

<b>Title</b>	MicreNet <sup>™</sup>
<b>Short description</b>	General Network, description and implementation

**Document history:**

<b>Revision</b>	<b>Date</b>	<b>Author/Who</b>	<b>What</b>
1	17-Jan-2005	PKB	Document created.

**MICREL NORWAY AS**  
*Strandveien 13 – 1366 Lysaker*  
**Tel: +47 67 83 08 00 \* Fax: +47 67 83 08 10**  
*post@micrel.no <http://www.micrel.com>*

# 1 Introduction

This document describes RF-based “nodes” in a network. A “User” may or may not be connected to a node.

The purpose of the network is to enable Users to exchange data. The “data” is typically short commands of 1-32 bytes.

Part I of this document describes how the nodes work together.

An implementation of the described network is made. A description of how to use the implemented network is given as part II of this document. The implemented network can be considered an example; an application-specific implementation can be different.

If you want to implement your own proprietary network, refer to Part I for tips and examples.

If you want to use the implemented network, you can go directly to Part II.

In both cases, reading both parts of the document is suggested.

## Table of Contents

1	Introduction .....	2
2	Terms.....	3
3	Frame structure.....	5
3.1	Level 1.....	5
3.2	Level 2.....	7
3.3	Level 3.....	8
4	Addresses and Routing.....	12
4.1	Properties of the network addressing .....	15
5	Time and Frequency Sync.....	16
6	Acknowledgement.....	16
7	Use of Frequencies .....	17
8	Summary .....	19
9	User-RF node Interface Format.....	20
10	Primitives .....	22
10.1	Reset.....	22
10.2	Set_Own_Address = Network Address.....	22
10.3	Get_Own_Address .....	23
10.4	Set_System_Address = System Address.....	23
10.5	Get_System_Address .....	23
11	Address Assignment.....	24
12	Transferring data between Users.....	27
13	Suggested Testing with 2 RF Nodes (A and B) .....	28
14	References .....	31

## 2 Terms

RF Node	Physical unit, able to transmit and receive via RF
User	Equipment connected to RF Node
Master	RF Node, syncs up Slaves, able to forward data
Slave	RF Node, synced by Master
Central Master	Master with no Master above in the hierarchy
Sub-Master	A Master directly below another Master in the hierarchy
Cluster	A “1 Master - n Slaves” combination
System	1 or n Clusters together
System Address	An ID common to all RF Nodes in a System
Network Address	An ID showing the RF Node’s position in the network hierarchy
Network Level	Central Master is at level 1, it’s sub-master is at level 2 and so on
Field	Specific information about a parameter
Frame	A set of fields
Source RF Node	The RF node that transmits a frame
Destination RF Node	The RF node the frame is intended for
Original Source RF Node	The origin of the frame, different from “Source” if the frame is forwarded
Final Destination RF Node	The final destination, different from “Destination” if the frame is forwarded
Ack	A special frame used to acknowledge a data frame
Retransmission	Data frame is retransmitted

# PART I

### 3 Frame structure

The nodes in the network (the RF units) format their transmissions in a specific way; “fields” are packed into “frames”.

It is practical to identify several levels of frames. On every level, there is a “payload” which is transparent to the level. Typically, the payload is an input from the next-higher level.

Level 1 is the “lowest level”. The structure at this level is equal for all transmissions.

In the following, frames at different levels are described.

#### 3.1 Level 1

Level 1 (frame structure equal for all transmissions):

1.1 Preamble Sequence	1.2 Start of Frame Indicator	1.3 System Address	1.4 Length	1.5 Level 1 Payload = Data to/from Level2	1.6 FCS
-----------------------------	------------------------------------	--------------------------	---------------	---	------------

1.1: Preamble Sequence: 1010... (24 “elements”)

1.2: Start of Frame Indicator: 11001100 (8 “elements”)

1.3: System Address (2 bytes @ 8 bits)

1.4: Length: Length of 1.4-1.6 (1 byte @ 8 bits)

1.5: Level 1 Payload: To or from Level 2 (a complete number of bytes @ 8 bits)

1.6: FCS: Result of CRC calculations on 1.4-1.5 (2 bytes @ 8 bits)

#### Note

The terms “elements” and “bits” are used. If the bits are not coded, “elements” and “bits” are equal. If, however, bits are coded: Every “bit” is coded into a number of “elements”.

Example: If Manchester code is used, every “bit” is coded into 2 “elements”.

#### Purpose of Level 1

At level 1, the node transmits or receives bits via the MICRFxxx transceiver. In receive mode, a flag is set if an OK frame is received (CRC check passed). The next level detects this flag, and takes action.

#### Details of Level 1 fields

##### 1.1. Preamble Sequence

This is a “training sequence”. The MICRFxxx has a built-in clock recovery mechanism: bits are clocked out of the chip. The MICRFxxx adjusts the clock internally based on the received bits. To make sure the clock samples the in-coming bits correctly, a training sequence is sent. This field is not tested in the receive-procedure.

Size of the field: 24 “elements” (no coding is applied to this field)

Contents: Fixed, “101010...”

##### 1.2: Start of Frame Indicator

This field identifies the start of the frame for the receive procedure. The MICRFxxx continuously samples the received signal and clocks out the resulting bits. If there is no transmission going on, the MICRFxxx clocks out noise. To separate a “frame” from noise, the receive procedure continuously searches the output from MICRFxxx, looking for the “Start of Frame Indicator”. If found, the receive procedure reads the next field.

Size of the field: 8 “elements” (no coding is applied to this field)  
Contents: Fixed, “11001100”

### **1.3. System Address**

This field can be considered as an extension of the “Start of Frame Indicator”. All RF nodes in a “system” must have the same “System Address”. The receive procedure reads the “System Address”, and continues to read the next field if the read “System Address” matches the system address of the node. If not, the receive procedure restarts the search (i.e. it starts to search for a “Start of Frame Indicator” again.)

Note that Level 1 tests the read System Address. The advantage of this is obvious: If 2 or more neighbouring “systems” are in reach of each other, frames will only be read by RF nodes in the correct system. If from another system, Level 1 detects this, and the remainder of the frame will not be read.

In the implemented network the value of the RF node’s system address is a programmable and readable parameter (stored in the micro controller).

Size of the field: 2 bytes @ 8 bits. (If Manchester code is used, every bit is coded into 2 elements)  
Contents: A 2-byte long address, identifying the system for which the frame is intended

### **1.4: Length**

This field identifies the number of bytes to read. When Level 1 has read “Start of Frame Indicator” and “System Address”, it reads the “Length” field. If the value of “Length” is 0 or greater than the maximum receiver buffer size: Level 1 restarts the search procedure. If the value of “Length” is OK: Level 1 starts to read the next field.

Included in “Length” is the length of the “Length”, “Level 1 Payload” and “FCS” fields. The length of “Length” is 1 byte; the length of “FCS” is 2 bytes, making the length of “Level 1 Payload” equal to “Length” – 3.

Size of the field: 1 byte @ 8 bits. (If Manchester code is used, every bit is coded into 2 elements)  
Contents: A 1-byte long value, identifying the number of bytes in the frame following the “System Address” field

### **1.5: Level 1 Payload**

From Level 1’s point of view, this field contains a “set of bytes”. The field is transparent to Level 1. In receive-mode, Level 2 gets the field, and separates it into Level 2 sub-fields. In transmit-mode, Level 1 gets the payload from Level 2.

