wireless control for everyone

ONE-NET CONFIGURATION OPTIONS, FEATURES, AND PORT CONSTANTS GUIDE

Version 2.3.0

August 14, 2012

Table of Contents

1 Overview

ONE-NET provides support for a wide variety of networks and devices. When setting up a ONE-NET network, one should consider what types of devices will be in this network and how the network will be used. This document discusses different options for devices and networks and the corresponding options in the code that need to match these options.

2 Intended Audience

This document is geared towards both programmers and non-programmers. Certain sections will only be of interest to programmers. Network designers should also read this document to know what the issues are and what decisions need to be made.

3 Issues To Consider When Designing a ONE-NET Network

Several issues need to be considered when designing a ONE-NET Network. Certain options and values should be defined based on these considerations. In addition, code can be optimized based on the network type, and certain application-level functions can either be left out or must be written. Some of the issues and questions, in no particular order, are as follows.

1. How crowded is the radio traffic? An area with lots of traffic is going to result in far more lost messages and the need to handle lost messages will increase.
2. How important is it that messages are not lost? In some cases the consequences of a lost message will be quite severe. In other cases, it will not be a big problem. The answer to this question will determine whether a message that does not get acknowledged will be sent again.
3. How many devices are in the network? How often do most devices send messages? The answers to both of these questions will have a big effect on how crowded the radio traffic is. A key change in a network with two devices can easily propogate through the entire network quickly. A key

change in a network with a thousand devices, most of which are asleep the majority of the time, will be much more complicated?

4. What are the devices' resources? Do all devices have the same resources? A simple client with 16K code space and 1K RAM and a transceiver that can only communicate at 38,400 baud cannot do what a more advanced device with 128K code space. 8K of RAM, and the ability to switch data rates can do. If all devices have the ability to function as multi-hop repeaters and switch data rates, life is much easier. If a simple device cannot switch data rates and must be able to communicate constantly with a device that can switch data rates, the device that can switch data rates will not be able to even though it can. The simple device will also be unable to handle multi-hop messages so that the range of devices that it can communicate will be smaller. The simple device will not be able to handle sophisticated timing messages so all network communication must take this into consideration. This may or may not be problematic, but it is something the programmers and network designers must consider. ONE-NET offers many advanced communication options. Simple devices with low resources will not be able to use all of them.
5. Do any devices sleep? How critical is battery life? ONE-NET is generally geared towards meeting the needs of low-resource battery-powered sensors which are asleep the vast majority of the time and wake up once an hour or once a day, report their status, and immediately go back to sleep. The prime consideration is to minimize battery consumption. This can have a drastic effect on the rest of the network. One, the radio waves must be clear when these devices check in order to minimize lost messages and hence preserve battery power. Two, devices which are primarily sleeping cannot receive messages when they are asleep, so administration messages like key changes must consider this. This may or may not have an effect on the other devices. If you have an intercom, for example, that device will be initiating many stream messages. If a stream message is initiated during the same period that a sleeping device is about to wake up, the master may decide to disallow the stream message because that message will interfere with the sleeping device's status report. The network designer must decide which message is more vital or if it matters. There is no correct answer. It will depend on the needs of the network.
6. How important is security? How important is it that messages get sent and received immediately? One of the primary concerns of ONE-NET is the prevention of replay attacks. These are vital for applications such as locking

and unlocking locks. It is generally not nearly as important for operations such as reporting humidity or turning up the volume on a stereo. One of the primary ways that ONE-NET prevents replay attacks is to make sure that no two legitimate commands are sent with the same payload. The main way this is done is to change the message ID after each command. When message IDs are about to run out, a device will request a key change and reset the Message IDs to a lower value. The key change must propagate to the entire network. During that propagation, potentially two devices might not be able to communicate with each other because one has the new key and the other does not. The network designer must decide how important replay attacks are and determine whether lost messages due to key changes outweigh this concern.

