Application Note

The Diagnostics Channel Protocol
Model P900

Revision: **1.04**

microhard SYSTEMS INC.

150 Country Hills Landing NW
Calgary, Alberta T3K 5P3
Phone: (403) 248-0028
Fax: (403) 248-2762
www.microhardcorp.com

Warranty

Microhard Systems Inc. warrants that each product will be free of defects in material and workmanship for a period of one (1) year for its products. The warranty commences on the date the product is shipped by Microhard Systems Inc. Microhard Systems Inc.'s sole liability and responsibility under this warranty is to repair or replace any product which is returned to it by the Buyer and which Microhard Systems Inc. determines does not conform to the warranty. Product returned to Microhard Systems Inc. for warranty service will be shipped to Microhard Systems Inc. at Buyer's expense and will be returned to Buyer at Microhard Systems Inc.'s expense. In no event shall Microhard Systems Inc. be responsible under this warranty for any defect which is caused by negligence, misuse or mistreatment of a product or for any unit which has been altered or modified in any way. The warranty of replacement shall terminate with the warranty of the product.

Warranty Disclaims

Microhard Systems Inc. makes no warranties of any nature of kind, expressed or implied, with respect to the hardware, software, and/or products and hereby disclaims any and all such warranties, including but not limited to warranty of non-infringement, implied warranties of merchantability for a particular purpose, any interruption or loss of the hardware, software, and/or product, any delay in providing the hardware, software, and/or product or correcting any defect in the hardware, software, and/or product, or any other warranty. The Purchaser represents and warrants that Microhard Systems Inc. has not made any such warranties to the Purchaser or its agents MICROHARD SYSTEMS INC. EXPRESS WARRANTY TO BUYER CONSTITUTES MICROHARD SYSTEMS INC. SOLE LIABILITY AND THE BUYER'S SOLE REMEDIES. EXCEPT AS THUS PROVIDED, MICROHARD SYSTEMS INC. DISCLAIMS ALL WARRANTIES, EXPRESS OR IMPLIED, INCLUDING ANY WARRANTY OF MERCHANTABILITY OR FITNESS FOR A PARTICULAR PROMISE.

MICROHARD SYSTEMS INC. PRODUCTS ARE NOT DESIGNED OR INTENDED TO BE USED IN ANY LIFE SUPPORT RELATED DEVICE OR SYSTEM RELATED FUNCTIONS NOR AS PART OF ANY OTHER CRITICAL SYSTEM AND ARE GRANTED NO FUNCTIONAL WARRANTY.

Indemnification

The Purchaser shall indemnify Microhard Systems Inc. and its respective directors, officers, employees, successors and assigns including any subsidiaries, related corporations, or affiliates, shall be released and discharged from any and all manner of action, causes of action, liability, losses, damages, suits, dues, sums of money, expenses (including legal fees), general damages, special damages, including without limitation, claims for personal injuries, death or property damage related to the products sold hereunder, costs and demands of every and any kind and nature whatsoever at law.

IN NO EVENT WILL MICROHARD SYSTEMS INC. BE LIABLE FOR ANY INDIRECT, SPECIAL, CONSEQUENTIAL, INCIDENTAL, BUSINESS INTERRUPTION, CATASTROPHIC, PUNITIVE OR OTHER DAMAGES WHICH MAY BE CLAIMED TO ARISE IN CONNECTION WITH THE HARDWARE, REGARDLESS OF THE LEGAL THEORY BEHIND SUCH CLAIMS, WHETHER IN TORT, CONTRACT OR UNDER ANY APPLICABLE STATUTORY OR REGULATORY LAWS, RULES, REGULATIONS, EXECUTIVE OR ADMINISTRATIVE ORDERS OR DECLARATIONS OR OTHERWISE, EVEN IF MICROHARD SYSTEMS INC. HAS BEEN ADVISED OR OTHERWISE HAS KNOWLEDGE OF THE POSSIBILITY OF SUCH DAMAGES AND TAKES NO ACTION TO PREVENT OR MINIMIZE SUCH DAMAGES. IN THE EVENT THAT REGARDLESS OF THE WARRANTY DISCLAIMERS AND HOLD HARMLESS PROVISIONS INCLUDED ABOVE MICROHARD SYSTEMS INC. IS SOMEHOW HELD LIABLE OR RESPONSIBLE FOR ANY DAMAGE OR INJURY, MICROHARD SYSTEMS INC.'S LIABILITY FOR ANYDAMAGES SHALL NOT EXCEED THE PROFIT REALIZED BY MICROHARD SYSTEMS INC. ON THE SALE OR PROVISION OF THE HARDWARE TO THE CUSTOMER.