Size of the field: (“Length” – 3) bytes @ 8 bits. (If Manchester code is used, every bit is coded into 2 elements)

Contents: Transparent to Level 1

**1.6: FCS (Frame Check Sequence)**

16-bit CRC (Cyclic Redundancy Check) calculations are made on the “Length” and “Level 1 Payload” fields. The result of the CRC calculations is the FCS.

The following polynomial is used:  $x^{16} + x^{12} + x^5 + 1$

Initial value: 0x0000

In receive mode, bits are shifted into the “CRC machine” as they are read. After the received FCS is shifted in, the resulting checksum should be 0x0000.

In transmit mode, bits are shifted into the “CRC machine” in the background program (bits are txed in interrupt routines). After the “Level 1 Payload” field is entered, 16 additional 0’s are entered.

Size of the field: 16 bits. (If Manchester code is used, every bit is coded into 2 elements)  
 Contents: Result of CRC calculations

**3.2 Level 2**

2.1 Source Address	2.2 Frame Type	2.3 Payload Level 2 = Data to/ from Level 3
-----------------------	-------------------	---

2.1: Source Address: Address of the transmitting unit (4 bytes @ 8 bits)

2.2: Frame Type: Explains how “Payload Level 2” should be read (1 byte @ 8 bits)

2.3: Payload Level 2: To or from Level 3 (a complete number of bytes @ 8 bits), depends on “Frame Type”

**Purpose of Level 2**

In receive mode, Level 2 detects that a frame is received (Level 1 sets a flag). Level 2 extracts the address of the source and the frame type. The next level takes action based on the frame type.

**Details of Level 2 fields**

**2.1: Source Address**

This field holds the network address of the RF node that actually transmits the frame. This will be different from the original source if the frame is forwarded via 1 or more RF nodes. Note that the address is the “network address” of the source node, in contrast to the “system address” at Level 1.

A RF node can be a “Master” or a “Slave”. A Slave can only receive frames from or transmit frames to, 1 Master. A Slave, then, reads “Source Address”, and if not from the slave’s master, the Slave will ignore the frame.

If the frame is from the RF node’s master (RF node being a Master or a Slave), the frame is always read (it contains timing info).

Size of the field: 4 bytes @ 8 bits. (If Manchester code is used, every bit is coded into 2 elements)

Contents: (Network) Address of the transmitting unit

### 2.2: Frame Type

Explains how “Payload Level 2” should be read (8 bits)

These are the different types of frames:

FRAME\_DATA  
FRAME\_ACK

FRAME\_BEACON  
FRAME\_BEACON\_REQ

FRAME\_FORWARD\_DATA ; Forward a data frame to another unit  
FRAME\_FORWARD\_ACK ; Forward an ack frame to another unit

Size of the field: 1 byte @ 8 bits. (If Manchester code is used, every bit is coded into 2 elements)

Contents: Frame type

### 2.3: Payload Level 2

From Level 2’s point of view, this field contains a “set of bytes”. The field is transparent to Level 2. In receive-mode, Level 3 gets the field, and separates it into Level 3 sub-fields. In transmit-mode, Level 2 gets the payload from Level 3.

Size of the field: Subtract 5 from the length of Level 1 payload, that is: (“Length” - 3 - 5) bytes = (“Length” - 8) bytes @ 8 bits. (If Manchester code is used, every bit is coded into 2 elements)

Contents: Transparent to Level 2

## 3.3 Level 3

At level 3, the “Level 2 Payload” is split into “Level 3 sub fields”.

Below, the sub-fields for different frame types are shown. Note: The field called “Time/Freq Info” is only included if the frame is sent from a master RF node.

Frame type = FRAME\_DATA:

3.11 Destination Address	3.21 Frame ID (this frame)	3.31 Data Bytes	3.61 Time/Freq Info
-----------------------------	-------------------------------	--------------------	------------------------

The FRAME\_DATA type is used to transmit data within a Master-Slave “Cluster”. That is: The original source and the final destination are in a Master-Slave relationship (not necessary to forward data).

Frame type = FRAME\_ACK:

3.11 Destination Address	3.41 Frame ID to ack	3.61 Time/Freq Info
-----------------------------	-------------------------	------------------------

The FRAME\_ACK type is used to acknowledge a data frame within a Master-Slave “Cluster”. That is: The original source and the final destination are in a Master-Slave relationship (not necessary to forward data).

Frame type = FRAME\_BEACON:

3.61 Time/Freq Info
------------------------

Note: For FRAME\_BEACON, “Destination Address” is not used.

This frame type is sent by all masters at regular intervals (when there has been no other activity, i.e. transmission of timing info), or when timing info is requested from a Slave or “sub-master”.

Frame type = FRAME\_BEACON\_REQ:

3.11 Destination Address
-----------------------------

The FRAME\_BEACON\_REQ is sent by Slaves or Masters after power-on or when they get out of sync. The only master that does not request timing info, is the “Central Master”.

Frame type = FRAME\_FORWARD\_DATA:

3.11 Destination Address	3.21 Frame ID (this frame)	3.31 Data Bytes	3.51 Original Source Address	3.52 Final Destination Address	3.61 Time/Freq Info
-----------------------------	-------------------------------	--------------------	---------------------------------	-----------------------------------	------------------------

A RF node constructs FRAME\_FORWARD\_DATA if the data is to a Master/Slave that can’t be reached directly, but must be forwarded.

Frame type = FRAME\_FORWARD\_ACK:

3.11 Destination Address	3.41 Frame ID to ack	3.51 Original Source Address	3.52 Final Destination Address	3.61 Time/Freq Info
-----------------------------	-------------------------	---------------------------------	-----------------------------------	------------------------

FRAME\_FORWARD\_ACK is used to ack forwarded data.

Note: As a special case, a “FRAME\_FORWARD\_DATA” can be used instead, echoing the received data to the original source.

- 3.11: Destination Address: Receiver’s address (4 bytes @ 8 bits)
- 3.21: Frame ID: ID of this frame, a number 1...254 (1 byte @ 8 bits)
- 3.31: Data Bytes: Input from or output to “User” (a complete number of bytes @ 8 bits)
- 3.41: Frame ID to ack: ID of a frame to acknowledge, a number 1...254 (1 byte @ 8 bits)
- 3.51: Original Source Address: Address of the original source unit (4 bytes @ 8 bits)
- 3.52: Final Destination Address: Address of the final unit (4 bytes @ 8 bits)
- 3.61: Time/Freq Info: Included only if frame is a “master” (3 bytes @ 8 bits)

### **Purpose of Level 3**

In receive mode, Level 3 extracts fields and takes action based on the fields.

In transmit mode, Level 3 packs fields together and gives it to Level 2.

Above Level 3 is the “User”, which inputs data to transmit or gets received bytes. Instead of “Data Bytes”, the term “Level 3 Payload” can be used, and instead of “User”, “Level 4” can be used. Note that “User” can be a human being using a PC, it can be a micro-controller or some simple circuitry ( a simple I/O device, like a light fixture, can be a “User”).

### **Details of Level 3 fields**

#### **3.11: Destination address**

The network address of the RF node for which the frame is intended. This will be different from the final destination if the frame is forwarded via 1 or more RF nodes. Note that the address is the “network address” of the destination node, in contrast to the “system address” at Level 1.

Size of the field: 4 bytes @ 8 bits. (If Manchester code is used, every bit is coded into 2 elements)

Contents: (Network) Address of the receiver unit

#### **3.21: ID of this frame**

A frame ID used to identify a duplicate transmission within a “Cluster”

Size of the field: 1 byte @ 8 bits. (If Manchester code is used, every bit is coded into 2 elements)

Contents: A number 1...254

#### **3.31: Data bytes**

Data to transport between Users. Transparent to Level 3.