7. How far away are the devices from each other? How long are the packets that need to be sent? Long packets reduce the range of a wireless message. If devices are close to each other, this will not be a problem. If devices are not close to each other, one or more client repeaters may be needed to transmit the message. In addition, devices with low resource may not be able to handle long packets. Any devices which communicate with a device that cannot handle long packets needs to know this.
8. How important is portability? Is this a "contained" network? If a ONE-NET network is going to be filled entirely with devices from the same manufacturer, the network designer can make all sorts of data messages and parsing mechanisms that are not specified by the ONE-NET protocol and all devices will know how to handle these messages. The goal of ONE-NET is generally to make the messages as universal as possible, so that a door/window sensor from one company can communicate with a wireless chime from another company and both can communicate with a master controller from a third company. To do that, all messages must comply to one of the ONE-NET defined message types and parsing mechanisms. There is a tradeoff. The ONE-NET specification allows for a significant variety of define messages, but there will still be times when a developer wants to add an internal message type undefined by ONE-NET in order to make the network run better. Doing so takes portability away. The network / product designer must take this into consideration.
9. What is the topology / "shape" of the network? Does almost all communication involve the master? Is the master as uninvolved as possible? If multi-hop communication is involved? If so, what devices are within range of what other devices and what devices can serve as repeaters? Is the

network largely unchanging or are nodes moving in and out of range or constantly added and removed?

10. How intricate are the timing of the messages? Are messages obsolete if delayed or should they still be sent, even if late? What are the ramifications?
11. Is ONE-NET the right protocol? ONE-NET is not a good fit for all applications. It has a much slower data rate than other protocols. If you need lots of data transferred quickly and reliably, you may need to consider another protocol. Similarly, if your network is quite large and needs routing tables, you may need to consider another protocol. ONE-NET is excellent for networks with low power needs, long range needs, where the bulk of the messages are short messages.

4 Examples Of Networks And How The Questions Above Are Relevant

Below are some examples of possible ONE-NET networks that can be set up and the decisions that are made based on their needs.

4.1 Smart-Home Setup On A Farm

Assume a fairly wide variety of devices, the vast majority of which are simple devices which do not sleep. The setup involves a master device located inside of a house, quite a few battery powered sensor devices that report rarely, an automated, programmable sprinkler devices that turn on and off, some intrusion detection systems, intercoms, some wireless devices that open and close windows, and some other wireless devices like stereo systems, heater, and air conditioning that are controlled through the network. In addition, there are several greenhouses. Assume that the peripheral devices are bought independently "off the rack" at stores like Radio Shack and are not all designed by the same manufacturer.

- There are battery powered sensors in remote and inaccessible locations that report their temperature and humidity every hour.
    - "Remote" and "Inaccessible" means that battery life is important and multi-hop is required.
    - The data they are sending is fairly simple, short packets, and the traffic is light. There are few, if any security concerns. The

consequences of a delayed message are close to 0, and the rare missed message is not too big of a deal either.

- The devices that open and close windows have security concerns and are good candidates for replay attacks. These are the devices that need to protect against replay attacks. They will want to ensure that a key change occurs when message IDs are about to be repeated.
    - Lost or delayed messages are not a big deal. We assume that a human being will simply press a button a second time if the message is lost.
    - Power consumption is not an issue. A power source must be nearby to provide the electricity to open and close the windows.
- The intrusion detection system devices like door / window systems are important. It is vital that any lost messages are re-sent, and in a timely manner. Lost messages mean that a break-in may not be reported. Delayed messages are also problematic.
    - Therefore any lost message must be re-sent. Therefore also no other device can be allowed to hog the airwaves for an extended period of time.
    - These will generally be very rare, short messages to the master.
    - They may be at remote barns, so repeaters are required.
- Sprinkler system messages are fairly rare, short, simple messages. Delayed messages are unimportant. Lost messages might be problematic and cause flooding, but will generally not be considered critical. Security is not an issue.
- The messages to and from the greenhouses will be fairly important, but fairly rare and short. Generally these messages will be to report and adjust lighting, temperature, and humidity values based on some agricultural schedule controlled by the master. Security is not an issue.
- The Home Control Devices such as turning on and off the stereo, changing channels, and adjusting volume, are controlled by a human. Lost and delayed messages are not a problem since the person will simply reissue commands. Multi-Hop is not an issue due to the short distance. The master need not be involved in the communication. Security is not an issue.
- The intercom system must pause occasionally and not hog all of the bandwidth in order to allow other traffic to go through.

- Multi-Hop Repeaters must be available at all times.
- Livestock may occasionally send their GPS positions to the master. An alarm might be sounded if one of the livestock leaves its assigned area.
- No non-ONE-NET specified messages are required.

4.2 Toy Train Set Using Remote Controls To Control The Trains And The Rails

Imagine a toy train set where a child has a remote control that can cause the trains to toot, speed up, slow down, set off steam, open and close a drawbridge, etc. Think of what is needed and what is not needed.

