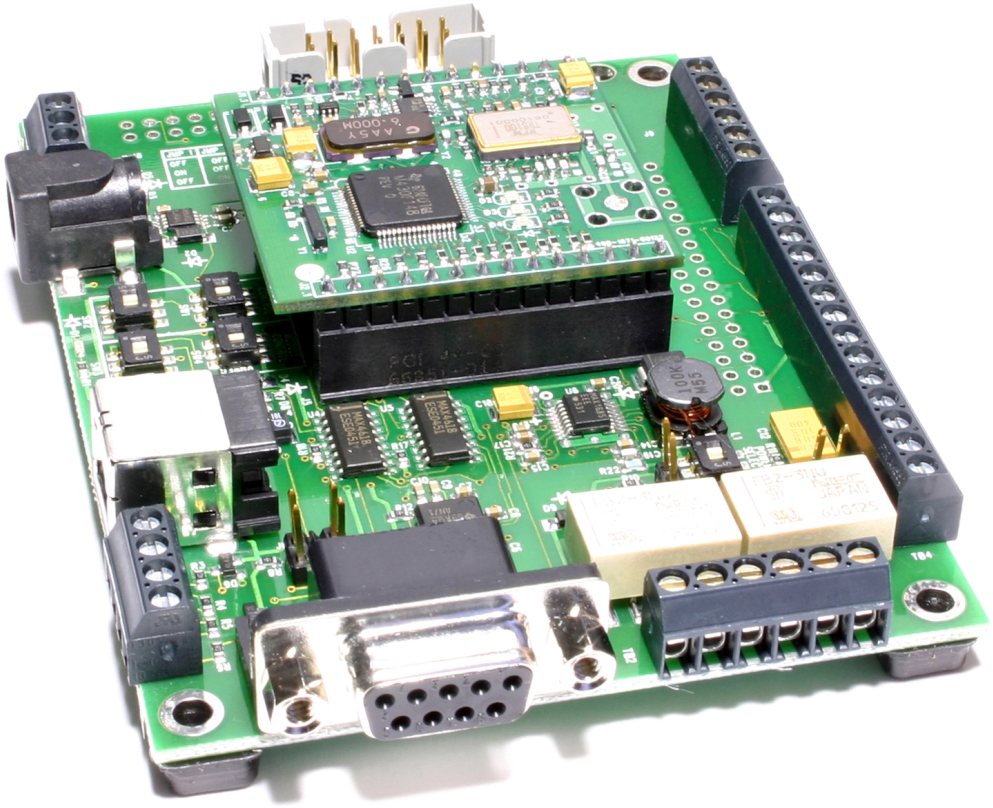
DM2200-916VM
Software User's Guide



Copyright 2006 - 2007 CIRRONET

1 INTRODUCTION **7**

1.1 DESIGN-IN SUPPORT 7
1.2 VERSAMESH™ MESSAGES 8
1.2.1 Node Configuration 8
1.2.2 Networking / Routing 9
1.2.3 Encryption 9
1.2.4 Power Management 9
1.2.5 Serial Port 9
1.2.6 Application I/O 10
1.2.7 Diagnostics 10

2 APPLICATIONS PROGRAMMING **11**

2.1 DM2200 NODE POWER-UP 12
2.1.1 Parameter Storage and Factory Defaults 12
2.1.2 LED Interpretation 13
2.2 DM2200 NODE RESET 14
2.3 COMMUNICATING WITH THE MASTER NODE 14
2.4 COMMANDS, RESPONSES, AND MESSAGES 14
2.4.1 Commands 14
2.4.1.1 Local and Remote 14
2.4.2 Responses 15
2.4.3 Event Messages 15
2.4.4 Data 16
2.5 CONFIGURING/DESIGNATING THE MASTER NODE 16
2.5.1 Example: Designating Master Node 0x0100 17
2.6 BINDING NODES INTO A NETWORK 18
2.6.1 Example: Binding Field Node 0x0025 18
2.6.2 Example: Binding Field Node 0x0035 19
2.6.3 Example: Binding Field Node 0x0055 19
2.7 BEACON OPERATION 19
2.7.1 Example: Setting Beacon on Master Node 0x0100 19
2.8 NETWORK MAINTENANCE 20
2.9 NODE MOBILITY (BEACONS) 20
2.10 POWER MANAGEMENT 20
2.10.1 Example: Setting Field Node Power Management 23
2.10.2 Example: Setting Master Node Power Management 23
2.11 CONFIGURING AND USING STATUS MESSAGES 23
2.11.1 Example: Setting Field Node 0x35 Status Message 23
2.12 RELIABLE REPORTING FROM FIELD NODES 23
2.12.1 Example: Setting Reliable Reporting Field Node 0x0025 24
2.13 APPLICATION I/O 24
2.13.1 Default Settings for I/O 25
2.13.2 Data Mode Serial Port Operation 25
2.13.3 Event Messages 26
2.13.4 Analog-to-Digital Converter (ADC) Operation 26
2.13.5 Digital Input Operation 26
2.13.6 Digital Output Operation 27
2.13.7 Application I/O Examples 27
2.13.7.1 Example: Pulse Counter 28
2.13.7.2 Example: Alarm GP2 28
2.13.7.3 Example: Reading Analog I/O 28
2.13.7.4 Example: Set Pins As Output High/Low 29
2.13.8 Pulse Monitoring Examples 29

Copyright 2006 - 2007 CIRRONET

2.13.8.1 Add CREG to comparator 29
2.13.8.2 Subtract CREG from the counter 30
2.13.8.3 Subtract comparator from the counter 30
2.14 ENCRYPTION 31

3 MESSAGE REFERENCE 32

3.1 VERSAMESH MESSAGE FORMATS: 32
3.2 EXAMPLE NETWORK 33
3.3 GENERAL MESSAGE LAYOUT 34
3.3.1 CMD_GET Messages 34
3.3.2 CMD_SET Messages 36
3.3.3 CMD_IO Messages 38
3.4 MESSAGE SPECIFIC FORMATS 40
3.4.1 msg_bindReq: 40
3.4.2 msg_io: 41
3.4.3 msg_data: 42
3.4.4 msg_br: 42
3.4.5 msg_traceAck: 43
3.4.6 msg_traceFAck: 43
3.4.7 msg_traceBAck: 44
3.4.8 msg_nhACK: 44
3.5 GLOBAL MESSAGES 45
3.5.1 CMD_LOCAL 45
3.5.2 CMD_MASTER 47
3.5.3 CMD_BIND 48
3.5.4 CMD_SACK 50
3.5.5 CMD_TRACE 52
3.5.6 CMD_RESET 54
3.6 GET MESSAGES 56
3.6.1 CMD_GET PAR_LH 56
3.6.2 CMD_GET ATTR_ESN 57
3.6.3 CMD_GET ATTR_MID 58
3.6.4 CMD_GET PAR_NID 59
3.6.5 CMD_GET PAR_TARP_F 60
3.6.6 CMD_GET PAR_TARP_L 62
3.6.7 CMD_GET PAR_TARP_S 64
3.6.8 CMD_GET PAR_TARP_R 66
3.6.9 CMD_GET PAR_BEAC 68
3.6.10 CMD_GET PAR_CON 69
3.6.11 CMD_GET PAR_RF 71
3.6.12 CMD_GET PAR_STAT_REP 72
3.6.13 CMD_GET PAR_RETRY 73
3.6.14 CMD_GET PAR_AUTOACK 74
3.6.15 CMD_GET ATTR_UPTIME 75
3.6.16 CMD_GET ATTR_RSSI 76
3.6.17 CMD_GET ATTR_MSG_COUNT_T 77
3.6.18 CMD_GET ATTR_MEM1 78
3.6.19 CMD_GET ATTR_MEM2 80
3.6.20 CMD_GET PAR_ENCR_M 82
3.6.21 CMD_GET PAR_ENCR_K 84
3.6.22 CMD_GET PAR_ENCR_D 86
3.6.23 CMD_GET ATTR_SYSVER 88
3.6.24 CMD_GET ATTR_NHOOD 89
3.6.25 CMD_GET PAR_BINDER 91
3.6.26 CMD_GET PAR_UART 92
3.6.27 CMD_GET PAR_CYC_CTRL 93

Copyright 2006 - 2007 CIRRONET

3.6.28 CMD_GET PAR_CYC_SLEEP 95
3.6.29 CMD_GET PAR_CYC_M_SYNC 96
3.6.30 CMD_GET PAR_CYC_M_REST 97
3.6.31 CMD_GET ATTR_CYC_LEFT 98
3.6.32 CMD_GET PAR_CMD_MODE 102
3.7 SET MESSAGES 103
3.7.1 CMD_SET PAR_LH 103
3.7.2 CMD_SET PAR_NID 105
3.7.3 CMD_SET PAR_TARP_F 107
3.7.4 CMD_SET PAR_TARP_L 109
3.7.5 CMD_SET PAR_TARP_S 111
3.7.6 CMD_SET PAR_TARP_R 113
3.7.7 CMD_SET PAR_BEAC 115
3.7.8 CMD_SET PAR_CON 117
3.7.9 CMD_SET PAR_RF 119
3.7.10 CMD_SET PAR_STAT_REP 121
3.7.11 CMD_SET PAR_RETRY 123
3.7.12 CMD_SET PAR_AUTOACK 125
3.7.13 CMD_SET PAR_ENCR_M 127
3.7.14 CMD_SET PAR_ENCR_K 129
3.7.15 CMD_SET PAR_ENCR_D 131
3.7.16 CMD_SET PAR_BINDER 133
3.7.17 CMD_SET PAR_UART 135
3.7.18 CMD_SET PAR_CYC_CTRL 137
3.7.19 CMD_SET PAR_CYC_SLEEP 140
3.7.20 CMD_SET PAR_CYC_M_SYNC 142
3.7.21 CMD_SET PAR_CYC_M_REST 145
3.7.22 CMD_SET PAR_CMD_MODE 147
3.8 I/O MESSAGES 149
3.8.1 CMD_IO IO_READ 149
3.8.2 CMD_IO IO_WRITE 152
3.8.3 CMD_IO IO_ADC 154
3.8.4 CMD_IO IO_CNT_START 156
3.8.5 CMD_IO IO_CMP_START 158
3.8.6 CMD_IO IO_NOT_START 159
3.8.7 CMD_IO IO_STOP 161
3.8.8 CMD_IO IO_CNT_GET 163
3.8.9 CMD_IO IO_CMP_GET 165
3.8.10 CMD_IO IO_STATE 167
3.8.11 CMD_IO IO_CREG_GET 169
3.8.12 CMD_IO IO_CREG_SET 171
3.8.13 CMD_IO IO_FLAGS_GET 173
3.8.14 CMD_IO IO_FLAGS_SET 175
3.8.15 CMD_IOACK 177
3.8.16 CMD_DAT 179
3.8.17 CMD_DATACK 184
4 GLOSSARY 186

Copyright 2006 - 2007 CIRRONET

FIGURES:

FIGURE 1 - APPLICATIONS GUIDE EXAMPLE 11
FIGURE 2 - LOCATION OF LEDS 13
FIGURE 3 - POWER MANAGEMENT EXAMPLE 21
FIGURE 4 – DM2200 BLOCK DIAGRAM 24
FIGURE 5 - I/O HOOKUPS USED IN EXAMPLES 27
FIGURE 6 - EXAMPLE NETWORK USED IN MESSAGES RESPONSES 33

Copyright 2006 - 2007 CIRRONET

DM2200-916VM and IM2200 Warranty

Limited Hardware Warranty. Cirronet, Inc. ("CIRRONET") warrants solely to the purchaser that the hardware DM2200-916VM and IM2200 modules will be free from defects in materials and workmanship under normal use for a period of 360 days from the date of shipment by CIRRONET. This limited warranty does not extend to any components which have been subjected to misuse, neglect, accident, or improper installation or application. CIRRONET's entire liability and the purchaser's sole and exclusive remedy for the breach of this Limited Hardware Warranty shall be, at CIRRONET's option, when accompanied by a valid receipt, either (i) repair or replacement of the defective components or (ii) upon return of the defective components, refund of the purchase price paid. **EXCEPT FOR THE LIMITED HARDWARE WARRANTY SET FORTH ABOVE, CIRRONET AND ITS LICENSORS PROVIDE THE HARDWARE ON AN "AS IS" BASIS, AND WITHOUT WARRANTY OF ANY KIND EITHER EXPRESS, IMPLIED OR STATUTORY, INCLUDING BUT NOT LIMITED TO THE IMPLIED WARRANTIES OF NONINFRINGEMENT, MERCHANTABILITY OR FITNESS FOR A PARTICULAR PURPOSE.** Some states do not allow the exclusion of implied warranties, so the above exclusion may not apply to you. This warranty gives you specific legal rights and you may also have other rights which vary from state to state.

Limitation of Liability. IN NO EVENT SHALL CIRRONET OR ITS SUPPLIERS BE LIABLE FOR ANY DAMAGES (WHETHER SPECIAL, INCIDENTAL, CONSEQUENTIAL OR OTHERWISE) IN EXCESS OF THE PRICE ACTUALLY PAID BY YOU TO CIRRONET, REGARDLESS OF UNDER WHAT LEGAL THEORY, TORT, OR CONTRACT SUCH DAMAGES MAY BE ALLEGED (INCLUDING, WITHOUT LIMITATION, ANY CLAIMS, DAMAGES, OR LIABILITIES FOR LOSS OF BUSINESS PROFITS, BUSINESS INTERRUPTION, LOSS OF BUSINESS INFORMATION, OR FOR INJURY TO PERSON OR PROPERTY) ARISING OUT OF THE USE OR INABILITY TO USE CIRRONET PRODUCTS, EVEN IF CIRRONET HAS BEEN ADVISED OF THE POSSIBILITY OF SUCH DAMAGES. BECAUSE SOME STATES DO NOT ALLOW THE EXCLUSION OR LIMITATION OF LIABILITY FOR CONSEQUENTIAL OR INCIDENTAL DAMAGES, THE ABOVE LIMITATION MAY NOT APPLY TO YOU.

Notice on the Restricted use of the DM2200-916VM and IM2200

The DM2200-916VM and IM2200, and any products employing them, operate on a shared radio band. Radio interference can occur in any place at any time, and thus communications may not be absolutely reliable. Products using the DM2200-916VM and IM2200 must be designed so that a loss of communications due to radio interference or otherwise will not endanger either people or property, and will not cause the loss of valuable data. CIRRONET assumes no liability for the performance of products which are designed or created using the DM2200-916VM and IM2200. **CIRRONET products are not suitable for use in life-support applications, biological hazard applications, nuclear control applications, or radioactive areas.**

Copyright 2006 - 2007 CIRRONET

1 Introduction

This *Software User's Guide* provides the DM2200-916VM application software developer with a walk-through of configuring and controlling DM2200 nodes, including examples of common operations with the corresponding messages to send, their responses (Section 2, p. 11), a complete listing of all messages supported by the DM2200 (Section 3, p. 32), and a glossary of terms. For further details about the DM2200, refer to the documents listed below.

1.1 Design-In Support

Cirronet offers comprehensive support to customers designing DM2200-916VM modules into their products and systems with:

- *DM2200-916VM System User's Guide* - The System User's Guide covers incorporating DM2200-916VM modules into products and systems, including how to install, configure, operate, and maintain a DM2200 network.

- *DM2200-916VM, DM2200-916VM-1 (RoHS), IM2200 and IM2200-1 (RoHS) Data Sheets* - These documents cover the specifications of the DM2200-916VM Transceiver Modules and the optional IM2200 and IM2200-1 Application Interface Modules.

- *DM2200-916VM Software User's Guide* - This document covers the use and format of all application programming interface (API) messages, plus how to configure and operate a DM2200 network with your application program using the API.

- *DM2200-916-DK Development Kits* - Each kit includes four DM2200-916VM nodes (a master and three field nodes), four IM2200-1 interface boards, four sets of power supplies, plus software utilities and documentation.

- *Factory and Field Application Support* - Cirronet's application engineering staff provides customer support by e-mail, phone, FAX, field seminars, and on-site visits.

- *DM2200 Customization* - Cirronet offers hardware and software customization services for specific applications.

Copyright 2006 - 2007 CIRRONET

1.2 VersaMESH™ Messages

DM2200's employ Cirronet's VersaMESH protocol. There are more than 80 command, response, event and data messages supported by VersaMESH. Messages are organized into the seven categories listed below. Refer to Section 3 for full details on each message.

The following is a list of mnemonics used in various commands:

ATTR	Attribute	ESN	Equipment Serial Number
BEAC	Beacon	MID	Master Node ID
CMD	Command	NHOOD	Neighborhood
CREG	Comparator Register	NID	Network ID
CYC	Cycle	PAR	Parameter

1.2.1 Node Configuration

Get Local Node Network Configuration	CMD_LOCAL	3.5.1	p. 45
Reset Node	CMD_RESET	3.5.6	p. 54
Designate Master Node	CMD_MASTER	3.5.2	p. 47
Get Master ID	CMD_GET ATTR_MID	3.6.3	p. 58
Set Network ID	CMD_SET PAR_NID	3.7.2	p. 105
Get Network ID	CMD_GET PAR_NID	3.6.4	p. 59
Set Node ID	CMD_SET PAR_LH	3.7.1	p. 103
Get Node ID	CMD_GET PAR_LH	3.6.1	p. 56
Get Firmware Version	CMD_GET ATTR_SYSVER	3.6.23	p. 88
Get Equipment Serial Number	CMD_GET ATTR_ESN	3.6.2	p. 57
Set Node Bind Enable State	CMD_SET PAR_BINDER	3.7.16	p. 133
Enable/Disable Master Automatic ACK	CMD_SET PAR_AUTOACK	3.7.12	p. 125
Get Beacon Interval	CMD_GET PAR_BEAC	3.6.9	p. 68
Get Event Message Retries and Timeout	CMD_GET PAR_RETRY	3.6.13	p. 73
Get Master RSSI	CMD_GET ATTR_RSSI	3.6.16	p. 76
Get Node Bind Enable State	CMD_GET PAR_BINDER	3.6.25	p. 91
Get Node Memory Stats (1)	CMD_GET ATTR_MEM1	3.6.18	p. 78
Get Node Memory Stats (2)	CMD_GET ATTR_MEM2	3.6.19	p. 80
Get Node to Request Neighbors to Report	CMD_GET ATTR_NHOOD	3.6.24	p. 89
Get Radio Mode	CMD_GET PAR_RF	3.6.11	p. 71
Set Radio Mode ON/OFF	CMD_SET PAR_RF	3.7.9	p. 119
Set Node Status Reporting Interval	CMD_SET PAR_STAT_REP	3.7.10	p. 121
Get Status Message Reporting Interval	CMD_GET PAR_STAT_REP	3.6.12	p. 72
Send Message ACK	CMD_SACK	3.5.4	p. 50
Set Event Message Retries and Timeout	CMD_SET PAR_RETRY	3.7.11	p. 123

Copyright 2006 - 2007 CIRRONET

1.2.2 Networking / Routing

Bind Request Event Message	*msg_bindreq*	3.4.1	p. 40
Status Event Message	*msg_br*	3.4.4	p. 42
Bind Node to Network	CMD_BIND	3.5.3	p. 48
Set Connectivity Data	CMD_SET PAR_CON	3.7.8	p. 117
Get Connectivity Data	CMD_GET PAR_CON	3.6.10	p. 69
Set Route Forwarding	CMD_SET PAR_TARP_F	3.7.3	p. 107
Get Route Forwarding	CMD_GET PAR_TARP_F	3.6.5	p. 60
Set Routing Level	CMD_SET PAR_TARP_L	3.7.4	p. 109
Get Routing Level	CMD_GET PAR_TARP_L	3.6.6	p. 62
Set Route Recovery	CMD_SET PAR_TRAP_R	3.7.6	p. 113
Get Route Recovery	CMD_GET PAR_TRAP_R	3.6.8	p. 66
Set Route Slack	CMD_SET PAR_TRAP_S	3.7.5	p. 111
Get Route Slack	CMD_GET PAR_TRAP_S	3.6.7	p. 64
Set Master Beacon Interval	CMD_SET PAR_BEAC	3.7.7	p. 115
Get Master Beacon Interval	CMD_GET PAR_BEAC	3.6.9	p. 68

1.2.3 Encryption

Set Encryption Configuration	CMD_SET PAR_ENCR_M	3.7.13	p. 127
Get Encryption Configuration	CMD_GET PAR_ENCR_M	3.6.20	p. 82
Set Encryption Configuration	CMD_SET PAR_ENCR_K	3.7.14	p. 129
Get Encryption Configuration	CMD_GET PAR_ENCR_K	3.6.21	p. 84
Set Encryption Configuration	CMD_SET PAR_ENCR_D	3.7.15	p. 131
Get Encryption Configuration	CMD_GET PAR_ENCR_D	3.6.22	p. 86

1.2.4 Power Management

Get Node Power Management Current State	CMD_GET ATTR_CYC_LEFT	3.6.31	p. 98
Set Node Sleep Mode Configuration	CMD_SET PAR_CYC_CTRL	3.7.18	p. 137
Get Node Sleep Mode Configuration	CMD_GET PAR_CYC_CTRL	3.6.27	p. 93
Set Master Node Power Mode Sync Period	CMD_SET PAR_CYC_M_SYNC	3.7.20	p. 142
Get Master Node Power Mode Sync Period	CMD_GET PAR_CYC_M_SYNC	3.6.29	p. 96
Set Master Node Rest Period	CMD_SET PAR_CYC_M_REST	3.7.21	p. 145
Get Master Node Rest Period	CMD_GET PAR_SYS_M_REST	3.6.30	p. 97
Set Sleep Period	CMD_SET PAR_CYC_SLEEP	3.7.19	p. 140
Get Sleep Period	CMD_GET PAR_SYS_SLEEP	3.6.28	p. 95

1.2.5 Serial Port

Set Serial Port Configuration (UART)	CMD_SET PAR_UART	3.7.17	p. 135
Get Serial Port Configuration (UART)	CMD_GET PAR_UART	3.6.26	p. 92
Set Serial Port Mode (CMD/DATA)	CMD_SET PAR_CMD_MODE	3.7.22	p. 147
Get Serial Port Mode (CMD/DATA)	CMD_GET PAR_CMD_MODE	3.6.32	p. 102

Copyright 2006 - 2007 CIRRONET

1.2.6 Application I/O

Set AutoACK Status	CMD_SET PAR_AUTOACK	3.7.12	p. 125
Get AutoACK Status	CMD_GET PAR_AUTOACK	3.6.14	p. 74
Comparator and GP2 Event Message	*msg_io*	3.4.2	p. 41
Output Payload to Serial Port	CMD_DAT	3.8.16	p. 179
Send ACK to a Data Message	CMD_DATACK	3.8.17	p. 184
Get Analog Input Measurement	CMD_IO IO_ADC	3.8.3	p. 154
Set Comparator Value	CMD_IO IO_CMP_START	3.8.5	p. 158
Get Comparator Value	CMD_IO IO_CMP_GET	3.8.9	p. 165
Start Pulse Counter (GP1) & Enable Comparator Event Messages	CMD_IO IO_CNT_START	3.8.4	p. 156
Get Pulse Counter	CMD_IO IO_CNT_GET	3.8.8	p. 163
Set Offset Value	CMD_IO IO_CREG_SET	3.8.12	p. 171
Get Offset Value	CMD_IO IO_CREG_GET	3.8.11	p. 169
Set Offset Mode	CMD_IO IO_FLAGS_SET	3.8.14	p. 175
Get Offset Mode	CMD_IO IO_FLAGS_GET	3.8.13	p. 173
Enables Event Message on GP2	CMD_IO IO_NOT_START	3.8.6	p. 159
Get Digital I/O State	CMD_IO IO_READ	3.8.1	p. 149
Get Pulse Counting State	CMD_IO IO_STATE	3.8.10	p. 167
Stop Pulse Counter	CMD_IO IO_STOP	3.8.7	p. 161
Set I/O Pin Configuration	CMD_IO IO_WRITE	3.8.2	p. 152
Acknowledges an I/O Event	CMD_IOACK	3.8.15	p. 177

1.2.7 Diagnostics

Trace Acknowledgement	*msg_traceAck,*	3.4.5	p. 43
	msg_traceFAck,	3.4.6	p. 43
	msg_traceBAck	3.4.7	p. 44
Get Uptime Status	CMD_GET ATTR_UPTIME	3.6.15	p. 75
Send Trace	CMD_TRACE	3.5.5	p. 52
Get TARP Message Counters	CMD_GET ATTR_MSG_COUNT_T	3.6.17	p. 77

Copyright 2006 - 2007

2 Applications Programming

This section provides examples of various procedures commonly used with the DM2200. The main example is a step-by-step method for configuring a master node and three field nodes into a working VersaMESH network. Figure 1 is used as the basis for the steps and examples throughout this section. The example is a master node and three field nodes with I/O associated with specific nodes.

Note: For this example, field node 0x0055 talks to field node 0x0035 and assumes that the node 0x0055 cannot hear master node 0x0100. This is only achieved when field node 0x0055 cannot hear master node 0x0100.

