



Application Note 51

Frequency Hopping Techniques

Rev 1.0

Introduction

The use of wireless radio connections is increasing. Operation in the license free bands (such as the 915MHz band in U.S.) should conform to FCC 15.247 regulations. "Frequency hopping" is an important part of these regulations. A system using "frequency hopping" will benefit from reduced multi-path fading, increased output power and reduced system-to-system interference. This document describes several aspects of the frequency hopping in the 915MHz band with emphasis on typical WPAN (Wireless Personal Area Network) properties (examples: low cost and low power). Micrel's MICRF505 IC solution is ideally suited for this type of operation.

Terminology

FHSS: Frequency Hopping Spread Spectrum.

Frame: A set of bits, formatted in a known way.

Beacon: A frame sent by a master device, used to maintain synchronization between a master and associated slave(s).

Power off mode: Completely turn power off.

Sleep mode: Some circuitry running, power consumption at minimum to maintain sync.

Awake mode: All circuitry running.

Polling: The process of switching between sleep and awake modes.

Master: A node knowing "correct time" and responsible for transmitting beacons. Also referred to as "coordinator".

Slave: A node that adjusts the clock to the "correct time" of the master.

RF unit, Node and Device: Used as a general term for a master or slave RF unit.

User: General term for the equipment connected directly to a RF unit.

Cluster: A combination of 1 Master and 1 or more Slave(s).

Acquisition: The process of getting a slave synchronized to a Master.

Synchronized jumping: Master and Slave(s) jump to a new (and the same) frequency at the same time.

Frequency Hopping

The parameters of a FCC15.247 frequency-hopping system include:

- Bandwidth (“20dB bandwidth”)
- Number of hopping frequencies
- Average time of occupancy
- Output power

These parameters are connected to each other:

1. If the 20dB bandwidth of the hopping channel is less than 250kHz:
 - a. Number of hopping frequencies must be > or = 50
 - b. Average time of occupancy on any frequency < or = 0.4 sec in a 20-second period
2. If the 20dB bandwidth of the hopping channel is at least 250kHz:
 - a. Number of hopping frequencies must be > or = 25
 - b. Average time of occupancy on any frequency < or = 0.4 sec in a 10-second period
3. If 1) or 2) is followed, output power can be as high as 1 watt (50-channel-case) or 250 mW (25-channel-case). If frequency hopping is not used, output power should be less than 1 mW (approx 0.75 mW) (FCC 15.249 regulations).

The following provisions are common to cases 1) and 2):

- The system shall hop to channel frequencies that are selected from a pseudo-randomly ordered list.
- On average, each frequency must be used equally.
- Systems are not required to use all available hopping frequencies for each transmission (note

the keyword “on average”).

Note that this is not a complete list of requirements. In this document, the emphasis will be on the requirements listed above.

The advantages of frequency hopping include:

- Multi-path reflections may cause fading (or even amplification). Fading can be a problem. For a single frequency, the signal strength can be reduced to a not-detectable level at some points in space. However, another frequency will be reflected differently; that is: points in space with reduced signal strength are different for the next frequency.
- Due to the random jumping, several clusters (1 master and at least 1 associated slave) can operate at the same time. Every cluster follows its own jump pattern (that is: Master and all its associated slaves are jumping synchronously). If cluster A and cluster B experience a frequency collision, it is most likely that the next frequency for A and B will be different. Or, if even that one is equal, it is most likely that the next is different and so on.
- It is more difficult to intentionally disturb (jam) a system, and it is more difficult to monitor the communication flow.

Implementation

The user application determines how to implement the frequency hopping. Below, some points to consider are described. Finally, an example is given in “Frequency Hopping Example”.

1.1 Parameters

The implementation of the frequency hopping must be based on the user’s application. Typical parameters (system-level parameters) to consider are shown in Table 1.

