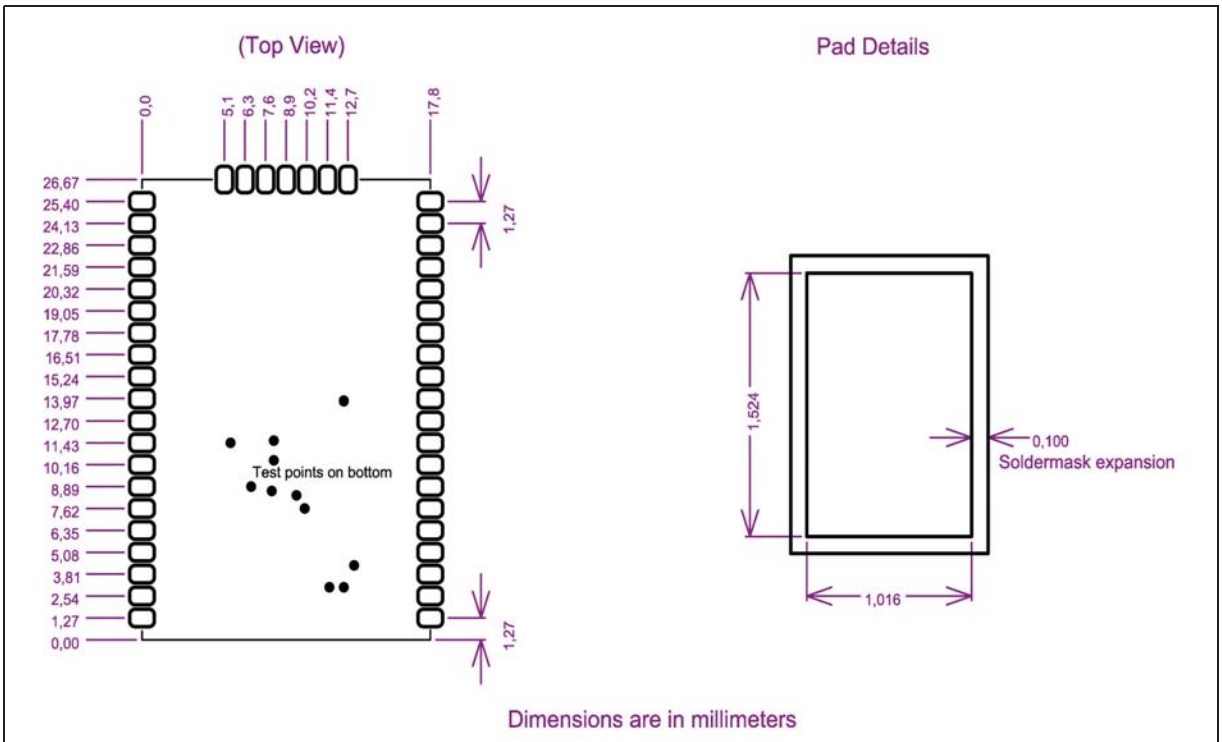


FIGURE 4-2: RECOMMENDED PCB FOOTPRINT



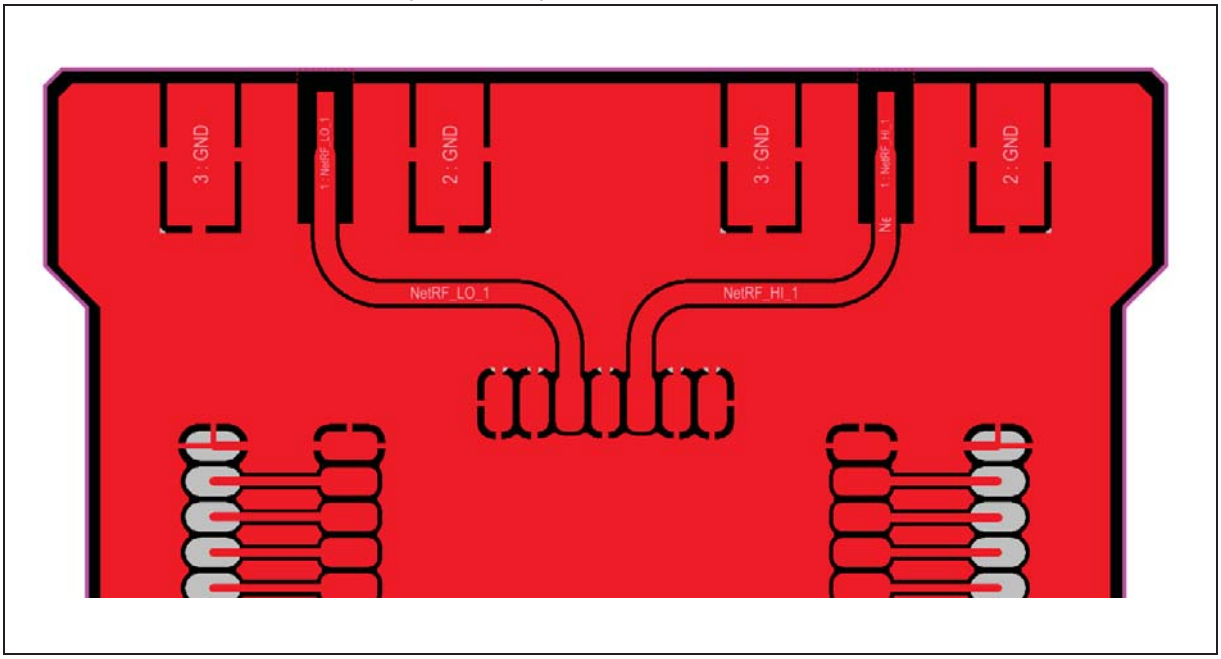
Note: Host PCB top layer copper traces must have solder mask to avoid shorting test pins on the bottom of the module.

5.0 APPLICATION INFORMATION

5.1 RF pins and strip line

The RF signals must be routed with properly terminated 50 Ohm strip lines. Use curves instead of sharp corners. Keep the routing path as short as possible. When routing the RF paths, use proper strip lines with an impedance of 50 Ohm. Figure 5-1 shows a routing example.

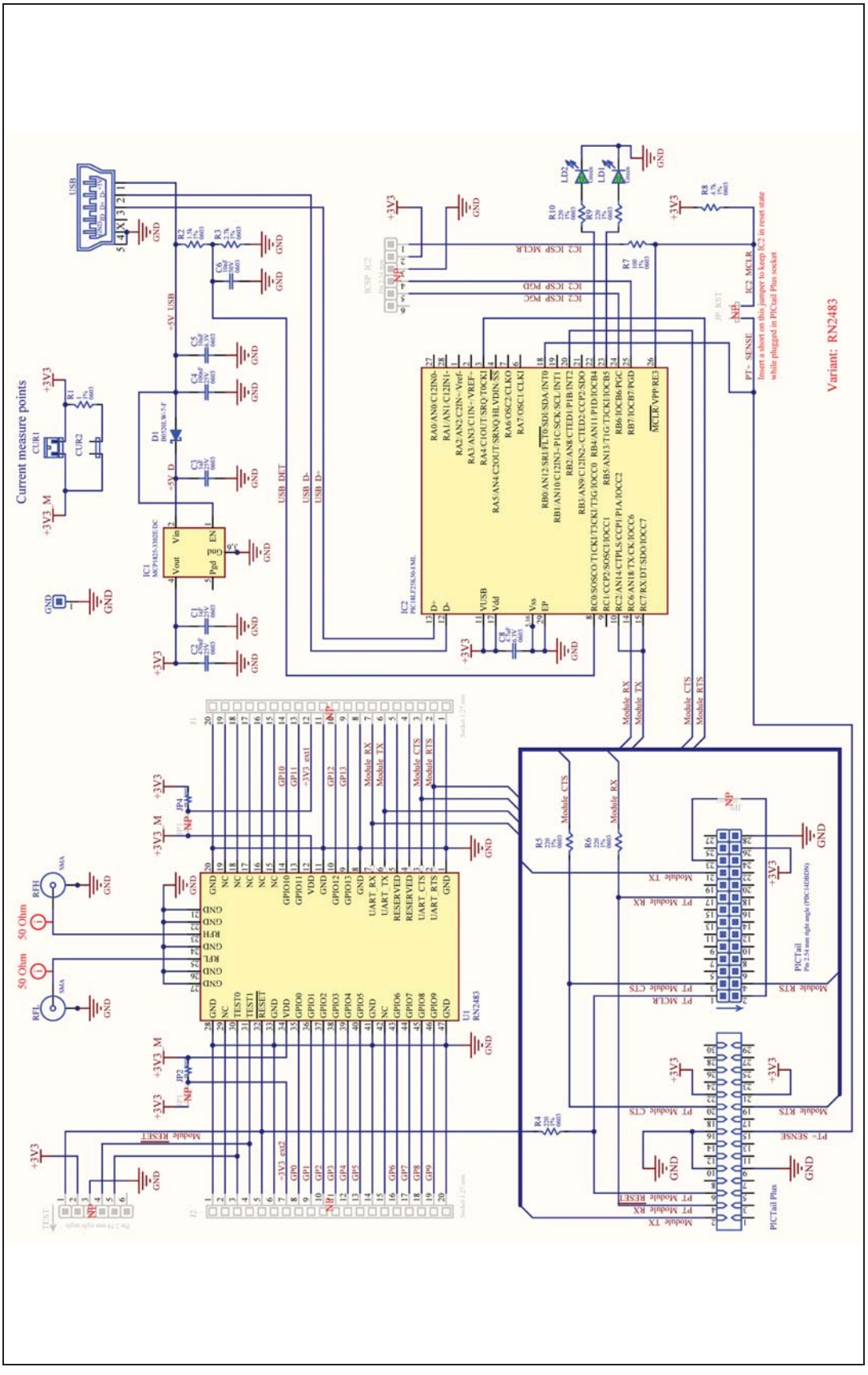
FIGURE 5-1: RF ROUTING (EXAMPLE)



5.2 APPLICATION SCHEMATIC

Figure 5-2 shows the schematic for the RN2483 PICtail™/PICtail Plus Daughter Board.

FIGURE 5-2: PICTAIL™/PICTAIL PLUS DAUGHTER BOARD SCHEMATIC



6.0 REGULATORY APPROVAL

This section outlines the regulatory information for the RN2483 module for Europe.

6.1 Europe

The RN2483 module is an R&TTE Directive assessed radio module that is CE marked and has been manufactured and tested with the intention of being integrated into a final product.

The RN2483 module has been tested to R&TTE Directive 1999/5/EC Essential Requirements for Health and Safety (Article 3.1a), Electromagnetic Compatibility (EMC) (Article 3.1b), and Radio (Article 3.2) and are summarized in Table 6-1: European Compliance Testing. A Notified Body Opinion has also been issued. All test reports are available on the product web page at http://www.microchip.com.

The R&TTE Compliance Association provides guidance on modular devices in document **Technical Guidance Note 01** available at http://www.rtteca.com/html/download_area.htm.

Note:	To maintain conformance to the testing listed in Table 6-1: European Compliance Testing, the module shall be installed in accordance with the installation instructions in this data sheet and shall not be modified. When integrating a radio module into a completed product the integrator becomes the manufacturer of the final product and is therefore responsible for demonstrating compliance of the final product with the essential requirements of the R&TTE Directive.

6.1.1 LABELING AND USER INFORMATION REQUIREMENTS

The label on the final product which contains the RN2483 module must follow CE marking requirements. The *"R&TTE Compliance Association Technical Guidance Note 01"* provides guidance on final product CE marking.

6.1.2 EXTERNAL ANTENNA REQUIREMENTS

From R&TTE Compliance Association document **Technical Guidance Note 01**:

Provided the integrator installing an assessed radio module with an integral or specific antenna and installed in conformance with the radio module manufacturer's installation instructions requires no further evaluation under Article 3.2 of the R&TTE Directive and does not require further involvement of an R&TTE Directive Notified Body for the final product (Section 2.2.4).

6.1.3 HELPFUL WEB SITES

A document that can be used as a starting point in understanding the use of Short Range Devices (SRD) in Europe is the European Radio Communications Committee (ERC) Recommendation 70-03 E, which can be downloaded from the European Radio Communications Office (ERO) at: http://www.ero.dk/.

Additional helpful web sites are:

- Radio and Telecommunications Terminal Equipment (R&TTE): http://ec.europa.eu/enterprise/sectors/rtte/regulatory-framework/index_en.htm
- European Conference of Postal and Telecommunications Administrations (CEPT): http://www.cept.org
- European Telecommunications Standards Institute (ETSI): http://www.etsi.org
- European Radio Communications Office (ERO): http://www.ero.dk/
- The Radio and Telecommunications Terminal Equipment Compliance Association (R&TTE CA): http://www.rtteca.com/

TABLE 6-1: EUROPEAN COMPLIANCE TESTING

Certification	Standards	Article	Laboratory	Report Number	Date
Safety	IEC 60950-1:2005 (2nd Ed: A1:2009)	(3.1a)	TRaC Global Ltd.	TRA-025134-43-00A	2/12/2015
Health	EN 62479	—	TRaC Global Ltd.	TRA-025134-01-47-03A	2/12/2015
EMC	EN 301 489-3 v1.6.1	(3.1b)	TRaC Global Ltd.	TRA-025134-43-00A	2/12/2015
Radio	EN 300 220-2 v2.4.1	(3.2)	TRaC Global Ltd.	TRA-025134-01-47-00A (433 MHz) TRA-025134-01-47-01A(868MHz)	2/12/2015

APPENDIX A: REVISION HISTORY

Revision A (March 2015)

This is the initial release of this document.

NOTES:

THE MICROCHIP WEB SITE

Microchip provides online support via our WWW site at www.microchip.com. This web site is used as a means to make files and information easily available to customers. Accessible by using your favorite Internet browser, the web site contains the following information:

- **Product Support** – Data sheets and errata, application notes and sample programs, design resources, user's guides and hardware support documents, latest software releases and archived software
- **General Technical Support** – Frequently Asked Questions (FAQ), technical support requests, online discussion groups, Microchip consultant program member listing
- **Business of Microchip** – Product selector and ordering guides, latest Microchip press releases, listing of seminars and events, listings of Microchip sales offices, distributors and factory representatives

CUSTOMER CHANGE NOTIFICATION SERVICE

Microchip's customer notification service helps keep customers current on Microchip products. Subscribers will receive e-mail notification whenever there are changes, updates, revisions or errata related to a specified product family or development tool of interest.

To register, access the Microchip web site at www.microchip.com. Under "Support", click on "Customer Change Notification" and follow the registration instructions.

CUSTOMER SUPPORT

Users of Microchip products can receive assistance through several channels:

- Distributor or Representative
- Local Sales Office
- Field Application Engineer (FAE)
- Technical Support

Customers should contact their distributor, representative or Field Application Engineer (FAE) for support. Local sales offices are also available to help customers. A listing of sales offices and locations is included in the back of this document.

Technical support is available through the web site at: http://microchip.com/support

NOTES:

PRODUCT IDENTIFICATION SYSTEM

To order or obtain information, e.g., on pricing or delivery, refer to the factory or the listed sales office.

PART NO.	I	RM	XXX
Device	Temperature Range	Package	Firmware Revision Number

Device:	RN2483:		Low-Power Long Range LoRa™ Technology Transceiver module
Temperature Range:	I	=	-40°C to +85°C (Industrial)
Package:	RM	=	Radio Module

Examples:

RN2483-I/RM: Industrial temperature

NOTES:

Note the following details of the code protection feature on Microchip devices:

- Microchip products meet the specification contained in their particular Microchip Data Sheet.
- Microchip believes that its family of products is one of the most secure families of its kind on the market today, when used in the intended manner and under normal conditions.
- There are dishonest and possibly illegal methods used to breach the code protection feature. All of these methods, to our knowledge, require using the Microchip products in a manner outside the operating specifications contained in Microchip's Data Sheets. Most likely, the person doing so is engaged in theft of intellectual property.
- Microchip is willing to work with the customer who is concerned about the integrity of their code.
- Neither Microchip nor any other semiconductor manufacturer can guarantee the security of their code. Code protection does not mean that we are guaranteeing the product as "unbreakable."

Code protection is constantly evolving. We at Microchip are committed to continuously improving the code protection features of our products. Attempts to break Microchip's code protection feature may be a violation of the Digital Millennium Copyright Act. If such acts allow unauthorized access to your software or other copyrighted work, you may have a right to sue for relief under that Act.

Trademarks

ISBN: 978-1-63277-123-0

QUALITY MANAGEMENT SYSTEM
CERTIFIED BY DNV
═ ISO/TS 16949 ═

Microchip received ISO/TS-16949:2009 certification for its worldwide headquarters, design and wafer fabrication facilities in Chandler and Tempe, Arizona; Gresham, Oregon and design centers in California and India. The Company's quality system processes and procedures are for its PIC® MCUs and dsPIC® DSCs, KEELOQ® code hopping devices, Serial EEPROMs, microperipherals, nonvolatile memory and analog products. In addition, Microchip's quality system for the design and manufacture of development systems is ISO 9001:2000 certified.

Worldwide Sales and Service

AMERICAS

Corporate Office
2355 West Chandler Blvd.
Chandler, AZ 85224-6199
Tel: 480-792-7200
Fax: 480-792-7277
Technical Support:
http://www.microchip.com/support
Web Address:
www.microchip.com

Atlanta
Duluth, GA
Tel: 678-957-9614
Fax: 678-957-1455

Austin, TX
Tel: 512-257-3370

Boston
Westborough, MA
Tel: 774-760-0087
Fax: 774-760-0088

Chicago
Itasca, IL
Tel: 630-285-0071
Fax: 630-285-0075

Cleveland
Independence, OH
Tel: 216-447-0464
Fax: 216-447-0643

Dallas
Addison, TX
Tel: 972-818-7423
Fax: 972-818-2924

Detroit
Novi, MI
Tel: 248-848-4000

Houston, TX
Tel: 281-894-5983

Indianapolis
Noblesville, IN
Tel: 317-773-8323
Fax: 317-773-5453

Los Angeles
Mission Viejo, CA
Tel: 949-462-9523
Fax: 949-462-9608

New York, NY
Tel: 631-435-6000

San Jose, CA
Tel: 408-735-9110

Canada - Toronto
Tel: 905-673-0699
Fax: 905-673-6509

ASIA/PACIFIC

Asia Pacific Office
Suites 3707-14, 37th Floor
Tower 6, The Gateway
Harbour City, Kowloon
Hong Kong
Tel: 852-2943-5100
Fax: 852-2401-3431

Australia - Sydney
Tel: 61-2-9868-6733
Fax: 61-2-9868-6755

China - Beijing
Tel: 86-10-8569-7000
Fax: 86-10-8528-2104

China - Chengdu
Tel: 86-28-8665-5511
Fax: 86-28-8665-7889

China - Chongqing
Tel: 86-23-8980-9588
Fax: 86-23-8980-9500

China - Dongguan
Tel: 86-769-8702-9880

China - Hangzhou
Tel: 86-571-8792-8115
Fax: 86-571-8792-8116

China - Hong Kong SAR
Tel: 852-2943-5100
Fax: 852-2401-3431

China - Nanjing
Tel: 86-25-8473-2460
Fax: 86-25-8473-2470

China - Qingdao
Tel: 86-532-8502-7355
Fax: 86-532-8502-7205

China - Shanghai
Tel: 86-21-5407-5533
Fax: 86-21-5407-5066

China - Shenyang
Tel: 86-24-2334-2829
Fax: 86-24-2334-2393

China - Shenzhen
Tel: 86-755-8864-2200
Fax: 86-755-8203-1760

China - Wuhan
Tel: 86-27-5980-5300
Fax: 86-27-5980-5118

China - Xian
Tel: 86-29-8833-7252
Fax: 86-29-8833-7256

ASIA/PACIFIC

China - Xiamen
Tel: 86-592-2388138
Fax: 86-592-2388130

China - Zhuhai
Tel: 86-756-3210040
Fax: 86-756-3210049

India - Bangalore
Tel: 91-80-3090-4444
Fax: 91-80-3090-4123

India - New Delhi
Tel: 91-11-4160-8631
Fax: 91-11-4160-8632

India - Pune
Tel: 91-20-3019-1500

Japan - Osaka
Tel: 81-6-6152-7160
Fax: 81-6-6152-9310

Japan - Tokyo
Tel: 81-3-6880- 3770
Fax: 81-3-6880-3771

Korea - Daegu
Tel: 82-53-744-4301
Fax: 82-53-744-4302

Korea - Seoul
Tel: 82-2-554-7200
Fax: 82-2-558-5932 or
82-2-558-5934

Malaysia - Kuala Lumpur
Tel: 60-3-6201-9857
Fax: 60-3-6201-9859

Malaysia - Penang
Tel: 60-4-227-8870
Fax: 60-4-227-4068

Philippines - Manila
Tel: 63-2-634-9065
Fax: 63-2-634-9069

Singapore
Tel: 65-6334-8870
Fax: 65-6334-8850

Taiwan - Hsin Chu
Tel: 886-3-5778-366
Fax: 886-3-5770-955

Taiwan - Kaohsiung
Tel: 886-7-213-7828

Taiwan - Taipei
Tel: 886-2-2508-8600
Fax: 886-2-2508-0102

Thailand - Bangkok
Tel: 66-2-694-1351
Fax: 66-2-694-1350

EUROPE

Austria - Wels
Tel: 43-7242-2244-39
Fax: 43-7242-2244-393

Denmark - Copenhagen
Tel: 45-4450-2828
Fax: 45-4485-2829

France - Paris
Tel: 33-1-69-53-63-20
Fax: 33-1-69-30-90-79

Germany - Dusseldorf
Tel: 49-2129-3766400

Germany - Munich
Tel: 49-89-627-144-0
Fax: 49-89-627-144-44

Germany - Pforzheim
Tel: 49-7231-424750

Italy - Milan
Tel: 39-0331-742611
Fax: 39-0331-466781

Italy - Venice
Tel: 39-049-7625286

Netherlands - Drunen
Tel: 31-416-690399
Fax: 31-416-690340

Poland - Warsaw
Tel: 48-22-3325737

Spain - Madrid
Tel: 34-91-708-08-90
Fax: 34-91-708-08-91

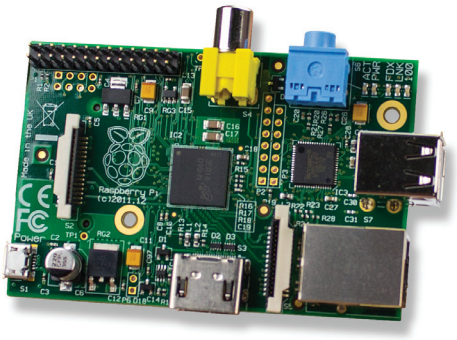
Sweden - Stockholm
Tel: 46-8-5090-4654

UK - Wokingham
Tel: 44-118-921-5800
Fax: 44-118-921-5820

01/27/15

DATASHEET RASPBERRY PI MODELO 1 B

Raspberry Pi



MODEL B

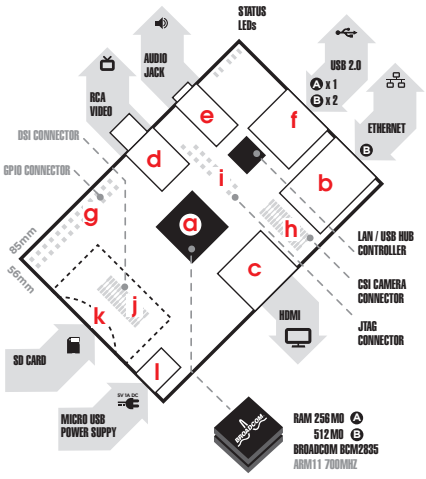
Product Name	**Raspberry Pi Model B**
Product Description	The Raspberry Pi is a small, powerful and lightweight ARM based computer which can do many of the things a desktop PC can do. The powerful graphics capabilities and HDMi video output make it ideal for multimedia applications such as media centres and narrowcasting solutions. The Raspberry Pi is based on a Broadcom BCM2835 chip. It does not feature a built-in hard disk or solid-state drive, instead relying on an SD card for booting and long-term storage.
RS Part Number	**756-8308**

Specifications

Chip	Broadcom BCM2835 SoC (a)
Core architecture	ARM11
CPU	700 MHz Low Power ARM1176JZFS Applications Processor
GPU	Dual Core VideoCore IV® Multimedia Co-Processor Provides Open GL ES 2.0, hardware-accelerated OpenVG, and 1080p30 H.264 high-profile decode Capable of 1Gpixel/s, 1.5Gtexel/s or 24GFLOPs with texture filtering and DMA infrastructure
Memory	512MB SDRAM
Operating System	Boots from SD card, running a version of the Linux operating system
Dimensions	85.6 x 53.98 x 17mm
Power	Micro USB socket 5V, 1.2A (l)

Connectors:

Ethernet	10/100 BaseT Ethernet socket (b)
Video Output	HDMI (rev 1.3 & 1.4) (c); Composite RCA (PAL and NTSC) (d)
Audio Output	3.5mm jack (e), HDMI
USB 2.0	Dual USB Connector (f)
GPIO Connector	26-pin 2.54 mm (100 mil) expansion header: 2x13 strip. Providing 8 GPIO pins plus access to I²C, SPI and UART as well as +3.3 V, +5 V and GND supply lines (g)
Camera Connector	15-pin MIPI Camera Serial Interface (CSI-2) (h)
JTAG	Not populated (i)
Display Connector	Display Serial Interface (DSI) 15 way flat flex cable connector with two data lanes and a clock lane (j)
Memory Card Slot	SDIO (k)

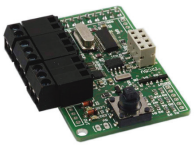


Accessories

▲ Camera Module
775-7731

▲ International power supply
765-3311

▲ 8GB SD card pre-programmed with NOOBS - **779-6770**



▲ Expansion board
772-2974

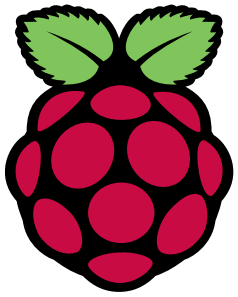
▲ WiFi dongle
760-3621

▲ 10400mAh Li-Ion battery pack
775-7517

▲ Raspberry Pi user guide
768-6686

DATASHEET RASPBERRY PI COMPUTE MODULE

DATASHEET



Raspberry Pi Compute Module (CM1)

Raspberry Pi Compute Module 3 (CM3)

Raspberry Pi Compute Module 3 Lite (CM3L)

Version 1.0, October 2016

Table 1: Revision History

Revision	Date	Description
1.0	13/10/2016	First release

Contents

List of Figures

List of Tables

1 Introduction

The Raspberry Pi Compute Module (CM1), Compute Module 3 (CM3) and Compute Module 3 Lite (CM3L) are DDR2-SODIMM-mechanically-compatible System on Modules (SoMs) containing processor, memory, eMMC Flash (for CM1 and CM3) and supporting power circuitry. These modules allow a designer to leverage the Raspberry Pi hardware and software stack in their own custom systems and form factors. In addition these module have extra IO interfaces over and above what is available on the Raspberry Pi model A/B boards opening up more options for the designer.

The CM1 contains a BCM2835 processor (as used on the original Raspberry Pi and Raspberry Pi B+ models), 512MByte LPDDR2 RAM and 4Gbytes eMMC Flash. The CM3 contains a BCM2837 processor (as used on the Raspberry Pi 3), 1Gbyte LPDDR2 RAM and 4Gbytes eMMC Flash. Finally the CM3L product is the same as CM3 except the eMMC Flash is not fitted, and the SD/eMMC interface pins are available for the user to connect their own SD/eMMC device.

Note that the BCM2837 processor is an evolution of the BCM2835 processor. The only real differences are that the BCM2837 can address more RAM (up to 1Gbyte) and the ARM CPU complex has been upgraded from a single core ARM11 in BCM2835 to a Quad core Cortex A53 with dedicated 512Kbyte L2 cache in BCM2837. All IO interfaces and peripherals stay the same and hence the two chips are largely software and hardware compatible.

The pinout of CM1 and CM3 are identical. Apart from the CPU upgrade and increase in RAM the other significant hwardware differences to be aware of are that CM3 has grown from 30mm to 31mm in height, the VBAT supply can now draw significantly more power under heavy CPU load, and the HDMI_HPD_N_1V8 (GPIO46_1V8 on CM1) and EMMC_EN_N_1V8 (GPIO47_1V8 on CM1) are now driven from an IO expander rather than the processor. If a designer of a CM1 product has a suitably specified VBAT, can accomodate the extra 1mm module height increase and has followed the design rules with respect to GPIO46_1V8 and GPIO47_1V8 then a CM3 should work fine in a board designed for a CM1.

2 Features

2.1 Hardware

- Low cost
- Low power
- High availability
- High reliability
  - Tested over millions of Raspberry Pis Produced to date
  - Module IO pins have 35u hard gold plating

2.2 Peripherals

- 48x GPIO
- 2x I2C
- 2x SPI
- 2x UART
- 2x SD/SDIO
- 1x HDMI 1.3a
- 1x USB2 HOST/OTG
- 1x DPI (Parallel RGB Display)
- 1x NAND interface (SMI)
- 1x 4-lane CSI Camera Interface (up to 1Gbps per lane)
- 1x 2-lane CSI Camera Interface (up to 1Gbps per lane)
- 1x 4-lane DSI Display Interface (up to 1Gbps per lane)
- 1x 2-lane DSI Display Interface (up to 1Gbps per lane)

2.3 Software

- ARMv6 (CM1) or ARMv7 (CM3, CM3L) Instruction Set
- Mature and stable Linux software stack
  - Latest Linux Kernel support
  - Many drivers upstreamed
  - Stable and well supported userland
  - Full availability of GPU functions using standard APIs

3 Block Diagram

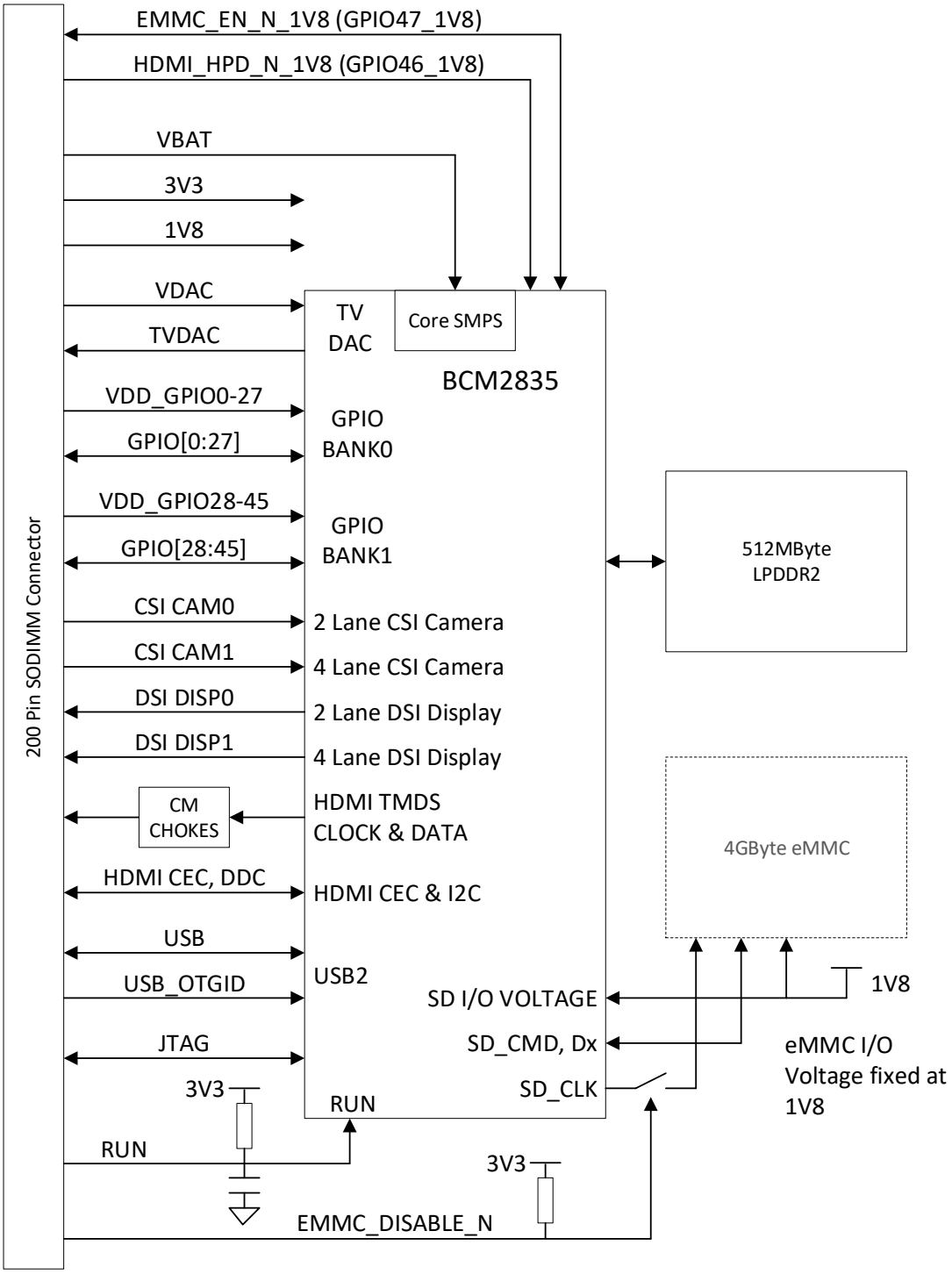


Figure 1: CM1 Block Diagram

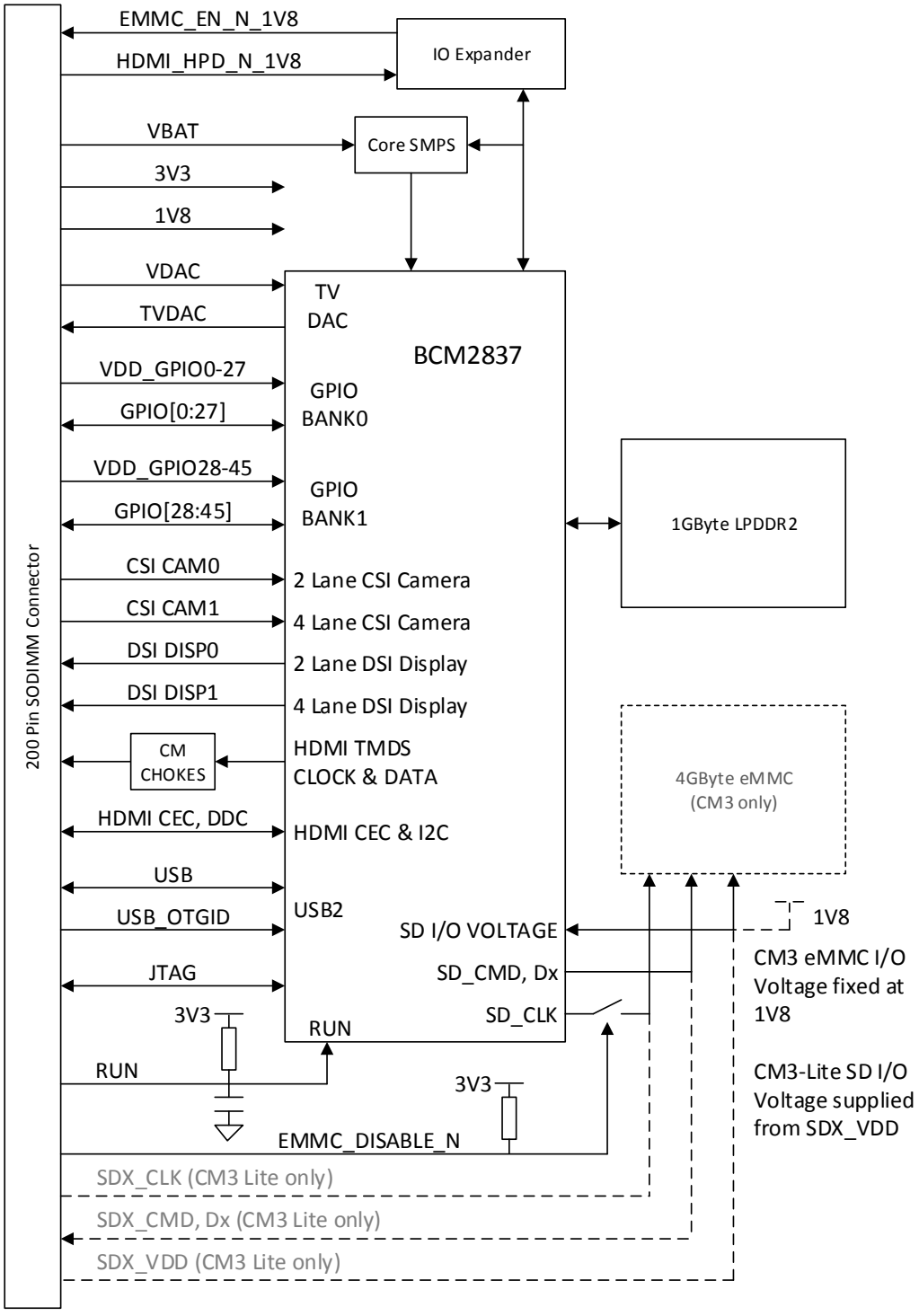


Figure 2: CM3/CM3L Block Diagram

4 Mechanical Specification

The Compute Modules conform to JEDEC MO-224 mechanical specification for 200 pin DDR2 (1.8V) SODIMM modules (with the exception that the CM3, CM3L modules are 31mm in height rather than 30mm of CM1) and therefore should work with the many DDR2 SODIMM sockets available on the market. **(Please note that the pinout of the Compute Module is not the same as a DDR2 SODIMM module; they are not electrically compatible.)**

The SODIMM form factor was chosen as a way to provide the 200 pin connections using a standard, readily available and low cost connector compatible with low cost PCB manufacture.

The maximum component height on the underside of the Compute Module is 1.2mm.

The maximum component height on the top side of the Compute Module is 1.5mm.

The Compute Module PCB thickness is 1.0mm +/- 0.1mm.

Note that the location and arrangement of components on the Compute Module may change slightly over time due to revisions for cost and manufacturing considerations; however, maximum component heights and PCB thickness will be kept as specified.

Figure 3 gives the CM1 mechanical dimensions. Figure 4 gives the CM3 and CM3L mechanical dimensions.

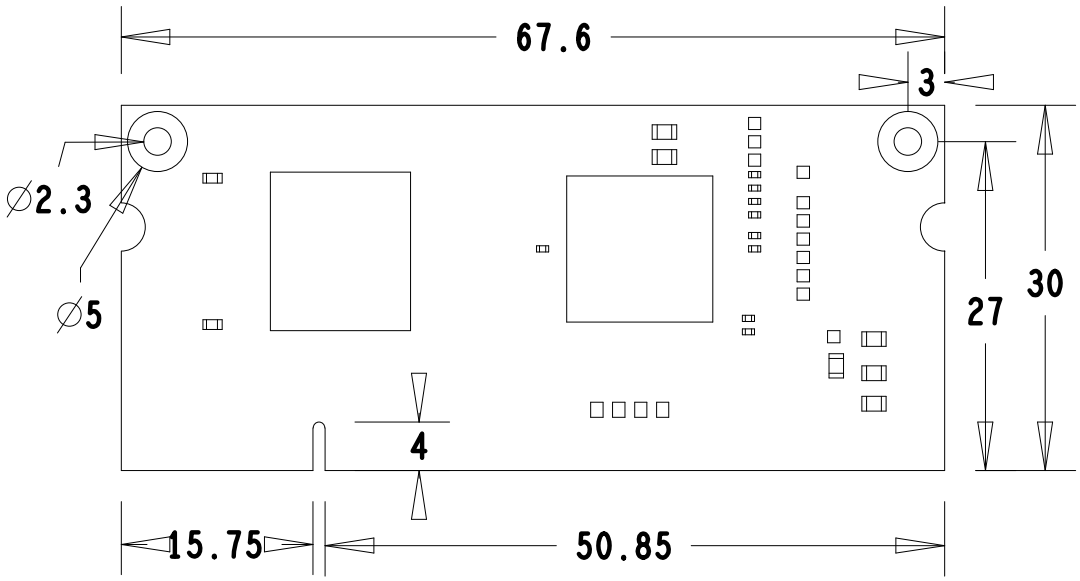


Figure 3: CM1 Mechanical Dimensions

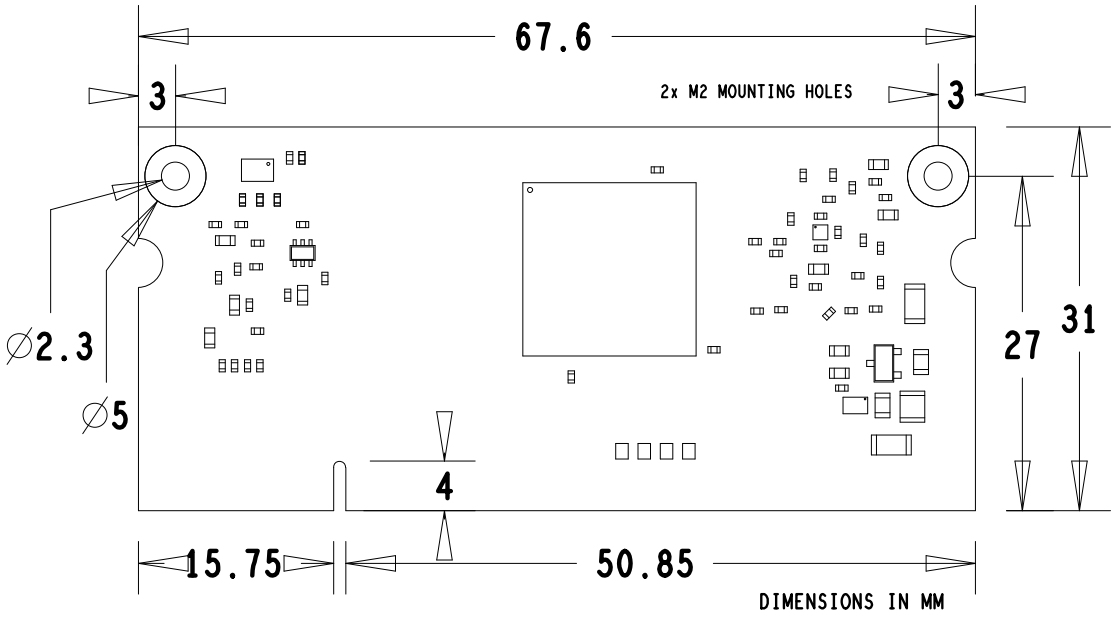


Figure 4: CM3 and CM3L Mechanical Dimensions

5 Pin Assignments

CM1	CM3-Lite	CM3	PIN		PIN	CM3	CM3-Lite	CM1
GND			1		2	EMMC_DISABLE_N		
GPIO0			3		4	NC	SDX_VDD	NC
GPIO1			5		6	NC	SDX_VDD	NC
GND			7		8	GND		NC
GPIO2			9		10	NC	SDX_CLK	NC
GPIO3			11		12	NC	SDX_CMD	NC
GND			13		14	GND		NC
GPIO4			15		16	NC	SDX_D0	NC
GPIO5			17		18	NC	SDX_D1	NC
GND			19		20	GND		NC
GPIO6			21		22	NC	SDX_D2	NC
GPIO7			23		24	NC	SDX_D3	NC
GND			25		26	GND		
GPIO8			27		28	GPIO28		
GPIO9			29		30	GPIO29		
GND			31		32	GND		
GPIO10			33		34	GPIO30		
GPIO11			35		36	GPIO31		
GND			37		38	GND		
GPIO0-27_VDD			39		40	GPIO0-27_VDD		
			KEY					
GPIO28-45_VDD			41		42	GPIO28-45_VDD		
GND			43		44	GND		
GPIO12			45		46	GPIO32		
GPIO13			47		48	GPIO33		
GND			49		50	GND		
GPIO14			51		52	GPIO34		
GPIO15			53		54	GPIO35		
GND			55		56	GND		
GPIO16			57		58	GPIO36		
GPIO17			59		60	GPIO37		
GND			61		62	GND		
GPIO18			63		64	GPIO38		
GPIO19			65		66	GPIO39		
GND			67		68	GND		
GPIO20			69		70	GPIO40		
GPIO21			71		72	GPIO41		
GND			73		74	GND		
GPIO22			75		76	GPIO42		
GPIO23			77		78	GPIO43		
GND			79		80	GND		
GPIO24			81		82	GPIO44		
GPIO25			83		84	GPIO45		
GND			85		86	GND		
GPIO26			87		88	HDMI_HPD_N_1V8		GPIO46_1V8
GPIO27			89		90	EMMC_EN_N_1V8		GPIO47_1V8
GND			91		92	GND		
DSI0_DN1			93		94	DSI1_DP0		
DSI0_DP1			95		96	DSI1_DN0		
GND			97		98	GND		
DSI0_DN0			99		100	DSI1_CP		
DSI0_DP0			101		102	DSI1_CN		
GND			103		104	GND		
DSI0_CN			105		106	DSI1_DP3		
DSI0_CP			107		108	DSI1_DN3		
GND			109		110	GND		
HDMI_CLK_N			111		112	DSI1_DP2		
HDMI_CLK_P			113		114	DSI1_DN2		
GND			115		116	GND		
HDMI_D0_N			117		118	DSI1_DP1		
HDMI_D0_P			119		120	DSI1_DN1		
GND			121		122	GND		
HDMI_D1_N			123		124	NC		
HDMI_D1_P			125		126	NC		
GND			127		128	NC		
HDMI_D2_N			129		130	NC		
HDMI_D2_P			131		132	NC		
GND			133		134	GND		
CAM1_DP3			135		136	CAM0_DP0		
CAM1_DN3			137		138	CAM0_DN0		
GND			139		140	GND		
CAM1_DP2			141		142	CAM0_CP		
CAM1_DN2			143		144	CAM0_CN		
GND			145		146	GND		
CAM1_CP			147		148	CAM0_DP1		
CAM1_CN			149		150	CAM0_DN1		
GND			151		152	GND		
CAM1_DP1			153		154	NC		
CAM1_DN1			155		156	NC		
GND			157		158	NC		
CAM1_DP0			159		160	NC		
CAM1_DN0			161		162	NC		
GND			163		164	GND		
USB_DP			165		166	TVDAC		
USB_DM			167		168	USB_OTGID		
GND			169		170	GND		
HDMI_CEC			171		172	VC_TRST_N		
HDMI_SDA			173		174	VC_TDI		
HDMI_SCL			175		176	VC_TMS		
RUN			177		178	VC_TDO		
VDD_CORE (DO NOT CONNECT)			179		180	VC_TCK		
GND			181		182	GND		
1V8			183		184	1V8		
1V8			185		186	1V8		
GND			187		188	GND		
VDAC			189		190	VDAC		
3V3			191		192	3V3		
3V3			193		194	3V3		
GND			195		196	GND		
VBAT			197		198	VBAT		
VBAT			199		200	VBAT		

Table 2: Compute Module SODIMM Connector Pinout

Table 2 gives the Compute Module pinout and Table 3 gives the Compute Module pin functions.

Pin Name	DIR	Voltage Ref	PDN[a] State	If Unused	Description/Notes
RUN and Boot Control (see text for usage guide)					
RUN	I	3V3[b]	Pull High	Leave open	Has internal 10k pull up
EMMC_DISABLE_N	I	3V3[b]	Pull High	Leave open	Has internal 10k pull up
EMMC_EN_N_1V8	O	1V8	Pull High	Leave open	Has internal 2k2 pull up
GPIO					
GPIO[27:0]	I/O	GPIO0-27_VDD	Pull or Hi-Z[c]	Leave open	GPIO Bank 0
GPIO[45:28]	I/O	GPIO28-45_VDD	Pull or Hi-Z[c]	Leave open	GPIO Bank 1
Primary SD Interface[d,e]					
SDX_CLK	O	SDX_VDD	Pull High	Leave open	Primary SD interface CLK
SDX_CMD	I/O	SDX_VDD	Pull High	Leave open	Primary SD interface CMD
SDX_Dx	I/O	SDX_VDD	Pull High	Leave open	Primary SD interface DATA
USB Interface					
USB_Dx	I/O	-	Z	Leave open	Serial interface
USB_OTGID	I	3V3		Tie to GND	OTG pin detect
HDMI Interface					
HDMI_SCL	I/O	3V3[b]	Z[f]	Leave open	DDC Clock (5.5V tolerant)
HDMI_SDA	I/O	3V3[b]	Z[f]	Leave open	DDC Data (5.5V tolerant)
HDMI_CEC	I/O	3V3	Z	Leave open	CEC (has internal 27k pull up)
HDMI_CLKx	O	-	Z	Leave open	HDMI serial clock
HDMI_Dx	O	-	Z	Leave open	HDMI serial data
HDMI_HPD_N_1V8	I	1V8	Pull High	Leave open	HDMI hotplug detect
CAM0 (CSI0) 2-lane Interface					
CAM0_Cx	I	-	Z	Leave open	Serial clock
CAM0_Dx	I	-	Z	Leave open	Serial data
CAM1 (CSI1) 4-lane Interface					
CAM1_Cx	I	-	Z	Leave open	Serial clock
CAM1_Dx	I	-	Z	Leave open	Serial data
DSI0 (Display 0) 2-lane Interface					
DSI0_Cx	O	-	Z	Leave open	Serial clock
DSI0_Dx	O	-	Z	Leave open	Serial data
DSI1 (Display 1) 4-lane Interface					
DSI1_Cx	O	-	Z	Leave open	Serial clock
DSI1_Dx	O	-	Z	Leave open	Serial data
TV Out					
TVDAC	O	-	Z	Leave open	Composite video DAC output
JTAG Interface					
TMS	I	3V3	Z	Leave open	Has internal 50k pull up
TRST_N	I	3V3	Z	Leave open	Has internal 50k pull up
TCK	I	3V3	Z	Leave open	Has internal 50k pull up
TDI	I	3V3	Z	Leave open	Has internal 50k pull up
TDO	O	3V3	O	Leave open	Has internal 50k pull up

[a] The PDN column indicates power-down state (when RUN pin LOW)
[b] Must be driven by an open-collector driver
[c] GPIO have software enabled pulls which keep state over power-down
[d] Only available on Lite variants
[e] The CM will always try to boot from this interface first
[f] Requires external pull-up resistor to 5V as per HDMI spec

Table 3: Pin Functions

6 Electrical Specification

Caution! Stresses above those listed in Table 4 may cause permanent damage to the device. This is a stress rating only; functional operation of the device under these or any other conditions above those listed in the operational sections of this specification is not implied. Exposure to absolute maximum rating conditions for extended periods may affect device reliability.

Symbol	Parameter	Minimum	Maximum	Unit
VBAT	Core SMPS Supply	-0.5	6.0	V
3V3	3V3 Supply Voltage	-0.5	4.10	V
1V8	1V8 Supply Voltage	-0.5	2.10	V
VDAC	TV DAC Supply	-0.5	4.10	V
GPIO0-27_VDD	GPIO0-27 I/O Supply Voltage	-0.5	4.10	V
GPIO28-45_VDD	GPIO28-27 I/O Supply Voltage	-0.5	4.10	V
SDX_VDD	Primary SD/eMMC Supply Voltage	-0.5	4.10	V

Table 4: Absolute Maximum Ratings

DC Characteristics are defined in Table 5

Symbol	Parameter	Conditions	Minimum	Typical	Maximum	Unit
V_{IL}	Input low voltage[a]	VDD_IO = 1.8V	-	-	0.6	V
		VDD_IO = 2.7V	-	-	0.8	V
V_{IH}	Input high voltage[a]	VDD_IO = 1.8V	1.0	-	-	V
		VDD_IO = 2.7V	1.3	-	-	V
I_{IL}	Input leakage current	TA = +85°C	-	-	5	μA
C_{IN}	Input capacitance	-	-	5	-	pF
V_{OL}	Output low voltage[b]	VDD_IO = 1.8V, IOL = -2mA	-	-	0.2	V
		VDD_IO = 2.7V, IOL = -2mA	-	-	0.15	V
V_{OH}	Output high voltage[b]	VDD_IO = 1.8V, IOH = 2mA	1.6	-	-	V
		VDD_IO = 2.7V, IOH = 2mA	2.5	-	-	V
I_{OL}	Output low current[c]	VDD_IO = 1.8V, VO = 0.4V	12	-	-	mA
		VDD_IO = 2.7V, VO = 0.4V	17	-	-	mA
I_{OH}	Output high current[c]	VDD_IO = 1.8V, VO = 1.4V	10	-	-	mA
		VDD_IO = 2.7V, VO = 2.3V	16	-	-	mA
R_{PU}	Pullup resistor	-	50	-	65	kΩ
R_{PD}	Pulldown resistor	-	50	-	65	kΩ

[a] Hysteresis enabled
[b] Default drive strength (8mA)
[c] Maximum drive strength (16mA)

Table 5: DC Characteristics

AC Characteristics are defined in Table 6 and Fig. 5.

Pin Name	Symbol	Parameter	Minimum	Typical	Maximum	Unit
Digital outputs	t_{rise}	10-90% rise time[a]	-	1.6	-	ns
Digital outputs	t_{fall}	90-10% fall time[a]	-	1.7	-	ns
GPCLK	t_{JOSC}	Oscillator-derived GPCLK cycle-cycle jitter (RMS)	-	-	20	ps
GPCLK	t_{JPLL}	PLL-derived GPCLK cycle-cycle jitter (RMS)	-	-	48	ps

[a] Default drive strength, CL = 5pF, VDD_IOx = 3.3V

Table 6: Digital I/O Pin AC Characteristics

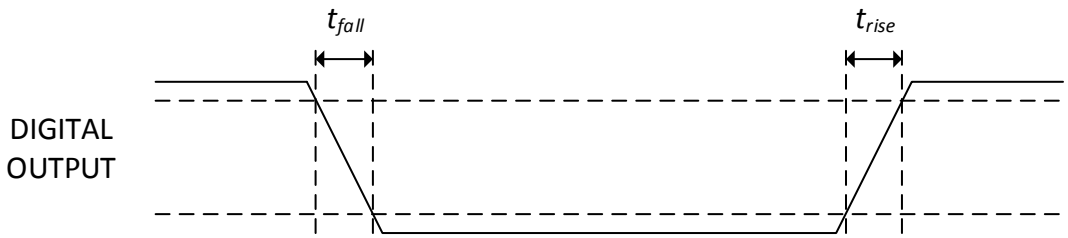


Figure 5: Digital IO Characteristics

7 Power Supplies

The Compute Module has six separate supplies that must be present and powered at all times; you cannot leave any of them unpowered, even if a specific interface or GPIO bank is unused. The six supplies are as follows:

1. VBAT is used to power the BCM283x processor core. It feeds the SMPS that generates the chip core voltage.
2. 3V3 powers various BCM283x PHYs, IO and the eMMC Flash.
3. 1V8 powers various BCM283x PHYs, IO and SDRAM.
4. VDAC powers the composite (TV-out) DAC.
5. GPIO0-27_VREF powers the GPIO 0-27 IO bank.
6. GPIO28-45_VREF powers the GPIO 28-45 IO bank.

Supply	Descripion	Minimum	Typical	Maximum	Unit
VBAT	Core SMPS Supply	2.5	-	5.0 + 5%	V
3V3	3V3 Supply Voltage	3.3 - 5%	3.3	3.3 + 5%	V
1V8	1V8 Supply Voltage	1.8 - 5%	1.8	1.8 + 5%	V
VDAC	TV DAC Supply[a]	2.5 - 5%	2.8	3.3 + 5%	V
GPIO0-27_VDD	GPIO0-27 I/O Supply Voltage	1.8 - 5%	-	3.3 + 5%	V
GPIO28-45_VDD	GPIO28-27 I/O Supply Voltage	1.8 - 5%	-	3.3 + 5%	V
SDX_VDD	Primary SD/eMMC Supply Voltage	1.8 - 5%	-	3.3 + 5%	V

[a] Requires a clean 2.5-2.8V supply if TV DAC is used, else connect to 3V3

Table 7: Power Supply Operating Ranges

7.1 Supply Sequencing

Supplies should be staggered so that the highest voltage comes up first, then the remaining voltages in descending order. This is to avoid forward biasing internal (on-chip) diodes between supplies, and causing latch-up. Alternatively supplies can be synchronised to come up at exactly the same time as long as at no point a lower voltage supply rail voltage exceeds a higher voltage supply rail voltage.

7.2 Power Requirements

Exact power requirements will be heavily dependent upon the individual use case. If an on-chip subsystem is unused, it is usually in a low power state or completely turned off. For instance, if your application does not use 3D graphics then a large part of the core digital logic will never turn on and need power. This is also the case for camera and display interfaces, HDMI, USB interfaces, video encoders and decoders, and so on.

Powerchain design is critical for stable and reliable operation of the Compute Module. We strongly recommend that designers spend time measuring and verifying power requirements for their particular use case and application, as well as paying careful attention to power supply sequencing and maximum supply voltage tolerance.

Table 8 specifies the recommneded minimum power supply outputs required to power the Compute Module.

Supply	Minimum Requirement	Unit
VBAT (CM1)	2000[a]	mW
VBAT (CM3,3L)	3500[a]	mW
3V3	250	mA
1V8	250	mA
VDAC	25	mA
GPIO0-27_VDD	50[b]	mA
GPIO28-45_VDD	50[b]	mA
SDX_VDD	50[b]	mA

[a] Recommended minimum. Actual power drawn is very dependent on use-case
[b] Each GPIO can supply up to 16mA, aggregate current per bank must not exceed 50mA

Table 8: Mimimum Power Supply Requirements

8 Booting

The 4GB eMMC Flash device on CM3 is directly connected to the primary BCM2837 SD/eMMC interface. These connections are not accessible on the module pins. On CM3L this SD interface is available on the SDX_ pins.

When initially powered on, or after the RUN pin has been held low and then released, the BCM2837 will try to access the primary SD/eMMC interface. It will then look for a file called bootcode.bin on the primary partition (which must be FAT) to start booting the system. If it cannot access the SD/eMMC device or the boot code cannot be found, it will fall back to waiting for boot code to be written to it over USB; in other words, its USB port is in slave mode waiting to accept boot code from a suitable host.

A USB boot tool is available on Github which allows a host PC running Linux to write the BCM2837 boot code over USB to the module. That boot code then runs and provides access to the SD/eMMC as a USB mass storage device, which can then be read and written using the host PC. Note that a Raspberry Pi can be used as the host machine. For those using Windows a precompiled and packeged tool is available. For more information see here.

The Compute Module has a pin called EMMC_DISABLE_N which when shorted to GND will disable the SD/eMMC interface (by physically disconnecting the SD_CMD pin), forcing BCM2837 to boot from USB. Note that when the eMMC is disabled in this way, it takes a couple of seconds from powering up for the processor to stop attempting to talk to the SD/eMMC device and fall back to booting from USB.

Note that once booted over USB, BCM2837 needs to re-enable the SD/eMMC device (by releasing EMMC_DISABLE_N) to allow access to it as mass storage. It expects to be able to do this by driving the EMMC_EN_N_1V8 pin LOW, which at boot is initially an input with a pull up to 1V8. If an end user wishes to add the ability to access the SD/eMMC over USB in their product, similar circuitry to that used on the Compute Module IO Board to enable/disable the USB boot and SD/eMMC must be used; that is, EMMC_DISABLE_N pulled low via MOSFET(s) and released again by MOSFET, with the gate controlled by EMMC_EN_N_1V8. Ensure you use MOSFETs suitable for switching at 1.8V (i.e. use a device with gate threshold voltage, Vt, suitable for 1.8V switching).

9 Peripherals

9.1 GPIO

BCM283x has in total 54 GPIO lines in 3 separate voltage banks. All GPIO pins have at least two alternative functions within the SoC. When not used for the alternate peripheral function, each GPIO pin may be set as an input (optionally as an interrupt) or an output. The alternate functions are usually peripheral I/Os, and most peripherals appear twice to allow flexibility on the choice of I/O voltage.