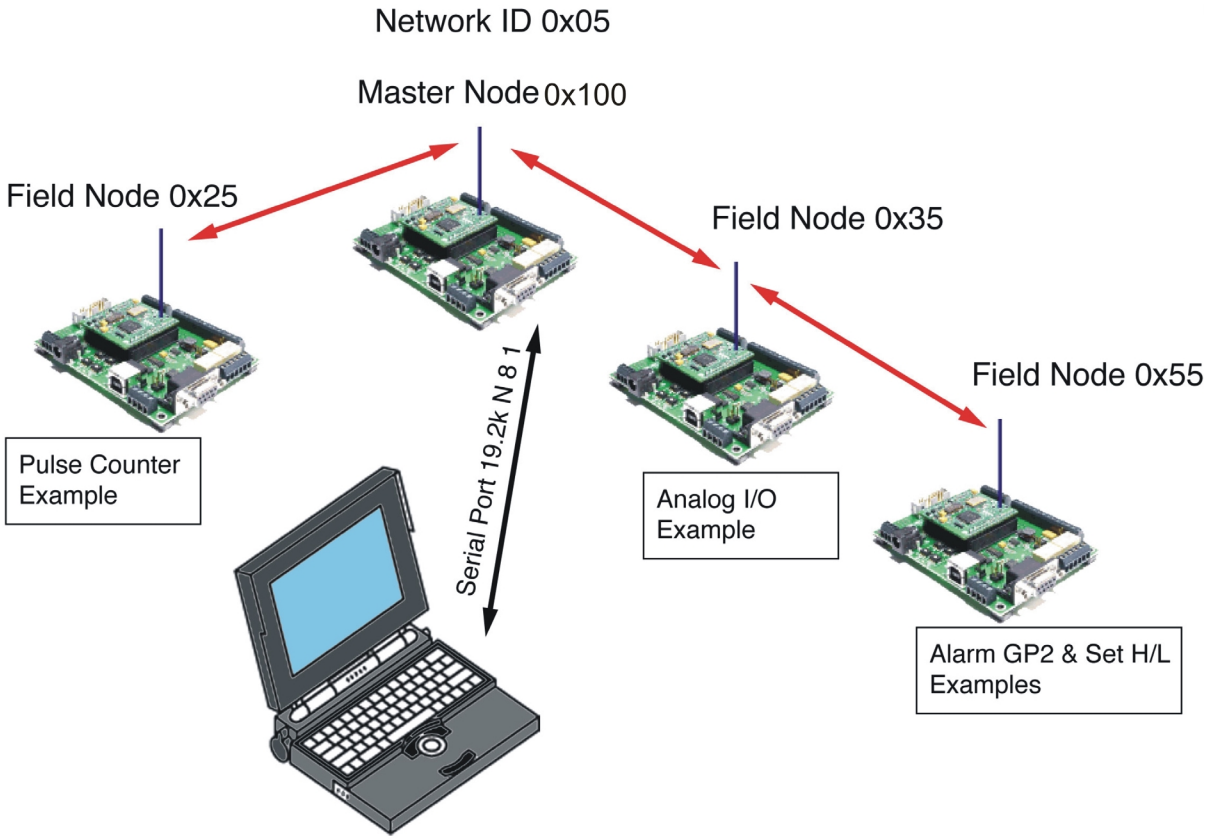


Figure 1 - Applications Guide Example

Copyright 2006 - 2007

2.1 DM2200 Node Power-up

Certain parameter values are stored in nonvolatile memory when changed from their factory defaults, and are retrieved to configure a DM2200 node when it is powered up. If there is no updated value stored in nonvolatile memory for one of these parameters, then the factory default value is used.

2.1.1 Parameter Storage and Factory Defaults

The factory default values for the parameters discussed in Section 2.1 are shown in the table below:

Parameter	*Value*
Network ID	0
Node ID	lower 16 bits of the node's ESN
Host ID	0
Encryption data	Mode = 0 Key = 0
Command mode flag	1
Bind enable flag	1 (listens and forwards all bind request messages from unbound nodes)
TARP network routing parameters	level = 2 slack = 2 route recovery = 1 forwarding = 1
Power management data	mode = network state = enabled on field nodes, disabled on master sleep synchronization/preparation time = 60 seconds sleep time = 90 seconds master rest time = 120 seconds
I/O data	comparator value = 0 counter value = 0 offset value = 0 state = all disabled

Changes to the following parameter values are not stored in nonvolatile memory, and are lost when the DM2200 is turned off. When a DM2200 is powered up, factory default values are used:

Number of retries for status and I/O event messages	3
Status reporting acknowledgement timeout	10 seconds
Number of missed beacons that triggers the connectivity LED to start blinking	3
Number of additional missed beacons that triggers the connectivity LED off	4
Beacon status reporting period	8 hours

After reading parameter values from nonvolatile memory and program memory, the node starts or resumes its functionality.

Copyright 2006 - 2007

2.1.2 LED Interpretation

There is a green, yellow and red LED located on top of the DM2200, as shown in Figure 2. The green LED blinks when the node receives a packet. The yellow LED blinks when the node transmits a packet. The red LED provides connectivity status as follows:

~5 blinks per sec (fast)	Node Unbound
~2 blinks per sec (slow)	Bound, Marginal Connection
Solid On	Bound, Good Connection
Off	Powered Off, in Sleep mode, or Bound but no recent Network Activity

- Fast blinking indicates the node is not bound to a network
- Slow blinking indicates the node is bound, but has not heard a packet from the master node since waking from sleep, powering up, or for a length of time longer than a specified number of beacon intervals.
- Solid ON indicates the node is bound and hearing most beacon traffic from the master node
- Solid OFF indicates the node is powered down, in a sleep cycle, or is on and bound, but has not heard a packet from the master node in a length of time longer than a specified number of beacon intervals.

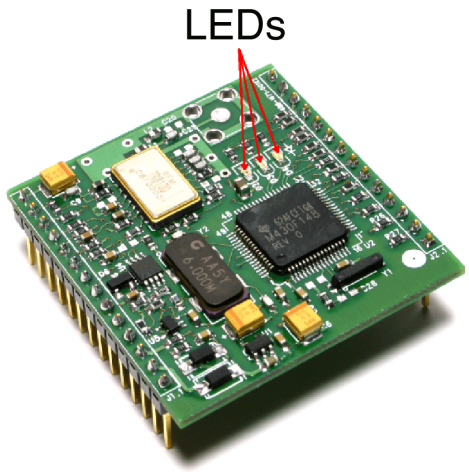


Figure 2 - Location of LEDs

Copyright 2006 - 2007

2.2 DM2200 Node Reset

It is possible to reset the configuration of a DM2200 module to one of two states by either grounding its reset pin, or by using CMD_RESET [3.5.6, p. 54]. The hardware reset pin on a DM2200 is CFG3. This pin must have an external pull-up resistor (100K) to assure stable operation. The hardware reset states are selected by either grounding the reset pin with a button or driving the reset pin low with an external logic signal as discussed below.

Press and hold for less than 4 seconds	Reset Node, Clear memory
Press and hold for more than 7 seconds	Reset Node, Reset Unit to Factory Defaults

If the button is released in less than four seconds after the LEDs blink, the DM2200 resets using parameters stored in nonvolatile memory. If the button is pressed for more than seven seconds, the LEDs come back on and the DM2200 erases its nonvolatile memory. The LEDs then blink seven times and the module resets using factory default configuration parameters.

2.3 Communicating with the Master Node

The nodes in the DM2200-916-DK Development Kit are preconfigured as one master (blue label) and three field nodes. Only the master node has to be connected to a host computer. The connection can be either an RS232 serial connection or a USB connection. The details for setting up the master node for RS232 or USB can be found in the DM2200-916-DK manual.

2.4 Commands, Responses, and Messages

The default serial port mode for a DM2200 is called the *command* mode. This mode handles four types of messages: commands, responses, event messages and data messages.

2.4.1 Commands

Command messages are used to control the operation and configuration of a DM2200 node. Commands always control the master node directly through its serial port, and they usually control field nodes through the RF network.

In the *command* mode, a node is expecting byte level commands and will send byte level responses. Examples of these are given in this section and in detail in section 3.

Most, but not all commands automatically invoke a response. Commands that does not automatically cause a response can be verified by setting an acknowledgement (ACK) request flag in the command. This causes the destination node to send a response back to the source of the command.

2.4.1.1 Local and Remote

The master node provides the local network connection to the application. Field nodes are usually accessed through the RF network, but can also process many commands directly though their serial port. However, some commands work differently when sent to the master node versus a field node.

Copyright 2006 - 2007

2.4.2 Responses

Response messages are returned for most commands. For example, the response to an IO_ADC command (read ADC input) is a message containing the ADC reading.

Some responses are simply a repeat back of the command with an additional byte indicating success.

For example, CMD_SET PAR_LH [3.7.1, p. 103] sets the node ID of a field node using the following command layout:

BOT	op_len	**CMD_SET**	op_ref	node_id	**payload**	EOT
0x00	0x07	**0x02**	X(1B)	X(2B)	X(3B)	0x04

Payload:

0x01	B1(node_id)	B2(node_id)

For example, the command to change the address of node 0x0025 to 0x0045 is:

```
00 07 02 BB 25 00 01 45 00 04
```

The response is:

BOT	op_len	**CMD_SET**	op_ref	**op_rc**	node_id	**payload**	EOT
0x00	0x08	**0x02**	X(1B)	X(1B)	X(2B)	X(3B)	0x04

Payload:

0x01	B1(node_id)	B2(node_id)

Note: The additional byte in the response message is called an *op_rc* byte. This is the *return code*. If *op_rc* is 0x00, then the command was successful, as in:

```
00 08 02 BB 00 25 00 01 45 00 04
```

2.4.3 Event Messages

Event messages are generated by field nodes under certain conditions, and are always directed to the master node. For example, field node can be configured to send an event message when the value in its pulse count register and its comparator register match.

Copyright 2006 - 2007

2.4.4 Data

The *data* mode permits serial data to be sent from a remote node to the master node, and then on to the host. The remote node must be placed in data mode using the CMD_SET PAR_CMD_MODE command. Once in data mode, a string of bytes input at the remote node is relayed to the master node.

Serial data may be sent from the host computer, via the master node, to a remote node using the CMD_DAT command.

2.5 Configuring/Designating the Master Node

The first configuration step in creating a DM2200 network is to designate a *master node*. The master node is unique in four ways:

- The master node can initiate RF commands to *bind* (add) other nodes into the network.
- The master node is always the destination for *status* and *event messages* generated by other nodes in the network.
- The master node periodically sends *beacons* that are used by the other nodes in the network to determine how many hops they are from the master node.
- The master node beacon orchestrates the network's power management sleep cycles.

A node is designated as the network master by sending it a sequence of configuration commands. Here is an overview of the command sequence:

- CMD_LOCAL [3.5.1, p. 45] is sent to the node to retrieve its current node ID, electronic serial number (ESN), and master ID.
- CMD_SET PAR_LH [3.7.1, p. 103] is then sent updating the node ID. This ID must be unique within the network.
- CMD_SET PAR_CON [3.7.8, p. 117] is sent, using the new node ID, to set the value of several beacon connectivity parameters. These will be passed on to any node being bound to the master node's network.
- CMD_SET PAR_NID [3.7.2, p. 105] is then sent to set the network ID. Only messages carrying this node ID will be handled by the network. This allows overlapping networks to operate independently.
- CMD_SET PAR_BEAC [3.7.7, p. 115] is sent to set the master node beacon interval, typically in the range of 10 to 60 seconds.
- Finally, CMD_MASTER [3.5.2, p. 47] is sent to set the operating mode of the node as a master node.

Copyright 2006 - 2007

A newly configured master node immediately sends out its first beacon to announce its designation as the network master. The master node configuration is preserved when a node is powered off. However, CMD_MASTER [3.5.2, p. 47] must be sent to the node again following power-up to reactive master node beacon operation.

2.5.1 Example: Designating Master Node 0x0100

Send the CMD_LOCAL command to get the ESN of the node:

SEND[7]: `00 04 0A 81 00 00 04`
RECV[14]: `00 0B 0A 81 00 BE 07 BE 07 00 00 00 00 04`
The node returned an ESN of `0x00 0x00 0x07 0xBE`

Using the two low order ESN bytes (0x07, 0xBE), set the node ID to 0x0100:

Send CMD_SET PAR_LH
SEND[10]: `00 07 02 82 BE 07 01 00 01 04`
RECV[11]: `00 08 02 82 00 BE 07 01 00 01 04`

Set node 0x0100 connectivity parameters:

Send CMD_SET PAR_CON
SEND[10]: `00 07 02 83 00 01 0A 1C 23 04`
RECV[11]: `00 08 02 83 00 00 01 0A 1C 23 04`

Set node 0x0100 network ID to 16 (0x0010):

Send CMD_SET PAR_NID
SEND[10]: `00 07 02 84 00 01 04 10 00 04`
RECV[11]: `00 08 02 84 00 00 01 04 10 00 04`

Set node 0x0100 beacon frequency to 15 (0x0F) seconds:

Send CMD_SET PAR_BEAC
SEND[9]: `00 06 02 85 00 01 09 0F 04`
RECV[10]: `00 07 02 85 00 00 00 01 0F 04`

Set node 0x0100 encryption key to 0x00:

Send CMD_SET PAR_ENCR_K
SEND[9]: `00 06 02 86 00 01 18 00 04`
RECV[10]: `00 07 02 86 00 00 01 18 00 04`

Set node 0x0100 encryption mode to 0x00:

Send CMD_SET PAR_ENCR_M
SEND[9]: `00 06 02 87 00 01 17 00 04`
RECV[10]: `00 07 02 87 00 00 01 17 00 04`

Make this node the master (use address 00 00) and beacon to all other nodes:

Send CMD_MASTER
SEND[7]: `00 04 03 88 00 00 04`
RECV[8]: `00 05 03 88 00 00 00 04`

Copyright 2006 - 2007

This last command will change the node's red LED to solid on.

2.6 Binding Nodes into a Network

Once a master node has been designated, the rest of the nodes can be bound to the network. When an unbound node is turned on (network ID is not set), it periodically transmits a bind request (*msg_bindreq* [3.4.1, p. 40]). The node sets its node ID to the least significant word of its ESN, and sends out periodic messages to advertise its presence. The connectivity LED is fast blinking. If the unbound node can communicate directly with the master node, the master acts on the request directly. Otherwise, the bind request is routed to the master node starting with a bound node(s) that can directly hear the bind request and is configured to forward it. Once a bind request is received by the master node, it is passed to the master node's host computer.

The role of the host computer in the binding process is to determine if the node generating the bind request is authorized to become part of the network. If so, the host computer issues CMD_BIND [3.5.3, p. 48] through the master node. If the unbound node can communicate directly with the master node, the command is sent directly to the unbound node. Otherwise, the command is sent to a bound node that is in direct communication with the unbound node.

It is good practice to power up and bind nodes one at a time before field installation. This avoids overloading a new network with bind requests and bind command packets.

When a node is bound, it receives its unique node ID, the network ID, the master node ID, connectivity parameters, and encryption data. Next, it sends a *msg_br* [3.4.4, p. 42] status message to the master node. The node is then ready to accept I/O configuration commands, and to participate in network activities. The interval between status messages is set with CMD_SET PAR_STAT_REP [3.7.10 p. 121]. The default interval is eight hours.

2.6.1 Example: Binding Field Node 0x0025

Connect power to the unbound field node.

Observe the *msg_bindreq* message:
RECV[13]: `00 0A FF 06 61 07 00 00 61 07 00 01 04`
Determine the ESN from the received message (0x00, 0x00, 0x07, 0x61).

Send the CMD_BIND message to the unbound field node.

Specify a node ID of 0x0025, setting the number of permitted hops to 1:
SEND[14]: `00 0B 05 81 00 01 61 07 00 00 25 00 01 04`
RECV[15]: `00 0C 05 81 00 00 01 61 07 00 00 25 00 01 04`
This message associates the new node with the master and the network.

The master node is beaconing at the established beacon rate. When node 0x0025 hears the beacon, binding is complete. The red LED changes to solid, and the node sends a *msg_br* status message to the master node.

Copyright 2006 - 2007

RECV[14]: 00 0B FF 08 25 00 00 23 00 00 00 00 01 04

2.6.2 Example: Binding Field Node 0x0035

Set the node ID to 0x0035:

RECV[13]: 00 0A FF 06 54 07 00 00 54 07 00 01 04

SEND[14]: 00 0B 05 82 00 01 54 07 00 00 35 00 01 04
RECV[15]: 00 0C 05 82 00 00 01 54 07 00 00 35 00 01 04

RECV[14]: 00 0B FF 08 35 00 00 23 00 00 00 00 01 04

2.6.3 Example: Binding Field Node 0x0055

Set node ID to 0x0055, and set the number of permitted hops to 2:

RECV[13]: 00 0A FF 06 C8 07 00 00 C8 07 00 01 04

SEND[14]: 00 0B 05 83 00 01 C8 07 00 00 55 00 02 04
RECV[15]: 00 0C 05 83 00 00 01 C8 07 00 00 55 00 02 04

RECV[14]: 00 0B FF 08 55 00 00 23 00 00 00 00 01 04

2.7 Beacon Operation

Beacons sent by the master node are key to network operation. Beacons are monitored by each field node to optimize and/or recover routing to the master node, to assess path robustness, and to synchronize power management cycles. The beacon period is a configurable parameter, with a default value of 29 seconds. Setting the beacon period longer with CMD_SET PAR_BEAC [3.7.7, p. 115] helps improve power management and extend battery life, with the trade-off of slower overall network dynamics. Setting the beacon period shorter improves the dynamics of a network, but lessens the effectiveness of power management.

The beacon interval must be coordinated with the maximum number of hops required by the network. As a general rule, 250 ms should be allocated for each hop in a DM2200 network. The beacon period should be set no less than 10 times the propagation time through the maximum number of hops in the network. For example, if a network uses a maximum for four hops, the maximum route propagation time is one second. The beacon period should then be at least 10 seconds.

2.7.1 Example: Setting Beacon on Master Node 0x0100

Set the master beacon to 30 seconds using CMD_SET PAR_BEAC.

Set the master node (0x0100) beacon to 30 (0x1E) seconds:

SEND[9]: 00 06 02 85 00 01 09 1E 04
RECV[10]: 00 07 02 85 00 00 01 09 1E 04

Copyright 2006 - 2007

2.7.2 Network Maintenance

Field nodes periodically send *msg_br* [3.4.4, p. 42] status messages to the master node. This allows the host computer at the master node to track connectivity to each field node without explicitly sending a command to it.

VersaMESH also includes trace diagnostics [3.5.5, p. 52] for pinpointing network problems is real time.

The red connectivity LED on each field node provides a local indication of the path robustness to the master node. See section 2.1.2.

2.8 Node Mobility (Beacons)

Master node beacons allow VersaMESH to maintain a node's connectivity as it physically moves through the network, provided the motion is not too fast compared to the beacon period. Also, a single network can consist of a number of isolated "cluster" locations, for example warehouses located in different cities. Provided the same network ID and encryption key is used at each cluster location, a node can leave one cluster and join another cluster without needing to rebind. Joining a new cluster is facilitated by receiving the master beacon, which allows the joining node to recognize the address of the local master.

2.9 Power Management

Network power management cycles are controlled by the master node, which is always *awake* or active. When power management is *enabled*, the master node alternates between two states, *synchronization* and *rest*. The period for each of these states is programmable. During the synchronization period, the master node adds "*seconds until sleep*" data to its master beacons. At the start of the sleep period, the master node switches to its rest state. During the rest period, the master node continues to send out beacons, but without sleep synchronization data. At the end of its rest period, the master node switches back to its synchronization state, beginning a new power management cycle.

The field node counterpart to synchronization is *preparation*. A field node only has to hear one master beacon with synchronization data to prepare for its sleep period. During a field node's preparation period, it continues normal operation but runs its own "*seconds to sleep*" timer that has been synchronized with the master beacon data.

The sleep *mode* and the sleep *period* are both programmable for each field node. There are four possible sleep modes:

- *Deep sleep*, with the radio turned off and the microprocessor running at minimum power consumption for a specified time period.

- *Light sleep*, with the radio turned off and the microprocessor configured to service interrupts for pulse counting, etc., for a specified time period.

Copyright 2006 - 2007

- *Awake*, for those nodes in the network running from mains power. Awake nodes operate continuously.

- *Transmit only*, for "leaf" nodes at the edge of a network that transmit only event messages. Transmit only nodes must be supported by routing paths consisting of awake nodes.

By default, a field node is configured with sleep enabled, the sleep mode set to deep sleep, and with a 90 second sleep period. Note that enabling sleep in a field node does not itself cause the node to sleep. Each sleep cycle is initiated by receiving a master beacon with synchronization data, which switches the field node into sleep preparation and then into its sleep period.

When a node becomes the master node, sleep is disabled, the synchronization period defaults to 60 seconds, and rest period defaults to 120 seconds. Any changes to default sleep mode parameters in either a master node or a field node are stored in nonvolatile memory, and are retrieved when the node is powered up.

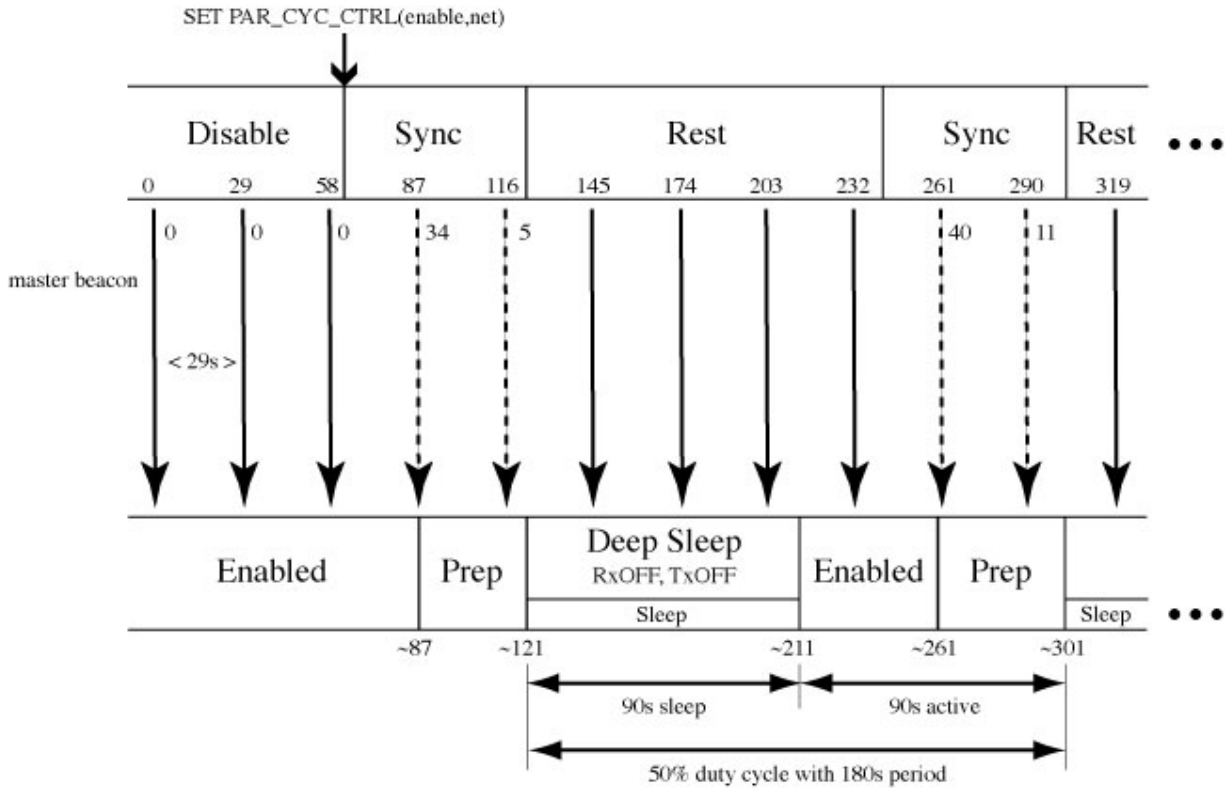


Figure 3 - Power Management Example

The sleep period for a field node is usually set shorter than the master node rest period, allowing an interval for active operation during each master node power management

Copyright 2006 - 2007

cycle. However, it is also possible to set the sleep period on a field node to span a number of master node cycles, where operation of the field node can be infrequent and very low power usage is important.

In an example above (Figure 3), SET PAR_CYC_CTRL is issued to the master node at time 61, specifying *rest* state (0x01) and *net* mode (0x00). This starts a process that cycles the master node between *synchronization* and *rest* states, according to the period settings for each state (in this case, a 60 second synchronization period and a 120 second rest period). During the *synchronization* period, master beacons will contain *"seconds until sleep"* information calculated by the master node. When a field node receives the synchronization data in its enabled state, it starts a process to stay active until the start of the sleep period (*preparation* state). It then transitions into its locally configured sleep state (*deep sleep* in the above example) for its locally configured sleep time (90 seconds). After the sleep interval, the node returns to active operation. This cycle repeats under the control of the master beacon.

Note that power management SET commands must be executed in a specific order.

For a field node:

- If sleep is enabled (default), disable sleep mode using SET PAR_CYC_CTRL with *state* set to disabled (0x00) and *mode* set to its current value (default is 0x00). This allows the sleep period and the sleep mode to be changed.
- Update the sleep period using SET PAR_CYC_SLEEP.
- Using SET PAR_CYC_CTRL again, set the *state* to enabled (0x01), and the sleep *mode* as desired (0x00, 0x01, 0x02, 0x03).

The sleep configuration for each field nodes is usually set before enabling power management at the master node.

For the master node:

- If power management is enabled, disable with SET PAR_CYC_CTRL, with *state* set to disabled (0x00) and *mode* set to net (0x00).
- Update the master node synchronization period using SET PAR_M_SYNC.
- Update the master node rest period using SET PAR_M_REST.
- Enable power management using SET PAR_CYC_CTRL, with *state* set to rest (0x01) and *mode* set to net (0x00).

Note that the master node synchronization period must be greater than the master beacon interval, or synchronization data may never be transmitted. The synchronization period is

Copyright 2006 - 2007

usually set to more than twice the master beacon interval.

2.9.1 Example: Setting Field Node Power Management

Send CMD_SET PAR_CYC_CTRL, state disabled (0x00), mode deep sleep (0x00):

SEND[10]: `00 07 02 88 35 00 1E 00 00 04`
RECV[11]: `00 08 02 88 01 35 00 1E 00 00 04`

Send CMD_SET PAR_CYC_SLEEP, sleep period of 60 seconds:

SEND[12]: `00 09 02 88 35 00 1F 3C 00 00 00 04`
RECV[13]: `00 0A 02 88 02 35 00 1F 3C 00 00 00 04`

Send CMD_SET PAR_CYC_CTRL, state enabled (0x01), mode deep sleep (0x00):

SEND[10]: `00 07 02 88 35 00 1E 01 00 04`
RECV[11]: `00 08 02 88 01 35 00 1E 01 00 04`

2.9.2 Example: Setting Master Node Power Management

Send CMD_SET PAR_CYC_CTRL, state disabled (0x00), mode net (0x00):

SEND[10]: `00 07 02 00 00 01 1E 00 00 04`

Send CMD_SET PAR_CYC_M_SYNC, master sync period of 15 seconds:

SEND[10]: `00 07 02 00 00 01 20 0F 00 04`

Send CMD_SET PAR_CYC_M_REST, master rest period of 60 seconds:

SEND[12]: `00 09 02 00 00 01 21 3C 00 00 00 04`

Send CMD_SET PAR_CYC_CTRL, state enabled (0x01), mode net (0x00):

SEND[10]: `00 07 02 00 00 01 1E 01 00 04`

2.10 Configuring and Using Status Messages

The interval between status messages is set with CMD_SET PAR_STAT_REP [3.7.10, p. 121]. The default interval is eight hours. A primary use of the status message is to assess and maintain network performance. Field nodes send status messages to the master node.

2.10.1 Example: Setting Field Node 0x0035 Status Message

Set field node 0x0035 status interval to 15 minutes.