Parameter	Issue	Trade-offs
Cost (price)	Low-ppm xtal for microcontroller	Time between beacons decrease if high-ppm crystals are used Accuracy of synchronization is less for a high-ppm crystal
	Microcontroller with large Program and Data memory	A large memory makes it possible to include sophisticated features (example: generate frequency jump-tables based on other RF activity in the area)
Power	Time between beacons (sync-info)	Increasing time between beacons reduce power consumption
	Stay in sync all the time or re-sync when needed	A device can be powered-off to save power, and then powered on when needed. A new synchronization may be

		needed Time delay for data throughput is decreased for an always-in-sync system
Data Rate	Use a large "beta" (large deviation compared to bitrate) for low data rates	The number of frequencies used can be reduced if bandwidth is increased

Table 1. Parameters

1.2 Type of System

At least two types of frequency hopping systems can be identified:

1. "Ad-hoc": A link is established between two devices, data is exchanged between them and the link is terminated. Typically, RF units are powered off. Depending upon the system, it is or it is not a "Master" and a "Slave".
2. "Always in sync": 1 master and 1 or several slaves are always in sync. Master transmits beacons (sync-info) regularly to maintain sync. RF units may be in a power-saving "sleep" mode during operation.

Note the difference between "power off" and "sleep". In power-off, power consumption is at an absolute minimum level, and sync is lost. In sleep, some circuitry is powered and sync is maintained.

Some applications may use a combination of the two systems. At regular intervals, power-on the RF units and stay powered for a long time. Maintain sync while powered on. (Still, RF units may enter into a low-power sleep mode). Then, the RF-units are completely powered-off. Example of a "combined system": on a daily basis, updates nodes in a network with relatively large amounts of data.

1.2.1 Ad-hoc

For systems with a very low power-on cycle, an "ad-hoc" approach can be used. An example of a typical ad-hoc frequency hopping system: measuring temperature on a one-time-per-day basis.

For these type of systems, power consumption may be a critical parameter. And it is advantageous to power-off as much circuitry as possible, including the RF part.

When powered-on, the RF units shall use the frequencies randomly and follow the average time of occupancy regulation. That is, a procedure for getting synched must be carried out. The additional "cost" (power and time delay) must be compared to the savings achieved by the complete power-off possibility.

Advantages:

Power-effective system

Nodes do not depend on sync-info from a "master"; they can establish a link that is independent of all other nodes

Disadvantages:

Increased time to transfer data due to the initial establish-link procedure

1.2.2 Always-in-Sync

Several applications have a critical parameter: Time-delay for data throughput. An example is an alarm system where a "user" must be told as quickly as possibly that an abnormal situation has occurred.

To reduce power consumption, some circuitry can be powered off, and then on again. The MICRF505 can enter into a low-power sleep mode ideally suited for this.

The polling rate (awake mode/sleep mode) is determined by the system specifications. The extreme case is to be awake all the time (to minimize data throughput time delay).

A "Master" – "Slave" system can be implemented, where all slave-to-slave communication is through the master. Or a "Super-Master" system can be made; 1 RF unit is "super-master" responsible for maintaining sync, the other RF units is capable of addressing each other directly.

Advantages:

Quick transfer of data; this is because the nodes are in sync when they want to exchange data.

Disadvantages

The system is based on a master tx'ing beacons; if master breaks down, a back-up mechanism could be implemented.

Nodes need to stay awake; or at least wake-up at regular intervals to get or send beacons.

1.3 Stages in Frequency Hopping Communication

There are two typical stages (this applies to both "Ad-hoc" and "Always-in-sync" systems):

1. Acquisition. Get the communicating nodes (like master and slave) in sync with each other. That is, establish a communication link. This step is performed after power-on (of master, slave or both) or when sync is lost (example: nodes are moved out of radio range of each other for a “long” time).
2. Maintain synchronization. For an “Ad-hoc” system, the sync is kept only for a limited time; then the link is closed. For an “Always-in-sync” system, the link is kept “forever” (i.e., as long as the devices are powered).

1.3.1 Acquisition

In this sub-section, nodes are referred to as “Master” and “Slave”. After power-on, or when the sync is lost, the synchronization must be re-established. In addition, the Master only or a single Slave can be powered off and on again, or a new Slave can be entered.

If a master has several slaves attached, all slaves will loose sync if master is power-cycled (turned off and then on again).

There are several possibilities for the acquisition stage. These may be used individually or combined. In general, application specs must be used as guidelines.