Size of the field: n bytes @ 8 bits. (If Manchester code is used, every bit is coded into 2 elements)

Contents: Transparent to Level 3.

### **3.41: Frame ID to ack**

Size of the field: 1 byte @ 8 bits. (If Manchester code is used, every bit is coded into 2 elements)

Contents: A number 1...254

### **3.51: Original Source Address**

Address of the original source unit. This is used in forwarded frames only.

Size of the field: 4 bytes @ 8 bits. (If Manchester code is used, every bit is coded into 2 elements)

Contents: The network address of the original source RF node

### **3.52: Final Destination Address**

Address of the final unit. This is used in forwarded frames only.

Size of the field: 4 bytes @ 8 bits. (If Manchester code is used, every bit is coded into 2 elements)

Contents: The network address of the final destination RF node

### **3.61: Time/Freq Info**

Included only if frame is from a “Master” node.

Size of the field: 3 bytes @ 8 bits. (If Manchester code is used, every bit is coded into 2 elements)

Contents: 1 byte for frequency info, and 2 bytes for timing info.

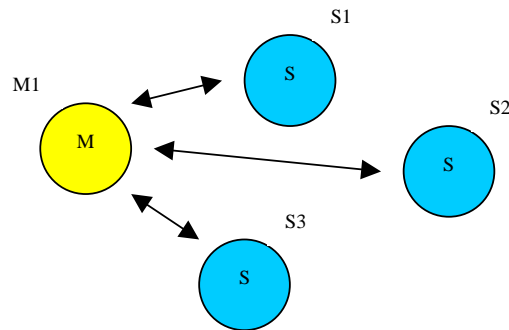
## 4 Addresses and Routing

In a simple star network, there is 1 Master and 1...n Slaves, referred to as a “Cluster”.

Within a “Cluster”, the Master can talk directly to any Slave, and a Slave can talk directly to the Master.

If a Slave wants to talk to another Slave in the same cluster, the communication must be via the Master, that is: the data must be “forwarded”.

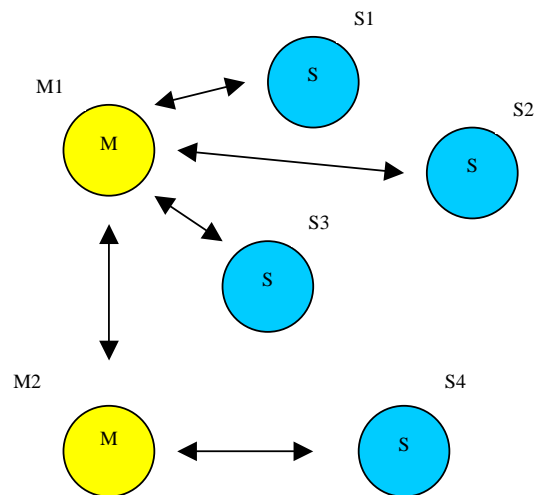
In the figure below, a “Cluster” with 1 Master and 3 Slaves are shown. They are called M1, S1, S2 and S3, respectively.



**Figure 1. A “Cluster” with 1 Master and 3 Slaves**

In a “System”, several clusters can be connected.

To increase range, 1 or more Masters (with or without own slaves) can be inserted. If M1 and S4 are out-of range, a 2<sup>nd</sup> Master is inserted. In this case, the inserted Master serves as a repeater. Refer to the figure below, where M2 and S4 are inserted.



**Figure 2. Inserting a Master to increase range**

In this document, there is 3 types of RF nodes:

S: Slave Node

M: Master Node

C: Central Master Node (a Master with special properties)

All RF nodes in a “System” have the same “system address”. This is a 2-byte long address, making it possible to have neighbouring systems that will not talk to each other.

In addition, every unit has a 4-byte “network address” that identifies the position of the RF node in the network.

Every RF node, except for the “Central Master”, has 1 Master directly “above”. Every Master can have 0, 1 or 2 “sub-Masters”, i.e. Masters directly below in the network hierarchy.

The first byte in the network address gives the “network level”. The Central Master is always at network level 1. If there is only 1 Master in the System, this is a Central Master. And: At network level 1, there is only one Master.

The last 3 bytes:

The n (n = network level) most significant bits show the Master address. The remaining 24-n bits are all-zeros for a Master, and for a slave: A running number not all-zeros and not all-ones.

A simple network with 1 Master and 3 Slaves, will have network addresses like this:

Master M1 (Central Master):

Network Level: 1

Master address (1 bit long): 1, Running address: (23 bits long): 0

Slave S1:

Network Level: 1

Master address (1 bit long): 1, Running address: (23 bits long): 1

Slave S2:

Network Level: 1

Master address (1 bit long): 1, Running address: (23 bits long): 2

Slave S3:

Network Level: 1

Master address (1 bit long): 1, Running address: (23 bits long): 3

In “dotted decimal”, the addresses are:

M1: 1.128.0.0.

S1: 1.128.0.1.

S2: 1.128.0.2.

S3: 1.128.0.3.

Note: If the msb of a byte is “1” and the remaining bits are “0”, the decimal value is 128.

Then, if M1 has 1 Master directly below it, the new Master will be at network level 2. Then 2 bits are used as a “Master address”, and  $24-2 = 22$  bits are used as a running number.

Note that the first bit is inherited from the Master, that is: at network level 2, the Master addresses are “10” and “11” (binary).

A Slave connected to a Master at level 2, will have a 22-bit long running number (starting at 1).

From the figures above:

Master M2:

Network Level: 2

Master address (2 bits long): “10” (binary), Running address: (22 bits long): 0

Slave S4:

Network Level: 2

Master address (2 bits long): “10” (binary), Running address: (22 bits long): 1

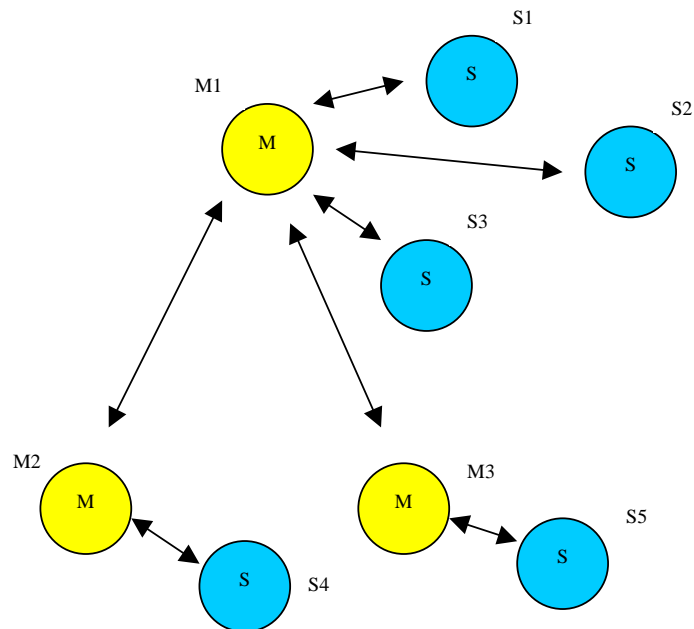
In “dotted decimal”, the addresses are:

M2: 2.128.0.0.

S4: 2.128.0.1.

Note: If the 2 msb of a byte is “10” and the remaining bits are “0”, the decimal value is 128.

Now, a 2<sup>nd</sup> Master can be inserted at network level 2. In the figure below, “M3” is inserted, having “S5” as it’s slave. M3 and S5 have the 2-bit master address”11”.