- Multi-Hop is not needed. Everything is within a few feet of everything else.
- Peer To Peer communication is not needed. All messages will be to or from the master remote control.
- Security concerns are non-existent. The risk of replay attacks is irrelevant. Potentially the child's mean older brother could hijack the system and cause the train to derail, but other than hurt feelings, no damage is done.
- An occasional block message might be needed if perhaps there is a whistle sound that the child can upload from the remote control to the train.
- No non-ONE-NET-specified messages are required. Generally all messages are quite short, generally either on/off messages or speed / percent messages.
- Conserving battery power is not a big deal. No devices will sleep.
- Lost messages will simply be re-sent. Lost messages due to congestion will not be a factor likely. The remote will send one message at a time
- All in all, this is a simple network that will work quite easily with ONE-NET

4.3 Retail Store Using Intercoms and Door Sensors To Alert The Employees

Imagine a retail store where the employees spend time in the back room and need to be alerted when customers arrive.

- The number of devices will be small. Perhaps one door sensor for every door, plus a motion sensor or two, plus one or two chimes.
- Peer To Peer communication will likely be used. Whenever a door sensor is triggered, it will send a message directly to a chime, which will make a sound.
- Messages that are not acknowledged should be re-sent. A slight delay is not a problem. It's better to chime twice than not at all. A missed message means that a customer goes away.
- Security concerns are probably not relevant. This is a chime to alert the employees of customers during business hours. Replay attacks are not a factor.
- Multi-Hop is likely not needed unless you perhaps have a large warehouse.
- Devices will not sleep. Battery consumption is not a concern.

4.4 Laser Tag Game

Imagine a laser tag game where each participant wears a device that monitors the participant's location, the amount of ammunition remaining, whether the participant is wounded, and allows constant updating of these values to be sent to the referee, fellow teammates, and perhaps the opposing team.

- Traffic level will be quite high. It's likely that each participant will be sending the information every few seconds.
- Delayed messages of location are useless. They should not be resent because this is a real-time gaming situations.
- Ditto many of the other messages. They quickly become obsolete.
- Due to the high traffic and the need for highly reliable, quick communication, a fairly intricate timesharing protocol may need to be designed. A token ring might be appropriate. In addition, there will be quite a few broadcast messages rather than messages directed to individuals.

- To allow for peer to peer streaming messages and at the same time to avoid hogging all of the air, switching channels when sending stream and block messages will be required.
- Multi-Hop will likely be infeasible and unnecessary.
- Security concerns may or may not be applicable (although of course some gamers take gaming extremely seriously and might resort to all sorts of electronic warfare!).
- To increase the speed of communications, higher data rates may be needed.
- All in all, even with higher data rates, ONE-NET might not be fast enough to satisfy the "real-time data" needs of serious laser tag gamers. Wi-Fi would probably be a better fit.

4.5 Smart-Home or Smart-Office Network In A Crowded Urban Environment

Many of the issues involved with the Smart-Home setup on a farm will apply here. However, there will be some differences.

- The airwaves will be much more crowded from traffic using the same frequencies.
- Buildings and other structures will reduce the range.
- Generally things will be closer and hence Multi-Hop will not be needed.
- There will be fewer devices in inaccessible areas where long battery life is the primary concern.

4.6 Three Light Switches Tied To Nine Lights

Master is device 001. It sets up the network then immediately gets out of the way. A device contains three switches. There are three banks of lights, each containing three lights. Each switch controls one back of lights.

- This is a perfect example of peer messaging. Each light switch will be peered with the light units the switch controls.
- Which switches controls which lights can be changed at any time by changing the peer table.

- The light switches may or may not be battery operated and sleep. This will depend on the overall design. The light switches will not sleep.

5 Queue Levels And Capacities

ONE-NET uses an outgoing message queue for outgoing messages. There are four "levels" of queue functionality. Devices should be assigned a queue level and capacity based on their resources and their needs and the needs and resources of the network. A high queue level may be needed for a device to be able to handle some of the more advanced options regarding timing that are intended to reduce message collisions. Higher queue levels and capacities require more code space and RAM. If the device can spare the code space and RAM, it should use the highest level queue as well as a high queue capacity for better performance.

1. Level 0 – No queue is used at all, or the queue size is 1, which means no messages can be added to the queue. Resources to send a message are either immediately available and messages are sent immediately or messages cannot be sent at all.
2. Level 1 – A priority queue is used, but whenever resources are available, messages must be sent immediately.
3. Level 2 – A priority queue is used and messages can be set to be sent at some future time.
4. Level 3 – A priority queue is used. Messages can be sent immediately or at a future time. In addition, messages can be assigned an "expiration" time meaning if they have not been sent by that time, they will not be sent at all.