Send CMD_IO PAR_STAT_REP, 15 minute status interval:
SEND[11]: `00 08 0B 88 35 00 0D 0E 00 00 04`
RECV[12]: `00 09 0B 88 01 35 00 0D 0E 00 00 04`

2.11 Reliable Reporting from Field Nodes

Field nodes can automatically generate messages related to events such as receiving a string of bytes when in the *data* mode, a change of state on a digital input, or reaching a count threshold on a pulse input. The maximum number of times a data or event message is sent by a field node and the timeout interval between transmissions can be configured for each node with CMD_SET PAR_RETRY [3.7.11, p. 123]. Event messages are repeated

Copyright 2006 - 2007

until the maximum is reached or an acknowledgement is received from the master node.

2.11.1 Example: Setting Reliable Reporting Field Node 0x0025

Set node 0x0025 for reliable reporting.

Send CMD_IO PAR_RETRY, 4 reliable reporting retries, timeout 18 seconds:

SEND[10]: `00 07 0B 88 25 00 0E 04 12 04`

RECV[11]: `00 08 0B 88 08 25 00 0E 04 12 04`

2.12 Application I/O

The DM2200 provides the following application hardware interfaces:

- Serial port for data communications
- 12-bit ADC for analog signal measurements
- Digital inputs
- Digital outputs

Referring to Figure 4, there are seven general purpose I/O ports available for application interfacing, GP1 - GP7, and three auxiliary digital ports, CFG0 - CFG2. Defaults for these ports are discussed in 2.12.1.

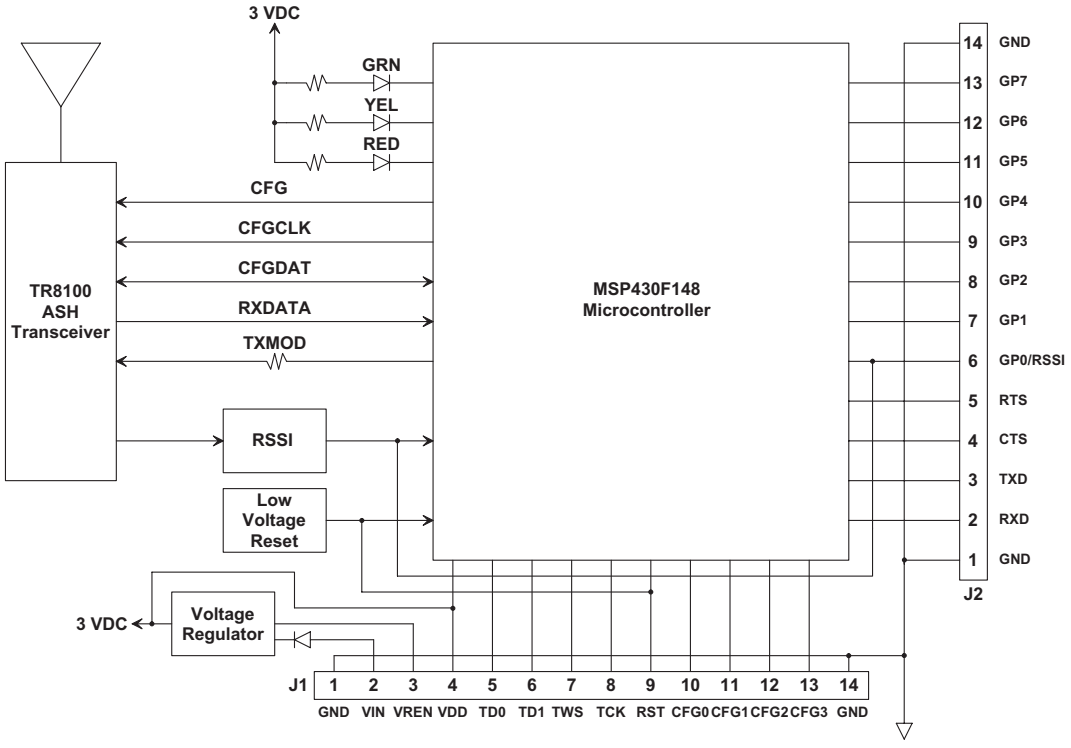


Figure 4 – DM2200-916VM Block Diagram

Port GP0 is dedicated to RSSI readings, and should always be defined as an analog input. Ports GP1 through GP7 can be individually configured as analog inputs, digital inputs or

Copyright 2006 - 2007

digital outputs. GP1 and GP2 have special capabilities when configured as digital inputs. GP4 and GP5 have special capabilities when configured as digital outputs. These capabilities are discussed below.

Port CFG3 is always defined as a logic input and is used to initiate software resets. CFG3 should be held at a logic high state for normal operation. If this input is brought low for one second, the three LEDs on the DM2200 will blink once. If CFG3 is brought high in less than four seconds after the LEDs blink, the DM2200 will initiate a “power on” software reset sequence. If this input is held low for seven seconds following the blink, the LEDs will go on solid and the DM2200 will erase all data store in its nonvolatile memory, restoring factory defaults. The LEDs will then blink seven more times and a “power on” reset will then be initiated. Auxiliary digital ports CFG0 - CFG2 have no dedicated functionality in VersaMESH and can be individually configured as digital inputs or digital outputs.

The DM2200 serial port can operate in two modes. The default mode is the command mode as discussed in section 2.4.1. An optional data mode is also provided for passively routing data without interaction, which is discussed below.

VersaMESH includes a rich set of commands to read, write, and configure the general purpose (GP) ports and auxiliary digital ports. These commands are discussed in Sections 3.8.1 through 3.8.15, and provide the following functionality:

- Configure a GP port as an analog input and make a 12-bit ADC measurement.
- Configure a GP port or auxiliary digital port as a digital input and read its state. The firmware also supports pulse counting on GP1, and event messaging on both GP1 and GP2.
- Configure a GP or auxiliary digital port as a digital output and set or read its state.

2.12.1 Default Settings for I/O

GP0	RSSI
GP1	Input with pulse counting and event messaging
GP2	Input with event messaging
GP3	Input
GP4	Output
GP5	Output
GP6	Analog input
GP7	Analog input

2.12.2 Data Mode Serial Port Operation

Any field node can be set to operate in data mode, to accept and output byte streams, and reliably forward them to the master node. CMD_SET PAR_CMD_MODE [3.6.32, p. 102] is used to select the serial port mode of operation, either command or data. When a field node is in the data mode it will send a string of bytes input to its serial port to the master node. The string is encapsulated in a packet and sent when terminated by an ASCII carriage

Copyright 2006 - 2007

return, the string reaches 64 bytes in length, or after a one second pause in the byte reception. After the data message is acknowledged by the master node, the serial port will accept a new string. Data can be sent for output at a field node with CMD_DAT [3.8.16, p. 179]. Note that when a node is placed in the data mode, it can only be switched back to the command mode through the RF link.

The data mode is useful in routing traffic through a DM2200 network to/from a data logger, printer, or other peripheral with fixed programming functionality.

2.12.3 Event Messages

Field nodes can automatically generate messages related to events such as a state change on a digital input, or reaching a count threshold on a pulse input. Event messages are addressed to the master node. The maximum number of times an event message is sent by a field node and the "time out" interval between transmissions can be configured for each node. Event messages are repeated until the maximum is reached or an acknowledgement is received back from the master node.

2.12.4 Analog-to-Digital Converter (ADC) Operation

The IO_ADC [3.8.3, p. 154] command is used to make an ADC measurement on GP1 through GP7. In addition to specifying the port number, the command includes several options. First, the ADC full-scale reading can be referenced to the 1.5 V or 2.5 V internal reference voltages, or to the 3 V power supply. Second, the sample-and-hold settling time for the ADC can be set from 1 to 64 ms.

2.12.5 Digital Input Operation

VersaMESH includes a rich set of commands related to digital inputs. Ports GP1 through GP7 and auxiliary digital ports CFG0 through CFG2 can be configured and read as digital inputs.

When GP1 is configured as a digital input, it can optionally count pulses used in applications such as utility sub-metering. GP1 is able to count up to 10 pulses/second, with the option to count low pulses (high-to-low edges) or high pulses (low-to-high edges). Pulse counts are stored in a 24-bit register that can be preset, read and reset. Note that GP1 can count pulses during a power management Light Sleep cycle. The pulse count can be stored in nonvolatile memory periodically to protect against loss due to a power outage. This period is a configurable parameter, with 15 minutes a typical value.

The pulse counter is supported by two 24-bit companion registers. The first companion register provides a comparator function. When the comparator register is loaded with a value and enabled, an event message can be triggered when the pulse count matches the value in the comparator. At this point the value of the comparator can be optionally subtracted from the pulse counter to reset it.

The second companion register is an offset register that can be used with either the pulse counter or the comparator register. The offset register supports two modes of operation. In the comparator offset mode, when the pulse-count register value reaches the comparator

Copyright 2006 - 2007

value, an event message is triggered, and the value in the comparator is increased by the value in the offset register. For example, if the value in the comparator register is initially 1000 and the value in the offset register is 100, an event message will be sent when the value in the pulse-count register reaches 1000, 1100, 1200, etc.

In the counter offset mode, when the pulse count reaches the comparator value, an event message is triggered, and the value in the pulse counter is decreased by the value in the offset register. For example, if the value in the comparator register is initially 1000 and the value in the offset register is 100, an event message will be sent when the value in the pulse counter reaches 1000. At this point, the value in the pulse counter is "backed down" 100 counts to a value of 900. When the value in the pulse counter again reaches 1000, the next event message with be sent and the value in the pulse counter will again be reset to 900. The pulse count register and comparator register exhibit 24-bit modulo behavior and will gracefully "wrap around" on an overflow or underflow.

When GP2 is configured as a digital input it can generate an event messages as a configuration option.

2.12.6 Digital Output Operation

Ports GP1 through GP7 and auxiliary digital ports CFG0 through CFG2 can be configured as digital outputs. Digital outputs can be set to a high or low logic state, and the state can be read. If GP4 and GP5 are configured as digital outputs, these signals can be used to drive the two Form-C relays on the IM2200 interface board.

2.12.7 Application I/O Examples

The following examples use the connections shown in Figure 5.

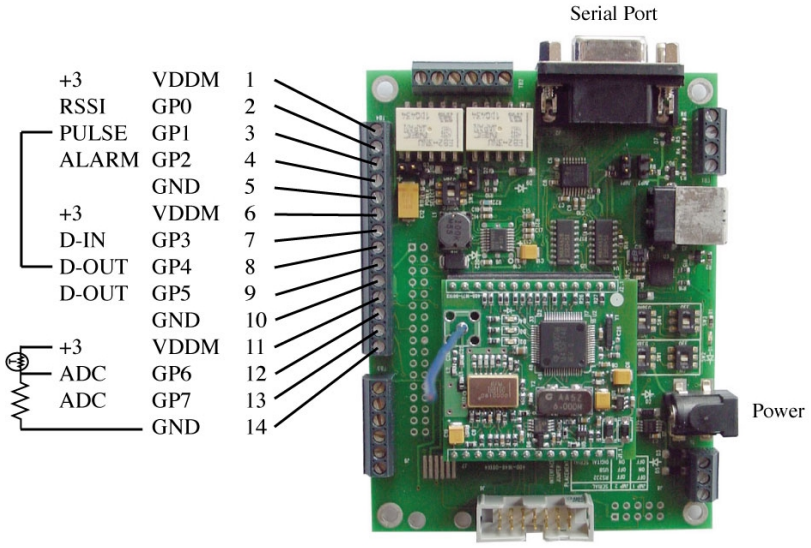


Figure 5 - I/O Hookups Used in Examples

Copyright 2006 - 2007

2.12.7.1 Example: Pulse Counter

Connect a jumper wire between GP4 and GP1 on node 0x0025.
Initialize the counter to 10 (0x0A) and count on the pulse rising edge.

Send CMD_IO IO_CNT_START:
SEND[11]: 00 08 0B 88 25 00 13 0A 00 00 04
RECV[12]: 00 09 0B 88 01 25 00 13 0A 00 00 04

Send 10 pulses by toggling GP4 with the following.

Set GP4 output bit to 1:
Send CMD_IO IO_WRITE
SEND[11]: 00 08 0B 8A 25 00 01 04 01 FF 04
RECV[12]: 00 09 0B 8A 01 25 00 01 04 01 FF 04

Set GP4 output bit to 0:
Send CMD_IO IO_WRITE
SEND[11]: 00 08 0B 8A 25 00 01 04 00 FF 04
RECV[12]: 00 09 0B 8A 01 25 00 01 04 00 FF 04

Repeat above 2 commands 10 times to generate 10 pulses.

Read the counter.
Send CMD_IO IO_CNT_GET to read the counter:
SEND[11]: 00 08 0B 88 25 00 07 00 00 00 04
RECV[12]: 00 09 0B 88 01 25 00 17 0B 00 00 04

2.12.7.2 Example: Alarm GP2

Demonstrate an alarm using the GP2 interrupt.

Send CMD_IO IO_NOT_START to alarm on rising edge (0x01) using GP2 on node 0x0035:
SEND[11]: 00 08 0B 88 35 00 05 01 00 00 04
RECV[12]: 00 09 0B 88 01 35 00 05 01 00 00 04

Since the pin is high, the alarm message is sent immediately:
RECV[12]: 00 09 FF 13 35 00 37 0B 00 00 01 04

2.12.7.3 Example: Reading Analog I/O

Read temperature using a thermistor circuit connected to node 0x0055.

Send CMD_IO IO_ADC to read the analog input:
SEND[11]: 00 08 0B 88 55 00 02 26 0A FF 04
RECV[12]: 00 09 0B 88 01 55 00 02 26 06 0E 04

Copyright 2006 - 2007

2.12.7.4 Example: Set Pins As Output High/Low

Digital output (operate relay on node 0x0035).
Connect jmp4 to connect GP5 to relay.

Send CMD_IO IO_WRITE to operate GP5 relay:
SEND[11]: `00 08 0B 88 35 00 01 05 01 FF 04`
RECV[12]: `00 09 0B 88 01 35 00 01 05 01 FF 04`

Send CMD_IO IO_WRITE to release GP5 relay:
SEND[11]: `00 08 0B 88 35 00 01 05 00 FF 04`
RECV[12]: `00 09 0B 88 01 35 00 01 05 00 FF 04`

2.12.8 Pulse Monitoring Examples

2.12.8.1 Add CREG to comparator

Send CMD_IO IO_CMP_START to set the comparator to 4 and start counting:
SEND[11]: `00 08 0B 88 25 00 04 04 00 00 04`
RECV[12]: `00 09 0B 88 01 25 00 04 04 00 00 04`

Send CMD_IO IO_CNT_START to start counting from 1 on rising edge of each pulse:
SEND[11]: `00 08 0B 88 25 00 13 01 00 00 04`
RECV[12]: `00 09 0B 88 01 25 00 13 01 00 00 04`

Send CMD_IO IO_CREG_SET to set CREG to 2:
SEND[11]: `00 08 0B 88 25 00 0B 02 00 00 04`
RECV[12]: `00 09 0B 88 01 25 00 0B 02 00 00 04`

Send CMD_IO IO_FLAGS_SET to set the backup timer to 15 minutes, adding CREG to comparator when triggered:
SEND[11]: `00 08 0B 88 25 00 0D 01 01 FF 04`
RECV[12]: `00 09 0B 88 01 25 00 0D 01 01 FF 04`

Send CMD_IO IO_CNT_GET to read the counter:
SEND[11]: `00 08 0B 88 25 00 07 00 00 00 04`
RECV[12]: `00 09 0B 88 01 25 00 17 01 00 00 04`

Receive the following message following 3 pulse rising edges:
RECV[12]: `00 09 FF 13 25 00 F0 04 00 00 01 04`

Copyright 2006 - 2007

2.12.8.2 Subtract CREG from the counter

Send CMD_IO IO_CMP_START to set the comparator to 12 and start comparing:

SEND[11]: `00 08 0B 88 25 00 04 0C 00 00 04`
RECV[12]: `00 09 0B 88 01 25 00 04 0C 00 00 04`

Send CMD_IO IO_CNT_START to start counting from 8 on rising edges:

SEND[11]: `00 08 0B 88 25 00 13 08 00 00 04`
RECV[12]: `00 09 0B 88 01 25 00 13 08 00 00 04`

Send CMD_IO IO_CREG_SET to set CREG to 2:

SEND[11]: `00 08 0B 88 25 00 0B 02 00 00 04`
RECV[12]: `00 09 0B 88 01 25 00 0B 02 00 00 04`

Send CMD_IO IO_FLAGS_SET to set the backup timer to 15 minutes, subtracting CREG from counter:

SEND[11]: `00 08 0B 88 25 00 0D 01 02 FF 04`
RECV[12]: `00 09 0B 88 01 25 00 0D 01 02 FF 04`

Send CMD_IO IO_CNT_GET to read the counter:

SEND[11]: `00 08 0B 88 25 00 07 00 00 00 04`
RECV[12]: `00 09 0B 88 01 25 00 17 08 00 00 04`

Receive the following message following 4 pulse rising edges:

RECV[12]: `00 09 FF 13 25 00 F0 0C 00 00 01 04`

2.12.8.3 Subtract comparator from the counter

Send CMD_IO IO_CMP_START to set the comparator to 10 and start comparing:

SEND[11]: `00 08 0B 88 25 00 04 0A 00 00 04`
RECV[12]: `00 09 0B 88 01 25 00 04 0A 00 00 04`

Send CMD_IO IO_CNT_START to start counting from 4 on rising edges:

SEND[11]: `00 08 0B 88 25 00 13 04 00 00 04`
RECV[12]: `00 09 0B 88 01 25 00 13 04 00 00 04`

Send CMD_IO IO_FLAGS_SET to set the backup timer to 15 minutes; Subtracting the comparator from the counter at interrupt:

SEND[11]: `00 08 0B 88 25 00 0D 01 03 FF 04`
RECV[12]: `00 09 0B 88 01 25 00 0D 01 03 FF 04`

Copyright 2006 - 2007

Send CMD_IO IO_CNT_GET to read the counter:
SEND[11]: `00 08 0B 88 25 00 07 00 00 00 04`
RECV[12]: `00 09 0B 88 01 25 00 17 04 00 00 04`

Receive the following message after 6 pulse rising edges:
RECV[12]: `00 09 FF 13 25 00 F0 0A 00 00 01 04`

2.13 Encryption

VersaMESH uses TEA (Tiny Encryption Algorithm) encryption methodology to provide a balance between computational efficiency and security. Two encryption modes and three selectable 128-bit encryption keys are preloaded in firmware.

- Mode 0 decrypts a message using the key index referenced in the header of the packet.

- Mode 1 decrypts a message using the key specified by an earlier configuration command.

The keys used within a given network must be the same for the sender and recipient(s) of any encrypted message.

Two parameters control the encryption:

- PAR_ENCR_M: Encryption mode. If "*free*" (0), the node uses the key indicated in an incoming message (or doesn't decrypt). If "*strict*" (1), the node decrypts incoming messages with the key indicated by its PAR_ENCR_K. Messages encrypted with other keys are dropped (but forwarded if hey are broadcast). PAR_ENCR_M defaults to "*free*" (0).

- PAR_ENCR_K: Encryption key # (0..3). Zero means "*no encryption*". Packets originating on a node with PAR_ENCR_K set are encrypted with this key *number*. PAR_ENCR_K defaults to "*no encryption*" (0).

- PAR_ENCR_D: Encryption data. Combines the mode and key, forming a shortcut convenient in some operations.

Tracing and a part of the binding process are never encrypted:

- Tracing (*msg_trace, msg_traceAck,* etc.) modifies packets on every hop, and the overhead of encrypting and decrypting them is not justified for this diagnostic oriented functionality.

- Binding: The messages originating and terminating on an unbound node are not encrypted. The newly bond node inherits PAR_ENCR_D from the binding node.

Copyright 2006 - 2007

3 Message Reference

3.1 VersaMESH Message Formats:

VersaMESH commands, responses and data/event messages follow a general byte structure. All messages start with a 0x00 *BOT* (beginning of text) byte, and end with a 0x04 *EOT* (end of text) byte.

The second byte in all messages, *op_len*, holds the number of bytes *between* the *op_len* byte and the EOT byte.

The third byte in the command and response messages is the *CMD* (command type) byte, and the third byte in the data/event messages is *msg_output* (data/event) byte.

The fourth byte in the command and response messages is an *op_ref*, or a message ID byte, which allows a command and its response to be matched up. *Consecutive messages sent from the master node must have different op_ref values for correct network operation.* The fourth byte in the data/event message format indicates the specific type of message.

The fifth byte in response messages is a an *op_rc* byte to indicate the error status of the response. *Command messages do not have this byte.*

The fifth and sixth bytes in a command and the sixth and seventh bytes in the response hold the *node_id*, or 16-bit address of the target node.

The number and function of the *payload or message specific* bytes in each message is discussed in Part 2. VersaMESH messages place multi-byte data, such as the *node_id*, in *little endian* format, where the lowest order byte is to the left. For example, the 32-bit number 0x12345678 would appear as 78 56 34 12 within a VersaMESH message.

Command message format:

BOT	op_len	**CMD**	op_ref	node_id	**payload**	EOT
0x00	0x05	**0x01**	X(1B)	X(2B)	X(1B)	0x04

Response message format:

BOT	op_len	**CMD**	op_ref	op_rc	node_id	**payload**	EOT
0x00	**X(nB+4)**	**0x01**	X(1B)	X(1B)	X(2B)	**X(nB)**	0x04

Data/event message format:

BOT	op_len	**msg_output**	msg_type	**msg_specific**	EOT
0x00	X(xB)	**0xFF**	**X(1B)**	**X(xB)**	0x04

Copyright 2006 - 2007

3.2 Example Network

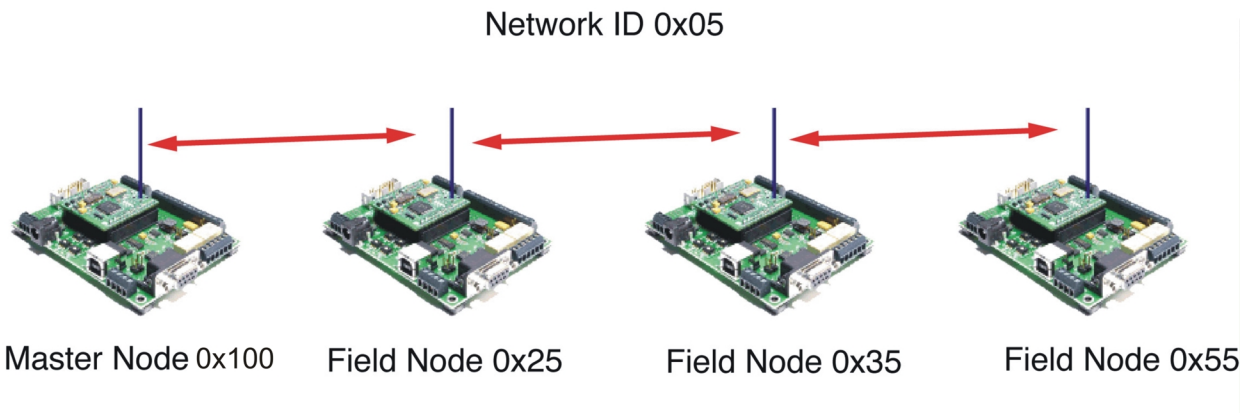


Figure 6 - Example Network Used in Messages Responses

Copyright 2006 - 2007

3.3 General Message Layout

The following is an overview of the command gets (CMD_GET), command sets (CMD_SET) and command IO (CMD_IO). The bulk of command messages fall into these three types.

3.3.1 CMD_GET Messages

Layout:

BOT	op_len	**CMD_GET**	op_ref	node_id	**payload**	EOT
0x00	0x05	**0x01**	X(1B)	X(2B)	X(1B)	0x04

Payload:

parameter

Synopsis:

CMD_GET (0x01): Get various parameters and attributes from the addressed node.

Factory Settings or Defaults:

Parameter specific

Valid Parameter Range:

Parameter specific

Power-up Value:

Parameter specific

Results Returned:

BOT	op_len	**CMD_GET**	op_ref	op_rc	node_id	**payload**	EOT
0x00	**X(nB+4)**	**0x01**	X(1B)	X(1B)	X(2B)	**X(nB)**	0x04

Payload:

Parameter(1B)	B1	B2	…	…

Except for CMD_GET ATTR_NHOOD [3.6.24, p. 89] command, all CMD_GET type commands return a response, even if the most significant bit in the op_ref byte is not set.

In response to a CMD_GET ATTR_NHOOD [3.6.24, p. 89] command, individual messages are generated by the neighbors around the target node. A response to this command is not sent back unless the most significant bit the *op_ref* byte of the command is set.

BOT	op_len	**msg_output**	msg_type	**msg_specific**	EOT
0x00	**X(nB+2)**	**0xFF**	X(1B)	**X(nB)**	0x04

Payload:

B1	B2	B3	…	…	…

Examples:

On the operations level (detailed below).

Copyright 2006 - 2007

Error Codes:

RC_OKRET	(0x01)	Success; data is returned to the sender
RC_ENET	(0x03)	Error: not network ID (unbound node)
RC_EPAR	(0x06)	Error: bad parameter
RC_ELEN	(0x08)	Error: bad input (payload) length

Note:

Other error codes are parameter specific.

Parameters:

PAR_LH	(0x01)	Node ID
ATTR_ESN	(0x02)	Equipment serial number
ATTR_MID	(0x03)	Master node ID
PAR_NID	(0x04)	Network ID
PAR_TARP_L	(0x05)	TARP level parameter
PAR_TARP_S	(0x06)	TARP slack parameter
PAR_TARP_R	(0x07)	TARP route recovery
PAR_TARP_F	(0x08)	TARP forwarding
PAR_BEAC	(0x09)	Beacon interval
PAR_CON	(0x0A)	Connectivity parameters
PAR_RF	(0x0B)	Status for TX and RX; TX ON/OFF, RX ON/OFF
PAR_STAT_REP	(0x0D)	Status report parameters (interval, time to next)
PAR_RETRY	(0x0E)	Number of tries, timeout
PAR_AUTOACK	(0x0F)	Auto-acknowledgment (ON/OFF)
ATTR_UPTIME	(0x10)	Uptime (in seconds)
ATTR_RSSI	(0x12)	RSSI of the last message sent by master
ATTR_MSG_COUNT_T	(0x13)	Message counters (tarp_rcv, _snd, _fwd)
ATTR_MEM1	(0x15)	Memory statistics (free memory, min free memory, min free stack)
ATTR_MEM2	(0x16)	Memory statistics (max free memory chunk, free chunks, static size)
PAR_ENCR_M	(0x17)	Encryption mode
PAR_ENCR_K	(0x18)	Encryption key (number)
PAR_ENCR_D	(0x19)	Encryption data (mode, key)
ATTR_SYSVER	(0x1A)	System version (0xMMmm – Major minor)
ATTR_NHOOD	(0x1B)	Neighbors around node report back
PAR_BINDER	(0x1C)	Bind enable flag
PAR_UART	(0x1D)	UART baud rate
PAR_CYC_CTRL	(0x1E)	Power management controls (state, mode)
PAR_CYC_SLEEP	(0x1F)	Power management sleep period in seconds
PAR_CYC_M_SYNC	(0x20)	Master sync period (or special modes)
PAR_CYC_M_REST	(0x21)	Master rest period in seconds
ATTR_CYC_LEFT	(0x22)	Seconds left in current state; state; mode
PAR_CMD_MODE	(0x23)	Serial port command mode (1) or data mode (0)

Notes:

Configurable data items are called *parameters*; read only data items are called *attributes*.