Proprietary Rights

The Buyer hereby acknowledges that Microhard Systems Inc. has a proprietary interest and intellectual property rights in the Hardware, Software and/or Products. The Purchaser shall not (i) remove any copyright, trade secret, trademark or other evidence of Microhard Systems Inc.'s ownership or proprietary interest or confidentiality other proprietary notices contained on, or in, the Hardware, Software or Products, (ii) reproduce or modify any Hardware, Software or Products or make any copies thereof, (iii) reverse assemble, reverse engineer or decompile any Software or copy thereof in whole or in part, (iv) sell, transfer or otherwise make available to others the Hardware, Software, or Products or documentation thereof or any copy thereof, except in accordance with this Agreement.

1 Overview

This application note describes how to use the Diagnostics channel for on-line diagnostics and set up.

2 Diagnostics Protocol and Structure of Diagnostics Commands

Diagnostic data on COM2 is exchanged in packets at 115.2 kbps rate and data format 8N1.
Packets originated by the user are requests; packets originated by the modem are responses.
There must be a time gap of at least 20 ms between the requests. This time interval is used by the modem to detect the end of a packet.
The packet content is protected by the optional 16 bit CRC.
Requests can be unicast and broadcast. Address 0 is reserved for local access. Responses always have the real address of the modem, including responses to requests with the local address. Address is represented as a 6 byte hexadecimal numeric value, not a string. For example, the address of 12:34:56:78:9A:BC has its high byte of 0x12, the lowest byte – 0xBC.
There is a 16 bit magic number which is user specific and returned unchanged with the response. It can be used by the user as a sequence number.

There are three types of requests: reads, writes and special functions. Each type has its unique command IDs followed by optional parameters.

A modem that receives a request will always generate a response. It can be data, error codes and/or function specific responses. A bit in control byte allows to suppress the response. This is useful when a broadcast command is sent to all modems and no response is needed.

There is a bit in the control byte to report an error in request processing.

IMPORTANT: There is little error checking on the value of parameters that the user can write to the diagnostics. This is a low level interface to the modem. All range checking and validation of parameters must be done by the user before sending data to the modem.

All diagnostics packets, requests and responses, have the same general structure (Figure 1).

Byte 1 Size	Bytes 2-7 MAC address (high byte first)	Bytes 8, 9 Magic number	Byte 10 Control byte	Byte 11 and following Payload	…	Byte Last-1 CRC(H)	Byte Last CRC(L)

Figure 1. Diagnostics packet structure.

For example, to read the COM1 baud rate locally the following packet must be sent to COM2

1	**2**	**3**	**4**	**5**	**6**
0xb,	0x0, 0x0, 0x0, 0x0, 0x0, 0x0,	0x12, 0x34	0x2	0x0, 0x2,	0xeb, 0x17

where
1 – Size (0xb). It is the size of the diag packet EXCLUDING the size field itself and two bytes of CRC.
2 - Six bytes of MAC address: local access MAC address 0x000000000000.
3 - User defined magic number: 0x1234.
4 - Control byte: 0x2.
5 - Two bytes of payload:
    0x0 - command ID to read parameters and
    0x2 - parameter ID for COM1 baud rate.

6 - Two bytes of CRC: 0xeb17.

Notes:

The minimum size value is 10 bytes: (6 bytes for address, 2 for magic number, one for control and one for command ID). The max size is 254 bytes.
MAC address: Contains 6 bytes of MAC address of the destination modem for this packet.
Local address is all zeroes. 0xFFFFFFFFFFFF is broadcast. The MAC address is sent high byte first.
Control byte:
Bit 0 - reserved, must be set to zero.
Bit 1 - response needed. When this bit is cleared a modem will suppress a response, even for read commands. It is useful to prevent flooding the network with responses to broadcasts requests.
Bit 2 - reserved, must be set to zero
Bit 3 - if set in response - the modem detected an error in the request. This bit must be set to zero in requests.
Bit 4 - if set in response - access was denied. Login via login function Command ID 0xC (or via AT command) is needed. This bit must be set to zero in requests.
Bits 5, 6 and 7 are reserved and must be set to zero.
Payload:
The first byte of payload is command ID (Cmd ID) that can be followed by optional data. Payload types and their structure will be described later.
CRC: CRC-16 ANSI, polynomial x16+x15+x2+1. Initial value - 0xffff.
CRC computation is performed on the size byte, address, control byte and payload. The 16 bit CRC is transmitted high byte first. See software example in the Appendix B.
For debug the CRC check can be disabled by setting S163 = 0. In this case instead of CRC two dummy bytes of any value must still be transmitted in the packet.

3 Diagnostics Commands

The diagnostic protocol allows the user to read parameters, such as user settings, temperature, statistics etc, write parameters and execute functions, such as flash upgrade, login, reset etc. The first byte of the request payload contains a command ID which identifies the action the user requests from the modem. The appendix A lists available parameter IDs.