Queue Capacity is irrelevant for Queue Level 0. For Queue Levels 1 through 3, a queue capacity must be specified. That capacity is the maximum number of messages that can be held in the queue. In addition a queue buffer size must also be specified. This buffer must hold the payloads of all messages in the queue.

6 Peer Functionality And Peer Table Capacity

Please see the document **ONE-NET Peer Messaging** for more information on how peer messaging can be used. Generally peer messaging is used to tie the status of

one unit of a device to another unit of another device. The most obvious examples of this are light switches and volume control devices. A light switch will be assigned to control one or more lights. The lights that the switch controls are considered its peers. When the switch is flipped, the light switch quickly goes through its peer table and sends messages containing its new status to all of its peers. Those peers, when they receive the message, set their state to the light switch's state. Similarly with a volume control device, the when the volume control knob is adjusted, it will send its new value to all of its peers, which will be speakers. The speakers will adjust their values to equal the volume control knob's status.

Peer functionality must be defined or undefined in a file called config_options.h (see section 8). In addition, if peer functionality is used, the size of the peer table must be defined. This will be based on the maximum number of devices that the device might peer with and RAM considerations.

7 Other Sizes Which Must Be Defined

Several more sizes must be defined. Among these are the following.

7.1 Recipient List Size

Each outgoing message will have one or more recipients. A recipient list will be stored in memory. The size of that recipient list must be defined at compile time. The size of the recipient list must be at least large enough to accommodate all units peered with the relevant unit (please see **ONE-NET Peer Messaging** for more details), plus room in general at least two more device / unit combinations: the master device and at least one other client device. If you have enough RAM available, you should make the recipient list size large enough to be more than big enough to accommodate all possible recipient lists.

7.2 Client List Size / Maximum Number Of Clients (Master-Capable Only)

All master devices maintain a table of all client devices in the network in an array. The size of this array is the maximum number of clients allowed in the network at any given time and must be defined.

7.3 Client Device Information List Size (Client-Capable Only)

All client devices keep track of the device information of other client devices. If the list does not have enough room, the two client devices need to "re-sync" their features and message IDs. In addition, if the two devices are not within range of each other, the number of hops involved in communicating between the two devices will be lost as well. This adds extra messages to the network and potentially also increases vulnerability to the system, so clients store other clients' information to avoid re-syncing. The number of clients that a client can keep track of must be defined.

7.4 Number Of Devices That Should Not Be Ignored (Relevant Only If RANGE_TESTING is defined)

Please see the **ONE-NET Multi-Hop And Routing** and **ONE-NET Debugging Tools And Techniques** documents for more information.

If RANGE_TESTING is defined for whatever reason (either as part of the debugging process or for some other reason), certain devices will be considered out-of-range even if they are not out of range. ONE-NET will keep a list of devices which it will not ignore. The size of this list needs to be defined. Again, this value is relevant only if RANGE_TESTING is defined.

7.5 Input / Output Text Buffer Lengths (Relevant Only For Devices With A Command-Line Interface Or Some Other Method Of Text Input And Output)

Any device that accepts input or sends output in the form of data from / to an external source / destination such as a serial cable or USB cable must define the maximum sizes of the input / output buffer.

7.6 ONE-NET Dynamic Memory / Heap Size (Relevant Only If ONE_NET_MEMORY is defined)

Using C commands like malloc and free to set aside dynamic memory can be problematic in many embedded systems. Thus ONE-NET provides an alternate simple option of managing dynamic memory for those who want it. If you choose

to use this feature, you must define ONE_NET MEMORY and define both the maximum number of heap entries you would like as well as a value for the overall number of bytes of memory that can be reserved by ONE-NET. Note that this is entirely separate from any dynamic memory management that you may use that is defined in the CString library or provided / defined at the chip level through an Integrated Development Environment or elsewhere.