**Figure 3. Two Masters below one Master**

The new nodes will have these addresses:

Master M3:

Network Level: 2

Master address (2 bits long): “11” (binary), Running address: (22 bits long): 0

Slave S5:

Network Level: 2

Master address (2 bits long): “11” (binary), Running address: (22 bits long): 1

In “dotted decimal”, the addresses are:

M3: 2.192.0.0.

S5: 2.192.0.1.

Note: If the 2 msb of a byte is “11” and the remaining bits are “0”, the decimal value is 192.

A complete list of the nodes in the figures above:

M1: 1.128.0.0.

S1: 1.128.0.1.

S2: 1.128.0.2.

S3: 1.128.0.3.

M2: 2.128.0.0.

S4: 2.128.0.1.

M3: 2.192.0.0.

S5: 2.192.0.1.

#### **4.1 Properties of the network addressing**

- At network Level 1, there is always 1 Master (Central Master)
- At network level n, there is 0, 1 or 2 Masters
- At network level n, n bits are used as the “Master address”
- Maximum number of Slaves at network level n is  $2^{(24-n)} - 2$  (“-2” for the all-ones and all-zeros cases)
- If, for a node at network level n, the (24-n) last bits are all-zeros, the node knows it is a Master (else it knows it is a Slave)
- For a Slave: The address of “my” Master is the n most significant bits of byte 2 ... byte4
- For a Master: The address of “my” Master is the n-1 most significant bits of byte 2 ... byte4
- A Slave will always transmit user data to it’s Master
- A Master will transmit user data to it’s Master, to one of it’s sub-masters or directly to a Slave

- Due to the structure, a Master will always know which Master to forward a packet to. It is not necessary to keep a routing directory. Refer to [1] for a discussion of “routing directories”.
- New Slaves and Masters can be added without updating any of the existing nodes

## 5 Time and Frequency Sync

All units are in sync all the time (exception: at start-up, a sync-routine is carried out).

A Master will sync up its associated Slaves. If there are more than 1 Master in the network, it is necessary to have a Central Master, referred to as a “C” unit. All time-sync’ing in the network is based on the C unit.

After power-on, a Master rf node will first request timing info from its Master, and then transmit timing info to its Slaves and Sub-Masters.

A Slave RF node will simply request timing info from its Master.

Timing info is included in all packets from all Masters. Whenever a RF node receives a packet from its Master, the timing info is used to adjust its internal timer.

How sophisticated the time-adjust procedures are, depends on the implementation of the network.

## 6 Acknowledgement

The original source node expects ack from the final destination node.

If the original source and final destination make a Master-Slave pair, the Master/Slave will ack the packet directly.

If 1 or more Masters repeat the packet, then there will be no ack’ing on the “inner levels”.

In the implemented network, the ack is an echo of the received data.

If the data transfer does not make use of “forwarding”, i.e. original source and final destination are in a direct link: If no ack is received, the source makes up to 16 retransmissions.

If the data transfer involves “forwarding”, no retransmissions are made. Instead, this task is raised to the User level. Note that the original User gets an ack if the transmission was successful. If the User does not get an echo (in a “reasonable time”), the User can select to make a new attempt.

## 7 Use of Frequencies

The number of frequencies used depends on the frequency band.

In the 915MHz band, 25 frequencies are used, and the use of frequencies follows the fcc part 15 regulations.

Ideally, all the network nodes are at the same frequency at the same time. Collisions will/may occur, but:

- If there is only 1 “initiator” of packets (e.g. the Central Master), no collisions will occur
- If there are several initiators: Collisions can occur. However, if initiators are physically out-of-range, the problem is reduced (specifically, this is the case if Masters are used as repeaters to achieve a long range.)
- Listen-before-talk can be implemented as an anti-collision mechanism.

# PART II

## 8 Summary

For a quick start, refer to the section called “Suggested Testing with 2 RF Nodes (A and B)”.

In a star network, there is 1 Master and 1...n Slaves. However, it is possible to include additional Masters to increase range, or to get a peer-peer network by transmitting via a Master.

In the network described in this document, there is 3 types of RF units:

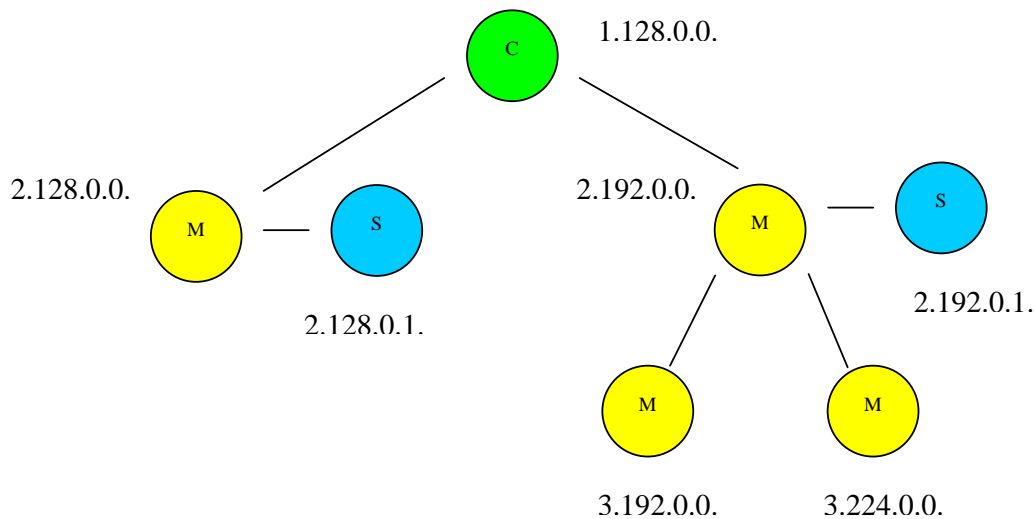
S: Slave Node

M: Master Node

C: Central Master Node

Every unit has a unique 4-byte address, programmed by User. The first byte gives the “network level”. Byte 2, 3, 4: At network level n, the n most significant bits are used as “master address”. The remaining bits are “all-zeros” for a Master, and a free-running number not equal to “all-zeros” and “all-ones” for a Slave. The first byte can be thought of as the number of 1’s in a “subnet mask”. Refer to [2].

Refer to Figure 1 and identify the different nodes and the address hierarchy.



**Figure 4. Network example: Identify nodes and addresses**

When the address is programmed into the node, the node knows:

- I am a Master, Slave or Central Master
- I know the address of my Master
- I know my own address

A number of Slaves are associated to one Master. That is: A Slave talks to the associated Master only, while a Master can talk to any of it's associated Slaves. In addition, the Master can talk to a neighbouring Master (on the level below/above) in the network structure.

A Master can have 0, 1 or 2 Masters directly below (at the next network level). In addition, a Master can have a number of Slaves (at the same network level). At network level n, n bits are used for "master address". That is, (24 - n) bits can be used for "host address". "All-zeros" and "All-ones" are reserved, making the total number of Slaves under a Master:

$$\#Slaves(\text{level } n) = 2^{(24-n)} - 2$$

All units are in sync all the time (exception: at start-up, a sync-routine is carried out).

A Master will sync up it's associated Slaves. If there are more than 1 Master in the network, it is necessary to have a Central Master, referred to as a "C" unit. All the "time-sync'ing" in the network is based on the C unit.