On CM1, CM3 and CM3L bank2 is used on the module to connect to the eMMC device and, on CM3 and CM3L, for an on-board I2C bus (to talk to the core SMPS and control the special function pins). On CM3L most of bank 2 is exposed to allow a user to connect their choice of SD card or eMMC device (if required).

Bank0 and 1 GPIOs are available for general use. GPIO0 to GPIO27 are bank 0 and GPIO28-45 make up bank1. GPIO0-27_VDD is the power supply for bank0 and GPIO28-45_VDD is the power supply for bank1. SDX_VDD is the supply for bank2 on CM3L. These supplies can be in the range 1.8V-3.3V (see Table 7) and are not optional; each bank must be powered, even when none of the GPIOs for that bank are used.

Note that the HDMI_HPD_N_1V8 and EMMC_EN_N_1V8 pins (on CM1 these were called GPIO46_1V8 and GPIO47_1V8 respectively) are 1.8V IO and are used for special functions (HDMI hot plug detect and boot control respectively). Please do not use these pins for any other purpose, as the software for the Compute Module will always expect these pins to have these special functions. If they are unused please leave them unconnected.

All GPIOs except GPIO28, 29, 44 and 45 have weak in-pad pull-ups or pull-downs enabled when the device is powered on. It is recommended to add off-chip pulls to GPIO28, 29, 44 and 45 to make sure they never float during power on and initial boot.

9.1.1 GPIO Alternate Functions

GPIO	Default Pull	ALT0	ALT1	ALT2	ALT3	ALT4	ALT5
0	High	SDA0	SA5	PCLK	-	-	-
1	High	SCL0	SA4	DE	-	-	-
2	High	SDA1	SA3	LCD_VSYNC	-	-	-
3	High	SCL1	SA2	LCD_HSYNC	-	-	-
4	High	GPCLK0	SA1	DPI_D0	-	-	ARM_TDI
5	High	GPCLK1	SA0	DPI_D1	-	-	ARM_TDO
6	High	GPCLK2	SOE_N	DPI_D2	-	-	ARM_RTCK
7	High	SPI0_CE1_N	SWE_N	DPI_D3	-	-	-
8	High	SPI0_CE0_N	SD0	DPI_D4	-	-	-
9	Low	SPI0_MISO	SD1	DPI_D5	-	-	-
10	Low	SPI0_MOSI	SD2	DPI_D6	-	-	-
11	Low	SPI0_SCLK	SD3	DPI_D7	-	-	-
12	Low	PWM0	SD4	DPI_D8	-	-	ARM_TMS
13	Low	PWM1	SD5	DPI_D9	-	-	ARM_TCK
14	Low	TXD0	SD6	DPI_D10	-	-	TXD1
15	Low	RXD0	SD7	DPI_D11	-	-	RXD1
16	Low	FL0	SD8	DPI_D12	CTS0	SPI1_CE2_N	CTS1
17	Low	FL1	SD9	DPI_D13	RTS0	SPI1_CE1_N	RTS1
18	Low	PCM_CLK	SD10	DPI_D14	-	SPI1_CE0_N	PWM0
19	Low	PCM_FS	SD11	DPI_D15	-	SPI1_MISO	PWM1
20	Low	PCM_DIN	SD12	DPI_D16	-	SPI1_MOSI	GPCLK0
21	Low	PCM_DOUT	SD13	DPI_D17	-	SPI1_SCLK	GPCLK1
22	Low	SD0_CLK	SD14	DPI_D18	SD1_CLK	ARM_TRST	-
23	Low	SD0_CMD	SD15	DPI_D19	SD1_CMD	ARM_RTCK	-
24	Low	SD0_DAT0	SD16	DPI_D20	SD1_DAT0	ARM_TDO	-
25	Low	SD0_DAT1	SD17	DPI_D21	SD1_DAT1	ARM_TCK	-
26	Low	SD0_DAT2	TE0	DPI_D22	SD1_DAT2	ARM_TDI	-
27	Low	SD0_DAT3	TE1	DPI_D23	SD1_DAT3	ARM_TMS	-

Table 9: GPIO Bank0 Alternate Functions

GPIO	Default Pull	ALT0	ALT1	ALT2	ALT3	ALT4	ALT5
28	None	SDA0	SA5	PCM_CLK	FL0	-	-
29	None	SCL0	SA4	PCM_FS	FL1	-	-
30	Low	TE0	SA3	PCM_DIN	CTS0	-	CTS1
31	Low	FL0	SA2	PCM_DOUT	RTS0	-	RTS1
32	Low	GPCLK0	SA1	RING_OCLK	TXD0	-	TXD1
33	Low	FL1	SA0	TE1	RXD0	-	RXD1
34	High	GPCLK0	SOE_N	TE2	SD1_CLK	-	-
35	High	SPI0_CE1_N	SWE_N	-	SD1_CMD	-	-
36	High	SPI0_CE0_N	SD0	TXD0	SD1_DAT0	-	-
37	Low	SPI0_MISO	SD1	RXD0	SD1_DAT1	-	-
38	Low	SPI0_MOSI	SD2	RTS0	SD1_DAT2	-	-
39	Low	SPI0_SCLK	SD3	CTS0	SD1_DAT3	-	-
40	Low	PWM0	SD4	-	SD1_DAT4	SPI2_MISO	TXD1
41	Low	PWM1	SD5	TE0	SD1_DAT5	SPI2_MOSI	RXD1
42	Low	GPCLK1	SD6	TE1	SD1_DAT6	SPI2_SCLK	RTS1
43	Low	GPCLK2	SD7	TE2	SD1_DAT7	SPI2_CE0_N	CTS1
44	None	GPCLK1	SDA0	SDA1	TE0	SPI2_CE1_N	-
45	None	PWM1	SCL0	SCL1	TE1	SPI2_CE2_N	-

Table 10: GPIO Bank1 Alternate Functions

Table 9 and Table 10 detail the default pin pull state and available alternate GPIO functions. Most of these alternate peripheral functions are described in detail in the Broadcom Peripherals Specification document and have Linux drivers available.

9.1.2 Secondary Memory Interface (SMI)

The SMI peripheral is an asynchronous NAND type bus supporting Intel mode80 type transfers at 8 or 16 bit widths and available in the ALT1 positions on GPIO banks 0 and 1 (see Table 9 and Table 10). It is not publicly documented in the Broadcom Peripherals Specification but a Linux driver is available in the Raspberry Pi Github Linux repository (bcm2835_smi.c in linux/drivers/misc).

9.1.3 Display Parallel Interface (DPI)

A standard parallel RGB (DPI) interface is available on bank 0 GPIOs. This up-to-24-bit parallel interface can support a secondary display. Again this interface is not documented in the Broadcom Peripherals Specification but documentation can be found here.

9.1.4 SD/SDIO Interface

The BCM283x supports two SD card interfaces, SD0 and SD1.

The first (SD0) is a proprietary Broadcom controller that does not support SDIO and is the primary interface used to boot and talk to the eMMC or SDX_x signals.

The second interface (SD1) is standards compliant and can interface to SD, SDIO and eMMC devices; for example on a Raspberry Pi 3 it is used to talk to the on-board BCM43438 WiFi device in SDIO mode.

Both interfaces can support speeds up to 50MHz single ended (SD High Speed Mode).

9.2 CSI (MIPI Serial Camera)

Currently the CSI interface is not openly documented and only CSI camera sensors supported by the official Raspberry Pi firmware will work with this interface. Supported sensors are the OmniVision OV5647 and Sony IMX219.

It is recommended to attach other cameras via USB.

9.3 DSI (MIPI Serial Display)

Currently the DSI interface is not openly documented and only DSI displays supported by the official Raspberry Pi firmware will work with this interface.

Displays can also be added via the parallel DPI interface which is available as a GPIO alternate function - see Table 9 and Section 9.1.3

9.4 USB

The BCM283x USB port is On-The-Go (OTG) capable. If using either as a fixed slave or fixed master, please tie the USB_OTGID pin to ground.

The USB port (Pins USB_DP and USB_DM) must be routed as 90 ohm differential PCB traces.

Note that the port is capable of being used as a true OTG port however there is no official documentation. Some users have had success making this work.

9.5 HDMI

BCM283x supports HDMI V1.3a.

It is recommended that users follow a similar arrangement to the Compute Module IO Board circuitry for HDMI output.

The HDMI CK_P/N (clock) and D0-D2_P/N (data) pins must each be routed as matched length 100 ohm differential PCB traces. It is also important to make sure that each differential pair is closely phase matched. Finally, keep HDMI traces well away from other noise sources and as short as possible.

Failure to observe these design rules is likely to result in EMC failure.

9.6 Composite (TV Out)

The TVDAC pin can be used to output composite video (PAL or NTSC). Please route this signal away from noise sources and use a 75 ohm PCB trace.

Note that the TV DAC is powered from the VDAC supply which must be a clean supply of 2.5-2.8V. It is recommended users generate this supply from 3V3 using a low noise LDO.

If the TVDAC output is not used VDAC can be connected to 3V3, but it must be powered even if the TV-out functionality is unused.

10 Thermals

The BCM283x SoC employs DVFS (Dynamic Voltage and Frequency Scaling) on the core voltage. When the processor is idle (low CPU utilisation), it will reduce the core frequency and voltage to reduce current draw and heat output. When the core utilisation exceeds a certain threshold the core votlage is increased and the core frequency is boosted to the maximum working frerquency. The voltage and frequency are throttled back when the CPU load reduces back to an 'idle' level OR when the silicon temperature as mesured by the on-chip temperature sensor exceeds 85C (thermal throttling).

A designer must pay careful attention to the thermal design of products using the CM3/CM3L so that performance is not artificially curtailed due to the processor thermal throttling, as the Quad ARM complex in the BCM2837 can generate significant heat output.

10.1 Temperature Range

The operating temperature range of the module is set by the lowest maximum and highest minimum of any of the components used.

The eMMC and LPDDR2 have the narrowest range, these are rated for -25 to +80 degrees Celsius. Therefore the nominal range for the CM3 and CM3L is -25C to +80C.

However, this range is the maximum for the silicon die; therefore, users would have to take into account the heat generated when in use and make sure this does not cause the temperature to exceed 80 degrees Celsius.

11 Availability

Raspberry Pi guarantee availability of CM1, CM3 and CM3 Lite until at least January 2023.

12 Support

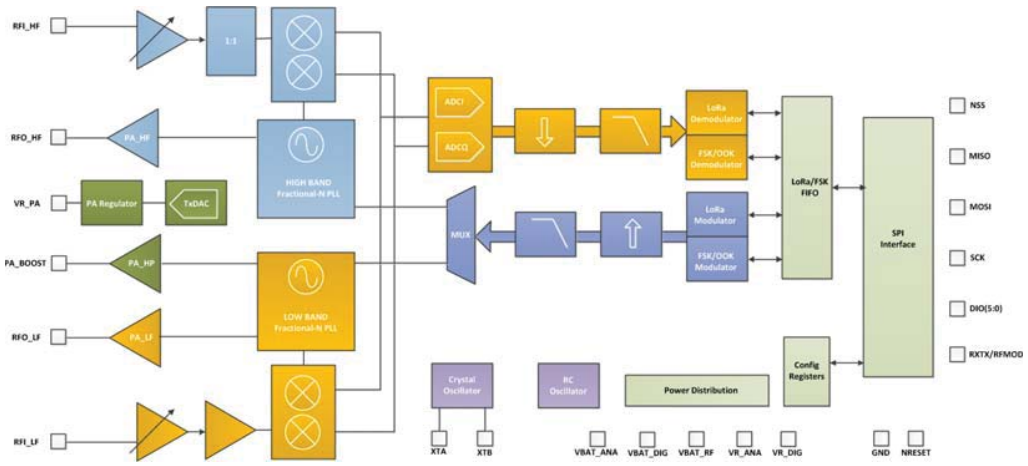
For support please see the hardware documentation section of the Raspberry Pi website and post questions to the Raspberry Pi forum.

DATASHEET SEMTECH SX1276

SX1276/77/78/79

WIRELESS, SENSING & TIMING **DATASHEET**

SX1276/77/78/79 - 137 MHz to 1020 MHz Low Power Long Range Transceiver



GENERAL DESCRIPTION

The SX1276/77/78/79 transceivers feature the LoRa™ long range modem that provides ultra-long range spread spectrum communication and high interference immunity whilst minimising current consumption.

Using Semtech's patented LoRa™ modulation technique SX1276/77/78/79 can achieve a sensitivity of over -148dBm using a low cost crystal and bill of materials. The high sensitivity combined with the integrated +20 dBm power amplifier yields industry leading link budget making it optimal for any application requiring range or robustness. LoRa™ provides significant advantages in both blocking and selectivity over conventional modulation techniques, solving the traditional design compromise between range, interference immunity and energy consumption.

These devices also support high performance (G)FSK modes for systems including WMBus, IEEE802.15.4g. The SX1276/77/78/79 deliver exceptional phase noise, selectivity, receiver linearity and IIP3 for significantly lower current consumption than competing devices.

ORDERING INFORMATION

Part Number	Delivery	MOQ / Multiple
SX1276IMLTRT	T&R	3000 pieces
SX1277IMLTRT	T&R	3000 pieces
SX1278IMLTRT	T&R	3000 pieces
SX1279IMLTRT	T&R	3000 pieces
SX1276WS	Wafer Form	1 Wafer (2000 dies)

- QFN 28 Package - Operating Range [-40;+85°C]
- Pb-free, Halogen free, RoHS/WEEE compliant product

KEY PRODUCT FEATURES

- LoRa™ Modem
- 168 dB maximum link budget
- +20 dBm - 100 mW constant RF output vs. V supply
- +14 dBm high efficiency PA
- Programmable bit rate up to 300 kbps
- High sensitivity: down to -148 dBm
- Bullet-proof front end: IIP3 = -11 dBm
- Excellent blocking immunity
- Low RX current of 9.9 mA, 200 nA register retention
- Fully integrated synthesizer with a resolution of 61 Hz
- FSK, GFSK, MSK, GMSK, LoRa™and OOK modulation
- Built-in bit synchronizer for clock recovery
- Preamble detection
- 127 dB Dynamic Range RSSI
- Automatic RF Sense and CAD with ultra-fast AFC
- Packet engine up to 256 bytes with CRC
- Built-in temperature sensor and low battery indicator

APPLICATIONS

- Automated Meter Reading.
- Home and Building Automation.
- Wireless Alarm and Security Systems.
- Industrial Monitoring and Control
- Long range Irrigation Systems

Table of contents

Section **Page**

Table of contents

Section **Page**

Table of contents

Section **Page**

Table of contents

Section **Page**

Table of contents

Section **Page**

Table of contents

Section **Page**

Table of contents

Section **Page**

1. General Description

The SX1276/77/78/79 incorporates the LoRa™ spread spectrum modem which is capable of achieving significantly longer range than existing systems based on FSK or OOK modulation. At maximum data rates of LoRa™ the sensitivity is 8dB better than FSK, but using a low cost bill of materials with a 20ppm XTAL LoRa™ can improve receiver sensitivity by more than 20dB compared to FSK. LoRa™ also provides significant advances in selectivity and blocking performance, further improving communication reliability. For maximum flexibility the user may decide on the spread spectrum modulation bandwidth (BW), spreading factor (SF) and error correction rate (CR). Another benefit of the spread modulation is that each spreading factor is orthogonal - thus multiple transmitted signals can occupy the same channel without interfering. This also permits simple coexistence with existing FSK based systems. Standard GFSK, FSK, OOK, and GMSK modulation is also provided to allow compatibility with existing systems or standards such as wireless MBUS and IEEE 802.15.4g.

The SX1276 and SX1279 offer bandwidth options ranging from 7.8 kHz to 500 kHz with spreading factors ranging from 6 to 12, and covering all available frequency bands. The SX1277 offers the same bandwidth and frequency band options with spreading factors from 6 to 9. The SX1278 offers bandwidths and spreading factor options, but only covers the lower UHF bands.

1.1. Simplified Block Diagram

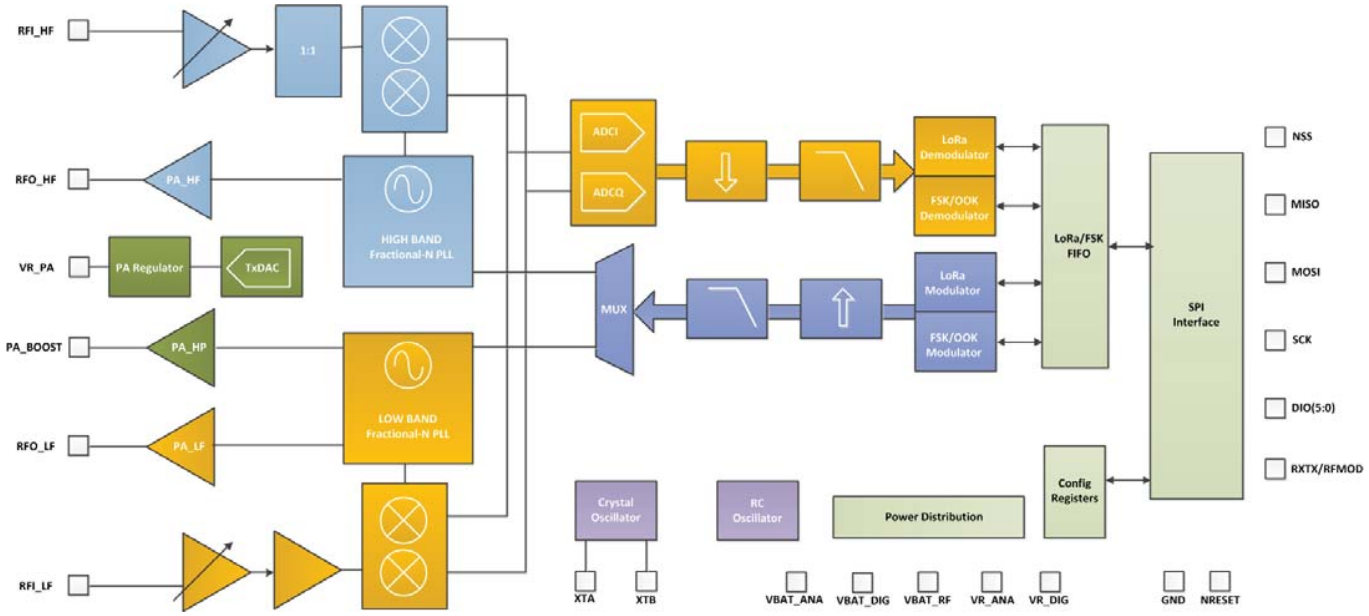


Figure 1. Block Diagram

1.2. Product Versions

The features of the four product variants are detailed in the following table.

Table 1 SX1276/77/78/79 Device Variants and Key Parameters

Part Number	Frequency Range	Spreading Factor	Bandwidth	Effective Bitrate	Est. Sensitivity
SX1276	137 - 1020 MHz	6 - 12	7.8 - 500 kHz	.018 - 37.5 kbps	-111 to -148 dBm
SX1277	137 - 1020 MHz	6 - 9	7.8 - 500 kHz	0.11 - 37.5 kbps	-111 to -139 dBm
SX1278	137 - 525 MHz	6- 12	7.8 - 500 kHz	.018 - 37.5 kbps	-111 to -148 dBm
SX1279	137 - 960MHz	6- 12	7.8 - 500 kHz	.018 - 37.5 kbps	-111 to -148 dBm

1.3. Pin Diagram

The following diagram shows the pin arrangement of the QFN package, top view.

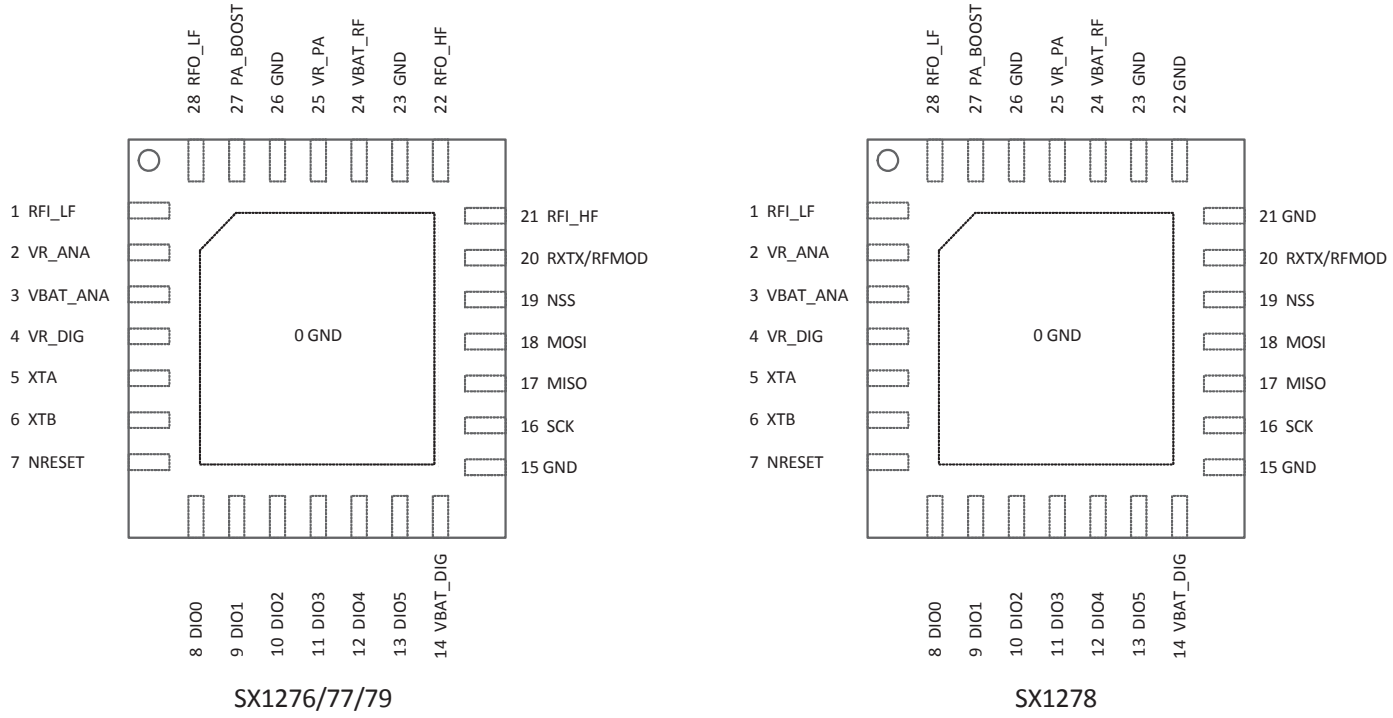


Figure 2. Pin Diagrams

1.4. Pin Description

Table 2 Pin Description

Number	Name	Type	Description
	SX1276/77/79/(78)	SX1276/77/79/(78)	SX1276/77/79/(78)
0	GROUND	-	Exposed ground pad
1	RFI_LF	I	RF input for bands 2&3
2	VR_ANA	-	Regulated supply voltage for analogue circuitry
3	VBAT_ANA	-	Supply voltage for analogue circuitry
4	VR_DIG	-	Regulated supply voltage for digital blocks
5	XTA	I/O	XTAL connection or TCXO input
6	XTB	I/O	XTAL connection
7	NRESET	I/O	Reset trigger input
8	DIO0	I/O	Digital I/O, software configured
9	DIO1/DCLK	I/O	Digital I/O, software configured
10	DIO2/DATA	I/O	Digital I/O, software configured
11	DIO3	I/O	Digital I/O, software configured
12	DIO4	I/O	Digital I/O, software configured
13	DIO5	I/O	Digital I/O, software configured
14	VBAT_DIG	-	Supply voltage for digital blocks
15	GND	-	Ground
16	SCK	I	SPI Clock input
17	MISO	O	SPI Data output
18	MOSI	I	SPI Data input
19	NSS	I	SPI Chip select input
20	RXTX/RF_MOD	O	Rx/Tx switch control: high in Tx
21	RFI_HF (GND)	I (-)	RF input for band 1 (Ground)
22	RFO_HF (GND)	O (-)	RF output for band 1 (Ground)
23	GND	-	Ground
24	VBAT_RF	-	Supply voltage for RF blocks
25	VR_PA	-	Regulated supply for the PA
26	GND	-	Ground
27	PA_BOOST	O	Optional high-power PA output, all frequency bands
28	RFO_LF	O	RF output for bands 2&3

1.5. Package Marking

TOP MARK	
CHAR	ROWS
7/7/7/7/7	5

Marking for the 6 x 6 mm MLPQ 28ld Lead package:

nnnnnn = Part Number (Example: SX1276)
yyww = Date Code (Example: 1352)
xxxxxxx = Semtech Lot No. (Example: EA90101)
xxxxxxx 0101-10)

Figure 3. Marking Diagram

2. Electrical Characteristics

2.1. ESD Notice

The SX1276/77/78/79 is a high performance radio frequency device. It satisfies:

- Class 2 of the JEDEC standard JESD22-A114 (Human Body Model) on all pins.
- Class III of the JEDEC standard JESD22-C101 (Charged Device Model) on all pins

It should thus be handled with all the necessary ESD precautions to avoid any permanent damage.

2.2. Absolute Maximum Ratings

Stresses above the values listed below may cause permanent device failure. Exposure to absolute maximum ratings for extended periods may affect device reliability.

Table 3 Absolute Maximum Ratings

Symbol	Description	Min	Max	Unit
VDDmr	Supply Voltage	-0.5	3.9	V
Tmr	Temperature	-55	+115	° C
Tj	Junction temperature	-	+125	° C
Pmr	RF Input Level	-	+10	dBm

Note Specific ratings apply to +20 dBm operation (see Section 5.4.3).

2.3. Operating Range

Table 4 Operating Range

Symbol	Description	Min	Max	Unit
VDDop	Supply voltage	1.8	3.7	V
Top	Operational temperature range	-40	+85	°C
Clop	Load capacitance on digital ports	-	25	pF
ML	RF Input Level	-	+10	dBm

Note A specific supply voltage range applies to +20 dBm operation (see Section 5.4.3).

2.4. Thermal Properties

Table 5 Thermal Properties

Symbol	Description	Min	Typ	Max	Unit
THETA_JA	Package θ_{ja} (Junction to ambient)	-	22.185	-	°C/W
THETA_JC	Package θ_{jc} (Junction to case ground paddle)	-	0.757	-	°C/W

2.5. Chip Specification

The tables below give the electrical specifications of the transceiver under the following conditions: Supply voltage VDD=3.3 V, temperature = 25 °C, *FXOSC* = 32 MHz, F_{RF} = 169/434/868/915 MHz (see specific indication), Pout = +13dBm, 2-level FSK modulation without pre-filtering, FDA = 5 kHz, Bit Rate = 4.8 kb/s and terminated in a matched 50 Ohm impedance, shared Rx and Tx path matching, unless otherwise specified.

Note *Specification whose symbol is appended with "_LF" corresponds to the performance in Band 2 and/or Band 3, as described in section 5.3.3. "_HF" refers to the upper Band 1*

2.5.1. Power Consumption

Table 6 Power Consumption Specification

Symbol	Description	Conditions	Min	Typ	Max	Unit
IDDSL	Supply current in Sleep mode		-	0.2	1	uA
IDDIDLE	Supply current in Idle mode	RC oscillator enabled	-	1.5	-	uA
IDDST	Supply current in Standby mode	Crystal oscillator enabled	-	1.6	1.8	mA
IDDFS	Supply current in Synthesizer mode	FSRx	-	5.8	-	mA
IDDR	Supply current in Receive mode	*LnaBoost* Off, band 1 *LnaBoost* On, band 1 Bands 2&3	- - -	10.8 11.5 12.0	- - -	mA
IDDT	Supply current in Transmit mode with impedance matching	RFOP = +20 dBm, on PA_BOOST RFOP = +17 dBm, on PA_BOOST RFOP = +13 dBm, on RFO_LF/HF pin RFOP = + 7 dBm, on RFO_LF/HF pin	- - - -	120 87 29 20	- - - -	mA mA mA mA

2.5.2. Frequency Synthesis

Table 7 Frequency Synthesizer Specification

Symbol	Description	Conditions	Min	Typ	Max	Unit
FR	Synthesizer frequency range	Programmable (*for SX1279) Band 3 Band 2 Band 1	137 410 862 (*779)	- - -	175 (*160) 525 (*480) 1020 (*960)	MHz
FXOSC	Crystal oscillator frequency		-	32	-	MHz
TS_OSC	Crystal oscillator wake-up time		-	250	-	us
TS_FS	Frequency synthesizer wake-up time to PllLock signal	From Standby mode	-	60	-	us

TS_HOP	Frequency synthesizer hop time at most 10 kHz away from the target frequency	200 kHz step 1 MHz step 5 MHz step 7 MHz step 12 MHz step 20 MHz step 25 MHz step	- - - - - - -	20 20 50 50 50 50 50	- - - - - - -	us us us us us us us
FSTEP	Frequency synthesizer step	FSTEP = FXOSC/2^{19}	-	61.0	-	Hz
FRC	RC Oscillator frequency	After calibration	-	62.5	-	kHz
BRF	Bit rate, FSK	Programmable values (1)	1.2	-	300	kbps
BRA	Bit rate Accuracy, FSK	ABS(wanted BR - available BR)	-	-	250	ppm
BRO	Bit rate, OOK	Programmable	1.2	-	32.768	kbps
BR_L	Bit rate, LoRa Mode	From SF6, BW=500kHz to SF12, BW=7.8kHz	0.018	-	37.5	kbps
FDA	Frequency deviation, FSK (1)	Programmable FDA + BRF/2 =< 250 kHz	0.6	-	200	kHz

Note: For Maximum Bit rate, the maximum modulation index is 0.5.

2.5.3. FSK/OOK Mode Receiver

All receiver tests are performed with RxBw = 10 kHz (Single Side Bandwidth) as programmed in *RegRxBw*, receiving a PN15 sequence. Sensitivities are reported for a 0.1% BER (with Bit Synchronizer enabled), unless otherwise specified. Blocking tests are performed with an unmodulated interferer. The wanted signal power for the Blocking Immunity, ACR, IIP2, IIP3 and AMR tests is set 3 dB above the receiver sensitivity level.

Table 8 FSK/OOK Receiver Specification

Symbol	Description	Conditions	Min	Typ	Max	Unit
RFS_F_LF	Direct tie of RFI and RFO pins, shared Rx, Tx paths FSK sensitivity, highest LNA gain. Bands 2&3	FDA = 5 kHz, BR = 1.2 kb/s FDA = 5 kHz, BR = 4.8 kb/s FDA = 40 kHz, BR = 38.4 kb/s* FDA = 20 kHz, BR = 38.4 kb/s** FDA = 62.5 kHz, BR = 250 kb/s***	- - - - -	-121 -117 -107 -108 -95	- - - - -	dBm dBm dBm dBm dBm
	Split RF paths, the RF switch insertion loss is not accounted for. Bands 2&3	FDA = 5 kHz, BR = 1.2 kb/s FDA = 5 kHz, BR = 4.8 kb/s FDA = 40 kHz, BR = 38.4 kb/s* FDA = 20 kHz, BR = 38.4 kb/s** FDA = 62.5 kHz, BR = 250 kb/s***	- - - - -	-123 -119 -109 -110 -97	- - - - -	dBm dBm dBm dBm dBm
RFS_F_HF	Direct tie of RFI and RFO pins, shared Rx, Tx paths FSK sensitivity, highest LNA gain. Band 1	FDA = 5 kHz, BR = 1.2 kb/s FDA = 5 kHz, BR = 4.8 kb/s FDA = 40 kHz, BR = 38.4 kb/s* FDA = 20 kHz, BR = 38.4 kb/s** FDA = 62.5 kHz, BR = 250 kb/s***	- - - - -	-119 -115 -105 -105 -92	- - - - -	dBm dBm dBm dBm dBm
	Split RF paths, *LnaBoost* is turned on, the RF switch insertion loss is not accounted for. Band 1	FDA = 5 kHz, BR = 1.2 kb/s FDA = 5 kHz, BR = 4.8 kb/s FDA = 40 kHz, BR = 38.4 kb/s* FDA = 20 kHz, BR = 38.4 kb/s** FDA = 62.5 kHz, BR = 250 kb/s***	- - - - -	-123 -119 -109 -109 -96	- - - - -	dBm dBm dBm dBm dBm
RFS_O	OOK sensitivity, highest LNA gain shared Rx, Tx paths	BR = 4.8 kb/s BR = 32 kb/s	- -	-117 -108	- -	dBm dBm
CCR	Co-Channel Rejection, FSK		-	-9	-	dB
ACR	Adjacent Channel Rejection	FDA = 5 kHz, BR=4.8kb/s Offset = +/- 25 kHz or +/- 50kHz Band 1 Band 2 Band 3	 - - -	 50 56 60	 - - -	 dB dB dB
BI_HF	Blocking Immunity, Band 1	Offset = +/- 1 MHz Offset = +/- 2 MHz Offset = +/- 10 MHz	- - -	71 76 84	- - -	dB dB dB
BI_LF	Blocking Immunity, Bands 2&3	Offset = +/- 1 MHz Offset = +/- 2 MHz Offset = +/- 10 MHz	- - -	71 72 78	- - -	dB dB dB

IIP2	2nd order Input Intercept Point Unwanted tones are 20 MHz above the LO	Highest LNA gain	-	+55	-	dBm
IIP3_HF	3rd order Input Intercept point Unwanted tones are 1MHz and 1.995 MHz above the LO	Band 1 Highest LNA gain G1 LNA gain G2, 5dB sensitivity hit	- -	-11 -6	- -	dBm dBm
IIP3_LF	3rd order Input Intercept point Unwanted tones are 1MHz and 1.995 MHz above the LO	Band 2 Highest LNA gain G1 LNA gain G2, 2.5dB sensitivity hit	- -	-22 -15	- -	dBm dBm
		Band 3 Highest LNA gain G1 LNA gain G2, 2.5dB sensitivity hit	- -	-15 -11	- -	dBm dBm
BW_SSB	Single Side channel filter BW	Programmable	2.7	-	250	kHz
IMR	Image Rejection	Wanted signal 3dB over sensitivity BER=0.1%	-	50	-	dB
IMA	Image Attenuation		-	57	-	dB
DR_RSSI	RSSI Dynamic Range	AGC enabled Min Max	- -	-127 0	- -	dBm dBm

* *RxBw = 83 kHz (Single Side Bandwidth)*

** *RxBw = 50 kHz (Single Side Bandwidth)*

*** *RxBw = 250 kHz (Single Side Bandwidth)*

2.5.4. FSK/OOK Mode Transmitter

Table 9 Transmitter Specification

Symbol	Description	Conditions	Min	Typ	Max	Unit
RF_OP	RF output power in 50 ohms on RFO pin (High efficiency PA).	Programmable with steps Max Min	 - -	 +14 -1	 - -	 dBm dBm
ΔRF_OP_V	RF output power stability on RFO pin versus voltage supply.	VDD = 2.5 V to 3.3 V VDD = 1.8 V to 3.7 V	- -	3 8	- -	dB dB
RF_OPH	RF output power in 50 ohms, on PA_BOOST pin (Regulated PA).	Programmable with 1dB steps Max Min	- -	+17 +2	- -	dBm dBm
RF_OPH_MAX	Max RF output power, on PA_BOOST pin	High power mode	-	+20	-	dBm
ΔRF_OPH_V	RF output power stability on PA_-BOOST pin versus voltage supply.	VDD = 2.4 V to 3.7 V	-	+/-1	-	dB
ΔRF_T	RF output power stability versus temperature on PA_BOOST pin.	From T = -40 °C to +85 °C	-	+/-1	-	dB

PHN	Transmitter Phase Noise	169 MHz, Band 3 10kHz Offset 50kHz Offset 400kHz Offset 1MHz Offset	 - - - -	 -118 -118 -128 -134	 - - - -	 dBc/Hz
		433 MHz, Band 2 10kHz Offset 50kHz Offset 400kHz Offset 1MHz Offset	 - - - -	 -110 -110 -122 -129	 - - - -	 dBc/Hz
		868/915 MHz, Band 1 10kHz Offset 50kHz Offset 400kHz Offset 1MHz Offset	 - - - -	 -103 -103 -115 -122	 - - - -	 dBc/Hz
ACP	Transmitter adjacent channel power (measured at 25 kHz offset)	BT=1. Measurement conditions as defined by EN 300 220-1 V2.3.1	-	-	-37	dBm
TS_TR	Transmitter wake up time, to the first rising edge of DCLK	Frequency Synthesizer enabled, *PaRamp* = 10us, BR = 4.8 kb/s	-	120	-	us

2.5.5. Electrical Specification for LoRa™ Modulation

The table below gives the electrical specifications for the transceiver operating with LoRa™ modulation. Following conditions apply unless otherwise specified:

- Supply voltage = 3.3 V
- Temperature = 25° C
- f_{XOSC} = 32 MHz
- bandwidth (BW) = 125 kHz
- Spreading Factor (SF) = 12
- Error Correction Code (EC) = 4/6
- Packet Error Rate (PER)= 1%
- CRC on payload enabled
- Output power = 13 dBm in transmission
- Payload length = 64 bytes
- Preamble Length = 12 symbols (programmed register *PreambleLength=8*)
- With matched impedances

Table 10 LoRa Receiver Specification

Symbol	Description	Conditions	Min.	Typ	Max	Unit
IDDR_L	Supply current in receiver LoRa™ mode, *LnaBoost* off	Bands 2&3, BW=7.8 to 62.5 kHz	-	11.0	-	mA
		Bands 2&3, BW = 125 kHz	-	11.5	-	mA
		Bands 2&3, BW = 250 kHz	-	12.4	-	mA
		Bands 2&3, BW = 500 kHz	-	13.8	-	mA
		Band 1, BW=7.8 to 62.5 kHz	-	9.9	-	mA
		Band 1, BW = 125 kHz	-	10.3	-	mA
		Band 1, BW = 250 kHz	-	11.1	-	mA
		Band 1, BW = 500 kHz	-	12.6	-	mA
IDDT_L	Supply current in transmitter mode	RFOP = 13 dBm	-	28	-	mA
		RFOP = 7 dBm	-	20	-	mA
IDDT_H_L	Supply current in transmitter mode with an external impedance transformation	Using PA_BOOST pin RFOP = 17 dBm	-	90	-	mA
BI_L	Blocking immunity, CW interferer	offset = +/- 1 MHz	-	89	-	dB
		offset = +/- 2 MHz		94		dB
		offset = +/- 10 MHz		100		dB
IIP2_L	2nd order Input Intercept Point Unwanted tones are 20 MHz above the LO	Highest LNA gain	-	+55	-	dBm
IIP3_L_HF	3rd order Input Intercept point Unwanted tones are 1MHz and 1.995 MHz above the LO	Band 1				
		Highest LNA gain G1	-	-11	-	dBm
		LNA gain G2, 5dB sensitivity hit	-	-6	-	dBm

Symbol	Description	Conditions	Min.	Typ	Max	Unit
IIP3_L_LF	3rd order Input Intercept point Unwanted tones are 1MHz and 1.995 MHz above the LO	Band 2 Highest LNA gain G1 LNA gain G2,2.5dB sensitivity hit	 - -	 -22 -15	 - -	 dBm dBm
RFS_L10_HF	RF sensitivity, Long-Range Mode, highest LNA gain, *LnaBoost* for Band 1, using split Rx/Tx path 10.4 kHz bandwidth	SF = 6 SF = 7 SF = 8 SF = 11	- - - -	-131 -134 -138 -146	- - - -	dBm dBm dBm dBm
RFS_L62_HF	RF sensitivity, Long-Range Mode, highest LNA gain, *LnaBoost* for Band 1, using split Rx/Tx path 62.5 kHz bandwidth	SF = 6 SF = 7 SF = 8 SF = 9 SF = 10 SF = 11 SF = 12	- - - - - - -	-121 -126 -129 -132 -135 -137 -139	- - - - - - -	dBm dBm dBm dBm dBm dBm dBm
RFS_L125_HF	RF sensitivity, Long-Range Mode, highest LNA gain, *LnaBoost* for Band 1, using split Rx/Tx path 125 kHz bandwidth	SF = 6 SF = 7 SF = 8 SF = 9 SF = 10 SF = 11 SF = 12	- - - - - - -	-118 -123 -126 -129 -132 -133 -136	- - - - - - -	dBm dBm dBm dBm dBm dBm dBm
RFS_L250_HF	RF sensitivity, Long-Range Mode, highest LNA gain, *LnaBoost* for Band 1, using split Rx/Tx path 250 kHz bandwidth	SF = 6 SF = 7 SF = 8 SF = 9 SF = 10 SF = 11 SF = 12	- - - - - - -	-115 -120 -123 -125 -128 -130 -133	- - - - - - -	dBm dBm dBm dBm dBm dBm dBm
RFS_L500_HF	RF sensitivity, Long-Range Mode, highest LNA gain, *LnaBoost* for Band 1, using split Rx/Tx path 500 kHz bandwidth	SF = 6 SF = 7 SF = 8 SF = 9 SF = 10 SF = 11 SF = 12	- - - - - - -	-111 -116 -119 -122 -125 -128 -130	- - - - - - -	dBm dBm dBm dBm dBm dBm dBm
RFS_L7.8_LF	RF sensitivity, Long-Range Mode, highest LNA gain, Band 2 or 3, using split Rx/Tx path 7.8 kHz bandwidth	SF = 12 SF = 11	- -	-148 -145	- -	dBm dBm
RFS_L10_LF	RF sensitivity, Long-Range Mode, highest LNA gain, Band 3, 10.4 kHz bandwidth	SF = 6 SF = 7 SF = 8	- - -	-132 -136 -138	- - -	dBm dBm dBm
RFS_L62_LF	RF sensitivity, Long-Range Mode, highest LNA gain, Band 3, 62.5 kHz bandwidth	SF = 6 SF = 7 SF = 8 SF = 9 SF = 10 SF = 11 SF = 12	- - - - - - -	-123 -128 -131 -134 -135 -137 -140	- - - - - - -	dBm dBm dBm dBm dBm dBm dBm

Symbol	Description	Conditions	Min.	Typ	Max	Unit
RFS_L125_LF	RF sensitivity, Long-Range Mode, highest LNA gain, Band 3, 125 kHz bandwidth	SF = 6 SF = 7 SF = 8 SF = 9 SF = 10 SF = 11 SF = 12	- - - - - - -	-121 -125 -128 -131 -134 -136 -137	- - - - - - -	dBm dBm dBm dBm dBm dBm dBm
RFS_L250_LF	RF sensitivity, Long-Range Mode, highest LNA gain, Band 3 250 kHz bandwidth	SF = 6 SF = 7 SF = 8 SF = 9 SF = 10 SF = 11 SF = 12	- - - - - - -	-118 -122 -125 -128 -131 -133 -134	- - - - - - -	dBm dBm dBm dBm dBm dBm dBm
RFS_L500_LF	RF sensitivity, Long-Range Mode, highest LNA gain, Band 3 500 kHz bandwidth	SF = 6 SF = 7 SF = 8 SF = 9 SF = 10 SF = 11 SF = 12	- - - - - - -	-112 -118 -121 -124 -127 -129 -130	- - - - - - -	dBm dBm dBm dBm dBm dBm dBm
CCR_LCW	Co-channel rejection Single CW tone = Sens +6 dB 1% PER	SF = 7 SF = 8 SF = 9 SF = 10 SF = 11 SF = 12	- - - - - -	5 9.5 12 14.4 17 19.5	- - - - - -	dB dB dB dB dB dB
CCR_LL	Co-channel rejection	Interferer is a LoRa™ signal using same BW and same SF. Pw = Sensitivity + 3 dB		-6		dB
ACR_LCW	Adjacent channel rejection	Interferer is 1.5*BW_L from the wanted signal center frequency 1% PER, Single CW tone = Sens + 3 dB SF = 7 SF = 12	 - -	 60 72	 - -	 dB dB
IMR_LCW	Image rejection after calibration.	1% PER, Single CW tone = Sens +3 dB	-	66	-	dB
FERR_L	Maximum tolerated frequency offset between transmitter and receiver, no sensitivity degradation, SF6 thru 12	All BW, +/-25% of BW The tighter limit applies (see below)		+/-25%		BW
	Maximum tolerated frequency offset between transmitter and receiver, no sensitivity degradation, SF10 thru 12	SF = 12 SF = 11 SF = 10	-50 -100 -200	- - -	50 100 200	ppm ppm ppm

2.5.6. Digital Specification

Conditions: Temp = 25° C, VDD = 3.3 V, FXOSC = 32 MHz, unless otherwise specified.

Table 11 Digital Specification

Symbol	Description	Conditions	Min	Typ	Max	Unit
V_{IH}	Digital input level high		0.8	-	-	VDD
V_{IL}	Digital input level low		-	-	0.2	VDD
V_{OH}	Digital output level high	Imax = 1 mA	0.9	-	-	VDD
V_{OL}	Digital output level low	Imax = -1 mA	-	-	0.1	VDD
F_{SCK}	SCK frequency		-	-	10	MHz
t_{ch}	SCK high time		50	-	-	ns
t_{cl}	SCK low time		50	-	-	ns
t_{rise}	SCK rise time		-	5	-	ns
t_{fall}	SCK fall time		-	5	-	ns
t_{setup}	MOSI setup time	From MOSI change to SCK rising edge.	30	-	-	ns
t_{hold}	MOSI hold time	From SCK rising edge to MOSI change.	20	-	-	ns
t_{nsetup}	NSS setup time	From NSS falling edge to SCK rising edge.	30	-	-	ns
t_{nhold}	NSS hold time	From SCK falling edge to NSS rising edge, normal mode.	100	-	-	ns
t_{nhigh}	NSS high time between SPI accesses		20	-	-	ns
T_DATA	DATA hold and setup time		250	-	-	ns

3. SX1276/77/78/79 Features

This section gives a high-level overview of the functionality of the SX1276/77/78/79 low-power, highly integrated transceiver. The following figure shows a simplified block diagram of the SX1276/77/78/79.

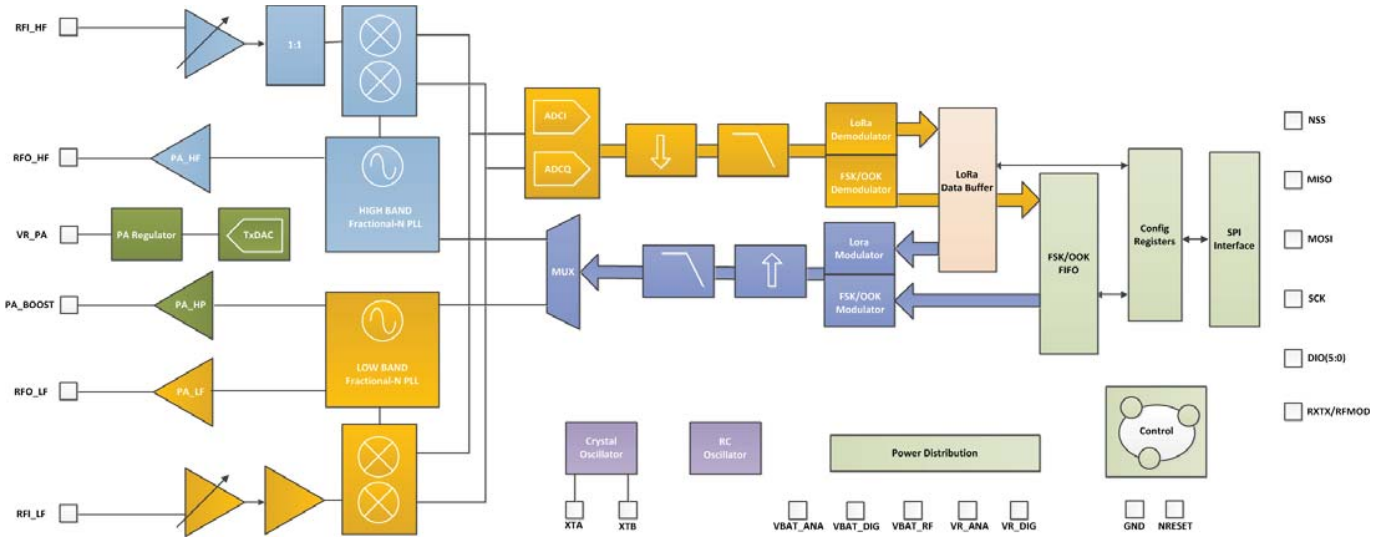


Figure 4. SX1276/77/78/79 Block Schematic Diagram

SX1276/77/78/79 is a half-duplex, low-IF transceiver. Here the received RF signal is first amplified by the LNA. The LNA inputs are single ended to minimize the external BoM and for ease of design. Following the LNA inputs, the conversion to differential is made to improve the second order linearity and harmonic rejection. The signal is then down-converted to in-phase and quadrature (I&Q) components at the intermediate frequency (IF) by the mixer stage. A pair of sigma delta ADCs then perform data conversion, with all subsequent signal processing and demodulation performed in the digital domain. The digital state machine also controls the automatic frequency correction (AFC), received signal strength indicator (RSSI) and automatic gain control (AGC). It also features the higher-level packet and protocol level functionality of the top level sequencer (TLS), only available with traditional FSK and OOK modulation schemes.

The frequency synthesizers generate the local oscillator (LO) frequency for both receiver and transmitter, one covering the lower UHF bands (up to 525 MHz), and the other one covering the upper UHF bands (from 779 MHz). The PLLs are optimized for user-transparent low lock time and fast auto-calibrating operation. In transmission, frequency modulation is performed digitally within the PLL bandwidth. The PLL also features optional pre-filtering of the bit stream to improve spectral purity.

SX1276/77/78/79 feature three distinct RF power amplifiers. Two of those, connected to RFO_LF and RFO_HF, can deliver up to +14 dBm, are unregulated for high power efficiency and can be connected directly to their respective RF receiver inputs via a pair of passive components to form a single antenna port high efficiency transceiver. The third PA, connected to the PA_BOOST pin and can deliver up to +20 dBm via a dedicated matching network. Unlike the high efficiency PAs, this high-stability PA covers all frequency bands that the frequency synthesizer addresses.

SX1276/77/78/79 also include two timing references, an RC oscillator and a 32 MHz crystal oscillator.

All major parameters of the RF front end and digital state machine are fully configurable via an SPI interface which gives access to SX1276/77/78/79's configuration registers. This includes a mode auto sequencer that oversees the transition and calibration of the SX1276/77/78/79 between intermediate modes of operation in the fastest time possible.

The SX1276/77/78/79 are equipped with both standard FSK and long range spread spectrum (LoRa™) modems. Depending upon the mode selected either conventional OOK or FSK modulation may be employed or the LoRa™ spread spectrum modem.

3.1. LoRa™ Modem

The LoRa™ modem uses a proprietary spread spectrum modulation technique. This modulation, in contrast to legacy modulation techniques, permits an increase in link budget and increased immunity to in-band interference. At the same time the frequency tolerance requirement of the crystal reference oscillator is relaxed - allowing a performance increase for a reduction in system cost. For a detailed description of the design trade-offs and operation of the SX1276/77/78/79 please consult Section 4.1 of the datasheet.

3.2. FSK/OOK Modem

In FSK/OOK mode the SX1276/77/78/79 supports standard modulation techniques including OOK, FSK, GFSK, MSK and GMSK. The SX1276/77/78/79 is especially suited to narrow band communication thanks the low-IF architecture employed and the built-in AFC functionality. For full information on the FSK/OOK modem please consult Section 4.2 of this document.

4. SX1276/77/78/79 Digital Electronics

4.1. The LoRa™ Modem

The LoRa™ modem uses spread spectrum modulation and forward error correction techniques to increase the range and robustness of radio communication links compared to traditional FSK or OOK based modulation. Examples of the performance improvement possible, for several possible settings, are summarised in the table below. Here the spreading factor and error correction rate are design variables that allow the designer to optimise the trade-off between occupied bandwidth, data rate, link budget improvement and immunity to interference.

*Table 12 Example **LoRa™** Modem Performances, 868MHz Band*

Bandwidth (kHz)	Spreading Factor	Coding rate	Nominal Rb (bps)	Sensitivity indication (dBm)	Frequency Reference
10.4	6	4/5	782	-131	TCXO
	12	4/5	24	-147	
20.8	6	4/5	1562	-128	
	12	4/5	49	-144	
62.5	6	4/5	4688	-121	XTAL
	12	4/5	146	-139	
125	6	4/5	9380	-118	
	12	4/5	293	-136	

Notes - for all bandwidths lower than 62.5 kHz, it is advised to use a TCXO as a frequency reference. This is required to meet the frequency error tolerance specifications given in the Electrical Specification

- Higher spreading factors and longer transmission times impose more stringent constraints on the short term frequency stability of the reference. Please get in touch with a Semtech representative to implement extremely low sensitivity products.

For European operation the range of crystal tolerances acceptable for each sub-band (of the ERC 70-03) is given in the specifications table. For US based operation a frequency hopping mode is available that automates both the LoRa™ spread spectrum and frequency hopping spread spectrum processes.

Another important facet of the LoRa™ modem is its increased immunity to interference. The LoRa™ modem is capable of co-channel GMSK rejection of up to 20 dB. This immunity to interference permits the simple coexistence of LoRa™ modulated systems either in bands of heavy spectral usage or in hybrid communication networks that use LoRa™ to extend range when legacy modulation schemes fail.

4.1.1. Link Design Using the LoRa™ Modem

4.1.1.1. Overview

The LoRa™ modem is setup as shown in the following figure. This configuration permits the simple replacement of the FSK modem with the LoRa™ modem via the configuration register setting *RegOpMode.* This change can be performed on the fly (in Sleep operating mode) thus permitting the use of both standard FSK or OOK in conjunction with the long range capability. The LoRa™ modulation and demodulation process is proprietary, it uses a form of spread spectrum modulation combined with cyclic error correction coding. The combined influence of these two factors is an increase in link budget and enhanced immunity to interference.

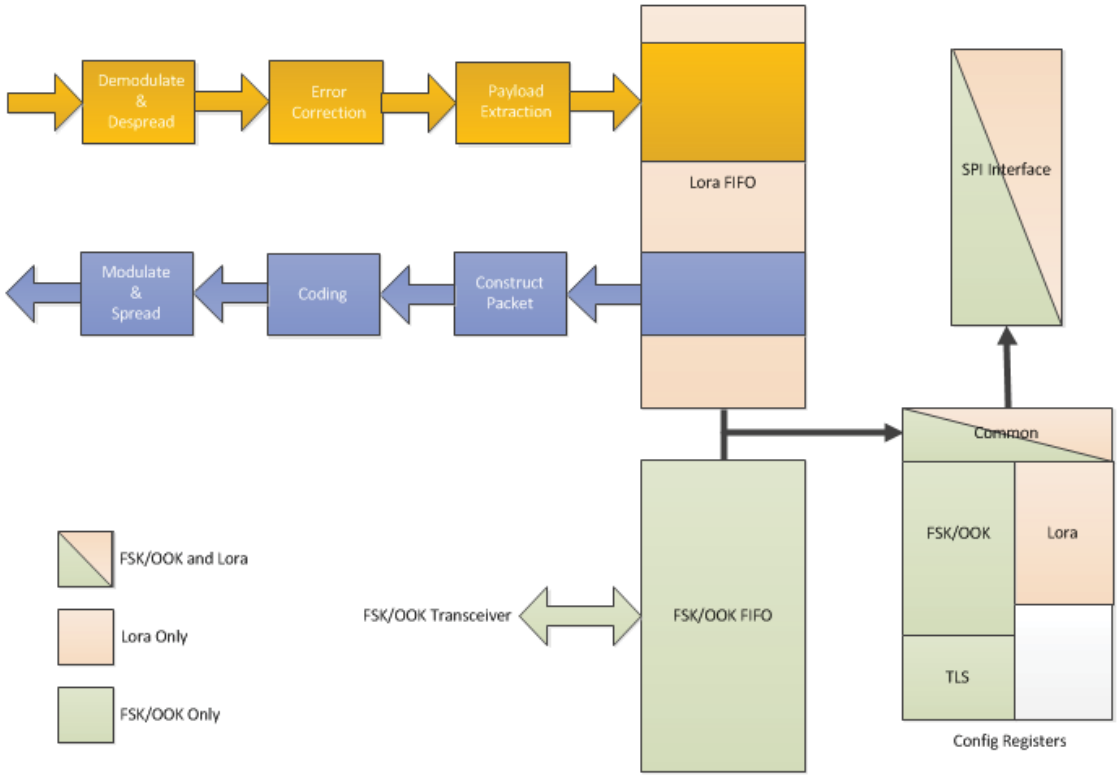


Figure 5. LoRa™ Modem Connectivity

A simplified outline of the transmit and receive processes is also shown above. Here we see that the LoRa™ modem has an independent dual port data buffer FIFO that is accessed through an SPI interface common to all modes. Upon selection of LoRa™ mode, the configuration register mapping of the SX1276/77/78/79 changes. For full details of this change please consult the register description of Section 6.

So that it is possible to optimise the LoRa™ modulation for a given application, access is given to the designer to three critical design parameters. Each one permitting a trade off between link budget, immunity to interference, spectral occupancy and nominal data rate. These parameters are spreading factor, modulation bandwidth and error coding rate.

4.1.1.2. Spreading Factor

The spread spectrum LoRa™ modulation is performed by representing each bit of payload information by multiple chips of information. The rate at which the spread information is sent is referred to as the symbol rate (Rs), the ratio between the nominal symbol rate and chip rate is the spreading factor and represents the number of symbols sent per bit of information. The range of values accessible with the LoRa™ modem are shown in the following table.

Table 13 Range of Spreading Factors

SpreadingFactor (RegModulationCfg)	Spreading Factor (Chips / symbol)	LoRa Demodulator SNR
6	64	-5 dB
7	128	-7.5 dB
8	256	-10 dB
9	512	-12.5 dB
10	1024	-15 dB
11	2048	-17.5 dB
12	4096	-20 dB

Note that the spreading factor, *SpreadingFactor*, must be known in advance on both transmit and receive sides of the link as different spreading factors are orthogonal to each other. Note also the resulting signal to noise ratio (SNR) required at the receiver input. It is the capability to receive signals with negative SNR that increases the sensitivity, so link budget and range, of the LoRa receiver.

Spreading Factor 6

SF = 6 Is a special use case for the highest data rate transmission possible with the LoRa modem. To this end several settings must be activated in the SX1276/77/78/79 registers when it is in use. These settings are only valid for SF6 and should be set back to their default values for other spreading factors:

- Set *SpreadingFactor* = 6 in *RegModemConfig2*
- The header must be set to Implicit mode.
- Set the bit field *DetectionOptimize* of register *RegLoRaDetectOptimize* to value "0b101".
- Write 0x0C in the register *RegDetectionThreshold*.

4.1.1.3. Coding Rate

To further improve the robustness of the link the LoRa™ modem employs cyclic error coding to perform forward error detection and correction. Such error coding incurs a transmission overhead - the resultant additional data overhead per transmission is shown in the table below.

Table 14 Cyclic Coding Overhead

CodingRate (RegTxCfg1)	Cyclic Coding Rate	Overhead Ratio
1	4/5	1.25
2	4/6	1.5
3	4/7	1.75
4	4/8	2

Forward error correction is particularly efficient in improving the reliability of the link in the presence of interference. So that the coding rate (and so robustness to interference) can be changed in response to channel conditions - the coding rate can optionally be included in the packet header for use by the receiver. Please consult Section 4.1.1.6 for more information on the LoRa™ packet and header.

4.1.1.4. Signal Bandwidth

An increase in signal bandwidth permits the use of a higher effective data rate, thus reducing transmission time at the expense of reduced sensitivity improvement. There are of course regulatory constraints in most countries on the permissible occupied bandwidth. Contrary to the FSK modem which is described in terms of the single sideband bandwidth, the LoRa™ modem bandwidth refers to the double sideband bandwidth (or total channel bandwidth). The range of bandwidths relevant to most regulatory situations is given in the LoRa™ modem specifications table (see Section 2.5.5).