1. When Master is powered-on, the Master can transmit “beacons” for a long time. The master should transmit beacons to ensure all slaves have received at least one beacon. This will solve the initial power-on case as well as later “power-cycling” cases. Refer to Figure1.
2. If a slave is powered-on after Master has finished tx’ing the initial beacon-sequence, or if sync is lost, the Slave can request a beacon from Master. Refer Figure 1.
3. If a slave has lost sync, it can stay “passive” (i.e., not transmitting a beacon request) and searching a “long time” on every frequency. It should search long enough to assume Master has transmitted a beacon on the present frequency. Then jump to next frequency. In this case, rx-side jumps slowly, and the tx-side jumps quickly.
4. Conversely, the slave can also stay “passive” a very short time; just long enough to detect if there is a transmission going on or not. In this case, the rx-side jumps fast and the tx-side jumps slowly – and transmits a “long” preamble (such as a 1010...pattern) - long enough for the rx-side to scan all frequencies for RF activity. This method is best suited for applications where a short lock-time is possible. For “high” data rates, the loop-filter can be fast and the lock-time short. For “low” data rates, the lock time will be higher and the preamble must be longer for the rx-side to scan through all

frequencies.

The frequency hopping can be done based upon a frequency table (refer to “Maintain Sync” section below). After acquisition, the nodes will jump to a new frequency at a regular rate. If a node is power-cycled, it does not know if the “other part” is in sync. Therefore, during the acquisition step, the node can jump following the same frequency table, but at a different rate:

After master is powered on, it transmits a beacon; then it immediately changes frequency and transmits a beacon on the next frequency and so on. If the master transmits initial beacons and changes frequency at a rate x times the “normal rate”, it will eventually reach the “slower-hopping” slave(s).

After a slave is powered-on, it can transmit beacon-requests in the same way; eventually, it will reach the “slower-hopping” master.

Probably, an “active” slave (requesting beacons) is the best choice for most applications. To reduce RF activity, the slave can transmit a request then wait a random time for response. And, if made a random number of attempts, be “passive” a randomly long time.

Below, a typical low-level protocol is described. Note the terms “Sync” and “Freq Index” and “Timer value when Sync was sent”.

Low-level frame:

Preamble	Sync	Length	Low-Level Payload	CRC checksum
----------	------	--------	-------------------	--------------

Preamble: a series of 1010... (used in bit-synchronizer)

Sync: A series of 11001100... (used to identify end of preamble)

Length: Number of bytes in frame, not including Preamble and Sync.

Low-Level Payload: Depends on type of frame, see the sequence below.

CRC checksum: Result of CRC calculations.

Low-Level Payload:

If frame is from Master:

Frame type	Level 2 Payload	Freq Index	Timer value when Sync was sent
------------	-----------------	------------	--------------------------------

If frame is from Slave:

Frame type	Level 2 Payload
------------	-----------------

Frame type: Used to identify the frame as “beacon”, “beacon-request”, “ack” or “data”.

Level 2 Payload: Typically, constructed by User-levels (i.e, the frame to be exchanged between Level 2 entities in Master and Slave User Units). For example, if Input is simply text to transfer, then Level 2 Payload will be a string of characters that is transmitted/received.

Freq Index: Present value of the index to use in Frequency table.

Timer value when Sync was sent: The captured value of

timer when Sync was completely tx'ed. Slave(s) uses this info to adjust it's internal timer.