Specific functionality is described at the parameter and attribute level on the following pages.

Copyright 2006 - 2007

3.3.2 CMD_SET Messages

Layout:

BOT	op_len	**CMD_SET**	op_ref	node_id	**payload**	EOT
0x00	**X(nB+4)**	**0x02**	X(1B)	X (2B)	**X(nB)**	0x04

Payload:

Parameter(1B)	B1	B2	…	…	…	…

Note: maximum payload size is 7 bytes

Synopsis:

Set various parameters in the addressed node.

Factory Settings or Defaults:

On the *parameters* level

Valid Parameter Range:

On the *parameters* level

Power-up Value:

On the *parameters* level

Results Returned:

In general, responses are returned to the host when it makes sense, e.g., no implicit confirmation is expected via a message output, the command failed, or the response carries retrieved data. However, the responses are forced back if the operation's reference code has the most significant bit set, e.g., a CMD_SET to a remote node with ref. code 0x0A will not return if successful (the result may be verified with a corresponding CMD_GET), but the op_ref code 0xAA will force the result back, even if it is marked with RC_OK.

Notes:

In the following CMD_SET documentation, the responses are forced back (the most significant bit in the operation's reference code is set) to help understand the format and structure of the commands. Therefore, results are always returned, in usual format, with added op_rc field:

BOT	op_len	**CMD_SET**	op_ref	op_rc	node_id	**payload**	EOT
0x00	**X(nB+5)**	**0x02**	X(1B)	X(1B)	X(2B)	**X(nB)**	0x04

Payload:

Parameter(1B)	B1	B2	…	…	…	…

Examples:

On the operations level (detailed below).

Copyright 2006 - 2007

Error Codes:

RC_OK	(0x00)	Success
RC_OKRET	(0x01)	Success with response returned to the issuer
RC_ENET	(0x03)	Error: not network ID (unbound node)
RC_EPAR	(0x06)	Error: bad parameter
RC_ELEN	(0x08)	Error: bad input (payload) length

Note:

Other error codes are described on the operation level

Parameters:

PAR_LH	(0x01)	Set node ID
PAR_NID	(0x04)	Set network ID
PAR_TARP_L	(0x05)	Set TARP level parameter
PAR_TARP_S	(0x06)	Set TARP slack parameter
PAR_TARP_R	(0x07)	Set TARP route recovery
PAR_TARP_F	(0x08)	Set TARP forwarding
PAR_BEAC	(0x09)	Set beacon interval
PAR_CON	(0x0A)	Set connectivity parameters
PAR_RF	(0x0B)	Set radio status (TX, RX)
PAR_STAT_REP	(0x0D)	Set status reporting interval, etc.
PAR_RETRY	(0x0E)	Set retry parameters (number of retries, timeout)
PAR_AUTOACK	(0x0F)	Set auto-acknowledgments
PAR_ENCR_M	(0x17)	Set encryption mode
PAR_ENCR_K	(0x18)	Set (select) encryption key
PAR_ENCR_D	(0x19)	Set encryption data
PAR_BINDER	(0x1C)	Set node bind enable flag
PAR_UART	(0x1D)	Set UART baud rate
PAR_CYC_CTRL	(0x1E)	Set power management controls (state, mode
PAR_CYC_SLEEP	(0x1F)	Set power management sleep period
PAR_CYC_M_SYNC	(0x20)	Set power management sync period (or special modes)
PAR_CYC_M_REST	(0x21)	Set power management rest period
PAR_CMD_MODE	(0x23)	Set command mode (1) or data mode (0)

Note:

Most functionality is described on the operation level.

Copyright 2006 - 2007

3.3.3 CMD_IO Messages

Layout:

BOT	op_len	**CMD_IO**	op_ref	node_id	**payload**	EOT
0x00	**0x08**	**0x0B**	X(1B)	X(2B)	**X(4B)**	0x04

Payload:

operation	B1	B2	B3

Synopsis:

CMD_IO (0x0B): This command configures and controls I/O operations:

I/O pin configuration, reads and writes
ADC configuration and reads
Interrupt-driven I/O operation

Specifics:

Pin mapping and configuration:

GP1 - 7 are mapped as pins 1 through 7 and CFG 0 - 2 are mapped as pins 8 through 10.
GP1 - may be configured as an interrupt driven 24-bit counter.
GP2 - may be configured as an interrupt driven input with state change notification.

Factory Settings or Defaults:

GP1, GP2 and GP3 are digital inputs, GP4 and GP5 are digital outputs, GP6 and GP7 are analog inputs. GP1 is supported by special commands for digital pulse counting and related event messages. GP2 is supported with special commands related to event messages.

Valid Parameter Range:

Compound ranges

Power-up Value:

Pin configurations are preserved and restored at power up.

Results Returned:

I/O operation results are returned without requiring the most significant bit of op_ref to be set. Results are described on the *operations* level.

Examples:

On the operations level (detailed below).

Copyright 2006 - 2007

Error Codes:

RC_OKRET	(0x01)	Success
RC_EPAR	(0x06)	bad operation code
RC_ELEN	(0x08)	bad op_len (bad payload length)

Note:

Other error codes are described on the operation level

Operations:

IO_READ	(0x00)	Read a pin(s), including bulk read variant
IO_WRITE	(0x01)	Configure a pin, write a pin, configure and write a pin
IO_ADC	(0x02)	Measure analog input (12 bits)
IO_CNT_START	(0xX3)	**S**tart interrupt driven counter on GP1
IO_CMP_START	(0x04)	Start comparator
IO_NOT_START	(0x05)	**S**tart interrupt driven state change notification on GP2
IO_STOP	(0x06)	Stop (suspend) any of the three pulse counting operations
IO_CNT_GET	(0x07)	Get the counter's current value
IO_CMP_GET	(0x08)	Get the comparator's current value
IO_STATE	(0x09)	Get the current state of pulse monitoring operations
IO_CREG_GET	(0x0A)	Get comparator's offset register
IO_CREG_SET	(0x0B)	**S**et comparator's offset register
IO_FLAGS_GET	(0x0C)	Get pulse monitoring control flags operations
IO_FLAGS_SET	(0x0D)	**S**et pulse monitoring control flags

Note:

Most functionality is described on the operation level.

Copyright 2006 - 2007

3.4 Message Specific Formats

The following default messages are predefined and referenced by other messages. These predefined specific formats include:

msg_bindReq	bind request message		
msg_io	I/O event message		
msg_data	data message		
msg_br	status message		
msg_traceAck	bidirectional trace acknowledgement		
msg_traceFAck	forward trace acknowledgement		
msg_traceBAck	backward trace acknowledgement		
msg_nhAck	neighborhood ACK message		

***General Message Layout*:**

BOT	op_len	**msg_output**	msg_type	**msg_specific**	EOT
0x00	X(xB)	**0xFF**	**X(1B)**	**X(xB)**	0x04

Payload:

B1	B2	B3	…	…	…	…	…

3.4.1 msg_bindReq:

This is the bind request message format:

Layout:

BOT	op_len	**msg_output**	msg_type	**msg_bindReq**	EOT
0x00	0x0A	**0xFF**	**0x06**	**X(8B)**	0x04

Message Data:

B1(newcomer's ESN)	B2(newcomer's ESN)	B3(newcomer's ESN)

…

B4(newcomer's ESN)	B5(newcomer's temp node_id)

…

B6(newcomer's temp node_id)	B7(binding node_id)

…

B8(binding node_id)

msg_bindReq:

byte B1 - B4: 2 words, newcomer's ESN
byte B5 - B6: word, newcomer's temporary node ID
byte B5 - B6: word, number of registered ESNs
byte B7 - B8: word, binding node ID

Copyright 2006 - 2007

3.4.2 msg_io:

This is the I/O event message format:

Layout:

BOT	op_len	**msg_output**	msg_type	**msg_io**	EOT
0x00	**0x09**	**0xFF**	**0x13**	**X(7B)**	0x04

Message IO:

B1(node_id)	B2(node_id)	B3(io state)	B4(counter)

...

B5(counter)	B6(counter)	B7 (autoACK)

The node ID (B1B2) is 0x0100

I/O state (B3): 0x77, or bit pattern 1110111

For GP2

- b0 set: edge is rising
- b1 set: notification is on
- b2 set: an event is pending
- b3 clear: not used

For GP1

- b4 set: counter edge is rising
- b5 set: counter is active
- b6 set: comparator is active
- b7 clear: a comparator event is not pending

The counter value (B4B5B6) is 0x00000B

AutoACK is ON (B7 is 0x01).

Example:

```
00 09 FF 13 00 01 77 0B 00 00 01 04
```


Copyright 2006 - 2007

3.4.3 msg_data:

This is the data message format:

Layout:

BOT	op_len	**msg_output**	msg_type	**msg_data**	EOT
0x00	**X(nB+2B)**	**0xFF**	**0x15**	**X(nB)**	0x04

Message Data:

B1(sr_node_id)	B2(sr_node_id)	B3(data_ref)	B4(data)	B5(data)

…

…	…	Bn(data)	Bn+1(autoACK)

msg_data:

byte B1 - B2: source node ID
byte B3: data reference
byte B4 - Bn: string of bytes (data)
byte Bn+1: Auto-acknowledgment (ON/OFF)

Example:

```
00 0B FF 15 25 00 B5 66 77 68 01 02 00 04
```

3.4.4 msg_br:

This is the status message format:

Layout:

BOT	op_len	**msg_output**	msg_type	**msg_br**	EOT
0x00	0x0B	**0xFF**	**0x08**	**X(9B)**	0x04

Message Data:

B1(br_node_id)	B2(br_node_id)	B3(connectivity)	B4(connectivity)

…

B5(# of ESNs)	B6(# of ESNs)	B7(# of statuses)

…

B8(# of statuses)	B9(autoACK)

msg_br:

byte B1 - B2: node ID
byte B3 - B4: word, connectivity: warn (4 bits), bad (4 bits), missed (byte 3)
byte B5 - B6: not used on DM2200
byte B7 - B8: not used on the DM2200
byte B9: autoACK status, if 1, ACKs are sent automatically

Copyright 2006 - 2007

3.4.5 msg_traceAck:

This is the bidirectional trace acknowledgement format:

Layout:

BOT	op_len	**msg_output**	msg_type	**msg_traceAck**	EOT
0x00	**X(xB)**	**0xFF**	**0x03**	**X(xB)**	0x04

Message Data:

B1(# of forward hops)	B2(# of backward hops)	B3..B4...Bn(node ids)*

The source/termination node ID of the tracing loop is not shown in the output. Only node IDs in requested direction are collected and reported.

msg_traceAck :

byte B1: number of forward hops
byte B2: number of backward hops
byte B3 - Bx: node ID's

3.4.6 msg_traceFAck:

This is the forward trace acknowledgement format:

Layout:

BOT	op_len	**msg_output**	msg_type	**msg_traceFAck**	EOT
0x00	**X(nB+2)**	**0xFF**	**0x10**	**X(nB)**	0x04

Message Data:

B1(# of forward hops)	B2(# of backward hops)	B3..B4...(node ids)

msg_traceFAck:

byte B1: number of forward hops
byte B2: number of backward hops
byte B3 - Bn: node IDs

Copyright 2006 - 2007

3.4.7 msg_traceBAck:

This is the backward trace acknowledgement format:

Layout:

BOT	op_len	**msg_output**	msg_type	**msg_traceBAck**	EOT
0x00	**X(nB+2)**	**0xFF**	**0x12**	**X(nB)**	0x04

Message Data:

B1(# of forward hops)	B2(# of backward hops)	B3..B4...(node ids)

msg_traceBAck:

byte B1: number of forward hops
byte B2: number of backward hops
byte B3 - Bn: node IDs

3.4.8 msg_nhACK:

This is the neighborhood ACK message format:

Layout:

BOT	op_len	**msg_output**	msg_type	**msg_nhAck**	EOT
0x00	0x0B	**0xFF**	**0x0E**	**X(9B)**	0x04

Message Data:

B1(pivotal node_id)	B2(pivotal node_id)	B3(ESN)	B4(ESN)

...

B5(ESN)	B6(ESN)	B7(node_id)	B8(node_id)	B9(# of hops)

msg_nhAck:

byte B1 - B2: pivotal node ID
byte B3 - B6: ESNs, 4 bytes each
byte B7 - B8: node ID
byte B9: number of hops

Copyright 2006 - 2007

3.5 Global Messages

3.5.1 CMD_LOCAL

Layout:

BOT	op_len	**CMD_LOCAL**	op_ref	**node_id**	EOT
0x00	0x04	**0x0A**	X(1B)	X(2B)	0x04

Synopsis:

CMD_LOCAL (0x0A): This serial port command returns local data (node ID, ESN, master ID).

Specifics:

Local serial port command only.

Factory Settings or Defaults:

N/A

Valid Parameter Range:

N/A

Power-up Value:

N/A

Results Returned:

Results are always returned, in usual format, with added op_rc field:

BOT	op_len	**CMD_LOCAL**	op_ref	op_rc	**payload**	EOT
0x00	0x0B	**0x0A**	X(1B)	X(1B)	**X(8B)**	0x04

Payload:

B1(net_id)	B2(net_id)	B3(ESN)	B4(ESN)	B5(ESN)	B6(ESN)

…

B7(master_id)	B8(master_id)

Error Codes:

RC_OK (**0x00**)**:** success
RC_ELEN (**0x08**)**:** error, bad length

Notes:

Node ID field is arbitrary for this command.

Copyright 2006 - 2007

Examples:

Return local data:

```
00 04 0A DD 00 00 04
```

B1(node_id)	B2(node_id)
0x00	0x00

Returns:

```
00 0B 0A DD 00 00 01 01 00 CA BA 00 01 04
```

B1(node_id)	B2(node_id)	B3(ESN)	B4(ESN)	B5(ESN)	B6(ESN)
0x00	0x01	0x01	0x00	0xCA	0xBA

…

B7(master_id)	B8(master_id)
0x00	0x01

Node ID is 0x0100.
ESN is 0xBACA0001.
Master ID is 0x0100.

Copyright 2006 - 2007

3.5.2 CMD_MASTER

Layout:

BOT	op_len	**CMD_MASTER**	op_ref	**node_id**	EOT
0x00	0x04	**0x03**	X(1B)	X(2B)	0x04

Synopsis:

CMD_MASTER (0x03): This command enables master node functionality.

Specifics:

This command sets the network master node. If this command is sent to a node's serial port with the address of *0x0000* (must use 0x0000 not the node's ID), the node will set its master ID to *its own node ID*, and will then forward its master designation to all other nodes in the network on its master beacon. Every bound node that receives the beacon will then update its master ID to the master node ID on the beacon (see 3.7.7 on setting the beacon interval).

If this command is sent addressed to a *single* node (node ID not 0x0000), the targeted node will change its master ID to the sending node's ID.

Factory Settings or Defaults:

N/A

Valid Parameter Range:

N/A

Power-up Value:

Although all pertinent data is restored from nonvolatile memory on power up, including the beacon interval, ***the first master beacon is not sent until a CMD_MASTER command is received by the node***.

Results Returned:

In usual format, with added op_rc field:

BOT	op_len	**CMD_MASTER**	op_ref	op_rc	**node_id**	EOT
0x00	0x05	**0x03**	X(1B)	X(1B)	X(2B)	0x04

Error Codes:

RC_OK **(0x00)**: successful (RC_OK returned whether or not the MSB of op_ref is set)
RC_ENET **(0x03)**: error, no network (unbound node)
RC_EFORK **(0x0A):** error, no memory to start the beacon.

Notes:

CMD_MASTER has no payload.

Examples:

Set a node to master mode:
`00 04 03 0D **00 00** 04`

Returns:
`00 05 03 0D 00 **00 00** 04`

The node is set to master mode.

Copyright 2006 - 2007

3.5.3 CMD_BIND

Layout:

BOT	op_len	**CMD_BIND**	op_ref	node_id	**payload**	EOT
0x00	0x0B	**0x05**	X(1B)	X(2B)	X(7B)	0x04

Payload:

B1(ESN)	B2(ESN)	B3(ESN)	B4(ESN)	B5(node_id)

…

B6(node_id)	B7(range)

Synopsis:

CMD_BIND (0x05): The command performs binding directly or requests binding be done by another node near the node to be bound.

Specifics:

The target node (node ID in the command) will broadcast a bind message within a specified number of hops, carrying pertinent binding data: ESN, new node ID, network ID, master ID, connectivity and encryption data. This command is usually issued after a bind request message is received from an unbound node.

Factory Settings or Defaults:

N/A

Valid Parameter Range:

N/A

Power-up Value:

N/A

Results Returned:

If forced by setting the most significant bit in op_ref, with added op_rc field:

BOT	op_len	**CMD_BIND**	op_ref	op_rc	node_id	**payload**	EOT
0x00	0x0C	**0x05**	X(1B)	X(1B)	X(2B)	X(7B)	0x04

Payload:

B1(ESN)	B2(ESN)	B3(ESN)	B4(ESN)	B5(node_id)	B6(node_id)	B7(range)

with the payload as in the original command.

Error Codes:

RC_OK **(0x00)**: successful
RC_ENET **(0x03)**: error, no network ID (unbound node)

Notes:

None

Copyright 2006 - 2007

Examples:

Bind request:

```
00 0A FF 06 01 00 CA BA 01 00 00 01 04
```

Returns (msg_bindReq):

```
00 0A FF 06 01 00 CA BA 01 00 00 01 04
```

B1(newcomer's ESN)	B2(newcomer's ESN)	B3(newcomer's ESN)
0x01	0x00	0xCA

…

B4(newcomer's ESN)	B5(newcomer's temp node_id)
0xBA	0x01

…

B6(newcomer's temp node_id)	B7(binding node_id)
0x00	0x00

…

B8(binding node_id)
0x01

CMD_BIND response:

```
00 0B 05 DD 00 01 01 00 CA BA 25 00 01 04
```

B1(ESN)	B2(ESN)	B3(ESN)	B4(ESN)	B5(node_id)	B6(node_id)	B7(range)
0x01	0x00	0xCA	0xBA	0x25	0x00	0x01

Returns:

```
00 0C 05 DD 00 00 01 01 00 CA BA 25 00 01 04
```

B1(ESN)	B2(ESN)	B3(ESN)	B4(ESN)	B5(node_id)	B6(node_id)	B7(range)
0x01	0x00	0xCA	0xBA	0x25	0x00	0x01

A node with an ESN of 0xBACA0001 is to be bound by node 0x0100, receiving a new node ID 0x0025. The bind request's range is one hop (the new node must be directly in the neighborhood of 0x0100). After the successful bind, a status message is sent from the newly bound node, and the message output appears:

msg_br

```
00 07 FF 08 25 00 00 34 01 04
```

B1(br_id)	B2(br_id)	B3(connectivity)	B4(connectivity)	B5(autoACK)
0x25	0x00	0x00	0x34	0x01

Copyright 2006 - 2007

3.5.4 CMD_SACK

Layout:

BOT	op_len	**CMD_SACK**	op_ref	node_id	**payload**	EOT
0x00	0x08	**0x06**	X(1B)	X(2B)	X(4B)	0x04

Payload:

B1(ESN)	B2(ESN)	B3(ESN)	B4(ESN)

Synopsis:

CMD_SACK (0x06): This command sends a message acknowledgement.

Specifics:

This command acknowledges a message sent by the target node. The ESN must be the last ESN reported or 0x00000000 for status message acknowledgement. Note that this functionality is automatic if PAR_AUTOACK is ON.

Factory Settings or Defaults:

N/A

Valid Parameter Range:

N/A

Power-up Value:

N/A

Results Returned:

A result is returned if forced by setting the most significant bit in op_ref, with added op_rc field:

BOT	op_len	**CMD_SACK**	op_ref	op_rc	node_id	**payload**	EOT
0x00	0x09	**0x06**	X(1B)	X(1B)	X(2B)	X(4B)	0x04

Payload:

B1(ESN)	B2(ESN)	B3(ESN)	B4(ESN)

with the payload as in the original command.

Error Codes:

RC_OK **(0x00):** success
RC_ENET **(0x03):** error, no network ID (unbound node)
RC_EMEM **0x0B):** error, no memory to send the message

Notes:

None

Copyright 2006 - 2007

Examples:

Send a status acknowledgement:

```
00 08 06 AA 25 00 00 00 00 00 04
```

B1(ESN)	B2(ESN)	B3(ESN)	B4(ESN)
0x00	0x00	0x00	0x00

Returns:

```
00 09 06 AA 01 25 00 00 00 00 00 04
```

B1(ESN)	B2(ESN)	B3(ESN)	B4(ESN)
0x00	0x00	0x00	0x00

The status message being sent by the target node is acknowledged.

Copyright 2006 - 2007

3.5.5 CMD_TRACE

Layout:

BOT	op_len	**CMD_TRACE**	op_ref	**node_id**	**payload**	EOT
0x00	0x06	**0x04**	X(1B)	**X(2B)**	**X(2B)**	0x04

Payload:

B1(hops)	B2(direction)

Synopsis:

CMD_TRACE (0x04): This command traces routes backwards, forwards or bi-directionally.

Specifics:

This command results in unencrypted messages building data on each hop (message length increases).
Directions: backward (0), forward (1), bidirectional (2).

Factory Settings or Defaults:

N/A

Valid Parameter Range:

N/A

Power-up Value:

N/A

Results Returned:

A response message is returned:

BOT	op_len	**msg_output**	msg_type	**msg_specific**	EOT
0x00	X(xB)	**0xFF**	X(1B)	**X(nB)**	0x04

Message:

B1 (# of forward hops)	B2 (# of backward hops)	B3...Bn (node ids)*

The source/termination node ID of the tracing loop is not shown in the output. Only the node IDs in the requested direction are collected and reported.

Error Codes:

RC_OK **(0x00):** successful
RC_ENET **(0x03):** error, no network ID (unbound node)
RC_ELEN **(0x08):** error, bad payload length

Notes:

None

Copyright 2006 - 2007

Examples:

Trace backwards to node 0x0035:

```
00 06 04 DD 35 00 04 00 04
```

B1(# of hops)	B2(direction)
0x04	0x00

The response message (msg_traceBAck) is 0x12:

```
00 06 FF 12 02 02 35 00 25 00 04
```

B1(# of forward hops)	B2(# of backward hops)	B3(node_id)
0x02	0x02	0x35

…

B4(node_id)	B5(node_id)	B6(node_id)
0x00	0x25	0x00

Do a bidirectional trace to node 0x0045:

```
00 06 04 DD 45 00 04 02 04
```

B1(# of hops)	B2(direction)
0x04	0x02

The response message (msg_traceAck) is 0x03:

```
00 0E FF 03 03 03 25 00 35 00 45 00 35 00 25 00 04
```

B1(# of forward hops)	B2(# of backward hops)	B3(node_id)
0x03	0x03	0x25

…

B4(node_id)	B5(node_id)	B6(node_id)	B7(node_id)
0x00	0x35	0x00	0x45

…

B8(node_id)	B9(node_id)	B10(node_id)	B11(node_id)
0x00	0x35	0x00	0x25

…

B12(node_id)
0x00

Trace forward to node 0x0035:

```
00 06 04 DD 35 00 03 01 04
```

B1(# of hops)	B2(direction)
0x03	0x01

The response message (msg_traceFAck) is 0x10:

```
00 06 FF 10 02 02 25 00 35 00 04
```

B1(# of forward hops)	B2(# of backward hops)	B3(node_id)
0x02	0x02	0x25

…

B4(node_id)	B5(node_id)	B6(node_id)
0x00	0x35	0x00

Copyright 2006 - 2007

3.5.6 CMD_RESET

Layout:

BOT	op_len	**CMD_RESET**	op_ref	node_id	**payload**	EOT
0x00	0x05	**0x09**	X(1B)	X(2B)	X(1B)	0x04

Payload:

B1(level)

Synopsis:

CMD_RESET (0x09): This command resets a node to various configurations.

Specifics:

B1 is zero: software restart

B1 is non zero: cold boot (all nonvolatile memory erased and software restarted)

Factory Settings or Defaults:

N/A

Valid Parameter Range:

zero or non-zero

Power-up Value:

N/A

Results Returned:

Result is returned if forced and possible, in usual format, with added op_rc field:

BOT	op_len	**CMD_SACK**	op_ref	op_rc	node_id	**payload**	EOT
0x00	0x06	**0x06**	X(1B)	X(1B)	X(2B)	X(1B)	0x04

Payload:

B1(level)

with payload as in the original command.

Error Codes:

RC_OK **(0x00):** success

RC_ELEN **(0x08):** error, bad length

Notes:

After reboots, no results are returned, but usual messages appear (bind requests, status reports).

Copyright 2006 - 2007

Examples:

Cold boot the software (all nonvolatile memory erased and software restarts):

```
00 05 09 DD 25 00 05 04
```

B1(level)
0x08

A bind request message is generated by the node that has been reset:

```
00 0A FF 06 01 00 CA BA 01 00 00 01 04
```

B1(Newcomer's ESN)	B2(Newcomer's ESN)	B3(Newcomer's ESN)
0x01	0x00	0xCA

…

B4(Newcomer's ESN)	B5(Newcomer's temp node_id)
0xBA	0x01

…

B6(Newcomer's temp node_id)	B7(binding node_id)
0x00	0x00

…

B8(binding node_id)
0x01

Returns:

```
00 06 09 DD 00 25 00 02 04
```

B1(level)
0x02

Software Reset :

```
00 05 09 DD 25 00 00 04
```

B1(level)
0x01

Returns:

```
00 07 FF 08 25 00 00 34 01 04
```

msg_br

B1(node_id)	B2(node_id)	B3(connectivity)	B4(connectivity)	B5(autoACK)
0x25	0x00	0x00	0x34	0x01

Copyright 2006 - 2007

3.6 GET Messages

3.6.1 CMD_GET PAR_LH

Layout:

BOT	op_len	**CMD_GET**	op_ref	node_id	**payload**	EOT
0x00	0x05	**0x01**	X(1B)	X(2B)	**0x01**	0x04

Synopsis:

PAR_LH (0x01): This command gets the 16-bit node ID of the target node.