Read/Write commands operate on parameters that are identified by their unique one byte parameter IDs. Some parameters have their AT command interface S-register equivalents, some – do not. Most parameters are read/written from/to a proxy structure. The proxy contains a shadow copy of user’s parameters and is not used by the modem during operation. Writes to this proxy will not affect the modem until saved in the NVRAM and the modem restarted. To save these parameters in NVRAM, a write command must be issued. This command will interrupt data communication and reset the modem.
Not every parameter can be saved – some of them are read only. Also, a write to some parameters immediately updates their working values but those values can't be saved. To permanently change them they must be written to the proxy structure and saved. An example is output power that has its proxy mapped parameter ID 3 and the immediate counterpart parameter ID 29. A write to the immediate power setting ID 29 will take effect immediately, but it will not affect the shadow memory.

Each parameter has a size – one, two or more bytes. Knowing the size of a parameter is important because in order to keep the efficiency of the protocol the size values of parameters are not included into the payload. If the size of a parameter is not known it is not possible to parse multi-parameter payloads.
Most parameters have a fixed size. For instance, operating mode S101 Par ID 13 is a one byte value, destination address S140 Par ID 16 is a 6 byte value. Some parameters, such as version ID 5 are null terminated strings and their size may change between product versions. Some parameters, like statistics ID100, are composite structures. They can also change their size between product releases as more values are added into those structures. Microhard guarantees that in order to preserve compatibility between different releases the changes to this type of parameters will only be additive. In other words, new parameters can be added, but nothing can be removed or re-arranged. Accessing these “variable size parameters” should be done by using a single parameter request. Then the size of the parameter can be computed by examining the total size of the returned packet and accounting for overhead – address, control byte, magic word, CRC etc. Writing into a variable size parameter requires knowing the exact size of the parameter. If the write command contains a number of bytes that is different from what the modem knows about the parameter, an error code will be returned and no write will take place. Therefore it is advisable to perform a read of such a parameter to identify its size before writing.

Multi-byte data is represented in the little endian format - lower byte first.

Appendix B shows the Calculation of CRC16 in C.

3.1. Read. Command ID 0.

This command can read multiple parameters in one request as long as the size of the payload in the response does not exceed 244 bytes. As mentioned before, variable size parameters are normally accessed as a single parameter read.

Below is an example of a complete local multi-parameter read request in which three parameters are requested: power setting ID 3, max packet size ID 8 and firmware version ID 5 (all numbers in hex).

0d	00 00 00 00 00 00	01 23	02	00	03 08 05	ab cd
Size	MAC	Magic word	Control	Cmd	par IDs	CRC (dummy, as S163=0)

The complete response (all numbers are in hex):

16	12 34 56 78 AB CD	01 23	00	00	03 1E	08 00 01	05 76 31 2E 30 39 00	F7 33
Size	MAC	Magic	Control	Cmd	Pwr – 30dBm	PktSize – 256	Ver “v1.09”	CRC

Size is 16 which is 22d and includes everything except the size field and CRC.
MAC is the actual MAC of the responding modem.
Magic Number is the same as in the request.
Control byte. The bit 3 is zero, meaning no errors in the request.
Cmd. The same as in request – read.
03 – par ID for power, 1E – 30d, the power setting of 30 dBm. The size is one byte.
08 - par ID for max packet size. 00 01 is the returned value, size - two bytes, lower byte = 0, high byte = 1, 0x100 = 256d
05 76 31 2E 30 39 00 - par ID for version and “v1.09” as data. Note that this is a variable size parameter, but it is a string and as such, null terminated. It allows for parsing multi-parameter responses that contain strings even though their size is unknown.
F7 33 – CRC. The modem will always produce correct CRC regardless of the value of S163.

If a parameter ID in the request is invalid, meaning that the modem doesn’t know this ID, the modem will skip it, continue parsing and return an error bit 3 in the control byte of the response.
For example, the following parameters are requested locally: invalid ID 200, power setting ID 3, invalid ID 200, max packet size ID 8
0e 00 00 00 00 00 00 01 23 02 00 c8 03 c8 08 ab cd,

The complete response:

0F	12 34 56 78 AB CD	01 23	08	00	03 1E	08 00 01	5C 42
Size	MAC	Magic	Control	Cmd	Power 30dBm	Max Pkt size 256	CRC

The control byte 0x8 has its bit 3 set, which indicates an error in processing. The user needs to examine the payload of the response to find out which IDs are missing. Those will be invalid IDs.