Table 15 LoRa Bandwidth Options

Bandwidth (kHz)	Spreading Factor	Coding rate	Nominal Rb (bps)
7.8	12	4/5	18
10.4	12	4/5	24
15.6	12	4/5	37
20.8	12	4/5	49
31.2	12	4/5	73
41.7	12	4/5	98
62.5	12	4/5	146
125	12	4/5	293
250	12	4/5	586
500	12	4/5	1172

Note In the lower band (169 MHz), the 250 kHz and 500 kHz bandwidths are not supported.

4.1.1.5. LoRa™ Transmission Parameter Relationship

With a knowledge of the key parameters that can be controlled by the user we define the LoRa™ symbol rate as:

$$Rs = \frac{BW}{2^{SF}}$$

where BW is the programmed bandwidth and SF is the spreading factor. The transmitted signal is a constant envelope signal. Equivalently, one chip is sent per second per Hz of bandwidth.

4.1.1.6. *LoRa™ Packet Structure*

The LoRa™ modem employs two types of packet format, explicit and implicit. The explicit packet includes a short header that contains information about the number of bytes, coding rate and whether a CRC is used in the packet. The packet format is shown in the following figure.

The LoRa™ packet comprises three elements:

- A preamble.
- An optional header.
- The data payload.

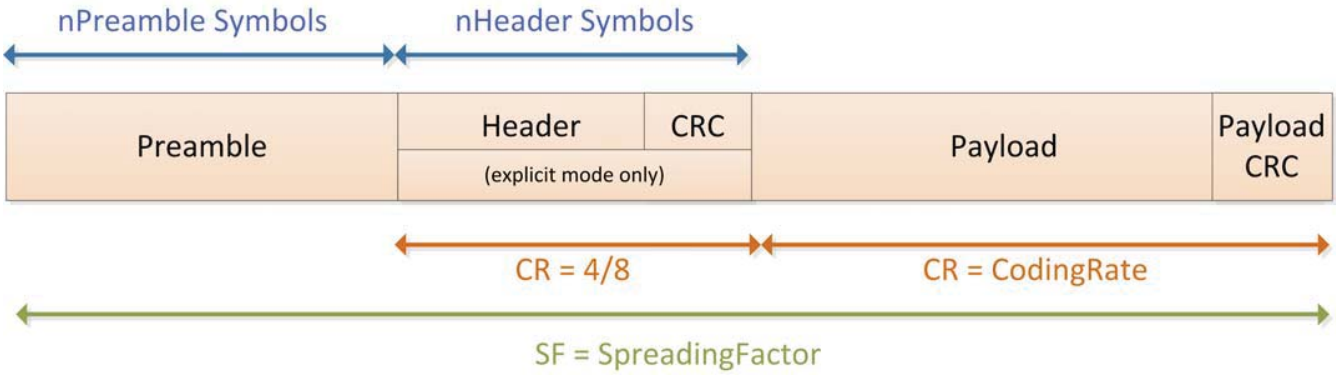


Figure 6. LoRa™ Packet Structure

Preamble

The preamble is used to synchronize receiver with the incoming data flow. By default the packet is configured with a 12 symbol long sequence. This is a programmable variable so the preamble length may be extended, for example in the interest of reducing to receiver duty cycle in receive intensive applications. However, the minimum length suffices for all communication. The transmitted preamble length may be changed by setting the register *PreambleLength* from 6 to 65535, yielding total preamble lengths of 6+4 to 65535+4 symbols, once the fixed overhead of the preamble data is considered. This permits the transmission of a near arbitrarily long preamble sequence.

The receiver undertakes a preamble detection process that periodically restarts. For this reason the preamble length should be configured identical to the transmitter preamble length. Where the preamble length is not known, or can vary, the maximum preamble length should be programmed on the receiver side.

Header

Depending upon the chosen mode of operation two types of header are available. The header type is selected by the *ImplicitHeaderModeOn* bit found within the *RegModemConfig1* register.

Explicit Header Mode

This is the default mode of operation. Here the header provides information on the payload, namely:

- The payload length in bytes.
- The forward error correction code rate
- The presence of an optional 16-bits CRC for the payload.

The header is transmitted with maximum error correction code (4/8). It also has its own CRC to allow the receiver to discard invalid headers.

Implicit Header Mode

In certain scenarios, where the payload, coding rate and CRC presence are fixed or known in advance, it may be advantageous to reduce transmission time by invoking implicit header mode. In this mode the header is removed from the packet. In this case the payload length, error coding rate and presence of the payload CRC must be manually configured on both sides of the radio link.

Note With SF = 6 selected, implicit header mode is the only mode of operation possible.

Explicit Header Mode:

In Explicit Header Mode, the presence of the CRC at the end of the payload in selected only on the transmitter side through the bit *RxPayloadCrcOn* in the register *RegModemConfig1*.

On the receiver side, the bit *RxPayloadCrcOn* in the register *RegModemConfig1* is not used and once the payload has been received, the user should check the bit *CrcOnPayload* in the register *RegHopChannel.* If the bit *CrcOnPayload* is at '1', the user should then check the Irq Flag *PayloadCrcError* to make sure the CRC is valid.

If the bit *CrcOnPayload* is at '0', it means there was no CRC on the payload and thus the IRQ Flag *PayloadCrcError* will not be trigged even if the payload has errors.

Explicit Header	Transmitter	Receiver	CRC Status
Value of the bit RxPayloadCrcOn	0	0	CRC is not checked
	0	1	CRC is not checked
	1	0	CRC is checked
	1	1	CRC is checked

Implicit Header Mode;

In Implicit Header Mode, it is necessary to set the bit RxPayloadCrcOn in the register *RegModemConfig1* on both sides (TX and RX)

Implicit Header	Transmitter	Receiver	CRC Status
Value of the bit RxPayloadCrcOn	0	0	CRC is not checked
	0	1	CRC is always wrong
	1	0	CRC is not checked
	1	1	CRC is checked

Low Data Rate Optimization

Given the potentially long duration of the packet at high spreading factors the option is given to improve the robustness of the transmission to variations in frequency over the duration of the packet transmission and reception. The bit *LowDataRateOptimize* increases the robustness of the LoRa link at these low effective data rates. Its use is mandated when the symbol duration exceeds 16ms. Note that both the transmitter and the receiver must have the same setting for *LowDataRateOptimize*.

Payload

The packet payload is a variable-length field that contains the actual data coded at the error rate either as specified in the header in explicit mode or in the register settings in implicit mode. An optional CRC may be appended. For more information on the payload and how it is loaded from the data buffer FIFO please see Section 4.1.2.3.

4.1.1.7. Time on air

For a given combination of spreading factor (SF), coding rate (CR) and signal bandwidth (BW) the total on-the-air transmission time of a LoRa™ packet can be calculated as follows. From the definition of the symbol rate it is convenient to define the symbol rate:

$$Ts = \frac{1}{Rs}$$

The LoRa packet duration is the sum of the duration of the preamble and the transmitted packet. The preamble length is calculated as follows:

$$T_{preamble} = (n_{preamble} + 4.25)T_{sym}$$

where $n_{preamble}$ is the programmed preamble length, taken from the registers *RegPreambleMsb* and *RegPreambleLsb*.The payload duration depends upon the header mode that is enabled. The following formula gives the number of payload symbols.

$$n_{payload} = 8 + max\left(ceil\left[\frac{(8PL - 4SF + 28 + 16CRC - 20IH)}{4(SF - 2DE)}\right](CR + 4), 0\right)$$

With the following dependencies:

- PL is the number of Payload bytes (1 to 255)
- SF is the spreading factor (6 to 12)
- IH=0 when the header is enabled, IH=1 when no header is present
- DE=1 when *LowDataRateOptimize*=1, DE=0 otherwise
- CR is the coding rate (1 corresponding to 4/5, 4 to 4/8)

The Payload duration is then the symbol period multiplied by the number of Payload symbols

$$T_{payload} = n_{payload} \times T_s$$

The time on air, or packet duration, in simply then the sum of the preamble and payload duration.

$$T_{packet} = T_{preamble} + T_{payload}$$

4.1.1.8. Frequency Hopping with LoRa™

Frequency hopping spread spectrum (FHSS) is typically employed when the duration of a single packet could exceed regulatory requirements relating to the maximum permissible channel dwell time. This is most notably the case in US operation where the 902 to 928 MHz ISM band which makes provision for frequency hopping operation. To ease the implementation of FHSS systems the frequency hopping mode of the LoRa™ modem can be enabled by setting *FreqHoppingPeriod* to a non-zero value in register *RegHopPeriod*.

Principle of Operation

The principle behind the FHSS scheme is that a portion of each LoRa™ packet is transmitted on each hopping channel from a look up table of frequencies managed by the host microcontroller. After a predetermined hopping period the transmitter and receiver change to the next channel in a predefined list of hopping frequencies to continue transmission and reception of the next portion of the packet. The time which the transmission will dwell in any given channel is determined by *FreqHoppingPeriod* which is an integer multiple of symbol periods:

$$HoppingPeriod = Ts \times FreqHoppingPeriod$$

The frequency hopping transmission and reception process starts at channel 0. The preamble and header are transmitted first on channel 0. At the beginning of each transmission the channel counter *FhssPresentChannel* (located in the register *RegHopChannel*) is incremented and the interrupt signal *FhssChangeChannel* is generated. The new frequency must then be programmed within the hopping period to ensure it is taken into account for the next hop, the interrupt *ChangeChannelFhss* is then to be cleared by writing a logical '1'.

FHSS Reception always starts on channel 0. The receiver waits for a valid preamble detection before starting the frequency hopping process as described above. Note that in the eventuality of header CRC corruption, the receiver will automatically request channel 0 and recommence the valid preamble detection process.

Timing of Channel Updates

The interrupt requesting the channel change, *FhssChangeChannel,* is generated upon transition to the new frequency. The frequency hopping process is illustrated in the diagram below:

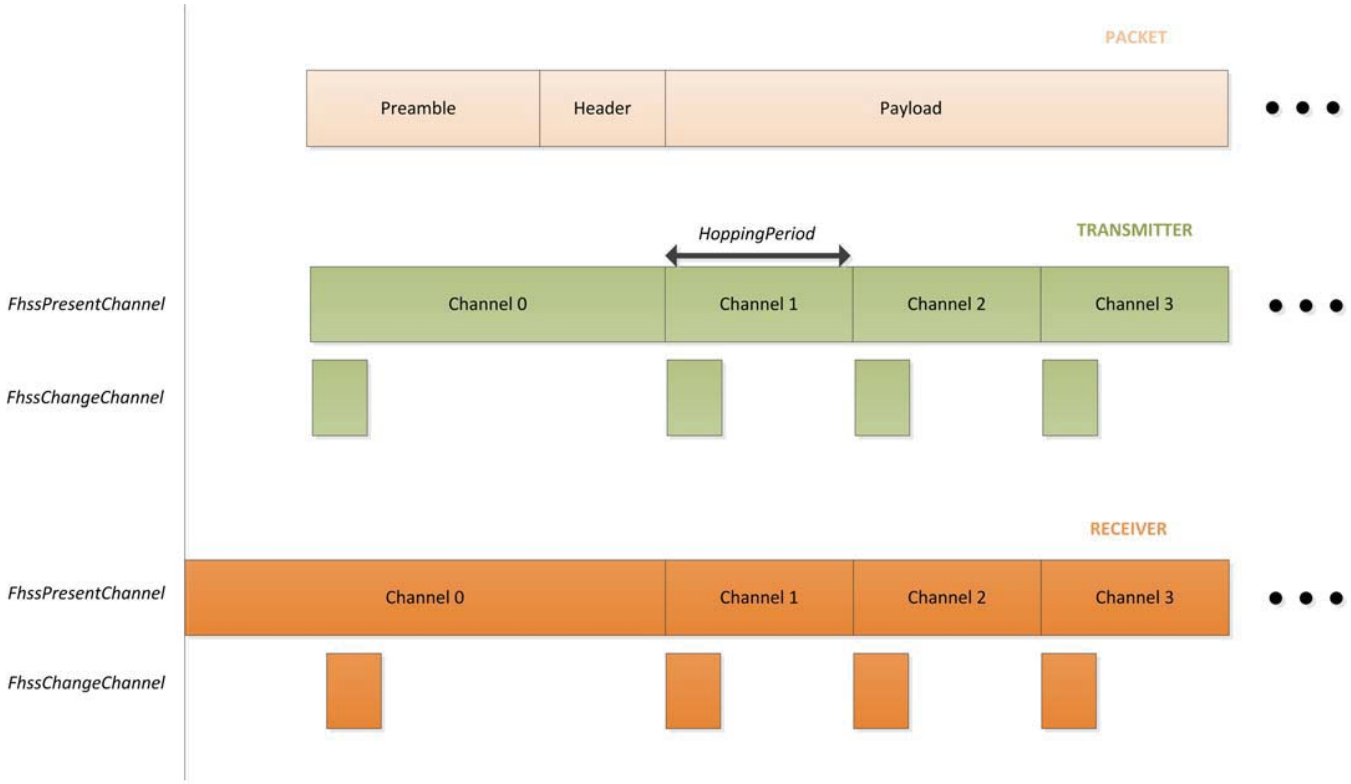


Figure 7. Interrupts Generated in the Case of Successful Frequency Hopping Communication.

4.1.2. LoRa™ Digital Interface

The LoRa™ modem comprises three types of digital interface, static configuration registers, status registers and a FIFO data buffer. All are accessed through the SX1276/77/78/79's SPI interface - full details of each type of register are given below. Full listings of the register addresses used for SPI access are given in Section 6.4.

4.1.2.1. LoRa™ Configuration Registers

Configuration registers are accessed through the SPI interface. Registers are readable in all device mode including Sleep. However, they should be **written only in Sleep and Standby modes**. Please note that **the automatic top level sequencer (TLS modes) are not available in** LoRa™ **mode** and **the configuration register mapping changes as shown in Table** 41**.** The content of the LoRa™ configuration registers is retained in FSK/OOK mode. For the functionality of mode registers common to both FSK/OOK and LoRa™ mode, please consult the Analog and RF Front End section of this document (Section 5).

4.1.2.2. Status Registers

Status registers provide status information during receiver operation.

4.1.2.3. LoRa™ Mode FIFO Data Buffer

Overview

The SX1276/77/78/79 is equipped with a 256 byte RAM data buffer which is uniquely accessible in LoRa mode. This RAM area, herein referred to as the FIFO Data buffer, is fully customizable by the user and allows access to the received, or to be transmitted, data. All access to the LoRa™ FIFO data buffer is done via the SPI interface. A diagram of the user defined memory mapping of the FIFO data buffer is shown below. These FIFO data buffer can be read in all operating modes except sleep and store data related to the last receive operation performed. It is automatically cleared of old content upon each new transition to receive mode.

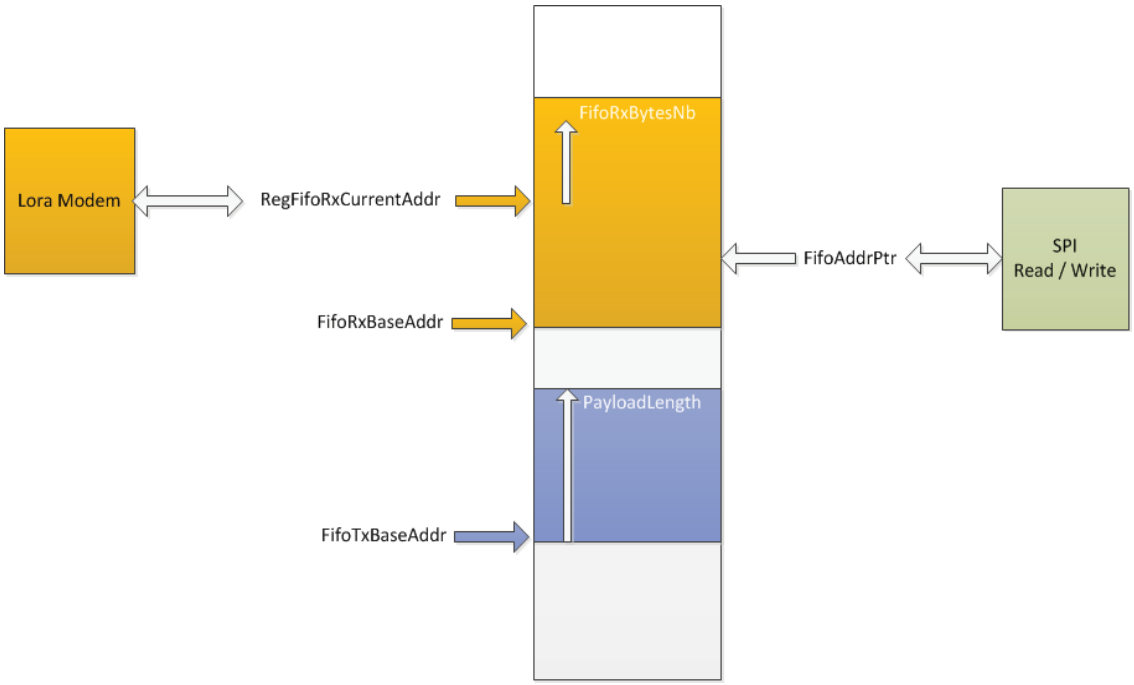


Figure 8. LoRa™ Data Buffer

Principle of Operation

Thanks to its dual port configuration, it is possible to simultaneously store both transmit and receive information in the FIFO data buffer. The register *RegFifoTxBaseAddr* specifies the point in memory where the transmit information is stored. Similarly, for receiver operation, the register *RegFifoRxBaseAddr* indicates the point in the data buffer where information will be written to in event of a receive operation.

By default, the device is configured at power up so that half of the available memory is dedicated to Rx (*RegFifoRxBaseAddr* initialized at address 0x00) and the other half is dedicated for Tx (*RegFifoTxBaseAddr* initialized at address 0x80).

However, due to the contiguous nature of the FIFO data buffer, the base addresses for Tx and Rx are fully configurable across the 256 byte memory area. Each pointer can be set independently anywhere within the FIFO. To exploit the maximum FIFO data buffer size in transmit or receive mode, the whole FIFO data buffer can be used in each mode by setting the base addresses *RegFifoTxBaseAddr* and *RegFifoRxBaseAddr* at the bottom of the memory (0x00).

The FIFO data buffer is cleared when the device is put in SLEEP mode, consequently no access to the FIFO data buffer is possible in sleep mode. However, the data in the FIFO data buffer are retained when switching across the other LoRa™ modes of operation, so that a received packet can be retransmitted with minimum data handling on the controller side. The FIFO data buffer is not self-clearing (unless if the device is put in sleep mode) and the data will only be “erased” when a new set of data is written into the occupied memory location.

The FIFO data buffer location to be read from, or written to, via the SPI interface is defined by the address pointer *RegFifoAddrPtr*. Before any read or write operation it is hence necessary to initialize this pointer to the corresponding base value. Upon reading or writing to the FIFO data buffer (*RegFifo*) the address pointer will then increment automatically.

The register *RegRxNbBytes* defines the size of the memory location to be written in the event of a successful receive operation. The register *RegPayloadLength* indicates the size of the memory location to be transmitted. In implicit header mode, the register *RegRxNbBytes* is not used as the number of payload bytes is known. Otherwise, in explicit header mode, the initial size of the receive buffer is set to the packet length in the received header. The register *RegFifoRxCurrentAddr* indicates the location of the last packet received in the FIFO so that the last packet received can be easily read by pointing the register *RegFifoAddrPtr* to this register.

It is important to notice that all the received data will be written to the FIFO data buffer even if the CRC is invalid, permitting user defined post processing of corrupted data. It is also important to note that when receiving, if the packet size exceeds the buffer memory allocated for the Rx, it will overwrite the transmit portion of the data buffer.

4.1.2.4. Interrupts in LoRa Mode

Two registers are used to control the IRQ in LoRa mode, the register *RegIrqFlagsMask* which is used to mask the interrupts and the register *RegIrqFlags* which indicates which IRQ has been trigged.
In the register *RegIrqFlagsMask,* setting a bit to ‘1’ will mask the interrupt, meaning this interrupt is disactivated. By default all the interrupt are available.
In the register *RegIrqFlags*, a ‘1’ indicates a given IRQ has been trigged and then the IRQ must be clear by writing a ‘1’.

4.1.3. Operation of the LoRa™ Modem

4.1.3.1. Operating Mode Control

The operating modes of the LoRa™ modem are accessed by enabling LoRa™ mode (setting the *LongRangeMode* bit of *RegOpMode*). Depending upon the operating mode selected the range of functionality and register access is given by the following table:

Table 16 LoRa™ Operating Mode Functionality

Operating Mode	Description
SLEEP	Low-power mode. In this mode only SPI and configuration registers are accessible. Lora FIFO is not accessible. Note that this is the only mode permissible to switch between FSK/OOK mode and LoRa mode.
STANDBY	both Crystal oscillator and Lora baseband blocks are turned on.RF part and PLLs are disabled
FSTX	This is a frequency synthesis mode for transmission. The PLL selected for transmission is locked and active at the transmit frequency. The RF part is off.
FSRX	This is a frequency synthesis mode for reception. The PLL selected for reception is locked and active at the receive frequency. The RF part is off.
TX	When activated the SX1276/77/78/79 powers all remaining blocks required for transmit, ramps the PA, transmits the packet and returns to Standby mode.
RXCONTINUOUS	When activated the SX1276/77/78/79 powers all remaining blocks required for reception, processing all received data until a new user request is made to change operating mode.
RXSINGLE	When activated the SX1276/77/78/79 powers all remaining blocks required for reception, remains in this state until a valid packet has been received and then returns to Standby mode.
CAD	When in CAD mode, the device will check a given channel to detect LoRa preamble signal

It is possible to access any mode from any other mode by changing the value in the *RegOpMode* register.

4.1.4. Frequency Settings

Recalling that the frequency step is given by:

$$F_{STEP} = \frac{F_{XOSC}}{2^{19}}$$

In order to set LO frequency values following registers are available.

Frf is a 24-bit register which defines carrier frequency. The carrier frequency relates to the register contents by following formula:

$$F_{RF} = F_{STEP} \times Frf(23,0)$$

4.1.5. Frequency Error Indication

The SX1276/77/78/79 derives its RF centre frequency from a crystal reference oscillator which has a finite frequency precision. Errors in reference frequency will manifest themselves as errors of the same proportion from the RF centre frequency.

In LoRa receive mode the SX1276/77/78/79 is capable of measuring the frequency offset between the receiver centre frequency and that of an incoming LoRa signal. The modem is intolerant of frequency offsets in the region of +/- 25% of the bandwidth and will accurately report the error over this same range.

The error is read by reading the three *RegFei* registers. The contents of which are a signed 20 bit two's compliment word, *FreqError*. The frequency error is determined from the register contents by:

$$F_{Error} = \frac{FreqError \times 2^{24}}{F_{xtal}} \times \frac{BW[kHz]}{500}$$

Where Fxtal is the crystal frequency.

To correct the measured frequency error there are two steps to be taken. First the frequency error is subtracted from the RF centre frequency. This calculation must be performed locally (or in a look-up-table), no provision is made in the circuit to apply the correction automatically.

Secondly, assuming that the frequency error is due to reference oscillator drift, the data rate of the LoRa modem must also be compensated accordingly. This is done by

*PpmOffset = 0.95 * measured Offset [PPM]*

Where PpmOffset is the value to be programmed into register 0x27 and the measured Offset is the PPM drift equivalent to the frequency error reported by the LoRa frequency error indicator. The *PpmOffset* value is a signed two's compliment value.

4.1.6. LoRa™ Modem State Machine Sequences

The sequence for transmission and reception of data to and from the LoRa™ modem, together with flow charts of typical sequences of operation, are detailed below.

Data Transmission Sequence

In transmit mode power consumption is optimized by enabling RF, PLL and PA blocks only when packet data needs to be transmitted. Figure 9 shows a typical LoRa™ transmit sequence.

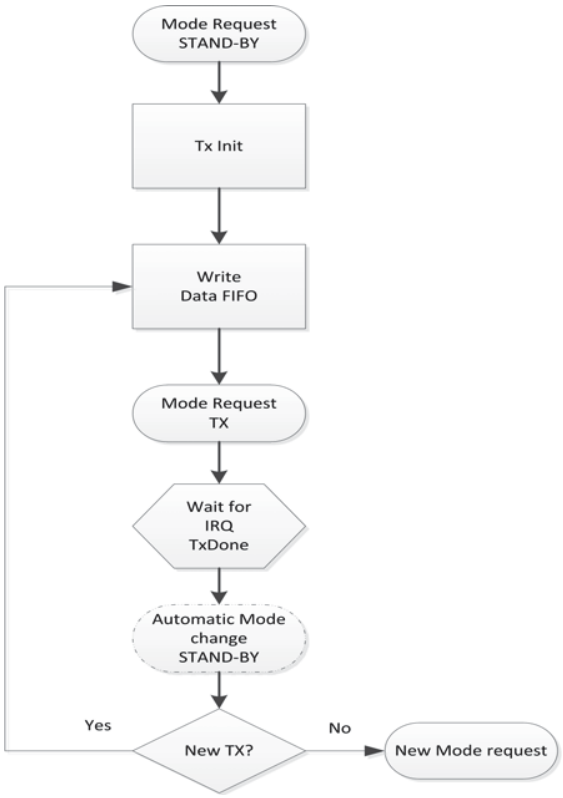


Figure 9. LoRa™ Modulation Transmission Sequence.

- Static configuration registers can only be accessed in Sleep mode, Standby mode or FSTX mode.
- The LoRa™ FIFO can only be filled in Standby mode.
- Data transmission is initiated by sending TX mode request.
- Upon completion the *TxDone* interrupt is issued and the radio returns to Standby mode.
- Following transmission the radio can be manually placed in Sleep mode or the FIFO refilled for a subsequent Tx operation.

LoRa™ Transmit Data FIFO Filling

In order to write packet data into FIFO user should:

1 Set *FifoPtrAddr* to *FifoTxPtrBase*.

2 Write *PayloadLength* bytes to the FIFO (*RegFifo*)

Data Reception Sequence

Figure 10 shows typical LoRa™ receive sequences for both single and continuous receiver modes of operation.

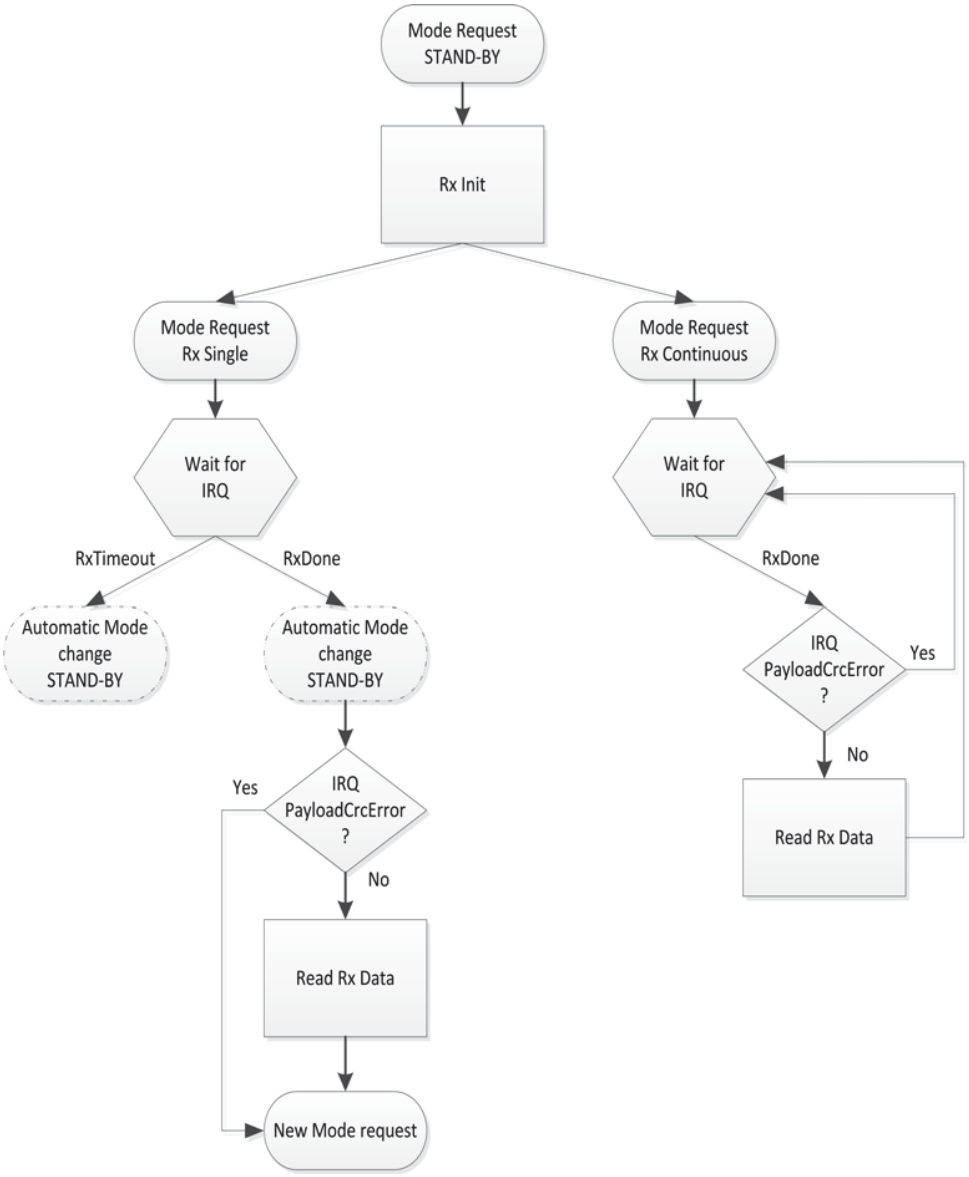


Figure 10. LoRa™ Receive Sequence.

The LoRa receive modem can work in two distinct mode

1. Single receive mode
2. Continuous receive mode

Those two modes correspond to different use cases and it is important to understand the subtle differences between them.

Single Reception Operating Mode

In this mode, the modem searches for a preamble during a given period of time. If a preamble hasn't been found at the end of the time window, the chip generates the *RxTimeout* interrupt and goes back to Standby mode. The length of the reception window (in symbols) is defined by the *RegSymbTimeout* register and should be in the range of 4 (minimum time for the modem to acquire lock on a preamble) up to 1023 symbols.

At the end of the payload, the *RxDone* interrupt is generated together with the interrupt *PayloadCrcError* if the payload CRC is not valid. However, even when the CRC is not valid, the data are written in the FIFO data buffer for post processing. Following the *RxDone* interrupt the radio goes to Standby mode.

The modem will also automatically return in Standby mode when the interrupts *RxDone* is generated. Therefore, this mode should only be used when the time window of arrival of the packet is known. In other cases, the RX continuous mode should be used.

In Rx single mode, low-power is achieved by turning off PLL and RF blocks as soon as a packet has been received. The flow is as follows:

1 *Set FifoAddrPtr* to *FifoRxBaseAddr*.

2 Static configuration register device can be written in either Sleep mode, Standby mode or FSRX mode.

3 A single packet receive operation is initiated by selecting the operating mode RXSINGLE.

4 The receiver will then await the reception of a valid preamble. Once received, the gain of the receive chain is set. Following the ensuing reception of a valid header, indicated by the *ValidHeader* interrupt in explicit mode. The packet reception process commences. Once the reception process is complete the *RxDone* interrupt is set. The radio then returns automatically to Standby mode to reduce power consumption.

5 The receiver status register *PayloadCrcError* should be checked for packet payload integrity.

6 If a valid packet payload has been received then the FIFO should be read (See Payload Data Extraction below). Should a subsequent single packet reception need to be triggered, then the RXSINGLE operating mode must be re-selected to launch the receive process again - taking care to reset the SPI pointer (*FifoAddrPtr*) to the base location in memory (*FifoRxBaseAddr*).

Continuous Reception Operating Mode

In continuous receive mode, the modem scans the channel continuously for a preamble. Each time a preamble is detected the modem tracks it until the packet is received and then carries on waiting for the next preamble.

If the preamble length exceeds the anticipated value set by the registers *RegPreambleMsb* and *RegPreambleLsb* (measured in symbol periods) the preamble will be dropped and the search for a preamble restarted. However, this scenario will not be flagged by any interrupt. In continuous RX mode, opposite to the single RX mode, the RxTimeout interrupt will never occur and the device will never go in Standby mode automatically.

It is also important to note that the demodulated bytes are written in the data buffer memory in the order received. Meaning, the first byte of a new packet is written just after the last byte of the preceding packet. The RX modem address pointer is never reset as long as this mode is enabled. It is therefore necessary for the companion microcontroller to handle the address pointer to make sure the FIFO data buffer is never full.

In continuous mode the received packet processing sequence is given below.

1 Whilst in Sleep or Standby mode select RXCONT mode.
2 Upon reception of a valid header CRC the *RxDone i*nterrupt is set. The radio remains in RXCONT mode waiting for the next RX LoRa™ packet.
3 The *PayloadCrcError* flag should be checked for packet integrity.
4 If packet has been correctly received the FIFO data buffer can be read (see below).
5 The reception process (steps 2 - 4) can be repeated or receiver operating mode exited as desired.

In continuous mode status information are available only for the last packet received, i.e. the corresponding registers should be read before the next *RxDone* arrives.

Rx Single and Rx Continuous Use Cases

The LoRa single reception mode is used mainly in battery operated systems or in systems where the companion microcontroller has a limited availability of timers. In such systems, the use of the timeout present in Rx Single reception mode allows the end user to limit the amount of time spent in reception (and thus limiting the power consumption) while not using any of the companion MCU timers (the MCU can then be in sleep mode while the radio is in the reception mode). The RxTimeout interrupt generated at the end of the reception period is then used to wake-up the companion MCU. One of the advantages of the RxSingle mode is that the interrupt RxTimeout will not be triggered if the device is currently receiving data, thus giving the priority to the reception of the data over the timeout. However, if during the reception, the device loses track of the data due to external perturbation, the device will drop the reception, flag the interrupt RxTimeout and go in StandBy mode to decrease the power consumption of the system.

On the other hand, The LoRa continuous reception mode is used in systems which do not have power restrictions or on system where the use of a companion MCU timer is preferred over the radio embedded timeout system. In RxContinuous mode, the radio will track any LoRa signal present in the air and carry on the reception of packets until the companion MCU sets the radio into another mode of operation. Upon reception the interrupt RxDone will be trigged but the device will stay in Rx Mode, ready for the reception of the next packet.

Payload Data Extraction from FIFO

In order to retrieve received data from FIFO the user must ensure that *ValidHeader*, *PayloadCrcError, RxDone* and *RxTimeout* interrupts in the status register RegIrqFlags are not asserted to ensure that packet reception has terminated successfully (i.e. no flags should be set).

In case of errors the steps below should be skipped and the packet discarded. In order to retrieve valid received data from the FIFO the user must:

- *RegRxNbBytes* Indicates the number of bytes that have been received thus far.
- *RegFifoAddr*Ptr is a dynamic pointer that indicates precisely where the Lora modem received data has been written up to.
- Set *RegFifoAddr*Ptr to *RegFifoRxCurrentAddr*. This sets the FIFO pointer to the location of the last packet received in the FIFO. The payload can then be extracted by reading the register *RegFifo, RegRxNbBytes* times.
- Alternatively, it is possible to manually point to the location of the last packet received, from the start of the current packet, by setting *RegFifoAddrPtr* to *RegFifoRxByteAddr* minus *RegRxNbBytes*. The payload bytes can then be read from the FIFO by reading the *RegFifo* address *RegRxNbBytes* times.

Packet Filtering based on Preamble Start

The LoRa modem does automatically filter received packets based upon any addressing. However the SX1276/77/78/79 permit software filtering of the received packets based on the contents of the first few bytes of payload. A brief example is given below for a 4 byte address, however, the address length can be selected by the designer.

The objective of the packet filtering process is to determine the presence, or otherwise, of a valid packet designed for the receiver. If the packet is not for the receiver then the radio returns to sleep mode in order to improve battery life.

The software packet filtering process follows the steps below:

- Each time the RxDone interrupt is received, latch the *RegFifoRxByteAddr*[7:0] register content in a variable, this variable will be called start_address. The *RegFifoRxByteAddr*[7:0] register of the SX1276/77/78/79 gives in real time the address of the last byte written in the data buffer + 1 (or the address at which the next byte will be written) by the receive LoRa modem. So by doing this, we make sure that the variable *start_address* always contains the start address of the next packet.

- Upon reception of the interrupt *ValidHeader*, start polling the *RegFifoRxByteAddr*[7:0] register until it begins to increment. The speed at which this register will increment depends on the Spreading factor, the error correction code and the modulation bandwidth. (Note that this interrupt is still generated in implicit mode).

- As soon as *RegFifoRxByteAddr*[7:0] >= start address + 4, the first 4 bytes (address) are stored in the FIFO data buffer. These can be read and tested to see if the packet is destined for the radio and either remaining in Rx mode to receive the packet or returning to sleep mode if not.

Receiver Timeout Operation

In LoRa™ Rx Single mode, a receiver timeout functionality is available that permits the receiver to listen for a predetermined period of time before generating an interrupt signal to indicate that no valid packets have been received. The timer is absolute and commences as soon as the radio is placed in single receive mode. The interrupt itself, *RxTimeout*, can be found in the interrupt register *RegIrqFlags*. In Rx Single mode, the device will return to Standby mode as soon as the interrupt occurs. The user must then clear the interrupt or go into Sleep mode before returning into Rx Single mode. The programmed timeout value is expressed as a multiple of the symbol period and is given by:

$$TimeOut = LoraRxTimeout \cdot Ts$$

Channel Activity Detection

The use of a spread spectrum modulation technique presents challenges in determining whether the channel is already in use by a signal that may be below the noise floor of the receiver. The use of the RSSI in this situation would clearly be impracticable. To this end the channel activity detector is used to detect the presence of other LoRa™ signals. Figure 11 shows the channel activity detection (CAD) process:

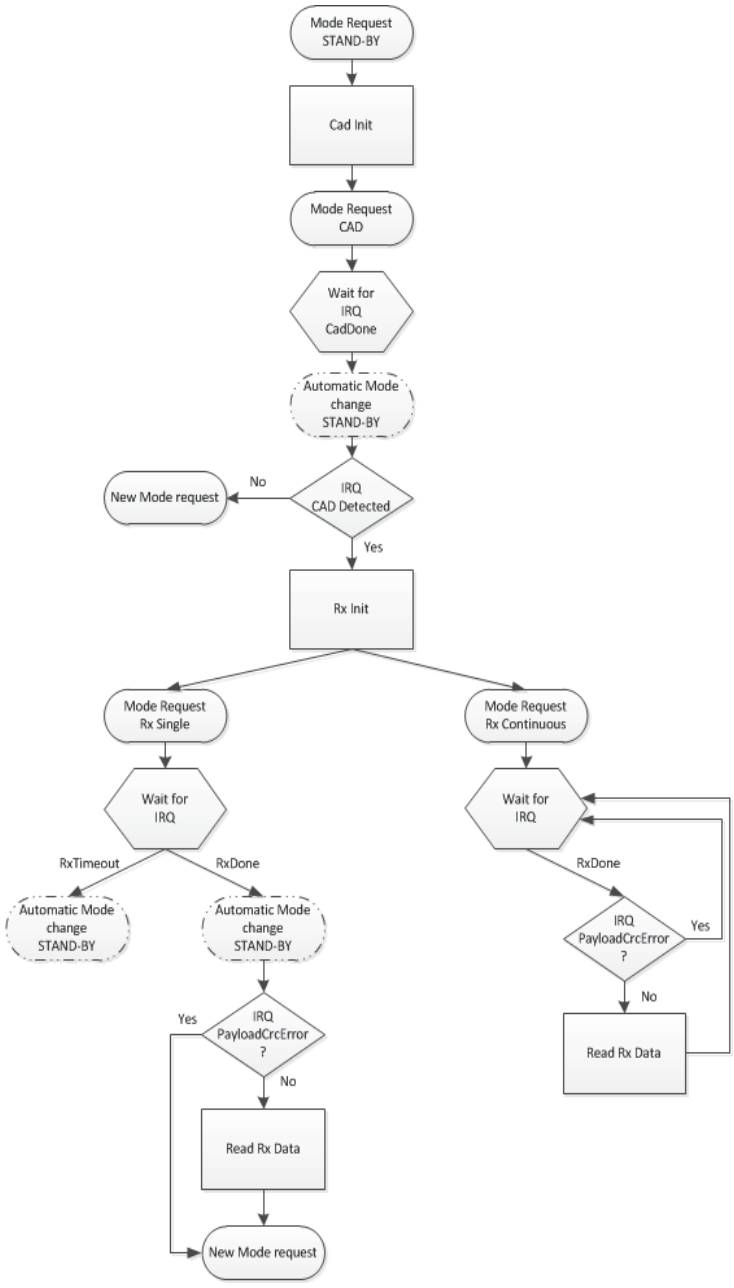


Figure 11. LoRa™ CAD Flow

Principle of Operation

The channel activity detection mode is designed to detect a LoRa preamble on the radio channel with the best possible power efficiency. Once in CAD mode, the SX1276/77/78/79 will perform a very quick scan of the band to detect a LoRa packet preamble.

During a CAD the following operations take place:

- The PLL locks
- The radio receiver captures LoRa preamble symbol of data from the channel. The radio current consumption during that phase corresponds to the specified Rx mode current
- The radio receiver and the PLL turn off, and the modem digital processing starts.
- The modem searches for a correlation between the radio captured samples and the ideal preamble waveform. This correlation process takes a little bit less than a symbol period to perform. The radio current consumption during that phase is greatly reduced.
- Once the calculation is finished the modem generates the CadDone interrupt. If the correlation was successful, CadDetected is generated simultaneously.
- The chip goes back to Standby mode.
- If a preamble was detected, clear the interrupt, then initiate the reception by putting the radio in RX single mode or RX continuous mode.

The time taken for the channel activity detection is dependent upon the LoRa modulation settings used. For a given configuration the typical CAD detection time is shown in the graph below, expressed as a multiple of the LoRa symbol period. Of this period the radio is in receiver mode for (2^{SF} + 32) / BW seconds. For the remainder of the CAD cycle the radio is in a reduced consumption state.

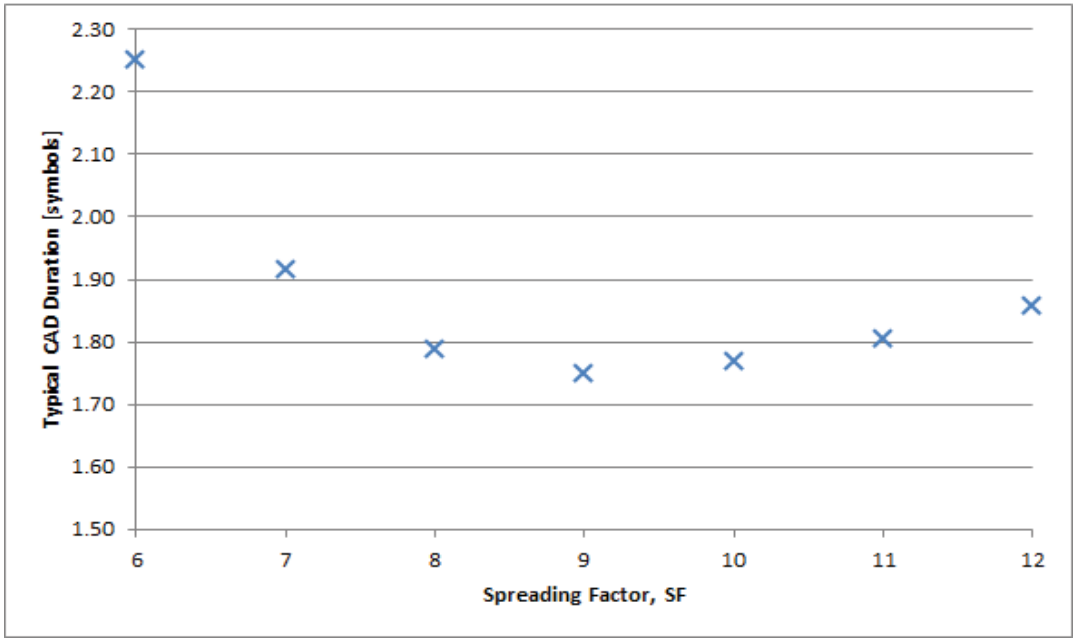


Figure 12. CAD Time as a Function of Spreading Factor

To illustrate this process and the respective consumption in each mode, the CAD process follows the sequence of events outlined below:

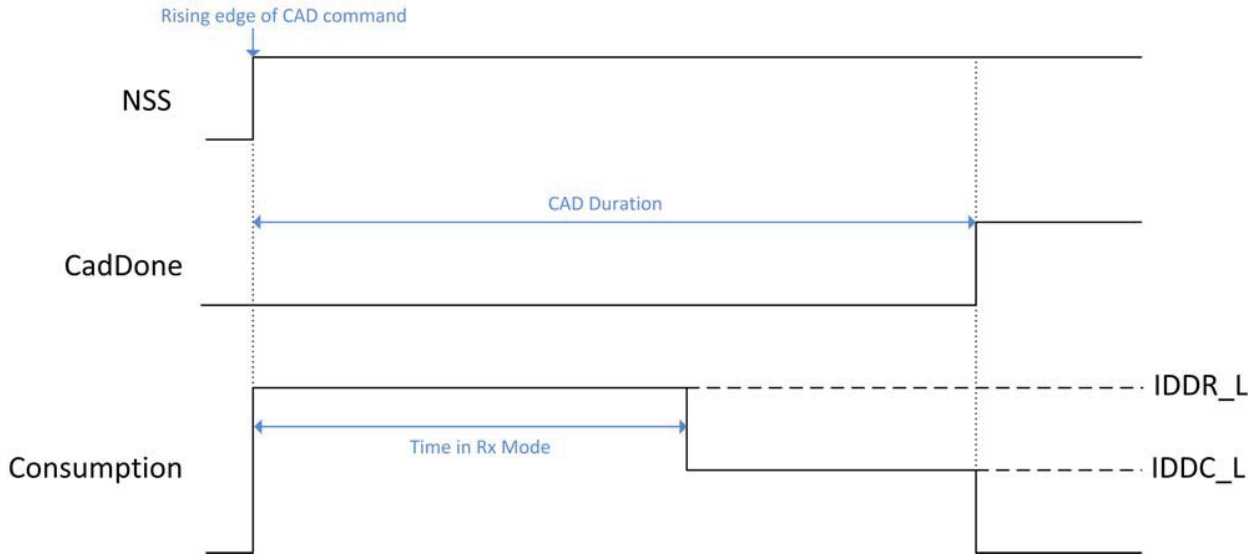


Figure 13. Consumption Profile of the LoRa CAD Process

The receiver is then in full receiver mode for just over half of the activity detection, followed by a reduced consumption processing phase where the consumption varies with the LoRa bandwidth as shown in the table below.

Table 17 LoRa CAD Consumption Figures

Bandwidth (kHz)	Full Rx, IDDR_L (mA)	Processing, IDDC_L (mA)
7.8 to 41.7	11	5.2
62.5	11	5.6
125	11.5	6
250	12.4	6.8
500	13.8	8.3

Note These numbers can be slightly lower when using Band 2 and 3, on the low frequency port.

4.1.6.1. Digital IO Pin Mapping

Six of SX1276/77/78/79's general purpose IO pins are available used in LoRa™ mode. Their mapping is shown below and depends upon the configuration of registers *RegDioMapping1* and *RegDioMapping2.*

Table 18 DIO Mapping LoRa™ Mode

Operating Mode	DIOx Mapping	DIO5	DIO4	DIO3	DIO2	DIO1	DIO0
ALL	00	ModeReady	CadDetected	CadDone	FhssChangeChannel	RxTimeout	RxDone
	01	ClkOut	PllLock	ValidHeader	FhssChangeChannel	FhssChangeChannel	TxDone
	10	ClkOut	PllLock	PayloadCrcError	FhssChangeChannel	CadDetected	CadDone
	11	-	-	-	-	-	-

4.1.7. Modem Status Indicators

The state of the LoRa modem is accessible with the *ModemStatus* bits in *RegModemStat.* They can mostly used for debug in Rx mode and the useful indicators are:

- Bit 0: *Signal Detected* indicates that a valid LoRa preamble has been detected
- Bit 1: *Signal Synchronized* indicates that the end of Preamble has been detected, the modem is in lock
- Bit 3: Header Info Valid toggles high when a valid Header (with correct CRC) is detected

4.2. FSK/OOK Modem

4.2.1. Bit Rate Setting

The bitrate setting is referenced to the crystal oscillator and provides a precise means of setting the bit rate (or equivalently chip) rate of the radio. In continuous transmit mode (Section 4.2.12) the data stream to be transmitted can be input directly to the modulator via pin 10 (DIO2/DATA) in an asynchronous manner, unless Gaussian filtering is used, in which case the DCLK signal on pin 9 (DIO1/DCLK) is used to synchronize the data stream. See section 4.2.2.3 for details on the Gaussian filter.

In Packet mode or in Continuous mode with Gaussian filtering enabled, the Bit Rate (BR) is controlled by bits *Bitrate* in *RegBitrateMsb and RegBitrateLsb*

$$BitRate = \frac{FXOSC}{BitRate(15,0) + \frac{BitrateFrac}{16}}$$

*Note: BitrateFrac bits have **no effect** (i.e may be considered equal to 0) **in OOK** modulation mode.*

The quantity *BitrateFrac* is hence designed to allow very high precision (max. 250 ppm programing resolution) for any bitrate in the programmable range. Table 19 below shows a range of standard bitrates and the accuracy to within which they may be reached.

Table 19 Bit Rate Examples

Type	BitRate (15:8)	BitRate (7:0)	(G)FSK (G)MSK	OOK	Actual BR (b/s)
Classical modem baud rates (multiples of 1.2 kbps)	0x68	0x2B	1.2 kbps	1.2 kbps	1200.015
	0x34	0x15	2.4 kbps	2.4 kbps	2400.060
	0x1A	0x0B	4.8 kbps	4.8 kbps	4799.760
	0x0D	0x05	9.6 kbps	9.6 kbps	9600.960
	0x06	0x83	19.2 kbps	19.2 kbps	19196.16
	0x03	0x41	38.4 kbps		38415.36
	0x01	0xA1	76.8 kbps		76738.60
	0x00	0xD0	153.6 kbps		153846.1
Classical modem baud rates (multiples of 0.9 kbps)	0x02	0x2C	57.6 kbps		57553.95
	0x01	0x16	115.2 kbps		115107.9
Round bit rates (multiples of 12.5, 25 and 50 kbps)	0x0A	0x00	12.5 kbps	12.5 kbps	12500.00
	0x05	0x00	25 kbps	25 kbps	25000.00
	0x80	0x00	50 kbps		50000.00
	0x01	0x40	100 kbps		100000.0
	0x00	0xD5	150 kbps		150234.7
	0x00	0xA0	200 kbps		200000.0
	0x00	0x80	250 kbps		250000.0
	0x00	0x6B	300 kbps		299065.4
Watch Xtal frequency	0x03	0xD1	32.768 kbps	32.768 kbps	32753.32

4.2.2. FSK/OOK Transmission

4.2.2.1. FSK Modulation

FSK modulation is performed inside the PLL bandwidth, by changing the fractional divider ratio in the feedback loop of the PLL. The high resolution of the sigma-delta modulator, allows for very narrow frequency deviation. The frequency deviation F_{DEV} is given by:

$$F_{DEV} = F_{STEP} \times Fdev(13,0)$$

To ensure correct modulation, the following limit applies:

$$F_{DEV} + \frac{BR}{2} \leq (250)kHz$$

Note No constraint applies to the modulation index of the transmitter, but the frequency deviation must be set between 600 Hz and 200 kHz.

4.2.2.2. OOK Modulation

OOK modulation is applied by switching on and off the power amplifier. Digital control and ramping are available to improve the transient power response of the OOK transmitter.

4.2.2.3. Modulation Shaping

Modulation shaping can be applied in both OOK and FSK modulation modes, to improve the narrow band response of the transmitter. Both shaping features are controlled with *PaRamp* bits in *RegPaRamp*.

- In FSK mode, a Gaussian filter with BT = 0.5 or 1 is used to filter the modulation stream, at the input of the sigma-delta modulator. If the Gaussian filter is enabled when the SX1276/77/78/79 is in Continuous mode, DCLK signal on pin 10 (DIO1/DCLK) will trigger an interrupt on the uC each time a new bit has to be transmitted. Please refer to section 4.2.12.2 for details.
- When OOK modulation is used, the PA bias voltages are ramped up and down smoothly when the PA is turned on and off, to reduce spectral splatter.

Note The transmitter must be restarted if the ModulationShaping setting is changed, in order to recalibrate the built-in filter.

4.2.3. FSK/OOK Reception

4.2.3.1. FSK Demodulator

The FSK demodulator of the SX1276/77/78/79 is designed to demodulate FSK, GFSK, MSK and GMSK modulated signals. It is most efficient when the modulation index of the signal is greater than 0.5 and below 10:

$$0.5 \leq \beta = \frac{2 \times F_{DEV}}{BR} \leq 10$$

The output of the FSK demodulator can be fed to the Bit Synchronizer to provide the companion processor with a synchronous data stream in Continuous mode.

4.2.3.2. OOK Demodulator

The OOK demodulator performs a comparison of the RSSI output and a threshold value. Three different threshold modes are available, configured through bits *OokThreshType* in *RegOokPeak*.

The recommended mode of operation is the “Peak” threshold mode, illustrated in Figure 14:

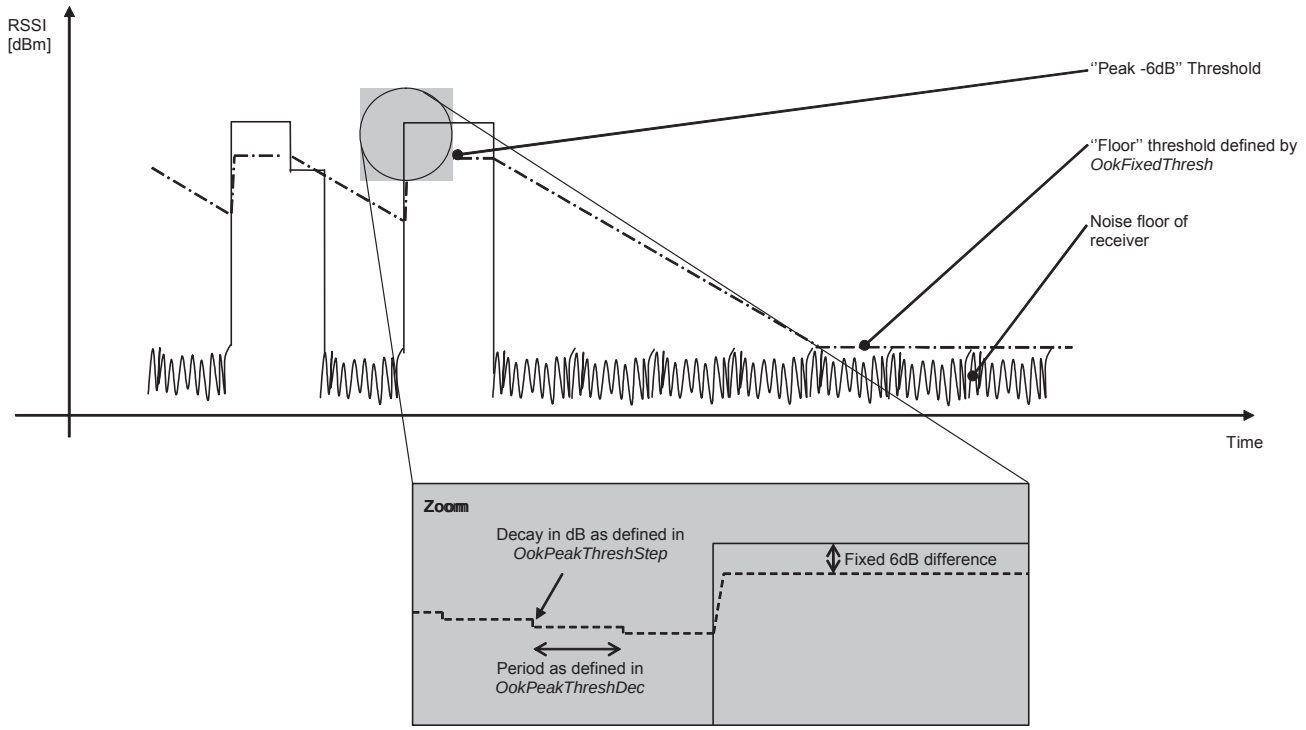


Figure 14. OOK Peak Demodulator Description

In peak threshold mode the comparison threshold level is the peak value of the RSSI, reduced by 6dB. In the absence of an input signal, or during the reception of a logical '0', the acquired peak value is decremented by one *OokPeakThreshStep* every *OokPeakThreshDec* period.

When the RSSI output is null for a long time (for instance after a long string of "0" received, or if no transmitter is present), the peak threshold level will continue falling until it reaches the "Floor Threshold", programmed in *OokFixedThresh*.

The default settings of the OOK demodulator lead to the performance stated in the electrical specification. However, in applications in which sudden signal drops are awaited during a reception, the three parameters should be optimized accordingly.

Optimizing the Floor Threshold

OokFixedThresh determines the sensitivity of the OOK receiver, as it sets the comparison threshold for weak input signals (i.e. those close to the noise floor). Significant sensitivity improvements can be generated if configured correctly.

Note that the noise floor of the receiver at the demodulator input depends on:

- The noise figure of the receiver.
- The gain of the receive chain from antenna to base band.
- The matching - including SAW filter if any.
- The bandwidth of the channel filters.

It is therefore important to note that the setting of *OokFixedThresh* will be application dependent. The following procedure is recommended to optimize *OokFixedThresh*.

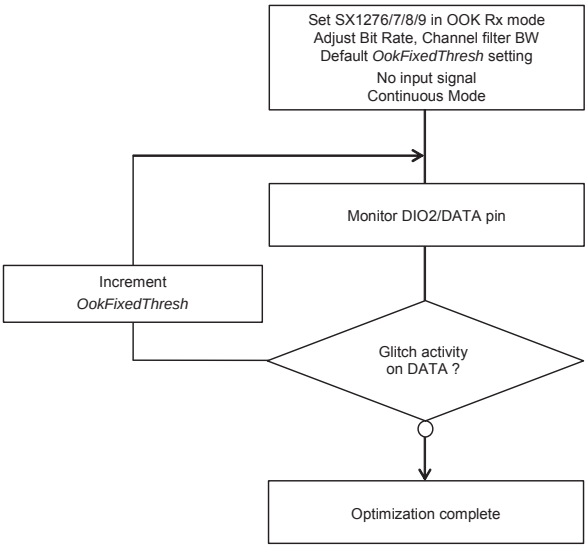


Figure 15. Floor Threshold Optimization

The new floor threshold value found during this test should be used for OOK reception with those receiver settings.

Optimizing OOK Demodulator for Fast Fading Signals

A sudden drop in signal strength can cause the bit error rate to increase. For applications where the expected signal drop can be estimated, the following OOK demodulator parameters *OokPeakThreshStep* and *OokPeakThreshDec* can be optimized as described below for a given number of threshold decrements per bit. Refer to *RegOokPeak* to access those settings.

Alternative OOK Demodulator Threshold Modes

In addition to the Peak OOK threshold mode, the user can alternatively select two other types of threshold detectors:

- Fixed Threshold: The value is selected through *OokFixedThresh*
- Average Threshold: Data supplied by the RSSI block is averaged, and this operation mode should only be used with DC-free encoded data.

4.2.3.3. Bit Synchronizer

The bit synchronizer provides a clean and synchronized digital output based upon timing recovery information gleaned from the received data edge transitions. Its output is made available on pin DIO1/DCLK in Continuous mode and can be disabled through register settings. However, for optimum receiver performance, especially in Continuous receive mode, its use is strongly advised.