Specifics:

None

Factory Settings or Defaults:

Lower 16 bits of the node's 32 bit electronic serial number (ESN)

Valid Parameter Range:

0x0001 to 0xFFFE (avoid 0x0000 or 0xFFFF)

Power-on Value:

Retrieved from nonvolatile memory.

Results Returned:

Results are always returned, in usual format, with added op_rc field:

BOT	op_len	CMD_GET	op_ref	op_rc	node_id	**payload**	EOT
0x00	0x08	0x01	X(1B)	X(1B)	X(2B)	**X(3B)**	0x04

Payload:

0x01	B1(node_id)	B2(node_id)

Error Codes:

None

Notes:

None

Examples:

Get local host (node ID):

00 05 **01** AA 25 00 **01** 04

parameter
0x01

Returns:

00 08 01 AA 01 25 00 01 **25 00** 04

parameter	B1 (node_id)	B2(node_id)
0x01	0x02	0x00

The target's node ID is: 0x0025

Copyright 2006 - 2007

3.6.2 CMD_GET ATTR_ESN

Layout:

BOT	op_len	**CMD_GET**	op_ref	node_id	**payload**	EOT
0x00	0x05	**0x01**	X(1B)	X(2B)	**0x02**	0x04

Synopsis:

PAR_ESN (0x02): This command gets the 32-bit equipment serial number (ESN).

Specifics:

None

Factory Settings or Defaults:

The 32-bit ESN is a factory installed constant unique to each DM2200 node.

Valid Parameter Range:

0x00000001 to 0xFFFFFFFE (note the lower 16 bits of the ESN are used as a temporary node ID when the node is bound to a network)

Power-up Value:

Constant retrieved from program memory.

Results Returned:

Results are always returned, in usual format, with added op_rc field:

BOT	op_len	CMD_GET	op_ref	op_rc	node_id	**payload**	EOT
0x00	0x0A	0x01	X(1B)	X(1B)	X(2B)	**X(5B)**	0x04

Payload:

0x02	B1(ESN)	B2(ESN)	B3(ESN)	B4(ESN)

Error Codes:

None

Notes:

None

Examples:

Get ESN:

00 05 **01** AA 25 00 **02** 04

parameter
0x02

Returns:

00 0A 01 AA 01 25 00 02 **01 00 CA BA** 04

parameter	B1(ESN)	B2(ESN)	B3(ESN)	B4(ESN)
0x02	0x01	0x00	0xCA	0xBA

The equipment serial number is 0xBACA0001.

Copyright 2006 - 2007

3.6.3 CMD_GET ATTR_MID

Layout:

BOT	op_len	**CMD_GET**	op_ref	node_id	**payload**	EOT
0x00	0x05	**0x01**	X(1B)	X(2B)	**0x03**	0x04

Synopsis:

ATTR_MID (0x03): This command gets the master node ID.

Specifics:

None

Factory Settings or Defaults:

0x00 (no master)

Valid Parameter Range:

0x0001 to 0xFFFE

Power-up Value:

Retrieved from nonvolatile memory.

Results Returned:

Results are always returned, in usual format, with added op_rc field:

BOT	op_len	CMD_GET	op_ref	op_rc	node_id	**payload**	EOT
0x00	0x08	0x01	X(1B)	X(1B)	X(2B)	**X(3B)**	0x04

Payload:

0x03	B1(master_id)	B2(master_id)

Error Codes:

None

Notes:

None

Examples:

Get Master Node ID:

```
00 05 01 AA 25 00 03 04
```

parameter
0x03

Returns:

```
00 08 01 AA 01 25 00 03 00 01 04
```

parameter	B1(master_id)	v2(master_id)
0x03	0x00	0x01

The master node ID is: 0x0100.

Copyright 2006 - 2007

3.6.4 CMD_GET PAR_NID

Layout:

BOT	op_len	**CMD_GET**	op_ref	node_id	**payload**	EOT
0x00	0x05	**0x01**	X(1B)	X(2B)	**0x04**	0x04

Synopsis:

PAR_NID (0x04): This command gets the 16-bit network ID.

Specifics:

This 16-bit ID indicates to which network a node belongs.

Factory Settings or Defaults:

0 (no network affiliation)

Valid Parameter Range:

0x0001 to 0xFFFE

Power-up Value:

Retrieved from nonvolatile memory.

Results Returned:

Results are always returned, in usual format, with added op_rc field:

BOT	op_len	CMD_GET	op_ref	op_rc	node_id	**payload**	EOT
0x00	0x08	0x01	X(1B)	X(1B)	X(2B)	**X(3B)**	0x04

Payload:

0x04	B1(net_id)	B2(net_id)

Error Codes:

None

Notes:

None

Examples:

Get Network ID:

```
00 05 01 AA 25 00 04 04
```

parameter
0x04

Returns:

```
00 08 01 AA 01 25 00 04 10 00 04
```

parameter	B1(net_id)	B2(net_id)
0x04	0x10	0x00

Network ID: 0x0010

Copyright 2006 - 2007

3.6.5 CMD_GET PAR_TARP_F

Layout:

BOT	op_len	**CMD_GET**	op_ref	node_id	**payload**	EOT
0x00	0x05	**0x01**	X(1B)	X(2B)	**0x08**	0x04

Synopsis:

PAR_TARP_F (0x08): This command gets the TARP mesh routing rule *forwarding* parameter.

Specifics:

TARP parameters: 1 byte (bit 0 is the least significant bit):

Bit 0: **Forwarding ON/OFF**
Bits 1 - 3: TARP slack (0 - 7)
Bits 4 - 5: TARP route recovery (0 - 3)
Bits 6 - 7: TARP level (0 - 2)

Factory settings, defaults

0xA3, or bit pattern 10 10 001 **1**

Forwarding (bit 0) is 1 (ON)
TARP slack (bit 1 - bit 3) is 001
TARP route recovery (bit 4 - bit 5) is 10
TARP level (bit 6 - bit 7)

Valid Parameter Range:

Compound ranges.

Power-up Value:

Retrieved from nonvolatile memory (routing is forced OFF on the master).

Results Returned:

Results are always returned, in usual format, with added op_rc field:

BOT	op_len	CMD_GET	op_ref	op_rc	node_id	**payload**	EOT
0x00	0x07	0x01	X(1B)	X(1B)	X(2B)	**X(2B)**	0x04

Payload:

0x08	B1(tarp)

Error Codes:

None

Notes:

TARP parameters are separated for consistency with SET commands. GET returns always all of them in a single byte.

Copyright 2006 - 2007

Examples:

Get TARP parameter *Forwarding*:

```
00 05 01 AA 25 00 08 04
```

parameter
0x08

Returns:

```
00 07 01 AA 01 25 00 08 A3 04
```

parameter	B1(tarp)
0x08	0xA3

TARP parameters: 0xA3, or bit pattern 10 10 001 **1**

Bit 0:	**Forwarding ON/OFF**
Bits 1 - 3:	TARP slack (0 - 7)
Bits 4 - 5:	TARP route recovery (0 - 3)
Bits 6 - 7:	TARP level (0 - 2)

Get TARP *forwarding* parameter:

```
00 05 01 AA 25 00 08 04
```

parameter
0x08

Returns:

```
00 07 01 AA 01 25 00 08 58 04
```

parameter	B1(tarp_f)
0x08	0x58

TARP parameters: 0x58, or bit pattern 01 01 100 **0**

Forwarding (bit 0) is 0 (OFF)
TARP slack (bit 1 - bit 3) is 001
TARP route recovery (bit 4 - bit 5) is 10
TARP level (bit 6 - bit 7)

Copyright 2006 - 2007

3.6.6 CMD_GET PAR_TARP_L

Layout:

BOT	op_len	**CMD_GET**	op_ref	node_id	**payload**	EOT
0x00	0x05	**0x01**	X(1B)	X(2B)	**0x05**	0x04

Synopsis:

PAR_TARP_L (0x05): This command gets the TARP *level* mesh routing parameter.

Specifics:

TARP parameters: 1 byte (bit 0 is the least significant bit):

Bit 0: Forwarding ON/OFF
Bits 1 - 3: TARP slack (0 - 7)
Bits 4 - 5: TARP route recovery (0 - 3)
Bits 6 - 7: TARP level (0 - 2)

TARP levels:

Level 0 - Full flooding with discard on maximum hop count.
Level 1 - Duplicate Discards (DD) - when ON, duplicate packets are discarded.
Level 2 - Duplicate Discards plus Suboptimal Path Discards (SPD). If the packet's destination is known, the node discards packets on suboptimal paths. Number of discarded packets to a destination is kept to play a role in route recovery, together with the TARP *slack* and *route recovery* parameters.

Factory Settings or Defaults:

0xA3, or bit pattern **10** 10 001 1

Forwarding (bit 0) is 1 (ON)
TARP slack (bit1 - bit 3) is 001 (1)
TARP route recovery (bit 4 - bit 5) is 10 (2)
TARP level (bit 6 - bit 7) is 10 (2)

Valid Parameter Range:

Compound ranges

Power-up Value:

Retrieved from nonvolatile memory (routing is forced off at the master).

Results Returned:

Results are always returned, in usual format, with added op_rc field:

BOT	op_len	CMD_GET	op_ref	op_rc	node_id	**payload**	EOT
0x00	0x08	0x01	X(1B)	X(1B)	X(2B)	**X(2B)**	0x04

Payload:

0x05	B1(tarp)

Copyright 2006 - 2007

Error Codes:

None

Notes:

TARP parameters are separated for consistency with SET commands. GET returns all of them in a single byte.

Examples:

Get TARP level:

```
00 05 01 AA 25 00 05 04
```

parameter
0x05

Returns:

```
00 07 01 AA 01 25 00 05 A3 04
```

parameter	B1(tarp)
0x05	0xA3

TARP parameters: 0xA3 or bit pattern 10 10 001 1

Forwarding (bit 0) is 1 (ON)
TARP slack (bit 1 - bit 3) is 001
TARP route recovery (bit 4 - bit 5) is 10
TARP level (bit 6 - bit 7) is 10

Get TARP level (after TARP level was SET to 1):

```
00 05 01 AA 25 00 05 04
```

parameter
0x05

Returns:

```
00 07 01 AA 01 25 00 05 63 04
```

parameter	B1(tarp)
0x05	0x63

TARP parameters: 0x63 or bit pattern 01 10 001 1

Forwarding (bit 0) is 1 (ON)
TARP slack (bit 1 - bit 3) is 001
TARP route recovery (bit 4 - bit 5) is 10
TARP level (bit 6 - bit 7) 01

Copyright 2006 - 2007

3.6.7 CMD_GET PAR_TARP_S

Layout:

BOT	op_len	**CMD_GET**	op_ref	node_id	**payload**	EOT
0x00	0x05	**0x01**	X(1B)	X(2B)	**0x06**	0x04

Synopsis:

PAR_TARP_S (0x06): This command gets the TARP *slack* mesh routing parameter.

Specifics:

TARP parameters: 1 byte (bit 0 is the least significant bit):

Bit 0: Forwarding ON/OFF
Bits 1 - 3: TARP slack (0 - 7)
Bits 4 - 5: TARP route recovery (0 - 3)
Bits 6 - 7: TARP level (0 - 2)

TARP slack:

Suboptimal path discard will add the slack value when examining the caches, so suboptimal paths may become satisfactory for route recovery. The default value for slack (1) should not be changed without consulting Cirronet.

Factory Settings or Defaults:

0xA3, or bit pattern 10 10 **001** 1

Forwarding (bit 0) is 1 (ON)
TARP slack (bit 1- bit 3) is 001
TARP route recovery (bit 4 - bit 5) is 10
TARP level (bit 6 - bit 7) is 10

Valid Parameter Range:

Compound ranges

Power-up Value:

Retrieved from nonvolatile memory (routing is forced off at the master).

Results Returned:

Results are always returned, in usual format, with added op_rc field:

BOT	op_len	CMD_GET	op_ref	op_rc	node_id	**payload**	EOT
0x00	0x07	0x01	X(1B)	X(1B)	X(2B)	**X(2B)**	0x04

Payload:

0x06	B1(tarp)

Error Codes:

None

Notes:

TARP parameters are separated for consistency with SET commands. GET returns all of them in a single byte.

Copyright 2006 - 2007

Examples:

Get TARP slack parameter:
`00 05 01 AA 25 00 06 04`

parameter
0x06

Returns:
`00 07 01 AA 01 25 00 06 A3 04`

parameter	B1(tarp)
0x05	0xA3

TARP parameters: 0xA3, or bit pattern 10 10 **001** 1
Forwarding (bit 0) is 1 (ON)
TARP slack (bit 1 - bit 3) is 001 (1)
TARP route recovery (bit 4 - bit 5) is 10 (2)
TARP level (bit 6 - bit 7) is 10 (2)

Get TARP Slack (after TARP level and slack were SET):
`00 05 01 AA 25 00 06 04`

parameter
0x06

Returns:
`00 07 01 AA 01 25 00 06 69 04`

parameter	B1(tarp)
0x05	0x69

TARP parameters: 0x69, or bit pattern 01 10 100 1
Forwarding (bit 0) is 1 (ON)
TARP slack (bit 1 - bit 3) is 100 (4)
TARP route recovery (bit 4 - bit 5) is 10 (2)
TARP level (bit 6 - bit 7) is 01 (1)

Copyright 2006 - 2007

3.6.8 CMD_GET PAR_TARP_R

Layout:

BOT	op_len	**CMD_GET**	op_ref	node_id	**payload**	EOT
0x00	0x05	**0x01**	X(1B)	X(2B)	**0x07**	0x04

Synopsis:

PAR_TARP_R (0x07): This command gets the TARP mesh routing rule *route recovery speed* parameter.

Specifics:

TARP parameters: 1 byte (bit 0 is the least significant bit):

Bit 0: Forwarding ON/OFF
Bits 1 - 3: TARP slack (0 - 7)
Bits 4 - 5: TARP route recovery (0 - 3)
Bits 6 - 7: TARP level (0 - 2)

TARP route recovery:

Suboptimal path discard exponentially weighs discards using the route recovery speed parameter:

0: additional hop for every discard
1: additional hop for every 4 discards
2: additional hop for every 8 discards
3: route discovery disabled (no discards counting)

The default value for route recovery (2) should not be changed without consulting Cirronet.

Factory Settings or Defaults:

0xA3, or bit pattern 10 **10** 001 1

Forwarding (bit 0) is 1 (ON)
TARP slack (bit 1 - bit 3) is 001
TARP route recovery (bit 4 - bit 5) is 10
TARP level (bit 6 - bit 7)

Valid Parameter Range:

Compound ranges

Power-up Value:

Retrieved from nonvolatile memory (routing is forced off at the master).

Results Returned:

Results are always returned, in usual format, with added op_rc field:

BOT	op_len	CMD_GET	op_ref	op_rc	node_id	**payload**	EOT
0x00	0x07	0x01	X(1B)	X(1B)	X(2B)	**X(2B)**	0x04

Payload:

0x07	B1(tarp)

Copyright 2006 - 2007

Error Codes:

None

Notes:

TARP parameters are separated for consistency with SET commands. GET returns always all of them in a single byte.

Examples:

Get TARP route recovery:

`00 05 **01** AA 25 00 **07** 04`

parameter
0x07

Returns:

`00 07 01 AA 01 25 00 07 **A3** 04`

parameter	B1(tarp_rr)
0x07	0xA3

TARP parameters: 0xA3, or bit pattern 10 10 001 1

Forwarding (bit 0) is 1 (ON)
TARP slack (bit 1 - bit 3) is 001
TARP route recovery (bit 4 - bit 5) is 10
TARP level (bit 6 - bit 7) is 10

Get TARP route recovery (after TARP level and route recovery were set):

`00 05 **01** AA 25 00 **07** 04`

parameter
0x07

Returns:

`00 07 01 AA 01 25 00 07 **59** 04`

parameter	B1(tarp_rr)
0x07	0x59

TARP parameters: 0x59, or bit pattern 01 01 001 1

Forwarding (bit 0) is 1 (ON)
TARP slack (bit 1 - bit 3) is 001
TARP route recovery (bit 4 - bit 5) is 01
TARP level (bit 6 - bit 7) is 01

Copyright 2006 - 2007

3.6.9 CMD_GET PAR_BEAC

Layout:

BOT	op_len	**CMD_GET**	op_ref	node_id	**payload**	EOT
0x00	0x05	**0x01**	X(1B)	X(2B)	**0x09**	0x04

Synopsis:

PAR_BEAC (0x09): This command gets the beacon interval.

Specifics:

Beacon interval for the master node
Bind request message interval for an unbound field node

Factory Settings or Defaults:

29 seconds for master node, 31 seconds for unbound node

Valid Parameter Range:

0 to 63, in seconds

Power-up Value:

Defaults retrieved from program memory:

- On unbound nodes: 31 seconds, (bind request message will be sent every 31 seconds).
- On the master: 29 seconds (master beacon will be sent every 29 seconds).
- Bound field nodes: 0 seconds (bind request messaging disabled).

Results Returned:

Results are always returned, in usual format, with added op_rc field:

BOT	op_len	CMD_GET	op_ref	op_rc	node_id	**payload**	EOT
0x00	0x07	0x01	X(1B)	X(1B)	X(2B)	**X(2B)**	0x04

Payload:

0x09	B1(b_freq)

Error Codes:

None

Notes:

None

Examples:

Get beacon interval:

```
00 05 01 AA 00 01 09 04
```

parameter
0x09

Returns:

```
00 07 01 AA 01 00 01 09 1E 04
```

parameter	B1(b_freq)
0x09	**0x1E**

The master beacon interval is: 0x1E (29 seconds).

Copyright 2006 - 2007

3.6.10 CMD_GET PAR_CON

Layout:

BOT	op_len	**CMD_GET**	op_ref	node_id	**payload**	EOT
0x00	0x05	**0x01**	X(1B)	X(2B)	**0x0A**	0x04

Synopsis:

PAR_CON (0x0A): This command gets connectivity data consisting of beacon interval, number of missed beacons to trigger a connectivity warning action, number of additional missed beacons to trigger a bad connectivity action, and number of missed beacons in audit period.

Specifics:

Connectivity:

Byte 1: audit period in seconds

Byte 2:

bits 4 - 7: number of missed beacons to warning (start blinking connectivity LED),

bits 0 - 3: number of additional missed beacons to bad status (connectivity LED OFF).

Byte 3: number of missed beacons

Factory Settings or Defaults:

Decided at power up (29 or 0, 3, 4).

Valid Parameter Range:

Audit period: 1 to 63 seconds, use 0 to disable audit

Missed beacons to warning (ON or bit pattern blink), 0, 15

Missed beacons to bad (blink or bit pattern OFF), 0, 15

Power-up Value:

Defaults retrieved from program memory:

Connectivity:

Master node: equal to beacon interval (29 seconds)

Other nodes: 0 (will be dynamically set by master beacon payload)

Number of missed beacons to warn: 3

Number of additional missed beacons to bad: 4

Results Returned:

Results are always returned, in usual format, with added op_rc field:

BOT	op_len	CMD_GET	op_ref	op_rc	node_id	**payload**	EOT
0x00	0x09	0x01	X(1B)	X(1B)	X(2B)	**X(4B)**	0x04

Payload:

0x0A	B1(audit)	B2(warn,bad)	B3(missed)

Copyright 2006 - 2007

Error Codes:

None

Notes:

None

Examples:

Get connectivity data:

```
00 05 01 AA 25 00 0A 04
```

parameter
0x0A

Returns:

```
00 09 01 AA 01 25 00 0A 1E 34 00 04
```

parameter	B1(audit)	B2(warn, bad)	B3(missed)
0x0A	0x1D	0x34	0x00

Connectivity data:

Byte 1 (a1): audit period, 0x1D

Byte 2 (a2):

missed beacons to warning (bits 4 - 7): 3 missed beacons

missed beacons to bad status (bits 0 - 3): next 4 missed beacons

Byte 3: number of missed beacons: 0

Copyright 2006 - 2007

3.6.11 CMD_GET PAR_RF

Layout:

BOT	op_len	**CMD_GET**	op_ref	node_id	**payload**	EOT
0x00	0x05	**0x01**	X(1B)	X(2B)	**0x0B**	0x04

Synopsis:

PAR_RF (0x0B): This command gets the Radio ON/OFF status (TX, RX).

Specifics:

TX is ON when Bit 1 is set, RX is ON when Bit 0 is set

Factory Settings or Defaults:

3 (TX ON and RX ON)

Valid Parameter Range:

0 to 3

Power-up Value:

Not backed up. Default retrieved from program memory.

Results Returned:

Results are always returned, in usual format, with added op_rc field:

BOT	op_len	CMD_GET	op_ref	op_rc	node_id	**payload**	EOT
0x00	0x07	0x01	X(1B)	X(1B)	X(2B)	**X(2B)**	0x04

Payload:

0x0B	B1(tx/rx)

Error Codes:

None

Notes:

RF status is also controlled by power management cycles.

Examples:

Get TX and RX status parameters:

00 05 **01** AA 25 00 **0B** 04

parameter
0x0B

Returns:

00 07 01 AA 01 25 00 0B **03** 04

parameter	B1(tx/rx)
0x0B	0x03

0x03, or bit pattern 11
Transmitter: ON
Receiver: ON

Copyright 2006 - 2007

3.6.12 CMD_GET PAR_STAT_REP

Layout:

BOT	op_len	**CMD_GET**	op_ref	node_id	**payload**	EOT
0x00	0x05	**0x01**	X(1B)	X(2B)	**0x0D**	0x04

Synopsis:

PAR_STAT_REP (0x0D): This command gets the status message reporting interval.

Specifics:

Byte 1 - 2: interval in 64 second increments
Byte 3: not used by the DM2200
Byte 4 - 5: time left to next reporting, in minutes.

Factory Settings or Defaults:

450 increments, or 8 hours (450*64 = 28,800 seconds = 8 hours)

Valid Parameter Range:

0x0000 to 0xFFFF

Power-up Value:

Default retrieved from program memory.

Results Returned:

Results are always returned, in usual format, with added op_rc field:

BOT	op_len	CMD_GET	op_ref	op_rc	node_id	**payload**	EOT
0x00	0x0B	0x01	X(1B)	X(1B)	X(2B)	**X(6B)**	0x04

Payload:

0x0D	B1(freq)	B2(freq)	B3(reporting)	B4(time)	B5(time)

Error Codes:

None

Examples:

Status report parameters:

00 05 **01** AA 25 00 **0D** 04

parameter
0x0D

Returns:

00 0B 01 AA 01 25 00 0D **C2 01 00 9C 01** 04

parameter	B1(freq)	B2(freq)	B3(reporting)	B4(time)	B5(time)
0x0D	0xC2	0x01	0x00	0x26	0x01

The reporting interval is 0x01C2, or 450 increments of 64 seconds.
Time left to next reporting is 0x0126 or 294 increments of 64 seconds.

Copyright 2006 - 2007

3.6.13 CMD_GET PAR_RETRY

Layout:

BOT	op_len	**CMD_GET**	op_ref	node_id	**payload**	EOT
0x00	0x05	**0x01**	X(1B)	X(2B)	**0x0E**	0x04

Synopsis:

PAR_RETRY (0x0E): This command gets the reliable reporting configuration - number of retries and timeout period.

Specifics:

None

Factory Settings or Defaults:

Number of retries: 3
Timeout in seconds: 10

Valid Parameter Range:

Number of retries: 1 to 15 (0 disables retries
Timeout: 1 to 15 seconds

Power-up Value:

Defaults retrieved from program memory.

Results Returned:

Results are always returned, in usual format, with added op_rc field:

BOT	op_len	CMD_GET	op_ref	op_rc	node_id	**payload**	EOT
0x00	0x08	0x01	X(1B)	X(1B)	X(2B)	**X(3B)**	0x04

Payload:

0x0E	B1(# of retries)	B2(timeout)

Error Codes:

None

Notes:

None

Examples:

Get the number of retries and the timeout period:

```
00 05 01 AA 25 00 0E 04
```

parameter
0x0E

Returns:

```
00 08 01 AA 01 25 00 0E 03 0A 04
```

parameter	B1(# of retries)	B2(timeout)
0x0E	0x03	0x0A

The number of retries is 3 and the timeout period is 0x0A (10 seconds).

Copyright 2006 - 2007

3.6.14 CMD_GET PAR_AUTOACK

Layout:

BOT	op_len	**CMD_GET**	op_ref	node_id	**payload**	EOT
0x00	0x05	**0x01**	X(1B)	X(2B)	**0x0F**	0x04

Synopsis:

PAR_AUTOACK (0x0F): This command gets the auto-acknowledgment ON/OFF status.

Specifics:

None

Factory Settings or Defaults:

1 (ON)

Valid Parameter Range:

0 (OFF) or 1 (ON)

Power-up Value:

Default retrieved from program memory.

Results Returned:

Results are always returned, in usual format, with added op_rc field:

BOT	op_len	CMD_GET	op_ref	op_rc	node_id	**payload**	EOT
0x00	0x07	0x01	X(1B)	X(1B)	X(2B)	**X(2B)**	0x04

Payload:

0x0F	B1(ACK)

Error Codes:

None

Notes:

Usually a master node parameter.

Examples:

Get the auto-acknowledgment status:

00 05 **01** AA 25 00 **0F** 04

parameter
0x0F

Returns:

00 07 01 AA 01 25 00 0F **01** 04

parameter	B1(ACK)
0x0F	0x01

Auto-acknowledgment is on.

Copyright 2006 - 2007

3.6.15 CMD_GET ATTR_UPTIME

Layout:

BOT	op_len	**CMD_GET**	op_ref	node_id	**payload**	EOT
0x00	0x05	**0x01**	X(1B)	X(2B)	**0x10**	0x04

Synopsis:

ATTR_UPTIME (0x10): This command gets the time in seconds since power up.

Specifics:

None

Factory Settings or Defaults:

N/A

Valid Parameter Range:

0x00000000 to 0xFFFFFFFF

Power-up Value:

0x00000000

Results Returned:

Results are always returned, in usual format, with added op_rc field:

BOT	op_len	**CMD_GET**	op_ref	op_rc	node_id	**payload**	EOT
0x00	0x0A	**0x01**	X(1B)	X(1B)	X(2B)	**X(5B)**	0x04

Payload:

0x10	B1(time)	B2(time)	B3(time)	B4(time)

Error Codes:

None

Notes:

None

Examples:

Get uptime:

```
00 05 01 AA 25 00 10 04
```

parameter
0x10

Returns:

```
00 0A 01 AA 01 25 00 10 3B 4E 00 00 04
```

parameter	B1(time)	B2(time)	B3(time)	B4(time)
0x10	0x3B	0x4E	0x00	0x00

Uptime is 0x00004E3B (20027 seconds).