3.2. Write. Command ID 4.

The structure of the write command is similar to that of a read response - a parameter ID is followed by its value. The modem parses the request using known sizes of parameters. If a wrong number of data bytes are sent as a parameter, the parsing will fail and wrong values will be written into parameters.
A successful write command will return error-free command code. For example, two parameters are written in one local request: a power level of 20 dBm is requested into the proxy structure Par ID 3 and the max packet size Par ID 8 of 200 bytes. The following command will be sent to diagnostics:

0f 00 00 00 00 00 00 01 23 02 04 03 14 08 1c 00 ab cd
Size MAC Magic Control Cmd Pwr 20dBm max pkt size 28 CRC
The control byte 0x2 has the bit 1 set to request the response.
The power setting ID 3 is followed by exactly one byte of the parameter value 0x14 which is 20dBm in decimal.
The max packet size ID 8 is a two byte value – 0x001c, which is 28 bytes.

The complete response:
0A 12 34 56 78 AB CD 01 23 00 04 4D 59
Size MAC Magic Control Cmd CRC
The control byte reports no errors – bit 3 is set to zero.

There are three possible causes of errors in write requests: invalid parameter ID, out of range parameter and invalid size of a parameter. On any error the bit 3 in control byte of the response is set and the payload of the response will report the cause of the error by listing offending parameters' ID followed by their error codes. The possible error codes are:
1 - unknown parameter ID
2 - parameter is out of range (when checking is done on the parameter value)
3 - wrong size of parameter
The modem stops parsing on the first unknown parameter ID or wrong size of the parameter. If a parameter exists but its value is out of range, parsing will continue, but an error code will be returned for this parameter in the response.

In the following example three parameters are written in one local request: a power level of 40 dBm (an out of range value) is requested into the proxy structure Par ID 3, the max packet size Par ID 8 of 28 bytes and an invalid parameter ID 150 (0x96) with a 2 bytes value of 0x1023. The following command will be sent to diagnostics:
12 00 00 00 00 00 00 01 23 02 04 03 28 08 1c 00 96 23 10 ab cd
Size MAC Magic Control Cmd Pwr 40dBm max pkt size 28 Invalid ID CRC

The complete response:
0E 12 34 56 78 AB CD 01 23 08 04 03 02 96 01 24 88
Size MAC Magic Control Cmd ID 3, error code 2 ID 150, error code 1 CRC
The control byte reports an error – bit 3 is set, and error codes are returned in the payload. ID3 reports an out of range parameter, ID 150 reports an unknown parameter ID.

If the size of a parameter doesn't match the number of bytes supplied in the request, the error code 3 will be returned. Obviously, this kind of checking can only work on the last (or the only) parameter in a request.
An attempt to write into a read only parameter is treated as out of range error.

3.3. Special Functions. Cmd ID 8.

Functions are different from the simple read/write in that they can execute complex and delayed actions like flash upgrade, reset or parameter save. Only one function can be executed in one request and each function has its own set of parameters and its own response. Function arguments are passed in the request payload. Data and error status will be returned in the response. If a function reports an error, the bit 3 of the command byte will be set in addition to function specific error codes in the payload of the response.

The following functions are supported:

Function ID	Description
1	Flash upgrade
2	Clear statistics
3	Saving proxy parameters in NVRAM
4	Modem reset
5	Rssi report
6	Current primary hop pattern info

3.3.1. Flash Upgrade Function.

Flash upgrade function requires a function command ID and optional parameters.
There are four commands:

Function Command ID	Description
2	Upload one 128 byte block of image
3	Start of image CRC checking
4	Arm for flash upgrade
5	Start flash upgrade

The response format for all command IDs is the same – the control byte will have its bit 3 set or cleared to indicate the overall status of the request. The payload will contain the following information:
Special functions command ID, Function ID, Function command ID, Error Code (one byte).
Error codes are specific for each function command ID. An unknown command IDs will return code 10.

Function command ID 2 - uploading a 128 byte block of image
The format of the request payload:
08 01 02 followed by 2 bytes of the block number and 128 bytes of data, where 08 is the command ID (functions), 01- function ID (Flash Upgrade), 02 – function specific command – image upload. The block number is in little endian format. The return error codes are: 0 – success, 6 – busy, 7 if the block number exceeds the limit of 2304 or 5 if the size of the command was wrong.

Function command ID 3 - start of image CRC checking
The format is 08 01 03
It returns the error code 8 if the format was wrong, 9 in case the modem was busy and 0 if success. Note that this command only starts the process of CRC checking of the downloaded image. Depending on how busy the modem is, it may take up to a few seconds to complete the check. Parameter ID 7 (see note 1, Appendix A) contains the status of the flash upgrade system, including the result of the image CRC check.

Function command ID 4 - arming the flash upgrade. The arming timeout is 20 seconds. The flash upgrade must be started within this time.
The format is (in hex) 08 01 04 17 71
Returns 0x0a if the size was wrong, 0x0b if the arming value parameter (0x7117) was wrong, 0x0c if the modem was busy and 0 if success.