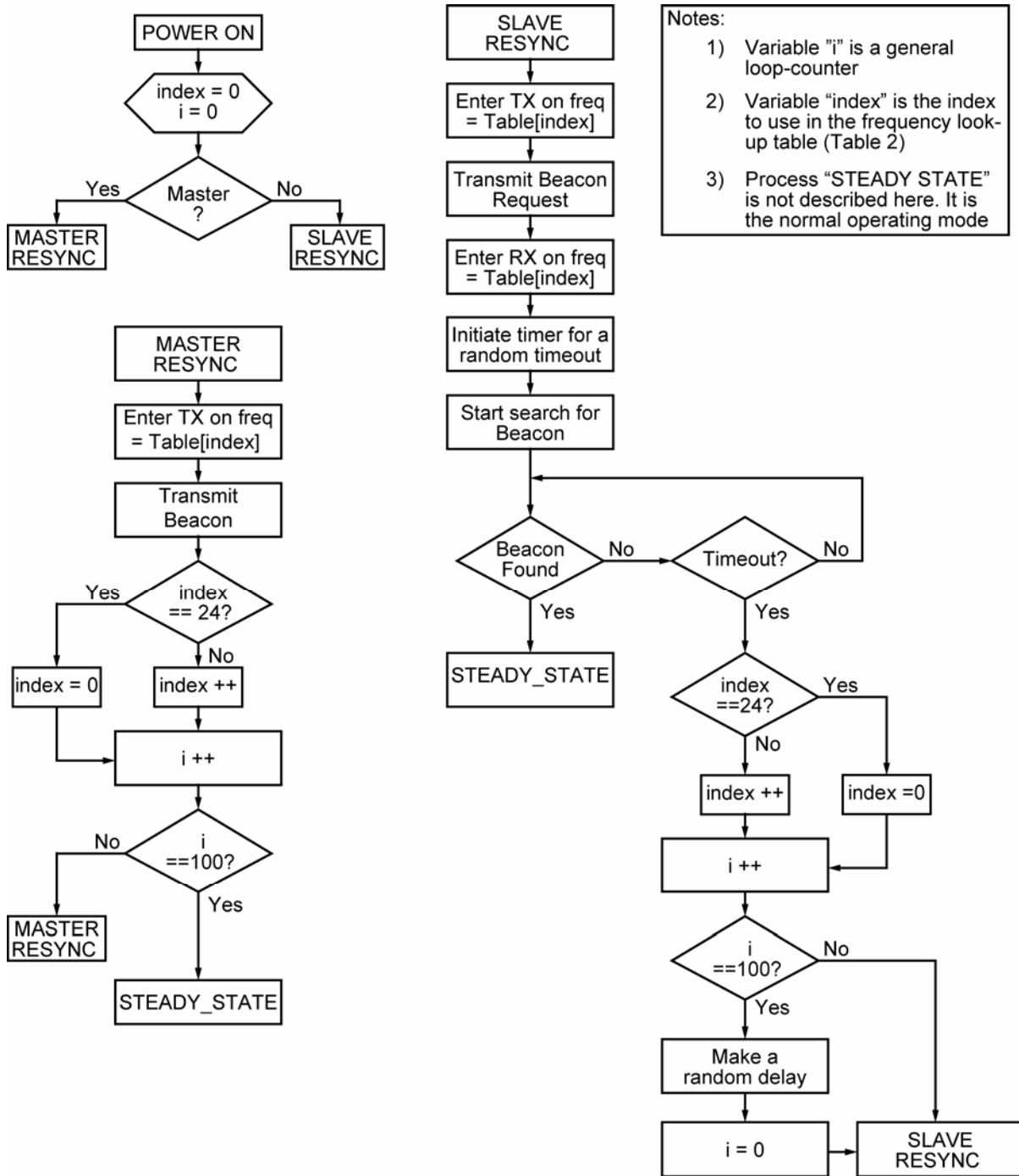


Figure 2. Flow chart of Master and Slave Program Execution After Power-On.

1.3.2 Maintain Sync

When the master and slave(s) are synched, they must know both when and where to jump. They must be both time and frequency synched.

Different methods can be used to achieve frequency sync:

- The units can agree on a common frequency table.
- The master can transmit the next x frequencies to use. Additional overhead must be included in the frames from master.
- The master includes a "seed" in transmitted frames. This "seed" will dictate the next frequencies used.

The frequency-table method is typically preferred in many applications. This is described in more detail below.

- The frequency table can be a prefixed, randomly ordered table of frequencies. Every frequency can appear 1 or more times in the table. In many applications, this will be sufficiently "random".
- To achieve sync, all nodes must know the present index of the table. This index can be transmitted from master in beacon frames (or in all frames from Master).
- An example of a frequency table is given below.

Index	Frequency Number	Index	Frequency Number
0	20	13	12
1	8	14	19
2	7	15	9
3	21	16	24
4	1	17	15
5	2	18	5
6	13	19	16
7	0	20	10
8	22	21	17
9	4	22	6
10	14	23	11
11	23	24	18
12	3		

Table 2. Frequency Table

To increase "the degree of randomness", implement features like the ones described below.

- The frequency table can be made based upon the present time. That is, when a sequence of all 25 (or 50) frequencies is through, the next sequence can be constructed based on the present time.
- If the RF units have a unique ID, then the frequency table can be made based upon this ID as well. Then, different RF units will have different jump patterns. For a cluster (1 master and 1...n slaves) the frequency table of the master can be used for all nodes; making the master-ID a "system ID".

It is important to know where to jump. And a unit must know when to jump as well. That is, the units must be time-synchronized. Two points of interest arise:

1. When to jump
2. How to synchronize timers

Generally, a node must know (or get from his "user") "correct time". Typically, a "Master" in a master-slave system will know the present time. The time units used in "correct time" should be selected as appropriately as possible. Typically, the units can be "timer-ticks". For example, if a 32.768kHz xtal is used as the source of "correct time", then a 16-bit or 24-bit register counting "32768-ticks" can be used as timer register. Many microcontrollers have a low-power state of operation where the 32768 xtal oscillator is running.