8 Configuration Options (config_options.h file)

Several ONE-NET files need a file called config_options.h, so the developer must supply one. Within the config_options.h fikle, many options and capabilities must be defined or undefined. Some have been discussed previously. The following list includes some of the options that should be defined or undefined. The best way to define the options is to use the Evaluation Board project config_options.h file as a skeleton and modify it. Certain options only make sense if certain other options are defined or not defined (for example, a device cannot be a client repeater if it is not also a client)

1. Master or Client
    a. ONE_NET_MASTER – define this option if this device will ever function as a master device.
    b. ONE_NET_CLIENT – define this option if this device will ever function as a client device.
    c. Either ONE_NET_MASTER or ONE_NET_CLIENT must be defined. Generally devices will have only one of these values defined. There are some devices, however, that sometimes switch back and forth between master and client. If the device does this, then both ONE_NET_MASTER and ONE_NET_CLIENT should be defined.
    d. ONE_NET_SIMPLE_CLIENT – define this if the device functions as a “Simple Client”. See the ONE-NET Specification document for the definition of a “Simple Client”.
2. Compile Without Warnings
    a. COMPILE_WO_WARNINGS should be defined if you are compiling using some of the more strict options like –W, -Wall, or –Werror in

the gcc or similar compilers. This will increase the code size slightly. Generally this option is unneeded in embedded systems.

3. Queue Level – As discussed in Section 5, exactly one of the four queue levels must be defined.
    a. SINGLE_QUEUE_LEVEL should be defined as exactly one of the following…
        i. NO_SINGLE_QUEUE_LEVEL
        ii. MIN_SINGLE_QUEUE_LEVEL
        iii. MED_SINGLE_QUEUE_LEVEL
        iv. MAX_SINGLE_QUEUE_LEVEL
4. EXTENDED_SINGLE
    a. This option should be defined if single data packets or acknowledgements can ever have a payload length longer than one XTEA block.
    b. This option can only be defined if the SINGLE_QUEUE_LEVEL is at least at level MED_SINGLE_QUEUE_LEVEL.
5. Multi-Hop
    a. ONE_NET_MULTI_HOP -- This option should be defined if the device can participate in multi-hop communication.
    b. ONE_NET_MH_CLIENT_MH_CLIENT_REPEATER – This option should be defined if the device is a client device and can serve as a multi-hop repeater. ONE_NET_CLIENT and ONE_NET_MULTI_HOP must also be defined.
6. ROUTE
    a. This should be defined if the device can participate in route messages.
    b. EXTENDED_SINGLE must be defined.
7. BLOCK_MESSAGES_ENABLED – Enable this option if this device can participate in block message transactions.
    a. ROUTE must be enabled as well.
8. STREAM_MESSAGES_ENABLED – Enable this option if this device can participate in stream message transactions.

a. BLOCK_MESSAGES_ENABLED must be defined as well for this option.

9. Other Block / Stream Options – Please also see the document **Block And Stream Messages in ONE-NET.**
    a. BLOCK_STREAM_REQUEST_MASTER_PERMISSION – Should b e defined if the master should be notified and asked for permission for block and stream communications not involving the master.
10. RANGE_TESTING
    a. This is generally a debugging option, but not always. Define this option if messages from some devices should be ignored.
    b. Please see the **ONE-NET Multi-Hop And Routing** and **ONE-NET Debugging Tools And Techniques** documents for more information.
11. PID_BLOCK
    a. This is generally a debugging option, but not always. Define this option if certain packet types should be ignored.
    b. Please see the **ONE-NET Multi-Hop And Routing** and **ONE-NET Debugging Tools And Techniques** documents for more information.
12. IDLE – Define this if the device ever goes into "idle" mode. Idle mode basically is a mode that where the device is not asleep but is also not listening for messages.
13. ENHANCED_INVITE – Define this if the device can specify a certain channel to listen for invite messages or can specify a timeout period.
14. Channels
    a. US_CHANNELS - Define if the device can operate using United States channels.
    b. EUROPE_CHANNELS – Define if the device can operate using European channels.
    c. At least one of the two options above must be defined. If the device can use both United States and European channels, both should be defined.
15. NON_VOLATILE_MEMORY – Define this if the device can save its network information to non-volatile memory that can be retrieved upon power cycling.

16. DATA_RATE_CHANNEL – Define this if the device has the ability to change it's data rate and / or channel at any time.
17. ONE_NET_MEMORY – See section 7.7. Define this if the device uses the dynamic memory allocation devices provided by ONE-NET.
18. PEER – Define this option if the device maintains a peer table.
19. UART – Define this option if there is a command line interface attached to the serial cable.
20. Evaluation Board Specific #define options

    These options are generally one applicable to the Evaluation Modules.