Function command ID 5 - start flash upgrade.
The format is 08 01 05.
It returns 0x0d if the size was wrong, 0x0e if the arming timeout has expired, 0x0f if the modem was busy or 0 if success. Flash upgrade is a delayed function – it will start in 10 seconds after the command is received. During flash upgrade the modem is not operational and will reset once the upgrade is done.

3.3.2. Clear Statistics.

This function clears statistics that is available for read via the read command Cmd 0, parameter ID 100. See Note 1. Since the normal operation on statistics (other than reading it) is to clear it, this special function allows for a convenient and bandwidth efficient way of doing it. It can also be cleared via the write command Cmd 4, parameter ID 97.
The format is 08 02, returns code 0 if success, 1- if wrong size.

3.3.3. Saving proxy parameters in NVRAM.

This function starts the process of saving the proxy structure that contains user settings into NVRAM.
The format is 08 03.

This is a delayed function - if started successfully, it will wait for 5 seconds before executing and resetting the modem.
It returns 0 if success, or 1 if the size was wrong or the modem is already in the process of saving and resetting.

3.3.4. Modem Reset.

This function starts the process of resetting the modem.
The format is 08 04.
This is a delayed function - if started successfully it will wait for 5 seconds before resetting the modem.
It returns 0 is success or 1 if the size was wrong or the modem is already in the process of resetting.

3.3.5. RSSI report.

This function reports noise and rssi values collected on all channels in the currently used hop pattern. This information may be useful in debugging performance problems. In mesh networks all remotes record their measurements as a Slave. Averaging is done over the last 8 measurements. Note that this function returns values referenced to a hop index, not to the actual channel at that index.

This function requires two parameters - an action and a selection
An action can be:
0 - get noise
1 - clear noise
2 - get RSSI
3 - clear RSSI
For each action there are selections.

The "Get noise" action has the following selection:
0 - an array of average noise values at each hop index for Master
1 - the same as above for slave
2 - an array of min noise values at each hop index for Master
3 - the same as above for slave
4 - an array of max noise values at each hop index for Master
5 - the same as above for slave

"Clear noise" has the following selections:
0 - clear the array of average noise values at each hop index for Master
1 - the same as above for slave
2 - clear the array of min noise values at each hop index for Master
3 - the same as above for slave
4 - clear the array of max noise values at each hop index for Master
5 - the same as above for slave

"Get RSSI" has two selections
0 - an array of average RSSI at each hop index for Master
1 - the same for slave

"Clear RSSI" has two selections
0 - clear the array of average RSSI at each hop index for Master
1 - the same for slave

The RSSI report returns error codes:
0 if success, 1 - if the size was wrong, 11 to 14 if a selection was wrong for the action chosen, 15 if the action was invalid.

The actions 0 and 2 return an array of RSSI. The size of the array is the size of the hop pattern. The hop pattern size for FCC compliant country codes is 50 and the modem returns 50 bytes of values starting from the hop index 0.

3.3.6. Current Primary Hop pattern information.

The format is 08 06.
Returns code 0 if success, 1 - if the format of the request is wrong.
The following data is returned in the little endian format:

1 byte Hopping mode. 0 - on hop pattern, 1 - on frequency table, 2 - on channel, 3 - on frequency.
2 bytes Test Channel
1 byte Hop Pattern size
2 bytes Min Channel
2 bytes Max Channel
2 bytes Channel Space in kHz
4 bytes Start Frequency in kHz,
Followed by the pattern size number of channels (two bytes each) in the current hop pattern.

3.4. Special Functions. Cmd ID 0xC

The structure of this function request is the same as for general functions. The difference is that special functions only work via local connection (address 000000000000) and they are processed even if the user is not logged in.
The following functions are supported:

Function ID	Description
1	Login

3.4.1. Login Function.

As with the login via AT command interface, the user has 5 attempts to enter login information, after which the modem will lock the user out for 10 minutes. After 10 minutes of complete inactivity on the diagnostics port the attempt count will be restored. If the power is cycled, even after 10 minutes, on power up the user will still have to wait for 10 minutes before being able to login again.

The payload format is 0c 01 00 <login password> 00 where
0c - special function request
01 - function ID - login function
00 - function parameter - sending login information
login password – 1 to 32 chars.
00 - string termination

The function returns error code 1 if size was wrong, code 2 if the login was incorrect but the user still has more attempts to try, 0x10 - if unknown command, 0x16 - if the user is locked out for 10 minutes. 0 - if success.

Appendix A

The list of supported parameter IDs.