There are at least three possible methods for deciding "when to jump" (refer to text below). Method number one can be used if a continuous transmission is necessary, methods number two and three can be used for a packet-switched type of system (i.e., data is split and transmitted as "packets". Each packet contains a maximum number of data bytes).

1. Always change frequency at a specific time. If a frame is being transmitted, then stop transmitting, hop and start transmitting again. This method requires the transmitter and receiver to be synchronized with a high degree of accuracy. In addition, the method must assume a fixed "time to jump" (i.e. duration of switch-time from one frequency to another), which is not always the case –implying that the longest "hop-time" must be used for all hops.
2. Transmit a frame only if there is enough time to completely transmit the frame before the next hop-time. This method is acceptable if the system is jumping relatively slowly and has relatively short frames.
3. Start to transmit a frame at the "present frequency", and transmit the frame completely, even if a hop-time border is crossed. The user must make sure the "average time of occupancy" regulation is followed (this will limit

the maximum length of the frame).

The application determines the accuracy of the time synchronization. As a general rule, a “high” hop-rate should have a better accuracy than a “low” hop-rate. If a relatively low accuracy is used, then it will be necessary to include a “window” around the hop-time border. In this window, a transmission should not be started. Example: If the timers in two RF units are running 1 msec differently, a RF unit should not start a transmission if there is < 1msec since the hop-time.

To synchronize the timers in a master-slave connection or within a cluster, one of the following methods can be used:

1. Master is always transmitting at a particular point in time (example: master always starts a transmission x msec after hop-time). A slave will adjust the internal clock based on when a frame from master is received
2. Timing information is contained in the frames from master. Master captures the “correct time” when e.g., sync is completely transmitted. This value is included in the frame. A slave captures the value of its internal timer when sync is received. And, if the frame is received OK (e.g.: passing a CRC test), the internal timer is adjusted, based upon the difference between the received “correct time” and the captured time when the sync was received.

Frequency Hopping Example

In the preceding sub-sections, different parameters were discussed. Below, a combination of parameters is presented. This can be considered as a typical low-cost specification for a frequency hopping system.

Optional: Refer to the source code examples for the development system (visit www.micrel.com).

- System spec
 - One Master and one Slave.
 - Both Master and Slave can initiate a transfer of data. From the RF units' point of view, data transfers start randomly.

- Master and Slave should always be in sync.
- In this example: 25 different frequencies are sufficient to fulfill current FCC regulations.
- How to get synched
- Master will transmit beacons after power-on, at regular times or when requested by the slave.
 - Master includes timing info in all frames. That is: “Beacons” are only tx'ed if there is no RF activity.
 - Slave requests a beacon after power-on or when sync is lost.
 - All frames from Master contain 1) Frequency table index and 2) Timer value when sync-pattern is tx'ed.
- Time sync
 - Master includes 2 bytes of timing info in all frames, equal to the timer value when sync is tx'ed.
 - Slave captures value of internal timer when a 'sync' is received. After receiving an OK frame, the internal timer is adjusted with (received sync time - captured sync time).
- Frequency jumping
 - A randomly ordered table of 25 frequencies is stored in both Master and Slave.
 - A table index is included in frames from Master.
- When to Jump
 - A frame is completely tx'ed on the present frequency, even if a hop-time border is crossed. Both Master and Slave can initiate a transfer.

MICREL, INC. 2180 FORTUNE DRIVE SAN JOSE, CA 95131 USA
TEL +1 (408) 944-0800 FAX +1 (408) 474-1000 WEB <http://www.micrel.com>

The information furnished by Micrel in this data sheet is believed to be accurate and reliable. However, no responsibility is assumed by Micrel for its use. Micrel reserves the right to change circuitry and specifications at any time without notification to the customer.

Micrel Products are not designed or authorized for use as components in life support appliances, devices or systems where malfunction of a product can reasonably be expected to result in personal injury. Life support devices or systems are devices or systems that (a) are intended for surgical implant into the body or (b) support or sustain life, and whose failure to perform can be reasonably expected to result in a significant injury to the user. A Purchaser's use or sale of Micrel Products for use in life support appliances, devices or systems is a Purchaser's own risk and Purchaser agrees to fully indemnify Micrel for any damages resulting from such use or sale.

© 2006 Micrel, Incorporated.