## 9 User-RF node Interface Format

UART interface: 9600-8-N-1

Hardware handshaking (RTS/CTS)

User is a "DTE" and RF node is a "DCE"

User can enter bytes into RF node when the CTS line is active (high)

RF node can enter bytes into User when the RTS line is active (high)

The interface can be used to

- 1) Transfer data bytes between Users
- 2) Program the node (set/get address information)

User must give input to the RF node in a specified way. The input must have these fields:

Address	Length_Of_User_Payload	User_Payload
---------	------------------------	--------------

The RF node will use the same fields when giving bytes to User.

The "Address" and "Length\_Of\_User\_Payload" fields must be entered with a special format: They must be entered as "dotted decimal". That is: Every byte is entered as a decimal number, entered using ascii chars, with a "." after every byte.

If an illegal character (not '0'-'9' and not '.') is entered, the user must restart entering the Address and Length\_Of\_User\_Payload fields. This provides the User with a "regret" possibility: If "User" is a human being entering bytes manually, it is most likely that he makes a mistake some time. If he makes a mistake, then, he can simply enter an illegal byte and restart. Note that he can't restart if the Address and Length\_Of\_User\_Payload fields are entered correctly.

If a byte in the Address field is > 255, the input is ignored.

If Length\_Of\_User\_Payload is 0, or it is too big to fit into buffer, the input is ignored. Max length of payload is 32 bytes.

In addition, by using “dotted decimal”, it is possible for a user to operate the RF node from a PC running e.g. HyperTerminal (“User” does not have to be a micro controller or a special PC program).

If user enters “Data”, the bytes in the “User\_Payload” field can be entered any way the User wants. If User enters a “Command”, the “User\_Payload” must have a special format.

“Data” and “Commands” can be given via the interface. If User enters data, the payload will be the data bytes to transfer. If User enters a command, the Payload will be a “primitive” (typically a “request” or a “confirm”).

To identify the payload as a command, a reserved address is used. The reserved address is called the “Universal\_Address”, and is equal to “0.0.0.0.”.

The length of the Address is 4 bytes.

The length of the Length\_Of\_User\_Payload field is 1 byte

The length of the User\_Payload is given in the Length\_Of\_User\_Payload field

## 10 Primitives

Primitives have a command type-field, followed by 1 or more parameters.

These primitives are available:

- Reset
- Set\_Own\_Address = Network Address
- Get\_Own\_Address
- Set\_System\_Address = System Address
- Get\_System\_Address

Below, the primitives are detailed. For the “Set\_Own\_Address” primitive, an example is shown.

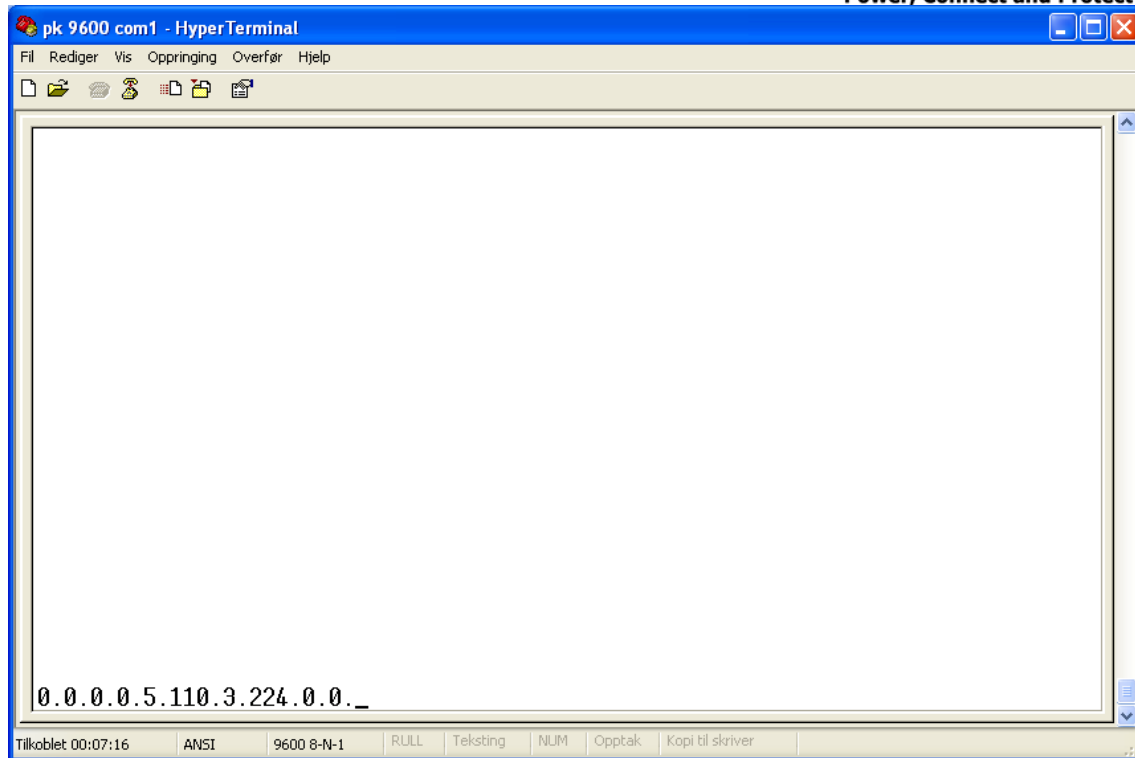
### 10.1 Reset

- Command length: 1. (1 byte for type and 0 bytes for parameters)
- Command type: 1. (dotted decimal)
- Purpose: Restarts the micro controller, equal to a power-on.

### 10.2 Set\_Own\_Address = Network Address

- Command length: 5. (1 byte for type and 4 bytes for address)
- Command type: 110. (dotted decimal)
- Purpose: Give the node it’s position in the network
- Note: the 4-byte long address is stored in EEPROM

Example: Using HyperTerminal to program a RF node with the address 3.224.0.0.: Refer to Figure 2.



**Figure 5. Using HyperTerminal to program address**

### **10.3 Get\_Own\_Address**

- Command length: 1. (1 byte for type and 0 bytes for parameters)
- Command type: 111. (dotted decimal)
- Purpose: Read the node's position in the network
- Note: the 4-byte long address stored in EEPROM is returned to user in dotted decimal form.

### **10.4 Set\_System\_Address = System Address**

- Command length: 3. (1 byte for type and 2 bytes for system address)
- Command type: 120. (dotted decimal)
- Purpose: Give the node its system address
- Note: the 2-byte long address is stored in EEPROM

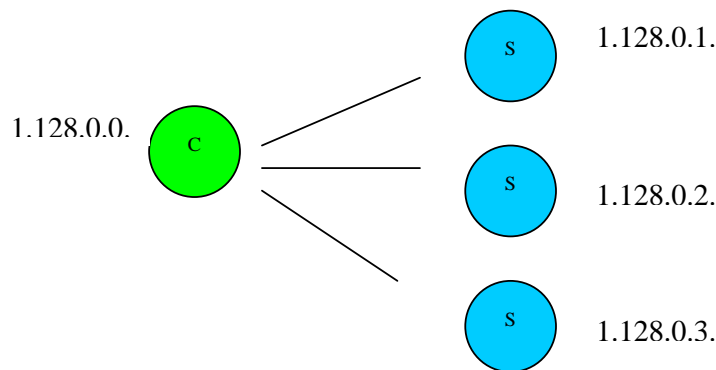
### **10.5 Get\_System\_Address**

- Command length: 1. (1 byte for type and 0 bytes for parameters)
- Command type: 121. (dotted decimal)
- Purpose: Read the node's system address
- Note: the 2-byte long address stored in EEPROM is returned to user in dotted decimal form