    1. DEFAULT_BAUD_RATE – The baud rate that the device is using. Should be either 38400 or 115200.
    2. AUTO_MODE – Define if the Evaluation Board can operate in "Auto Mode".
    3. SNIFFER_MODE – Define if the Evaluation Board can function as a sniffer.
    4. DEBUGGING_TOOLS – Define if the debugging tools are to be enabled. See the **ONE-NET Debugging Tools And Techniques** document**.**
    5. DEBUG_VERBOSE_LEVEL – Define as a number from 0 to 6. The higher the value, the more verbose the output will be.
    6. CLOCK_SLOW_DOWN – Another debugging option. Allows time to "slow down" so messages can be seen by the human eye.
        a. CLOCK_SLOW_DOWN_FACTOR – Should also be defined if CLOCK_SLOW_DOWN is defined. This is the factor that the clock should slow down to.
    7. WRITE_PAUSE – Another debugging option intended to slow things down. This is the time the device pauses before sending a message through the transceiver.
        a. WRITE_PAUSE_FACTOR should also be defined if WRITE_PAUSE is defined.

9 Port Constants (one_net_port_const.h, one_net_client_port_const.h,

one_net_master_port_const.h files)

Any constants that vary from device to device should be placed in one of these files. The naming convention provides clues to which of the three files that values should be placed in. Values that are specific to clients should be placed in one_net_client_port_const.h. Values that are specific to masters should be placed in one_net_master_port_const.h. Values that are generic to both masters and clients should be placed in one_net_port_const.h. The following values should be defined…

1. ONE_NET_MAX_PEER_UNIT – Define only if PEER is defined. This should be the maximum number of peers in the peer table.
2. ONE_NET_MAX_PEER_PER_TXN – Define only if PEER is defined. The maximum number of peers than any one unit can have in the peer table.
3. ONE_NET_MAX_RECIPIENTS – The maximum number of recipients for any single ONE_NET message.
4. ON_MAX_HOPS_LIMIT – Define only if ONE_NET_MULTI_HOP is defined. The maximum number of hops that will be attempted to communicate with another device.
5. RANGE_TEST_ARRAY_SIZE – Define only if RANGE_TESTING is defined. The number of devices that are not ignored.
6. ONE_NET_NUM_UNIT_TYPES – The number of different unit types that a device has. For example, if a device has 2 switches, 3 humidity sensors, and 4 temperature sensors, that is three separate types of units, so ONE_NET_NUM_UNIT_TYPES should equal 3.
7. ONE_NET_NUM_UNITS – The total number of units of any type that a device has. For example, if a device has 2 switches, 3 humidity sensors, and 4 temperature sensors, 2 + 3 + 4 equals 9 units total, so ONE_NET_NUM_UNITS should equal 9.
8. ONE_NET_HEAP_SIZE – The amount of memory that should be used if using ONE-NET's dynamic memory code. This should be defined if ONE_NET_MEMORY is defined.
9. ONE_NET_HEAP_NUM_ENTRIES – The number of separate memory blocks that can be allocated if using ONE-NET's dynamic memory code. This should be defined if ONE_NET_MEMORY is defined.

10. Data Rates – ONE-NET has several different data rates. All devices must be able to operate at the base 38,400 data rate. Each data rate should be defined if and only if the device can operate at that data rate.
    a. DATA_RATE_38_4_CAPABLE – define if the device can operate at the 38,400 data rate. ALL DEVICES MUST HAVE THIS VALUE DEFINED!
    b. DATA_RATE_76_8_CAPABLE – define if the device can operate at the 76,800 data rate.
    c. DATA_RATE_115_2_CAPABLE – define if the device can operate at the 115,200 data rate.
    d. DATA_RATE_153_6_CAPABLE – define if the device can operate at the 153,600 data rate.
    e. DATA_RATE_192_0_CAPABLE – define if the device can operate at the 192,000 data rate.
    f. DATA_RATE_230_4_CAPABLE – define if the device can operate at the 230,400 data rate.
11. Block Messages Default Transfer Values
    a. Define only if BLOCK_MESSAGES_ENABLED is defined.
    b. These are simply the default values that ONE-NET will use. They can be changed at any time by the application code.
    c. DEFAULT_BS_CHUNK_SIZE – The number of packets in a "chunk" of a long block message. Please also see the document **Block And Stream Messages in ONE-NET.**
    d. DEFAULT_BS_CHUNK_DELAY – The length of time to pause between chunks of a block message in order to allow other messages to proceed. Please also see the document **Block And Stream Messages in ONE-NET.**
12. DEVICE_SLEEPS – This is a client-only option. This should be defined if and only if the device ever goes to sleep.
13. ONE_NET_RX_FROM_DEVICE_COUNT – This is a client-only value. This is the maximum number of client devices that a client can keep track of.
14. ONE_NET_MASTER_MAX_CLIENTS – This is a master-only value. This is the maximum number of client devices that are allowed in a network.