The Bit Synchronizer is automatically activated in Packet mode. Its bit rate is controlled by *BitRateMsb* and *BitRateLsb* in *RegBitrate.*

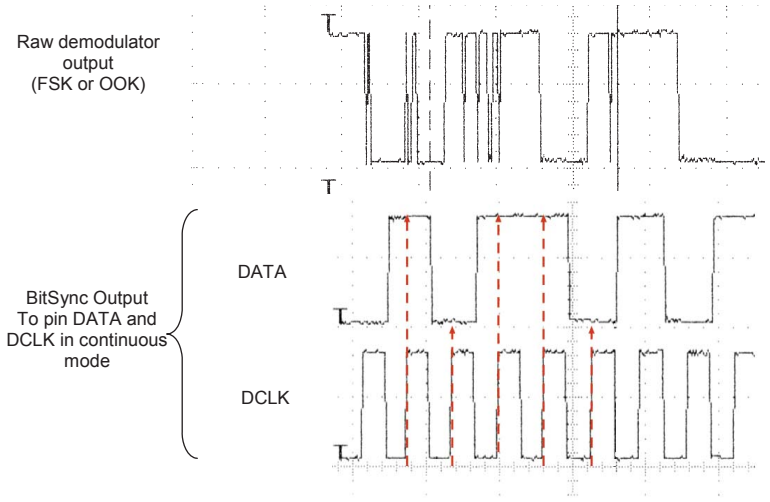


Figure 16. Bit Synchronizer Description

To ensure correct operation of the Bit Synchronizer, the following conditions have to be satisfied:

- A preamble (0x55 or 0xAA) of at least 12 bits is required for synchronization, the longer the synchronization phase is the better the ensuing packet detection rate will be.
- The subsequent payload bit stream must have at least one edge transition (either rising or falling) every 16 bits during data transmission.
- The absolute error between transmitted and received bit rate must not exceed 6.5%.

4.2.3.4. Frequency Error Indicator

This frequency error indicator measures the frequency error between the programmed RF centre frequency and the carrier frequency of the modulated input signal to the receiver. When the FEI is performed, the frequency error is measured and the signed result is loaded in *FeiValue* in *RegFei*, in 2's complement format. The time required for an FEI evaluation is 4 bit periods.

To ensure correct operation of the FEI:

- The measurement must be launched during the reception of preamble.
- The sum of the frequency offset and the 20 dB signal bandwidth must be lower than the base band filter bandwidth. i.e. The whole modulated spectrum must be received.

The 20 dB bandwidth of the signal can be evaluated as follows (double-side bandwidth):

$$BW_{20dB} = 2 \times \left(F_{DEV} + \frac{BR}{2} \right)$$

The frequency error, in Hz, can be calculated with the following formula:

$$FEI = F_{STEP} \times FeiValue$$

The FEI is enabled automatically upon the transition to receive mode and automatically updated every 4 bits.

4.2.3.5. AFC

The AFC is based on the FEI measurement, therefore the same input signal and receiver setting conditions apply. When the AFC procedure is performed the *AfcValue* is directly subtracted from the register that defines the frequency of operation of the chip, F_{RF}. The AFC is executed each time the receiver is enabled, if *AfcAutoOn* = 1.

When the AFC is enabled (*AfcAutoOn* = 1), the user has the option to:

- Clear the former AFC correction value, if *AfcAutoClearOn* = 1. Allowing the next frequency correction to be performed from the initial centre frequency.
- Start the AFC evaluation from the previously corrected frequency. This may be useful in systems in which the centre frequency experiences cumulative drift - such as the ageing of a crystal reference.

The SX1276/77/78/79 offers an alternate receiver bandwidth setting during the AFC phase allowing the accommodation of larger frequency errors. The setting *RegAfcBw* sets the receive bandwidth during the AFC process. In a typical receiver application the, once the AFC is performed, the radio will revert to the receiver communication or channel bandwidth (*RegRxBw*) for the ensuing communication phase.

Note that the FEI measurement is valid only during the reception of preamble. The provision of the *PreambleDetect* flag can hence be used to detect this condition and allow a reliable AFC or FEI operation to be triggered. This process can be performed automatically by using the appropriate options in *StartDemodOnPreamble* found in the *RegRxConfig* register.

A detailed description of the receiver setup to enable the AFC is provided in section 4.2.6.

4.2.3.6. Preamble Detector

The Preamble Detector indicates the reception of a carrier modulated with a 0101...sequence. It is insensitive to the frequency offset, as long as the receiver bandwidth is large enough. The size of detection can be programmed from 1 to 3 bytes with *PreambleDetectorSize* in *RegPreambleDetect* as defined in the next table.

Table 20 Preamble Detector Settings

PreambleDetectorSize	# of Bytes
00	1
01	2 (recommended)
10	3
11	reserved

For normal operation, *PreambleDetectTol* should be set to be set to 10 (0x0A), with a qualifying preamble size of 2 bytes.

The *PreambleDetect* interrupt (either in *RegIrqFlags1* or mapped to a specific DIO) then goes high every time a valid preamble is detected, assuming *PreambleDetectorOn*=1.

The preamble detector can also be used as a gate to ensure that AFC and AGC are performed on valid preamble. See section 4.2.6. for details.

4.2.3.7. Image Rejection Mixer

The SX1276/77/78/79 employs an image rejection mixer (IRM) which, uncalibrated, 35 dB image rejection. A low phase noise PLL is used to perform calibration of the receiver chain. This increases the typical image rejection to 48 dB.

4.2.3.8. Image and RSSI Calibration

An automated process is implemented to calibrate the phase and gain imbalances of I and Q receive paths. This calibration enhances image rejection and improves RSSI precision. It is launched under the following circumstances:

- Automatically at Power On Reset or after a Manual Reset of the chip (refer to section 7.2), only for the Low Frequency front-end, and is performed at 434MHz
- Automatically when a pre-defined temperature change is observed, if the option is enabled. A selectable temperature change, set with *TempThreshold* (5, 10, 15 or 20°C), is detected and reported in *TempChange*, if the temperature monitoring is turned On with *TempMonitorOff*=0.This interrupt flag can be used by the application to launch a new image calibration at a convenient time if *AutoImageCalOn*=0, or immediately when this temperature variation is detected, if *AutoImageCalOn*=1
- Upon user request, by setting bit *ImageCalStart* in *RegImageCal*, when the device is in Standby mode

Notes - The calibration procedure takes approximately 10ms. It is recommended to disable the fully automated (temperature-dependent) calibration, to better control when it is triggered (and avoid unexpected packet losses)

- To perform the calibration, the radio must be temporarily returned to FSK/OOK mode

- The automatic IQ and RSSI calibration done at POR and Reset is only valid at 434 MHz (the value of RegFrf at POR). To improve accuracy of RSSI and image rejection, this calibration should be replicated at the frequency (ies)

of interest, for instance a calibration should be launched with Frf set to 868.3 MHz if the high frequency port supports communication in this frequency band. Conversely if the product is used at 169 MHz, the calibration should be repeated with Frf=169MHz

- FormerTemp and TempChange in SX1276/77/79 are frequency-specific and the IC keeps a copy of these variables when switching between the low frequency and the high frequency domains (along with the corresponding calibration values, stored in test registers)

- FormerTemp and TempChange cannot be read in Sleep mode (although they are saved). They should be read in Standby mode

4.2.3.9. Timeout Function

The SX1276/77/78/79 includes a Timeout function, which allows it to automatically shut-down the receiver after a receive sequence and therefore save energy.

- *Timeout* interrupt is generated *TimeoutRxRssi x 16 x Tbit* after switching to Rx mode if the *Rssi* flag does not raise within this time frame (*RssiValue* > *RssiThreshold*)
- *Timeout* interrupt is generated *TimeoutRxPreamble x 16 x Tbit* after switching to Rx mode if the *PreambleDetect* flag does not raise within this time frame
- *Timeout* interrupt is generated *TimeoutSignalSync x 16 x Tbit* after switching to Rx mode if the *SyncAddress* flag does not raise within this time frame

This timeout interrupt can be used to warn the companion processor to shut down the receiver and return to a lower power mode. To become active, these timeouts must also be enabled by setting the correct *RxTrigger* parameters in *RegRxConfig:*

Table 21 RxTrigger Settings to Enable Timeout Interrupts

Receiver Triggering Event	*RxTrigger (2:0)*	*Timeout on Rssi*	*Timeout on Preamble*	*Timeout on SyncAddress*
None	000	Off	Off	Active
Rssi Interrupt	001	Active	Off	
PreambleDetect	110	Off	Active	
Rssi Interrupt & *PreambleDetect*	111	Active	Active	

4.2.4. Operating Modes in FSK/OOK Mode

The SX1276/77/78/79 has several working modes, manually programmed in *RegOpMode*. Fully automated mode selection, packet transmission and reception is also possible using the Top Level Sequencer described in Section 4.2.8.

Table 22 Basic Transceiver Modes

Mode	Selected mode	Symbol	Enabled blocks
000	Sleep mode	Sleep	None
001	Standby mode	Stdby	Top regulator and crystal oscillator
010	Frequency synthesiser to Tx frequency	FSTx	Frequency synthesizer at Tx frequency (Frf)
011	Transmit mode	Tx	Frequency synthesizer and transmitter

Mode	Selected mode	Symbol	Enabled blocks
100	Frequency synthesiser to Rx frequency	FSRx	Frequency synthesizer at frequency for reception (Frf-IF)
101	Receive mode	Rx	Frequency synthesizer and receiver

When switching from a mode to another the sub-blocks are woken up according to a pre-defined optimized sequence.

4.2.5. Startup Times

The startup time of the transmitter or the receiver is Dependant upon which mode the transceiver was in at the beginning. For a complete description, Figure 17 below shows a complete startup process, from the lower power mode “Sleep”.

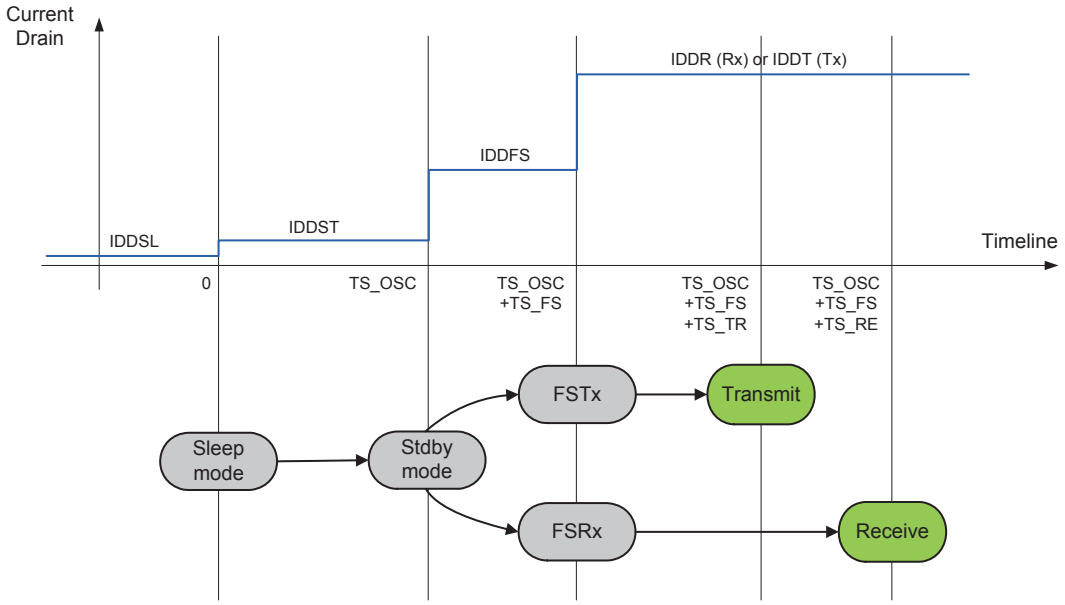


Figure 17. Startup Process

TS_OSC is the startup time of the crystal oscillator which depends on the electrical characteristics of the crystal. TS_FS is the startup time of the PLL including systematic calibration of the VCO.

Typical values of TS_OSC and TS_FS are given in Section 2.5.2.

4.2.5.1. Transmitter Startup Time

The transmitter startup time, TS_TR, is calculated as follows in FSK mode:

$$TS\_TR = 5\mu s + 1.25 \times PaRamp + \frac{1}{2} \times Tbit$$

,

where *PaRamp* is the ramp-up time programmed in *RegPaRamp* and *Tbit* is the bit time.

In OOK mode, this equation can be simplified to the following:

$$TS\_TR = 5\mu s + \frac{1}{2} \times Tbit$$

4.2.5.2. Receiver Startup Time

The receiver startup time, TS_RE, only depends upon the receiver bandwidth effective at the time of startup. When AFC is enabled (*AfcAutoOn*=1), *AfcBw* should be used instead of *RxBw* to extract the receiver startup time:

Table 23 Receiver Startup Time Summary

RxBw if *AfcAutoOn=0* *RxBwAfc* if *AfcAutoOn=1*	TS_RE (+/-5%)
2.6 kHz	2.33 ms
3.1 kHz	1.94 ms
3.9 kHz	1.56 ms
5.2 kHz	1.18 ms
6.3 kHz	984 us
7.8 kHz	791 us
10.4 kHz	601 us
12.5 kHz	504 us
15.6 kHz	407 us
20.8 kHz	313 us
25.0 kHz	264 us
31.3 kHz	215 us
41.7 kHz	169 us
50.0 kHz	144 us
62.5 kHz	119 us
83.3 kHz	97 us
100.0 kHz	84 us
125.0 kHz	71 us
166.7 kHz	85 us
200.0 kHz	74 us
250.0 kHz	63 us

TS_RE or later after setting the device in Receive mode, any incoming packet will be detected and demodulated by the transceiver.

4.2.5.3. Time to RSSI Evaluation

The first RSSI sample will be available TS_RSSI after the receiver is ready, in other words TS_RE + TS_RSSI after the receiver was requested to turn on.

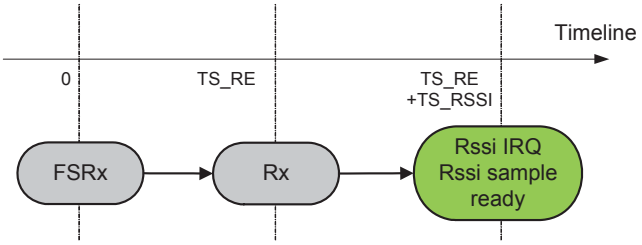


Figure 18. Time to RSSI Sample

TS_RSSI depends on the receiver bandwidth, as well as the *RssiSmoothing* option that was selected. The formula used to calculate TS_RSSI is provided in section 5.5.4.

4.2.5.4. Tx to Rx Turnaround Time

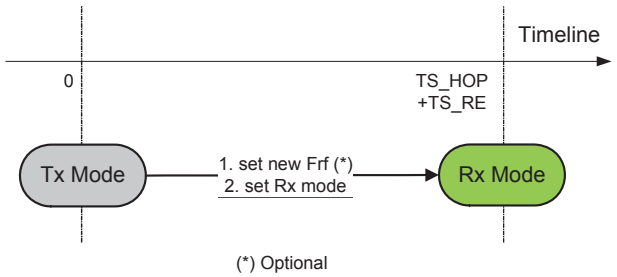


Figure 19. Tx to Rx Turnaround

Note The SPI instruction times are omitted, as they can generally be very small as compared to other timings (up to 10MHz SPI clock).

4.2.5.5. Rx to Tx

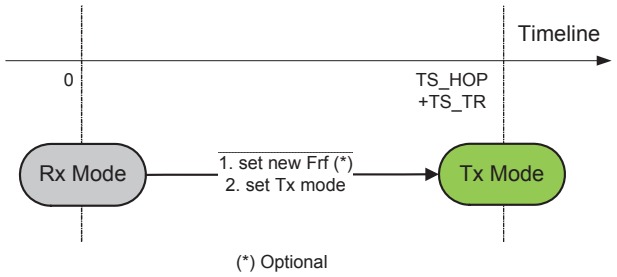


Figure 20. Rx to Tx Turnaround

4.2.5.6. Receiver Hopping, Rx to Rx

Two methods are possible:

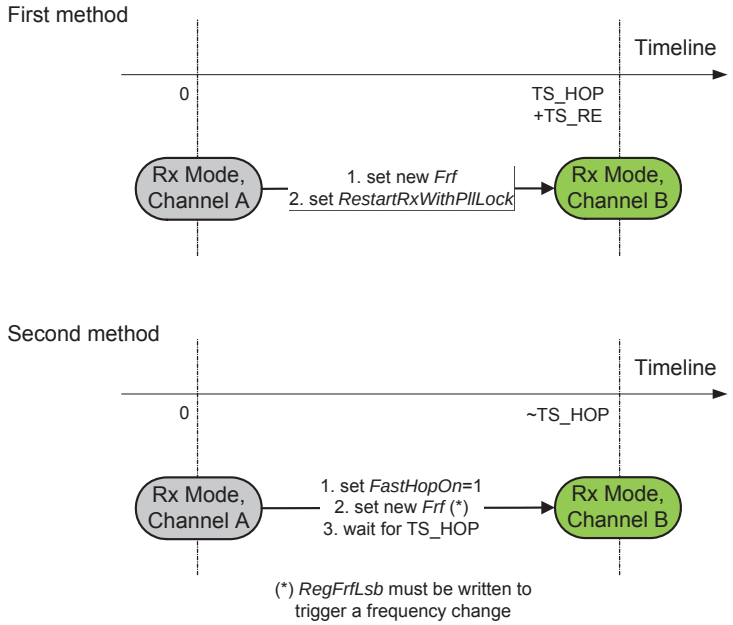


Figure 21. Receiver Hopping

The second method is quicker, and should be used if a very quick RF sniffing mechanism is to be implemented.

4.2.5.7. Tx to Tx

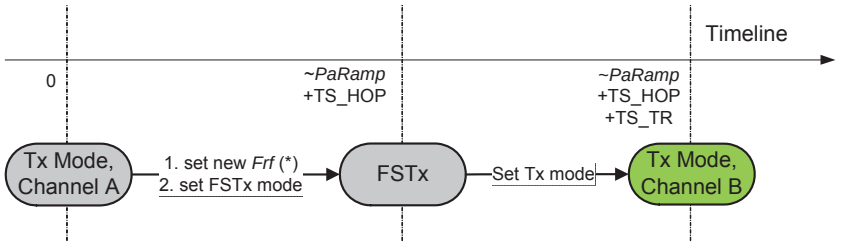


Figure 22. Transmitter Hopping

4.2.6. Receiver Startup Options

The SX1276/77/78/79 receiver can automatically control the gain of the receive chain (AGC) and adjust the receiver LO frequency (AFC). Those processes are carried out on a packet-by-packet basis. They occur:

- When the receiver is turned On.
- When the Receiver is restarted upon user request, through the use of trigger bits *RestartRxWithoutPllLock* or *RestartRxWithPllLock*, in *RegRxConfig.*
- When the receiver is automatically restarted after the reception of a valid packet, or after a packet collision.

Automatic restart capabilities are detailed in Section 4.2.7.

The receiver startup options available in SX1276/77/78/79 are described in Table 24.

Table 24 Receiver Startup Options

Triggering Event	Realized Function	*AgcAutoOn*	*AfcAutoOn*	*RxTrigger (2:0)*
None	None	0	0	000
Rssi Interrupt	AGC	1	0	001
	AGC & AFC	1	1	001
PreambleDetect	AGC	1	0	110
	AGC & AFC	1	1	110
Rssi Interrupt & *PreambleDetect*	AGC	1	0	111
	AGC & AFC	1	1	111

When *AgcAutoOn*=0, the LNA gain is manually selected by choosing *LnaGain* bits in *RegLna.*

4.2.7. Receiver Restart Methods

The options for restart of the receiver are covered below. This is typically of use to prepare for the reception of a new signal whose strength or carrier frequency is different from the preceding packet to allow the AGC or AFC to be re-evaluated.

4.2.7.1. Restart Upon User Request

In Receive mode the user can request a receiver restart - this can be useful in conjunction with the use of a Timeout interrupt following a period of inactivity in the channel of interest. Two options are available:

- No change in the Local Oscillator upon restart: the AFC is disabled, and the *Frf* register has not been changed through SPI before the restart instruction: set bit *RestartRxWithoutPllLock* in *RegRxConfig* to 1.
- Local Oscillator change upon restart: if AFC is enabled (*AfcAutoOn*=1), and/or the *Frf* register had been changed during the last Rx period: set bit *RestartRxWithPllLock* in *RegRxConfig* to 1.

Note ModeReady must be at logic level 1 for a new RestartRx command to be taken into account.

4.2.7.2. Automatic Restart after valid Packet Reception

The bits *AutoRestartRxMode* in *RegSyncConfig* control the automatic restart feature of the SX1276/77/78/79 receiver, when a valid packet has been received:

- If *AutoRestartRxMode* = 00, the function is off, and the user should manually restart the receiver upon valid packet reception (see section 4.2.7.1*).*
- If *AutoRestartRxMode* = 01, after the user has emptied the FIFO following a *PayloadReady* interrupt, the receiver will automatically restart itself after a delay of *InterPacketRxDelay*, allowing for the distant transmitter to ramp down, hence avoiding a false RSSI detection on the 'tail' of the previous packet.
- If *AutoRestartRxMode* = 10 should be used if the next reception is expected on a new frequency, i.e. *Frf* is changed after the reception of the previous packet. An additional delay is systematically added, in order for the PLL to lock at a new frequency.

4.2.7.3. Automatic Restart when Packet Collision is Detected

In receive mode the SX1276/77/78/79 is able to detect packet collision and restart the receiver. Collisions are detected by a sudden rise in received signal strength, detected by the RSSI. This functionality can be useful in network configurations where many asynchronous slaves attempt periodic communication with a single a master node.

The collision detector is enabled by setting bit *RestartRxOnCollision* to 1.

The decision to restart the receiver is based on the detection of RSSI change. The sensitivity of the system can be adjusted in 1 dB steps by using register *RssiCollisionThreshold* in *RegRxConfig*.

4.2.8. Top Level Sequencer

Depending on the application, it is desirable to be able to change the mode of the circuit according to a predefined sequence without access to the serial interface. In order to define different sequences or scenarios, a user-programmable state machine, called Top Level Sequencer (Sequencer in short), can automatically control the chip modes.

NOTE THAT THIS FUNCTIONALITY IS ONLY AVAILABLE IN FSK/OOK MODE.

The Sequencer is activated by setting the *SequencerStart* bit in *RegSeqConfig1* to 1 in Sleep or Standby mode (called initial mode).

It is also possible to force the Sequencer off by setting the *Stop* bit in *RegSeqConfig1* to 1 at any time.

Note SequencerStart and Stop bit must never be set at the same time.

4.2.8.1. Sequencer States

As shown in the table below, with the aid of a pair of interrupt timers (T1 and T2), the sequencer can take control of the chip operation in all modes.

Table 25 Sequencer States

Sequencer State	Description
SequencerOff State	The Sequencer is not activated. Sending a *SequencerStart* command will launch it. When coming from **LowPowerSelection** state, the Sequencer will be Off, whilst the chip will return to its initial mode (either Sleep or Standby mode).
Idle State	The chip is in low-power mode, either *Standby* or *Sleep*, as defined by *IdleMode* in *RegSeqConfig1*. The Sequencer waits only for the *T1* interrupt.
Transmit State	The transmitter in on.
Receive State	The receiver in on.
PacketReceived	The receiver is on and a packet has been received. It is stored in the FIFO.
LowPowerSelection	Selects low power state (**SequencerOff** or **Idle** State)
RxTimeout	Defines the action to be taken on a RxTimeout interrupt. RxTimeout interrupt can be a *TimeoutRxRssi*, *TimeoutRxPreamble* or *TimeoutSignalSync* interrupt.

4.2.8.2. Sequencer Transitions

The transitions between sequencer states are listed in the forthcoming table.

Table 26 Sequencer Transition Options

Variable	Transition
IdleMode	Selects the chip mode during **Idle** state: 0: *Standby* mode 1: *Sleep* mode
FromStart	Controls the Sequencer transition when the *SequencerStart* bit is set to 1 in *Sleep* or *Standby* mode: 00: to **LowPowerSelection** 01: to **Receive** state 10: to **Transmit** state 11: to **Transmit** state on a *FifoThreshold* interrupt
LowPowerSelection	Selects Sequencer LowPower state after a *to LowPowerSelection* transition 0: **SequencerOff** state with chip on Initial mode 1: **Idle** state with chip on *Standby* or *Sleep* mode depending on **IdleMode** Note: Initial mode is the chip LowPower mode at Sequencer start.
FromIdle	Controls the Sequencer transition from the **Idle** state on a *T1* interrupt: 0: to **Transmit** state 1: to **Receive** state
FromTransmit	Controls the Sequencer transition from the **Transmit** state: 0: to **LowPowerSelection** on a *PacketSent* interrupt 1: to **Receive** state on a *PacketSent* interrupt
FromReceive	Controls the Sequencer transition from the **Receive** state: 000 and 111: unused 001: to **PacketReceived** state on a *PayloadReady* interrupt 010: to **LowPowerSelection** on a *PayloadReady* interrupt 011: to **PacketReceived** state on a *CrcOk* interrupt. If CRC is wrong (corrupted packet, with CRC on but CrcAutoClearOn is off), the PayloadReady interrupt will drive the sequencer to RxTimeout state. 100: to **SequencerOff** state on a *Rssi* interrupt 101: to **SequencerOff** state on a *SyncAddress* interrupt 110: to **SequencerOff** state on a *PreambleDetect* interrupt Irrespective of this setting, transition to **LowPowerSelection** on a *T2* interrupt
FromRxTimeout	Controls the state-machine transition from the **Receive** state on a *RxTimeout* interrupt (and on *PayloadReady* if **FromReceive** = 011): 00: to **Receive** state via *ReceiveRestart* 01: to **Transmit** state 10: to **LowPowerSelection** 11: to **SequencerOff** state Note: RxTimeout interrupt is a *TimeoutRxRssi*, *TimeoutRxPreamble* or *TimeoutSignalSync* interrupt.
FromPacketReceived	Controls the state-machine transition from the **PacketReceived** state: 000: to **SequencerOff** state 001: to **Transmit** on a *FifoEmpty* interrupt 010: to **LowPowerSelection** 011: to **Receive** via *FS* mode, if frequency was changed 100: to **Receive** state (no frequency change)

4.2.8.3. Timers

Two timers (Timer1 and Timer2) are also available in order to define periodic sequences. These timers are used to generate interrupts, which can trigger transitions of the Sequencer.

T1 interrupt is generated (Timer1Resolution * Timer1Coefficient) after ***T2* interrupt** or ***SequencerStart***. command.
T2 interrupt is generated (Timer2Resolution * Timer2Coefficient) after ***T1* interrupt**.

The timers' mechanism is summarized on the following diagram.

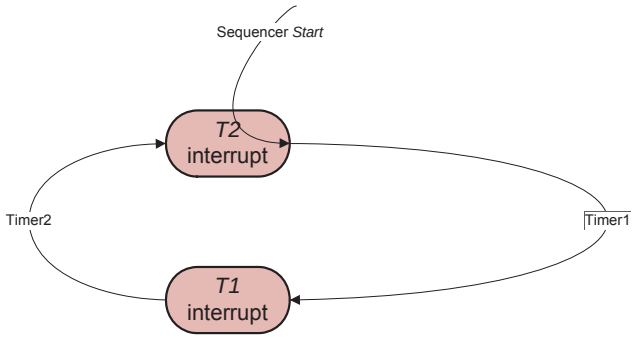


Figure 23. Timer1 and Timer2 Mechanism

Note The timer sequence is completed independently of the actual Sequencer state. Thus, both timers need to be on to achieve periodic cycling.

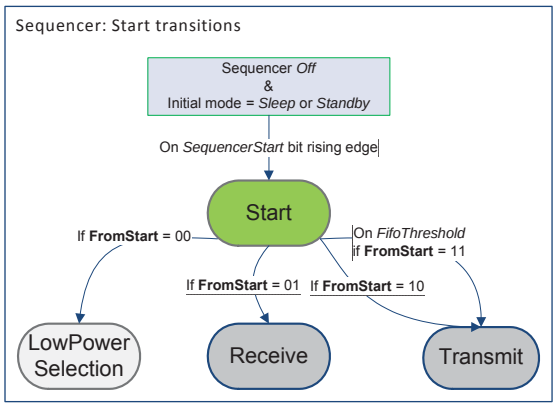
Table 27 Sequencer Timer Settings

Variable	Description
Timer1Resolution	Resolution of Timer1 00: disabled 01: 64 us 10: 4.1 ms 11: 262 ms
Timer2Resolution	Resolution of Timer2 00: disabled 01: 64 us 10: 4.1 ms 11: 262 ms
Timer1Coefficient	Multiplying coefficient for Timer1
Timer2Coefficient	Multiplying coefficient for Timer2

4.2.8.4. Sequencer State Machine

The following graphs summarize every possible transition between each Sequencer state. The Sequencer states are highlighted in grey. The transitions are represented by arrows. The condition activating them is described over the transition arrow. For better readability, the start transitions are separated from the rest of the graph.

Transitory states are highlighted in light grey, and exit states are represented in red. It is also possible to force the Sequencer off by setting the *Stop* bit in *RegSeqConfig1* to 1 at any time.



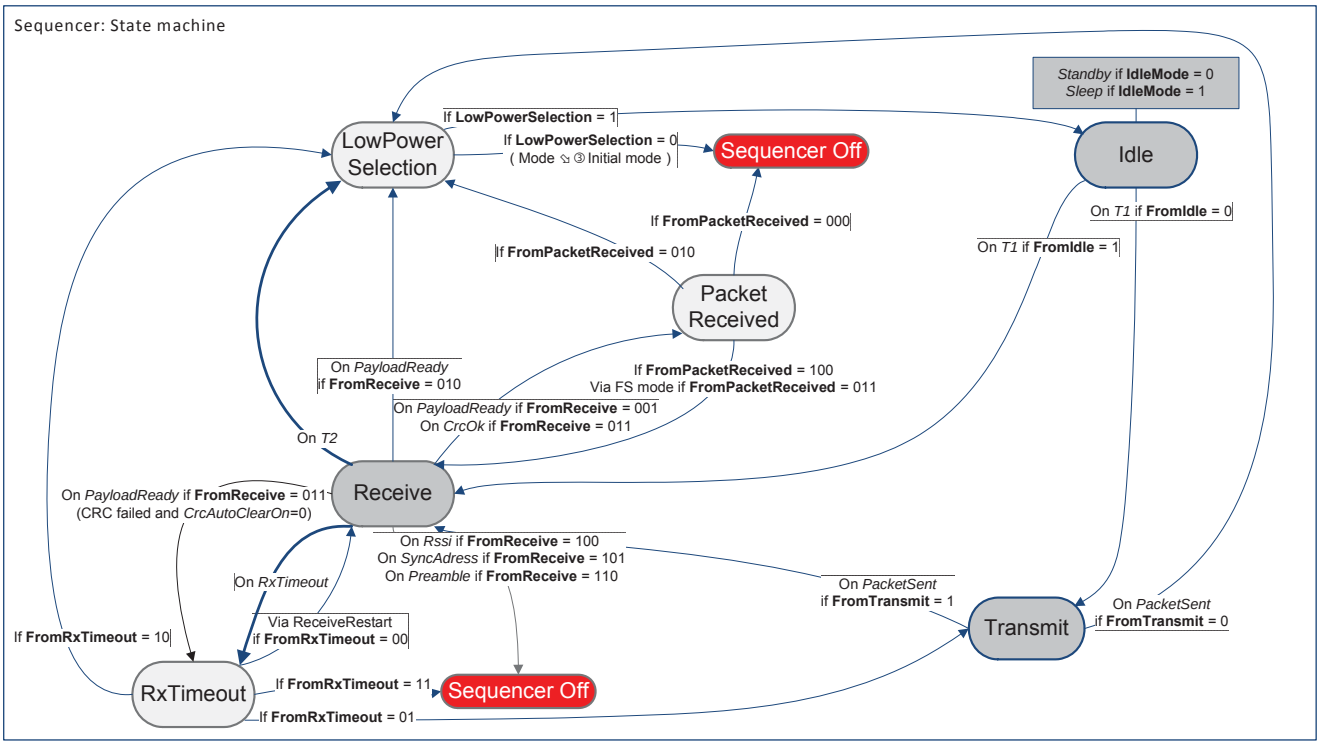


Figure 24. Sequencer State Machine

4.2.9. Data Processing in FSK/OOK Mode

4.2.9.1. Block Diagram

Figure below illustrates the SX1276/77/78/79 data processing circuit. Its role is to interface the data to/from the modulator/ demodulator and the uC access points (SPI and DIO pins). It also controls all the configuration registers.

The circuit contains several control blocks which are described in the following paragraphs.

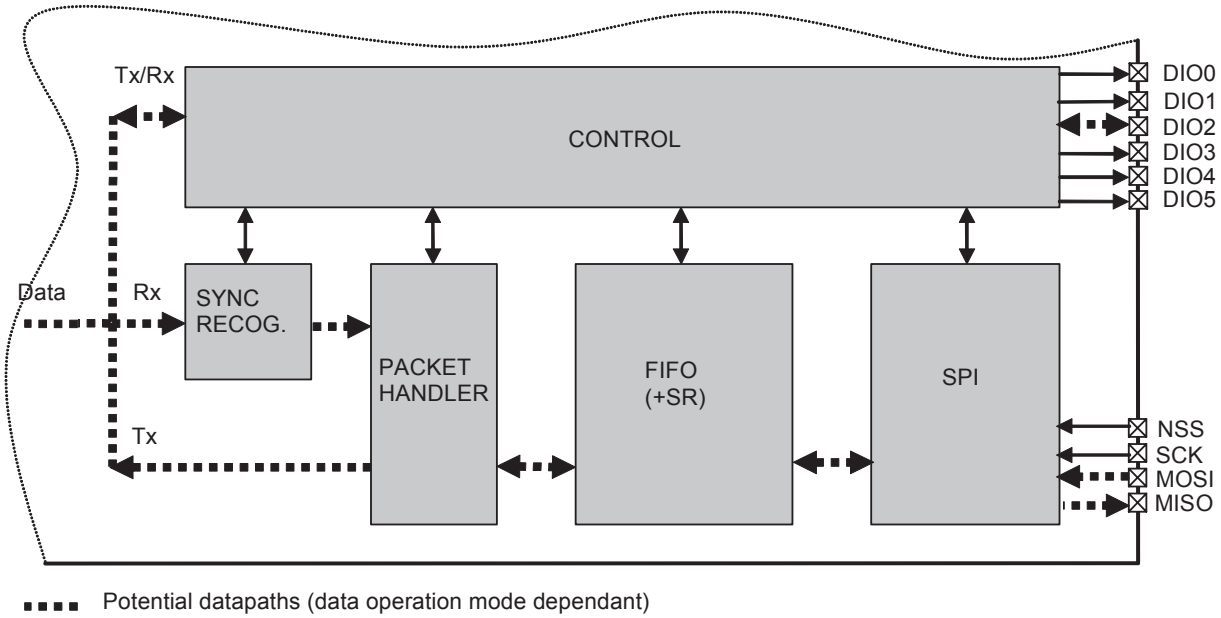


Figure 25. SX1276/77/78/79 Data Processing Conceptual View

The SX1276/77/78/79 implements several data operation modes, each with their own data path through the data processing. Depending on the data operation mode selected, some control blocks are active whilst others remain disabled.

4.2.9.2. Data Operation Modes

The SX1276/77/78/79 has two different data operation modes selectable by the user:

- Continuous mode: each bit transmitted or received is accessed in real time at the DIO2/DATA pin. This mode may be used if adequate external signal processing is available.
- Packet mode (recommended): user only provides/retrieves payload bytes to/from the FIFO. The packet is automatically built with preamble, Sync word, and optional CRC and DC-free encoding schemes The reverse operation is performed in reception. The uC processing overhead is hence significantly reduced compared to Continuous mode. Depending on the optional features activated (CRC, etc) the maximum payload length is limited to 255, 2047 bytes or unlimited.

Each of these data operation modes is fully described in the following s.

4.2.10. FIFO

Overview and Shift Register (SR)

In packet mode of operation, both data to be transmitted and that has been received are stored in a configurable FIFO (First In First Out) device. It is accessed via the SPI interface and provides several interrupts for transfer management.

The FIFO is 1 byte wide hence it only performs byte (parallel) operations, whereas the demodulator functions serially. A shift register is therefore employed to interface the two devices. In transmit mode it takes bytes from the FIFO and outputs them serially (MSB first) at the programmed bit rate to the modulator. Similarly, in Rx the shift register gets bit by bit data from the demodulator and writes them byte by byte to the FIFO. This is illustrated in figure below.

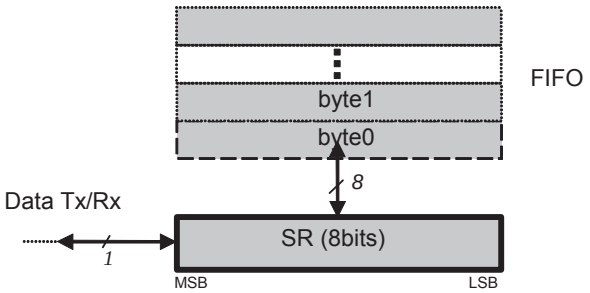


Figure 26. FIFO and Shift Register (SR)

Note When switching to Sleep mode, the FIFO can only be used once the ModeReady flag is set (quasi immediate from all modes except from Tx)

The FIFO size is fixed to 64 bytes.

Interrupt Sources and Flags

- *FifoEmpty*: *FifoEmpty* interrupt source is high when byte 0, i.e. whole FIFO, is empty. Otherwise it is low. Note that when retrieving data from the FIFO, *FifoEmpty* is updated on NSS falling edge, i.e. when *FifoEmpty* is updated to low state the currently started read operation must be completed. In other words, *FifoEmpty* state must be checked after each read operation for a decision on the next one (*FifoEmpty* = 0: more byte(s) to read; *FifoEmpty* = 1: no more byte to read).
- *FifoFull*: *FifoFull* interrupt source is high when the last FIFO byte, i.e. the whole FIFO, is full. Otherwise it is low.
- *FifoOverrunFlag*: *FifoOverrunFlag* is set when a new byte is written by the user (in Tx or Standby modes) or the SR (in Rx mode) while the FIFO is already full. Data is lost and the flag should be cleared by writing a 1, note that the FIFO will also be cleared.
- *PacketSent*: *PacketSent* interrupt source goes high when the SR's last bit has been sent.
- *FifoLevel*: Threshold can be programmed by *FifoThreshold* in *RegFifoThresh*. Its behavior is illustrated in figure below.

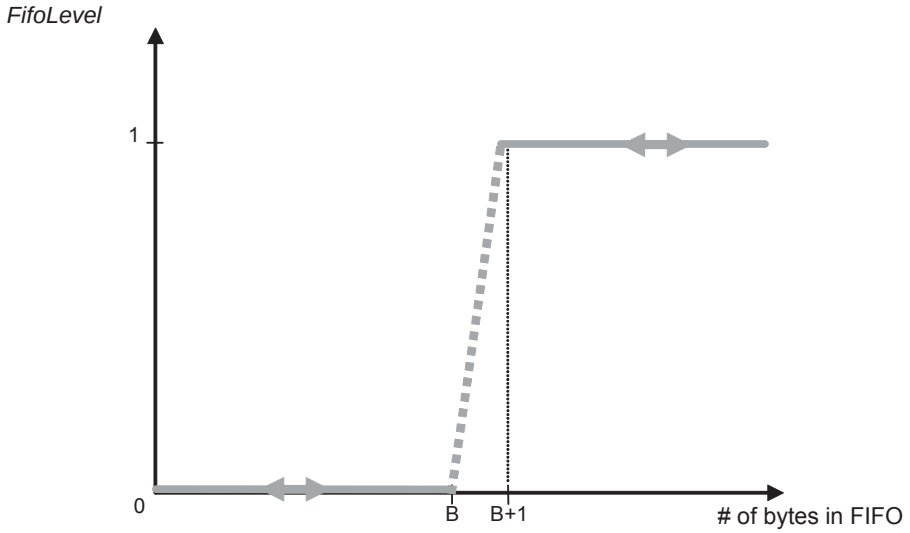


Figure 27. FifoLevel IRQ Source Behavior

Notes - FifoLevel interrupt is updated only after a read or write operation on the FIFO. Thus the interrupt cannot be dynamically updated by only changing the FifoThreshold parameter

- FifoLevel interrupt is valid as long as FifoFull does not occur. An empty FIFO will restore its normal operation

FIFO Clearing

Table below summarizes the status of the FIFO when switching between different modes

Table 28 Status of FIFO when Switching Between Different Modes of the Chip

From	To	FIFO status	Comments
Stdby	Sleep	Not cleared	
Sleep	Stdby	Not cleared	
Stdby/Sleep	Tx	Not cleared	To allow the user to write the FIFO in Stdby/Sleep before Tx
Stdby/Sleep	Rx	Cleared	
Rx	Tx	Cleared	
Rx	Stdby/Sleep	Not cleared	To allow the user to read FIFO in Stdby/Sleep mode after Rx
Tx	Any	Cleared	

4.2.10.1. Sync Word Recognition

Overview

Sync word recognition (also called Pattern recognition) is activated by setting *SyncOn* in *RegSyncConfig*. The bit synchronizer must also be activated in Continuous mode (automatically done in Packet mode).

The block behaves like a shift register; it continuously compares the incoming data with its internally programmed Sync word and sets *SyncAddressMatch* when a match is detected. This is illustrated in Figure 28 below.

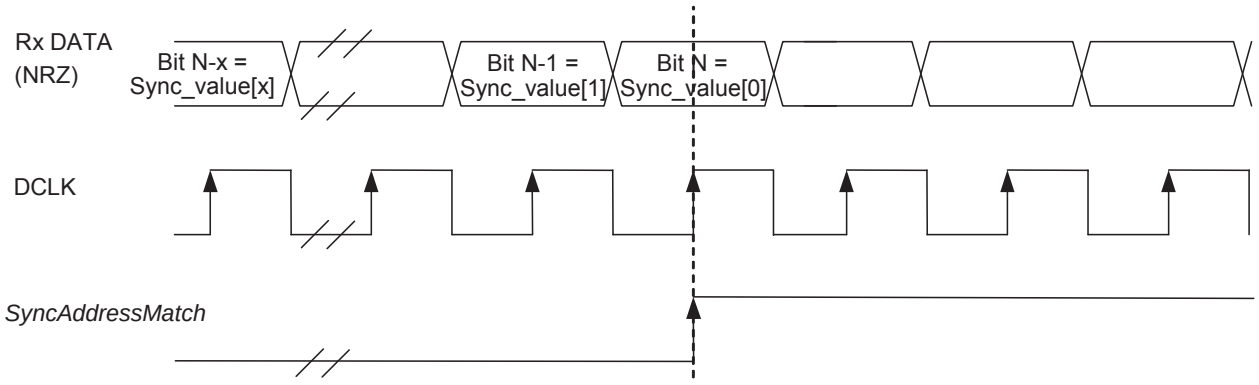


Figure 28. Sync Word Recognition

During the comparison of the demodulated data, the first bit received is compared with bit 7 (MSB) of *RegSyncValue1* and the last bit received is compared with bit 0 (LSB) of the last byte whose address is determined by the length of the Sync word.

When the programmed Sync word is detected the user can assume that this incoming packet is for the node and can be processed accordingly.

SyncAddressMatch is cleared when leaving Rx or FIFO is emptied.

Configuration

- Size: Sync word size can be set from 1 to 8 bytes (i.e. 8 to 64 bits) via *SyncSize* in *RegSyncConfig*. In Packet mode this field is also used for Sync word generation in Tx mode.
- Value: The Sync word value is configured in *SyncValue(63:0)*. In Packet mode this field is also used for Sync word generation in Tx mode.

Note SyncValue choices containing 0x00 bytes are not allowed

Packet Handler

The packet handler is the block used in Packet mode. Its functionality is fully described in Section 4.2.13.

Control

The control block configures and controls the full chip's behavior according to the settings programmed in the configuration registers.

4.2.11. Digital IO Pins Mapping

Six general purpose IO pins are available on the SX1276/77/78/79, and their configuration in Continuous or Packet mode is controlled through *RegDioMapping1* and *RegDioMapping2.*

Table 29 DIO Mapping, Continuous Mode

	DIOx Mapping	Sleep	Standby	FSRx/Tx	Rx	Tx
DIO0	00	-			SyncAddress	TxReady
	01	-			Rssi / PreambleDetect	-
	10	-			RxReady	TxReady
	11	-				
DIO1	00	-			Dclk	
	01	-			Rssi / PreambleDetect	-
	10	-				
	11	-				
DIO2	00	-			Data	
	01	-			Data	
	10	-			Data	
	11	-			Data	
DIO3	00	-			Timeout	-
	01	-			Rssi / PreambleDetect	-
	10	-				
	11	-	TempChange / LowBat		TempChange / LowBat	
DIO4	00	-		TempChange / LowBat		
	01	-		PllLock		
	10	-			TimeOut	-
	11	-	ModeReady		ModeReady	
DIO5	00	ClkOut if RC	ClkOut		ClkOut	
	01	-		PllLock		
	10	-			Rssi / PreambleDetect	-
	11	-	ModeReady		ModeReady	

Table 30 DIO Mapping, Packet Mode

	DIOx Mapping	Sleep	Standby	FSRx/Tx	Rx	Tx
DIO0	00	-			PayloadReady	PacketSent
	01	-			CrcOk	-
	10	-				
	11	-	TempChange / LowBat		TempChange / LowBat	
DIO1	00	FifoLevel		FifoLevel	FifoLevel	
	01	FifoEmpty		FifoEmpty	FifoEmpty	
	10	FifoFull		FifoFull	FifoFull	
	11	-				
DIO2	00	FifoFull		FifoFull	FifoFull	
	01	-			RxReady	-
	10	FifoFull			TimeOut	FifoFull
	11	FifoFull			SyncAddress	FifoFull
DIO3	00	FifoEmpty		FifoEmpty	FifoEmpty	
	01	-				TxReady
	10	FifoEmpty		FifoEmpty	FifoEmpty	
	11	FifoEmpty		FifoEmpty	FifoEmpty	
DIO4	00	-	TempChange / LowBat		TempChange / LowBat	
	01	-		PllLock		
	10	-			TimeOut	-
	11	-			Rssi / PreambleDetect	-
DIO5	00	ClkOut if RC	ClkOut		ClkOut	
	01	-		PllLock		
	10	-			Data	
	11	-	ModeReady		ModeReady	

4.2.12. Continuous Mode

4.2.12.1. General Description

As illustrated in Figure 29, in Continuous mode the NRZ data to (from) the (de)modulator is directly accessed by the uC on the bidirectional DIO2/DATA pin. The FIFO and packet handler are thus inactive.

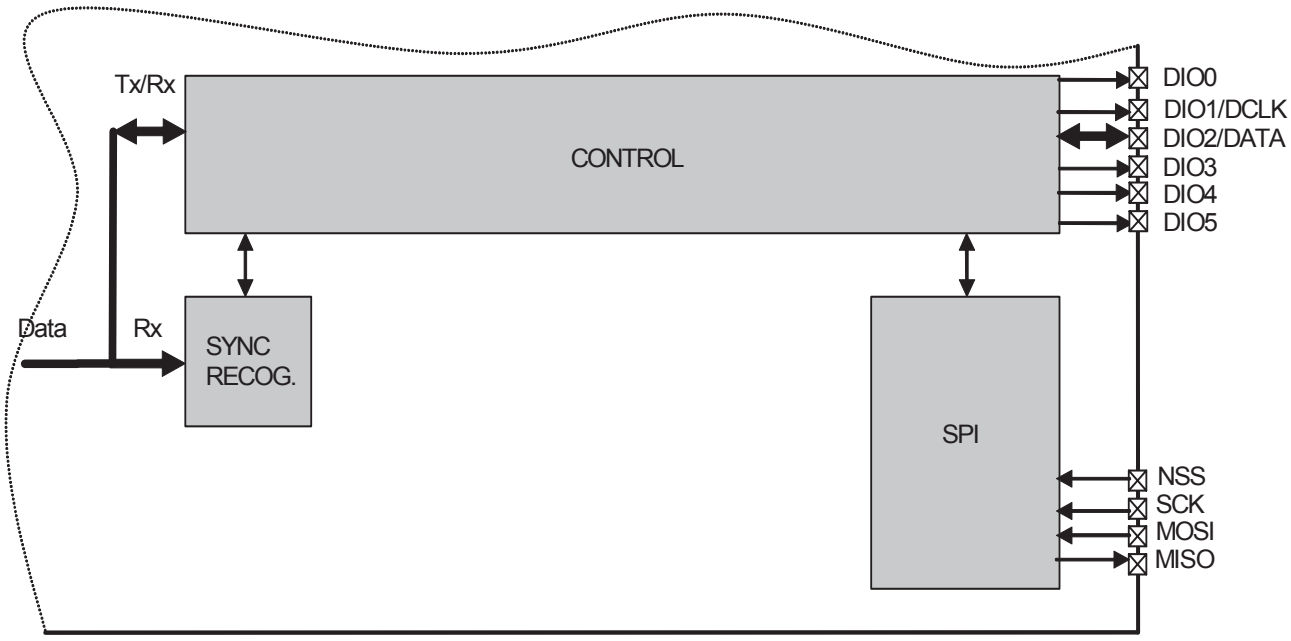


Figure 29. Continuous Mode Conceptual View

4.2.12.2. Tx Processing

In Tx mode, a synchronous data clock for an external uC is provided on DIO1/DCLK pin. Clock timing with respect to the data is illustrated in Figure 30. DATA is internally sampled on the rising edge of DCLK so the uC can change logic state anytime outside the grayed out setup/hold zone.

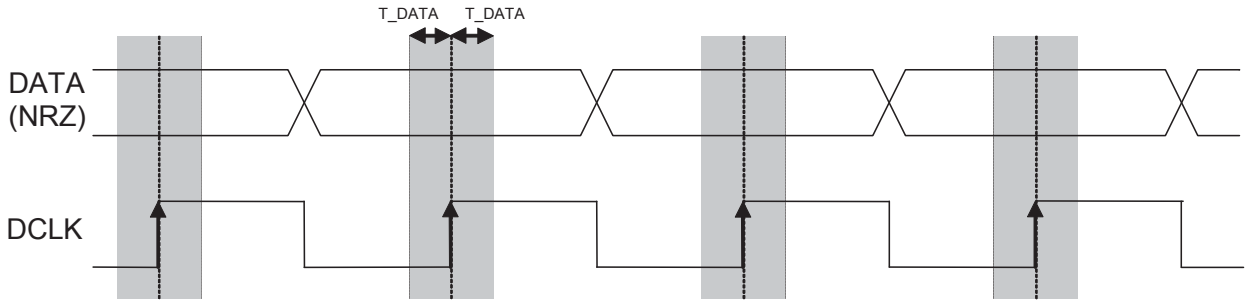


Figure 30. Tx Processing in Continuous Mode

Note the use of DCLK is required when the modulation shaping is enabled.

4.2.12.3. Rx Processing

If the bit synchronizer is disabled, the raw demodulator output is made directly available on DATA pin and no DCLK signal is provided.

Conversely, if the bit synchronizer is enabled, synchronous cleaned data and clock are made available respectively on DIO2/DATA and DIO1/DCLK pins. DATA is sampled on the rising edge of DCLK and updated on the falling edge as illustrated below.

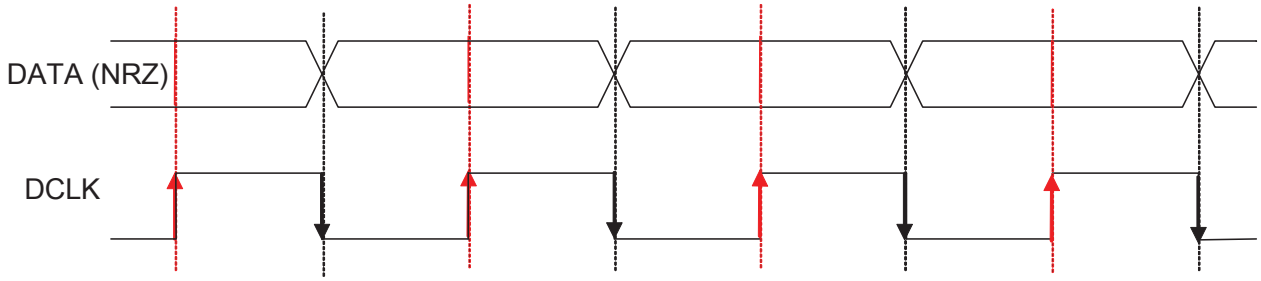


Figure 31. Rx Processing in Continuous Mode

Note In Continuous mode it is always recommended to enable the bit synchronizer to clean the DATA signal even if the DCLK signal is not used by the uC (bit synchronizer is automatically enabled in Packet mode).

4.2.13. Packet Mode

4.2.13.1. General Description

In Packet mode the NRZ data to (from) the (de)modulator is not directly accessed by the uC but stored in the FIFO and accessed via the SPI interface.

In addition, the SX1276/77/78/79 packet handler performs several packet oriented tasks such as Preamble and Sync word generation, CRC calculation/check, whitening/dewhitening of data, Manchester encoding/decoding, address filtering, etc. This simplifies software and reduces uC overhead by performing these repetitive tasks within the RF chip itself.

Another important feature is ability to fill and empty the FIFO in Sleep/Stdby mode, ensuring optimum power consumption and adding more flexibility for the software.

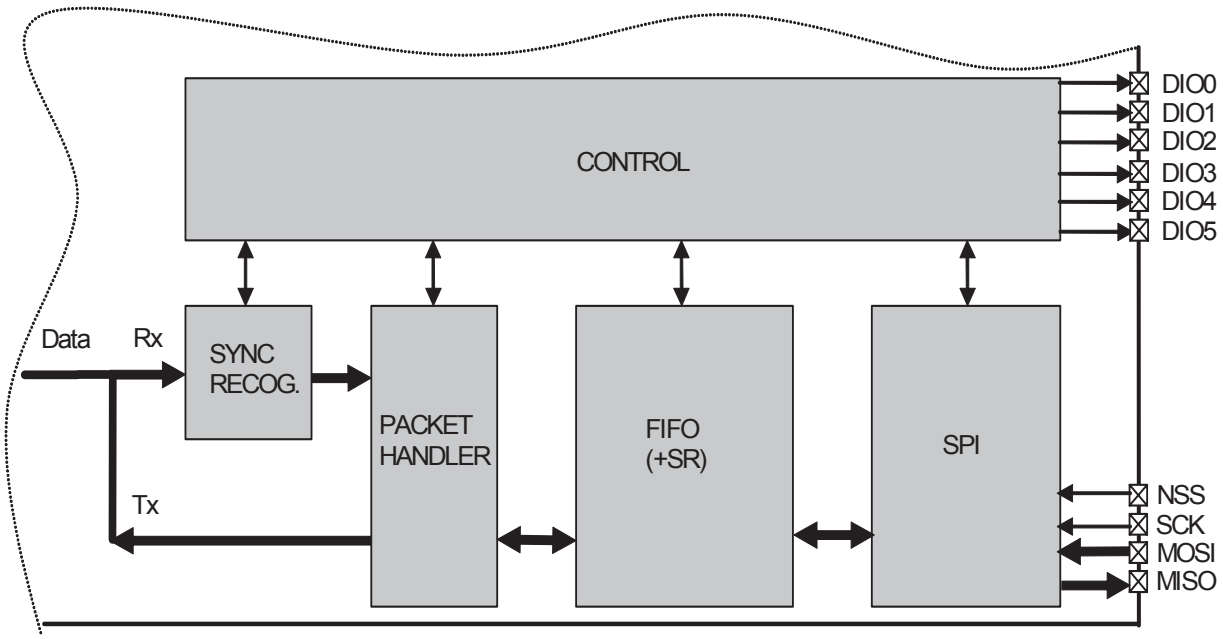


Figure 32. Packet Mode Conceptual View

Note The Bit Synchronizer is automatically enabled in Packet mode.

4.2.13.2. Packet Format

Fixed Length Packet Format

Fixed length packet format is selected when bit *PacketFormat* is set to 0 and *PayloadLength* is set to any value greater than 0.

In applications where the packet length is fixed in advance, this mode of operation may be of interest to minimize RF overhead (no length byte field is required). All nodes, whether Tx only, Rx only, or Tx/Rx should be programmed with the same packet length value.

The length of the payload is limited to 2047 bytes.

The length programmed in *PayloadLength* relates only to the payload which includes the message and the optional address byte. In this mode, the payload must contain at least one byte, i.e. address or message byte.

An illustration of a fixed length packet is shown below. It contains the following fields:

- Preamble (1010...)
- Sync word (Network ID)
- Optional Address byte (Node ID)
- Message data
- Optional 2-bytes CRC checksum

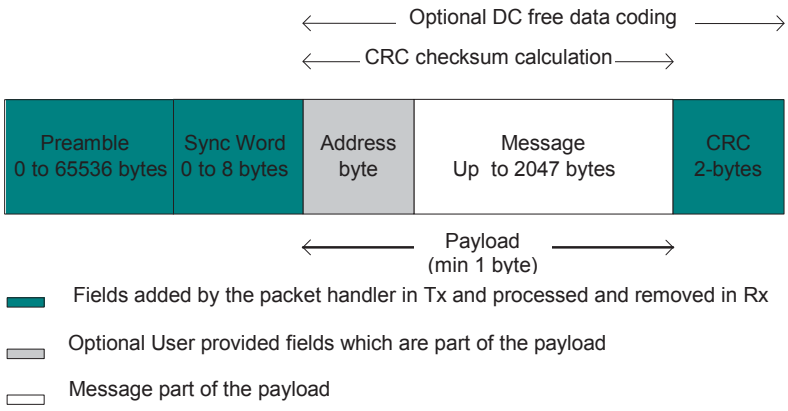


Figure 33. Fixed Length Packet Format

Variable Length Packet Format

Variable length packet format is selected when bit *PacketFormat* is set to 1.

This mode is useful in applications where the length of the packet is not known in advance and can vary over time. It is then necessary for the transmitter to send the length information together with each packet in order for the receiver to operate properly.

In this mode the length of the payload, indicated by the length byte, is given by the first byte of the FIFO and is limited to 255 bytes. Note that the length byte itself is not included in its calculation. In this mode, the payload must contain at least 2 bytes, i.e. length + address or message byte.

An illustration of a variable length packet is shown below. It contains the following fields:

- Preamble (1010...)
- Sync word (Network ID)
- Length byte
- Optional Address byte (Node ID)
- Message data

- Optional 2-bytes CRC checksum

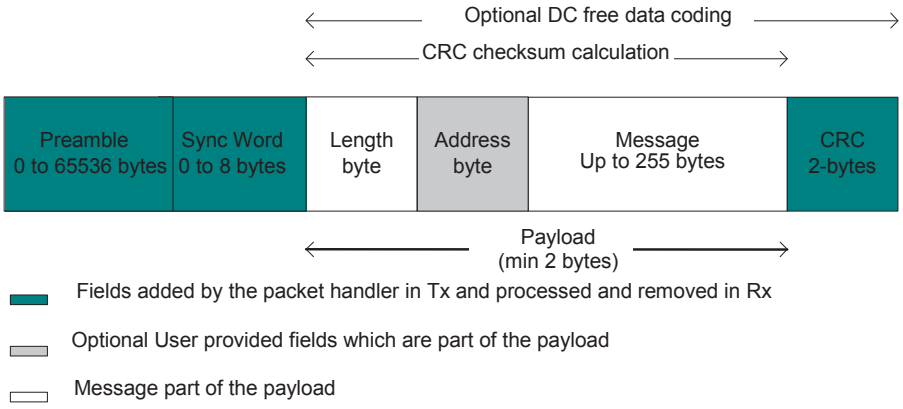


Figure 34. Variable Length Packet Format

Unlimited Length Packet Format

Unlimited length packet format is selected when bit *PacketFormat* is set to 0 and *PayloadLength* is set to 0. The user can then transmit and receive packet of arbitrary length and *PayloadLength* register is not used in Tx/Rx modes for counting the length of the bytes transmitted/received.