## 11 Address Assignment

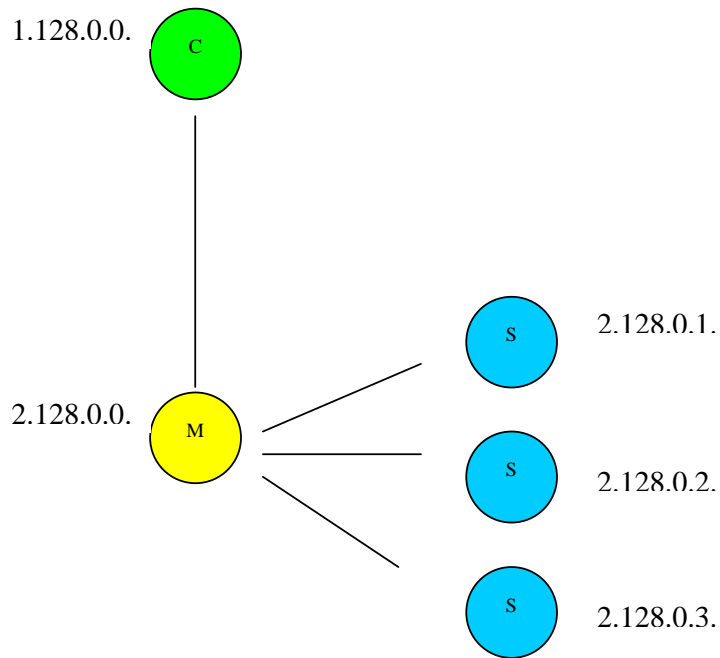
A 4-byte address is used; the bytes are called Byte1, Byte2, Byte3 and Byte4.

Byte 1 identifies the network level, or the number of significant bits in the address. If only 1 Master is used, it must be at network level 1: Only 1 bit is used for the “network level”; it can be “0” or “1”. To avoid the “universal address”, “0” should be avoided. The only address left, then, is “1”. A “1” in the ms position gives “128.” in “dotted decimal” format. The remaining 23 bits are all-zeros, indicating a Master. The Slaves have the same network level and same “master address”. Refer to the figure below: Identify the single Master and the 3 Slaves, and their addresses.



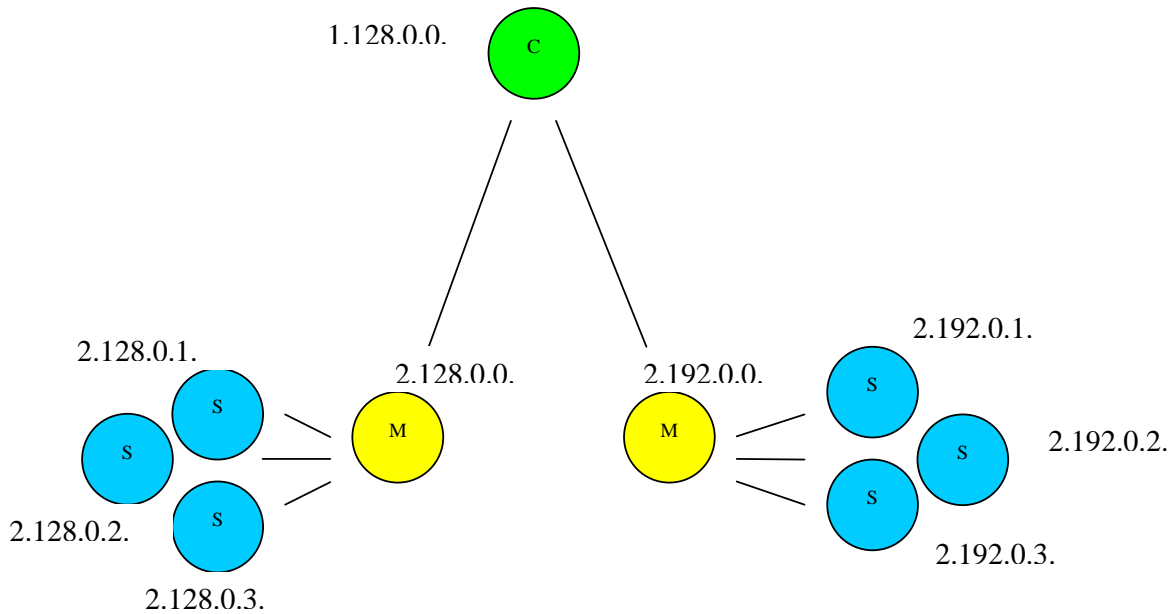
**Figure 6. A Single-Master Star network**

If a Master and a Slave is out-of range (due to a long distance or the physical environment; floors, walls etc): 1 or more additional Masters can be inserted. The inserted Masters will function as repeaters. At every new network level, the first byte (Byte1) is incremented by 1. Directly below the network level 1, then, is network level 2. At network level 2, 2 bits are used for the “master address”. And so on. Refer to the figure below. The inserted Master has a 2-bit “master address”, equal to “10”. In dotted decimal, this is “128.” (Note: An additional Master can be inserted at the same level, this will be “11” or “192.”).



**Figure 7. Star network with a Master working as repeater**

A Master can have maximum 2 Masters directly below it, in addition to a number of Slaves. Refer to the figure below and identify the use of addresses.



**Figure 8. Star network with 2 Masters below 1 Master**

As a suggested exercise, take a look at the general network below and identify all addresses. Note that the “master address” is printed in every RF node in binary form (note the number of bits)