15. The following are master-only constants that should be defined in one_net_master_port_const.h as either TRUE or FALSE. Some of these values may be irrelevant if other options are not defined as well. Inparticular, several of the options below deal with only Block and/or Stream Communications. These are the default options only. These options can be changed at run-time and can vary from client to client. Please also see the document **Block And Stream Messages in ONE-NET** for more details**.** Note that four constants appear to appear twice. This is intentional and is not an error. Notice that four definitions have the phrase "MASTER_CLIENT" and four have the phrase "MASTER_MASTER". The four "MASTER_MASTER" definitions involve block / stream communications where the master is one of the end nodes. The four "MASTER_CLIENT" definitions involve block / stream communications where the master is NOT one of the end nodes. Very often the master will invoke a different policy for communications where it is one of the endpoints versus when it is not.
    a. ONE_NET_MASTER_SEND_TO_MASTER – Define as true if the master wants clients to report status changes to the master.
    b. ONE_NET_MASTER_REJECT_INVALID_MSG_ID – Define as true if replay attacks are a factor in the network. If defined, devices will never accept messages with the same Message ID and the same key. Please also see the **Preventing Replay Attacks And Other Security Considerations In ONE-NET** document.
    c. ONE_NET_MASTER_CLIENT_ALLOW_LONG_BLOCK_STREAM – Define if the master allows "long" block and stream messages in client-to-client communications. See **Block And Stream Messages in ONE-NET** for a definition of "long"**.**
    d. ONE_NET_MASTER_CLIENT_BLOCK_STREAM_ELEVATE_DATA_RATE – Define if the master wants devices participating in "long" block and stream messages to change data rates in client-to-client communications. See **Block And Stream Messages in ONE-NET** for more details.
    e. ONE_NET_MASTER_CLIENT_BLOCK_STREAM_CHANGE_CHANNEL – Define if the master wants devices participating in "long" block and stream messages to change channels in client-to-client communications. See **Block And Stream Messages in ONE-NET** for more details.

f. ONE_NET_MASTER_CLIENT_BLOCK_STREAM_HIGFH_PRIORITY – Define if the master wants "long" block and stream messages to proceed at "high priority" in client-to-client communications. See **Block And Stream Messages in ONE-NET** for more details and a definition of "high priority".

g. ONE_NET_MASTER_MASTER_ALLOW_LONG_BLOCK_STREAM – Define if the master allows "long" block and stream messages in master-to-client communications. See **Block And Stream Messages in ONE-NET** for a definition of "long"**.**

h. ONE_NET_MASTER_MASTER_BLOCK_STREAM_ELEVATE_DATA_RATE – Define if the master wants devices participating in "long" block and stream messages to change data rates in master-to-client communications. See **Block And Stream Messages in ONE-NET** for more details.

i. ONE_NET_MASTER_MASTER_BLOCK_STREAM_CHANGE_CHANNEL – Define if the master wants devices participating in "long" block and stream messages to change channels in master-to-client communications. See **Block And Stream Messages in ONE-NET** for more details.

j. ONE_NET_MASTER_MASTER_BLOCK_STREAM_HIGFH_PRIORITY – Define if the master wants "long" block and stream messages to proceed at "high priority" in master-to-client communications. See **Block And Stream Messages in ONE-NET** for more details and a definition of "high priority".

10 Device Features

Every device has four "features" bytes that can be thought of as capabilities. The first thing that is done when one device communicates with another is that the two devices exchange features bytes. The features bytes are determined entirely by the values defined in sections 8 and 9. Please see the code in the one_net_features.h file for how these four bytes are determined at compile time based on what options / features / values are defined in the config_options.h file and the three port constants files defined above. The four bytes are determined at compile time and are as follows.

Byte 0 Boolean Flags	Byte 1 Data Rates / Boolean Flags	Byte 2 Queue Level And Capacity	Byte 3 Peer Table Size / Max Hops

Figure 10.1 – The four features bytes

Byte 0 should be interpreted as 8 boolean flags. 1 means true, 0 means false. Bit 0 is the LEAST significant bit. Please see the file one_net_features.h for more information.

1. Bit 0 – True if the device can change channels and / or data rates, false otherwise.
2. Bit 1 – True if the device maintains a peer table, false otherwise.
3. Bit 2 – True if the device is NOT a “Simple Client”, false if the device IS a “Simple Client”.
4. Bit 3 – True if the device never goes to sleep, False if the device goes to sleep.
5. Bit 4 – True if the device can participate in Block message transactions, false if it cannot.
6. Bit 5 – True if the device is Multi-Hop Capable, false otherwise.
7. Bit 6 – True if the device can function as a multi-hop repeater, false otherwise.
8. Bit 7 – True if the device can participate in Stream message transactions, false otherwise.