In Tx the data is transmitted depending on the *TxStartCondition* bit. On the Rx side the data processing features like Address filtering, Manchester encoding and data whitening are not available if the sync pattern length is set to zero (*SyncOn = 0*). The CRC detection in Rx is also not supported in this mode of the packet handler, however CRC generation in Tx is operational. The interrupts like *CrcOk* & *PayloadReady* are not available either.

An unlimited length packet shown below is made up of the following fields:

- Preamble (1010...).
- Sync word (Network ID).
- Optional Address byte (Node ID).
- Message data
- Optional 2-bytes CRC checksum (Tx only)

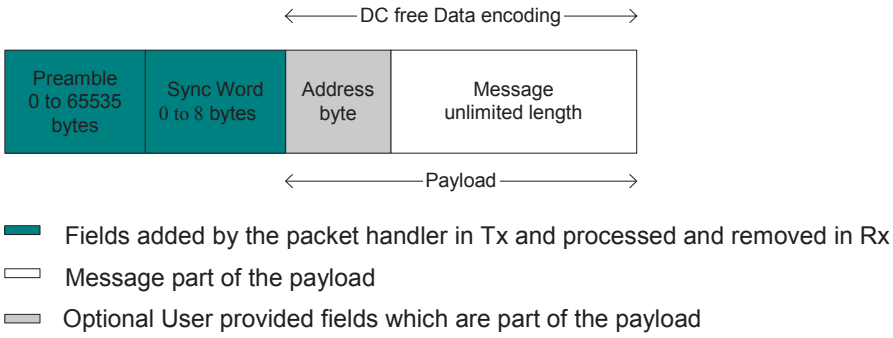


Figure 35. Unlimited Length Packet Format

4.2.13.3. Tx Processing

In Tx mode the packet handler dynamically builds the packet by performing the following operations on the payload available in the FIFO:

- Add a programmable number of preamble bytes
- Add a programmable Sync word
- Optionally calculating CRC over complete payload field (optional length byte + optional address byte + message) and appending the 2 bytes checksum.
- Optional DC-free encoding of the data (Manchester or whitening)

Only the payload (including optional address and length fields) is required to be provided by the user in the FIFO.

The transmission of packet data is initiated by the Packet Handler only if the chip is in Tx mode and the transmission condition defined by *TxStartCondition* is fulfilled. If transmission condition is not fulfilled then the packet handler transmits a preamble sequence until the condition is met. This happens only if the preamble length /= 0, otherwise it transmits a zero or one until the condition is met to transmit the packet data.

The transmission condition itself is defined as:

- if *TxStartCondition* = 1, the packet handler waits until the first byte is written into the FIFO, then it starts sending the preamble followed by the sync word and user payload
- If *TxStartCondition* = 0, the packet handler waits until the number of bytes written in the FIFO is equal to the number defined in *RegFifoThresh* + 1
- If the condition for transmission was already fulfilled i.e. the FIFO was filled in Sleep/Stdby then the transmission of packet starts immediately on enabling Tx

4.2.13.4. Rx Processing

In Rx mode the packet handler extracts the user payload to the FIFO by performing the following operations:

- Receiving the preamble and stripping it off
- Detecting the Sync word and stripping it off
- Optional DC-free decoding of data
- Optionally checking the address byte
- Optionally checking CRC and reflecting the result on *CrcOk.*

Only the payload (including optional address and length fields) is made available in the FIFO.

When the Rx mode is enabled the demodulator receives the preamble followed by the detection of sync word. If fixed length packet format is enabled then the number of bytes received as the payload is given by the *PayloadLength* parameter.

In variable length mode the first byte received after the sync word is interpreted as the length of the received packet. The internal length counter is initialized to this received length. The *PayloadLength* register is set to a value which is greater than the maximum expected length of the received packet. If the received length is greater than the maximum length stored in *PayloadLength* register the packet is discarded otherwise the complete packet is received.

If the address check is enabled then the second byte received in case of variable length and first byte in case of fixed length is the address byte. If the address matches to the one in the *NodeAddress* field, reception of the data continues otherwise it's stopped. The CRC check is performed if *CrcOn* = 1 and the result is available in *CrcOk* indicating that the

CRC was successful. An interrupt (*PayloadReady*) is also generated on DIO0 as soon as the payload is available in the FIFO. The payload available in the FIFO can also be read in Sleep/Standby mode.

If the CRC fails the *PayloadReady* interrupt is not generated and the FIFO is cleared. This function can be overridden by setting *CrcAutoClearOff* = 1, forcing the availability of *PayloadReady* interrupt and the payload in the FIFO even if the CRC fails.

4.2.13.5. Handling Large Packets

When *PayloadLength* exceeds FIFO size (64 bytes) whether in fixed, variable or unlimited length packet format, in addition to *PacketSent* in Tx and *PayloadReady* or *CrcOk* in Rx, the FIFO interrupts/flags can be used as described below:

- For Tx:

FIFO can be prefilled in Sleep/Standby but must be refilled “on-the-fly” during Tx with the rest of the payload.

1) Pre-fill FIFO (in Sleep/Standby first or directly in Tx mode) until *FifoThreshold* or *FifoFull* is set

2) In Tx, wait for *FifoThreshold* or *FifoEmpty* to be set (i.e. FIFO is nearly empty)

3) Write bytes into the FIFO until *FifoThreshold* or *FifoFull* is set.

4) Continue to step 2 until the entire message has been written to the FIFO (*PacketSent* will fire when the last bit of the packet has been sent).

- For Rx:

FIFO must be unfilled “on-the-fly” during Rx to prevent FIFO overrun.

1) Start reading bytes from the FIFO when *FifoEmpty* is cleared or *FifoThreshold* becomes set.

2) Suspend reading from the FIFO if *FifoEmpty* fires before all bytes of the message have been read

3) Continue to step 1 until *PayloadReady* or *CrcOk* fires

4) Read all remaining bytes from the FIFO either in Rx or Sleep/Standby mode

4.2.13.6. Packet Filtering

The SX1276/77/78/79 packet handler offers several mechanisms for packet filtering, ensuring that only useful packets are made available to the uC, reducing significantly system power consumption and software complexity.

Sync Word Based

Sync word filtering/recognition is used for identifying the start of the payload and also for network identification. As previously described, the Sync word recognition block is configured (size, value) in *RegSyncConfig* and *RegSyncValue(i)* registers. This information is used, both for appending Sync word in Tx, and filtering packets in Rx.

Every received packet which does not start with this locally configured Sync word is automatically discarded and no interrupt is generated.

When the Sync word is detected, payload reception automatically starts and *SyncAddressMatch* is asserted.

Note Sync Word values containing 0x00 byte(s) are forbidden

Address Based

Address filtering can be enabled via the *AddressFiltering* bits. It adds another level of filtering, above Sync word (i.e. Sync must match first), typically useful in a multi-node networks where a network ID is shared between all nodes (Sync word) and each node has its own ID (address).

Two address based filtering options are available:

- *AddressFiltering = 01*: Received address field is compared with internal register *NodeAddress*. If they match then the packet is accepted and processed, otherwise it is discarded.
- *AddressFiltering = 10*: Received address field is compared with internal registers *NodeAddress* and *BroadcastAddress*. If either is a match, the received packet is accepted and processed, otherwise it is discarded. This additional check with a constant is useful for implementing broadcast in a multi-node networks

Please note that the received address byte, as part of the payload, is not stripped off the packet and is made available in the FIFO. In addition, *NodeAddress* and *AddressFiltering* only apply to Rx. On Tx side, if address filtering is expected, the address byte should simply be put into the FIFO like any other byte of the payload.

As address filtering requires a Sync word match, both features share the same interrupt flag *SyncAddressMatch*.

Length Based

In variable length Packet mode, *PayloadLength* must be programmed with the maximum payload length permitted. If received length byte is smaller than this maximum then the packet is accepted and processed, otherwise it is discarded.

Please note that the received length byte, as part of the payload, is not stripped off the packet and is made available in the FIFO.

To disable this function the user should set the value of the *PayloadLength* to 2047.

CRC Based

The CRC check is enabled by setting bit *CrcOn* in *RegPacketConfig1*. It is used for checking the integrity of the message.

- On Tx side a two byte CRC checksum is calculated on the payload part of the packet and appended to the end of the message
- On Rx side the checksum is calculated on the received payload and compared with the two checksum bytes received. The result of the comparison is stored in bit *CrcOk.*

By default, if the CRC check fails then the FIFO is automatically cleared and no interrupt is generated. This filtering function can be disabled via *CrcAutoClearOff* bit and in this case, even if CRC fails, the FIFO is not cleared and only *PayloadReady* interrupt goes high. Please note that in both cases, the two CRC checksum bytes are stripped off by the packet handler and only the payload is made available in the FIFO. Two CRC implementations are selected with bit *CrcWhiteningType*.

Table 31 CRC Description

Crc Type	*CrcWhiteningType*	Polynomial	Seed Value	Complemented
CCITT	0 (default)	$X^{16} + X^{12} + X^5 + 1$	0x1D0F	Yes
IBM	1	$X^{16} + X^{15} + X^2 + 1$	0xFFFF	No

A C code implementation of each CRC type is proposed in Application Section 7.

4.2.13.7. DC-Free Data Mechanisms

The payload to be transmitted may contain long sequences of 1's and 0's, which introduces a DC bias in the transmitted signal. The radio signal thus produced has a non uniform power distribution over the occupied channel bandwidth. It also introduces data dependencies in the normal operation of the demodulator. Thus it is useful if the transmitted data is random and DC free.

For such purposes, two techniques are made available in the packet handler: Manchester encoding and data whitening.

Note Only one of the two methods can be enabled at a time.

Manchester Encoding

Manchester encoding/decoding is enabled if *DcFree = 01* and can only be used in Packet mode.

The NRZ data is converted to Manchester code by coding '1' as “10” and '0' as “01”.

In this case, the maximum chip rate is the maximum bit rate given in the specifications and the actual bit rate is half the chip rate.

Manchester encoding and decoding is only applied to the payload and CRC checksum while preamble and Sync word are kept NRZ. However, the chip rate from preamble to CRC is the same and defined by *BitRate* in *RegBitRate* (Chip Rate = Bit Rate NRZ = 2 x Bit Rate Manchester).

Manchester encoding/decoding is thus made transparent for the user, who still provides/retrieves NRZ data to/from the FIFO.

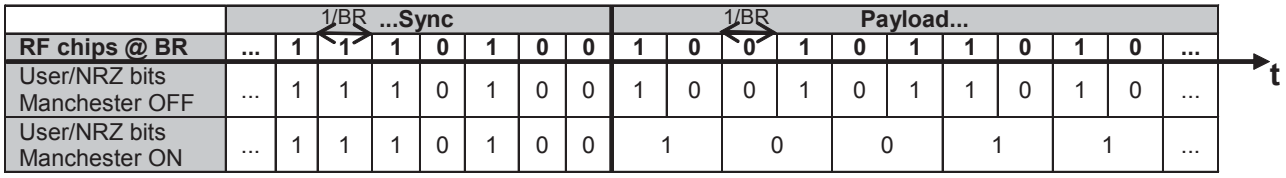


Figure 36. Manchester Encoding/Decoding

Data Whitening

Another technique called whitening or scrambling is widely used for randomizing the user data before radio transmission. The data is whitened using a random sequence on the Tx side and de-whitened on the Rx side using the same sequence. Comparing to Manchester technique it has the advantage of keeping NRZ data rate i.e. actual bit rate is not halved.

The whitening/de-whitening process is enabled if *DcFree = 10*. A 9-bit LFSR is used to generate a random sequence. The payload and 2-byte CRC checksum is then XORed with this random sequence as shown below. The data is de-whitened on the receiver side by XORing with the same random sequence.

Payload whitening/de-whitening is thus made transparent for the user, who still provides/retrieves NRZ data to/from the FIFO.

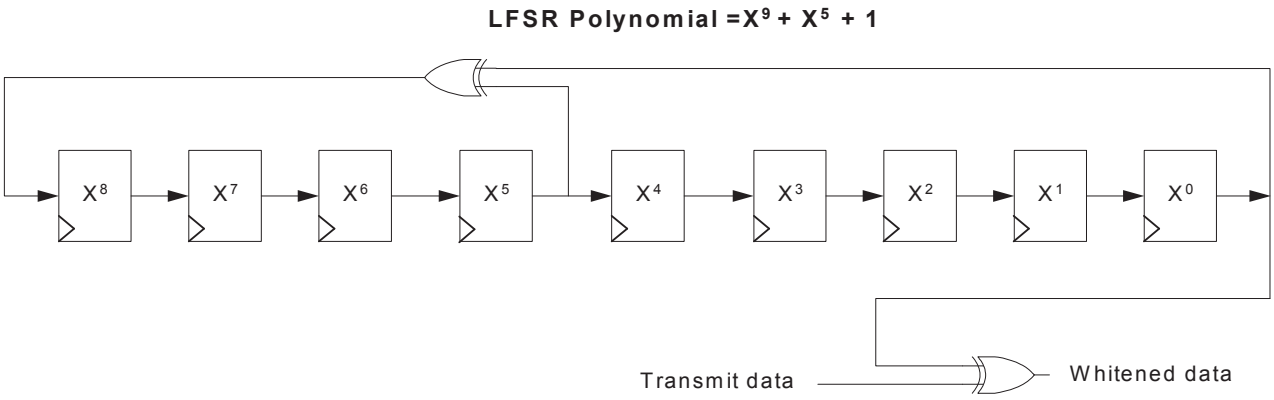


Figure 37. Data Whitening Polynomial

4.2.13.8. Beacon Tx Mode

In some short range wireless network topologies a repetitive message, also known as beacon, is transmitted periodically by a transmitter. The Beacon Tx mode allows for the re-transmission of the same packet without having to fill the FIFO multiple times with the same data.

When *BeaconOn* in *RegPacketConfig2* is set to 1, the FIFO can be filled only once in Sleep or Stdby mode with the required payload. After a first transmission, *FifoEmpty* will go high as usual, but the FIFO content will be restored when the chip exits Transmit mode. *FifoEmpty*, *FifoFull* and *FifoLevel* flags are also restored.

This feature is only available in Fixed packet format, with the Payload Length smaller than the FIFO size. The control of the chip modes (Tx-Sleep-Tx....) can either be undertaken by the microcontroller, or be automated in the Top Sequencer. See example in Section 4.2.13.8.

The Beacon Tx mode is exited by setting *BeaconOn* to 0, and clearing the FIFO by setting *FifoOverrun* to 1.

4.2.14. io-homecontrol® Compatibility Mode

The SX1276/77/78/79 features a io-homecontrol® compatibility mode. Please contact your local Semtech representative for details on its implementation.

4.3. SPI Interface

The SPI interface gives access to the configuration register via a synchronous full-duplex protocol corresponding to CPOL = 0 and CPHA = 0 in Motorola/Freescale nomenclature. Only the slave side is implemented.

Three access modes to the registers are provided:

- SINGLE access: an address byte followed by a data byte is sent for a write access whereas an address byte is sent and a read byte is received for the read access. The NSS pin goes low at the beginning of the frame and goes high after the data byte.
- BURST access: the address byte is followed by several data bytes. The address is automatically incremented internally between each data byte. This mode is available for both read and write accesses. The NSS pin goes low at the beginning of the frame and stay low between each byte. It goes high only after the last byte transfer.
- FIFO access: if the address byte corresponds to the address of the FIFO, then succeeding data byte will address the FIFO. The address is not automatically incremented but is memorized and does not need to be sent between each data byte. The NSS pin goes low at the beginning of the frame and stay low between each byte. It goes high only after the last byte transfer.

The figure below shows a typical SPI single access to a register.

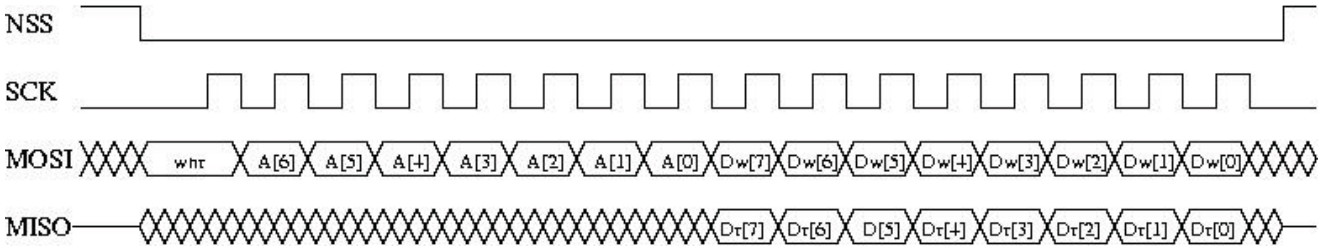


Figure 38. SPI Timing Diagram (single access)

MOSI is generated by the master on the falling edge of SCK and is sampled by the slave (i.e. this SPI interface) on the rising edge of SCK. MISO is generated by the slave on the falling edge of SCK.

A transfer is always started by the NSS pin going low. MISO is high impedance when NSS is high.

The first byte is the address byte. It is comprises:

- A wnr bit, which is 1 for write access and 0 for read access.
- Then 7 bits of address, MSB first.

The second byte is a data byte, either sent on MOSI by the master in case of a write access or received by the master on MISO in case of read access. The data byte is transmitted MSB first.

Proceeding bytes may be sent on MOSI (for write access) or received on MISO (for read access) without a rising NSS edge and re-sending the address. In FIFO mode, if the address was the FIFO address then the bytes will be written / read at the FIFO address. In Burst mode, if the address was not the FIFO address, then it is automatically incremented for each new byte received.

The frame ends when NSS goes high. The next frame must start with an address byte. The SINGLE access mode is therefore a special case of FIFO / BURST mode with only 1 data byte transferred.

During the write access, the byte transferred from the slave to the master on the MISO line is the value of the written register before the write operation.

5. SX1276/77/78/79 Analog & RF Frontend Electronics

5.1. Power Supply Strategy

The SX1276/77/78/79 employs an internal voltage regulation scheme which provides stable operating voltage, and hence device characteristics, over the full industrial temperature and operating voltage range of operation. This includes up to +17 dBm of RF output power which is maintained from 1.8 V to 3.7 V and +20 dBm from 2.4 V to 3.7 V.

The SX1276/77/78/79 can be powered from any low-noise voltage source via pins VBAT_ANA, VBAT_RF and VBAT_DIG. Decoupling capacitors should be connected, as suggested in the reference design of the applications section of this document, on VR_PA, VR_DIG and VR_ANA pins to ensure correct operation of the built-in voltage regulators.

5.2. Low Battery Detector

A low battery detector is also included allowing the generation of an interrupt signal in response to the supply voltage dropping below a programmable threshold that is adjustable through the register *RegLowBat*. The interrupt signal can be mapped to any of the DIO pins by programming *RegDioMapping*.

5.3. Frequency Synthesis

5.3.1. Crystal Oscillator

The crystal oscillator is the main timing reference of the SX1276/77/78/79. It is used as the reference for the PLL's frequency synthesis and as the clock signal for all digital processing.

The crystal oscillator startup time, TS_OSC, depends on the electrical characteristics of the crystal reference used, for more information on the electrical specification of the crystal see section 7.1. The crystal connects to the Pierce oscillator on pins XTA and XTB. The SX1276/77/78/79 optimizes the startup time and automatically triggers the PLL when the oscillator signal is stable.

Optionally, an external clock can be used to replace the crystal oscillator. This typically takes the form of a tight tolerance temperature compensated crystal oscillator (TCXO). When using an external clock source the bit *TcxoInputOn* of register *RegTcxo* should be set to 1 and the external clock has to be provided on XTA (pin 5). XTB (pin 6) should be left open.

The peak-peak amplitude of the input signal must never exceed 1.8 V. Please consult your TCXO supplier for an appropriate value of decoupling capacitor, C_D.

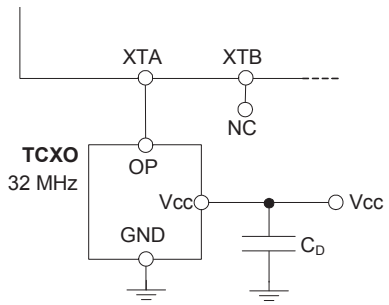


Figure 39. TCXO Connection

5.3.2. CLKOUT Output

The reference frequency, or a fraction of it, can be provided on DIO5 (pin 13) by modifying bits *ClkOut* in *RegDioMapping2*. Two typical applications of the CLKOUT output include:

- To provide a clock output for a companion processor, thus saving the cost of an additional oscillator. CLKOUT can be made available in any operation mode except Sleep mode and is automatically enabled at power on reset.
- To provide an oscillator reference output. Measurement of the CLKOUT signal enables simple software trimming of the initial crystal tolerance.

Note *To minimize the current consumption of the SX1276/77/78/79, please ensure that the CLKOUT signal is disabled when not required.*

5.3.3. PLL

The local oscillator of the SX1276/77/78/79 is derived from two almost identical fractional-N PLLs that are referenced to the crystal oscillator circuit. Both PLLs feature a programmable bandwidth setting where one of four discrete preset bandwidths may be accessed.

The SX1276/77/78/79 PLL uses a 19-bit sigma-delta modulator whose frequency resolution, constant over the whole frequency range, is given by:

$$F_{STEP} = \frac{F_{XOSC}}{2^{19}}$$

The carrier frequency is programmed through *RegFrf*, split across addresses 0x06 to 0x08:

$$F_{RF} = F_{STEP} \times Frf(23,0)$$

Note *The Frf setting is split across 3 bytes. A change in the center frequency will only be taken into account when the least significant byte FrfLsb in RegFrfLsb is written. This allows the potential for user generation of m-ary FSK at very low bit rates. This is possible where frequency modulation is achieved by direct programming of the programmed RF centre frequency. To enable this functionality set the FastHopOn bit of register RegPllHop.*

Three frequency bands are supported, defined as follows:

Table 32 Frequency Bands

Name	Frequency Limits	Products
Band 1 (HF)	862 (*779)-1020 (*960) MHz	SX1276/77/79
Band 2 (LF)	410-525 (*480) MHz	SX1276/77/78/79
Band 3 (LF)	137-175 (*160)MHz	SX1276/77/78/79

** For SX1279*

5.3.4. RC Oscillator

All timing operations in the low-power Sleep state of the Top Level Sequencer rely on the accuracy of the internal low-power RC oscillator. This oscillator is automatically calibrated at the device power-up not requiring any user input.

5.4. Transmitter Description

The transmitter of SX1276/77/78/79 comprises the frequency synthesizer, modulator (both LoRa™ and FSK/OOK) and power amplifier blocks, together with the DC biasing and ramping functionality that is provided through the VR_PA block.

5.4.1. Architecture Description

The architecture of the RF front end is shown in the following diagram:

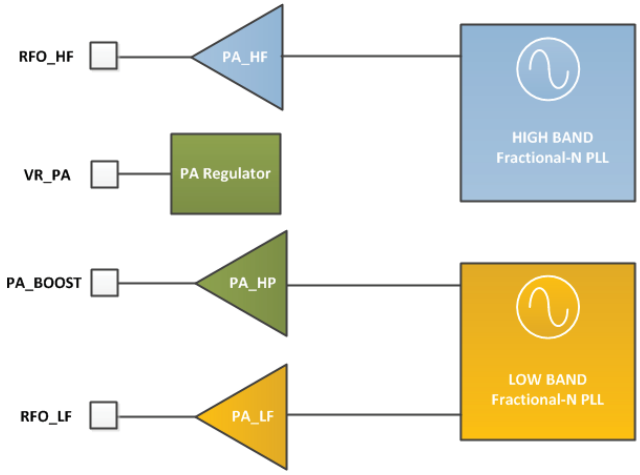


Figure 40. RF Front-end Architecture Shows the Internal PA Configuration.

5.4.2. RF Power Amplifiers

PA_HF and PA_LF are high efficiency amplifiers capable of yielding RF power programmable in 1 dB steps from -4 to +14dBm directly into a 50 ohm load with low current consumption. PA_LF covers the lower bands (up to 525 MHz), whilst PA_HF will cover the upper bands (from 779 MHz). The output power is sensitive to the power supply voltage, and typically their performance is expressed at 3.3V.

PA_HP (High Power), connected to the PA_BOOST pin, covers all frequency bands that the chip addresses. It permits continuous operation at up to +17 dBm and duty cycled operation at up to +20dBm. For full details of operation at +20dBm please consult section 5.4.3

Table 33 Power Amplifier Mode Selection Truth Table

PaSelect	Mode	Power Range	Pout Formula
0	PA_HF or PA_LF on RFO_HF or RFO_LF	-4 to +15dBm	Pout=Pmax-(15-OutputPower) Pmax=10.8+0.6*MaxPower [dBm]
1	PA_HP on PA_BOOST, any frequency	+2 to +17dBm	Pout=17-(15-OutputPower) [dBm]

Notes - For +20 dBm restrictions on operation please consult the following .

- To ensure correct operation at the highest power levels ensure that the current limiter OcpTrim is adjusted to permit delivery of the requisite supply current.

- If the PA_BOOST pin is not used it may be left floating.

5.4.3. High Power +20 dBm Operation

The SX1276/77/78/79 have a high power +20 dBm capability on PA_BOOST pin, with the following settings:

Table 34 High Power Settings

Register	*Address*	*Value for High Power*	*Default value PA_HF/LF or +17dBm*	Description
RegPaDac	0x4d	0x87	0x84	Set Pmax to +20dBm for PA_HP

Notes - High Power settings must be turned off when using PA_LF or PA_HF

- The Over Current Protection limit should be adapted to the actual power level, in RegOcp

Specific Absolute Maximum Ratings and Operating Range restrictions apply to the +20 dBm operation. They are listed in Table 35 and Table 36.

Table 35 Operating Range, +20dBm Operation

Symbol	Description	Min	Max	Unit
DC_20dBm	Duty Cycle of transmission at +20 dBm output	-	1	%
VSWR_20dBm	Maximum VSWR at antenna port, +20 dBm output	-	3:1	-

Table 36 Operating Range, +20dBm Operation

Symbol	Description	Min	Max	Unit
VDDop_20dBm	Supply voltage, +20 dBm output	2.4	3.7	V

The duty cycle of transmission at +20 dBm is limited to 1%, with a maximum VSWR of 3:1 at antenna port, over the standard operating range [-40;+85°C]. For any other operating condition, contact your Semtech representative.

5.4.4. Over Current Protection

The power amplifiers of SX1276/77/78/79 are protected against current over supply in adverse RF load conditions by the over current protection block. This has the added benefit of protecting battery chemistries with limited peak current capability and minimising worst case PA consumption in battery life calculation. The current limiter value is controlled by the *OcpTrim* bits in *RegOcp*, and is calculated according to the following formulae:

Table 37 Trimming of the OCP Current

OcpTrim	I_{MAX}	Imax Formula
0 to 15	45 to 120 mA	45 + 5**OcpTrim* [mA]
16 to 27	130 to 240 mA	-30 + 10**OcpTrim* [mA]
27+	240 mA	*240 mA*

Note Imax sets a limit on the current drain of the Power Amplifier only, hence the maximum current drain of the SX1276/77/78/79 *is equal to Imax + IDDFS*.

5.5. Receiver Description

5.5.1. Overview

The SX1276/77/78/79 features a digital receiver with the analog to digital conversion process being performed directly following the LNA-Mixers block. In addition to the LoRa™ modulation scheme the low-IF receiver is able to demodulate ASK, OOK, (G)FSK and (G)MSK modulation. All filtering, demodulation, gain control, synchronization and packet handling is performed digitally allowing a high degree of programmable flexibility. The receiver also has automatic gain calibration, this improves the precision of RSSI measurement and enhances image rejection.

5.5.2. Receiver Enabled and Receiver Active States

In the receiver operating mode two states of functionality are defined. Upon initial transition to receiver operating mode the receiver is in the ‘receiver-enabled’ state. In this state the receiver awaits for either the user defined valid preamble or RSSI detection criterion to be fulfilled. Once met the receiver enters ‘receiver-active’ state. In this second state the received signal is processed by the packet engine and top level sequencer. For a complete description of the digital functions of the SX1276/77/78/79 receiver please see section 4 of the datasheet.

5.5.3. Automatic Gain Control In FSK/OOK Mode

The AGC feature allows receiver to handle a wide Rx input dynamic range from the sensitivity level up to maximum input level of 0dBm or more, whilst optimizing the system linearity.

The following table shows typical NF and IIP3 performances for the SX1276/77/78/79 LNA gains available.

Table 38 LNA Gain Control and Performances

RX input level (Pin)	*Gain Setting*	*LnaGain*	*Relative LNA Gain [dB]*	NF Band 3/2/1 [dB]	IIP3 Band 3/2/1 [dBm]
Pin <= AgcThresh1	G1	'001'	0 dB	4/5.5/7	-15/-22/-11
AgcThresh1 < Pin <= AgcThresh2	G2	'010'	-6 dB	6.5/8/12	-11/-15/-6
AgcThresh2 < Pin <= AgcThresh3	G3	'011'	-12 dB	11/12/17	-11/-12/0
AgcThresh3 < Pin <= AgcThresh4	G4	'100'	-24 dB	20/21/27	2/3/9
AgcThresh4 < Pin <= AgcThresh5	G5	'110'	-26 dB	32/33/35	10/10/14
AgcThresh5 < Pin	G6	'111'	-48 dB	44/45/43	11/12/14

5.5.4. RSSI in FSK/OOK Mode

The RSSI provides a measure of the incoming signal power at RF input port, measured within the receiver bandwidth. The signal power is available in *RssiValue*. This value is absolute in units of dBm and with a resolution of 0.5 dB. The formula below relates the register value to the absolute input signal level at the RF input port:

$$RssiValue = -2 \cdot RF\ level\left[dBm\right] + RssiOffset\ \left[dB\right]$$

The RSSI value can be compensated to take into account the loss in the matching network or even the gain of an additional LNA by using *RssiOffset*. The offset can be chosen in 1 dB steps from -16 to +15 dB. When compensation is applied, the effective signal strength is read as follows:

$$RSSI\left[dBm\right] = -\frac{RssiValue}{2}$$

The RSSI value is smoothed on a user defined number of measured RSSI samples. The precision of the RSSI value is related to the number of RSSI samples used. *RssiSmoothing* selects the number of RSSI samples from a minimum of 2 samples up to 256 samples in increments of power of 2. Table 39 gives the estimation of the RSSI accuracy for a 10 dB SNR and response time versus the number of RSSI samples programmed in *RssiSmoothing*.

Table 39 RssiSmoothing Options

RssiSmoothing	Number of Samples	Estimated Accuracy	Response Time
'000'	2	± 6 dB	$\frac{2^{(RssiSmoothing+1)}}{4 \cdot RxBw\left[kHz\right]}\left[ms\right]$
'001'	4	± 5 dB	
'010'	8	± 4 dB	
'011'	16	± 3 dB	
'100'	32	± 2 dB	
'101'	64	± 1.5 dB	
'110'	128	± 1.2 dB	
'111'	256	± 1.1 dB	

The RSSI is calibrated when the image and RSSI calibration process is launched.

5.5.5. RSSI and SNR in LoRa™ Mode

The RSSI values reported by the LoRa™ modem differ from those expressed by the FSK/OOK modem. The following formula shows the method used to interpret the LoRa™ RSSI values:

RSSI (dBm) = -157 + *Rssi*, (when using the High Frequency (HF) port)

or

RSSI (dBm) = -164 + *Rssi*, (when using the Low Frequency (LF) port)

The same formula can be re-used to evaluate the signal strength of the received packet:

Packet Strength (dBm) = -157 + *Rssi*, (when using the High Frequency (HF) port)

or

Packet Strength (dBm) = -164 + *Rssi*, (when using the Low Frequency (LF) port)

Due to the nature of the LoRa modulation, it is possible to receive packets below the noise floor. In this situation, the SNR is used in conjunction of the PacketRssi to compute the signal strength of the received packet:

Packet Strength (dBm) = -157 + *PacketRssi* + *PacketSnr* * 0.25 (when using the HF port and SNR < 0)

or

Packet Strength (dBm) = -164 + *PacketRssi* + *PacketSnr* * 0.25 (when using the LF port and SNR < 0)

Note:

1. *PacketRssi* (in RegPktRssiValue), is an averaged version of *Rssi* (in RegRssiValue). *Rssi* can be read at any time (during packet reception or not), and should be averaged to give more precise results.

2. The constants, -157 and -164, may vary with the front-end setup of the SX1276/77/78/79 (*LnaBoost* =1 or 0, presence of an external LNA, mismatch at the LNA input…). It is recommended to adjust these values with a single-point calibration procedure to increase RSSI accuracy.

3. As signal strength increases (RSSI>-100dBm), the linearity of PacketRssi is not guaranteed and results will diverge from the ideal 1dB/dB ideal curve. When very good RSSI precision is required over the whole dynamic range of the receiver, two options are proposed:

- *Rssi* in RegRssiValue offers better linearity. *Rssi* can be sampled during the reception of the payload (between ValidHeader and RxDone IRQ), and used to extract a more high-signal RSSI measurement
- When SNR>=0, the standard formula can be adjusted to correct the slope:

RSSI = -157+16/15 * PacketRssi (or RSSI = -164+16/15 * PacketRssi)

5.5.6. Channel Filter

The role of the channel filter is to reject noise and interference outside of the wanted channel. The SX1276/77/78/79 channel filtering is implemented with a 16-tap finite impulse response (FIR) filter. Rejection of the filter is high enough that the filter stop-band performance is not the dominant influence on adjacent channel rejection performance. This is instead limited by the SX1276/77/78/79 local oscillator phase noise.

Note *To respect sampling criterion in the decimation chain of the receiver, the communication bit rate cannot be set at a higher than twice the single side receiver bandwidth (BitRate < 2 x RxBw)*

The single-side channel filter bandwidth *RxBw* is controlled by the parameters *RxBwMant* and *RxBwExp* in *RegRxBw:*

$$RxBw = \frac{FXOSC}{RxBwMant \times 2^{RxBwExp+2}}$$

The following channel filter bandwidths are hence accessible in the case of a 32 MHz reference oscillator:

Table 40 Available RxBw Settings

RxBwMant (binary/value)	*RxBwExp* (decimal)	*RxBw (kHz)* FSK/OOK
10b / 24	7	2.6
01b / 20	7	3.1
00b / 16	7	3.9
10b / 24	6	5.2
01b / 20	6	6.3
00b / 16	6	7.8
10b / 24	5	10.4
01b / 20	5	12.5
00b / 16	5	15.6
10b / 24	4	20.8
01b / 20	4	25.0
00b / 16	4	31.3
10b / 24	3	41.7
01b / 20	3	50.0
00b / 16	3	62.5
10b / 24	2	83.3
01b / 20	2	100.0
00b / 16	2	125.0
10b / 24	1	166.7
01b / 20	1	200.0
00b / 16	1	250.0
Other settings		reserved

5.5.7. Temperature Measurement

A stand alone temperature measurement block is used in order to measure the temperature in any mode except Sleep and Standby. It is enabled by default, and can be stopped by setting *TempMonitorOff* to 1. The result of the measurement is stored in *TempValue* in *RegTemp*.

Due to process variations, the absolute accuracy of the result is +/- 10 °C. Higher precision requires a calibration procedure at a known temperature. The figure below shows the influence of just such a calibration process. For more information, including source code, please consult the applications section of this document.

Example temperature curve, typical device

	Correction Factor		15
Actual Temp [Celsius]	RegTemp [Dec]	Temp before calibration [°C]	Temp after calibration [°C]
85	181	74	89
75	190	65	80
65	201	54	69
55	211	44	59
45	222	33	48
35	232	23	38
25	245	10	25
15	0	0	15
5	10	-10	5
-5	21	-21	-6
-15	33	-33	-18
-25	44	-44	-29
-35	56	-56	-41
-40	63	-63	-48

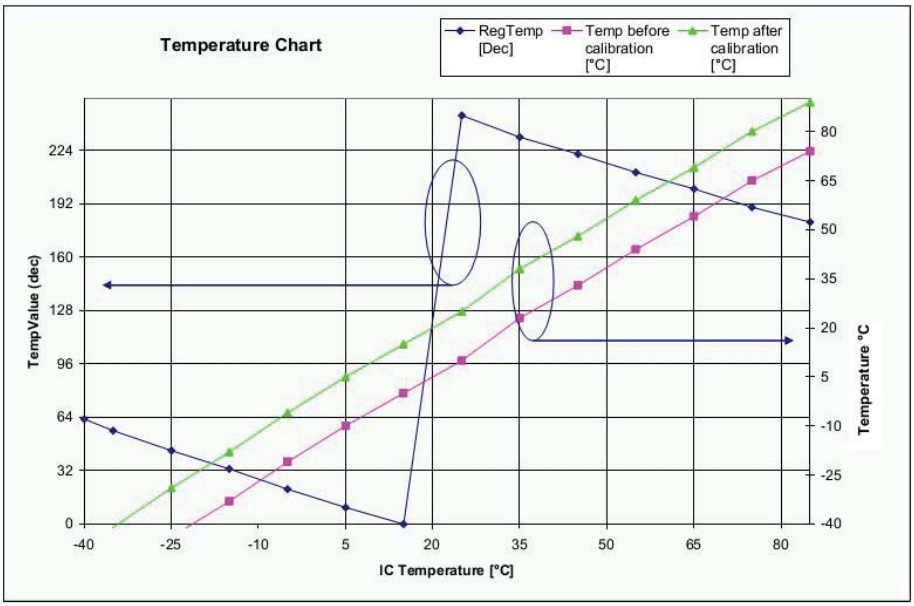


Figure 41. Temperature Sensor Response

When using the temperature sensor in the application, the following sequence should be followed:

- Set the device to Standby and wait for oscillator startup
- Set the device to FSRx mode
- Set *TempMonitorOff* = 0 (enables the sensor). It is not required to wait for the PLL Lock indication
- Wait for 140 microseconds
- Set *TempMonitorOff* = 1
- Set device back to Sleep of Standby mode
- Access temperature value in *RegTemp*

6. Description of the Registers

The register mapping depends upon whether FSK/OOK or LoRa™ mode has been selected. The following table summarises the location and function of each register and gives an overview of the changes in register mapping between both modes of operation.

6.1. Register Table Summary

Table 41 Registers Summary

Address	Register Name		Reset (POR)	Default (FSK)	Description	
	FSK/OOK Mode	LoRa™ Mode			FSK Mode	LoRa™ Mode
0x00	RegFifo		0x00		FIFO read/write access	
0x01	RegOpMode		0x01		Operating mode & LoRa™ / FSK selection	
0x02	RegBitrateMsb	Unused	0x1A		Bit Rate setting, Most Significant Bits	
0x03	RegBitrateLsb		0x0B		Bit Rate setting, Least Significant Bits	
0x04	RegFdevMsb		0x00		Frequency Deviation setting, Most Significant Bits	
0x05	RegFdevLsb		0x52		Frequency Deviation setting, Least Significant Bits	
0x06	RegFrfMsb		0x6C		RF Carrier Frequency, Most Significant Bits	
0x07	RegFrfMid		0x80		RF Carrier Frequency, Intermediate Bits	
0x08	RegFrfLsb		0x00		RF Carrier Frequency, Least Significant Bits	
0x09	RegPaConfig		0x4F		PA selection and Output Power control	
0x0A	RegPaRamp		0x09		Control of PA ramp time, low phase noise PLL	
0x0B	RegOcp		0x2B		Over Current Protection control	
0x0C	RegLna		0x20		LNA settings	
0x0D	RegRxConfig	RegFifoAddrPtr	0x08	0x0E	AFC, AGC, ctrl	FIFO SPI pointer
0x0E	RegRssiConfig	RegFifoTxBaseAddr	0x02		RSSI	Start Tx data
0x0F	RegRssiCollision	RegFifoRxBaseAddr	0x0A		RSSI Collision detector	Start Rx data
0x10	RegRssiThresh	FifoRxCurrentAddr	0xFF		RSSI Threshold control	Start address of last packet received
0x11	RegRssiValue	RegIrqFlagsMask	n/a	n/a	RSSI value in dBm	Optional IRQ flag mask
0x12	RegRxBw	RegIrqFlags	0x15		Channel Filter BW Control	IRQ flags
0x13	RegAfcBw	RegRxNbBytes	0x0B		AFC Channel Filter BW	Number of received bytes
0x14	RegOokPeak	RegRxHeaderCntValueMsb	0x28		OOK demodulator	Number of valid headers received
0x15	RegOokFix	RegRxHeaderCntValueLsb	0x0C		Threshold of the OOK demod	
0x16	RegOokAvg	RegRxPacketCntValueMsb	0x12		Average of the OOK demod	Number of valid packets received
0x17	Reserved17	RegRxPacketCntValueLsb	0x47		-	
0x18	Reserved18	RegModemStat	0x32		-	Live LoRaTM modem status
0x19	Reserved19	RegPktSnrValue	0x3E		-	Espimation of last packet SNR
0x1A	RegAfcFei	RegPktRssiValue	0x00		AFC and FEI control	RSSI of last packet

Address	Register Name		Reset (POR)	Default (FSK)	Description	
	FSK/OOK Mode	LoRa™ Mode			FSK Mode	LoRa™ Mode
0x1B	RegAfcMsb	RegRssiValue	0x00	n/a	Frequency correction value of the AFC	Current RSSI
0x1C	RegAfcLsb	RegHopChannel	0x00	n/a		FHSS start channel
0x1D	RegFeiMsb	RegModemConfig 1	0x00	n/a	Value of the calculated frequency error	Modem PHY config 1
0x1E	RegFeiLsb	RegModemConfig 2	0x00	n/a		Modem PHY config 2
0x1F	RegPreambleDetect	RegSymbTimeoutLsb	0x40	0xAA	Settings of the Preamble Detector	Receiver timeout value
0x20	RegRxTimeout1	RegPreambleMsb	0x00		Timeout Rx request and RSSI	Size of preamble
0x21	RegRxTimeout2	RegPreambleLsb	0x00		Timeout RSSI and *PayloadReady*	
0x22	RegRxTimeout3	RegPayloadLength	0x00		Timeout RSSI and *SyncAddress*	LoRaTM payload length
0x23	RegRxDelay	RegMaxPayloadLength	0x00		Delay between Rx cycles	LoRaTM maximum payload length
0x24	RegOsc	RegHopPeriod	0x05	0x07	RC Oscillators Settings, CLKOUT frequency	FHSS Hop period
0x25	RegPreambleMsb	RegFifoRxByteAddr	0x00		Preamble length, MSB	Address of last byte written in FIFO
0x26	RegPreambleLsb	RegModemConfig3	0x03		Preamble length, LSB	Modem PHY config 3
0x27	RegSyncConfig	RESERVED	0x93		Sync Word Recognition control	RESERVED
0x28	RegSyncValue1	RegFeiMsb	0x55	0x01	Sync Word bytes 1	Estimated frequency error
0x29	RegSyncValue2	RegFeiMid	0x55	0x01	Sync Word bytes 2	
0x2A	RegSyncValue3	RegFeiLsb	0x55	0x01	Sync Word bytes 3	
0x2B	RegSyncValue4	RESERVED	0x55	0x01	Sync Word bytes 4	RESERVED
0x2C	RegSyncValue5	RegRssiWideband	0x55	0x01	Sync Word bytes 5	Wideband RSSI measurement
0x2D-0x2F	RegSyncValue6-8	RESERVED	0x55	0x01	Sync Word bytes, 6 to 8	RESERVED
0x30	RegPacketConfig1	RESERVED	0x90		Packet mode settings	
0x31	RegPacketConfig2	RegDetectOptimize	0x40		Packet mode settings	LoRa detection Optimize for SF6
0x32	RegPayloadLength	RESERVED	0x40		Payload length setting	RESERVED
0x33	RegNodeAdrs	RegInvertIQ	0x00		Node address	Invert LoRa I and Q signals
0x34	RegBroadcastAdrs	RESERVED	0x00		Broadcast address	RESERVED
0x35	RegFifoThresh		0x0F	0x1F	Fifo threshold, Tx start condition	
0x36	RegSeqConfig1		0x00		Top level Sequencer settings	
0x37	RegSeqConfig2	RegDetectionThreshold	0x00		Top level Sequencer settings	LoRa detection threshold for SF6
0x38	RegTimerResol	RESERVED	0x00		Timer 1 and 2 resolution control	RESERVED
0x39	RegTimer1Coef	RegSyncWord	0xF5	0x12	Timer 1 setting	LoRa Sync Word

Address	Register Name		Reset (POR)	Default (FSK)	Description	
	FSK/OOK Mode	LoRa™ Mode			FSK Mode	LoRa™ Mode
0x3A	RegTimer2Coef	RESERVED	0x20		Timer 2 setting	RESERVED
0x3B	RegImageCal		0x82	0x02	Image calibration engine control	
0x3C	RegTemp		-		Temperature Sensor value	
0x3D	RegLowBat		0x02		Low Battery Indicator Settings	
0x3E	RegIrqFlags1		0x80		Status register: PLL Lock state, Timeout, RSSI	
0x3F	RegIrqFlags2		0x40		Status register: FIFO handling flags, Low Battery	
0x40	RegDioMapping1		0x00		Mapping of pins DIO0 to DIO3	
0x41	RegDioMapping2		0x00		Mapping of pins DIO4 and DIO5, ClkOut frequency	
0x42	RegVersion		0x12		Semtech ID relating the silicon revision	
0x44	RegPllHop	Unused	0x2D		Control the fast frequency hopping mode	Unused
0x4B	RegTcxo		0x09		TCXO or XTAL input setting	
0x4D	RegPaDac		0x84		Higher power settings of the PA	
0x5B	RegFormerTemp		-		Stored temperature during the former IQ Calibration	
0x5D	RegBitRateFrac	Unused	0x00		Fractional part in the Bit Rate division ratio	Unused
0x61	RegAgcRef		0x13		Adjustment of the AGC thresholds	
0x62	RegAgcThresh1		0x0E			
0x63	RegAgcThresh2		0x5B			
0x64	RegAgcThresh3		0xDB			
0x70	RegPll		0xD0		Control of the PLL bandwidth	
others	RegTest		-		Internal test registers. Do not overwrite	

Note - Reset values are automatically refreshed in the chip at Power On Reset

- Default values are the Semtech recommended register values, optimizing the device operation

*- Registers for which the Default value differs from the Reset value are denoted by a * in the tables of section 6.2*

6.2. FSK/OOK Mode Register Map

This section details the SX1276/77/78/79 register mapping and the precise contents of each register in FSK/OOK mode.

Convention: r: read, w: write, t:trigger, c: clear

Table 42 Register Map

Name (Address)	Bits	Variable Name	Mode	Default value	FSK/OOK Description
RegFifo (0x00)	7-0	Fifo	rw	0x00	FIFO data input/output
Registers for Common settings					
RegOpMode (0x01)	7	LongRangeMode	r	0x00	0 → FSK/OOK Mode 1→ LoRa™ Mode This bit can be modified only in Sleep mode. A write operation on other device modes is ignored.
	6-5	ModulationType	rw	0x00	Modulation scheme: 00 → FSK 01 → OOK 10 → 11 → reserved
	4	reserved	r	0x0	reserved
	3	LowFrequencyModeOn	rw	0x01	Access Low Frequency Mode registers (from address 0x61 on) 0 → High Frequency Mode (access to HF test registers) 1 → Low Frequency Mode (access to LF test registers)
	2-0	Mode	rw	0x01	Transceiver modes 000 → Sleep mode 001 → Stdby mode 010 → FS mode TX (FSTx) 011 → Transmitter mode (Tx) 100 → FS mode RX (FSRx) 101 → Receiver mode (Rx) 110 → reserved 111 → reserved
RegBitrateMsb (0x02)	7-0	BitRate(15:8)	rw	0x1a	MSB of Bit Rate (chip rate if Manchester encoding is enabled)
RegBitrateLsb (0x03)	7-0	BitRate(7:0)	rw	0x0b	LSB of bit rate (chip rate if Manchester encoding is enabled) $BitRate = \frac{FXOSC}{BitRate(15,0) + \frac{BitrateFrac}{16}}$ Default value: 4.8 kb/s
RegFdevMsb (0x04)	7-6	reserved	rw	0x00	reserved
	5-0	Fdev(13:8)	rw	0x00	MSB of the frequency deviation
RegFdevLsb (0x05)	7-0	Fdev(7:0)	rw	0x52	LSB of the frequency deviation $Fdev = Fstep \times Fdev(15,0)$ Default value: 5 kHz

Name (Address)	Bits	Variable Name	Mode	Default value	FSK/OOK Description
RegFrfMsb (0x06)	7-0	Frf(23:16)	rw	0x6c	MSB of the RF carrier frequency
RegFrfMid (0x07)	7-0	Frf(15:8)	rw	0x80	MSB of the RF carrier frequency
RegFrfLsb (0x08)	7-0	Frf(7:0)	rw	0x00	LSB of RF carrier frequency $Frf = Fstep \times Frf(23;0)$ Default value: 434.000 MHz The RF frequency is taken into account internally only when: - entering FSRX/FSTX modes - re-starting the receiver
Registers for the Transmitter					
RegPaConfig (0x09)	7	PaSelect	rw	0x00	Selects PA output pin 0 → RFO pin. Maximum power of +14 dBm 1 → PA_BOOST pin. Maximum power of +20 dBm
	6-4	MaxPower	rw	0x04	Select max output power: Pmax=10.8+0.6*MaxPower [dBm]
	3-0	OutputPower	rw	0x0f	Pout=Pmax-(15-OutputPower) if PaSelect = 0 (RFO pins) Pout=17-(15-OutputPower) if PaSelect = 1 (PA_BOOST pin)
RegPaRamp (0x0A)	7	unused	r	0x00	unused
	6-5	ModulationShaping	rw	0x00	Data shaping: In FSK: 00 → no shaping 01 → Gaussian filter BT = 1.0 10 → Gaussian filter BT = 0.5 11 → Gaussian filter BT = 0.3 In OOK: 00 → no shaping 01 → filtering with fcutoff = bit_rate 10 → filtering with fcutoff = 2*bit_rate (for bit_rate < 125 kb/s) 11 → reserved
	4	reserved	rw	0x00	reserved
	3-0	PaRamp	rw	0x09	Rise/Fall time of ramp up/down in FSK 0000 → 3.4 ms 0001 → 2 ms 0010 → 1 ms 0011 → 500 us 0100 → 250 us 0101 → 125 us 0110 → 100 us 0111 → 62 us 1000 → 50 us 1001 → 40 us (d) 1010 → 31 us 1011 → 25 us 1100 → 20 us 1101 → 15 us 1110 → 12 us 1111 → 10 us

Name (Address)	Bits	Variable Name	Mode	Default value	FSK/OOK Description
RegOcp (0x0B)	7-6	unused	r	0x00	unused
	5	OcpOn	rw	0x01	Enables overload current protection (OCP) for the PA: 0 → OCP disabled 1 → OCP enabled
	4-0	OcpTrim	rw	0x0b	Trimming of OCP current: I_{max} = 45+5*OcpTrim [mA] if OcpTrim <= 15 (120 mA) / I_{max} = -30+10*OcpTrim [mA] if 15 < OcpTrim <= 27 (130 to 240 mA) I_{max} = 240mA for higher settings Default I_{max} = 100mA
Registers for the Receiver					
RegLna (0x0C)	7-5	LnaGain	rw	0x01	LNA gain setting: 000 → reserved 001 → G1 = highest gain 010 → G2 = highest gain – 6 dB 011 → G3 = highest gain – 12 dB 100 → G4 = highest gain – 24 dB 101 → G5 = highest gain – 36 dB 110 → G6 = highest gain – 48 dB 111 → reserved Note: Reading this address always returns the current LNA gain (which may be different from what had been previously selected if AGC is enabled.
	4-3	LnaBoostLf	rw	0x00	Low Frequency (RFI_LF) LNA current adjustment 00 → Default LNA current Other → Reserved
	2	reserved	rw	0x00	reserved
	1-0	LnaBoostHf	rw	0x00	High Frequency (RFI_HF) LNA current adjustment 00 → Default LNA current 11 → Boost on, 150% LNA current

Name (Address)	Bits	Variable Name	Mode	Default value	FSK/OOK Description
RegRxConfig (0x0d)	7	RestartRxOnCollision	rw	0x00	Turns on the mechanism restarting the receiver automatically if it gets saturated or a packet collision is detected 0 → No automatic Restart 1 → Automatic restart On
	6	RestartRxWithoutPllLock	wt	0x00	Triggers a manual Restart of the Receiver chain when set to 1. Use this bit when there is no frequency change, RestartRxWithPllLock otherwise.
	5	RestartRxWithPllLock	wt	0x00	Triggers a manual Restart of the Receiver chain when set to 1. Use this bit when there is a frequency change, requiring some time for the PLL to re-lock.
	4	AfcAutoOn	rw	0x00	0 → No AFC performed at receiver startup 1 → AFC is performed at each receiver startup
	3	AgcAutoOn	rw	0x01	0 → LNA gain forced by the LnaGain Setting 1 → LNA gain is controlled by the AGC
	2-0	RxTrigger	rw	0x06 *	Selects the event triggering AGC and/or AFC at receiver startup. See Table 24 for a description.
RegRssiConfig (0x0e)	7-3	RssiOffset	rw	0x00	Signed RSSI offset, to compensate for the possible losses/gains in the front-end (LNA, SAW filter...) 1dB / LSB, 2's complement format
	2-0	RssiSmoothing	rw	0x02	Defines the number of samples taken to average the RSSI result: 000 → 2 samples used 001 → 4 samples used 010 → 8 samples used 011 → 16 samples used 100 → 32 samples used 101 → 64 samples used 110 → 128 samples used 111 → 256 samples used
RegRssiCollision (0x0f)	7-0	RssiCollisionThreshold	rw	0x0a	Sets the threshold used to consider that an interferer is detected, witnessing a packet collision. 1dB/LSB (only RSSI increase) Default: 10dB
RegRssiThresh (0x10)	7-0	RssiThreshold	rw	0xff	RSSI trigger level for the Rssi interrupt: - RssiThreshold / 2 [dBm]
RegRssiValue (0x11)	7-0	RssiValue	r	-	Absolute value of the RSSI in dBm, 0.5dB steps. RSSI = - RssiValue/2 [dBm]
RegRxBw (0x12)	7	unused	r	-	unused
	6-5	reserved	rw	0x00	reserved
	4-3	RxBwMant	rw	0x02	Channel filter bandwidth control: 00 → RxBwMant = 16 10 → RxBwMant = 24 01 → RxBwMant = 20 11 → reserved
	2-0	RxBwExp	rw	0x05	Channel filter bandwidth control
RegAfcBw (0x13)	7-5	reserved	rw	0x00	reserved
	4-3	RxBwMantAfc	rw	0x01	RxBwMant parameter used during the AFC
	2-0	RxBwExpAfc	rw	0x03	RxBwExp parameter used during the AFC

Name (Address)	Bits	Variable Name	Mode	Default value	FSK/OOK Description
RegOokPeak (0x14)	7-6	reserved	rw	0x00	reserved
	5	BitSyncOn	rw	0x01	Enables the Bit Synchronizer. 0 → Bit Sync disabled (not possible in Packet mode) 1 → Bit Sync enabled
	4-3	OokThreshType	rw	0x01	Selects the type of threshold in the OOK data slicer: 00 → fixed threshold 10 → average mode 01 → peak mode (default) 11 → reserved
	2-0	OokPeakTheshStep	rw	0x00	Size of each decrement of the RSSI threshold in the OOK demodulator: 000 → 0.5 dB 001 → 1.0 dB 010 → 1.5 dB 011 → 2.0 dB 100 → 3.0 dB 101 → 4.0 dB 110 → 5.0 dB 111 → 6.0 dB
RegOokFix (0x15)	7-0	OokFixedThreshold	rw	0x0C	Fixed threshold for the Data Slicer in OOK mode Floor threshold for the Data Slicer in OOK when Peak mode is used
RegOokAvg (0x16)	7-5	OokPeakThreshDec	rw	0x00	Period of decrement of the RSSI threshold in the OOK demodulator: 000 → once per chip 001 → once every 2 chips 010 → once every 4 chips 011 → once every 8 chips 100 → twice in each chip 101 → 4 times in each chip 110 → 8 times in each chip 111 → 16 times in each chip
	4	reserved	rw	0x01	reserved
	3-2	OokAverageOffset	rw	0x00	Static offset added to the threshold in average mode in order to reduce glitching activity (OOK only): 00 → 0.0 dB 10 → 4.0 dB 01 → 2.0 dB 11 → 6.0 dB
	1-0	OokAverageThreshFilt	rw	0x02	Filter coefficients in average mode of the OOK demodulator: 00 → $f_C \approx$ chip rate / 32.π 01 → $f_C \approx$ chip rate / 8.π 10 → $f_C \approx$ chip rate / 4.π 11 → $f_C \approx$ chip rate / 2.π
RegRes17 to RegRes19	7-0	reserved	rw	0x47 0x32 0x3E	reserved. Keep the Reset values.
RegAfcFei (0x1a)	7-5	unused	r	-	unused
	4	AgcStart	wt	0x00	Triggers an AGC sequence when set to 1.
	3	reserved	rw	0x00	reserved
	2	unused	-	-	unused
	1	AfcClear	wc	0x00	Clear AFC register set in Rx mode. Always reads 0.
	0	AfcAutoClearOn	rw	0x00	Only valid if AfcAutoOn is set 0 → AFC register is not cleared at the beginning of the automatic AFC phase 1 → AFC register is cleared at the beginning of the automatic AFC phase

Name (Address)	Bits	Variable Name	Mode	Default value	FSK/OOK Description
RegAfcMsb (0x1b)	7-0	AfcValue(15:8)	rw	0x00	MSB of the AfcValue, 2's complement format. Can be used to overwrite the current AFC value
RegAfcLsb (0x1c)	7-0	AfcValue(7:0)	rw	0x00	LSB of the AfcValue, 2's complement format. Can be used to overwrite the current AFC value
RegFeiMsb (0x1d)	7-0	FeiValue(15:8)	rw	-	MSB of the measured frequency offset, 2's complement. Must be read before RegFeiLsb.
RegFeiLsb (0x1e)	7-0	FeiValue(7:0)	rw	-	LSB of the measured frequency offset, 2's complement *Frequency error* = FeiValue x Fstep
RegPreambleDetect (0x1f)	7	PreambleDetectorOn	rw	0x01 *	Enables Preamble detector when set to 1. The AGC settings supersede this bit during the startup / AGC phase. 0 → Turned off 1 → Turned on
	6-5	PreambleDetectorSize	rw	0x01 *	Number of Preamble bytes to detect to trigger an interrupt 00 → 1 byte 10 → 3 bytes 01 → 2 bytes 11 → Reserved
	4-0	PreambleDetectorTol	rw	0x0A *	Number or chip errors tolerated over PreambleDetectorSize. 4 chips per bit.
RegRxTimeout1 (0x20)	7-0	TimeoutRxRssi	rw	0x00	*Timeout* interrupt is generated *TimeoutRxRssi**16*T_{bit} after switching to Rx mode if *Rssi* interrupt doesn't occur (i.e. *RssiValue* > *RssiThreshold)* 0x00: *TimeoutRxRssi* is disabled
RegRxTimeout2 (0x21)	7-0	TimeoutRxPreamble	rw	0x00	*Timeout* interrupt is generated *TimeoutRxPreamble**16*T_{bit} after switching to Rx mode if *Preamble* interrupt doesn't occur 0x00: *TimeoutRxPreamble* is disabled
RegRxTimeout3 (0x22)	7-0	TimeoutSignalSync	rw	0x00	*Timeout* interrupt is generated *TimeoutSignalSync**16*T_{bit} after the Rx mode is programmed, if *SyncAddress* doesn't occur 0x00: *TimeoutSignalSync* is disabled
RegRxDelay (0x23)	7-0	InterPacketRxDelay	rw	0x00	Additional delay before an automatic receiver restart is launched: Delay = InterPacketRxDelay*4*Tbit
RC Oscillator registers					
RegOsc (0x24)	7-4	unused	r	-	unused
	3	RcCalStart	wt	0x00	Triggers the calibration of the RC oscillator when set. Always reads 0. RC calibration must be triggered in Standby mode.
	2-0	ClkOut	rw	0x07 *	Selects CLKOUT frequency: 000 → FXOSC 001 → FXOSC / 2 010 → FXOSC / 4 011 → FXOSC / 8 100 → FXOSC / 16 101 → FXOSC / 32 110 → RC (automatically enabled) 111 → OFF
Packet Handling registers					

Name (Address)	Bits	Variable Name	Mode	Default value	FSK/OOK Description
RegPreambleMsb (0x25)	7-0	PreambleSize(15:8)	rw	0x00	Size of the preamble to be sent (from *TxStartCondition* fulfilled). (MSB byte)
RegPreambleLsb (0x26)	7-0	PreambleSize(7:0)	rw	0x03	Size of the preamble to be sent (from *TxStartCondition* fulfilled). (LSB byte)
RegSyncConfig (0x27)	7-6	AutoRestartRxMode	rw	0x02	Controls the automatic restart of the receiver after the reception of a valid packet (PayloadReady or CrcOk): 00 → Off 01 → On, without waiting for the PLL to re-lock 10 → On, wait for the PLL to lock (frequency changed) 11 → reserved
	5	PreamblePolarity	rw	0x00	Sets the polarity of the Preamble 0 → 0xAA (default) 1 → 0x55
	4	SyncOn	rw	0x01	Enables the Sync word generation and detection: 0 → Off 1 → On
	3	reserved	rw	0x00	reserved
	2-0	SyncSize	rw	0x03	Size of the Sync word: (*SyncSize* + 1) bytes, (*SyncSize*) bytes if *ioHomeOn*=1
RegSyncValue1 (0x28)	7-0	SyncValue(63:56)	rw	0x01 *	1st byte of Sync word. (MSB byte) Used if *SyncOn* is set.
RegSyncValue2 (0x29)	7-0	SyncValue(55:48)	rw	0x01 *	2nd byte of Sync word Used if *SyncOn* is set and *(SyncSize +1)* >= 2.
RegSyncValue3 (0x2a)	7-0	SyncValue(47:40)	rw	0x01 *	3rd byte of Sync word. Used if *SyncOn* is set and *(SyncSize +1)* >= 3.
RegSyncValue4 (0x2b)	7-0	SyncValue(39:32)	rw	0x01 *	4th byte of Sync word. Used if *SyncOn* is set and *(SyncSize +1)* >= 4.
RegSyncValue5 (0x2c)	7-0	SyncValue(31:24)	rw	0x01 *	5th byte of Sync word. Used if *SyncOn* is set and *(SyncSize +1)* >= 5.
RegSyncValue6 (0x2d)	7-0	SyncValue(23:16)	rw	0x01 *	6th byte of Sync word. Used if *SyncOn* is set and *(SyncSize +1)* >= 6.
RegSyncValue7 (0x2e)	7-0	SyncValue(15:8)	rw	0x01 *	7th byte of Sync word. Used if *SyncOn* is set and *(SyncSize +1)* >= 7.
RegSyncValue8 (0x2f)	7-0	SyncValue(7:0)	rw	0x01 *	8th byte of Sync word. Used if *SyncOn* is set and *(SyncSize +1)* = 8.