Parameter ID	Parameter	Size, Bytes	S-Register	Comments
1	Serial Channel Mode	1	S142	Proxy
2	Serial Baud Rate	1	S102	Proxy
3	Output Power, dBm (effective after reset)	1	S108	Proxy
4	Hop Time	1	S109	Proxy
5	String Version	var	ATI3	Read only
6	Packet Size Minimum	2	S111	Proxy
7	Status 1	1	System	Note 1
8	Packet Size Maximum	2	S112	Proxy
9	Packet Retransmissions	1	S113	Proxy
10	Repeaters in System	1	S141	Proxy
11	Protocol Type	1	S217	Proxy
12	Handshaking	1	&K	Proxy
13	Operating Mode	1	S101	Proxy
14	Wireless Link Rate	1	S103	Proxy
15	Escape Character	1	S2	Proxy
16	Destination Address	6	S140	Proxy
17	Diagnostics CRC Control	1	S163	Proxy
18	Power Up Mode	1	S0	Proxy
19	Serial Data Format	1	S110	Proxy
20	DCD control	1	&C	Proxy
21	Master Bandwidth, %	1	S250	Proxy
22	Records Time to Live, sec.	2	S83	Proxy
23	DCD Pulse Period	1	S165	Proxy
24	Master System Time	4	System	Read only
25	Network ID	4	S104	Proxy
26	Repeat Interval	1	S115	Proxy
27	Character Time-Out	1	S116	Proxy
28	Roaming	2	S118	Proxy
29	Output Power, dBm	1	S108	Immediate
30	Channel Access Mode	1	S244	Proxy
31	Data Priority	1	Menu	Proxy
32	Diagnostics Priority	1	Menu	Proxy
33	Diagnostics Retransmissions	1	S214	Proxy
34	Routing Retransmissions	1	Menu	Proxy
35	Diagnostics Priority	1	Menu	Immediate
36	Diagnostics Retransmissions	1	S214	Immediate
37	Max Buffers IN Storage	2	S232	Proxy
38	Sync Time-Out	2	S248	Proxy
39	Packets Per Hop Tx Limit	1	S249	Proxy
40	Slave Channel Allocation Limit	1	S252	Proxy
41	Forward Error Correction	1	S158	Proxy
42	Network Type	1	S133	Proxy
43	Routing Time to Live, sec.	1	S235	Proxy
44	Master Channel Request Time-Out	1	S234	Proxy

Parameter ID	Parameter	Size, Bytes	S-Register	Comments
45	Max Buffers OUT Storage	2	S236	Proxy
46	DSR control	1	&S	Proxy
47	DTR control	1	&D	Proxy
48	Transmit Done Time-Out	1	S146	Proxy
49	Receive Done Time-Out	1	S147	Proxy
50	Fast Sync Packets	2	S151	Proxy
51	Mesh Coordinator Rank	1	S220	Proxy
52	Address Tag	1	S153	Proxy
53	User String	32	ATI0	Proxy
54	Country Code	1	S247	Read only
55	Temperature	1	System	Read only Note 2
56	Synchronized level	2	System	Read only
57	Request Slots	1	S156	Proxy
58	Data Time to Live	2	S184	Proxy
59	Encryption Mode	1	S159	Proxy
60	RSSI averaged master	1	System	Read only
61	RSSI averaged slave	1	System	Read only
62	NOISE aggregate master	1	System	Read only
63	NOISE aggregate slave	1	System	Read only
64	Hop Pattern	1	S106	Proxy
65	Secondary Hop Pattern	1	S206	Proxy
66	Hop Zone	1	S180	Proxy
67	Secondary Zone	1	S181	Proxy
68	Max Power	1	System	Read only
69	Standby Trip Level	1	S224	Proxy
70	Tx Profile	1	S80	Proxy
71	CS Threshold	1	S81	Proxy
72	Attempts Before re-Route	1	S126	Proxy
73	Input Framing	1	S218	Proxy
74	Routing Request Time to live	2	S219	Proxy
75	Number of Aloha Slots	1	S214	Proxy
76	Number of Mesh Sync Slots	1	S215	Proxy
77	Mesh Sync Duty Cycle	1	S216	Proxy
78	Routing	1	S223	Proxy
79	MAC Address	6	factory	Read only
80	Unit Address	2	S105	Proxy
81	Command Echo	1	ATE	Proxy
82	Quiet Mode	1	ATQ	Proxy
83	Verbose Mode	1	ATV	Proxy
84	Words Result Mode	1	ATW	Proxy
85	Set Factory Defaults	1	AT&F	Proxy Note 3
86	DA in PP/PMP	2	S140	Proxy Note 4
87	Message To User (up to 128 characters)	var	System	Read only
88	No Sync Data Intake	1	S130	Proxy
89	Tx on Slot	1	S221	Proxy
90	Use S105 as MAC	1	S87	Proxy
91	Sleep mode	1	S143	Proxy
92	Sleep time	2	S144	Proxy
93	Wake time	2	S145	Proxy