Copyright 2006 - 2007

3.6.16 CMD_GET ATTR_RSSI

Layout:

BOT	op_len	**CMD_GET**	op_ref	node_id	**payload**	EOT
0x00	0x05	**0x01**	X(1B)	X(2B)	**0x12**	0x04

Synopsis:

ATTR_RSSI (0x12): This command gets the received signal strength (RSSI) of the last message sent by the master.

Specifics:

For the DM2200, the first byte is the RSSI level and the second byte is 0.

Factory Settings or Defaults:

N/A

Valid Parameter Range:

0 to 255 for the DM2200

Power-up Value:

N/A

Results Returned:

Results are always returned, in usual format, with added op_rc field:

BOT	op_len	**CMD_GET**	op_ref	op_rc	node_id	**payload**	EOT
0x00	0x08	**0x01**	X(1B)	X(1B)	X(2B)	**X(3B)**	0x04

Payload:

0x12	B1(RSSI)	B2(0)

Error Codes:

None

Notes:

B1: RSSI, B2: 0

Examples:

Get the RSSI level:

```
00 05 01 AA 25 00 12 04
```

parameter
0x12

Returns

```
00 08 01 AA 01 25 00 12 6E 00 04
```

parameter	B1(RSSI)	B2(LQ)
0x12	0x6E	0x00

The RSSI value is 0x6E.

Copyright 2006 - 2007

3.6.17 CMD_GET ATTR_MSG_COUNT_T

Layout:

BOT	op_len	**CMD_GET**	op_ref	node_id	**payload**	EOT
0x00	0x05	**0x01**	X(1B)	X(2B)	**0x13**	0x04

Synopsis:

ATTR_MSG_COUNT_T (0x13): This command gets the TARP message counters: tarp_rcv, tarp_snd, and tarp_fwd.

Specifics:

Useful in evaluating network routing statistics.

Factory Settings or Defaults:

N/A

Valid Parameter Range:

0x0000 to 0xFFFF for each counter

Power-up Value:

N/A

Results Returned:

Results are always returned, in usual format, with added op_rc field:

BOT	op_len	CMD_GET	op_ref	op_rc	node_id	**payload**	EOT
0x00	0x0C	0x01	X(1B)	X(1B)	X(2B)	**X(7B)**	0x04

Payload:

0x13	B1(rcv)	B2(recv)	B3(snd)	B4(snd)	B5(fwd)	B6(fwd)

Error Codes:

None

Notes:

None

Examples:

Get the TARP message counters:

00 05 **01** AA 25 00 **13** 04

parameter
0x13

Returns:

00 0C 01 AA 01 25 00 13 **A1 02 BB 02 00 00** 04

parameter	B1(rcv)	B2(recv)	B3(snd)	B4(snd)	B5(fwd)	B6(fwd)
0x13	0xA1	0x02	0xBB	0x02	0x00	0x00

Message counters: 0x02A1 (tarp_rcv), 0x02BB (tarp_snd), 0x0000 (tapr_fwd)

Copyright 2006 - 2007

3.6.18 CMD_GET ATTR_MEM1

Layout:

BOT	op_len	**CMD_GET**	op_ref	node_id	**payload**	EOT
0x00	0x05	**0x01**	X(1B)	X(2B)	**0x15**	0x04

Synopsis:

ATTR_MEM1 (0x15): This command gets the memory statistics - free memory, minimum free memory, minimum free stack.

Specifics:

Current free memory
Minimum free memory so far
Minimum free stack so far

Factory Settings or Defaults:

N/A

Valid Parameter Range:

0x0000 to 0xFFFF for each parameter

Power-up Value:

N/A

Results Returned:

Results are always returned, in usual format, with added op_rc field:

BOT	op_len	CMD_GET	op_ref	op_rc	node_id	**payload**	EOT
0x00	0x0C	0x01	X(1B)	X(1B)	X(2B)	**X(7B)**	0x04

Payload:

0x15	B1(free memory)	B2(free memory)	B3(min free memory)

...

B4(min free memory)	B5(min stack)	B6(min stack)

Error Codes:

None

Notes:

None

Copyright 2006 - 2007

Examples:

Get memory statistics:

```
00 05 01 AA 00 01 15 04
```

operation
0x15

Returns:

```
00 0C 01 AA 01 00 01 15 9F 01 7C 01 51 00 04
```

operation	B1(free memory)	B2(free memory)	B3(min free memory)
0x15	0x9F	0x01	0x7C

…

B4(min free memory)	B5(free stack)	B6(free stack)
0x01	0x51	0x00

Memory statistics:

current free memory is 0x019F
minimum free memory so far is 0x017C
minimum free stack so far is 0x0051

Copyright 2006 - 2007

3.6.19 CMD_GET ATTR_MEM2

Layout:

BOT	op_len	**CMD_GET**	op_ref	node_id	**payload**	EOT
0x00	0x05	**0x01**	X(1B)	X(2B)	**0x16**	0x04

Synopsis:

ATTR_MEM12 (0x16): This command gets additional memory statistics - max free chunk, free chunks, static size.

Specifics:

Maximum free memory chunk
Number of free chunks
Size of the static segment

Factory Settings or Defaults:

N/A

Valid Parameter Range:

0x0000 to 0xFFFF for each parameter

Power-up Value:

N/A

Results Returned:

Results are always returned, in usual format, with added op_rc field:

BOT	op_len	**CMD_GET**	op_ref	op_rc	node_id	**payload**	EOT
0x00	0x0C	**0x01**	X(1B)	X(1B)	X(2B)	**X(7B)**	0x04

Payload:

0x16	B1(max free memory)	B2(max free memory)	B3(free chunks)

…

B4(free chunks)	B5(static size)	B6(static size)

Error Codes:

None

Notes:

None

Copyright 2006 - 2007

Examples:

Get additional memory statistics:

00 05 **01** AA 00 01 **16** 04

parameter
0x16

Returns:

00 0C 01 AA 01 00 01 16 **96 01 02 00 AC 01** 04

0x16	B1(max free memory)	B2(max free memory)	B3(free chunks)
0x16	0x96	0x01	0x02

…

B4(free chunks)	B5(static size)	B6(static size)
0x00	0xAC	0x01

Memory statistics:

maximum free memory is 0x0196
number of free chunks is 0x0002
static size is 0x01AC

Copyright 2006 - 2007

3.6.20 CMD_GET PAR_ENCR_M

Layout:

BOT	op_len	**CMD_GET**	op_ref	node_id	**payload**	EOT
0x00	0x05	**0x01**	X(1B)	X(2B)	**0x17**	0x04

Synopsis:

PAR_ENCR_M (0x17): This command gets the encryption mode.

Specifics:

Encryption data:

Bits 1 - 0: key number

Bit 2: mode

Factory Settings or Defaults:

Encryption data:

Key number: 0 (no encryption)

Mode: 0 (*free*)

Valid Parameter Range:

Key: 0 - 3

Mode: 0 or 1

Power-up Value:

Retrieved from nonvolatile memory.

Results Returned:

Results are always returned, in usual format, with added op_rc field:

BOT	op_len	**CMD_GET**	op_ref	op_rc	node_id	**payload**	EOT
0x00	0x07	**0x01**	X(1B)	X(1B)	X(2B)	**X(2B)**	0x04

Payload:

0x17	B1(key_mode)

Error Codes:

None

Notes:

None

Copyright 2006 - 2007

Examples:

Get encryption mode:

```
00 05 01 AA 00 01 17 04
```

parameter
0x17

Returns:

```
00 07 01 AA 01 00 01 17 00 04
```

parameter	B1(key_mode)
0x17	0x00

Mode in bit 2 is 0.

Copyright 2006 - 2007

3.6.21 CMD_GET PAR_ENCR_K

Layout:

BOT	op_len	**CMD_GET**	op_ref	node_id	**payload**	EOT
0x00	0x05	**0x01**	X(1B)	X(2B)	**0x18**	0x04

Synopsis:

PAR_ENCR_K (0x18): This command gets the encryption key number.

Specifics:

Encryption data:

Bits 1 - 0: key number

Bit 2: mode

Factory Settings or Defaults:

Encryption data:

Key number: 0 (no encryption)

Mode: 0 (*free*)

Valid Parameter Range:

Key: 0 - 3

Mode: 0 or 1

Power-up Value:

Retrieved from nonvolatile memory.

Results Returned:

Results are always returned, in usual format, with added op_rc field:

BOT	op_len	**CMD_GET**	op_ref	op_rc	node_id	**payload**	EOT
0x00	0x07	**0x01**	X(1B)	X(1B)	X(2B)	**X(2B)**	0x04

Payload:

0x18	B1(key#)

Error Codes:

None

Notes:

None

Copyright 2006 - 2007

Examples:

Get encryption key number:

```
00 05 01 AA 00 01 18 04
```

parameter
0x18

Returns:

```
00 07 01 AA 01 00 01 18 00 04
```

parameter	B1(key#)
0x17	0x00

The key number in bits 0 - 1 is 0.

Copyright 2006 - 2007

3.6.22 CMD_GET PAR_ENCR_D

Layout:

BOT	op_len	**CMD_GET**	op_ref	node_id	**payload**	EOT
0x00	0x05	**0x01**	X(1B)	X(2B)	**0x19**	0x04

Synopsis:

PAR_ENCR_D (0x19): This command gets all encryption data (mode, key).

Specifics:

Encryption data:

Bits 1 - 0: key number

Bit 2: mode

Factory Settings or Defaults:

Encryption data:

Key number: 0 (no encryption)

Mode: 0 (*free*)

Valid Parameter Range:

Key: 0 - 3

Mode: 0 or 1

Power-up Value:

Retrieved from nonvolatile memory.

Results Returned:

Results are always returned, in usual format, with added op_rc field:

BOT	op_len	**CMD_GET**	op_ref	op_rc	node_id	**payload**	EOT
0x00	0x07	**0x01**	X(1B)	X(1B)	X(2B)	**X(2B)**	0x04

Payload:

0x19	B1(key and mode)

Error Codes:

None

Notes:

None

Copyright 2006 - 2007

Examples:

Get all encryption data:

```
00 05 01 AA 00 01 19 04
```

parameter
0x19

Returns:

```
00 07 01 AA 01 00 01 19 00 04
```

parameter	B1(key and mode)
0x19	0x00

The key number is in bits 0 -1 (0) and the mode is in bit 2 (0).

Copyright 2006 - 2007

3.6.23 CMD_GET ATTR_SYSVER

Layout:

BOT	op_len	**CMD_GET**	op_ref	node_id	**payload**	EOT
0x00	0x05	**0x01**	X(1B)	X(2B)	**0x1A**	0x04

Synopsis:

ATTR_SYSVER (0x1A): This command gets the system version.

Specifics:

System version (MM – major, mm – minor)

Factory Settings or Defaults:

N/A

Valid Parameter Range:

Major: 0x00 to 0xFF
Minor: 0x00 to 0xFF

Power-up Value:

Defaults retrieved from program memory.

Results Returned:

Results are always returned, in usual format, with added op_rc field:

BOT	op_len	**CMD_GET**	op_ref	op_rc	node_id	**payload**	EOT
0x00	0x08	**0x01**	X(1B)	X(1B)	X(2B)	**X(3B)**	0x04

Payload:

0x1A	B1(MM)	B2(mm)

Error Codes:

None

Notes:

None

Examples:

Get the system (firmware) version:

00 05 01 AA 00 01 **1A** 04

parameter
0x1A

Returns:

00 08 01 AA 01 00 01 **1A 02 00** 04

parameter	B1(MM)	B2(mm)
0x1A	0x02	0x00

Copyright 2006 - 2007

3.6.24 CMD_GET ATTR_NHOOD

Layout:

BOT	op_len	**CMD_GET**	op_ref	node_id	**payload**	EOT
0x00	0x05	**0x01**	X(1B)	X(2B)	**0x1B**	0x04

Synopsis:

ATTR_NHOOD (0x1B): This command gets the target node's neighbors (nodes within a single hop) to transmit msg_nhAck messages.

Specifics:

The pivotal node refers to the target node's ID
ESN refers to the electronic serial number of the reporting node
Number of hops in the response refers to the distance of the reporting node from the node that issued the command.

Factory Settings or Defaults:

N/A

Valid Parameter Range:

N/A

Power-up Value:

N/A

Results Returned:

Result is returned only if forced or with an error:

BOT	op_len	**CMD_GET**	op_ref	op_rc	node_id	**payload**	EOT
0x00	0x06	**0x01**	X(1B)	X(1B)	X(2B)	**0x1B**	0x04

And reports with msg_nhAck in format:

BOT	op_len	**msg_output**	msg_type	**msg_specific**	EOT
0x00	0x08	**0xFF**	X(1B)	**X(9B)**	0x04

Payload:

B1(pivotal node_id)	B2(pivotal node_id)	B3(ESN)	B4(ESN)

…

B5(ESN)	B6(ESN)	B7(node_id)	B8(node_id)	B9(# of hops)

Error Codes:

None

Notes:

None

Examples:

Get neighbors (from node 0x0035, 2 hops from the issuer):

`00 05 **01** **AA** 35 00 **1B** 04`

parameter
0x1B

Copyright 2006 - 2007

Returns:
00 06 01 AA **00** 35 00 **1B** 04

parameter
0x1B

Neighbors will report with msg_nhAck(0x0E):
00 0B FF 0E **35 00 01 00 CA BA 25 00 01** 04

msg_nhAck:

B1(pivotal node_id)	B2(pivotal node_id)	B3(ESN)	B4(ESN)
0x35	0x00	0x01	0x00

…

B5(ESN)	B6(ESN)	B7(node_id)	B8(node_id)	B9(# of hops)
0xCA	0xBA	0x25	0x00	0x01

00 0B FF 0E **35 00 01 00 CA BA 45 00 03** 04

msg_nhAck:

B1(pivotal node_id)	B2(pivotal node_id)	B3(ESN)	B4(ESN)
0x35	0x00	0x01	0x00

…

B5(ESN)	B6(ESN)	B7(node_id)	B8(node_id)	B9(# of hops)
0xCA	0xBA	0x45	0x00	0x03

B1-B2: 0x0035 - pivotal node id
B3-B6: 0xBACA0001 - ESN
B7-B8: 0x0045 - node id
B9: 0x03 - number of hops

Get neighbors (from node 0x0045, 3 hops away):
00 05 01 AA 45 00 **1B** 04

parameter
0x1B

Returns:
00 06 01 AA 00 45 00 **1B** 04

parameter
0x1B

Neighbors will report with a msg_nhAck message format:
00 0B FF 0E **45 00 01 00 CA BA 35 00 02** 04

B1(pivotal node_id)	B2(pivotal node_id)	B3(ESN)	B4(ESN)
0x45	0x00	0x01	0x00

…

B5(ESN)	B6(ESN)	B7(node_id)	B8(node_id)	B9(# of hops)
0xCA	0xBA	0x35	0x00	0x02

B1-B2: 0x0045 - pivotal node id
B3-B6: 0xBACA0001 - ESN
B7-B8: 0x0035 - node id
B9: 0x02 - number of hops

Copyright 2006 - 2007

3.6.25 CMD_GET PAR_BINDER

Layout:

BOT	op_len	**CMD_GET**	op_ref	node_id	**payload**	EOT
0x00	0x05	**0x01**	X(1B)	X(2B)	**0x1C**	0x04

Synopsis:

PAR_BINDER (0x1C): This command gets the bind enable flag state.

Specifics:

ON/OFF: when OFF, the node ignores bind request messages from unbound nodes.

Factory Settings or Defaults:

1 (ON)

Valid Parameter Range:

0 or 1

Power-up Value:

Retrieved from nonvolatile memory.

Results Returned:

Results are always returned, in usual format, with added op_rc field:

BOT	op_len	CMD_GET	op_ref	op_rc	node_id	**payload**	EOT
0x00	0x08	0x01	X(1B)	X(1B)	X(2B)	**X(3B)**	0x04

Payload:

0x1C	B1(ON/OFF)

Error Codes:

None

Notes:

None

Examples:

Get bind enable flag:

```
00 05 01 AA 35 00 1C 04
```

parameter
0x1C

Returns:

```
00 07 01 AA 01 35 00 1C 01 04
```

parameter	B1(yes/no)
0x1C	0x01

The bind flag is set and the node will forward bind request messages.

Copyright 2006 - 2007

3.6.26 CMD_GET PAR_UART

Layout:

BOT	op_len	**CMD_GET**	op_ref	node_id	**payload**	EOT
0x00	0x05	**0x01**	X(1B)	X(2B)	**0x1D**	0x04

Synopsis:

PAR_UART (0x1D): This command gets the UART baud rate.

Specifics:

Value represents 1/100 of the UART baud rate, for example, 192 represents 19,200 b/s. Note that the UART configuration is always N, 8, 1.

Factory Settings or Defaults:

192

Valid Parameter Range:

12, 24, 48, 96, 192, 384

Power-up Value:

Data retrieved from nonvolatile memory.

Results Returned:

Results are always returned, in usual format, with added op_rc field:

BOT	op_len	**CMD_GET**	op_ref	op_rc	node_id	**payload**	EOT
0x00	0x08	**0x01**	X(1B)	X(1B)	X(2B)	**X(2B)**	0x04

Payload:

0x1D	B1(speed)	B2(speed)

Error Codes:

None

Notes:

None

Examples:

Get the UART baud rate:

00 05 **01** AA 35 00 **1D** 04

parameter
0x1D

Returns:

00 08 01 AA 01 35 00 1D **C0 00** 04

parameter	B1(speed)	B2(speed)
0x1D	0xC0	0x00

The UART baud rate is 0X00C0 or 19,200 b/s.

Copyright 2006 - 2007

3.6.27 CMD_GET PAR_CYC_CTRL

Layout:

BOT	op_len	**CMD_GET**	op_ref	node_id	**payload**	EOT
0x00	0x05	**0x01**	X(1B)	X(2B)	**0x1E**	0x04

Synopsis:

PAR_CYC_CTRL (0x1E): This command gets the power management control parameters *state* and *mode.*

Specifics:

States:

Master: disabled (0x00), rest (0x01), sync (0x02). Disabled, rest and sync can be set by the user.

Field node: disabled (0x00), enabled (0x01), prep (0x02), sleep (0x03). Only disabled and enabled can be set by the user.

Modes:

Master: net (0x00). Master accepts net mode (0x00) only.
Field node: deep sleep (0x00), light sleep (0x01), awake (0x02), transmit off (0x03).

Factory Settings or Defaults:

State: disabled (0) on the master, enabled (1) on field nodes
Mode: 0

Valid Parameter Range:

As above

Power-up Value:

Retrieved from nonvolatile memory. At power up these parameters set to resume cycling as before the power failure.

Results Returned:

Results are always returned, in usual format, with added op_rc field:

BOT	op_len	**CMD_GET**	op_ref	op_rc	node_id	**payload**	EOT
0x00	0x08	**0x01**	X(1B)	X(1B)	X(2B)	**X(3B)**	0x04

Payload:

0x1E	B1(sleep state)	B2(sleep mode)

Error Codes:

None

Notes:

None

Copyright 2006 - 2007

Examples:

Get the master node power management state and mode:

```
00 05 01 AA 00 01 1E 04
```

parameter
0x1E

Returns:

```
00 08 01 AA 01 00 01 1E 00 00 04
```

parameter	B1(state)	B2(mode)
0x1E	0x00	0x00

The power management state is 0 (disabled).
The power management duty cycle mode is 0.

Get a field node power management state and mode:

```
00 05 01 AA 25 00 1E 04
```

parameter
0x1E

Returns:

```
00 08 01 AA 01 25 00 1E 01 00 04
```

parameter	B1(state)	B2(mode)
0x1E	0x01	0x00

The power management state is 1 (enabled).
The power management mode is 0.

Copyright 2006 - 2007

3.6.28 CMD_GET PAR_CYC_SLEEP

Layout:

BOT	op_len	**CMD_GET**	op_ref	node_id	**payload**	EOT
0x00	0x05	**0x01**	X(1B)	X(2B)	**0x1F**	0x04

Synopsis:

PAR_CYC_SLEEP (0x1F): This command gets the power management sleep period.

Specifics:

Field node only.

Factory Settings or Defaults:

90 seconds (good for testing, should be adjusted for each application)

Valid Parameter Range:

Such that master node's *rest* is in the range of 0x00000000 to 0xFFFFFFFF seconds.

Power-up Value:

Retrieved from nonvolatile memory.

Results Returned:

Results are always returned, in usual format, with added op_rc field:

BOT	op_len	**CMD_GET**	op_ref	op_rc	node_id	**payload**	EOT
0x00	0x0A	**0x01**	X(1B)	X(1B)	X(2B)	**X(5B)**	0x04

Payload:

0x1F	B1(period)	B2(period)	B3(period)	B4(period)

Error Codes:

RC_EMAS (0x04): attempt to query the master (sleep period does not apply to master)

Notes:

None

Examples:

Get the power management sleep period (on the remote node):

00 05 **01** AA 35 00 **1F** 04

parameter
0x1F

Returns:

00 0A 01 AA 01 35 00 1F **5A 00 00 00** 04

parameter	B1(period)	B2(period)	B3(period)	B4(period)
0x1F	0x5A	0x00	0x00	0x00

The power management sleep period is 90 seconds.

Copyright 2006 - 2007

3.6.29 CMD_GET PAR_CYC_M_SYNC

Layout:

BOT	op_len	**CMD_GET**	op_ref	node_id	**payload**	EOT
0x00	0x05	**0x01**	X(1B)	X(2B)	**0x20**	0x04

Synopsis:

PAR_CYC_M_SYNC (0x20): This command gets the master node power management synchronization period (or special mode).

Specifics:

Master node only. Synchronization period, or 0xDFFF or 0xEFFF

Factory Settings or Defaults:

60 seconds (good for testing, should be adjusted to suit application)

Valid Parameter Range:

0x0000 to 0x0FFF seconds, 0xDFFF, 0xEFFF

Power-up Value:

Retrieved from nonvolatile memory.

Results Returned:

Results are always returned, in usual format, with added op_rc field:

BOT	op_len	**CMD_GET**	op_ref	op_rc	node_id	**payload**	EOT
0x00	0x08	**0x01**	X(1B)	X(1B)	X(2B)	**X(3B)**	0x04

Payload:

0x20	B1(period)	B2(period)

Error Codes:

RC_EMAS (0x04): attempt to query a field node (master sync period applies only to masters)

Notes:

None

Examples:

Get the power management synchronization period (master node):

00 05 **01** AA 00 01 **20** 04

parameter
0x20

Returns:

00 08 01 AA 01 00 01 20 **3C 00** 04

parameter	B1(period)	B2(period)
0x20	0x3C	0x00

The master node synchronization period is 60 seconds (0x003C)

Copyright 2006 - 2007

3.6.30 CMD_GET PAR_CYC_M_REST

Layout:

BOT	op_len	**CMD_GET**	op_ref	node_id	**payload**	EOT
0x00	0x05	**0x01**	X(1B)	X(2B)	**0x21**	0x04

Synopsis:

PAR_CYC_M_REST (0x21): This command gets the power management master node rest period.

Specifics:

Master node only

Factory Settings or Defaults:

120 seconds (90 plus 30 seconds, good for testing, should be adjusted to suit application).

Valid Parameter Range:

0x00000000 to 0xFFFFFFFF

Power-up Value:

Retrieved from nonvolatile memory

Results Returned:

Results are always returned, in usual format, with added op_rc field:

BOT	op_len	**CMD_GET**	op_ref	op_rc	node_id	**payload**	EOT
0x00	0x0A	**0x01**	X(1B)	X(1B)	X(2B)	**X(5B)**	0x04

Payload:

0x21	B1(period)	B2(period)	B3(period)	B4(period)

Error Codes:

RC_EMAS (0x04): attempt to get from a field node (applies only to the master)

Notes:

None

Examples:

Get the power management master node rest period:

00 05 **01** AA 00 01 **21** 04

parameter
0x21

Returns:

00 0A 01 AA 01 00 01 21 **78 00 00 00** 04

parameter	B1(period)	B2(period)	B3(period)	B4(period)
0x21	0x78	0x00	0x00	0x00

The rest period is 120 seconds.

Copyright 2006 - 2007

3.6.31 CMD_GET ATTR_CYC_LEFT

Layout:

BOT	op_len	**CMD_GET**	op_ref	node_id	**payload**	EOT
0x00	0x05	**0x01**	X(1B)	X(2B)	**0x22**	0x04

Synopsis:

ATTR_CYC_LEFT (0x22): This command gets the power management (sleep cycle) current data.

Specifics:

Seconds left in the current state, state, mode

States:

Master: disabled (0x00), rest (0x01), sync (0x02)
Field node: disabled (0x00), enabled (0x01), prep (0x02), sleep (0x03)

Modes:

Master: net (0x00). Master accepts net mode (0x00) only.
Field node: deep sleep (0x00), light sleep (0x01), awake (0x02), transmit off (0x03)

Factory Settings or Defaults:

N/A

Valid Parameter Range:

N/A

Power-up Value:

N/A

Results Returned:

Results are always returned, in usual format, with added op_rc field:

BOT	op_len	CMD_GET	op_ref	op_rc	node_id	**payload**	EOT
0x00	0x0C	0x01	X(1B)	X(1B)	X(2B)	**X(7B)**	0x04

Payload:

0x22	B1(period)	B2(period)	B3(period)	B4(period)

...