Name (Address)	Bits	Variable Name	Mode	Default value	FSK/OOK Description
RegPacketConfig1 (0x30)	7	PacketFormat	rw	0x01	Defines the packet format used: 0 → Fixed length 1 → Variable length
	6-5	DcFree	rw	0x00	Defines DC-free encoding/decoding performed: 00 → None (Off) 01 → Manchester 10 → Whitening 11 → reserved
	4	CrcOn	rw	0x01	Enables CRC calculation/check (Tx/Rx): 0 → Off 1 → On
	3	CrcAutoClearOff	rw	0x00	Defines the behavior of the packet handler when CRC check fails: 0 → Clear FIFO and restart new packet reception. No *PayloadReady* interrupt issued. 1 → Do not clear FIFO. *PayloadReady* interrupt issued.
	2-1	AddressFiltering	rw	0x00	Defines address based filtering in Rx: 00 → None (Off) 01 → Address field must match *NodeAddress* 10 → Address field must match *NodeAddress* or *BroadcastAddress* 11 → reserved
	0	CrcWhiteningType	rw	0x00	Selects the CRC and whitening algorithms: 0 → CCITT CRC implementation with standard whitening 1 → IBM CRC implementation with alternate whitening
RegPacketConfig2 (0x31)	7	unused	r	-	unused
	6	DataMode	rw	0x01	Data processing mode: 0 → Continuous mode 1 → Packet mode
	5	IoHomeOn	rw	0x00	Enables the io-homecontrol® compatibility mode 0 → Disabled 1 → Enabled
	4	IoHomePowerFrame	rw	0x00	reserved - Linked to io-homecontrol® compatibility mode
	3	BeaconOn	rw	0x00	Enables the Beacon mode in Fixed packet format
	2-0	PayloadLength(10:8)	rw	0x00	Packet Length Most significant bits
RegPayloadLength (0x32)	7-0	PayloadLength(7:0)	rw	0x40	If PacketFormat = 0 (fixed), payload length. If PacketFormat = 1 (variable), max length in Rx, not used in Tx.
RegNodeAdrs (0x33)	7-0	NodeAddress	rw	0x00	Node address used in address filtering.
RegBroadcastAdrs (0x34)	7-0	BroadcastAddress	rw	0x00	Broadcast address used in address filtering.

Name (Address)	Bits	Variable Name	Mode	Default value	FSK/OOK Description
RegFifoThresh (0x35)	7	TxStartCondition	rw	0x01 *	Defines the condition to start packet transmission: 0 → *FifoLevel* (i.e. the number of bytes in the FIFO exceeds *FifoThreshold)* 1 → *FifoEmpty goes low*(i.e. at least one byte in the FIFO)
	6	unused	r	-	unused
	5-0	FifoThreshold	rw	0x0f	Used to trigger *FifoLevel* interrupt, when: number of bytes in FIFO >= FifoThreshold + 1
Sequencer registers					
RegSeqConfig1 (0x36)	7	SequencerStart	wt	0x00	Controls the top level Sequencer When set to '1', executes the "Start" transition. The sequencer can only be enabled when the chip is in Sleep or Standby mode.
	6	SequencerStop	wt	0x00	Forces the Sequencer Off. Always reads '0'
	5	IdleMode	rw	0x00	Selects chip mode during the state: 0: Standby mode 1: Sleep mode
	4-3	FromStart	rw	0x00	Controls the Sequencer transition when *SequencerStart* is set to 1 in Sleep or Standby mode: 00: to LowPowerSelection 01: to Receive state 10: to Transmit state 11: to Transmit state on a *FifoLevel* interrupt
	2	LowPowerSelection	rw	0x00	Selects the Sequencer LowPower state after a *to LowPowerSelection* transition: 0: SequencerOff state with chip on Initial mode 1: Idle state with chip on *Standby* or *Sleep* mode depending on *IdleMode* *Note: Initial mode is the chip LowPower mode at Sequencer Start.*
	1	FromIdle	rw	0x00	Controls the Sequencer transition from the Idle state on a T1 interrupt: 0: to Transmit state 1: to Receive state
	0	FromTransmit	rw	0x00	Controls the Sequencer transition from the Transmit state: 0: to LowPowerSelection on a *PacketSent* interrupt 1: to Receive state on a *PacketSent* interrupt

Name (Address)	Bits	Variable Name	Mode	Default value	FSK/OOK Description
RegSeqConfig2 (0x37)	7-5	FromReceive	rw	0x00	Controls the Sequencer transition from the Receive state 000 and 111: unused 001: to PacketReceived state on a *PayloadReady* interrupt 010: to LowPowerSelection on a *PayloadReady* interrupt 011: to PacketReceived state on a *CrcOk* interrupt (1) 100: to SequencerOff state on a *Rssi* interrupt 101: to SequencerOff state on a *SyncAddress* interrupt 110: to SequencerOff state on a *PreambleDetect* interrupt Irrespective of this setting, transition to LowPowerSelection on a T2 interrupt (1) If the CRC is wrong (corrupted packet, with CRC on but *CrcAutoClearOn*=0), the *PayloadReady* interrupt will drive the sequencer to RxTimeout state.
	4-3	FromRxTimeout	rw	0x00	Controls the state-machine transition from the Receive state on a *RxTimeout* interrupt (and on *PayloadReady* if FromReceive = 011): 00: to Receive State, via ReceiveRestart 01: to Transmit state 10: to LowPowerSelection 11: to SequencerOff state *Note: RxTimeout interrupt is a TimeoutRxRssi, TimeoutRxPreamble or TimeoutSignalSync interrupt*
	2-0	FromPacketReceived	rw	0x00	Controls the state-machine transition from the PacketReceived state: 000: to SequencerOff state 001: to Transmit state on a *FifoEmpty* interrupt 010: to LowPowerSelection 011: to Receive via FS mode, if frequency was changed 100: to Receive state (no frequency change)
RegTimerResol (0x38)	7-4	unused	r	-	unused
	3-2	Timer1Resolution	rw	0x00	Resolution of Timer 1 00: Timer1 disabled 01: 64 us 10: 4.1 ms 11: 262 ms
	1-0	Timer2Resolution	rw	0x00	Resolution of Timer 2 00: Timer2 disabled 01: 64 us 10: 4.1 ms 11: 262 ms
RegTimer1Coef (0x39)	7-0	Timer1Coefficient	rw	0xf5	Multiplying coefficient for Timer 1
RegTimer2Coef (0x3a)	7-0	Timer2Coefficient	rw	0x20	Multiplying coefficient for Timer 2

Name (Address)	Bits	Variable Name	Mode	Default value	FSK/OOK Description
Service registers					
RegImageCal (0x3b)	7	AutoImageCalOn	rw	0x00 *	Controls the Image calibration mechanism 0 → Calibration of the receiver depending on the temperature is disabled 1 → Calibration of the receiver depending on the temperature enabled.
	6	ImageCalStart	wt	-	Triggers the IQ and RSSI calibration when set in Standby mode.
	5	ImageCalRunning	r	0x00	Set to 1 while the Image and RSSI calibration are running. Toggles back to 0 when the process is completed
	4	unused	r	-	unused
	3	TempChange	r	0x00	IRQ flag witnessing a temperature change exceeding TempThreshold since the last Image and RSSI calibration: 0 → Temperature change lower than TempThreshold 1 → Temperature change greater than TempThreshold
	2-1	TempThreshold	rw	0x01	Temperature change threshold to trigger a new I/Q calibration 00 → 5 °C 01 → 10 °C 10 → 15 °C 11 → 20 °C
	0	TempMonitorOff	rw	0x00	Controls the temperature monitor operation: 0 → Temperature monitoring done in all modes except Sleep and Standby 1 → Temperature monitoring stopped.
RegTemp (0x3c)	7-0	TempValue	r	-	Measured temperature -1°C per Lsb Needs calibration for absolute accuracy
RegLowBat (0x3d)	7-4	unused	r	-	unused
	3	LowBatOn	rw	0x00	Low Battery detector enable signal 0 → LowBat detector disabled 1 → LowBat detector enabled
	2-0	LowBatTrim	rw	0x02	Trimming of the LowBat threshold: 000 → 1.695 V 001 → 1.764 V 010 → 1.835 V (d) 011 → 1.905 V 100 → 1.976 V 101 → 2.045 V 110 → 2.116 V 111 → 2.185 V
Status registers					

Name (Address)	Bits	Variable Name	Mode	Default value	FSK/OOK Description
RegIrqFlags1 (0x3e)	7	ModeReady	r	-	Set when the operation mode requested in *Mode*, is ready - Sleep: Entering Sleep mode - Standby: XO is running - FS: PLL is locked - Rx: RSSI sampling starts - Tx: PA ramp-up completed Cleared when changing the operating mode.
	6	RxReady	r	-	Set in Rx mode, after RSSI, AGC and AFC. Cleared when leaving Rx.
	5	TxReady	r	-	Set in Tx mode, after PA ramp-up. Cleared when leaving Tx.
	4	PllLock	r	-	Set (in FS, Rx or Tx) when the PLL is locked. Cleared when it is not.
	3	Rssi	rwc	-	Set in Rx when the *RssiValue* exceeds *RssiThreshold.* Cleared when leaving Rx or setting this bit to 1.
	2	Timeout	r	-	Set when a timeout occurs Cleared when leaving Rx or FIFO is emptied.
	1	PreambleDetect	rwc	-	Set when the Preamble Detector has found valid Preamble. bit clear when set to 1
	0	SyncAddressMatch	rwc	-	Set when Sync and Address (if enabled) are detected. Cleared when leaving Rx or FIFO is emptied. This bit is read only in Packet mode, rwc in Continuous mode
RegIrqFlags2 (0x3f)	7	FifoFull	r	-	Set when FIFO is full (i.e. contains 66 bytes), else cleared.
	6	FifoEmpty	r	-	Set when FIFO is empty, and cleared when there is at least 1 byte in the FIFO.
	5	FifoLevel	r	-	Set when the number of bytes in the FIFO strictly exceeds *FifoThreshold*, else cleared.
	4	FifoOverrun	rwc	-	Set when FIFO overrun occurs. (except in Sleep mode) Flag(s) and FIFO are cleared when this bit is set. The FIFO then becomes immediately available for the next transmission / reception.
	3	PacketSent	r	-	Set in Tx when the complete packet has been sent. Cleared when exiting Tx
	2	PayloadReady	r	-	Set in Rx when the payload is ready (i.e. last byte received and CRC, if enabled and *CrcAutoClearOff* is cleared, is Ok). Cleared when FIFO is empty.
	1	CrcOk	r	-	Set in Rx when the CRC of the payload is Ok. Cleared when FIFO is empty.
	0	LowBat	rwc	-	Set when the battery voltage drops below the Low Battery threshold. Cleared only when set to 1 by the user.
IO control registers					

Name (Address)	Bits	Variable Name	Mode	Default value	FSK/OOK Description
RegDioMapping1 (0x40)	7-6	Dio0Mapping	rw	0x00	Mapping of pins DIO0 to DIO5 See Table 18 for mapping in LoRa mode See Table 29 for mapping in Continuous mode SeeTable 30 for mapping in Packet mode
	5-4	Dio1Mapping	rw	0x00	
	3-2	Dio2Mapping	rw	0x00	
	1-0	Dio3Mapping	rw	0x00	
RegDioMapping2 (0x41)	7-6	Dio4Mapping	rw	0x00	
	5-4	Dio5Mapping	rw	0x00	
	3-1	reserved	rw	0x00	reserved. Retain default value
	0	MapPreambleDetect	rw	0x00	Allows the mapping of either *Rssi* Or *PreambleDetect* to the DIO pins, as summarized on Table 29 and Table 30 0 → *Rssi* interrupt 1 → *PreambleDetect* interrupt
Version register					
RegVersion (0x42)	7-0	Version	r	0x12	Version code of the chip. Bits 7-4 give the full revision number; bits 3-0 give the metal mask revision number.
Additional registers					
RegPllHop (0x44)	7	FastHopOn	rw	0x00	Bypasses the main state machine for a quick frequency hop. Writing RegFrfLsb will trigger the frequency change. 0 → Frf is validated when FSTx or FSRx is requested 1 → Frf is validated triggered when RegFrfLsb is written
	6-0	reserved	rw	0x2d	reserved
RegTcxo (0x4b)	7-5	reserved	rw	0x00	reserved. Retain default value
	4	TcxoInputOn	rw	0x00	Controls the crystal oscillator 0 → Crystal Oscillator with external Crystal 1 → External clipped sine TCXO AC-connected to XTA pin
	3-0	reserved	rw	0x09	Reserved. Retain default value.
RegPaDac (0x4d)	7-3	reserved	rw	0x10	reserved. Retain default value
	2-0	PaDac	rw	0x04	Enables the +20dBm option on PA_BOOST pin 0x04 → Default value 0x07 → +20dBm on PA_BOOST when OutputPower=1111
RegFormerTemp (0x5b)	7-0	FormerTemp	rw	-	Temperature saved during the latest IQ (RSSI and Image) calibration. Same format as *TempValue* in *RegTemp*.
RegBitrateFrac (0x5d)	7-4	unused	r	0x00	unused
	3-0	BitRateFrac	rw	0x00	Fractional part of the bit rate divider (Only valid for FSK) If *BitRateFrac*> 0 then: $BitRate = \frac{FXOSC}{BitRate(15,0) + \frac{BitrateFrac}{16}}$

Name (Address)	Bits	Variable Name	Mode	Default value	FSK/OOK Description
RegAgcRef (0x61)	7-6	unused	r	-	unused
	5-0	AgcReferenceLevel	rw	0x19	Sets the floor reference for all AGC thresholds: AGC Reference[dBm]= -174dBm+10*log(2**RxBw*)+SNR+*AgcReferenceLevel* SNR = 8dB, fixed value
RegAgcThresh1 (0x62)	7-5	unused	r	-	unused
	4-0	AgcStep1	rw	0x0c	Defines the 1st AGC Threshold
RegAgcThresh2 (0x63)	7-4	AgcStep2	rw	0x04	Defines the 2nd AGC Threshold:
	3-0	AgcStep3	rw	0x0b	Defines the 3rd AGC Threshold:
RegAgcThresh3 (0x64)	7-4	AgcStep4	rw	0x0c	Defines the 4th AGC Threshold:
	3-0	AgcStep5	rw	0x0c	Defines the 5th AGC Threshold:

6.3. Band Specific Additional Registers

The registers in the address space from 0x61 to 0x73 are specific for operation in the lower frequency bands (below 525 MHz), or in the upper frequency bands (above 779 MHz). Their programmed value may differ, and are retained when switching from lower to high frequency and vice-versa. The access to the band specific registers is granted by enabling or disabling the bit 3 *LowFrequencyModeOn* of the *RegOpMode* register. By default, the bit *LowFrequencyModeOn* is at '1' indicating that the registers are configured for the low frequency band.

Table 43 Low Frequency Additional Registers

Name (Address)	Bits	Variable Name	Mode	Default value	Low Frequency Additional Registers
RegAgcRefLf (0x61)	7-6	unused	r	-	unused
	5-0	AgcReferenceLevel	rw	0x19	Sets the floor reference for all AGC thresholds: AGC Reference[dBm]= -174dBm+10*log(2**RxBw*)+SNR+*AgcReferenceLevel* SNR = 8dB, fixed value
RegAgcThresh1Lf (0x62)	7-5	unused	r	-	unused
	4-0	AgcStep1	rw	0x0c	Defines the 1st AGC Threshold
RegAgcThresh2Lf (0x63)	7-4	AgcStep2	rw	0x04	Defines the 2nd AGC Threshold:
	3-0	AgcStep3	rw	0x0b	Defines the 3rd AGC Threshold:
RegAgcThresh3Lf (0x64)	7-4	AgcStep4	rw	0x0c	Defines the 4th AGC Threshold:
	3-0	AgcStep5	rw	0x0c	Defines the 5th AGC Threshold:
RegPllLf (0x70)	7-6	PllBandwidth	rw	0x03	Controls the PLL bandwidth: 00 → 75 kHz 10 → 225 kHz 01 → 150 kHz 11 → 300 kHz
	5-0	reserved	rw	0x10	reserved. Retain default value

Table 44 High Frequency Additional Registers

Name (Address)	Bits	Variable Name	Mode	Default value	Low Frequency Additional Registers
RegAgcRefHf (0x61)	7-6	unused	r	-	unused
	5-0	AgcReferenceLevel	rw	0x1c	Sets the floor reference for all AGC thresholds: AGC Reference[dBm]= -174dBm+10*log(2**RxBw*)+SNR+*AgcReferenceLevel* SNR = 8dB, fixed value
RegAgcThresh1Hf (0x62)	7-5	unused	r	-	unused
	4-0	AgcStep1	rw	0x0e	Defines the 1st AGC Threshold
RegAgcThresh2Hf (0x63)	7-4	AgcStep2	rw	0x05	Defines the 2nd AGC Threshold:
	3-0	AgcStep3	rw	0x0b	Defines the 3rd AGC Threshold:
RegAgcThresh3Hf (0x64)	7-4	AgcStep4	rw	0x0c	Defines the 4th AGC Threshold:
	3-0	AgcStep5	rw	0x0c	Defines the 5th AGC Threshold:
RegPllHf (0x70)	7-6	PllBandwidth	rw	0x03	Controls the PLL bandwidth: 00 → 75 kHz 10 → 225 kHz 01 → 150 kHz 11 → 300 kHz
	5-0	reserved	rw	0x10	reserved. Retain default value

6.4. LoRa™ Mode Register Map

This details the SX1276/77/78/79 register mapping and the precise contents of each register in LoRa™ mode.

It is essential to understand that the LoRa™ modem is controlled independently of the FSK modem. Therefore, care should be taken when accessing the registers, especially as some register may have the same name in LoRa™ or FSK mode.

The LoRa registers are only accessible when the device is set in Lora mode (and, in the same way, the FSK register are only accessible in FSK mode). However, in some cases, it may be necessary to access some of the FSK register while in LoRa mode. To this aim, the *AccesSharedReg* bit was created in the *RegOpMode* register. This bit, when set to '1', will grant access to the FSK register 0x0D up to the register 0x3F. Once the setup has been done, it is strongly recommended to clear this bit so that LoRa register can be accessed normally.

Convention: r: read, w: write, c : set to clear and t: trigger.

Name (Address)	Bits	Variable Name	Mode	Reset	LoRa™ Description
RegFifo (0x00)	7-0	Fifo	rw	0x00	LoRa™ base-band FIFO data input/output. FIFO is cleared an not accessible when device is in SLEEP mode
Common Register Settings					
RegOpMode (0x01)	7	LongRangeMode	rw	0x0	0 → FSK/OOK Mode 1 → LoRa™ Mode This bit can be modified only in Sleep mode. A write operation on other device modes is ignored.
	6	AccessSharedReg	rw	0x0	This bit operates when device is in Lora mode; if set it allows access to FSK registers page located in address space (0x0D:0x3F) while in LoRa mode 0 → Access LoRa registers page 0x0D: 0x3F 1 → Access FSK registers page (in mode LoRa) 0x0D: 0x3F
	5-4	reserved	r	0x00	reserved
	3	LowFrequencyModeOn	rw	0x01	Access Low Frequency Mode registers 0 → High Frequency Mode (access to HF test registers) 1 → Low Frequency Mode (access to LF test registers)
	2-0	Mode	rwt	0x01	Device modes 000 → SLEEP 001 → STDBY 010 → Frequency synthesis TX (FSTX) 011 → Transmit (TX) 100 → Frequency synthesis RX (FSRX) 101 → Receive continuous (RXCONTINUOUS) 110 → receive single (RXSINGLE) 111 → Channel activity detection (CAD)
(0x02)	7-0	reserved	r	0x00	-
(0x03)	7-0	reserved	r	0x00	-
(0x04)	7-0	reserved	rw	0x00	-
(0x05)	7-0	reserved	r	0x00	-
RegFrMsb (0x06)	7-0	Frf(23:16)	rw	0x6c	MSB of RF carrier frequency

Name (Address)	Bits	Variable Name	Mode	Reset	LoRa™ Description
RegFrMid (0x07)	7-0	Frf(15:8)	rw	0x80	MSB of RF carrier frequency
RegFrLsb (0x08)	7-0	Frf(7:0)	rwt	0x00	LSB of RF carrier frequency $f_{RF} = \frac{F(XOSC) \cdot Frf}{2^{19}}$ Resolution is 61.035 Hz if F(XOSC) = 32 MHz. Default value is 0x6c8000 = 434 MHz. Register values must be modified only when device is in SLEEP or STAND-BY mode.
Registers for RF blocks					
RegPaConfig (0x09)	7	PaSelect	rw	0x00	Selects PA output pin 0 → RFO pin. Output power is limited to +14 dBm. 1 → PA_BOOST pin. Output power is limited to +20 dBm
	6-4	MaxPower	rw	0x04	Select max output power: Pmax=10.8+0.6*MaxPower [dBm]
	3-0	OutputPower	rw	0x0f	Pout=Pmax-(15-OutputPower) if PaSelect = 0 (RFO pin) Pout=17-(15-OutputPower) if PaSelect = 1 (PA_BOOST pin)
RegPaRamp (0x0A)	7-5	unused	r	-	unused
	4	reserved	rw	0x00	reserved
	3-0	PaRamp(3:0)	rw	0x09	Rise/Fall time of ramp up/down in FSK 0000 → 3.4 ms 0001 → 2 ms 0010 → 1 ms 0011 → 500 us 0100 → 250 us 0101 → 125 us 0110 → 100 us 0111 → 62 us 1000 → 50 us 1001 → 40 us 1010 → 31 us 1011 → 25 us 1100 → 20 us 1101 → 15 us 1110 → 12 us 1111 → 10 us
RegOcp (0x0B	7-6	unused	r	0x00	unused
	5	OcpOn	rw	0x01	Enables overload current protection (OCP) for PA: 0 → OCP disabled 1 → OCP enabled
	4-0	OcpTrim	rw	0x0b	Trimming of OCP current: Imax = 45+5*OcpTrim [mA] if OcpTrim <= 15 (120 mA) / Imax = -30+10*OcpTrim [mA] if 15 < OcpTrim <= 27 (130 to 240 mA) Imax = 240mA for higher settings Default Imax = 100mA

Name (Address)	Bits	Variable Name	Mode	Reset	LoRa™ Description
RegLna (0x0C)	7-5	LnaGain	rwx	0x01	LNA gain setting: 000 → not used 001 → G1 = maximum gain 010 → G2 011 → G3 100 → G4 101 → G5 110 → G6 = minimum gain 111 → not used
	4-3	LnaBoostLf	rw	0x00	Low Frequency (RFI_LF) LNA current adjustment 00 → Default LNA current Other → Reserved
	2	reserved	rw	0x00	reserved
	1-0	LnaBoostHf	rw	0x00	High Frequency (RFI_HF) LNA current adjustment 00 → Default LNA current 11 → Boost on, 150% LNA current
Lora page registers					
RegFifoAddrPtr (0x0D)	7-0	FifoAddrPtr	rw	0x00	SPI interface address pointer in FIFO data buffer.
RegFifoTxBaseAddr (0x0E)	7-0	FifoTxBaseAddr	rw	0x80	write base address in FIFO data buffer for TX modulator
RegFifoRxBaseAddr (0x0F)	7-0	FifoRxBaseAddr	rw	0x00	read base address in FIFO data buffer for RX demodulator
RegFifoRxCurrentAddr (0x10)	7-0	FifoRxCurrentAddr	r	n/a	Start address (in data buffer) of last packet received
RegIrqFlagsMask (0x11)	7	RxTimeoutMask	rw	0x00	Timeout interrupt mask: setting this bit masks the corresponding IRQ in RegIrqFlags
	6	RxDoneMask	rw	0x00	Packet reception complete interrupt mask: setting this bit masks the corresponding IRQ in RegIrqFlags
	5	PayloadCrcErrorMask	rw	0x00	Payload CRC error interrupt mask: setting this bit masks the corresponding IRQ in RegIrqFlags
	4	ValidHeaderMask	rw	0x00	Valid header received in Rx mask: setting this bit masks the corresponding IRQ in RegIrqFlags
	3	TxDoneMask	rw	0x00	FIFO Payload transmission complete interrupt mask: setting this bit masks the corresponding IRQ in RegIrqFlags
	2	CadDoneMask	rw	0x00	CAD complete interrupt mask: setting this bit masks the corresponding IRQ in RegIrqFlags
	1	FhssChangeChannelMask	rw	0x00	FHSS change channel interrupt mask: setting this bit masks the corresponding IRQ in RegIrqFlags
	0	CadDetectedMask	rw	0x00	Cad Detected Interrupt Mask: setting this bit masks the corresponding IRQ in RegIrqFlags

Name (Address)	Bits	Variable Name	Mode	Reset	LoRa™ Description
RegIrqFlags (0x12)	7	RxTimeout	rc	0x00	Timeout interrupt: writing a 1 clears the IRQ
	6	RxDone	rc	0x00	Packet reception complete interrupt: writing a 1 clears the IRQ
	5	PayloadCrcError	rc	0x00	Payload CRC error interrupt: writing a 1 clears the IRQ
	4	ValidHeader	rc	0x00	Valid header received in Rx: writing a 1 clears the IRQ
	3	TxDone	rc	0x00	FIFO Payload transmission complete interrupt: writing a 1 clears the IRQ
	2	CadDone	rc	0x00	CAD complete: write to clear: writing a 1 clears the IRQ
	1	FhssChangeChannel	rc	0x00	FHSS change channel interrupt: writing a 1 clears the IRQ
	0	CadDetected	rc	0x00	Valid Lora signal detected during CAD operation: writing a 1 clears the IRQ
RegRxNbBytes (0x13)	7-0	FifoRxBytesNb	r	n/a	Number of payload bytes of latest packet received
RegRxHeaderCntValueMsb (0x14)	7-0	ValidHeaderCntMsb(15:8)	r	n/a	Number of valid headers received since last transition into Rx mode, MSB(15:8). Header and packet counters are reseted in Sleep mode.
RegRxHeaderCntValueLsb (0x15)	7-0	ValidHeaderCntLsb(7:0)	r	n/a	Number of valid headers received since last transition into Rx mode, LSB(7:0). Header and packet counters are reseted in Sleep mode.
RegRxPacketCntValueMsb (0x16)	7-0	ValidPacketCntMsb(15:8)	rc	n/a	Number of valid packets received since last transition into Rx mode, MSB(15:8). Header and packet counters are reseted in Sleep mode.
RegRxPacketCntValueLsb (0x17)	7-0	ValidPacketCntLsb(7:0)	r	n/a	Number of valid packets received since last transition into Rx mode, LSB(7:0). Header and packet counters are reseted in Sleep mode.
RegModemStat (0x18)	7-5	RxCodingRate	r	n/a	Coding rate of last header received
	4	ModemStatus	r	‘1’	Modem clear
	3		r	‘0’	Header info valid
	2		r	‘0’	RX on-going
	1		r	‘0’	Signal synchronized
	0		r	‘0’	Signal detected

Name (Address)	Bits	Variable Name	Mode	Reset	LoRa™ Description
RegPktSnrValue (0x19)	7-0	PacketSnr	r	n/a	Estimation of SNR on last packet received.In two's compliment format mutiplied by 4. $SNR[dB] = \frac{PacketSnr[twos\ complement]}{4}$
RegPktRssiValue (0x1A)	7-0	PacketRssi	r	n/a	RSSI of the latest packet received (dBm): RSSI[dBm] = -157 + Rssi (using HF output port, SNR >= 0) or RSSI[dBm] = -164 + Rssi (using LF output port, SNR >= 0) (see section 5.5.5 for details)
RegRssiValue (0x1B)	7-0	Rssi	r	n/a	Current RSSI value (dBm) RSSI[dBm] = -157 + Rssi (using HF output port) or RSSI[dBm] = -164 + Rssi (using LF output port) (see section 5.5.5 for details)
RegHopChannel (0x1C)	7	PllTimeout	r	n/a	PLL failed to lock while attempting a TX/RX/CAD operation 1 → PLL did not lock 0 → PLL did lock
	6	CrcOnPayload	r	n/a	CRC Information extracted from the received packet header (Explicit header mode only) 0 → Header indicates CRC off 1 → Header indicates CRC on
	5-0	FhssPresentChannel	r	n/a	Current value of frequency hopping channel in use.

Name (Address)	Bits	Variable Name	Mode	Reset	LoRa™ Description
RegModemConfig1 (0x1D)	7-4	Bw	rw	0x07	Signal bandwidth: 0000 → 7.8 kHz 0001 → 10.4 kHz 0010 → 15.6 kHz 0011 → 20.8kHz 0100 → 31.25 kHz 0101 → 41.7 kHz 0110 → 62.5 kHz 0111 → 125 kHz 1000 → 250 kHz 1001 → 500 kHz other values → reserved In the lower band (169MHz), signal bandwidths 8&9 are not supported)
	3-1	CodingRate	rw	'001'	Error coding rate 001 → 4/5 010 → 4/6 011 → 4/7 100 → 4/8 All other values → reserved In implicit header mode should be set on receiver to determine expected coding rate. See 4.1.1.3
	0	ImplicitHeaderModeOn	rw	0x0	0 → Explicit Header mode 1 → Implicit Header mode
RegModemConfig2 (0x1E)	7-4	SpreadingFactor	rw	0x07	SF rate (expressed as a base-2 logarithm) 6 → 64 chips / symbol 7 → 128 chips / symbol 8 → 256 chips / symbol 9 → 512 chips / symbol 10 → 1024 chips / symbol 11 → 2048 chips / symbol 12 → 4096 chips / symbol other values reserved.
	3	TxContinuousMode	rw	0	0 → normal mode, a single packet is sent 1 → continuous mode, send multiple packets across the FIFO (used for spectral analysis)
	2	RxPayloadCrcOn	rw	0x00	Enable CRC generation and check on payload: 0 → CRC disable 1 → CRC enable If CRC is needed, RxPayloadCrcOn should be set: - in Implicit header mode: on Tx and Rx side - in Explicit header mode: on the Tx side alone (recovered from the header in Rx side)
	1-0	SymbTimeout(9:8)	rw	0x00	RX Time-Out MSB
RegSymbTimeoutLsb (0x1F)	7-0	SymbTimeout(7:0)	rw	0x64	RX Time-Out LSB RX operation time-out value expressed as number of symbols: $TimeOut = SymbTimeout \cdot Ts$

Name (Address)	Bits	Variable Name	Mode	Reset	LoRa™ Description
RegPreambleMsb (0x20)	7-0	PreambleLength(15:8)	rw	0x0	Preamble length MSB, = PreambleLength + 4.25 Symbols See 4.1.1 for more details.
RegPreambleLsb (0x21)	7-0	PreambleLength(7:0)	rw	0x8	Preamble Length LSB
RegPayloadLength (0x22)	7-0	PayloadLength(7:0)	rw	0x1	Payload length in bytes. The register needs to be set in implicit header mode for the expected packet length. A 0 value is not permitted
RegMaxPayloadLength (0x23)	7-0	PayloadMaxLength(7:0)	rw	0xff	Maximum payload length; if header payload length exceeds value a header CRC error is generated. Allows filtering of packet with a bad size.
RegHopPeriod (0x24)	7-0	FreqHoppingPeriod(7:0)	rw	0x0	Symbol periods between frequency hops. (0 = disabled). 1st hop always happen after the 1st header symbol
RegFifoRxByteAddr (0x25)	7-0	FifoRxByteAddrPtr	r	n/a	Current value of RX databuffer pointer (address of last byte written by Lora receiver)
RegModemConfig3 (0x26)	7-4	Unused	r	0x00	
	3	LowDataRateOptimize	rw	0x00	0 → Disabled 1 → Enabled; mandated for when the symbol length exceeds 16ms
	2	AgcAutoOn	rw	0x00	0 → LNA gain set by register LnaGain 1 → LNA gain set by the internal AGC loop
	1-0	Reserved	rw	0x00	Reserved
(0x27)	7-0	PpmCorrection	rw	0x00	Data rate offset value, used in conjunction with AFC
RegFeiMsb (0x28)	7-4	Reserved	r	n/a	Reserved
	3-0	FreqError(19:16)	r	0x0	Estimated frequency error from modem MSB of RF Frequency Error $F_{Error} = \frac{FreqError \times 2^{24}}{F_{xtal}} \times \frac{BW[kHz]}{500}$
RegFeiMid (0x29)	7-0	FreqError(15:8)	r	0x0	Middle byte of RF Frequency Error
RegFeiLsb (0x2A)	7-0	FreqError(7:0)	r	0x0	LSB of RF Frequency Error
(0x2B)	-	Reserved	r	n/a	Reserved
RegRssiWideband (0x2C)	7-0	RssiWideband(7:0)	r	n/a	Wideband RSSI measurement used to locally generate a random number
(0x2D) - (0x30)	-	Reserved	r	n/a	Reserved

Name (Address)	Bits	Variable Name	Mode	Reset	LoRa™ Description
RegDetectOptimize (0x31)	7-3	Reserved	r	0xC0	Reserved
	2-0	DetectionOptimize	rw	0x03	LoRa Detection Optimize 0x03 → SF7 to SF12 0x05 → SF6
(0x32)	-	Reserved	r	n/a	Reserved
RegInvertIQ (0x33)	7	Reserved	rw	0x0	Reserved
	6	InvertIQ	rw	0x0	Invert the LoRa I and Q signals 0 → normal mode 1 → I and Q signals are inverted
	5-0	Reserved	rw	0x27	Reserved
(0x34) - (0x36)	7-0	Reserved	r	n/a	Reserved
RegDetectionThreshold (0x37)	7-0	DetectionThreshold	rw	0x0A	LoRa detection threshold 0x0A → SF7 to SF12 0x0C → SF6
(0x38)	-	Reserved	r	n/a	Reserved
RegSyncWord (0x39)	7-0	SyncWord	rw	0x12	LoRa Sync Word Value 0x34 is reserved for LoRaWAN networks
(0x3A) - (0x3F)	-	Reserved	r	n/a	Reserved

7. Application Information

7.1. Crystal Resonator Specification

Table 45 shows the crystal resonator specification for the crystal reference oscillator circuit of the SX1276/77/78/79. This specification covers the full range of operation of the SX1276/77/78/79 and is employed in the reference design.

Table 45 Crystal Specification

Symbol	Description	Conditions	Min	Typ	Max	Unit
FXOSC	XTAL Frequency		-	32	-	MHz
RS	XTAL Serial Resistance		-	15	100	ohms
C0	XTAL Shunt Capacitance		-	1	3	pF
CFOOT	External Foot Capacitance	On each pin XTA and XTB	10	15	22	pF
CLOAD	Crystal Load Capacitance		6	-	12	pF

Notes - the initial frequency tolerance, temperature stability and aging performance should be chosen in accordance with the target operating temperature range and the receiver bandwidth selected.

- the loading capacitance should be applied externally, and adapted to the actual Cload specification of the XTAL.

7.2. Reset of the Chip

A power-on reset of the SX1276/77/78/79 is triggered at power up. Additionally, a manual reset can be issued by controlling pin 7.

7.2.1. POR

If the application requires the disconnection of VDD from the SX1276/77/78/79, despite of the extremely low Sleep Mode current, the user should wait for 10 ms from of the end of the POR cycle before commencing communications over the SPI bus. Pin 7 (NRESET) should be left floating during the POR sequence.

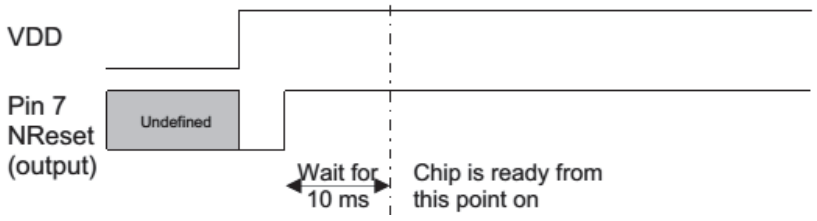


Figure 42. POR Timing Diagram

Please note that any CLKOUT activity can also be used to detect that the chip is ready.

7.2.2. Manual Reset

A manual reset of the SX1276/77/78/79 is possible even for applications in which VDD cannot be physically disconnected. Pin 7 should be pulled low for a hundred microseconds, and then released. The user should then wait for 5 ms before using the chip.

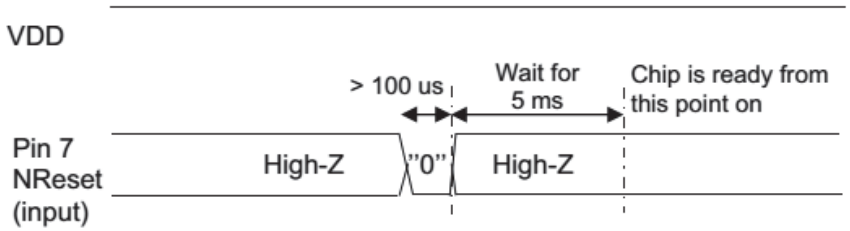


Figure 43. Manual Reset Timing Diagram

Note whilst pin 7 is driven low, an over current consumption of up to one milliampere can be seen on VDD.

7.3. Top Sequencer: Listen Mode Examples

In this scenario, the circuit spends most of the time in Idle mode, during which only the RC oscillator is on. Periodically the receiver wakes up and looks for incoming signal. If a wanted signal is detected, the receiver is kept on and data are analyzed. Otherwise, if there was no wanted signal for a defined period of time, the receiver is switched off until the next receive period.

During Listen mode, the Radio stays most of the time in a Low Power mode, resulting in very low average power consumption. The general timing diagram of this scenario is given in Figure 44.

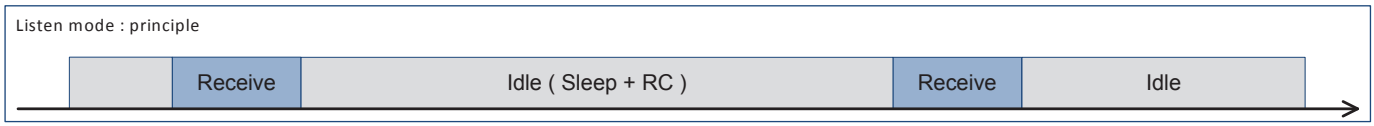


Figure 44. Listen Mode: Principle

An interrupt request is generated on a packet reception. The user can then take appropriate action.

Depending on the application and environment, there are several ways to implement Listen mode:

- Wake on a *PreambleDetect* interrupt
- Wake on *a SyncAddress* interrupt
- Wake on *a PayloadReady* interrupt

7.3.1. Wake on Preamble Interrupt

In one possible scenario, the sequencer polls for a Preamble detection. If a preamble signal is detected, the sequencer is switched off and the circuit stays in Receive mode until the user switches modes. Otherwise, the receiver is switched off until the next Rx period.

7.3.1.1. Timing Diagram

When no signal is received, the circuit wakes every Timer1 + Timer2 and switches to Receive mode for a time defined by Timer2, as shown on the following diagram. If no Preamble is detected, it then switches back to Idle mode, i.e. Sleep mode with RC oscillator on.

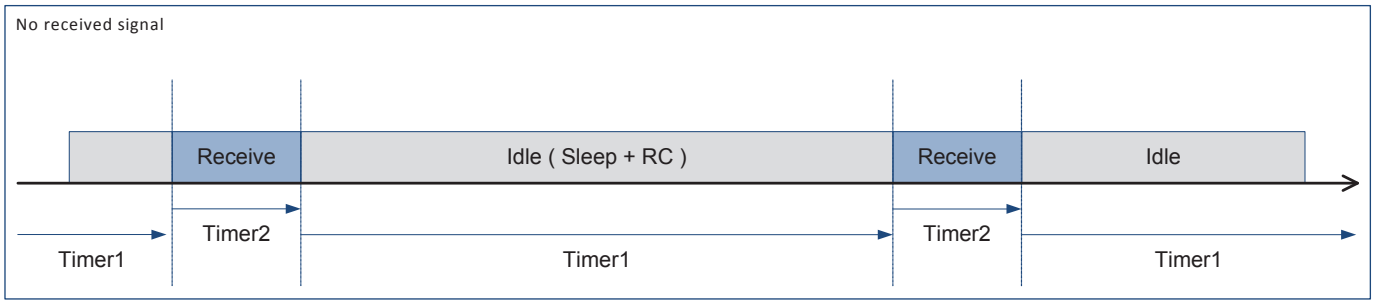


Figure 45. Listen Mode with No Preamble Received

If a Preamble signal is detected, the Sequencer is switched off. The *PreambleDetect* signal can be mapped to DIO4, in order to request the user's attention. The user can then take appropriate action.

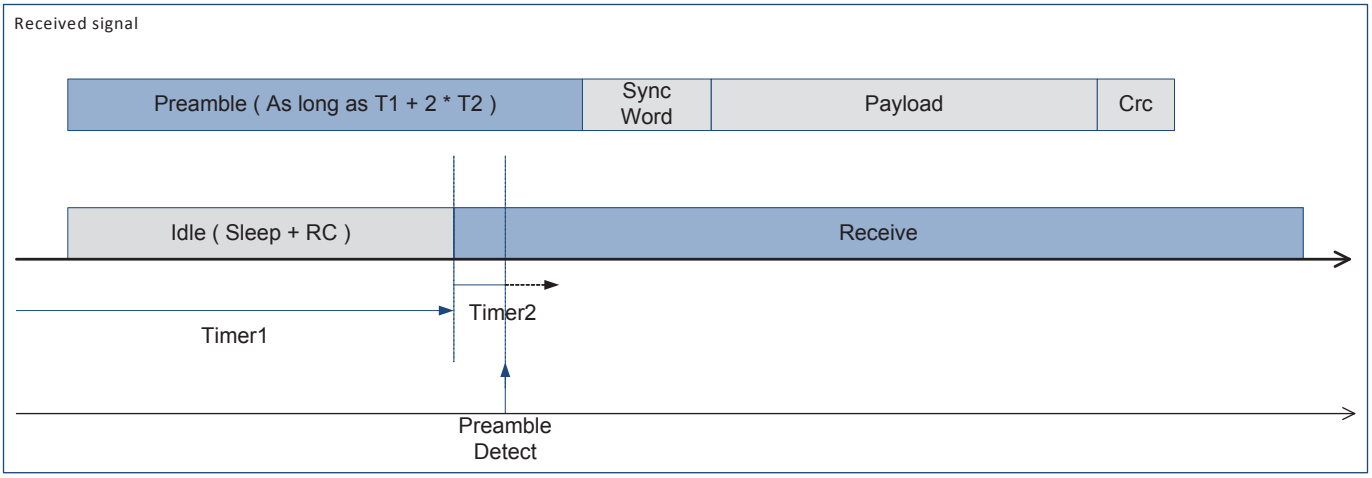


Figure 46. Listen Mode with Preamble Received

7.3.1.2. Sequencer Configuration

The following graph shows Listen mode - Wake on *PreambleDetect* state machine:

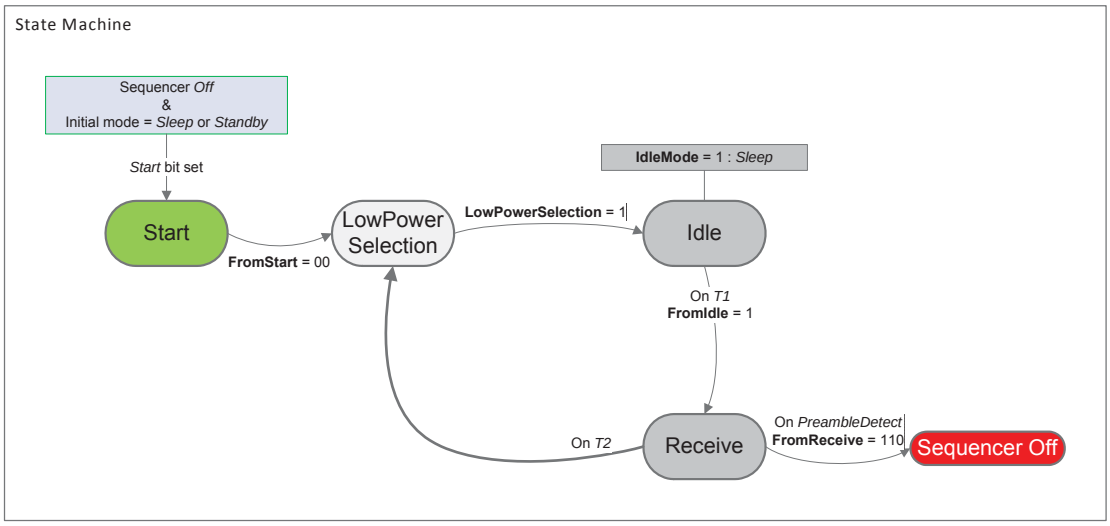


Figure 47. Wake On PreambleDetect State Machine

This example configuration is achieved as follows:

Table 46 Listen Mode with PreambleDetect Condition Settings

Variable	Effect
IdleMode	1: **Sleep** mode
FromStart	00: To **LowPowerSelection**
LowPowerSelection	1: To **Idle** state
FromIdle	1: To **Receive** state on *T1* interrupt
FromReceive	110: To **Sequencer Off** on *PreambleDetect* interrupt

T_{Timer2} defines the maximum duration the chip stays in Receive mode as long as no Preamble is detected. In order to optimize power consumption, Timer2 must be set just long enough for Preamble detection.

T_{Timer1} + T_{Timer2} defines the cycling period, i.e. time between two Preamble polling starts. In order to optimize average power consumption, Timer1 should be relatively long. However, increasing Timer1 also extends packet reception duration.

In order to insure packet detection and optimize the receiver's power consumption, the received packet Preamble should be as long as T_{Timer1} + 2 x T_{Timer2}.

An example of DIO configuration for this mode is described in the following table:

Table 47 Listen Mode with PreambleDetect Condition Recommended DIO Mapping

DIO	Value	Description
0	01	CrcOk
1	00	FifoLevel
3	00	FifoEmpty
4	11	PreambleDetect – Note: *MapPreambleDetect* bit should be set.

7.3.2. Wake on SyncAddress Interrupt

In another possible scenario, the sequencer polls for a Preamble detection and then for a valid *SyncAddress* interrupt. If events occur, the sequencer is switched off and the circuit stays in Receive mode until the user switches modes. Otherwise, the receiver is switched off until the next Rx period.

7.3.2.1. Timing Diagram

Most of the sequencer running time is spent while no wanted signal is received. As shown by the timing diagram in Figure 48, the circuit wakes periodically for a short time, defined by RxTimeout. The circuit is in a Low Power mode for the rest of Timer1 + Timer2 (i.e. Timer1 + Timer2 - TrxTimeout)

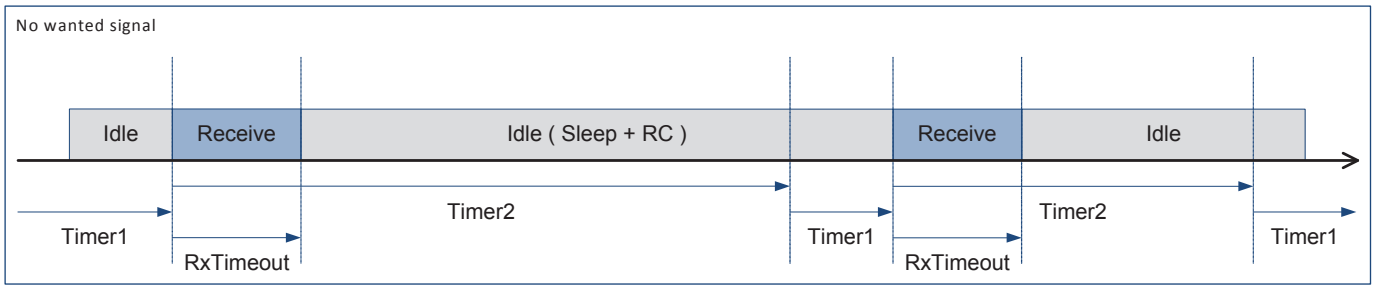


Figure 48. Listen Mode with no SyncAddress Detected

If a preamble is detected before *RxTimeout* timer ends, the circuit stays in Receive mode and waits for a valid *SyncAddress* detection. If none is detected by the end of Timer2, Receive mode is deactivated and the polling cycle resumes, without any user intervention.

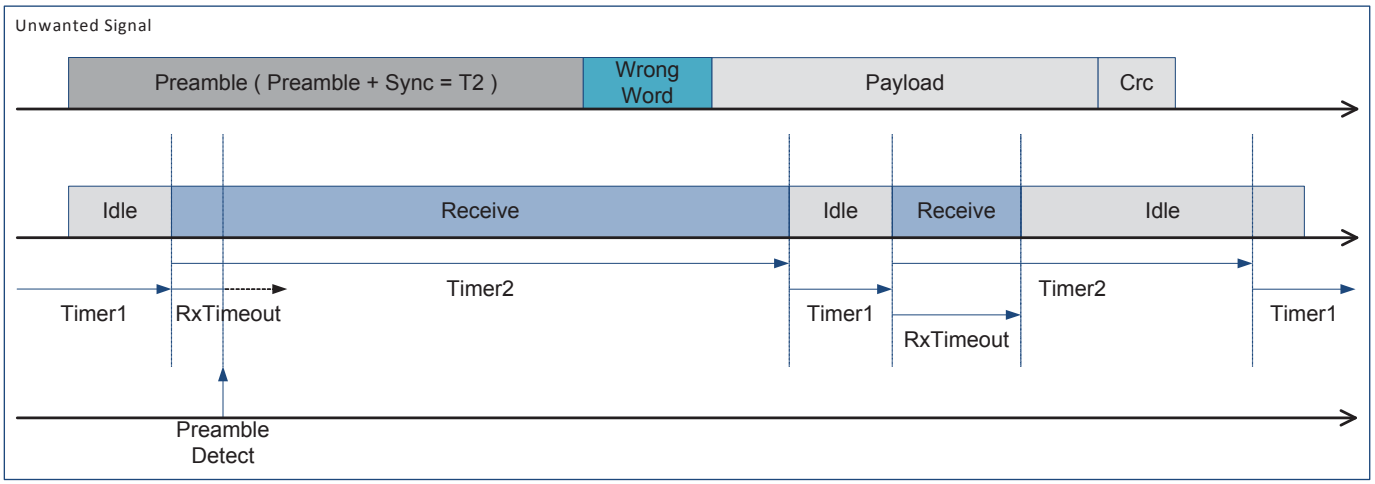


Figure 49. Listen Mode with Preamble Received and no SyncAddress

But if a valid Sync Word is detected, a *SyncAddress* interrupt is fired, the Sequencer is switched off and the circuit stays in Receive mode as long as the user doesn't switch modes.

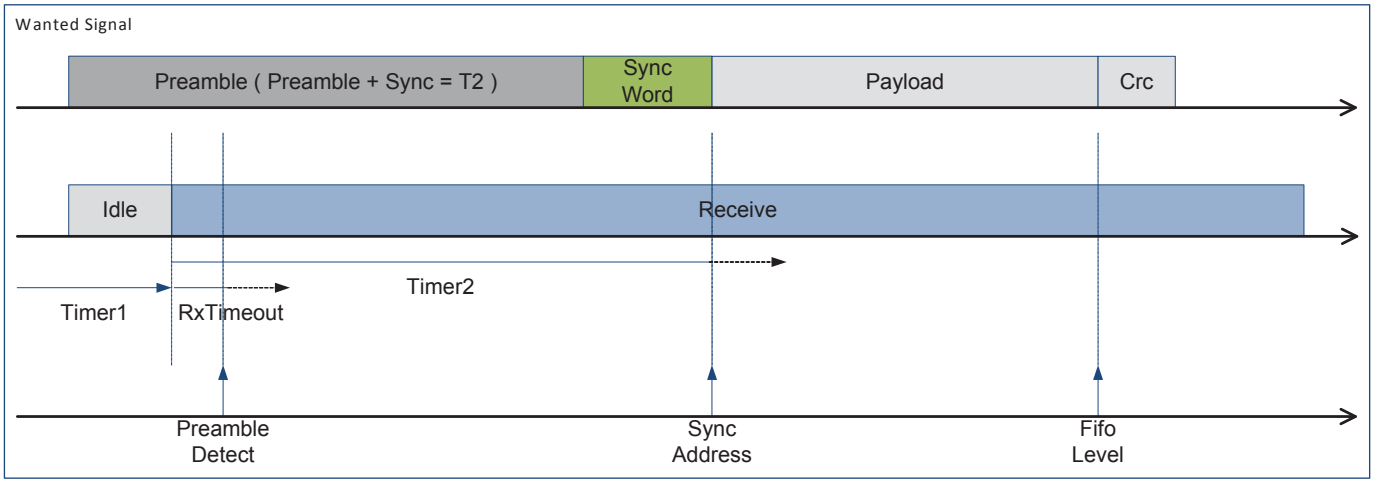


Figure 50. Listen Mode with Preamble Received & Valid SyncAddress

7.3.2.2. Sequencer Configuration

The following graph shows Listen mode - Wake on SyncAddress state machine:

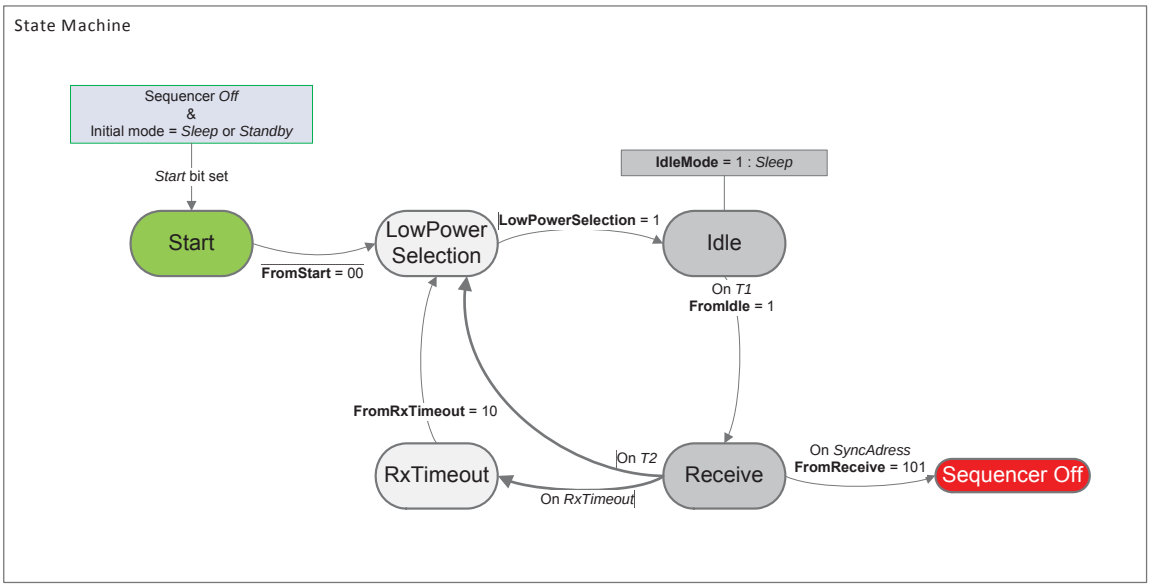


Figure 51. Wake On SyncAddress State Machine

This example configuration is achieved as follows:

Table 48 Listen Mode with SyncAddress Condition Settings

Variable	Effect
IdleMode	1: **Sleep** mode
FromStart	00: To **LowPowerSelection**
LowPowerSelection	1: To **Idle** state
FromIdle	1: To **Receive** state on *T1* interrupt
FromReceive	101: To **Sequencer off** on *SyncAddress* interrupt
FromRxTimeout	10: To **LowPowerSelection**

$T_{TimeoutRxPreamble}$ should be set to just long enough to catch a preamble (depends on *PreambleDetectSize* and *BitRate*).

T_{Timer1} should be set to 64 µs (shortest possible duration).

T_{Timer2} is set so that $T_{Timer1} + T_{Timer2}$ defines the time between two start of reception.

In order to insure packet detection and optimize the receiver power consumption, the received packet Preamble should be defined so that $T_{Preamble} = T_{Timer2} - T_{SyncAddress}$ with $T_{SyncAddress} = (SyncSize + 1)*8/BitRate$.

An example of DIO configuration for this mode is described in the following table:

Table 49 Listen Mode with PreambleDetect Condition Recommended DIO Mapping

DIO	Value	Description
0	01	CrcOk
1	00	FifoLevel
2	11	SyncAddress
3	00	FifoEmpty
4	11	PreambleDetect – Note: *MapPreambleDetect* bit should be set.

7.4. Top Sequencer: Beacon Mode

In this mode, a repetitive message is transmitted periodically. If the Payload being sent is always identical, and *PayloadLength* is smaller than the FIFO size, the use of the *BeaconOn* bit in *RegPacketConfig2* together with the Sequencer permit to achieve periodic beacon without any user intervention.

7.4.1. Timing diagram

In this mode, the Radio is switched to Transmit mode every $T_{Timer1} + T_{Timer2}$ and back to Idle mode after *PacketSent*, as shown in the diagram below. The Sequencer insures minimal time is spent in Transmit mode, and therefore power consumption is optimized.