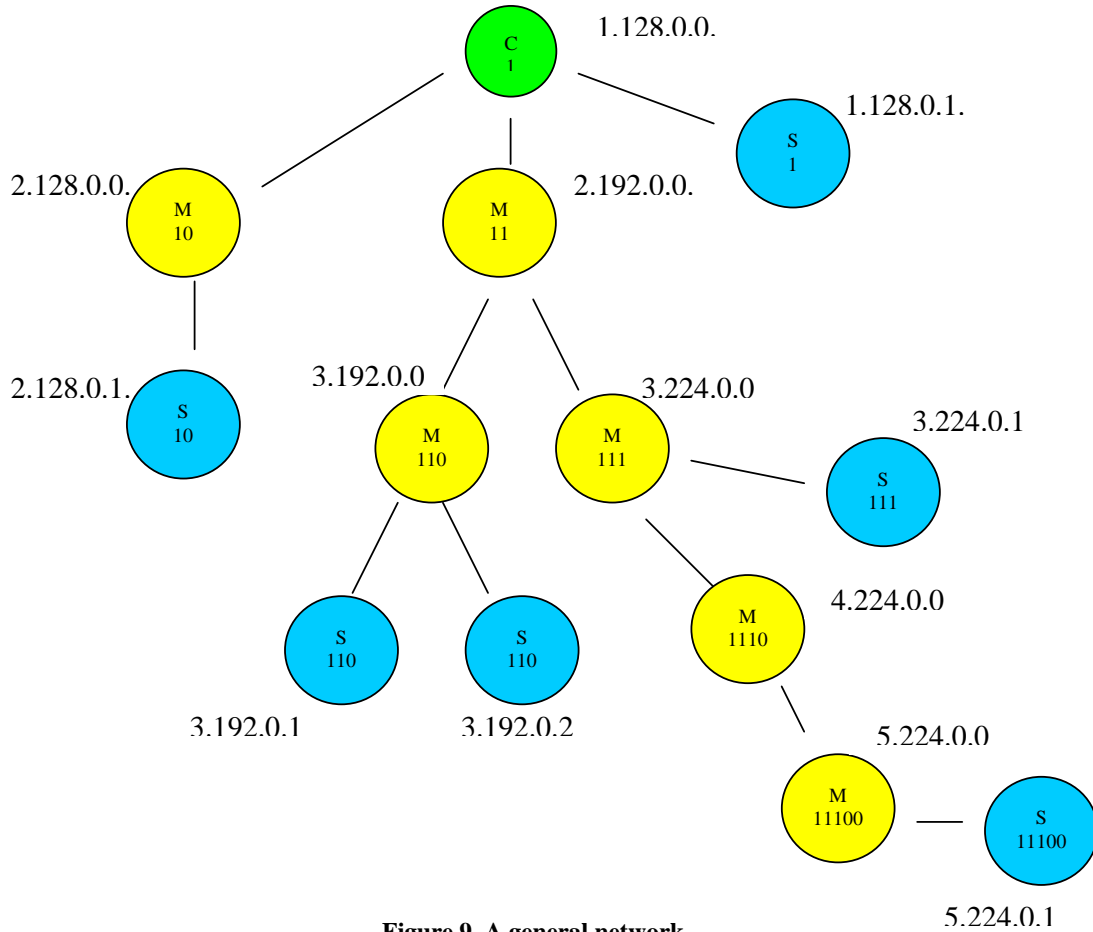


Figure 9. A general network

## 12 Transferring data between Users

Refer to the next section as well.

The user-rf node interface format must be followed.

Ack: When ack is received, the original data is echoed out to user.

Note: Retransmissions (when no ack is received) will only take place if the source and destination are in a direct link.

Example: The User connected to "1.128.0.1." wants to transfer the 6 ascii characters "Hello!" to "2.128.0.1."

User enters this:  
2.128.0.1.6.Hello!

The User connected to the destination node gets this:  
1.128.0.1.6.Hello!

And finally, the source User gets ack and echoes the string back to user:  
2.128.0.1.6.Hello!

## 13 Suggested Testing with 2 RF Nodes (A and B)

Use Hyper Terminal to program card A as Master, card B as Slave:

Connect Dev board A to PC via 1:1 cable

Start HyperTerminal, 9600-8-N-1, hardware handshake

Enter: “0.0.0.0.5.110.1.128.0.0.” (This will set the A board as a Master, at the top level)

0.0.0.0. = the “universal address”, indicating “this is a command”  
5. = length of payload, - indicating 5 bytes are coming  
110. = type of command – indicating “set network address”  
1.128.0.0. = address to program

Enter:” 0.0.0.0.3.120.1.2.” (This will set system-address to 1.2.)

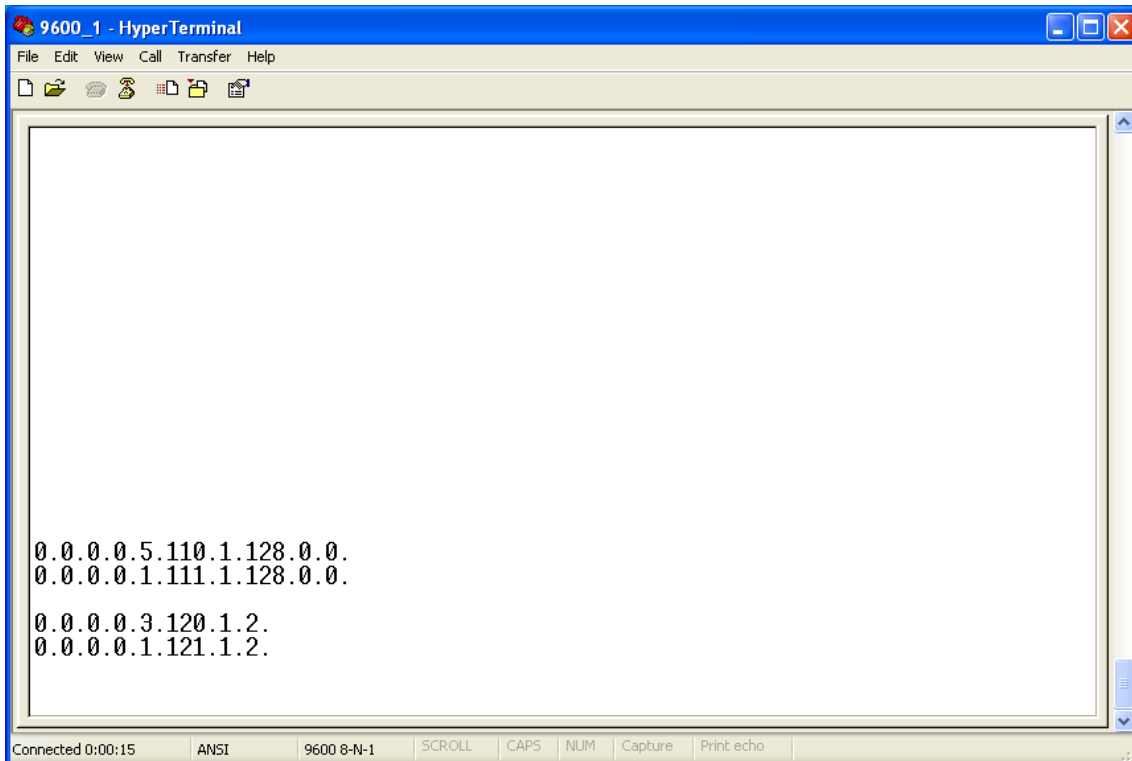
0.0.0.0. = the “universal address”, indicating “this is a command”  
3. = length of payload, - indicating 3 bytes are coming  
120. = type of command – indicating “set system address”  
1.2. = address to program

Repeat for B, but enter: “0.0.0.0.5.110.1.128.0.1.”

(This will set the B board as a Slave, with the A board as it’s Master)

(Make sure the same system address is entered)

In the figure below, HyperTerminal is used to set parameters on A. After a parameter is programmed, it is confirmed. Several “carriage returns” are added to make it easier to read.