Parameter ID	Parameter	Size, Bytes	S-Register	Comments
94	Mesh Roam Enable	1	S222	Proxy
95	Hop zone	1	System	Read only
96	Hop pattern	1	System	Read only
97	Statistics	var	System	Note 5
98	Memory usage	var	System	Read only Note 6
99	Data Compression Mode	1	S225	Proxy
100	USR_AN0	2	System	Read only, mV, Pin 15, Note 7
101	USR_AN1	2	System	Read only, mV, Pin 16, Note 7
102	Output 0	1	System	Pin 10, Note 8
103	Output 1	1	System	Pin 11, Note 8
104	String Product	var	ATI1	Read only
105	Encryption Key	32	S107	Write only, Note 9
106	Encryption Mode	1	S159	Note 9
107	Co-located Network Type	1	S154	
108	Reverse RSSI LEDs	1	S88	
109	Mesh Sys Time	4	System	Read only
110	Error Log	var	System	
111	Hardware Revision	1	System	Read only
112	Sync Slot To Tx	1	S226	Proxy
113	Repeat Interval for Broadcast	1	S237	Proxy
117	V bat, mV	2	System	Read only
118	Vcc, mV	2	System	Read only
119	Diag Time To Live	2	S185	Proxy
120	Diag Time To Live	2	S185	Immediate
121	Diag Response Address	1	S89	Immediate
122	Diag Response Address	1	S89	Proxy
123	Cost of hop in Mesh	1	S245	Proxy
124	Ack errors per channel	100	System	

Note 1

Par ID 7 - status1 in one byte that represents the following status bits used in flash upgrade.

D0 - CRCcheckDone;
D1 - CRCcheckInProgress;
D2 - CRCcheckResult;
D3 - flashPreloadDone;
D4 - flashPreloadInProgress;
D5 - upgradeScheduled;
D6 - paramSaveScheduled;
D7 - resetScheduled;

Note 2

Temperature is offset by 55. To obtain actual temperature subtract 55 from the reading. For example reading 95 indicates the actual temperature of 95-55 = +40C. Reading 40 means 40-55 = -15C.

Note 3

Parameter ID	Description
1	Mesh Primary Coordinator
2	Mesh Remote
3	Mesh Secondary Coordinator
7	PMP Master
8	PMP Slave
9	PMP Repeater
10	PP Master
11	PP Slave
12	PP Repeater
13	PMP Master 57K
14	PMP Slave 57K

Note 4
Only included for Microhard Radio Network use.

Note 5
Par ID 97 - statistics represents the following structure:

```
typedef struct stat {
  u32 kBytesPayloadRx;
  u32 kBytesRx;
  u32 kBytesPayloadTx;
  u32 kBytesTx;
  u32 errCorrect;
  u32 packetsDroppedDueToMemory;
  u32 RxpacketsDroppedDueToAge;
  u32 packetsTx;
  u32 packetsRx;
  u32 errCRC;
  u32 syncLost;
  u32 syncCount;
  u32 pktError;
  u32 pktsDroppedDuetoPayloadCRCerror;
  u32 TxpacketsDroppedDueToAge;
  u32 macTxBusyTimeTotal;          // in msec from the moment mac is loaded for tx to till it is done
  u32 macTxAckExpected;
  u32 macTxAckMissed;
  u32 numOfCtsReceived;
  u32 numOfRtsSent;
  u32 droppedDueToUnresolvedRouting;
  u32 invalidatedRoutes;
  u32 receiverBusy;
  u32 chAccessTime;
  u32 chAccessCounter;
};
```

Note 6
Par ID 98 represents the sizes of all buffer queues in the following order, one byte per entry.

Q_FREE	0
Q_USER	1

mesh transmit priority queues

Q_TX_PRIO_0	2 - lowest priority queue
Q_TX_PRIO_1	3
Q_TX_PRIO_2	4
Q_TX_PRIO_3	5
Q_TX_PRIO_4	6
Q_TX_PRIO_5	7
Q_TX_PRIO_6	8
Q_TX_PRIO_7	9 - highest priority queue

PP and PMP queues

Q_UP	10
Q_DOWN	11
Q_DOWN_TX	12

Note 7
Analog values are updated once every second.

Note 8
Bit 0 represents the value of the output. When read, it returns the value written to the port, not the input voltage.