B5(sleep state)	B6(sleep mode)

Error Codes:

None

Notes:

None

Copyright 2006 - 2007

Examples:

Base station sleep cycle data:

```
00 05 01 AA 00 01 22 04
```

payload

parameter
0x22

returns

```
00 0C 01 AA 01 00 01 22 29 00 00 00 02 00 04
```

payload

parameter	B1(period)	B2(period)	B3(period)
0x22	0x29	0x00	0x00

…

B4(period)	B5(state)	B6(mode)
0x00	0x02	0x00

```
B1-B4: 0x00000029 - Seconds left in the current state
B5:      0x02       - State (Sync)
B6:      0x00       - Mode
```

Base station sleep cycle data:

```
00 05 01 AA 00 01 22 04
```

payload

parameter
0x22

returns

```
00 0C 01 AA 01 00 01 22 75 00 00 00 01 00 04
```

payload

parameter	B1(period)	B2(period)	B3(period)
0x22	0x75	0x00	0x00

…

B4(period)	B5(state)	B6(mode)
0x00	0x01	0x00

```
B1-B4: 0x00000075 - Seconds left in the current state
B5:      0x01       - State (Rest)
B6:      0x00       - Mode
```


Copyright 2006 - 2007

Field node sleep cycle data:

```
00 05 01 AA 25 00 22 04
```

payload

parameter
0x22

returns

```
00 0C 01 AA 01 25 00 22 03 00 00 00 02 01 04
```

payload

parameter	B1(period)	B2(period)	B3(period)
0x22	0x03	0x00	0x00

…

B4(period)	B5(state)	B6(mode)
0x00	0x02	0x01

```
B1-B4: 0x00000003 - Seconds left in the current state
B5:      0x02       - State (Prep)
B6:      0x01       - Mode (light sleep)
```

Field node sleep cycle data:

```
00 05 01 AA 25 00 22 04
```

payload

parameter
0x22

Returns

```
00 0C 01 AA 01 25 00 22 0C 00 00 00 03 01 04
(radio off, serial port access only)
```

payload

parameter	B1(period)	B2(period)	B3(period)
0x22	0x0C	0x00	0x00

B4(period)	B5(state)	B6(mode)
0x00	0x03	0x01

```
B1-B4: 0x0000000C - Seconds left in the current state
B5:      0x03       - State (Sleep)
B6:      0x01       - Mode (light sleep)
```


Copyright 2006 - 2007

Field node sleep cycle data:

```
00 05 01 AA 25 00 22 04
```

payload

parameter
0x22

returns

```
00 0C 01 AA 01 25 00 22 00 00 00 00 01 01 04
```

payload

parameter	B1(period)	B2(period)	B3(period)
0x22	0x0C	0x00	0x00

…

B4(period)	B5(state)	B6(mode)
0x00	0x01	0x01

```
B1-B4: 0x00000000 – Counter inactive for this specific state.
B5:      0x01        - State (enabled)
B6:      0x01        - Mode (light sleep)
```


Copyright 2006 - 2007

3.6.32 CMD_GET PAR_CMD_MODE

Layout:

BOT	op_len	**CMD_GET**	op_ref	node_id	**payload**	EOT
0x00	0x06	**0x01**	X(1B)	X(2B)	**0x23**	0x04

Synopsis:

PAR_CMD_MODE (0x23): This command gets the serial port mode - command (1) or data mode (0).

Specifics:

None

Factory Settings or Defaults:

1 (command mode)

Valid Parameter Range:

0 or 1

Power-up Value:

Retrieved from nonvolatile memory.

Results Returned:

Results are always returned, in usual format, with added op_rc field:

BOT	op_len	**CMD_GET**	op_ref	op_rc	node_id	**payload**	EOT
0x00	0x07	**0x01**	X(1B)	X(1B)	X(2B)	**X(2B)**	0x04

Payload:

0x23	B1(serial port mode)

Error Codes:

None

Notes:

None

Examples:

Get the serial port mode:

```
00 05 01 33 25 00 23 04
```

parameter
0x23

Returns

```
00 07 01 33 01 25 00 23 01 04
```

parameter	B1(serial port mode)
0x23	0x01

The serial port is in command mode (1).

Copyright 2006 - 2007

3.7 SET Messages

3.7.1 CMD_SET PAR_LH

Layout:

BOT	op_len	**CMD_SET**	op_ref	node_id	**payload**	EOT
0x00	0x07	**0x02**	X(1B)	X(2B)	X(3B)	0x04

Payload:

0x01	B1(node_id)	B2(node_id)

Synopsis:

PAR_LH (0x01): This command sets the node ID of the target node.

Specifics:

None

Factory Settings or Defaults:

Lower word of ESN

Valid Parameter Range:

0x0001 to 0xFFFE (avoid 0x0000 or 0xFFFF)

Power-up Value:

Retrieved from nonvolatile memory.

Results Returned:

If a response is forced back (the most significant bit in the operation's reference code is set), the results are always returned, in usual format, with added op_rc field:

BOT	op_len	**CMD_SET**	op_ref	op_rc	node_id	**payload**	EOT
0x00	0x08	**0x02**	X(1B)	X(1B)	X(2B)	X(3B)	0x04

Payload:

0x01	B1(node_id)	B2(node_id)

with the payload as in the original command.

Error Codes:

RC_EVAL (**0x07)** error, bad value

Notes:

None

Copyright 2006 - 2007

Examples:

Set the node ID:

```
00 07 02 BB 25 00 01 45 00 04
```

parameter	B1(node_id)	B2(node_id)
0x01	0x45	0x00

Returns:

```
00 08 02 BB 00 25 00 01 45 00 04
```

parameter	B1(node_id)	B2(node_id)
0x01	0x45	0x00

The old node ID is 0x0025.
The new node ID is 0x0045.

Copyright 2006 - 2007

3.7.2 CMD_SET PAR_NID

Layout:

BOT	op_len	**CMD_SET**	op_ref	node_id	**payload**	EOT
0x00	0x07	**0x02**	X(1B)	X(2B)	X(3B)	0x04

Payload:

0x04	B1(net_id)	B2(net_id)

Synopsis:

PAR_NID (0x04): This command sets the 16-bit network ID.

Specifics:

This 16-bit ID is used to identify a node's specific network affiliation.

Factory Settings or Defaults:

0 (no network affiliation)

Valid Parameter Range:

0x0001 to 0xFFFE

Power-up Value:

Retrieved from nonvolatile memory.

Results Returned:

If a response is forced back (the most significant bit in the operation's reference code is set), the results are always returned, in usual format, with added op_rc field:

BOT	op_len	CMD_SET	op_ref	op_rc	node_id	**payload**	EOT
0x00	0x08	0x02	X(1B)	X(1B)	X(2B)	X(3B)	0x04

Payload:

0x04	B1(net_id)	B2(net_id)

with the payload as in the original command.

Error Codes:

None

Notes:

Setting the node's network ID to a non-zero value disables bind request messaging.

Copyright 2006 - 2007

Examples:

Set the network ID:

```
00 07 02 BB 00 01 04 10 00 04
```

parameter	B1(net_id)	B2(net_id)
0x04	0x10	0x00

Returns:

```
00 08 02 BB 00 00 01 04 10 00 04
```

parameter	B1(set_id)	B2(set_id)
0x04	0x10	0x00

The network ID is set to 0x0010.

Copyright 2006 - 2007

3.7.3 CMD_SET PAR_TARP_F

Layout:

BOT	op_len	**CMD_SET**	op_ref	node_id	**payload**	EOT
0x00	0x06	**0x02**	X(1B)	X(2B)	X(2B)	0x04

Payload:

0x08	B1(on/off)

Synopsis:

PAR_TARP_F (0x08): This command sets the TARP mesh routing rule *forwarding* parameter.

Specifics:

Forwarding ON/OFF

Factory settings, defaults

1 (ON)

Valid Parameter Range:

0 or 1; only least significant bit has meaning.

Power-up Value:

Retrieved from nonvolatile memory (routing is forced OFF on the master)

Results Returned:

If a response is forced back (the most significant bit in the operation's reference code is set), the results are always returned, in usual format, with added op_rc field:

BOT	Op_len	**CMD_SET**	op_ref	op_rc	node_id	**payload**	EOT
0x00	0x07	**0x02**	X(1B)	X(1B)	X(2B)	X(2B)	0x04

Payload:

0x08	B1(on/off)

with the payload as in the original command.

Error Codes:

None

Notes:

None

Copyright 2006 - 2007

Examples:

Set TARP forwarding:

```
00 06 02 CC 25 00 08 00 04
```

parameter	B1(on/off)
0x08	0x00

Returns:

```
00 07 02 CC 00 25 00 08 00 04
```

parameter	B1(on/off)
0x08	0x00

Forwarding is OFF.

Copyright 2006 - 2007

3.7.4 CMD_SET PAR_TARP_L

Layout:

BOT	op_len	**CMD_SET**	op_ref	node_id	**payload**	EOT
0x00	0x06	**0x02**	X(1B)	X(2B)	X(2B)	0x04

Payload:

0x05	B1(level)

Synopsis:

PAR_TARP_L (0x05): This command sets the TARP *level* mesh routing parameter.

Specifics:

TARP parameters: 1 byte (bit 0 is the least significant bit):

Bit 0: Forwarding ON/OFF
Bits 1 - 3: TARP slack (0 - 7)
Bits 4 - 5: TARP route recovery (0 - 3)
Bits 6 - 7: TARP level (0 - 2)

TARP levels:

Level 0 - Full flooding with discard on maximum hop count.
Level 1 - Duplicate Discards (DD) - when ON, duplicate packets are discarded.
Level 2 - Duplicate Discards plus Suboptimal Path Discards (SPD). If the packet's destination is known, the node discards packets on suboptimal paths. Number of discarded packets to a destination is kept to play a role in route recovery, together with the TARP *slack* and *route recovery* parameters.

Factory Settings or Defaults:

0xA3, or bit pattern **10** 10 001 1
Forwarding (bit 0) is 1 (ON)
TARP slack (bit1 - bit 3) is 001 (1)
TARP route recovery (bit 4 - bit 5) is 10 (2)
TARP level (bit 6 - bit 7) is 10 (2)

Valid Parameter Range:

Compound ranges

Power-up Value:

Retrieved from nonvolatile memory (routing is forced off at the master).

Copyright 2006 - 2007

Results Returned:

If a response is forced back (the most significant bit in the operation's reference code is set), the results are always returned, in usual format, with added op_rc field:

BOT	op_len	**CMD_SET**	op_ref	op_rc	node_id	**payload**	EOT
0x00	0x07	**0x02**	X(1B)	X(1B)	X(2B)	X(2B)	0x04

Payload:

0x05	B1(level)

with the payload as in the original command.

Error Codes:

None

Notes:

None

Examples:

Set the TARP level:

00 06 02 CC 25 00 **05 01** 04

parameter	B1(level)
0x05	0x01

Returns:

00 07 02 CC 00 25 00 **05 01** 04

parameter	B1(level)
0x05	0x01

The TARP level is 0x01.

Copyright 2006 - 2007

3.7.5 CMD_SET PAR_TARP_S

Layout:

BOT	op_len	**CMD_SET**	op_ref	node_id	**payload**	EOT
0x00	0x06	**0x02**	X(1B)	X(2B)	X(2B)	0x04

Payload:

0x06	B1(slack)

Synopsis:

PAR_TARP_S (0x06): This command sets the TARP *slack* mesh routing parameter.

Specifics:

TARP slack:

Suboptimal path discard will add the slack value when examining the caches, so suboptimal paths may become satisfactory for route recovery. The default value for slack (1) should not be changed without consulting Cirronet.

Factory Settings or Defaults:

1

Valid Parameter Range:

0 to 7, only the 3 least significant bits have meaning

Power-up Value:

Retrieved from nonvolatile memory (routing is forced OFF on the master).

Results Returned:

If a response is forced back (the most significant bit in the operation's reference code is set), the results are always returned, in usual format, with added op_rc field:

BOT	op_len	**CMD_SET**	op_ref	op_rc	node_id	**payload**	EOT
0x00	0x07	**0x02**	X(1B)	X(1B)	X(2B)	X(2B)	0x04

Payload:

0x06	B1(slack)

with the payload as in the original command.

Error Codes:

None

Notes:

None

Copyright 2006 - 2007

Examples:

Set the TARP slack:

```
00 06 02 CC 25 00 06 04 04
```

parameter	B1(slack)
0x06	0x04

Returns:

```
00 07 02 CC 00 25 00 06 04 04
```

parameter	B1(slack)
0x06	0x04

The TARP slack is set to 0x04.

Copyright 2006 - 2007

3.7.6 CMD_SET PAR_TARP_R

Layout:

BOT	op_len	**CMD_SET**	op_ref	node_id	**payload**	EOT
0x00	0x06	**0x02**	X(1B)	X(2B)	X(2B)	0x04

Payload:

0x07	B1(rte)

Synopsis:

PAR_TARP_R (0x07): This command sets the TARP mesh routing rule *route recovery speed* parameter.

Specifics:

TARP route recovery:

Suboptimal path discard exponentially weighs discards using the route recovery speed parameter:

0: additional hop for every discard
1: additional hop for every 4 discards
2: additional hop for every 8 discards
3: route discovery disabled (no discards counting)

The default value for route recovery (2) should not be changed without consulting Cirronet.

Factory Settings or Defaults:

2

Valid Parameter Range:

0 to 3, only 2 least significant bits have meaning

Power-up Value:

Retrieved from nonvolatile memory (routing is forced off at the master).

Results Returned

If a response is forced back (the most significant bit in the operation's reference code is set), the results are always returned, in usual format, with added op_rc field:

BOT	op_len	**CMD_SET**	op_ref	op_rc	node_id	**payload**	EOT
0x00	0x07	**0x02**	X(1B)	X(1B)	X(2B)	X(2B)	0x04

Payload:

0x07	B1(rte)

with the payload as in the original command.

Copyright 2006 - 2007

Error Codes:

None

Notes:

None

Examples:

Set the TARP route recovery:

```
00 06 02 CC 25 00 07 01 04
```

parameter	B1(par)
0x07	0x01

Returns:

```
00 07 02 CC 00 25 00 07 01 04
```

parameter	B1(par)
0x07	0x01

The TARP route recovery is set to 0x01.

Copyright 2006 - 2007

3.7.7 CMD_SET PAR_BEAC

Layout:

BOT	Op_len	**CMD_SET**	op_ref	node_id	**payload**	EOT
0x00	0x06	**0x02**	X(1B)	X(2B)	X(2B)	0x04

Payload:

0x09	B1(freq)

Synopsis:

PAR_BEAC (0x09): This command sets the beacon interval.

Specifics:

Beacon interval for the mater node
Bind request message interval for an unbound field node

Factory Settings or Defaults:

29 seconds for master node, 31 seconds for unbound node

Valid Parameter Range:

0 to 63, in seconds

Power-up Value:

Defaults retrieved from program memory:

- On unbound nodes: 31 seconds, (bind request message will be sent every 31 seconds).
- On the master: 29 seconds (master beacon will be sent every 29 seconds).
- Bound field nodes: 0 seconds (bind request messaging disabled).

Results Returned:

If a response is forced back (the most significant bit in the operation's reference code is set), the results are always returned, in usual format, with added op_rc field:

BOT	op_len	CMD_SET	op_ref	op_rc	node_id	**payload**	EOT
0x00	0x07	0x02	X(1B)	X(1B)	X(2B)	X(2B)	0x04

Payload:

0x09	B1(freq)

with the payload as in the original command.

Error Codes:

None

Notes:

None

Copyright 2006 - 2007

Examples:

Set the master beacon interval:

```
00 06 02 CC 00 01 09 1F 04
```

parameter	B1(freq)
0x09	0x1F

Returns:

```
00 07 02 CC 00 00 01 09 1F 04
```

parameter	B1(freq)
0x09	0x1F

The master beacon interval is set to 31 seconds (0x1F).

Copyright 2006 - 2007

3.7.8 CMD_SET PAR_CON

Layout:

BOT	op_len	**CMD_SET**	op_ref	node_id	**payload**	EOT
0x00	0x07	**0x02**	X(1B)	X(2B)	X(3B)	0x04

Payload:

0x0A	B1(freq)	B2(missed)

Synopsis:

PAR_CON (0x0A): This command sets the connectivity data (beacon interval, number of missed beacons to trigger a connectivity warning action, number of additional missed beacons to trigger a bad connectivity action, number of missed beacons in audit period).

Specifics:

Connectivity:

Byte 1: audit period in seconds

Byte 2:

bits 7 - 4: number of missed beacons to warning (start blinking connectivity LED),

bits 3 - 0: number of additional missed beacons to bad status (connectivity LED OFF)

Factory Settings or Defaults:

Defaults retrieved from program memory. See power-up value below.

Valid Parameter Range:

Audit period: 1 to 63 seconds, use 0 to disable audit

Missed beacons to warning (LED blinking), 0 to 15

Additional missed beacons to bad (LED OFF), 0 to 15

Power-up Value:

Defaults retrieved from program memory:

Connectivity:

Master node: audit interval equal to beacon interval (29 seconds)

Other nodes: audit interval is 0 (set by master beacon payload)

Missed beacons to warn: 3

Missed beacons to bad: 4

Copyright 2006 - 2007

Results Returned:

If a response is forced back (the most significant bit in the operation's reference code is set), the results are always returned, in usual format, with added op_rc field:

BOT	op_len	**CMD_SET**	op_ref	op_rc	node_id	**payload**	EOT
0x00	0x08	**0x02**	X(1B)	X(1B)	X(2B)	X(3B)	0x04

Payload:

0x0A	B1(freq)	B2(missed)

with the payload as in the original command.

Error Codes:

None

Notes:

None

Examples:

Set the connectivity data:

00 07 02 CC 00 01 **0A 1E 34** 04

parameter	B1(freq)	B2(missed)
0x0A	0x1E	0x34

Returns:

00 08 02 CC 00 00 01 **0A 1E 34** 04

parameter	B1(freq)	B2(missed)
0x0A	0x1E	0x34

The audit interval is 0x1E (30 seconds).
The number of missed beacons to warning is 3.
The number of additional missed beacons to bad status is 4.

Copyright 2006 - 2007

3.7.9 CMD_SET PAR_RF

Layout:

BOT	op_len	**CMD_SET**	op_ref	node_id	**payload**	EOT
0x00	0x06	**0x02**	X(1B)	X(2B)	X(2B)	0x04

Payload:

0x0B	B1(status)

Synopsis:

PAR_RF (0x0B): This command sets the Radio ON/OFF status (TX, RX).

Specifics:

TX is ON when Bit 1 is set, RX is ON when Bit 0 is set

Factory Settings or Defaults:

3 (TX ON and RX ON)

Valid Parameter Range:

0 to 3

Power-up Value:

Default retrieved from program memory.

Results Returned:

If a response is forced back (the most significant bit in the operation's reference code is set), the results are always returned, in usual format, with added op_rc field:

BOT	op_len	**CMD_SET**	op_ref	op_rc	node_id	**payload**	EOT
0x00	0x07	**0x02**	X(1B)	X(1B)	X(2B)	X(2B)	0x04

Payload:

0x0B	B1(status)

with the payload as in the original command.

Error Codes:

None

Notes:

RF status is also controlled by power management cycles.

Copyright 2006 - 2007

Examples:

Set TX and RX ON/OFF parameters:

```
00 06 02 A1 25 00 0B 02 04
```

parameter	B1(status)
0x0B	0x02

Returns:

```
00 07 02 A1 00 25 00 0B 02 04
```

parameter	B1(status)
0x0B	0x02

0x02, or bit pattern 10
The transmitter is ON.
The receiver is OFF.

Set TX and RX ON/OFF parameters:

```
00 06 02 A1 25 00 0B 03 04
```

parameter	B1(status)
0x0B	0x03

Returns:

```
00 07 02 A1 00 25 00 0B 03 04
```

parameter	B1(status)
0x0B	0x03

0x03, or bit pattern 11
The transmitter is ON.
The receiver is ON.

Copyright 2006 - 2007

3.7.10 CMD_SET PAR_STAT_REP

Layout:

BOT	op_len	**CMD_SET**	op_ref	node_id	**payload**	EOT
0x00	0x08	**0x02**	X(1B)	X(2B)	X(4B)	0x04

Payload:

0x0D	B1(freq)	B2(freq)	B3(report_par)

Synopsis:

PAR_STAT_REP (0x0D): This command sets the status reporting interval.

Specifics:

Byte 1 - 2: interval in 64 second increments
Byte 3: not used on the DM2200

Factory Settings or Defaults:

450 increments, or 8 hours (450*64 = 28,800 seconds = 8 hours)

Valid Parameter Range:

0x0000 to 0xFFFF

Power-up Value:

Default retrieved from program memory.

Results Returned:

If a response is forced back (the most significant bit in the operation's reference code is set), the results are always returned, in usual format, with added op_rc field:

BOT	op_len	**CMD_SET**	op_ref	node_id	**payload**	EOT
0x00	0x08	**0x02**	X(1B)	X(2B)	X(4B)	0x04

Payload:

0x0D	B1(freq)	B2(freq)	B3(report_par)

with the payload as in the original command.

Error Codes:

None

Notes:

None

Copyright 2006 - 2007

Examples:

Set the status reporting interval:

```
00 08 02 AA 25 00 0D 04 00 01 04
```

parameter	B1(freq)	B2(freq)	B3(report_par)
0x0D	0x04	0x00	0x00

Returns:

```
00 09 02 AA 00 25 00 0D 04 00 01 04
```

parameter	B1(freq)	B2(freq)	B3(report_par)
0x0D	0x04	0x00	0x00

The interval for status reporting is 4 increments of 64 seconds.

After setting a new status reporting interval a *msg_br* status message is sent from the node, and the message output appears:

```
00 07 FF 08 25 00 00 34 01 04
```

B1(node_id)	B2(node_id)	B3(connectivity)	B4(connectivity)	B5(autoACK)
0x25	0x00	0x00	0x34	0x01

B1-B2: 0x0025 - node ID
B3-B4: 0x3400 - connectivity (missed beacons to warning, missed beacons to bad status, missed bits)
B3: 0x00 - missed bits
B4: (bits 7-4, bits 3-0)
bits 7-4: **3** - missed beacons to warning
bits 3-0: **4** - missed beacons to bad status
B5: 0x01 - autoACK

Copyright 2006 - 2007

3.7.11 CMD_SET PAR_RETRY

Layout:

BOT	op_len	**CMD_SET**	op_ref	node_id	**payload**	EOT
0x00	0x07	**0x02**	X(1B)	X(2B)	X(3B)	0x04

Payload:

0x0E	B1(retry)	B2(timeout)

Synopsis:

PAR_RETRY (0x0E): This command sets the reliable message reporting parameters - the number of retries and the retry timeout period.

Specifics:

None

Factory Settings or Defaults:

Number of retries: 3
Timeout period: 0x0A (10 seconds)

Valid Parameter Range:

Number of retries: 0 to 15
Timeout period: 0 to 15 seconds

Power-up Value:

Default retrieved from program memory.

Results Returned:

If a response is forced back (the most significant bit in the operation's reference code is set), the results are always returned, in usual format, with added op_rc field:

BOT	op_len	**CMD_SET**	op_ref	op_rc	node_id	**payload**	EOT
0x00	0x07	**0x02**	X(1B)	X(1B)	X(2B)	X(3B)	0x04

Payload:

0x0E	B1(retry)	B2(timeout)

with the payload as in the original command.

Error Codes:

RC_ELEN (**0x08):** error, bad length

Notes:

None

Copyright 2006 - 2007

Examples:

Set the number of retries and the timeout period in seconds:

```
00 07 02 AA 25 00 0E 05 0F 04
```

parameter	B1(retry)	B2(timeout)
0x0E	0x05	0x0F

Returns:

```
00 08 02 AA 00 25 00 0E 05 0F 04
```

parameter	B1(retry)	B2(timeout)
0x0E	0x05	0x0F

The number of retries is set to 5.
The timeout period in seconds is set to 15.

Copyright 2006 - 2007

3.7.12 CMD_SET PAR_AUTOACK

Layout:

BOT	op_len	**CMD_SET**	op_ref	node_id	**payload**	EOT
0x00	0x06	**0x02**	X(1B)	X(2B)	X(2B)	0x04

Payload:

0x0F	B1(ACK)

Synopsis:

PAR_AUTOACK (0x0F): This command sets the auto-acknowledgment ON/OFF.

Specifics:

None

Factory Settings or Defaults:

1 (ON)

Valid Parameter Range:

0 (OFF) or 1 (ON)

Power-up Value:

Default retrieved from program memory.

Results Returned:

If a response is forced back (the most significant bit in the operation's reference code is set), the results are always returned, in usual format, with added op_rc field:

BOT	op_len	**CMD_SET**	op_ref	op_rc	node_id	**payload**	EOT
0x00	0x07	**0x02**	X(1B)	X(1B)	X(2B)	X(2B)	0x04

Payload:

0x0F	B1(ACK)

with the payload as in the original command.

Error Codes:

None

Notes:

Usually a master node parameter.

Copyright 2006 - 2007

Examples:

Set auto-acknowledgment OFF:

```
00 06 02 CC 00 01 0F 00 04
```

parameter	B1(ACK)
0x0F	0x00

Returns:

```
00 07 02 CC 00 00 01 0F 00 04
```

parameter	B1(ACK)
0x0F	0x00

Auto-acknowledgment is set to OFF.

Copyright 2006 - 2007

3.7.13 CMD_SET PAR_ENCR_M

Layout:

BOT	op_len	**CMD_SET**	op_ref	node_id	**payload**	EOT
0x00	0x06	**0x02**	X(1B)	X(2B)	X(2B)	0x04

Payload:

0x17	B1(mode)

Synopsis:

PAR_ENCR_M (0x17): This command sets the encryption mode.

Specifics:

Encryption mode 0 (free) or 1 (strict)

Factory Settings or Defaults:

0 (free)

Valid Parameter Range:

0 or 1 (mode)

Power-up Value:

Retrieved from nonvolatile memory.

Results Returned:

If a response is forced back (the most significant bit in the operation's reference code is set), the results are always returned, in usual format, with added op_rc field:

BOT	op_len	**CMD_SET**	op_ref	op_rc	node_id	**payload**	EOT
0x00	0x07	**0x02**	X(1B)	X(1B)	X(2B)	X(2B)	0x04

Payload:

0x17	B1(mode)

with the payload as in the original command.

Error Codes:

None

Notes:

None

Copyright 2006 - 2007

Examples:

Set the encryption mode to strict:

```
00 06 02 AA 00 01 17 01 04
```

parameter	B1(mode)
0x17	0x01

Returns:

```
00 07 02 AA 00 00 01 17 01 04
```

parameter	B1(mode)
0x17	0x01

The encryption mode is set to strict (1).

Copyright 2006 - 2007

3.7.14 CMD_SET PAR_ENCR_K

Layout:

BOT	op_len	**CMD_SET**	op_ref	node_id	**payload**	EOT
0x00	0x06	**0x02**	X(1B)	X(2B)	X(2B)	0x04

Payload:

0x18	B1(key)

Synopsis:

PAR_ENCR_K (0x18): This command sets the encryption key number.

Specifics:

None

Factory Settings or Defaults:

0 (no encryption)

Valid Parameter Range:

Key: 0 - 3

Power-up Value:

Retrieved from nonvolatile memory.

Results Returned:

If a response is forced back (the most significant bit in the operation's reference code is set), the results are always returned, in usual format, with added op_rc field:

BOT	op_len	**CMD_SET**	op_ref	op_rc	node_id	**payload**	EOT
0x00	0x07	**0x02**	X(1B)	X(1B)	X(2B)	X(2B)	0x04

Payload:

0x18	B1(key)

with the payload as in the original command.