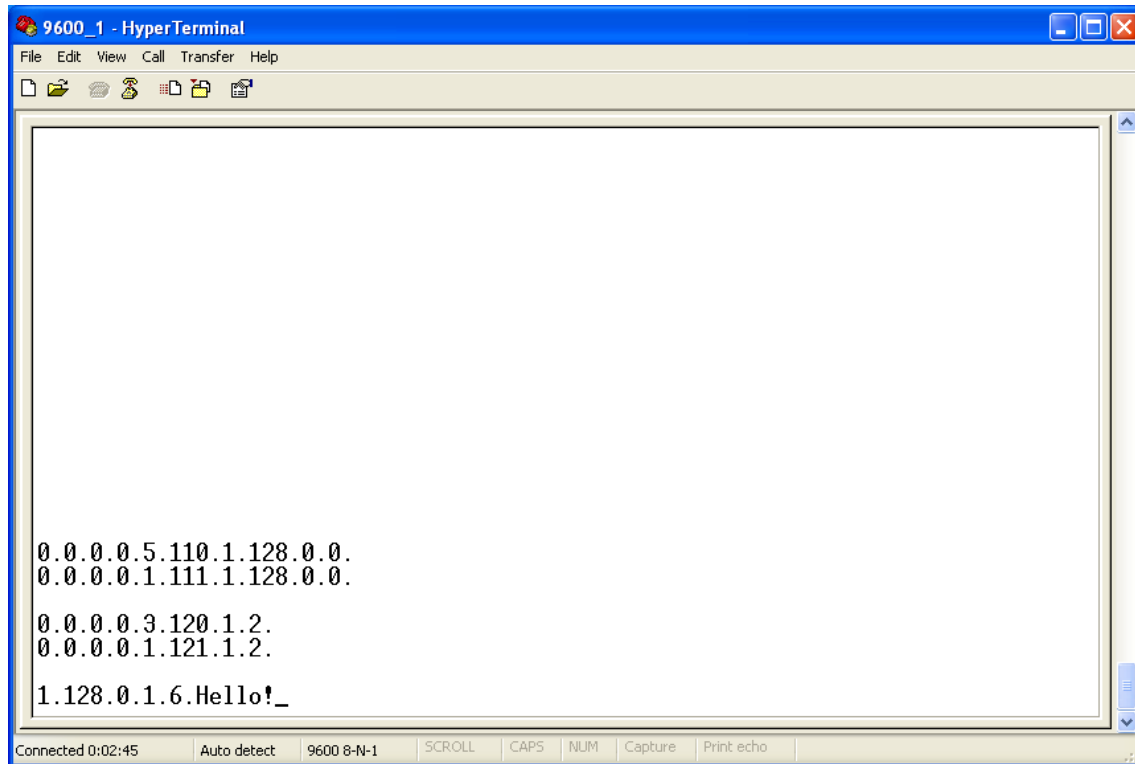


**Figure 10. Using HyperTerminal to set and verify parameters**

Now try to transmit data between A and B:

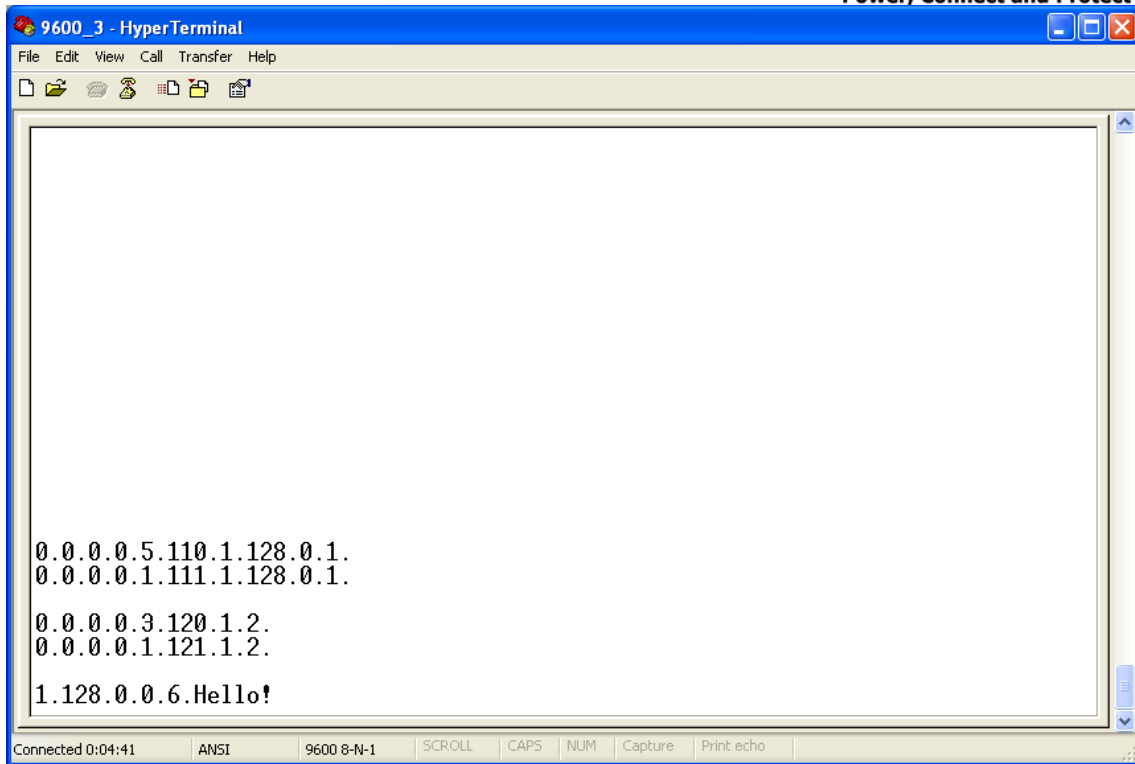
On A:

Enter destination address, then number of bytes, then the bytes:



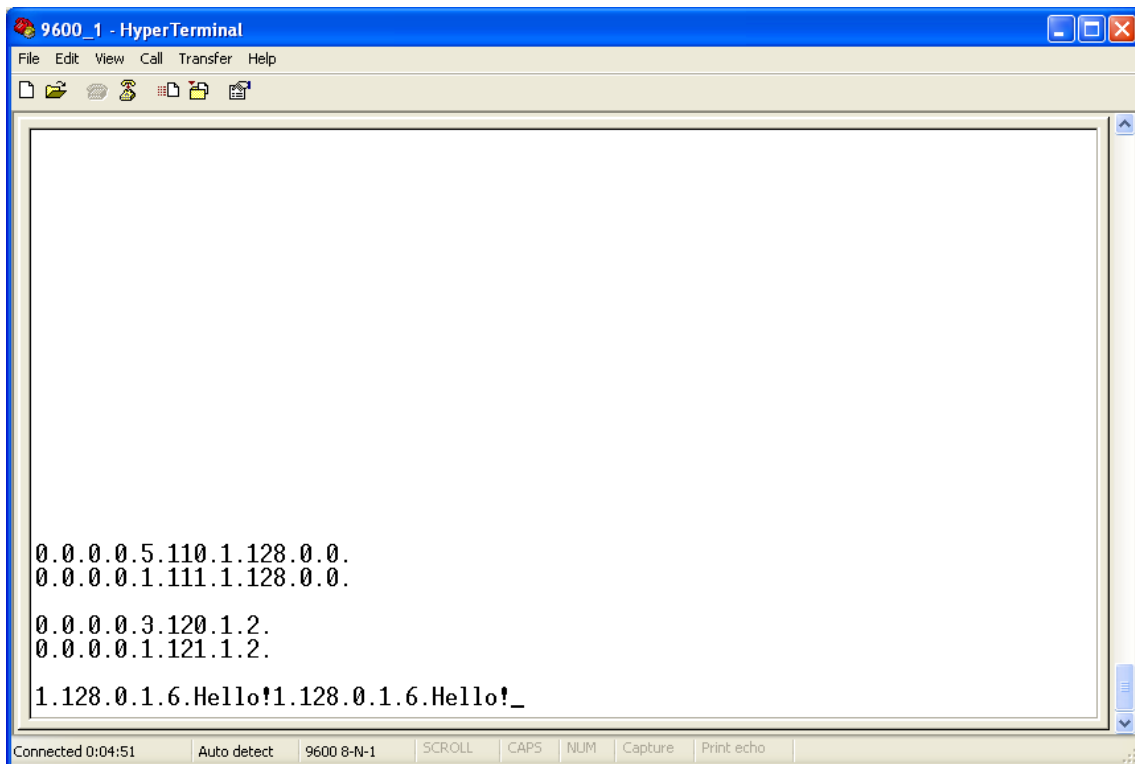
**Figure 11. Using HyperTerminal to initiate data transfer**

On B, watch that the source address, the number of bytes, then the bytes, pops up:



**Figure 12. Using HyperTerminal to get data**

And finally: Observe that the original data pops up at A, to acknowledge the data.



**Figure 13. Using HyperTerminal to get ack from destination**

## 14 References

[1] Stallings, William: “Data and computer communications”, fifth edition 1997, Prentice-Hall, ISBN 0-13-571274-2

[2] Hall, Eric: “Internet Core Protocols: The Definitive Guide” (Appendix B), O’Reilly 2000, ISBN 1-56592-572-6