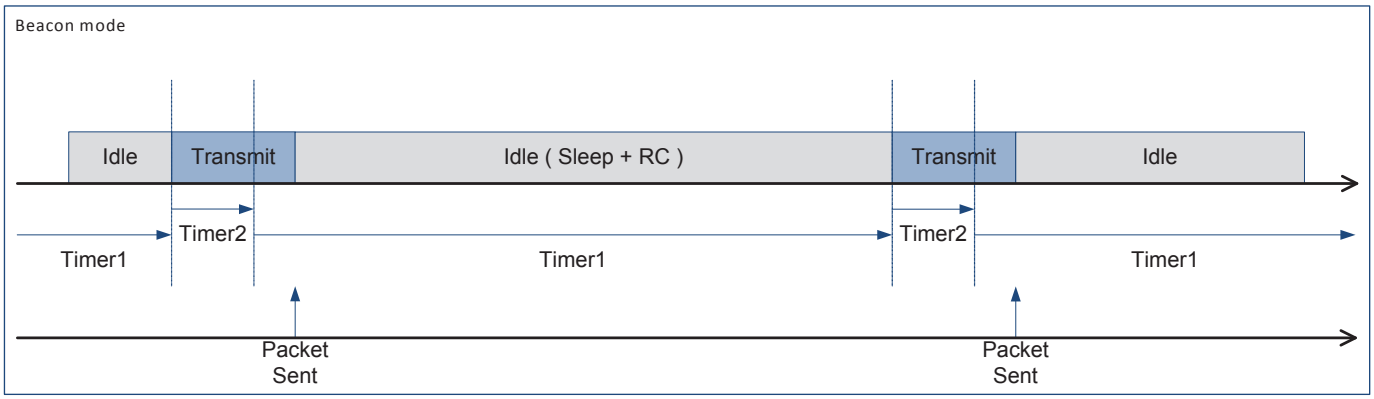


Figure 52. Beacon Mode Timing Diagram

7.4.2. Sequencer Configuration

The Beacon mode state machine is presented in the following graph. It is noticeable that the sequencer enters an infinite loop and can only be stopped by setting *SequencerStop* bit in *RegSeqConfig1*.

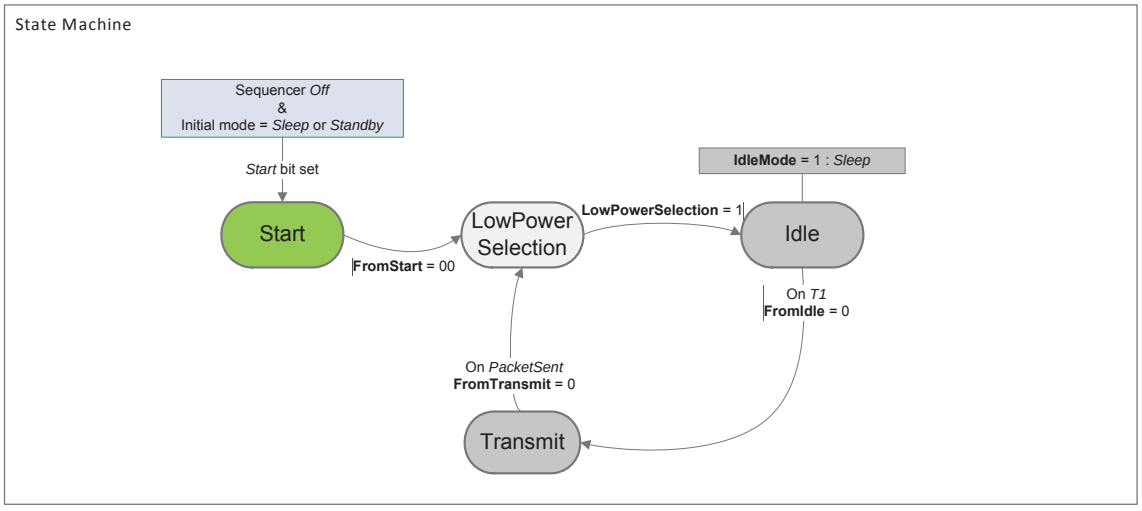


Figure 53. Beacon Mode State Machine

This example is achieved by programming the Sequencer as follows:

Table 50 Beacon Mode Settings

Variable	Effect
IdleMode	1: **Sleep** mode
FromStart	00: To **LowPowerSelection**
LowPowerSelection	1: To **Idle** state
FromIdle	0: To **Transmit** state on *T1* interrupt
FromTransmit	0: To **LowPowerSelection** on *PacketSent* interrupt

$T_{Timer1} + T_{Timer2}$ define the time between the start of two transmissions.

7.5. Example CRC Calculation

The following routine(s) may be implemented to mimic the CRC calculation of the SX1276/77/78/79:

```
// CRC types
#define CRC_TYPE_CCITT                              0
#define CRC_TYPE_IBM                                1

// Polynomial = X^16 + X^12 + X^5 + 1
#define POLYNOMIAL_CCITT                            0x1021
// Polynomial = X^16 + X^15 + X^2 + 1
#define POLYNOMIAL_IBM                              0x8005

// Seeds
#define CRC_IBM_SEED                                0xFFFF
#define CRC_CCITT_SEED                              0x1D0F

/*
 * CRC algorithm implementation
 *
 * \param[IN] crc Previous CRC value
 * \param[IN] data New data to be added to the CRC
 * \param[IN] polynomial CRC polynomial selection [CRC_TYPE_CCITT, CRC_TYPE_IBM]
 *
 * \retval crc New computed CRC
 */
U16 ComputeCrc( U16 crc, U8 data, U16 polynomial )
{
    U8 i;
    for( i = 0; i < 8; i++ )
    {
        if( ( ( ( crc & 0x8000 ) >> 8 ) ^ ( data & 0x80 ) ) != 0 )
        {
            crc <<= 1;                  // shift left once
            crc ^= polynomial;          // XOR with polynomial
        }
        else
        {
            crc <<= 1;                  // shift left once
        }
        data <<= 1;                     // Next data bit
    }
    return crc;
}

/*
 * CRC algorithm implementation
 *
 * \param[IN] buffer Array containing the data
 * \param[IN] bufferLength Buffer length
 * \param[IN] crcType Selects the CRC polynomial[CRC_TYPE_CCITT, CRC_TYPE_IBM]
 *
 * \retval crc Buffer computed CRC
 */
U16 RadioPacketComputeCrc( U8 *buffer, U8 bufferLength, U8 crcType )
{
    U8 i;
    U16 crc;
    U16 polynomial;

    polynomial = ( crcType == CRC_TYPE_IBM ) ? POLYNOMIAL_IBM : POLYNOMIAL_CCITT;
    crc = ( crcType == CRC_TYPE_IBM ) ? CRC_IBM_SEED : CRC_CCITT_SEED;

    for( i = 0; i < bufferLength; i++ )
    {
        crc = ComputeCrc( crc, buffer[i], polynomial );
    }

    if( crcType == CRC_TYPE_IBM )
    {
        return crc;
    }
    else
    {
        return ( U16 )( ~crc );
    }
}
```

Figure 54. Example CRC Code

7.6. Example Temperature Reading

The following routine(s) may be implemented to read the temperature and calibrate the sensor:

Temperature.c

```
/*!
 * Reads the raw temperature
 * \retval temperature New raw temperature reading in 2's complement format
 */
int8 RadioGetRawTemp( void )
{
    int8_t temp = 0;
    uint8_t previousOpMode;

    // Save current Operation Mode
    SX1276Read( REG_OPMODE, &SX1276->RegOpMode );
    previousOpMode = SX1276->RegOpMode;

    // Pass through LoRa sleep only necessary if reading temperature while in LoRa Mode
    if( ( previousOpMode & RFLR_OPMODE_LONGRANGEMODE_ON ) == RFLR_OPMODE_LONGRANGEMODE_ON )
    {
        SX1276->RegOpMode = RFLR_OPMODE_SLEEP;
        SX1276Write( REG_OPMODE, SX1276->RegOpMode );   // put device in LoRa Sleep Mode
    }

    // Put device in FSK Sleep Mode
    SX1276->RegOpMode = RF_OPMODE_SLEEP;
    SX1276Write( REG_OPMODE, SX1276->RegOpMode );
    // Put device in FSK RxSynth
    SX1276->RegOpMode = RF_OPMODE_SYNTHESIZER_RX;
    SX1276Write( REG_OPMODE, SX1276->RegOpMode );
    // Enable Temperature reading
    SX1276Read( REG_IMAGECAL, &SX1276->RegImageCal );
    SX1276->RegImageCal = ( SX1276->RegImageCal & RF_IMAGECAL_TEMPMONITOR_MASK ) | RF_IMAGECAL_TEMPMONITOR_ON;
    SX1276Write( REG_IMAGECAL, SX1276->RegImageCal );

    // Wait 150us
    Delay( 150 );

    // Disable Temperature reading
    SX1276Read( REG_IMAGECAL, &SX1276->RegImageCal );
    SX1276->RegImageCal = ( SX1276->RegImageCal & RF_IMAGECAL_TEMPMONITOR_MASK ) | RF_IMAGECAL_TEMPMONITOR_OFF;
    SX1276Write( REG_IMAGECAL, SX1276->RegImageCal );

    // Put device in FSK Sleep Mode
    SX1276->RegOpMode = RF_OPMODE_SLEEP;
    SX1276Write( REG_OPMODE, SX1276->RegOpMode );

    // Read temperature
    SX1276Read( REG_TEMP, &SX1276->RegTemp );

    if( ( SX1276->RegTemp & 0x80 ) == 0x80 )
    {
        temp = 255 - SX1276->RegTemp;
    }
    else
    {
        temp = SX1276->RegTemp;
        temp *= -1;
    }
    // We were in LoRa Mode prior to the temperature reading
    if( ( previousOpMode & RFLR_OPMODE_LONGRANGEMODE_ON ) == RFLR_OPMODE_LONGRANGEMODE_ON )
    {
        SX1276->RegOpMode = RFLR_OPMODE_SLEEP;
        SX1276Write( REG_OPMODE, SX1276->RegOpMode );   // put device in LoRa Sleep Mode
    }

    // Reload previous Op Mode
    SX1276Write( REG_OPMODE, previousOpMode );
    return temp;
}
```

Figure 55. Example Temperature Reading

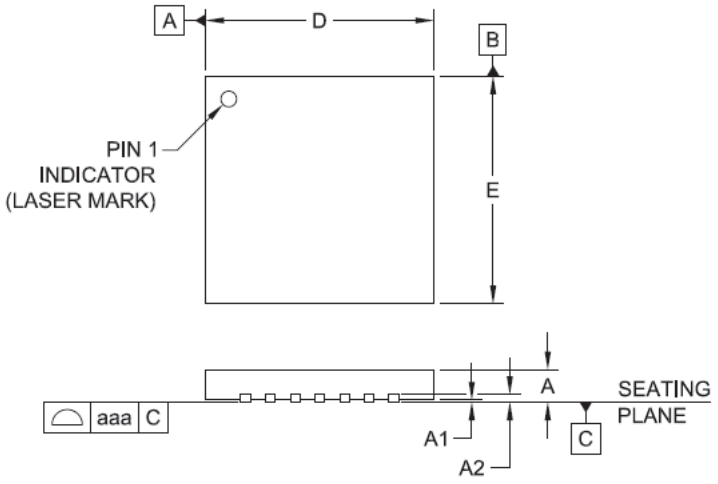
```
68
69  /*!
70   * Computes the temperature compensation factor
71   * \param [IN] actualTemp Actual temperature measured by an external device
72   * \retval compensationFactor Computed compensation factor
73   */
74  S8 RadioCalibreateTemp( S8 actualTemp )
75  {
76      return actualTemp - RadioGetRawTemp( );
77  }
78
79  /*!
80   * Gets the actual compensated temperature
81   * \param [IN] compensationFactor Return value of the calibration function
82   * \retval New compensated temperature value
83   */
84  S8 RadioGetTemp( S8 compensationFactor )
85  {
86      return RadioGetRawTemp( ) + compensationFactor;
87  }
88
89  /*!
90   * Usage example
91   */
92  void main( void )
93  {
94      S8 temp;
95      S8 actualTemp = 0;
96      S8 compensationFactor = 0;
97
98      // Ask user for the temperature during calibration
99      actualTemp = AskUserTemperature( );
100     compensationFactor = RadioCalibreateTemp( actualTemp );
101
102     while( True )
103     {
104         temp = RadioGetTemp( compensationFactor );
105     }
106 }
```

Figure 56. Example Temperature Reading (continued)

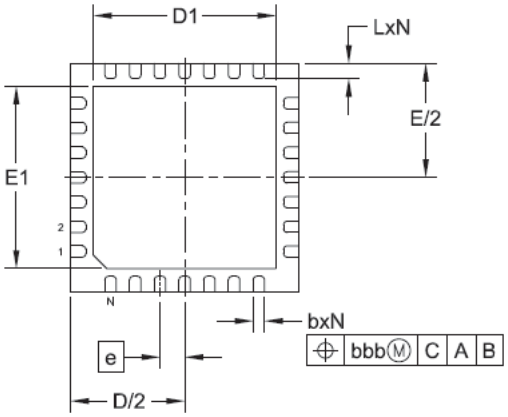
8. Packaging Information

8.1. Package Outline Drawing

The SX1276/77/78/79 is available in a 28-lead QFN package as shown in Figure 57.



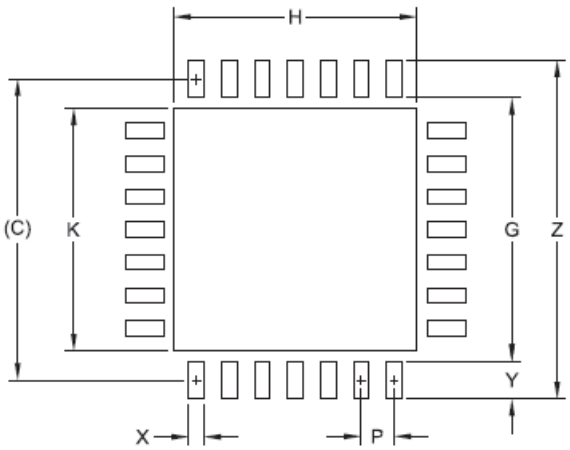
DIMENSIONS			
DIM	MILLIMETERS		
	MIN	NOM	MAX
A	0.80	—	1.00
A1	0.00	—	0.05
A2		(0.20)	
b	0.25	0.30	0.35
D	5.90	6.00	6.10
D1	4.65	4.80	4.90
E	5.90	6.00	6.10
E1	4.65	4.80	4.90
e		0.65 BSC	
L	0.30	0.40	0.50
N		28	
aaa		0.08	
bbb		0.10	



NOTES:
1. CONTROLLING DIMENSIONS ARE IN MILLIMETERS (ANGLES IN DEGREES).
2. COPLANARITY APPLIES TO THE EXPOSED PAD AS WELL AS THE TERMINALS.

Figure 57. Package Outline Drawing

8.2. Recommended Land Pattern



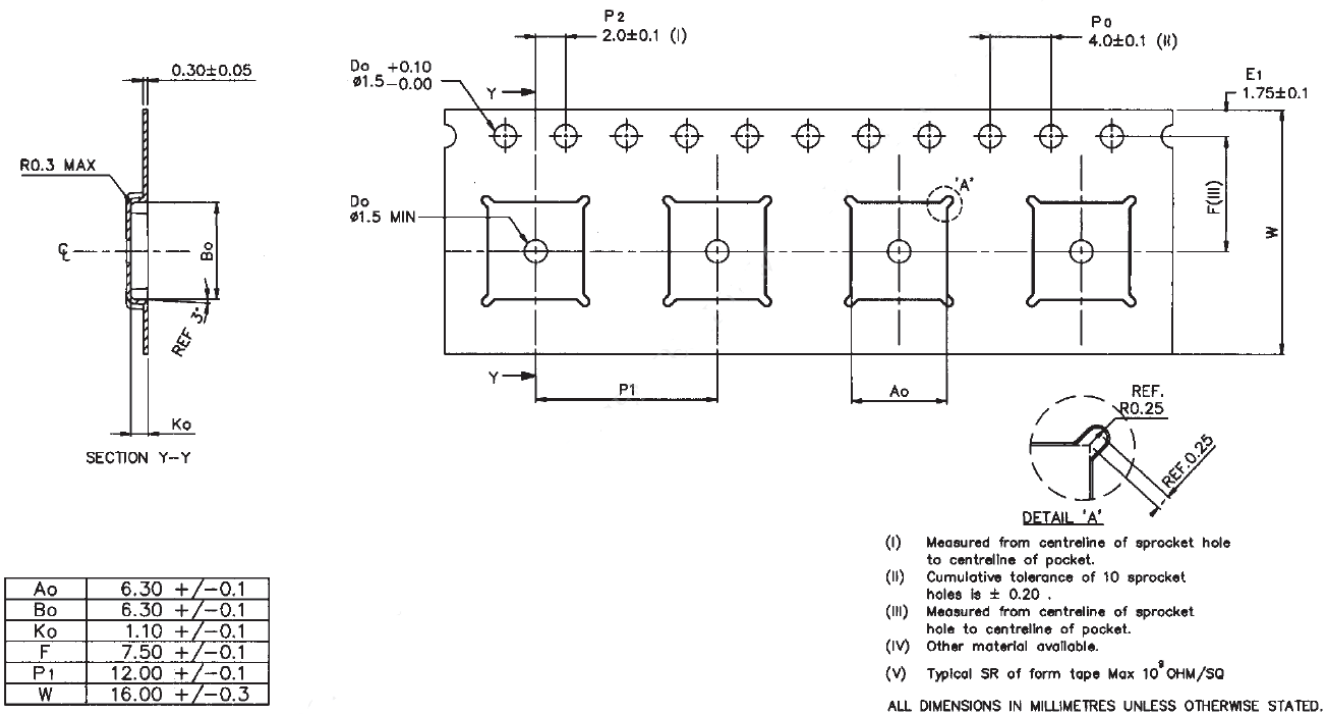
DIMENSIONS	
DIM	MILLIMETERS
C	(5,95)
G	5.20
H	4.80
K	4.80
P	0.65
X	0.30
Y	0.75
Z	6.70

NOTES:

1. CONTROLLING DIMENSIONS ARE IN MILLIMETERS (ANGLES IN DEGREES).
2. THIS LAND PATTERN IS FOR REFERENCE PURPOSE ONLY.
   CONSULT YOUR MANUFACTURING GROUP TO ENSURE YOUR COMPANY'S
   MANUFACTURING GUIDELINES ARE MET.
3. THERMAL VIAS IN THE LAND PATTERN OF THE EXPOSED PAD SHALL BE CONNECTED
   TO A SYSTEM GROUND PLANE. FAILURE TO DO SO MAY COMPROMISE THE THERMAL
   AND/OR FUNCTIONAL PERFORMANCE OF THE DEVICE.
4. SQUARE PACKAGE - DIMENSIONS APPLY IN BOTH " X " AND " Y " DIRECTIONS.

Figure 58. Recommended Land Pattern

8.3. Tape & Reel Information



Ao	6.30 +/−0.1
Bo	6.30 +/−0.1
Ko	1.10 +/−0.1
F	7.50 +/−0.1
P1	12.00 +/−0.1
W	16.00 +/−0.3

Figure 59. Tape and Reel Information

8.4. Wafer Delivery

Please get in touch with a Semtech representative for wafer delivery option.

Semtech will deliver probed wafers with a die picking map.

The specifications given in *Section 2.5. "Chip Specification" on page 14* are determined from the reference design with QFN 28 package (see the reference design available on the Semtech website). In order to fulfill these specifications, the die assembly must be done with similar wire constraints as in the QFN 28 package, see Addendum2 in the SX1276WS devivery specification.

A complete wafer delivery specification is available on request.

9. Revision History

Table 51 Revision History

Revision	Date	Comment
1	Sept 2013	First FINAL release
2	Nov 2014	Miscellaneous typographical corrections Correction of *RxPayloadCrcOn* description Improve description in the RSSI and IQ calibration mechanism Correction of ToA formulae Inclusion of FEI and automatic frequency correction for LoRa Corrected Rssi Formula in Lora mode
3	Nov 2014	Addition of part SX1279
4	March 2015	Clarified operation modes for Rx Single and Rx Continuous mode in LoRa Added use cases for Rx Single and Rx Continuous mode in LoRa mode Clarified used of LoRa RxPayloadCrcOn in Register Table Added description of register RegSyncWord in LoRa register table Changed Stand-By typo into Standby
5	August 2016	Addition of part SX1276WS

Contact information

Semtech Corporation
Wireless,Sensing & Timing Products Division
200 Flynn Road, Camarillo, CA 93012
Phone: (805) 498-2111 Fax: (805) 498-3804
E-mail: sales@semtech.com
support_rf@semtech.com
Internet: http://www.semtech.com

UNIVERSIDADE DA CORUÑA

ESCUELA POLITÉCNICA SUPERIOR

TRABAJO FIN DE MÁSTER

CURSO 2016/2017

DISEÑO Y DESARROLLO DEL SISTEMA DE COMUNICACIONES INALÁMBRICO DE UN ENJAMBRE DE UAV'S COLABORATIVOS

MÁSTER EN INGENIERÍA INDUSTRIAL

DOCUMENTO DEL PROYECTO

ANEXO VI – GUÍAS DE USUARIO

RN2483 TECHNOLOGY MODULE COMMAND REFERENCE USER'S GUIDE

RN2483 LoRa® Technology Module Command Reference User's Guide

Note the following details of the code protection feature on Microchip devices:

- Microchip products meet the specification contained in their particular Microchip Data Sheet.
- Microchip believes that its family of products is one of the most secure families of its kind on the market today, when used in the intended manner and under normal conditions.
- There are dishonest and possibly illegal methods used to breach the code protection feature. All of these methods, to our knowledge, require using the Microchip products in a manner outside the operating specifications contained in Microchip's Data Sheets. Most likely, the person doing so is engaged in theft of intellectual property.
- Microchip is willing to work with the customer who is concerned about the integrity of their code.
- Neither Microchip nor any other semiconductor manufacturer can guarantee the security of their code. Code protection does not mean that we are guaranteeing the product as "unbreakable."

Code protection is constantly evolving. We at Microchip are committed to continuously improving the code protection features of our products. Attempts to break Microchip's code protection feature may be a violation of the Digital Millennium Copyright Act. If such acts allow unauthorized access to your software or other copyrighted work, you may have a right to sue for relief under that Act.

Microchip received ISO/TS-16949:2009 certification for its worldwide headquarters, design and wafer fabrication facilities in Chandler and Tempe, Arizona; Gresham, Oregon and design centers in California and India. The Company's quality system processes and procedures are for its PIC® MCUs and dsPIC® DSCs, KEELOQ® code hopping devices, Serial EEPROMs, microperipherals, nonvolatile memory and analog products. In addition, Microchip's quality system for the design and manufacture of development systems is ISO 9001:2000 certified.

QUALITY MANAGEMENT SYSTEM
CERTIFIED BY DNV
═ ISO/TS 16949 ═

Trademarks

ISBN: 978-1-5224-1484-1

Table of Contents

NOTES:

Preface

NOTICE TO CUSTOMERS

All documentation becomes dated, and this manual is no exception. Microchip tools and documentation are constantly evolving to meet customer needs, so some actual dialogs and/or tool descriptions may differ from those in this document. Please refer to our website (www.microchip.com) to obtain the latest documentation available.

Documents are identified with a "DS" number. This number is located on the bottom of each page, in front of the page number. The numbering convention for the DS number is "DSXXXXXA", where "XXXXX" is the document number and "A" is the revision level of the document.

For the most up-to-date information on development tools, see the MPLAB® IDE online help. Select the Help menu, and then Topics to open a list of available online help files.

INTRODUCTION

This chapter contains general information that will be useful to know before using the RN2483 module. Topics discussed in this chapter include:

- Document Layout
- Conventions Used in this Guide
- Recommended Reading
- The Microchip Website
- Development Systems Customer Change Notification Service
- Customer Support
- Revision History

DOCUMENT LAYOUT

This command reference user's guide provides information for configuring the RN2483 low-power long-range LoRa® technology transceiver module, including a description of communication and command references. The document is organized as follows:

- **Chapter 1. "Introduction"** – This chapter introduces the RN2483 module and provides a brief overview of its features.
- **Chapter 2. "Command Reference"** – This chapter provides information on the commands used to configure the RN2483 module with examples.
- **Chapter 3. "Bootloader Usage"** - This chapter gives further information on the bootloader usage and protocol commands.
- **Appendix A. "Current Firmware Features and Fixes "** – This chapter provides information on the release notes for each revision of the firmware.

CONVENTIONS USED IN THIS GUIDE

This manual uses the following documentation conventions:

DOCUMENTATION CONVENTIONS

Description	Represents	Examples
Arial font:		
Italic characters	Referenced books	*MPLAB® IDE User's Guide*
	Emphasized text	...is the *only* compiler...
Initial caps	A window	the Output window
	A dialog	the Settings dialog
	A menu selection	select Enable Programmer
Quotes	A field name in a window or dialog	"Save project before build"
Underlined, italic text with right angle bracket	A menu path	*File>Save*
Bold characters	A dialog button	Click **OK**
	A tab	Click the **Power** tab
N'Rnnnn	A number in verilog format, where N is the total number of digits, R is the radix and n is a digit.	4'b0010, 2'hF1
Text in angle brackets < >	A key on the keyboard	Press <Enter>, <F1>
Courier New font:		
Plain Courier New	Sample source code	`#define START`
	Filenames	`autoexec.bat`
	File paths	`c:\mcc18\h`
	Keywords	`_asm, _endasm, static`
	Command-line options	`-Opa+, -Opa-`
	Bit values	`0, 1`
	Constants	`0xFF, 'A'`
Italic Courier New	A variable argument	*`file`*`.o`, where *`file`* can be any valid filename
Square brackets [ ]	Optional arguments	`mcc18 [options]` *`file`* `[options]`
Curly brackets and pipe character: { | }	Choice of mutually exclusive arguments; an OR selection	`errorlevel {0|1}`
Ellipses...	Replaces repeated text	`var_name [, var_name...]`
	Represents code supplied by user	`void main (void) { ... }`

RECOMMENDED READING

This command reference user's guide describes how to configure the RN2483 module. The module-specific data sheet contains current information on the module specifications. Other useful documents are listed below. The following documents are available and recommended as supplemental reference resources:

RN2483 Low-Power Long-Range LoRa® Technology Transceiver Module Data Sheet (DS50002346)

This data sheet provides detailed specifications for the RN2483 module.

LoRa® Alliance: LoRaWAN™ Specification

This document describes the LoRaWAN Class A protocol, which is optimized for battery-powered end devices. This specification is available from the LoRa Alliance at http://www.lora-alliance.org.

To obtain any of Microchip's documents, visit the Microchip website at www.microchip.com.

THE MICROCHIP WEBSITE

Microchip provides online support via our website at www.microchip.com. This website is used as a means to make files and information easily available to customers. Accessible by using your favorite Internet browser, the website contains the following information:

- **Product Support** – Data sheets and errata, application notes and sample programs, design resources, user's guides and hardware support documents, latest software releases and archived software
- **General Technical Support** – Frequently Asked Questions (FAQs), technical support requests, online discussion groups, Microchip consultant program member listing
- **Business of Microchip** – Product selector and ordering guides, latest Microchip press releases, listing of seminars and events, listings of Microchip sales offices, distributors and factory representatives

DEVELOPMENT SYSTEMS CUSTOMER CHANGE NOTIFICATION SERVICE

Microchip's customer notification service helps keep customers current on Microchip products. Subscribers will receive e-mail notification whenever there are changes, updates, revisions or errata related to a specified product family or development tool of interest.

To register, access the Microchip website at www.microchip.com, click on Customer Change Notification and follow the registration instructions.

The Development Systems product group categories are:

- **Compilers** – The latest information on Microchip C compilers, assemblers, linkers and other language tools. These include all MPLAB C compilers; all MPLAB assemblers (including MPASM™ assembler); all MPLAB linkers (including MPLINK™ object linker); and all MPLAB librarians (including MPLIB™ object librarian).
- **Emulators** – The latest information on Microchip in-circuit emulators.This includes the MPLAB REAL ICE™ and MPLAB ICE 2000 in-circuit emulators.
- **In-Circuit Debuggers** – The latest information on the Microchip in-circuit debuggers. This includes MPLAB ICD 3 in-circuit debuggers and PICkit™ 3 debug express.
- **MPLAB® IDE** – The latest information on Microchip MPLAB IDE, the Windows® Integrated Development Environment for development systems tools. This list is focused on the MPLAB IDE, MPLAB IDE Project Manager, MPLAB Editor and MPLAB SIM simulator, as well as general editing and debugging features.
- **Programmers** – The latest information on Microchip programmers. These include production programmers such as MPLAB REAL ICE in-circuit emulator, MPLAB ICD 3 in-circuit debugger and MPLAB PM3 device programmers. Also included are non-production development programmers such as PICSTART® Plus and PICkit 2 and 3.

CUSTOMER SUPPORT

Users of Microchip products can receive assistance through several channels:

- Distributor or Representative
- Local Sales Office
- Field Application Engineer (FAE)
- Technical Support

Customers should contact their distributor, representative or field application engineer (FAE) for support. Local sales offices are also available to help customers. A listing of sales offices and locations is included in the back of this document.

Technical support is available through the website at:

http://www.microchip.com/support.

REVISION HISTORY

Revision A (March 2015)

Initial release of the document.

Revision B (March 2015)

Update to Section 1.4.

Revision C (November 2015)

Added 2.3.6.5, 2.3.6.6, 2.3.6.7, 2.4.8.16, 2.4.8.17 sections; Updated 2-4, 2-6, 2-8 and 2-14 Tables, Updated 2.3.5.2, 2.4.4, 2.4.9.7, 2.4.9.18, and 2.5.5.17 sections; Other minor corrections.

Revision D (February 2016)

Added a new Note box in section 2.4.9.2, updated section 2.4.9.16 and Figure 2-1, added A.3 section; Other minor corrections.

Revision E (February 2016)

Removed Version 1.0.2 in section A.4; Other minor corrections.

Revision F (March 2017)

Added Chapter 3 (Bootloader Usage); Other minor corrections.

NOTES:

Chapter 1. Introduction

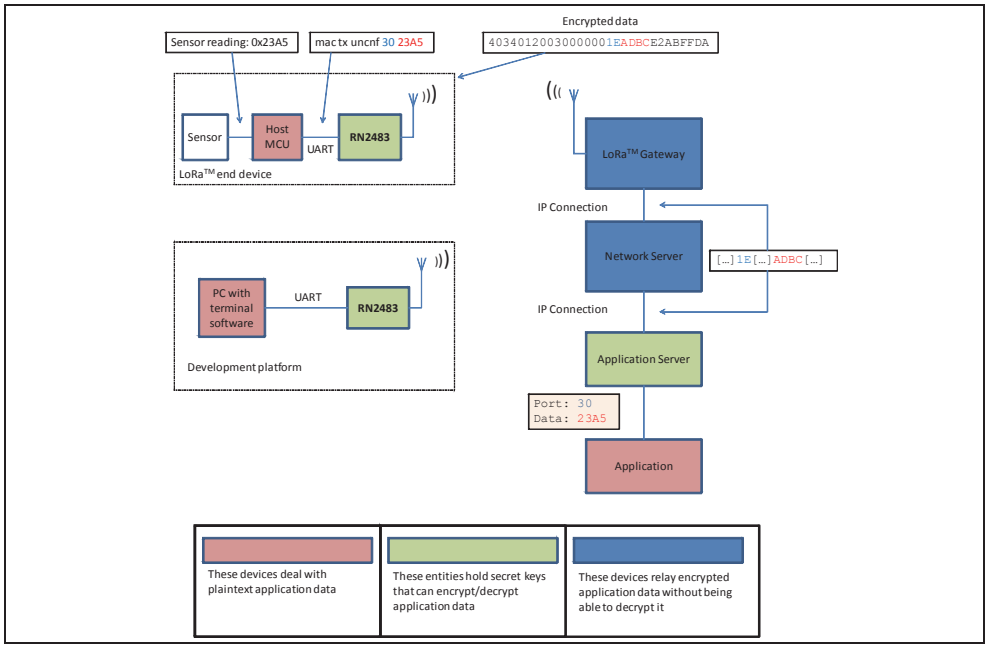
1.1 OVERVIEW

The Microchip RN2483 module provides LoRaWAN™ protocol connectivity using a simple UART interface. This module handles the LoRaWAN Class A protocol and provides an optimized text command/response interface to the host system. This document is intended to describe an implementation of the LoRaWAN Class A protocol. LoRaWAN protocol terms are described in more detail in the *LoRaWAN Specification* available from the LoRa Alliance (http://www.lora-alliance.org). Thus, it is recommended to review the *LoRaWAN Specification* before using the RN2483 module.

The required configuration for accessing a LoRa technology network is minimal and can be stored in the module's EEPROM, allowing for factory configuration of these parameters, lowering the requirements for the host system while also increasing system security. The module also features GPIO pins that can be configured through the UART interface.

A simple use case is described in Figure 1-1 where an end device, containing a host MCU which reads a sensor, commands the RN2483 to transmit the sensor reading over the LoRa network. Data are encrypted by the RN2483 and the radio packet is received by one or multiple gateways which forward it to the network server. The network server sends the data to the application server which has the key to decrypt the application data. Similarly, a development platform may consist of an RN2483 directly connected over UART to a PC which becomes the host system in this case. Users can then type commands into the module using a terminal program.

FIGURE 1-1: SIMPLE LoRa® TECHNOLOGY NETWORK DIAGRAM



The flow of data can be followed as it gets generated by an end device and transported on the network.

1.2 FEATURES

- LoRaWAN Class A protocol compliance
- Integrated FSK, GFSK and LoRa technology transceiver allowing the user to transmit custom packets using these protocols
- Globally unique 64-bit identifier (EUI-64™)
- Configurable GPIOs
- Intelligent Low-Power mode with programmable/on-demand wake-up
- Bootloader for firmware upgrade
- All configuration and control done over UART using simple ASCII commands

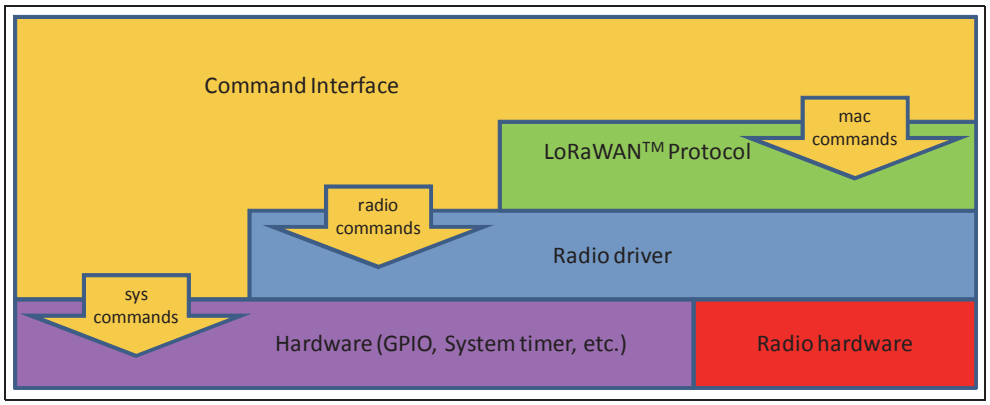
Refer to the *RN2483, Low-Power Long-Range LoRa® Technology Transceiver Module Data Sheet* (DS50002346) for details on the hardware specifications of the module.

1.3 CONFIGURATION

The RN2483 module's architecture is described in Figure 1-2 from the command interface point of view. There are three types of commands that can be used, and each allows access to different module functions:

- LoRaWAN Class A configuration and control, using the `mac` group of commands
- Radio configuration and control, using the `radio` group of commands
- Other module functions, using the `sys` group of commands

FIGURE 1-2: RN2483 COMMAND INTERFACE (YELLOW) AND ITS RELATIONSHIP TO THE MODULE'S INTERNAL COMPONENTS



The available commands can be used to configure and control the LoRaWAN protocol layer, the radio driver and some system peripherals.

In order to communicate with a LoRa network, a specific number of parameters need to be configured. Since two distinctive methods are offered for a device to become part of the network, each of these requires different parameters:

- Over-the-Air Activation (OTAA), where a device negotiates network encryption keys at the time it joins the network. For this, the device EUI, application EUI and application key need to be configured and then the OTAA procedure can start.
- Activation by Personalization (ABP) where the device already contains the network keys and can directly start communication with the network. Configuring the device address, network session key and application session key is sufficient for this type of initialization.

For increased security, these parameters can be configured and stored in the module's EEPROM during manufacturing of devices requiring LoRaWAN connectivity. Thus, the keys do not need to be sent over the UART interface by the host system every time the device powers up.

1.4 UART INTERFACE

All of the RN2483 module's settings and commands are transmitted over UART using the ASCII interface.

All commands need to be terminated with <CR><LF> and any replies they generate will also be terminated by the same sequence.

The default settings for the UART interface are 57600 bps, 8 bits, no parity, 1 Stop bit, no flow control. The baud rate can be changed by triggering the auto-baud detection sequence of the module. To do this, the host system needs to transmit to the module a break condition followed by a 0x55 character at the new baud rate. The auto-baud detection mechanism can also be triggered during Sleep to wake the module up before the predetermined time has expired.

Note: A break condition is signaled to the module by keeping the UART_RX pin low for longer than the time to transmit a complete character. For example, at the default baud rate of 57600 bps keeping the UART_RX pin low for 938 μs is a valid break condition, whereas at 9600 bps this would be interpreted as a 0x00 character. Thus, the break condition needs to be long enough to still be interpreted as such at the baud rate that is currently in use.

NOTES:

Chapter 2. Command Reference

The RN2483 LoRa technology module supports a variety of commands for configuration. This section describes these commands in detail and provides examples.

2.1 COMMAND SYNTAX

To issue commands to the RN2483 module, the user sends keywords followed by optional parameters. Commands (keywords) are case sensitive, and spaces must not be used in parameters. Hex input data can be uppercase or lowercase. String text data, such as `OTAA` used for the join procedure, is case-insensitive.

The use of shorthand for parameters is *NOT* supported.

Depending on the command, the parameter may expect values in either decimal or hexadecimal form; refer to the command description for the expected form. For example, when configuring the frequency, the command expects a decimal value in Hertz such as `868100000` (868.1 MHz). Alternatively, when configuring the LoRaWAN device address, the hex value is entered into the parameter as `aabbccdd`. To enter a number in hex form, use the value directly. For example, the hex value `0xFF` would be entered as `FF`.

2.2 COMMAND ORGANIZATION

There are three general command categories, as shown in Table 2-1.

TABLE 2-1: COMMAND TYPES

Command Type	Keyword	Description
System	`<sys>`	Issues system level behavior actions, gathers status information on the firmware and hardware version, or accesses the module user EEPROM memory.
LoRaWAN™ Class A Protocol	`<mac>`	Issues LoRaWAN Class A protocol network communication behaviors, actions and configurations commands.
Transceiver commands	`<radio>`	Issues radio specific configurations, directly accessing and updating the transceiver setup.

Once the LoRaWAN Class A protocol configuration is complete, the user must save the settings to store the configuration data, otherwise it will not take effect upon reboot or Reset.

Note: Upon successful reception of commands, the module will respond with one of the following:

- `ok`
- `invalid_param`
- Requested Information
- Descriptive Error Message

Note: To facilitate the sharing of the radio between user custom applications and the LoRaWAN MAC, please refer to the `mac pause` and `mac resume` commands. Since no sharing exists between `sys` and other types of commands, there is no need for additional `pause` commands.

2.3 SYSTEM COMMANDS

System commands begin with the system keyword `<sys>` and include the categories shown in Table 2-2, Table 2-3 and Table 2-4.

TABLE 2-2: SYSTEM COMMANDS

Parameter	Description
`sleep`	Puts the system in Sleep for a finite number of milliseconds.
`reset`	Resets and restarts the RN2483 module.
`eraseFW`	Deletes the current RN2483 module application firmware and prepares it for firmware upgrade. The RN2483 module bootloader is ready to receive new firmware.
`factoryRESET`	Resets the RN2483 module's configuration data and user EEPROM to factory default values and restarts the RN2483 module.
`set`(1)	Sets specified system parameter values.
`get`(1)	Gets specified system parameter values.

Note 1: Refer to Table 2-3 for system `<set>` and Table 2-4 for system `<get>` command summaries.

2.3.1 sys sleep <length>

`<length>`: decimal number representing the number of milliseconds the system is put to Sleep, from 100 to 4294967296.

Response: `ok` after the system gets back from Sleep mode

`invalid_param` if the length is not valid

This command puts the system to Sleep for the specified number of milliseconds. The module can be forced to exit from Sleep by sending a break condition followed by a `0x55` character at the new baud rate. Note that the break condition needs to be long enough not to be interpreted as a valid character at the current baud rate.

Example: **`sys sleep 120`** // Puts the system to Sleep for 120 ms.

2.3.2 sys reset

Response: `RN2483 X.Y.Z MMM DD YYYY HH:MM:SS`, where X.Y.Z is firmware version, MMM is month, DD is day, HH:MM:SS is hour, minutes, seconds (format: [HW] [FW] [Date] [Time]). [Date] and [Time] refer to the release of the firmware.

This command resets and restarts the RN2483 module; stored internal configurations will be loaded automatically upon reboot.

Example: **`sys reset`** // Resets and restarts the RN2483 module.

2.3.3 sys eraseFW

Response: no response

This command deletes the current RN2483 module application firmware and prepares it for firmware upgrade. The RN2483 module bootloader is ready to receive new firmware.

Example: **`sys eraseFW`** // Deletes the current RN2483 module application firmware.

2.3.4 sys factoryRESET

Response: RN2483 X.Y.Z MMM DD YYYY HH:MM:SS, where X.Y.Z is firmware version, MMM is month, DD is day, HH:MM:SS is hour, minutes, seconds (format: [HW] [FW] [Date] [Time]). [Date] and [Time] refer to the release of the firmware.

This command resets the module's configuration data and user EEPROM to factory default values and restarts the module. After factoryRESET, the RN2483 module will automatically reset and all configuration parameters are restored to factory default values.

Example: **sys factoryRESET** // Restores factory default values.

2.3.5 System Set Commands

TABLE 2-3: SYSTEM SET COMMANDS

Parameter	Description
nvm	Stores <data> to a location <address> of user EEPROM.
pindig	Allows user to set and clear available digital pins.
pinmode	Allows user to set the functionality of a pin to either digital input, digital output or analog input (if available).

2.3.5.1 sys set nvm <address> <data>

<address>: hexadecimal number representing user EEPROM address, from 300 to 3FF

<data>: hexadecimal number representing data, from 00 to FF

Response: ok if the parameters (address and data) are valid

invalid_param if the parameters (address and data) are not valid

This command allows the user to modify the user EEPROM at <address> with the value supplied by <data>. Both <address> and <data> must be entered as hex values. The user EEPROM memory is located inside the MCU on the module.

Example: **sys set nvm 300 A5** // Stores the value 0xA5 at user EEPROM address 0x300.

2.3.5.2 sys set pinmode <pinname> <pinFunc>

<pinname>: string representing the pin. Parameters can be: GPIO0 - GPIO13, UART_CTS, UART_RTS, TEST0, TEST1

<pinFunc>: string representing the function of the pin. Parameters can be: digout, digin or ana.

Response: ok if the parameters are valid

invalid_param if the parameters are not valid

This command allows the user to configure the function on a pin. A pin can be configured as digital output by using the digout parameter. A pin can be configured as digital input by using the digin parameter. A pin can be configured as analog input by using the ana parameter.

Note: Not all pins have analog input functionality.

Example: `sys set pinmode GPIO0 ana` //Configures GPIO0 as analog input

> **Note:** This command must be called prior to reading or setting the value of a pin in order to have correct behavior.

2.3.5.3 `sys set pindig <pinname> <pinstate>`

`<pinname>`: string representing the pin. Parameter values can be: `GPIO0 - GPIO13`, `UART_CTS`, `UART_RTS`, `TEST0`, `TEST1`

`<pinstate>`: decimal number representing the state. Parameter values can be: 0 or 1.

Response: `ok` if the parameters (`<pinname>`, `<pinstate>`) are valid

`invalid_param` if the parameters (`<pinname>`, `<pinstate>`) are not valid

This command allows the user to modify the unused pins available for use by the module. The selected `<pinname>` is driven high or low depending on the desired `<pinstate>`.

Default: `GPIO0-GPIO13`, `UART_CTS`, `UART_RTS`, `TEST0` and `TEST1` are driven low (value 0).

Example: **`sys set pindig GPIO5 1`** // Drives GPIO5 high 1, VDD.

> **Note:** In order for the pin to be driven to a value, make sure you have first configured the pin to be a digital output using the command `sys set pinmode <pinname> digout`.

2.3.6 System Get Commands

TABLE 2-4: SYSTEM GET COMMANDS

Parameter	Description
`ver`	Returns the information on hardware platform, firmware version, release date.
`nvm`	Returns data from the requested user EEPROM `<address>`.
`vdd`	Returns measured voltage in mV.
`hweui`	Returns the preprogrammed EUI node address.
`pindig`	Returns the state of a digital input.
`pinana`	Returns the state of an analog input.

2.3.6.1 `sys get ver`

Response: `RN2483 X.Y.Z MMM DD YYYY HH:MM:SS`, where X.Y.Z is firmware version, MMM is month, DD is day, HH:MM:SS is hour, minutes, seconds (format: [HW] [FW] [Date] [Time]). [Date] and [Time] refer to the release of the firmware.

This command returns the information related to the hardware platform, firmware version, release date and time stamp on firmware creation.

Example: **`sys get ver`** // Returns version-related information.

2.3.6.2 sys get nvm <address>

<address>: hexadecimal number representing user EEPROM address, from 300 to 3FF

Response: 00 – FF (hexadecimal value from 00 to FF) if the address is valid

invalid_param if the address is not valid

This command returns the data stored in the user EEPROM of the RN2483 module at the requested <address> location.

Example: **sys get nvm 300** // Returns the 8-bit hex value stored at 300.

2.3.6.3 sys get vdd

Response: 0–3600 (decimal value from 0 to 3600)

This command informs the RN2483 module to do an ADC conversion on the VDD. The measurement is converted and returned as a voltage (mV).

Example: **sys get vdd** // Returns mV measured on the VDD module.

2.3.6.4 sys get hweui

Response: hexadecimal number representing the preprogrammed EUI node address

This command reads the preprogrammed EUI node address from the RN2483 module. The value returned by this command is a globally unique number provided by Microchip.

Example: **sys get hweui** // Reads the preprogrammed EUI node address.

> **Note:** The preprogrammed EUI node address is a read-only value and cannot be changed or erased. This value can be used to configure the device EUI using the mac set deveui command (see Section 2.4.8.2).

2.3.6.5 sys get pindig <pinname>

<pinname>: string representing the pin. Parameters can be: GPIO0 – GPIO13, UART_CTS, UART_RTS, TEST0, TEST1

Response: decimal number representing the state (either 0 or 1).

This command allows the user to read the state of a digital input. To be used as a digital input, a pin needs to be configured using the sys set pinmode command.

Example: sys get pindig GPIO0 //Reads the state of the GPIO0 digital input

> **Note:** The sys set pinmode <pinname> digin command must be called to configure the function of the pin prior to reading its digital input value.

2.3.6.6 sys get pinana <pinname>

<pinname>: string representing the pin. Parameters can be: GPIO0 - GPIO3, GPIO5 - GPIO13

Response: decimal number representing the result of the conversion, from 0 to 1023, where 0 represents 0V and 1023 is VDD, the supply voltage of the module.

This command allows the user to read the state of an analog input. To be used as an analog input, a pin needs to be configured using the sys set pinmode command.

Example: sys get pinana GPIO0 //Reads the state of the GPIO0 analog input

Note: The `sys set pinmode <pinname> ana` command must be called to configure the function of the pin prior to reading its analog input value.

2.4 MAC COMMANDS

LoRaWAN Class A protocol commands begin with the system keyword `mac` and include the categories shown in Table 2-5 through Table 2-9.

TABLE 2-5: MAC COMMANDS

Parameter	Description
`reset`	Resets the RN2483 module to a specific frequency band.
`tx`	Sends the data string on a specified port number and sets default values for most of the LoRaWAN™ parameters.
`join`	Informs the RN2483 module to join the configured network.
`save`	Saves LoRaWAN Class A configuration parameters to the user EEPROM.
`forceENABLE`	Enables the RN2483 module after the LoRaWAN network server commanded the end device to become silent immediately.
`pause`	Pauses LoRaWAN stack functionality to allow transceiver (radio) configuration.
`resume`	Restores the LoRaWAN stack functionality.
`set`	Accesses and modifies specific MAC related parameters.
`get`	Reads back current MAC related parameters from the module.

2.4.1 `mac reset <band>`

`<band>`: decimal number representing the frequency band, either 868 or 433

Response: `ok` if band is valid

`invalid_param` if band is not valid

This command will automatically reset the software LoRaWAN stack and initialize it with the parameters for the selected band.

Example: **`mac reset 868`** // Sets the default values and selects the 868 default band.

Note: This command will set default values for most of the LoRaWAN™ parameters. Everything set prior to this command will lose its set value.

2.4.2 mac tx <type> <portno> <data>

<type>: string representing the uplink payload type, either cnf or uncnf (cnf – confirmed, uncnf – unconfirmed)

<portno>: decimal number representing the port number, from 1 to 223

<data>: hexadecimal value. The length of <data> bytes capable of being transmitted are dependent upon the set data rate (please refer to the *LoRaWAN™ Specification* for further details).

Response: this command may reply with two responses. The first response will be received immediately after entering the command. In case the command is valid (ok reply received), a second reply will be received after the end of the uplink transmission. Please refer to the *LoRaWAN™ Specification* for further details.

Response after entering the command:

- ok – if parameters and configurations are valid and the packet was forwarded to the radio transceiver for transmission
- invalid_param – if parameters (<type> <portno> <data>) are not valid
- not_joined – if the network is not joined
- no_free_ch – if all channels are busy
- silent – if the module is in a Silent Immediately state
- frame_counter_err_rejoin_needed – if the frame counter rolled over
- busy – if MAC state is not in an Idle state
- mac_paused – if MAC was paused and not resumed back
- invalid_data_len if application payload length is greater than the maximum application payload length corresponding to the current data rate

Response after the uplink transmission:

- mac_tx_ok if uplink transmission was successful and no downlink data was received back from the server;
- mac_rx <portno> <data> if transmission was successful, <portno>: port number, from 1 to 223; <data>: hexadecimal value that was received from the server;
- mac_err if transmission was unsuccessful, ACK not received back from the server
- invalid_data_len if application payload length is greater than the maximum application payload length corresponding to the current data rate

A confirmed message will expect an acknowledgment from the server; otherwise, the message will be retransmitted by the number indicated by the command mac set retx <value>, whereas an unconfirmed message will not expect any acknowledgment back from the server. Please refer to the *LoRaWAN™ Specification* for further details.

If the automatic reply feature is enabled and the server sets the Frame Pending bit or initiates downlink confirmed transmissions, multiple responses will be displayed after each downlink packet is received by the module. A typical scenario for this case would be (prerequisites: free LoRaWAN channels available and automatic reply enabled):

- The module sends a packet on port 4 with application payload `0xAB`
- Radio transmission is successful and the module will display the first response: `ok`
- The server needs to send two separate downlink confirmed packets back on port 1 with the following data: `0xAC`, then `0xAF`. First it will transmit the first one (`0xAC`) and will set the Frame Pending bit. The module will display the second response `mac_rx 1 AC`
- The module will initiate an automatic uplink unconfirmed transmission with no application payload on the first free channel because the Frame Pending bit was set in the downlink transmission
- The server will send back the second confirmed packet (`0xAF`). The module will display a third response `mac_rx 1 AF`
- The module will initiate an automatic unconfirmed transmission with no application payload on the first free channel because the last downlink transmission was confirmed, so the server needs an ACK
- If no reply is received back from the server, the module will display the fourth response after the end of the second Receive window: `mac_tx_ok`
- After this scenario, the user is allowed to send packets when at least one enabled channel is free

Based on this scenario, the following responses will be displayed by the module:

- `mac tx cnf 4 AB`
- `ok`
- `mac_rx 1 AC`
- `mac_rx 1 AF`
- `mac_tx_ok`

Example: **`mac tx cnf 4 5A5B5B`** // Sends a confirmed frame on port 4 with application payload 5A5B5B.

2.4.3 mac join <mode>

<mode>: string representing the join procedure type (case-insensitive), either otaa or abp (otaa – over-the-air activation, abp – activation by personalization).

Response: this command may reply with two responses. The first response will be received immediately after entering the command. In case the command is valid (ok reply received) a second reply will be received after the end of the join procedure. Please refer to the *LoRaWAN™ Specification* for further details.

Response after entering the command:

- ok – if parameters and configurations are valid and the join request packet was forwarded to the radio transceiver for transmission
- invalid_param – if <mode> is not valid
- keys_not_init – if the keys corresponding to the Join mode (otaa or abp) were not configured
- no_free_ch – if all channels are busy
- silent – if the device is in a Silent Immediately state
- busy – if MAC state is not in an Idle state
- mac_paused – if MAC was paused and not resumed back

Response after the join procedure:

- denied if the join procedure was unsuccessful (the module attempted to join the network, but was rejected);
- accepted if the join procedure was successful;

This command informs the RN2483 module it should attempt to join the configured network. Module activation type is selected with <mode>. Parameter values can be otaa (over-the-air activation) or abp (activation by personalization). The <mode> parameter is not case sensitive. Before joining the network, the specific parameters for each activation type should be configured (for over the air activation: device EUI, application EUI, application key; for activation by personalization: device address, network session key, application session key).

Example: **mac join otaa** // Attempts to join the network using over-the-air activation.

2.4.4 mac save

Response: `ok`

The `mac save` command must be issued after configuration parameters have been appropriately entered from the `mac set <cmd>` commands. This command will save LoRaWAN Class A protocol configuration parameters to the user EEPROM. When the next `sys reset` command is issued, the LoRaWAN Class A protocol configuration will be initialized with the last saved parameters.

The LoRaWAN Class A protocol configuration savable parameters are:

- `band`: Band
- `fcntup`: Uplink Frame Counter
- `fcntdown`: Downlink Frame Counter
- `dr`: Data Rate
- `rx2dr`: Data Rate parameter for the second receive window
- `rx2freq`: Frequency parameter for the second receive window
- `adr`: Adaptive Data Rate state
- `deveui`: End-Device Identifier
- `appeui`: Application Identifier
- `appkey`: Application Key
- `nwkskey`: Network Session Key
- `appskey`: Application Session Key
- `devaddr`: End Device Address
- `ch`: All Channel Parameter
  - `freq`: Frequency
  - `dcycle`: Duty Cycle
  - `drrange`: Data Rate Range
  - `status`: Status

Example: **`mac save`** // Saves the LoRaWAN Class A protocol configuration parameters to the user EEPROM.

2.4.5 mac forceENABLE

Response: `ok`

The network can issue a certain command (Duty Cycle Request frame with parameter 255) that would require the RN2483 module to go silent immediately. This mechanism disables any further communication of the module, effectively isolating it from the network. Using `mac forceENABLE`, after this network command has been received, restores the module's connectivity by allowing it to send data.

Example: **`mac forceENABLE`** // Disables the Silent Immediately state.

2.4.6 mac pause

Response: 0 – 4294967295 (decimal number representing the number of milliseconds the mac can be paused)

This command pauses the LoRaWAN stack functionality to allow transceiver (radio) configuration. Through the use of `mac pause`, radio commands can be generated between a LoRaWAN Class A protocol uplink application (`mac tx` command), and the LoRaWAN Class A protocol Receive windows (second response for the `mac tx` command). This command will reply with the time interval in milliseconds that the transceiver can be used without affecting the LoRaWAN functionality. The maximum value (4294967295) is returned whenever the LoRaWAN stack functionality is in Idle state and the transceiver can be used without restrictions. '`0`' is returned when the LoRaWAN stack functionality cannot be paused. After the radio configuration is complete, the `mac resume` command should be used to return to LoRaWAN Class A protocol commands.

Example: **`mac pause`** // Pauses the LoRaWAN stack functionality if the response is different from 0.

Note:	If already joined to a network, this command *MUST* be called *BEFORE* configuring the radio parameters, initiating radio reception, or transmission.

2.4.7 mac resume

Response: `ok`

This command resumes LoRaWAN stack functionality, in order to continue normal functionality after being paused.

Example: **`mac resume`** // Resumes the LoRaWAN stack functionality.

Note:	This command *MUST* be called *AFTER* all radio commands have been issued and all the corresponding asynchronous messages have been replied.

2.4.8 MAC Set Commands

TABLE 2-6: MAC SET COMMANDS

Parameter	Description
devaddr	Sets the unique network device address for the RN2483 module.
deveui	Sets the globally unique identifier for the RN2483 module.
appeui	Sets the application identifier for the RN2483 module.
nwkskey	Sets the network session key for the RN2483 module.
appskey	Sets the application session key for the RN2483 module.
appkey	Sets the application key for the RN2483 module.
pwridx	Sets the output power to be used on the next transmissions.
dr	Sets the data rate to be used for the next transmissions.
adr	Sets if the adaptive data rate is to be enabled, or disabled.
bat	Sets the battery level needed for Device Status Answer frame command response.
retx	Sets the number of retransmissions to be used for an uplink confirmed packet.
linkchk	Sets the time interval for the link check process to be triggered.
rxdelay1	Sets the value used for the first Receive window delay.
ar	Sets the state of the automatic reply.
rx2	Sets the data rate and frequency used for the second Receive window.
sync	Sets the synchronization word for the LoRaWAN™ communication.
upctr	Sets the value of the uplink frame counter that will be used for the next uplink transmission.
dnctr	Sets the value of the downlink frame counter that will be used for the next downlink reception.
ch	Allows modification of channel related parameters.

2.4.8.1 `mac set devaddr <address>`

`<address>`: 4-byte hexadecimal number representing the device address, from 00000000 – FFFFFFFF

Response: `ok` if address is valid

`invalid_param` if address is not valid

This command configures the module with a 4-byte unique network device address `<address>`. The `<address>` *MUST* be *UNIQUE* to the current network. This must be directly set solely for activation by personalization devices. This parameter must not be set before attempting to join using over-the-air activation because it will be overwritten once the join process is over.

Example: **`mac set devaddr ABCDEF01`**

> **Note:** If this parameter had previously been saved to user EEPROM by issuing the `mac save` command, after modifying its value, the `mac save` command should be called again.

2.4.8.2 mac set deveui <devEUI>

<devEUI>: 8-byte hexadecimal number representing the device EUI

Response: ok if address is valid

invalid_param if address is not valid

This command sets the globally unique device identifier for the module. The identifier must be set by the host MCU. The module contains a pre-programmed unique EUI and can be retrieved using the sys get hweui command (see Section 2.3.6.4) or user provided EUI can be configured using the mac set deveui command.

Example: **mac set deveui 0004A30B001A55ED**

> **Note:** If this parameter was previously saved to user EEPROM by issuing the mac save command, after modifying its value, the mac save command should be called again.

2.4.8.3 mac set appeui <appEUI>

<appEUI>: 8-byte hexadecimal number representing the application EUI

Response: ok if EUI is valid

invalid_param if EUI is not valid

This command sets the application identifier for the module. The application identifier should be used to identify device types (sensor device, lighting device, etc.) within the network.

Example: **mac set appeui FEDCBA9876543210**

> **Note:** If this parameter was previously saved to user EEPROM by issuing the mac save command, after modifying its value, the mac save command should be called again.

2.4.8.4 mac set nwkskey <nwksesskey>

<nwkSessKey>: 16-byte hexadecimal number representing the network session key

Response: ok if key is valid

invalid_param if key is not valid

This command sets the network session key for the module. This key is 16 bytes in length, and provides security for communication between the module and network server.