Note 9
Only for modems with AES

Appendix B

Calculation of CRC16 in C.

```
static unsigned short crc_value__;

const unsigned short crc_16table[] = {
    0x0000, 0xc0c1, 0xc181, 0x0140, 0xc301, 0x03c0, 0x0280, 0xc241,
    0xc601, 0x06c0, 0x0780, 0xc741, 0x0500, 0xc5c1, 0xc481, 0x0440,
    0xcc01, 0x0cc0, 0x0d80, 0xcd41, 0x0f00, 0xcfc1, 0xce81, 0x0e40,
    0x0a00, 0xcac1, 0xcb81, 0x0b40, 0xc901, 0x09c0, 0x0880, 0xc841,
    0xd801, 0x18c0, 0x1980, 0xd941, 0x1b00, 0xdbc1, 0xda81, 0x1a40,
    0x1e00, 0xdec1, 0xdf81, 0x1f40, 0xdd01, 0x1dc0, 0x1c80, 0xdc41,
    0x1400, 0xd4c1, 0xd581, 0x1540, 0xd701, 0x17c0, 0x1680, 0xd641,
    0xd201, 0x12c0, 0x1380, 0xd341, 0x1100, 0xd1c1, 0xd081, 0x1040,
    0xf001, 0x30c0, 0x3180, 0xf141, 0x3300, 0xf3c1, 0xf281, 0x3240,
    0x3600, 0xf6c1, 0xf781, 0x3740, 0xf501, 0x35c0, 0x3480, 0xf441,
    0x3c00, 0xfcc1, 0xfd81, 0x3d40, 0xff01, 0x3fc0, 0x3e80, 0xfe41,
    0xfa01, 0x3ac0, 0x3b80, 0xfb41, 0x3900, 0xf9c1, 0xf881, 0x3840,
    0x2800, 0xe8c1, 0xe981, 0x2940, 0xeb01, 0x2bc0, 0x2a80, 0xea41,
    0xee01, 0x2ec0, 0x2f80, 0xef41, 0x2d00, 0xedc1, 0xec81, 0x2c40,
    0xe401, 0x24c0, 0x2580, 0xe541, 0x2700, 0xe7c1, 0xe681, 0x2640,
    0x2200, 0xe2c1, 0xe381, 0x2340, 0xe101, 0x21c0, 0x2080, 0xe041,
    0xa001, 0x60c0, 0x6180, 0xa141, 0x6300, 0xa3c1, 0xa281, 0x6240,
    0x6600, 0xa6c1, 0xa781, 0x6740, 0xa501, 0x65c0, 0x6480, 0xa441,
    0x6c00, 0xacc1, 0xad81, 0x6d40, 0xaf01, 0x6fc0, 0x6e80, 0xae41,
    0xaa01, 0x6ac0, 0x6b80, 0xab41, 0x6900, 0xa9c1, 0xa881, 0x6840,
    0x7800, 0xb8c1, 0xb981, 0x7940, 0xbb01, 0x7bc0, 0x7a80, 0xba41,
    0xbe01, 0x7ec0, 0x7f80, 0xbf41, 0x7d00, 0xbdc1, 0xbc81, 0x7c40,
    0xb401, 0x74c0, 0x7580, 0xb541, 0x7700, 0xb7c1, 0xb681, 0x7640,
    0x7200, 0xb2c1, 0xb381, 0x7340, 0xb101, 0x71c0, 0x7080, 0xb041,
    0x5000, 0x90c1, 0x9181, 0x5140, 0x9301, 0x53c0, 0x5280, 0x9241,
    0x9601, 0x56c0, 0x5780, 0x9741, 0x5500, 0x95c1, 0x9481, 0x5440,
    0x9c01, 0x5cc0, 0x5d80, 0x9d41, 0x5f00, 0x9fc1, 0x9e81, 0x5e40,
    0x5a00, 0x9ac1, 0x9b81, 0x5b40, 0x9901, 0x59c0, 0x5880, 0x9841,
    0x8801, 0x48c0, 0x4980, 0x8941, 0x4b00, 0x8bc1, 0x8a81, 0x4a40,
    0x4e00, 0x8ec1, 0x8f81, 0x4f40, 0x8d01, 0x4dc0, 0x4c80, 0x8c41,
    0x4400, 0x84c1, 0x8581, 0x4540, 0x8701, 0x47c0, 0x4680, 0x8641,
    0x8201, 0x42c0, 0x4380, 0x8341, 0x4100, 0x81c1, 0x8081, 0x4040};

//
// Description:
//   Inits crc.
//
void crcInit16(void)
{
  crc_value__ = 0xffff;
}


//
// Function: crc16Add(unsigned char value)
//
// Description:
//
void crc16Add(unsigned char value)
{
  unsigned char x = (unsigned char)(crc_value__ ^ (unsigned short)value);
  crc_value__ = (unsigned short)(crc_diagIn_value__ >> 8);
  crc_value__ ^= crc_16table[x];
```

```
}
//
// Function: unsigned short crcGet16(void)
//
// Description:
//   Returns current crc value
//
unsigned short crcGet16(void)
{
  return crc_value__;
}
```