Byte 1 should be interpreted as follows. For the six least significant bits, each bit represents whether the device can function at a certain data rate. Bit 0 is the least significant bit. Bit 6 is a Boolean bit representing whether the device can handle an “Extended Single” message (i.e. a single message with 24 encrypted payload bytes). Bit 7 is a Boolean bit representing whether the device can participate in “Route” messages.

1. Bit 0 – True if the device can function at the 38,400 data rate, false otherwise. Note that ALL devices must be able to communicate at this data rate, so this bit will always be true.
2. Bit 1 – True if the device can function at the 76,800 data rate, false otherwise.

3. Bit 2 – True if the device can function at the 115,200 data rate, false otherwise.
4. Bit 3 – True if the device can function at the 153,600 data rate, false otherwise.
5. Bit 4 – True if the device can function at the 192,000 data rate, false otherwise.
6. Bit 5 – True if the device can function at the 230,400 data rate, false otherwise.
7. Bit 6 – True if the device can participate in Single Transactions with 24-bit payload blocks, false otherwise.
8. Bit 7 – True if the device can participate in Route Messages, false otherwise.

Byte 2 contains the queue level and queue capacity. The four most significant bits represent the queue capacity. They are relevant only if a queue is used. Bits 2 and 3 represent the queue level. Bits 0 and 1 are unused.

Byte 3 contains the maximum number of peers in the peer table and the maximum number of hops allowed in multi-hop communications. They are relevant only if ONE_NET_MULTI_HOP and PEER are defined. The four most significant bytes are the peer table size. The four least significant bytes are the maximum number of hops allowed.

Please see the functions in the one_net_features.h file for a list of functions that parse the function bytes for certain information.

10.1 Features Bytes Example

Assume a device has the following four feature bytes…3F 47 38 67.

Parsing the bytes one byte at a time…

Byte 0 – 0x3F – Bits – 00111111 (Recall that bit 0 is the LEAST significant bit)

1. Bit 0 – True. The device can change channels and / or data rates.
2. Bit 1 – True. The device maintains a peer table..

3. Bit 2 – True. The device is NOT a “Simple Client”.
4. Bit 3 – True. The device never goes to sleep.
5. Bit 4 – True. The device can participate in Block message transactions.
6. Bit 5 – True. The device is Multi-Hop Capable.
7. Bit 6 – False. The device cannot function as a multi-hop repeater.
8. Bit 7 – False. The device cannot participate in Stream message transactions.

Byte 1 – 0x47 – Bits – 01000111 (Recall that bit 0 is the LEAST significant bit)

1. Bit 0 – True. The device can function at the 38,400 data rate.
2. Bit 1 – True. The device can function at the 76,800 data rate.
3. Bit 2 – True. The device can function at the 115,200 data rate.
4. Bit 3 – False. The device cannot function at the 153,600 data rate.
5. Bit 4 – False. The device cannot function at the 192,000 data rate.
6. Bit 5 – False. The device can function at the 230,400 data rate.
7. Bit 6 – True. The device can participate in Extended Single message transactions.
8. Bit 7 – False. The device cannot participate in Route message transactions.

Byte 2 – 0x38 – Bits – 00111000 (Recall that bit 0 is the LEAST significant bit)

1. Bits 2 and 3 – 10 – Represent the queue level. 10 in binary is queue level 2, or MED_SINGLE_QUEUE_LEVEL. See section 5 above. This device uses a queue. This queue has the ability to send messages at a time in the future, but messages cannot “expire”.
2. Bits 4 to 7 – 0011. This is the queue capacity. 0011 is equal to 3. This device has a queue message capacity of 3 messages.

Byte 3 – 0x67 – Bits – 01100111 (Recall that bit 0 is the LEAST significant bit)

1. Bits 0 to 3 – 0111 – Represent the maximum number of hops. 0111 is 7, so this device will allow up to 7 hops in multi-hop communication.
2. Bits 4 to 7 – 0110. This is the capacity of this device’s peer table. 0110 is 6, so this device can handle up to 6 peers in its table.

Knowing this device's features helps other devices communicate with this device. For example, devices know not to bother to ask this device to engage in stream communications or to change to the 230,400 data rate because its features bytes make it clear that it can accommodate neither request.