Example: **mac set nwkskey 1029384756AFBECD5647382910DACFEB**

> **Note:** If this parameter was previously saved to user EEPROM by issuing the mac save command, after modifying its value, the mac save command should be called again.

2.4.8.5 `mac set appskey <appSesskey>`

`<appSessKey>`: 16-byte hexadecimal number representing the application session key

Response: `ok` if key is valid

`invalid_param` if key is not valid

This command sets the application session key for the module. This key provides security for communication between module and application server.

Example: **`mac set appskey AFBECD56473829100192837465FAEBDC`**

> **Note:** If this parameter was previously saved to user EEPROM by issuing the `mac save` command, after modifying its value, the `mac save` command should be called again.

2.4.8.6 `mac set appkey <appKey>`

`<appKey>`: 16-byte hexadecimal number representing the application key

Response: `ok` if key is valid

`invalid_param` if key is not valid

This command sets the application key for the module. The application key is used to derive the security credentials for communication during over-the-air activation.

Example: **`mac set appkey 00112233445566778899AABBCCDDEEFF`**

> **Note:** If this parameter was previously saved to user EEPROM by issuing the `mac save` command, after modifying its value, the `mac save` command should be called again.

2.4.8.7 `mac set pwridx <pwrIndex>`

`<pwrIndex>`: decimal number representing the index value for the output power, from 0 to 5 for 433 MHz frequency band and from 1 to 5 for 868 MHz frequency band.

Response: `ok` if power index is valid

`invalid_param` if power index is not valid

This command sets the output power to be used on the next transmissions. Refer to the *LoRaWAN™ Specification* for the output power corresponding to the `<pwrIndex>` and also to the *RN2483 Low-Power Long-Range LoRa® Technology Transceiver Module Data Sheet* (DS50002346) for the actual radio power capabilities.

Example: **`mac set pwridx 1`** // Sets the TX output power to 14 dBm on the next transmission for a 868 MHz EU module.

2.4.8.8 `mac set dr <dataRate>`

`<dataRate>`: decimal number representing the data rate, from 0 and 7, but within the limits of the data rate range for the defined channels.

Response: `ok` if data rate is valid

`invalid_param` if data rate is not valid

This command sets the data rate to be used for the next transmission. Please refer to the *LoRaWAN™ Specification* for the description of data rates and the corresponding spreading factors.

Example: **`mac set dr 5`** // On EU863-870; SF7/125 kHz.

> **Note:** If this parameter had previously been saved to user EEPROM by issuing the `mac save` command, after modifying its value, the `mac save` command should be called again.

2.4.8.9 `mac set adr <state>`

`<state>`: string value representing the state, either `on` or `off`.

Response: `ok` if state is valid

`invalid_param` if state is not valid

This command sets if the adaptive data rate (ADR) is to be enabled, or disabled. The server is informed about the status of the module's ADR in every uplink frame it receives from the ADR field in uplink data packet. If ADR is enabled, the server will optimize the data rate and the transmission power of the module based on the information collected from the network.

Example: **`mac set adr on`** // This will enable the ADR mechanism.

> **Note:** If this parameter had previously been saved to user EEPROM by issuing the `mac save` command, after modifying its value, the `mac save` command should be called again.

2.4.8.10 `mac set bat <level>`

`<level>`: decimal number representing the level of the battery, from 0 to 255. '`0`' means external power, '`1`' means low level, 254 means high level, 255 means the end device was not able to measure the battery level.

Response: `ok` if the battery level is valid

`invalid_param` if the battery level is not valid

This command sets the battery level required for Device Status Answer frame in use with the LoRaWAN Class A protocol.

Example: **`mac set bat 127`** // Battery is set to ~50%.

2.4.8.11 `mac set retx <reTxNb>`

`<reTxNb>`: decimal number representing the number of retransmissions for an uplink confirmed packet, from 0 to 255.

Response: `ok` if `<retx>` is valid

`invalid_param` if `<retx>` is not valid

This command sets the number of retransmissions to be used for an uplink confirmed packet, if no downlink acknowledgment is received from the server.

Example: **`mac set retx 5`** // The number of retransmissions made for an uplink confirmed packet is set to 5.

2.4.8.12 `mac set linkchk <linkCheck>`

`<linkCheck>`: decimal number that sets the time interval in seconds for the link check process, from 0 to 65535

Response: `ok` if the time interval is valid

`invalid_param` if the time interval is not valid

This command sets the time interval for the link check process to be triggered periodically. A `<value>` of '0' will disable the link check process. When the time interval expires, the next application packet that will be sent to the server will include also a link check MAC command. Please refer to the *LoRaWAN™ Specification* for more information on the Link Check MAC command.

Example: **`mac set linkchk 600`** // The module will attempt a link check process at 600-second intervals.

Note: If the command `mac reset` is issued, the link check process will be set as disabled.

2.4.8.13 `mac set rxdelay1 <rxDelay>`

`<rxDelay>`: decimal number representing the delay between the transmission and the first Reception window in milliseconds, from 0 to 65535.

Response: `ok` if `<rxDelay>` is valid

`invalid_param` if `<rxDelay>` is not valid

This command will set the delay between the transmission and the first Reception window to the `<rxDelay>` in milliseconds. The delay between the transmission and the second Reception window is calculated in software as the delay between the transmission and the first Reception window + 1000 (ms).

Example: **`mac set rxdelay1 1000`** // Set the delay between the transmission and the first Receive window to 1000 ms.

2.4.8.14 `mac set ar <state>`

`<state>`: string value representing the state, either `on` or `off`.

Response: `ok` if state is valid

`invalid_param` if state is not valid

This command sets the state of the automatic reply. By enabling the automatic reply, the module will transmit a packet without a payload immediately after a confirmed downlink is received, or when the Frame Pending bit has been set by the server. If set to OFF, no automatic reply will be transmitted.

Example: **`mac set ar on`** // Enables the automatic reply process inside the module.

Note: The RN2483 module implementation will initiate automatic transmissions with no application payload if the automatic reply feature is enabled and the server sets the Frame Pending bit or initiates a confirmed downlink transmission. In this case, if all enabled channels are busy due to duty cycle limitations, the stack will wait for the first channel that will become free to transmit. The user will not be able to initiate uplink transmissions until the automatic transmissions are done.

2.4.8.15 `mac set rx2 <dataRate> <frequency>`

`<dataRate>`: decimal number representing the data rate, from 0 to 7.

`<frequency>`: decimal number representing the frequency, from 863000000 to 870000000 or from 433050000 to 434790000, in Hz.

Response: `ok` if parameters are valid

`invalid_param` if parameters are not valid

This command sets the data rate and frequency used for the second Receive window. The configuration of the Receive window parameters should be in concordance with the server configuration.

Example: **`mac set rx2 3 865000000`** // Receive window 2 is configured with SF9/125 kHz data rate with a center frequency of 865 MHz.

Note: If this parameter had previously been saved to user EEPROM by issuing the `mac save` command, after modifying its value, the `mac save` command should be called again.

2.4.8.16 `mac set sync <synchWord>`

`<synchWord>`: one byte long hexadecimal number representing the synchronization word for the LoRaWAN communication

Response: `ok` if parameters are valid

`invalid_param` if parameter is not valid

This command sets the synchronization word for the LoRaWAN communication. The configuration of the synchronization word should be in concordance with the Gateway configuration.

Example: `mac set sync 34` //Synchronization word is configured to use the 0x34 value

2.4.8.17 `mac set upctr <fCntUp>`

`<fCntUp>`: decimal number representing the value of the uplink frame counter that will be used for the next uplink transmission, from 0 to 4294967295.

Response: `ok` if parameter is valid

`invalid_param` if parameter is not valid

This command sets the value of the uplink frame counter that will be used for the next uplink transmission.

Example: `mac set upctr 10`

Note: If this parameter had previously been saved to user EEPROM by issuing the `mac save` command, after modifying its value, the `mac save` command should be called again.

2.4.8.18 `mac set dnctr <fCntDown>`

`<fCntDown>`: decimal number representing the value of the downlink frame counter that will be used for the next downlink reception, from 0 to 4294967295.

Response: `ok` if parameter is valid

`invalid_param` if parameter is not valid

This command sets the value of the downlink frame counter that will be used for the next downlink reception.

Example: `mac set dnctr 30`

> **Note:** If this parameter had previously been saved to user EEPROM by issuing the `mac save` command, after modifying its value, the `mac save` command should be called again.

2.4.8.19 MAC SET CHANNEL COMMANDS

TABLE 2-7: MAC SET CHANNEL COMMANDS

Parameter	Description
`freq`	Sets the module operation frequency on a given channel ID.
`dcycle`	Sets the module operation duty cycle on a given channel ID.
`drrange`	Sets the module allowed data rate range (min.- max.) allowed on a given channel ID.
`status`	Sets the use of the specified channel ID.

2.4.8.19.1 `mac set ch freq <channelID> <frequency>`

`<channelId>`: decimal number representing the channel number, from 3 to 15.

`<frequency>`: decimal number representing the frequency, from 863000000 to 870000000 or from 433050000 to 434790000, in Hz.

Response: `ok` if parameters are valid

`invalid_param` if parameters are not valid

This command sets the operational frequency on the given channel ID. The default channels (0-2) cannot be modified in terms of frequency.

Example: **`mac set ch freq 13 864000000`** // Define frequency for channel 13 to be 864 MHz.

> **Note:** If this parameter was previously saved to user EEPROM by issuing the `mac save` command, after modifying its value, the `mac save` command should be called again.

2.4.8.19.2 `mac set ch dcycle <channelID> <dutyCycle>`

`<channelId>`: decimal number representing the channel number, from 0 to 15.

`<dutyCycle>`: decimal number representing the duty cycle, from 0 to 65535.

Response: `ok` if parameters are valid

`invalid_param` if parameters are not valid

This command sets the duty cycle used on the given channel ID on the module. The `<dutyCycle>` value that needs to be configured can be obtained from the actual duty cycle X (in percentage) using the following formula: `<dutyCycle>` = (100/X) – 1. The default settings consider only the three default channels (0-2), and their default duty cycle is 0.33%. If a new channel is created either by the server or by the user, all the channels (including the default ones) must be updated by the user in terms of duty cycle to comply with the ETSI regulations.

Example: **`mac set ch dcycle 13 9`** // Defines duty cycle for channel 13 to be 10%. Since (100/10) – 1 = 9, the parameter that gets configured is 9.

Note: If this parameter was previously saved to user EEPROM by issuing the `mac save` command, after modifying its value, the `mac save` command should be called again.

2.4.8.19.3 `mac set ch drrange <channelID> <minRange> <maxRange>`

`<channelId>`: decimal number representing the channel number, from 0 to 15

`<minRange>`: decimal number representing the minimum data rate, from 0 to 7

`<maxRange>`: decimal number representing the maximum data rate, from 0 to 7

Response: `ok` if parameters are valid

`invalid_param` if parameters are not valid

This command sets the operating data rate range, min. to max., for the given `<channelId>`. By doing this the module can vary data rates between the `<minRange>` and `<maxRange>` on the specified `<channelId>`. Please refer to the *LoRaWAN™ Specification* for the actual values of the data rates and the corresponding spreading factors (SF).

Example: **`mac set ch drrange 13 0 2`** // Using EU863-870 band: on channel 13 the data rate can range from 0 (SF12/125 kHz) to 2 (SF10/125 kHz) as required.

Note: If this parameter was previously saved to user EEPROM by issuing the `mac save` command, after modifying its value, the `mac save` command should be called again.

2.4.8.19.4 `mac set ch status <channel ID> <status>`

`<channelId>`: decimal number representing the channel number, from 0 to 15.

`<status>`: string value representing the state, either `on` or `off`.

Response: `ok` if parameters are valid

`invalid_param` if parameters are not valid

This command sets the operation of the given `<channelId>`.

Example: **`mac set ch status 4 off`** // Channel ID 4 is disabled from use.

WARNING

`<ChannelId>` parameters (frequency, data range, duty cycle) must be issued prior to enabling the status of that channel.

Note: If this parameter was previously saved to user EEPROM by issuing the `mac save` command, after modifying its value, the `mac save` command should be called again.

2.4.9 MAC Get Commands

TABLE 2-8: MAC GET COMMANDS

Parameter	Description
devaddr	Gets the current stored unique network device address for that specific end device.
deveui	Gets the current stored globally unique identifier for that specific end device.
appeui	Gets the application identifier for the end device.
dr	Gets the data rate to be used for the next transmission.
band	Gets the current frequency band in operation.
pwridx	Gets the output power index value.
adr	Gets the state of adaptive data rate for the device.
retx	Gets the number of retransmissions to be used for an uplink confirmed packet.
rxdelay1	Gets the interval value stored for `rxdelay1`.
rxdelay2	Gets the interval value stored for `rxdelay2`.
ar	Gets the state of the automatic reply.
rx2	Gets the data rate and frequency used for the second Receive window.
dcycleps	Gets the duty cycle prescaler which can only be configured by the server.
mrgn	Gets the demodulation margin as received in the last Link Check Answer frame.
gwnb	Gets the number of gateways that successfully received the last Link Check Request frame.
status	Gets the current status of the RN2483 module.
sync	Gets the synchronization word for the LoRaWAN communication.
upctr	Gets the value of the uplink frame counter that will be used for the next uplink transmission.
dnctr	Gets the value of the downlink frame counter that will be used for the next downlink reception.
ch	Gets parameters related information which pertains to channel operation and behaviors.

2.4.9.1 mac get devaddr

Response: 4-byte hexadecimal number representing the device address, from 00000000 to FFFFFFFF.

This command will return the current end-device address of the module.

Default: 00000000

Example: **mac get devaddr**

2.4.9.2 mac get deveui

Response: 8-byte hexadecimal number representing the device EUI.

This command returns the globally unique end-device identifier, as set in the module.

Default: pre-programmed EUI node address

Example: **mac get deveui**

> **Note:** After the `mac reset <band>` command is explicitly called, the device EUI value will be set to all zeros. Make certain that a valid value is given to the device EUI.

2.4.9.3 mac get appeui

Response: 8-byte hexadecimal number representing the application EUI.

This command will return the application identifier for the module. The application identifier is a value given to the device by the network.

Default: 0000000000000000

Example: **mac get appeui**

2.4.9.4 mac get dr

Response: decimal number representing the current data rate.

This command will return the current data rate.

Default: 5

Example: **mac get dr**

2.4.9.5 mac get band

Response: decimal number representing the frequency band, either 868 or 433.

This command returns the current frequency band of operation. The band reflects the module's operation types.

Default: 868

Example: **mac get band**

2.4.9.6 mac get pwridx

Response: decimal number representing the current output power index value, from 0 to 5.

This command returns the current output power index value.

Default: 1

Example: **mac get pwridx**

2.4.9.7 mac get adr

Response: string representing the state of the adaptive data rate mechanism, either `on` or `off`.

This command will return the state of the adaptive data rate mechanism. It will reflect if the ADR is `on` or `off` on the requested device.

Default: `off`

Example: **`mac get adr`**

2.4.9.8 mac get retx

Response: decimal number representing the number of retransmissions, from 0 to 255.

This command will return the currently configured number of retransmissions which are attempted for a confirmed uplink communication when no downlink response has been received.

Default: 7

Example: **`mac get retx`**

2.4.9.9 mac get rxdelay1

Response: decimal number representing the interval, in milliseconds, for `rxdelay1`, from 0 to 65535.

This command will return the interval, in milliseconds, for `rxdelay1`.

Default: 1000

Example: **`mac get rxdelay1`**

2.4.9.10 mac get rxdelay2

Response: decimal number representing the interval, in milliseconds, for `rxdelay2`, from 0 to 65535.

This command will return the interval, in milliseconds, for `rxdelay2`.

Default: 2000

Example: **`mac get rxdelay2`**

2.4.9.11 mac get ar

Response: string representing the state of the automatic reply, either `on` or `off`.

This command will return the current state for the automatic reply (AR) parameter. The response will indicate if the AR is on or off.

Default: `off`

Example: **`mac get ar`**

2.4.9.12 mac get rx2 <freqBand>

<freqBand>: decimal number representing the frequency band, either 868 or 433.

Response: decimal number representing the data rate configured for the second Receive window, from 0 to 7 and a decimal number for the frequency configured for the second Receive window, from 863000000 to 870000000 or from 433050000 to 434790000, in Hz.

This command will return the current data rate and frequency configured to be used during the second Receive window.

Default: 0 869525000 // for 868 band

0 434665000 // for 433 band

Example: **mac get rx2 868**

2.4.9.13 mac get dcycleps

Response: decimal number representing the prescaler value, from 0 to 65535.

This command returns the duty cycle prescaler. The value of the prescaler can be configured *ONLY* by the *SERVER* through use of the Duty Cycle Request frame. Upon reception of this command from the server, the duty cycle prescaler is changed for all enabled channels.

Default: 1

Example: **mac get dcycleps**

2.4.9.14 mac get mrgn

Response: decimal number representing the demodulation margin, from 0 to 255.

This command will return the demodulation margin as received in the last Link Check Answer frame. Please refer to the *LoRaWAN™ Specification* for the description of the values.

Default: 255

Example: **mac get mrgn**

2.4.9.15 mac get gwnb

Response: decimal number representing the number of gateways, from 0 to 255.

This command will return the number of gateways that successfully received the last Link Check Request frame command, as received in the last Link Check Answer.

Default: 0

Example: **mac get gwnb**

2.4.9.16 `mac get status`

Response: 4-byte hexadecimal number representing the current status of the module.

This command will return the current status of the module. The value returned is a bit mask represented in hexadecimal form. Please refer to Figure 2-1 for the significance of the bit mask.

Default: 00000000

Example: **`mac get status`**

2.4.9.17 `mac get sync`

Response: one byte long hexadecimal number representing the synchronization word for the LoRaWAN communication.
This command will return the synchronization word for the LoRaWAN communication.
Default: 34

Example: `mac get sync`

2.4.9.18 `mac get upctr`

Response: decimal number representing the value of the uplink frame counter that will be used for the next uplink transmission, from 0 to 4294967295.
This command will return the value of the uplink frame counter that will be used for the next uplink transmission.
Default: 0

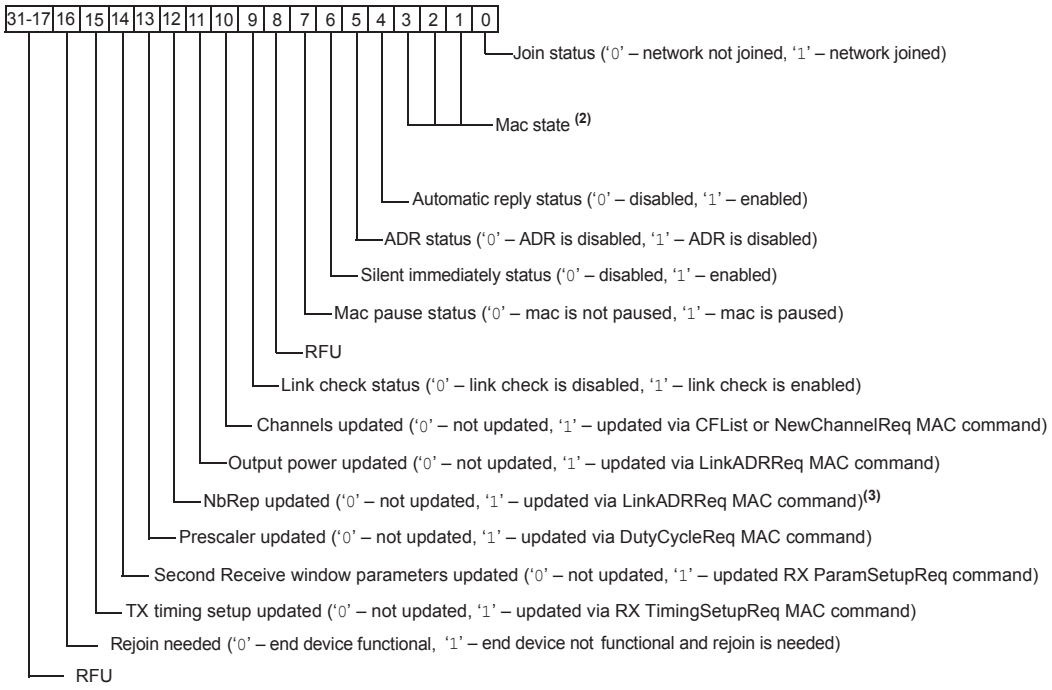
Example: `mac get upctr`

2.4.9.19 `mac get dnctr`

Response: decimal number representing the value of the downlink frame counter that will be used for the next downlink reception, from 0 to 4294967295.
This command will return the value of the downlink frame counter that will be used for the next downlink reception.
Default: 0

Example: `mac get dnctr`

FIGURE 2-1: MAC STATUS BIT-MAPPED REGISTER (1)



Note 1: Bits 10 (Channels updated), 11 (Output power updated), 12 (NbRep updated), 13 (Prescaler updated), 14 (Second Receive window parameters updated) and 15 (TX timing setup updated) are cleared after issuing a "`mac get status`" command.

2: Mac state:

0 – Idle (transmissions are possible)

1 – Transmission occurring

2 – Before the opening of Receive window 1

3 – Receive window 1 is open

4 – Between Receive window 1 and Receive window 2

5 – Receive window 2 is open

6 – Ack_timeout (Ack_timeout is described in more detail in the *LoRaWAN™ specification*)

3: NbRep is the number of repetitions for unconfirmed packets (please refer to the *LoRaWAN™ Specification* for more details).

2.4.9.20 MAC GET CHANNEL COMMANDS

TABLE 2-9: MAC GET CHANNEL COMMANDS

Parameter	Description
`freq`	Gets the module operation frequency for the specified channel ID.
`dcycle`	Gets the module duty cycle used for transmission on the specified channel ID.
`drrange`	Gets the valid data rate range (min. to max.) allowed for the module on the specified channel ID
`status`	Gets the status for the specified channel ID to indicate if it is enabled for use.

TABLE 2-10: DEFAULT PARAMETERS FOR CHANNELS

Channel Number	Parameters	Frequency band	
		868	433
Channel 0	Frequency (Hz)	868100000	433175000
	Duty cycle (1)	302	302
	Data rate range	0-5	0-5
	Status	On	On
Channel 1	Frequency (Hz)	868300000	433375000
	Duty cycle(1)	302	302
	Data rate range	0-5	0-5
	Status	On	On
Channel 2	Frequency (Hz)	868500000	433575000
	Duty cycle(1)	302	302
	Data rate range	0-5	0-5
	Status	On	On
Channels 3-15	Frequency (Hz)	0	0
	Duty cycle(1)	65535	65535
	Data rate range	15 15	15 15
	Status	Off	Off

Note 1: The default settings consider only the three default channels (0-2), and their default duty cycle is 0.33%. If a new channel is created either by the server or by the user, all the channels (including the default ones) must be updated by the user in terms of duty cycle to comply with the ETSI regulations.

2.4.9.20.1 `mac get ch freq <ChannelId>`

`<channelId>`: decimal number representing the channel number, from 0 to 15.

Response: decimal number representing the frequency of the channel, from 863000000 to 870000000 or from 433050000 to 434790000, in Hz, depending on the frequency band selected.

This command returns the frequency on the requested `<channelId>`, entered in decimal form.

Default: see Table 2-10

Example: **`mac get ch freq 0`**

2.4.9.20.2 `mac get ch dcycle <channelId>`

`<channelId>`: decimal number representing the channel number, from 0 to 15.

Response: decimal number representing the duty cycle of the channel, from 0 to 65535.

This command returns the duty cycle on the requested `<channelId>`. The duty cycle is returned in decimal value. The actual duty cycle (in percentage) can be obtained using the returned value V as: percent = 100/(V + 1).

Default: see Table 2-10

Example: **`mac get ch dcycle 0`** // Reads back duty cycle setting on Channel ID 0. If the value reported back is 99, the actual duty cycle on the channel (in percentage) is 100/(99 + 1) = 1.

2.4.9.20.3 `mac get ch drrange <channelId>`

`<channelId>`: decimal number representing the channel number, from 0 to 15.

Response: decimal number representing the minimum data rate of the channel, from 0 to 7 and a decimal number representing the maximum data rate of the channel, from 0 to 7

This command returns the allowed data rate index range on the requested `<channelId>`, entered in decimal form. The `<minRate>` and `<maxRate>` index values are returned in decimal form and reflect index values. Please refer to the *LoRaWAN™ Specification* for the description of data rates and the corresponding spreading factors.

Default: see Table 2-10

Example: **`mac get ch drrange 0`**

2.4.9.20.4 `mac get ch status <channelId>`

`<channelId>`: decimal number representing the channel number, from 0 to 15.

Response: string representing the state of the channel, either `on` or `off`.

This command returns if `<channelId>` is currently enabled for use. `<channelId>` is entered in decimal form and the response will be `on` or `off` reflecting the channel is enabled or disabled appropriately.

Default: see Table 2-10

Example: **`mac get ch status 2`**

Note: `<ChannelId>` parameters must be issued prior to enabling the status of that channel. If a channel is disabled through the `<status>`, all channel parameters must be reconfigured prior to enabling.

2.5 RADIO COMMANDS

TABLE 2-11: RADIO COMMANDS(1)

Parameter	Description
rx	This command configures the radio to receive simple radio packets according to prior configuration settings.
tx	This command configures a simple radio packet transmission according to prior configuration settings.
cw	This command will put the module into a Continuous Wave (cw) Transmission for system tuning or certification use.
set	This command allows modification to the radio setting directly. This command allows for the user to change the method of radio operation within module type band limits.
get	This command grants the ability to read out radio settings as they are currently configured.

Note 1: The `mac pause` command must be called before any radio transmission or reception, even if no MAC operations have been initiated before.

TABLE 2-12: RADIO PARAMETERS AVAILABILITY FOR DIFFERENT OPERATIONS

Command	`radio get`	`radio set`	Availability for LoRa® Modulation	Availability for FSK Modulation
bt	√	√	—	√
mod	√	√	√	√
freq	√	√	√	√
pwr	√	√	√	√
sf	√	√	√	—
afcbw	√	√	—	√
rxbw	√	√	—	√
bitrate	√	√	—	√
fdev	√	√	—	√
prlen	√	√	—	√
crc	√	√	√	√
iqi	√	√	√	—
cr	√	√	√	—
wdt	√	√	√	√
sync	√	√	√	√
bw	√	√	√	—
snr	√	—	√	—

2.5.1 radio rx <rxWindowSize>

<rxWindowSize>: decimal number representing the number of symbols (for LoRa modulation) or time-out (in milliseconds, for FSK modulation) that the receiver will be opened, from 0 to 65535. Set <rxWindowSize> to '0' in order to enable the Continuous Reception mode. Continuous Reception mode will be exited once a valid packet is received.

Response: this command may reply with two responses. The first response will be received immediately after entering the command. If the command is valid (ok reply received), a second reply will be received after the reception of a packet or after the time-out occurred.

Response after entering the command:

- ok – if parameter is valid and the transceiver is configured in Receive mode
- invalid_param – if parameter is not valid
- busy – if the transceiver is currently busy

Response after the receive process:

- radio_rx <data> – if reception was successful, <data>: hexadecimal value that was received;
- radio_err – if reception was not successful, reception time-out occurred

Example: **radio rx 0** // Puts the radio into continuous Receive mode.

Ensure the radio Watchdog Timer time-out is higher than the Receive window size.

Note: The mac pause command must be called before any radio transmission or reception, even if no MAC operations have been initiated before.

Note: When transmitting FSK packets, the payload and the 2-byte CRC is whitened by being XORed with a pseudorandom sequence generated by an LFSR with the polynomial $X^9 + X^5 + 1$. This process is automatically reverted on reception so that it is transparent to the user.

2.5.2 radio tx <data>

<data>: hexadecimal value representing the data to be transmitted, from 0 to 255 bytes for LoRa modulation and from 0 to 64 bytes for FSK modulation.

Response: this command may reply with two responses. The first response will be received immediately after entering the command. If the command is valid (ok reply received), a second reply will be received after the effective transmission.

Response after entering the command:

- ok – if parameter is valid and the transceiver is configured in Transmit mode
- invalid_param – if parameter is not valid
- busy – if the transceiver is currently busy

Response after the effective transmission:

- radio_tx_ok – if transmission was successful
- radio_err – if transmission was unsuccessful (interrupted by radio Watchdog Timer time-out)

This command transmits the <data> passed.

Example: **radio tx 48656c6C6F** // Transmits a packet of [0x48][0x65][0x6c][0x6C][0x6F]; Hello.

Note: In order to meet ETSI regulations in the given frequency bands, the radio has to use either Listen Before Talk (LBT) + Adaptive Frequency Agility (AFA) or duty cycle limitations. By issuing the radio tx <data> command the module does not perform LBT before transmission, thus the user has to make sure that duty cycle limits are not violated. For more information on duty cycle limits please check the EN 300 220-2 v2.4.1 standard.

Note: The mac pause command must be called before any radio transmission or reception, even if no MAC operations have been initiated before.

Note: When transmitting FSK packets, the payload and the 2-byte CRC is whitened by being XORed with a pseudorandom sequence generated by an LFSR with the polynomial $X^9 + X^5 + 1$. This process is automatically reverted on reception so that it is transparent to the user.

2.5.3 radio cw <state>

<state>: string representing the state of the Continuous Wave (CW) mode, either on or off.

Response: ok if state is valid

invalid_param if state is not valid

This command will enable or disable the CW mode on the module. CW mode allows the user to put the transceiver into Transmission mode to observe the generated signal. By altering the settings for the radio the user can observe the changes in transmissions levels.

Example: **radio cw on**

Note: Please note that using radio cw off resets the module, this command being semantically identical to sys reset.

2.5.4 Radio Set Commands

TABLE 2-13: RADIO SET COMMANDS

Parameter	Description
bt	Set the data shaping for frequency shift keying (FSK) modulation type.
mod	Set the module Modulation mode.
freq	Set the current operation frequency for the radio.
pwr	Set the output power level used by the radio during transmission.
sf	Set the requested spreading factor (SF) to be used during transmission.
afcbw	Set the value used by the automatic frequency correction bandwidth.
rxbw	Set the operational receive bandwidth.
bitrate	Set the frequency shift keying (FSK) bit rate.
fdev	Set the frequency deviation allowed by the end device.
prlen	Set the preamble length used during transmissions.
crc	Set if a CRC header is to be used.
iqi	Set if IQ inversion is used.
cr	Set the coding rate used by the radio.
wdt	Set the time-out limit for the radio Watchdog Timer.
sync	Set the sync word used.
bw	Set the value used for the radio bandwidth.

2.5.4.1 radio set bt <gfBT>

<gfBT>: string representing the Gaussian baseband data shaping, enabling GFSK modulation. Parameter values can be: none, 1.0, 0.5, 0.3.

Response: ok if the data shaping is valid

invalid_param if the data shaping is not valid

This command modifies the data shaping applied to FSK transmissions. Entering any <gfBT> other than none will result in a Gaussian Filter BT being applied to transmissions in FSK mode.

Example: **radio set bt none** // Data shaping in FSK mode is disabled or null.

2.5.4.2 radio set mod <mode>

<mode>: string representing the modulation method, either lora or fsk.

Response: ok if the modulation is valid

invalid_param if the modulation is not valid

This command changes the modulation method being used by the module. Altering the mode of operation does not affect previously set parameters, variables or registers. FSK mode also allows GFSK transmissions when data shaping is enabled.

Example: **radio set mod lora**

2.5.4.3 radio set freq <frequency>

<frequency>: decimal representing the frequency, from 433050000 to 434790000 or from 863000000 to 870000000, in Hz.

Response: ok if the frequency is valid

invalid_param if the frequency is not valid

This command changes the communication frequency of the radio transceiver.

Example: **radio set freq 868000000**

2.5.4.4 `radio set pwr <pwrout>`

`<pwrOut>`: signed decimal number representing the transceiver output power, from -3 to 15.

Response: `ok` if the output power is valid

`invalid_param` if the output power is not valid

This command changes the transceiver output power. However, note that the transceiver is designed to transmit a maximum of +14 dBm. It is possible to set the output power above the regulatory limits. This power setting allows some compensation on the cable or transmission line loss. For more details on output power please check the *RN2483 Low-Power Long-Range LoRa® Technology Transceiver Module Data Sheet*.

Example: **`radio set pwr 14`**

2.5.4.5 `radio set sf <spreadingfactor>`

`<spreadingFactor>`: string representing the spreading factor. Parameter values can be: `sf7`, `sf8`, `sf9`, `sf10`, `sf11` or `sf12`.

Response: `ok` if the spreading factor is valid

`invalid_param` if the spreading factor is not valid

This command sets the spreading factor used during transmission.

Example: **`radio set sf sf7`**

2.5.4.6 `radio set afcbw <autoFreqBand>`

`<autoFreqBand>`: float representing the automatic frequency correction, in kHz. Parameter values can be: 250, 125, 62.5, 31.3, 15.6, 7.8, 3.9, 200, 100, 50, 25, 12.5, 6.3, 3.1, 166.7, 83.3, 41.7, 20.8, 10.4, 5.2, 2.6.

Response: `ok` if the automatic frequency correction is valid

`invalid_param` if the automatic frequency correction is not valid

This command modifies the automatic frequency correction bandwidth for receiving/transmitting.

Example: **`radio set afcbw 125`**

2.5.4.7 `radio set rxbw <rxbandwidth>`

`<rxBandwidth>`: float representing the signal bandwidth, in kHz. Parameter values can be: 250, 125, 62.5, 31.3, 15.6, 7.8, 3.9, 200, 100, 50, 25, 12.5, 6.3, 3.1, 166.7, 83.3, 41.7, 20.8, 10.4, 5.2, 2.6.

Response: `ok` if the signal bandwidth is valid

`invalid_param` if signal bandwidth is not valid

This command sets the signal bandwidth when receiving.

Example: **`radio set rxbw 250`** // Signal bandwidth for receiving is 250 kHz.

2.5.4.8 `radio set bitrate <fskBitRate>`

`<fskBitRate>`: decimal number representing the FSK bit rate value, from 1 to 300000.

Response: `ok` if the bit rate value is valid

`invalid_param` if the bit rate value is not valid

This command sets the FSK bit rate value.

Example: **radio set bitrate 5000** // FSK bit rate is set to 5 kb/s.

2.5.4.9 `radio set fdev <freqdev>`

`<freqDev>`: decimal number representing the frequency deviation, from 0 to 200000.

Response: `ok` if the frequency deviation is valid

`invalid_param` if frequency deviation is not valid

This command sets the frequency deviation during operation.

Example: **radio set fdev 5000** // Frequency deviation is 5 kHz.

2.5.4.10 `radio set prlen <preamble>`

`<preamble>`: decimal number representing the preamble length, from 0 to 65535.

Response: `ok` if the preamble length is valid

`invalid_param` if the preamble length is not valid

This command sets the preamble length for transmit/receive.

Example: **radio set prlen 8** // Preamble length is 8.

2.5.4.11 `radio set crc < crcHeader >`

`<crcHeader>`: string representing the state of the CRC header, either `on` or `off`.

Response: `ok` if the state is valid

`invalid_param` if the state is not valid

This command enables or disables the CRC header for communications.

Example: **radio set crc on** // Enables the CRC header.

2.5.4.12 `radio set iqi <iqInvert>`

`<iqInvert>`: string representing the state of the invert IQ, either `on` or `off`.

Response: `ok` if the state is valid

`invalid_param` if the state is not valid

This command enables or disables the Invert IQ for communications.

Example: **radio set iqi on** // Invert IQ is enabled.

2.5.4.13 `radio set cr <codingRate>`

`<codingRate>`: string representing the coding rate. Parameter values can be: `4/5`, `4/6`, `4/7`, `4/8`.

Response: `ok` if the coding rate is valid

`invalid_param` if the coding rate is not valid

This command modifies the coding rate currently being used by the radio.

Example: `radio set cr 4/7` // The coding rate is set to 4/7.

2.5.4.14 radio set wdt <watchDog>

<watchDog>: decimal number representing the time-out length for the Watchdog Timer, from 0 to 4294967295. Set to '0' to disable this functionality.

Response: ok if the watchdog time-out is valid

invalid_param if the watchdog time-out is not valid

This command updates the time-out length, in milliseconds, applied to the radio Watchdog Timer. If this functionality is enabled, then the Watchdog Timer is started for every transceiver reception or transmission. The Watchdog Timer is stopped when the operation in progress in finished.

Example: **radio set wdt 2000** // The Watchdog Timer is configured for 2000 ms.

Note: Ensure the value configured for the Watchdog Timer matches the radio configurations. For example, set the <watchDog> value to '0' in order to disable this functionality during the radio continuous reception.

2.5.4.15 radio set sync <syncWord>

<syncWord>: hexadecimal value representing the Sync word used during communication. For LoRa modulation one byte is used, for FSK up to eight bytes can be entered.

Response: ok if the sync word is valid

invalid_param if the sync word is not valid

This command configures the sync word used during communication.

Example: **radio set sync 12** // Set the sync word to a single byte with the value 0x12.

2.5.4.16 radio set bw <bandWidth>

<bandWidth>: decimal representing the operating radio bandwidth, in kHz. Parameter values can be: 125, 250, 500.

Response: ok if the bandwidth is valid

invalid_param if the bandwidth is not valid

This command sets the operating radio bandwidth for LoRa operation.

Example: **radio set bw 250** // The operating bandwidth is 250 kHz.

2.5.5 Radio Get Commands

TABLE 2-14: RADIO GET COMMANDS

Parameter	Description
bt	Get the data shaping for frequency shift keying (FSK) modulation type.
mod	Get the module Modulation mode.
freq	Get the current operation frequency for the radio.
pwr	Get the output power level used by the radio during transmission.
sf	Get the requested spreading factor (SF) to be used during transmission.
afcbw	Get the value used by the automatic frequency correction bandwidth.
rxbw	Get the operational receive bandwidth.
bitrate	Get the frequency shift keying (FSK) bit rate.
fdev	Get the frequency deviation allowed by the end device.
prlen	Get the preamble length used during transmissions.
crc	Get if a CRC header is to be used.
iqi	Get if IQ inversion is used.
cr	Get the coding rate used by the radio.
wdt	Get the time-out limit for the Watchdog Timer.
bw	Get the value used for the radio bandwidth.
snr	Get the signal noise ratio (SNR) of the last received packet.
sync	Get the synchronization word used for communication.

2.5.5.1 `radio get bt`

Response: string representing the configuration for data shaping. Parameter values can be: `none`, `1.0`, `0.5`, `0.3`.

This command reads back the current configuration for data shaping applied to FSK transmissions.

Default: `0.5`

Example: **`radio get bt`** // Reads the current data shaping FSK configuration.

2.5.5.2 `radio get mod`

Response: string representing the current mode of operation of the module, either `lora` or `fsk`.

This command reads back the current mode of operation of the module.

Default: `lora`

Example: **`radio get mod`** // Reads if module is modulating in LoRa or FSK.

2.5.5.3 radio get freq

Response: decimal number representing the frequency, from 433050000 to 434790000 or from 863000000 to 870000000, in Hz.

This command reads back the current operation frequency of the module.

Default: 868100000

Example: **radio get freq** // Reads back the current frequency the transceiver communicates on.

2.5.5.4 radio get pwr

Response: signed decimal representing the current power level, from -3 to 15.

This command reads back the current power level settings used in operation.

Default: 1

Example: **radio get pwr** // Reads back the current transmit output power.

2.5.5.5 radio get sf

Response: string representing the current spreading factor.

This command reads back the current spreading factor being used by the transceiver. Parameter values can be: sf7, sf8, sf9, sf10, sf11, sf12"

Default: sf12

Example: **radio get sf** // Reads back the current spreading factor settings.

2.5.5.6 radio get afcbw

Response: float representing the automatic frequency correction band, in kHz. Parameter values can be: 250, 125, 62.5, 31.3, 15.6, 7.8, 3.9, 200, 100, 50, 25, 12.5, 6.3, 3.1, 166.7, 83.3, 41.7, 20.8, 10.4, 5.2, 2.6.

This command reads back the status of the Automatic Frequency Correction Bandwidth.

Default: 41.7

Example: **radio get afcbw** // Reads back the current automatic frequency correction bandwidth.

2.5.5.7 radio get rxbw

Response: float representing the signal bandwidth, in kHz. Parameter values can be: 250, 125, 62.5, 31.3, 15.6, 7.8, 3.9, 200, 100, 50, 25, 12.5, 6.3, 3.1, 166.7, 83.3, 41.7, 20.8, 10.4, 5.2, 2.6.

This command reads back the signal bandwidth used for receiving.

Default: 25

Example: **radio get rxbw** // Reads back the receive signal bandwidth.

2.5.5.8 radio get bitrate

Response: signed decimal representing the configured bit rate, from 1 to 300000.

This command reads back the configured bit rate for FSK communications.

Default: 50000

Example: **radio get bitrate** // Reads back the current FSK bit rate setting.

2.5.5.9 radio get fdev

Response: signed decimal representing the frequency deviation setting, from 0 to 200000.

This command reads frequency deviation setting on the transceiver.

Default: 25000

Example: **radio get fdev** // Reads back current configured frequency deviation setting.

2.5.5.10 radio get prlen

Response: signed decimal representing the preamble length, from 0 to 65535.

This command reads the current preamble length used for communication.

Default: 8

Example: **radio get prlen** // Reads back the preamble length used by the transceiver.

2.5.5.11 radio get crc

Response: string representing the status of the CRC header, either `on` or `off`

This command reads back the status of the CRC header, to determine if it is to be included during operation.

Default: `on`

Example: **radio get crc** // Reads back if the CRC header is enabled for use.

2.5.5.12 radio get iqi

Response: string representing the status of the Invert IQ functionality, either `on` or `off`.

This command reads back the status of the Invert IQ functionality.

Default: `off`

Example: **radio get iqi** // Reads back the status of the Invert IQ functionality.

2.5.5.13 radio get cr

Response: string representing the current value settings used for the coding rate. Parameter values can be: `4/5`, `4/6`, `4/7`, `4/8`.

This command reads back the current value settings used for the coding rate during communication.

Default: `4/5`

Example: **radio get cr** // Reads back the current coding rate transceiver settings.

2.5.5.14 radio get wdt

Response: decimal number representing the length used for the watchdog time-out, from 0 to 4294967295.

This command reads back, in milliseconds, the length used for the watchdog time-out.

Default: 15000

Example: **radio get wdt** // Reads back the current time-out value applied to the Watchdog Timer

2.5.5.15 radio get bw

Response: decimal representing the current operating radio bandwidth, in kHz. Parameter values can be: 125, 250 or 500.

This command reads back the current operating radio bandwidth used by the transceiver.

Default: 125

Example: **radio get bw** // Reads back the current operational bandwidth applied to transmissions.

2.5.5.16 radio get snr

Response: signed decimal number representing the signal to noise ratio (SNR), from -128 to 127.

This command reads back the Signal Noise Ratio (SNR) for the last received packet.

Default: -128

Example: **radio get snr** // Reads back the measured SNR for the previously packet reception.

2.5.5.17 radio get sync

Response: hexadecimal number representing the synchronization word used for radio communication.

This command reads back the configured synchronization word used for radio communication. One byte long synchronization word is used for the LoRa modulation while up to eight bytes can be entered for FSK.

Default: 34

Example: radio get sync

Chapter 3. Bootloader Usage

Introduction

This chapter describes the operation of the bootloader that exists on Microchip RN2483 LoRa modules and the process to upgrade the firmware using this bootloader. This paper assumes that there is a microcontroller or other similar device attached to the UART lines of the RN2483 module, and that this microcontroller has enough storage to hold the image that is to be programmed into the module.

The bootloader on the RN2483 module is based on the standard 8-bit UART bootloader which can be found on Microchip's website at http://www.microchip.com/bootloader.

In the Protocol section of the bootloader generator user's guide, there is a table of supported commands. The RN2483 bootloader supports each of these commands; however, there are some subtle nuances that need to be followed for successful operation.

TABLE 3-1: SUPPORTED COMMANDS

Command	Description
0x00	Get Version and other info
0x01	Read Flash
0x02	Write Flash
0x03	Erase Flash
0x04	Read EE Data
0x05	Write EE Data
0x06	Read Configuration Words
0x07	Write Configuration Words
0x08	Calculate and return Flash checksum
0x09	Reset Device

3.1 PROTOCOL

The standard 8-bit UART bootloader has a common command protocol for all commands.

TABLE 3-2: COMMAND PROTOCOL

Byte:	0	1	2	3	4	5	6	7	8
Fields:	CMD	Length(LSB··MSB)		Key 1	Key 2	Address (LSB··MSB)			

<Command><LenLSB><LenMSB><Key1><Key2><address(4 bytes)LSB··MSB>

Byte Order

There are two multi-byte fields common to all commands. These are the Length field, and the Address field. These values are sent in little-endian format. This means that the low-order byte (Least Significant Byte) is sent first, and the high-order byte (Most Significant Byte) is sent last.

Write Operations

When an Erase or Write command is issued, the two key fields must be supplied with correct values. For read operations, they key fields are not used. The values of the keys are always:

- Key1 = 0x55
- Key2 = 0xaa

General Differences from 8-Bit Bootloader

- Module bootloader only uses the first-length byte and ignores the second-length byte in all commands. The protocol still requires that the second length-byte be sent, but the bootloader ignores this value.
- Each command is preceded by 0x55 (ASCII 'U'); this is used for auto-baud detection.

3.2 RN MODULE BOOTLOADER COMMANDS

Because of the differences between the normal Microchip 8-bit bootloader and the bootloader in the RN2483 module, the command format has an initial byte of 0x55, and the second byte of the length field (MSB) is always 0x00.

TABLE 3-3: RN MODULE BOOTLOADER COMMANDS

Byte:	0	1	2	3	4	5	6	7	8	9
Fields:	0x55	CMD	Len	0x00	Key1	Key2	Address (LSB··MSB)			

<0x55><Command><Len><0x00><Key1><Key2><address(4 bytes)LSB··MSB>

3.3 COMMAND DETAILS

TABLE 3-4: GET VERSION INFO

Byte:	0	1	2	3	4	5	6	7	8	9
Values:	0x55	0x00	0x00	0x00	0x00	0x00	0x00	0x00	0x00	0x00

This command returns bootloader version and memory information.

Response:

<0x55><0x01><Len><0x00><0x00><0x00><address(4 bytes)LSB··MSB>

TABLE 3-5:

Byte	Value
0	Bootloader version – low byte
1	Bootloader version – high byte
2	Max. packet size – low byte (not used)
3	Max. packet size – high byte (not used)
4	ACK packet size – low byte (not used)
5	ACK packet size – high byte (not used)
6	Device ID – low byte
7	Device ID – high byte
8	(not used)
9	(not used)
10	Erase Row size
11	Write Latch size
12	User ID 1
13	User ID 2
14	User ID 3
15	User ID 4

`Len` bytes of information follow the address field.

TABLE 3-6: READ FLASH

Byte:	0	1	2	3	4	5	6	7	8	9
Values:	0x55	0x01	Len	0x00	0x00	0x00	Address			

This command returns data read from the Flash memory. The length field can range from 0 to 255. This determines the number of bytes read from Flash and returned by the bootloader in the response.

Response:

`<0x55><0x01><Len><0x00><0x00><0x00><address(4 bytes)LSB··MSB><byte0><byte1>···<byteN>`

`Len` bytes are returned after the address field.

TABLE 3-7: WRITE FLASH

Byte:	0	1	2	3	4	5	6	7	8	9
Values:	0x55	0x02	Len	0x00	Key1	Key2	Address			

Byte:	`Len` Number of bytes (9 through 8+Len)
Values:	Data[0]-Data[Len-1]

This command writes data into the Flash memory. The length field can range from 0 to 255. This determines the number of bytes written to Flash. The Write Flash command does not erase any Flash memory before writing data; therefore, this command should be preceded by an Erase Flash command for proper operation.

Response:

`<0x55><0x02><Len><0x00><Key1><Key2><address(4 bytes)LSB··MSB><Status>`

`Status` is either a '`0`' indicating that the command failed, or a '`1`' indicating the command was successful

.

TABLE 3-8: ERASE FLASH

Byte:	0	1	2	3	4	5	6	7	8	9
Values:	0x55	0x03	Blocks	0x00	Key1	Key2	Address			

This command erases one or more blocks of Flash memory, starting at address *Address*. The Blocks field can range from 0 to 255. The 1-255 value of the Blocks field represents the number of blocks to erase. If the Blocks field is '0', the bootloader will erase 256 blocks.

Response:

<0x55><0x03><Blocks><0x00><Key1><Key2><address(4 bytes)LSB··MSB><Status>

Status is either a '0' indicating that the command failed, or a '1' indicating the command was successful.

TABLE 3-9: READ EE

Byte:	0	1	2	3	4	5	6	7	8	9
Values:	0x55	0x04	Len	0x00	0x00	0x00	Address			

This command reads data from the EEPROM memory located on the embedded PIC® device in the module beginning at address *Address*. Len can range from 0 to 255.

Response:

<0x55><0x04><Len><0x00><0x00><0x00><address(4 bytes)LSB··MSB><byte0><byte1>···<byteN>

Len bytes are returned after the Address field.

TABLE 3-10: WRITE EE

Byte:	0	1	2	3	4	5	6	7	8	9
Values:	0x55	0x05	Len	0x00	Key1	Key2	Address			

Byte:	Len Number of bytes (9 through 8+Len)
Values:	Data[0]-Data[Len-1]

This command writes data into the EEPROM memory of the embedded PIC device. The Len field can range from 0 to 255. This determines the number of bytes written to EEPROM. The Write EE command does not require any form of Erase command to be issued prior to writing data into the EEPROM.

Response:

<0x55><0x05><Len><0x00><Key1><Key2><address(4 bytes)LSB··MSB><Status>

Status is either a '0' indicating that the command failed, or a '1' indicating the command was successful.

TABLE 3-11: READ CONFIGURATION WORDS

Byte:	0	1	2	3	4	5	6	7	8	9
Values:	0x55	0x06	Len	0x00	0x00	0x00	Address			

This command reads data from the Configuration memory located on the embedded microcontroller in the module beginning at address *Address*. The Len field can range from 0 to 255. There are only 14 Configuration Words, and data repeats if the Len(gth) is greater than the number of Configuration Words.

Response:

<0x55><0x06><Len><0x00><0x00><0x00><address(4 bytes)LSB··MSB><byte0><byte1>···<byteN>

Len bytes are returned after the Address field.

TABLE 3-12: WRITE CONFIGURATION WORDS

Byte:	0	1	2	3	4	5	6	7	8	9
Values:	0x55	0x07	Len	0x00	Key1	Key2	Address			

Byte:	*Len* Number of bytes (9 through 8+Len)
Values:	Data[0]-Data[Len-1]

This command writes data into the Configuration memory. The Length field can range from 0 to 255. This determines the number of bytes written to the Configuration memory. The Write Configuration Words command does not erase any Flash memory before writing data; therefore, this command should be preceded by an Erase Flash command for proper operation.

Response:

<0x55><0x07><Len><0x00><Key1><Key2><address(4 bytes)LSB··MSB><Status>

Status is either a '0' indicating that the command failed, or a '1' indicating the command was successful.

TABLE 3-13: CALCULATE AND RETURN CHECKSUM

Byte:	0	1	2	3	4	5	6	7	8	9
Values:	0x55	0x08	Len	0x00	0x00	0x00	Address			

This command returns the checksum calculated over the Flash memory range beginning at Address for a length of Len bytes. If Len is odd, 1 is added to make Len even.

The checksum algorithm treats the Flash memory as an array of 16-bit values during the calculation, which is not performed byte by byte. In MPLAB® X, this is known as Checksum Algorithm 2.

Response:

<0x55><0x08><Len><0x00><Key1><Key2><address(4 bytes)LSB··MSB><CSumLSB><CSumMSB>

Status is either a '0' indicating that the command failed, or a '1' indicating the command was successful.

TABLE 3-14: RESET DEVICE

Byte:	0	1	2	3	4	5	6	7	8	9
Values:	0x55	0x09	0x00	0x00	0x00	0x00	0x00000000			

This command does not generate a response and immediately performs a software reset of the module.

Appendix A. Current Firmware Features and Fixes

Please check the product web page for the current RN2483 firmware version at www.microchip.com/lora.

A.1. Version 0.9.5

Initial release of the firmware.

A.2. Version 1.0.0

Release for LoRaWAN™ specification V1.0

- Added support for additional RN2483 commands:

  `mac set sync`

  `mac get sync`

  `mac set upctr`

  `mac get upctr`

  `mac set dnctr`

  `mac get dnctr`

  `sys set pinmode`

  `sys get pindig`

  `sys get pinana`

  `radio get sync`

- Added new parameters to be saved in nonvolatile memory whenever a `mac save` command is triggered

  LoRaWAN current data rate

  LoRaWAN RX2 window parameters (data rate and frequency)

  Adaptive Data Rate status

  LoRaWAN uplink frame counter

  LoRaWAN downlink frame counter

- Changed the default value for the LoRaWAN End-Device Identifier (deveui)
- Changed the valid range for the `radio set fdev` parameter to [0.. 200000]
- Changed the valid range for the `radio set bitrate` parameter to [1.. 300000]
- Changed `sys sleep` command behavior to not influence the GPIO configuration
- Changed the 433 MHz radio frequency band to [433050000 .. 434790000]
- Fixed an issue that may have caused the RN2483 module to mishandle data on LoRaWAN port 0
- Fixed an issue that may have caused the module to fail joining
- Fixed `radio get snr` command to display correct value

A.3. Version 1.0.1

Release containing modifications needed to successfully pass the LoRaWAN Certification Program testing in 868 MHz band.

- Added support for usage of the reserved ports, from 224 to 255 for transmitting and receiving Application Data
- Increased the size of the value returned by the `mac get status` command from 2 bytes to 4 bytes
- Updated the size of the LoRaWAN receive windows and the moment in time at which these are opened
- Fixed an issue that may have caused the RN2483 module to mishandle packets received with DR = 7
- Fixed an issue that may have caused the RN2483 module to mishandle the LoRaWAN RXParamSetupReq command
- Fixed an issue that may have caused the RN2483 module to mishandle the usage of LoraWAN ADRACKReq in packets

Worldwide Sales and Service

AMERICAS

Corporate Office
2355 West Chandler Blvd.
Chandler, AZ 85224-6199
Tel: 480-792-7200
Fax: 480-792-7277
Technical Support:
http://www.microchip.com/support
Web Address:
www.microchip.com

Atlanta
Duluth, GA
Tel: 678-957-9614
Fax: 678-957-1455

Austin, TX
Tel: 512-257-3370

Boston
Westborough, MA
Tel: 774-760-0087
Fax: 774-760-0088

Chicago
Itasca, IL
Tel: 630-285-0071
Fax: 630-285-0075

Dallas
Addison, TX
Tel: 972-818-7423
Fax: 972-818-2924

Detroit
Novi, MI
Tel: 248-848-4000

Houston, TX
Tel: 281-894-5983

Indianapolis
Noblesville, IN
Tel: 317-773-8323
Fax: 317-773-5453
Tel: 317-536-2380

Los Angeles
Mission Viejo, CA
Tel: 949-462-9523
Fax: 949-462-9608
Tel: 951-273-7800

Raleigh, NC
Tel: 919-844-7510

New York, NY
Tel: 631-435-6000

San Jose, CA
Tel: 408-735-9110
Tel: 408-436-4270

Canada - Toronto
Tel: 905-695-1980
Fax: 905-695-2078

ASIA/PACIFIC

Asia Pacific Office
Suites 3707-14, 37th Floor
Tower 6, The Gateway
Harbour City, Kowloon

Hong Kong
Tel: 852-2943-5100
Fax: 852-2401-3431

Australia - Sydney
Tel: 61-2-9868-6733
Fax: 61-2-9868-6755

China - Beijing
Tel: 86-10-8569-7000
Fax: 86-10-8528-2104

China - Chengdu
Tel: 86-28-8665-5511
Fax: 86-28-8665-7889

China - Chongqing
Tel: 86-23-8980-9588
Fax: 86-23-8980-9500

China - Dongguan
Tel: 86-769-8702-9880

China - Guangzhou
Tel: 86-20-8755-8029

China - Hangzhou
Tel: 86-571-8792-8115
Fax: 86-571-8792-8116

China - Hong Kong SAR
Tel: 852-2943-5100
Fax: 852-2401-3431

China - Nanjing
Tel: 86-25-8473-2460
Fax: 86-25-8473-2470

China - Qingdao
Tel: 86-532-8502-7355
Fax: 86-532-8502-7205

China - Shanghai
Tel: 86-21-3326-8000
Fax: 86-21-3326-8021

China - Shenyang
Tel: 86-24-2334-2829
Fax: 86-24-2334-2393

China - Shenzhen
Tel: 86-755-8864-2200
Fax: 86-755-8203-1760

China - Wuhan
Tel: 86-27-5980-5300
Fax: 86-27-5980-5118

China - Xian
Tel: 86-29-8833-7252
Fax: 86-29-8833-7256

ASIA/PACIFIC

China - Xiamen
Tel: 86-592-2388138
Fax: 86-592-2388130

China - Zhuhai
Tel: 86-756-3210040
Fax: 86-756-3210049

India - Bangalore
Tel: 91-80-3090-4444
Fax: 91-80-3090-4123

India - New Delhi
Tel: 91-11-4160-8631
Fax: 91-11-4160-8632

India - Pune
Tel: 91-20-3019-1500

Japan - Osaka
Tel: 81-6-6152-7160
Fax: 81-6-6152-9310

Japan - Tokyo
Tel: 81-3-6880- 3770
Fax: 81-3-6880-3771

Korea - Daegu
Tel: 82-53-744-4301
Fax: 82-53-744-4302

Korea - Seoul
Tel: 82-2-554-7200
Fax: 82-2-558-5932 or
82-2-558-5934

Malaysia - Kuala Lumpur
Tel: 60-3-6201-9857
Fax: 60-3-6201-9859

Malaysia - Penang
Tel: 60-4-227-8870
Fax: 60-4-227-4068

Philippines - Manila
Tel: 63-2-634-9065
Fax: 63-2-634-9069

Singapore
Tel: 65-6334-8870
Fax: 65-6334-8850

Taiwan - Hsin Chu
Tel: 886-3-5778-366
Fax: 886-3-5770-955

Taiwan - Kaohsiung
Tel: 886-7-213-7830

Taiwan - Taipei
Tel: 886-2-2508-8600
Fax: 886-2-2508-0102

Thailand - Bangkok
Tel: 66-2-694-1351
Fax: 66-2-694-1350

EUROPE

Austria - Wels
Tel: 43-7242-2244-39
Fax: 43-7242-2244-393

Denmark - Copenhagen
Tel: 45-4450-2828
Fax: 45-4485-2829

Finland - Espoo
Tel: 358-9-4520-820

France - Paris
Tel: 33-1-69-53-63-20
Fax: 33-1-69-30-90-79

France - Saint Cloud
Tel: 33-1-30-60-70-00

Germany - Garching
Tel: 49-8931-9700

Germany - Haan
Tel: 49-2129-3766400

Germany - Heilbronn
Tel: 49-7131-67-3636

Germany - Karlsruhe
Tel: 49-721-625370

Germany - Munich
Tel: 49-89-627-144-0
Fax: 49-89-627-144-44

Germany - Rosenheim
Tel: 49-8031-354-560

Israel - Ra'anana
Tel: 972-9-744-7705

Italy - Milan
Tel: 39-0331-742611
Fax: 39-0331-466781

Italy - Padova
Tel: 39-049-7625286

Netherlands - Drunen
Tel: 31-416-690399
Fax: 31-416-690340

Norway - Trondheim
Tel: 47-7289-7561

Poland - Warsaw
Tel: 48-22-3325737

Romania - Bucharest
Tel: 40-21-407-87-50

Spain - Madrid
Tel: 34-91-708-08-90
Fax: 34-91-708-08-91

Sweden - Gothenberg
Tel: 46-31-704-60-40

Sweden - Stockholm
Tel: 46-8-5090-4654

UK - Wokingham
Tel: 44-118-921-5800
Fax: 44-118-921-5820