Error Codes:

None

Notes:

None

Copyright 2006 - 2007

Examples:

Set the encryption key to 2:

```
00 06 02 CC 00 01 18 01 04
```

parameter	B1(key)
0x18	0x02

Returns:

```
00 07 02 CC 00 00 01 18 01 04
```

parameter	B1(key)
0x18	0x02

The encryption key is 2.

Copyright 2006 - 2007

3.7.15 CMD_SET PAR_ENCR_D

Layout:

BOT	op_len	**CMD_SET**	op_ref	node_id	**payload**	EOT
0x00	0x06	**0x02**	X(1B)	X(2B)	X(2B)	0x04

Payload:

0x19	B1(data)

Synopsis:

PAR_ENCR_D (0x19): This command sets the encryption data (mode, key).

Specifics:

Encryption data:

Bits 1 - 0: key number

Bit 2: mode

Factory Settings or Defaults:

Encryption data:

Key number: 0 (no encryption)

Mode: 0 (*free*)

Valid Parameter Range:

Key: 0 - 3

Mode: 0 or 1

Power-up Value:

Retrieved from nonvolatile memory.

Results Returned:

If a response is forced back (the most significant bit in the operation's reference code is set), the results are always returned, in usual format, with added op_rc field:

BOT	op_len	**CMD_SET**	op_ref	op_rc	node_id	**payload**	EOT
0x00	0x07	**0x02**	X(1B)	X(1B)	X(2B)	X(2B)	0x04

Payload:

0x19	B1(data)

with the payload as in the original command.

Error Codes:

None

Notes:

None

Copyright 2006 - 2007

Examples:

Set the encryption data (mode, key numbering):

```
00 06 02 CC 00 01 19 05 04
```

parameter	B1(data)
0x19	0x05

Returns:

```
00 07 02 CC 00 00 01 19 05 04
```

parameter	B1(data)
0x19	0x05

The encryption data is set to 0x05 (binary 101).
The key number is 1, and the mode is strict (1).

Copyright 2006 - 2007

3.7.16 CMD_SET PAR_BINDER

Layout:

BOT	op_len	**CMD_SET**	op_ref	node_id	**payload**	EOT
0x00	0x07	**0x02**	X(1B)	X(2B)	X(2B)	0x04

Payload:

0x1C	B1(can\not)

Synopsis:

PAR_BINDER (0x1C): This command sets the bind enable flag state.

Specifics:

ON/OFF: when OFF, the node will not forward bind request messages from unbound nodes.

Factory Settings or Defaults:

ON (1)

Valid Parameter Range:

0 or 1

Power-up Value:

Retrieved from nonvolatile memory.

Results Returned:

If a response is forced back (the most significant bit in the operation's reference code is set), the results are always returned, in usual format, with added op_rc field:

BOT	op_len	**CMD_SET**	op_ref	op_rc	node_id	**payload**	EOT
0x00	0x08	**0x02**	X(1B)	X(1B)	X(2B)	X(3B)	0x04

Payload:

0x1C	B1(ON/OFF)

with the payload as in the original command.

Error Codes:

None

Notes:

None

Copyright 2006 - 2007

Examples:

Set the bind enable flag:

```
00 06 02 AA 25 00 1C 01 04
```

parameter	B1(can/not)
0x1C	0x01

Returns:

```
00 07 02 AA 00 25 00 1C 01 04
```

parameter	B1(can/not)
0x1C	0x00

The bind enable flag is set (1).

Copyright 2006 - 2007

3.7.17 CMD_SET PAR_UART

Layout:

BOT	op_len	**CMD_SET**	op_ref	node_id	**payload**	EOT
0x00	0x06	**0x02**	X(1B)	X(2B)	X(2B)	0x04

Payload:

0x1D	B2(speed)	B2(speed)

Synopsis:

PAR_UART (0x1D): This command sets the UART baud rate.

Specifics:

Value represents 1/100 of the UART baud rate, for example, 192 represents 19,200 b/s. Note that the UART configuration is always N, 8, 1.

Factory Settings or Defaults:

192

Valid Parameter Range:

12, 24, 48, 96, 192, 384

Power-up Value:

Default retrieved from program memory.

Results Returned:

If a response is forced back (the most significant bit in the operation's reference code is set), the results are always returned, in usual format, with added op_rc field:

BOT	op_len	**CMD_SET**	op_ref	op_rc	node_id	**payload**	EOT
0x00	0x08	**0x02**	X(1B)	X(1B)	X(2B)	X(3B)	0x04

Payload:

0x1D	B1(speed)	B2(speed)

with the payload as in the original command.

Error Codes:

RC_EVAL **(0x07):** error, bad parameter value

Notes:

None

Copyright 2006 - 2007

Examples:

Set the UART baud rate to 1200 b/s:

```
00 07 02 BB 00 01 1D 0C 00 04
```

parameter	B1(speed)	B2(speed)
0x1D	0x0C	0x00

Returns:

```
00 08 02 BB 00 00 01 1D 0C 00 04
```

parameter	B1(speed)	B2(speed)
0x1D	0x0C	0x00

The UART baud rate is set to 1200 b/s (0x0C).

Copyright 2006 - 2007

3.7.18 CMD_SET PAR_CYC_CTRL

Layout:

BOT	op_len	**CMD_SET**	op_ref	node_id	**payload**	EOT
0x00	0x07	**0x02**	X(1B)	X(2B)	X(3B)	0x04

Payload:

0x1E	B1(state)	B2(mode)

Synopsis:

PAR_CYC_CTRL (0x1E): This command sets the power management control parameters *state* and *mode.*

Specifics:

States:

Master: disabled (0x00), rest (0x01), sync (0x02). Disabled, rest and sync state can be set by the user.

Field node: disabled (0x00), enabled (0x01), prep (0x02), sleep (0x03). Only disabled or enabled can be set by the user.

Modes:

Master: net (0x00).
Master only accepts net mode.

Field node: deep sleep (0x00), light sleep (0x01), awake (0x02), transmit only (0x03).

Factory Settings or Defaults:

State: disabled (0) on the master, enabled (1) on field nodes
Mode: 0

Valid Parameter Range:

As above

Power-up Value:

Retrieved from nonvolatile memory. At power up, depending on the context, they're set to safe values, ready to continue cycling as before the power failure.

Copyright 2006 - 2007

Results Returned:

If a response is forced back (the most significant bit in the operation's reference code is set), the results are always returned, in usual format, with added op_rc field:

BOT	op_len	**CMD_SET**	op_ref	op_rc	node_id	**payload**	EOT
0x00	0x08	**0x02**	X(1B)	X(1B)	X(2B)	X(3B)	0x04

Payload:

0x1E	B1(state)	B2(mode)

with the payload as in the original command.

Error Codes:

RC_ERES **(0x02):** error, resources:
- Master: either the old or the new state must be *disabled*
- Field node: mode can only be changed in the *disabled* state
- No change is allowed in *sleep* or *prep* state

RC_EMAS **(0x04**): error, master context:
- Master: attempt to change mode
- Attempt to change state to *sleep*

RC_EVAL **(0x07):** error, bad value:
- All: either state or mode greater than 3
- Field node: illogical combination of state and mode

Notes:

None

Examples:

Set duty cycle state and mode:
(currently node is in enabled state)

```
00 07 02 AA 00 01 1E 02 00 04
```

payload

parameter	B1(state)	B2(mode)
0x1E	0x02	0x00

returns:

payload

parameter	B1(state)	B2(mode)
0x1E	0x02	0x00

```
B1: 0x02 - State (Sync)
B2: 0x00 - Mode (net)
```


Copyright 2006 - 2007

Set duty cycle state and mode:
(currently node is in enabled state)

```
00 07 02 AA 25 00 1E 00 00 04
```

payload

parameter	B1(state)	B2(mode)
0x1E	0x00	0x00

returns

payload

parameter	B1(state)	B2(mode)
0x1E	0x00	0x00

```
B1: 0x00 - State (Disable)
B2: 0x00 - Mode (net)
```


Copyright 2006 - 2007

3.7.19 CMD_SET PAR_CYC_SLEEP

Layout:

BOT	op_len	**CMD_SET**	op_ref	node_id	**payload**	EOT
0x00	0x09	**0x02**	X(1B)	X(2B)	X(5B)	0x04

Payload:

0x1F	B1(period)	B2(period)	B3(period)	B4(period)

Synopsis:

PAR_CYC_SLEEP (0x1F): This command sets the power management sleep period.

Specifics:

Set the power management sleep period in seconds.
To set sleep period, node must be in the disabled state.
PAR_CYC_SLEEP is not applicable to the master node.

Factory Settings or Defaults:

90 seconds (good for testing, should be adjusted to suit the application)

Valid Parameter Range:

Such that master node's *rest* is in the range of 0x00000000 to 0xFFFFFFFF seconds.

Power-up Value:

Retrieved from nonvolatile memory.

Results Returned:

If a response is forced back (the most significant bit in the operation's reference code is set), the results are always returned, in usual format, with added op_rc field:

BOT	op_len	**CMD_SET**	op_ref	op_rc	node_id	**payload**	EOT
0x00	0x0A	**0x02**	X(1B)	X(1B)	X(2B)	X(5B)	0x04

Payload:

0x1F	B1(period)	B2(period)	B3(period)	B4(period)

with the payload as in the original command.

Error Codes:

RC_EMAS **(0x04):** error - master
RC_ERES **(0x02):** error - only allowed when in the disabled state

Notes:

None

Copyright 2006 - 2007

Examples:

Set the power management sleep period:

```
00 09 02 AA 25 00 1F 78 00 00 00 04
```

parameter	B1(period)	B2(period)	B3(period)	B4(period)
0x1F	0x78	0x00	0x00	0x00

Returns:

```
00 0A 02 AA 00 25 00 1F 78 00 00 00 04
```

parameter	B1(period)	B2(period)	B3(period)	B4(period)
0x1F	0x78	0x00	0x00	0x00

The sleep period is 0x00000078 (120 seconds).

Copyright 2006 - 2007

3.7.20 CMD_SET PAR_CYC_M_SYNC

Layout:

BOT	op_len	**CMD_SET**	op_ref	node_id	**payload**	EOT
0x00	0x07	**0x02**	X(1B)	X(2B)	X(3B)	0x04

Payload:

0x20	B1(par)	B2(par)

Synopsis:

PAR_CYC_M_SYNC (0x20): This command sets the power management master node synchronization period, plus special modes.

Specifics:

Sets the synchronization period in seconds, or modes 0xDFFF and 0xEFFF
PAR_CYC _M_SYNC can be set to force the network to a desired state.
0xEFFF moves all listeners to the *enabled* state, 0xDFFF to the *disabled* state.
PAR_CYC _M_SYNC is a master node only parameter.

Factory Settings or Defaults:

60 seconds (good for testing, should be adjusted to suit the application)

Valid Parameter Range:

0x0000 to 0x0FFF seconds, plus special modes 0xDFFF and 0xEFFF

Power-up Value:

Retrieved from nonvolatile memory.

Results Returned:

If a response is forced back (the most significant bit in the operation's reference code is set), the results are always returned, in usual format, with added op_rc field:

BOT	op_len	**CMD_SET**	op_ref	op_rc	node_id	**payload**	EOT
0x00	0x08	**0x02**	X(1B)	X(1B)	X(2B)	X(3B)	0x04

Payload:

0x20	B1(par)	B2(par)

with the payload as in the original command.

Error Codes:

RC_EMAS **(0x04):** error - not a master
RC_ERES **(0x02):** error - in synchronization state, time cannot be changed (special values 0xDFFF and 0xEFFF are allowed)
RC_EVAL **(0x07):** error - bad value (greater than 0x0FFF, but not 0xDFFF or 0xEFFF)

Copyright 2006 - 2007

Notes:

None

Examples:

Set duty cycle synchronization period:
(master only, master must be in disabled state)

```
00 07 02 AA 00 01 20 FF EF 04
```

payload

parameter	B1(par)	B2(par)
0x20	0xFF	0xEF

returns

```
00 08 02 AA 00 00 01 20 FF EF 04
```

payload

parameter	B1(par)	B2(par)
0x20	0xFF	0xEF

B1-B2: 0xEFFF - Duty cycle (moves all listeners to *enabled* state).

Set duty cycle synchronization to specific mode:
(master only, master must be in disabled state)

```
00 07 02 AA 00 01 20 FF DF 04
```

payload

parameter	B1(par)	B2(par)
0x20	0xFF	0xDF

returns

```
00 08 02 AA 00 00 01 20 FF DF 04
```

payload

parameter	B1(par)	B2(par)
0x20	0xFF	0xDF

B1-B2: 0xDFFF - Duty cycle: (moves all listeners to *disabled* state).

Copyright 2006 - 2007

Set duty cycle synchronization period:
(master only, master must be in disabled state)

```
00 07 02 AA 00 01 20 28 00 04
```

payload

parameter	B1(par)	B2(par)
0x20	0x28	0x00

returns

```
00 08 02 AA 00 00 01 20 28 00 04
```

payload

parameter	B1(par)	B2(par)
0x20	0x28	0x00

```
B1-B2: 0x0028 - Duty cycle, synchronization period 40 seconds.
```


Copyright 2006 - 2007

3.7.21 CMD_SET PAR_CYC_M_REST

Layout:

BOT	op_len	**CMD_SET**	op_ref	node_id	**payload**	EOT
0x00	0x09	**0x02**	X(1B)	X(2B)	X(5B)	0x04

Payload:

0x21	B1(par)	B2(par)	B3(par)	B4(par)

Synopsis:

PAR_CYC_M_REST (0x21): This command sets the power management rest period.

Specifics:

Master only

Factory Settings or Defaults:

120 second (90 plus 30 seconds, good for testing, adjust to suit application)

Valid Parameter Range:

0x00000000 to 0xFFFFFFFF

Power-up Value:

Retrieved from nonvolatile memory.

Results Returned:

If a response is forced back (the most significant bit in the operation's reference code is set), the results are always returned, in usual format, with added op_rc field:

BOT	op_len	CMD_SET	op_ref	op_rc	node_id	**payload**	EOT
0x00	0x0A	0x02	X(1B)	X(1B)	X(2B)	X(5B)	0x04

Payload:

0x21	B1(par)	B2(par)	B3(par)	B4(par)

with the payload as in the original command.

Error Codes:

RC_EMAS (**0x04**)**:** error - not a master

Notes:

Rest is a master node parameter.

Copyright 2006 - 2007

Examples:

Set the master node rest period:

```
00 09 02 AA 00 01 21 68 01 00 00 04
```

parameter	B1(par)	B2(par)	B3(par)	B4(par)
0x21	0x68	0x01	0x00	0x00

Returns:

```
00 0A 02 AA 00 00 01 21 68 01 00 00 04
```

parameter	B1(par)	B2(par)	B3(par)	B4(par)
0x21	0x68	0x01	0x00	0x00

The rest period is 0x00000168 (360 seconds).

Copyright 2006 - 2007

3.7.22 CMD_SET PAR_CMD_MODE

Layout:

BOT	op_len	**CMD_SET**	op_ref	node_id	**payload**	EOT
0x00	0x06	**0x02**	X(1B)	X(2B)	X(2B)	0x04

Payload:

0x23	B1(mode)

Synopsis:

PAR_CMD_MODE (0x23): This command sets the serial port to command mode (1), or data mode (0).

Specifics:

None

Factory Settings or Defaults:

1 (command mode)

Valid Parameter Range:

0 or 1

Power-up Value:

Retrieved from nonvolatile memory

Results Returned:

If a response is forced back (the most significant bit in the operation's reference code is set), the results are always returned, in usual format, with added op_rc field:

BOT	op_len	CMD_SET	op_ref	op_rc	node_id	**payload**	EOT
0x00	0x07	0x02	X(1B)	X(1B)	X(2B)	X(2B)	0x04

Payload:

0x23	B1(mode)

with the payload as in the original command.

Error Codes:

RC_EMAS (**0x04):** error, master always in command mode.

Notes:

None

Copyright 2006 - 2007

Examples:

Set the serial port mode:

```
00 06 02 A3 25 00 23 00 04
```

parameter	B1(mode)
0x23	0x00

Returns:

```
00 07 02 A3 00 25 00 23 00 04
```

parameter	B1(mode)
0x23	0x00

The serial port is in data mode (0x00).

Copyright 2006 - 2007

3.8 I/O Messages

3.8.1 CMD_IO IO_READ

Layout:

BOT	op_len	CMD_IO	op_ref	node_id	**payload**	EOT
0x00	0x08	**0x0B**	X(1B)	X(2B)	X(4B)	0x04

Payload:

0x00	B1(pin)	B2(bits b2-b0)	X(1B)

Synopsis:

IO_READ (0x00): This command reads the value or configuration of the pin designated in B1, and the result is returned in B2, bits b2 - b0:

Configuration	Bit b2	Bit b1	Bit b0
Analog Input	1	0	0
Digital Output High	0	1	1
Digital Output Low	0	1	0
Digital Input Event High*	1	1	1
Digital Input Event Low*	1	1	0
Digital Input High	0	0	1
Digital Input Low	0	0	0

* Interrupt events apply to GP1 and GP2 only.

Specifics:

A bulk read is performed if B1 is set to 0xFF. All 4 bytes of the response are used to show all 10 pins, encoded 3 bits per pin. The 4 bytes should be treated as a little endian long word (32 bits). See the last example below for the details of a bulk IO_READ. **X(1B)** - the value for this byte is not used.

If the defaults are changed, the new configuration is held.

Factory Settings or Defaults:

All pins default to factory settings. See 2.12.1, p. 25.

Valid Parameter Range:

Allowed pin numbers, or 0xFF for a bulk read.

Power-up Value:

See above

Results Returned:

Results are returned, in usual format, with added op_rc field:

BOT	op_len	CMD_IO	op_ref	op_rc	node_id	**payload**	EOT
0x00	0x09	**0x0B**	X(1B)	X(1B)	X(2B)	X(4B)	0x04

Payload:

0x00	B1(pin)	B2(value)	X(1B)

with the payload as in *Synopsis*. Bulk read uses all 4 bytes, modifying the original

Copyright 2006 - 2007

command.

Error Codes:

RC_EVAL **(0x07):** pin number is 0 or greater than 10, or B1 is not 0xFF (for bulk read)

Notes:

None

Examples:

Read the CFG2 (pin 10):

```
00 08 0B CC 25 00 00 0A FF FF 04
```

operation	B1(pin)	X(1B)	X(1B)
0x00	0x0A	0xFF	0xFF

Note the values in X(1B) are arbitrary.

Returns:

```
00 09 0B CC 01 25 00 00 0A 01 FF 04
```

operation	B1(pin)	B2(value)	X(1B)
0x00	0x0A	0x01	0xFF

The result is returned in B2. CFG2 is a digital input and the input is high.

Bulk read:

```
00 08 0B CC 35 00 00 FF FF FF 04
```

operation	B1(pin)	X (1B)	X(1B)
0x00	0xFF	0xFF	0xFF

Returns:

```
00 09 0B CC 01 35 00 40 36 32 09 04
```

A bulk read uses all 4 payload bytes to show data on the 10 I/O pins.

B1(value)	B2(value)	B3(value)	B4(value)
0x40	0x36	0x32	0x09

```
B1= 0x40 = 01000000
B2= 0x36 = 00110110
B3= 0x32 = 00110010
B4= 0x09 = 00001001
```

as little endian long word:

```
00001001 00110010 00110110 01000000

00 001 001 001 100 100 011 011 001 000 000

Pin 1:  000 (0x00) digital input
Pin 2:  000 (0x00) digital input
Pin 3:  001 (0x00) digital input
Pin 4:  011 (0x03) digital output high
Pin 5:  011 (0x03) digital output high
Pin 6:  100 (0x04) analog input
Pin 7:  100 (0x04) analog input
Pin 8:  001 (0x01) digital input
Pin 9:  001 (0x01) digital input
```


Copyright 2006 - 2007

```
Pin 10:  001(0x01) digital input
```

Combined in a long word:
00**011**011**011**011**011**011**011**011**011**011

Ten pins, all digital output high.

Read individual pin (at default configuration) – XX pin number.
00 08 0B CC 25 00 00 **XX** FF FF 04

operation	B1(pin)	X(1B))	X(1B)
0x00	0xXX	0xFF	0xFF

returns:

```
for pin 1
00 09 0B CC 01 25 00 00 01 00 FF 04

for pin 2
00 09 0B CC 01 25 00 00 02 00 FF 04

for pin 3
00 09 0B CC 01 25 00 00 03 00 FF 04

for pin 4
00 09 0B CC 01 25 00 00 04 03 FF 04

for pin 5
00 09 0B CC 01 25 00 00 05 03 FF 04

for pin 6
00 09 0B CC 01 25 00 00 06 04 FF 04

for pin 7
00 09 0B CC 01 25 00 00 07 04 FF 04

for pin 8
00 09 0B CC 01 25 00 00 08 01 FF 04

for pin 9
00 09 0B CC 01 25 00 00 09 01 FF 04

for pin 10
00 09 0B CC 01 25 00 00 0A 01 FF 04
```


Copyright 2006 - 2007

3.8.2 CMD_IO IO_WRITE

Layout:

BOT	op_len	CMD_IO	op_ref	node_id	**payload**	EOT
0x00	0x08	**0x0B**	X(1B)	X(2B)	X(4B)	0x04

Payload:

0x01	B1(pin)	B2(bits b2-b0)	X(1B)

Synopsis:

IO_WRITE (0x01): This command configures and/or writes to the pin designated in B1 according to the flag values in B2, bits b2 - b0:

Configure and/or Write	Bit b2	Bit b1	Bit b0
Analog Input	1	0	0
Digital Input	0	1	0
Digital Output, Write High	0	0	1
Digital Output, Write Low	0	0	0

Specifics:

X(1B) - the value in this byte is unused.

Factory Settings or Defaults:

N/A

Valid Parameter Range:

Valid pin numbers

Power-up Value:

N/A

Results Returned:

Results are always returned, in usual format, with added op_rc field:

BOT	op_len	CMD_IO	op_ref	op_rc	node_id	**payload**	EOT
0x00	0x09	**0x0B**	X(1B)	X(1B)	X(2B)	X(4B)	0x04

Payload:

0x01	B1(pin)	B2(val)	X(1B)

with the payload as in the original command.

Error Codes:

RC_EVAL **(0x07):** pin number is 0 or greater than 10, or pin greater than 7 is being configured analog input.

RC_ERES **(0x02):** attempt to configure interrupt-driven GP1 or GP2.

Copyright 2006 - 2007

Notes:

The IO_WRITE command cannot override a pulse counter configuration on GP1 or a digital input with event messaging configuration on GP2.

Examples:

Set pin GP2 as a digital output low:

```
00 08 0B CC 25 00 01 02 00 FF 04
```

operation	B1(pin)	B2(val)	X(1B)
0x01	0x02	0x00	0xFF

Returns:

```
00 09 0B CC 01 25 00 01 02 00 FF 04
```

operation	B1(pin)	B2(val)	X(1B)
0x01	0x02	0x00	0xFF

If event messaging on GP2 is not running.

Configure GP1 as an analog input:

```
00 08 0B CC 25 00 01 01 04 FF 04
```

operation	B1(pin)	B2(val)	X(1B)
0x01	0x01	0x04	0xFF

Returns:

```
00 09 0B CC 01 25 00 01 01 04 FF 04
```

operation	B1(pin)	B2(val)	X(1B)
0x01	0x01	0x04	0xFF

If the pulse counter on GP1 is not running.

Copyright 2006 - 2007

3.8.3 CMD_IO IO_ADC

Layout:

BOT	op_len	CMD_IO	op_ref	node_id	**payload**	EOT
0x00	0x08	**0x0B**	X(1B)	X(2B)	X(4B)	0x04

Payload:

0x02	B1(vref,pin)	B2(stref)	X(1B)

Synopsis:

IO_ADC (0x02): This command make an ADC input measurement.

Specifics:

The ADC measurement is made with the following parameters:

- B1 bits b4 - b7 select the ADC reference: 0 for 1.5 V, 1 for 2.5 V, 2 for 3 V supply voltage
- B1 bits b0 - b3 select the pin number GP1 through GP7
- B2: sampling time reference in milliseconds; maximum is 64 ms, 0 defaults to 1 ms
- B2B3: returned 12-bit ADC measurement, little endian format. Note that the sampling time *stref* is overwritten by the ADC measurement.

X(1B) - value for this byte is not used.

Factory Settings or Defaults:

0 sampling time; defaults to 1 millisecond

Valid Parameter Range:

As given in *Specifics*

Power-up Value:

N/A

Results Returned:

Results are always returned, in usual format, with added op_rc field:

BOT	op_len	CMD_IO	op_ref	op_rc	node_id	**payload**	EOT
0x00	0x09	**0x0B**	X(1B)	X(1B)	X(2B)	X(4B)	0x04

Payload:

0x02	B1(vref,pin)	B2(ret value)	B3(ret value)

and with part of the payload overwritten with ADC data.

Error Codes:

RC_ERES **(0x02):** attempt to configure interrupt-driven GP1 or GP2
RC_EVAL **(0x07):** pin number is 0 or greater than 7, or ADC sample time greater than 64, or reference voltage selection is greater than 2.

Copyright 2006 - 2007

Notes:

The pin does not have to be pre-configured as analog input.
RF receiver is switched off for the sampling time.
The sample time byte is overwritten by the returned ADC measurement.

Examples:

Read ADC on GP6 (2.27 V is connected to pin 6).
Vref: Vcc (2)
Pin 6
Stref- sampling time 10 ms (0x0A)

```
00 08 0B CC 25 00 02 26 0A FF 04
```

operation	B1(vref,pin)	B2(stref)	X(1B)
0x02	**0x26**	0x0A	0xFF

B1: b7-b4 selects vref , b3-b0 selects pin
Vref: 0010, Pin: 0110, byte is 00100110 or 0x26
B2: stref is 0x0A

Returns:

```
00 09 0B CC 01 25 00 02 26 B7 03 04
```

operation	B1(vref,pin)	B2(ret value)	B3(ret value)
0x02	0x26	0xB7	0x03

B2-B3: measured value is 0x03B7

Read ADC on GP6 (2.27V is connected to pin 6).
Vref 2.5V (1)
Stref sampling time 1 ms (0)
Pin 6

```
00 08 0B CC 25 00 02 16 00 FF 04
```

operation	B1(vref,pin)	B2(stref)	X(1B)
0x02	**0x16**	0x00	0xFF

B1: b7-b4 selects vref , b3-b0 selects pin
Vref: 0001, Pin: 0110, byte is 00010110 or 0x16
B2: stref is 0x00

Returns:

```
00 09 0B CC 01 25 00 02 13 3F 0F 04
```

operation	B1(vref,pin)	B2(ret value)	B3(ret value)
0x02	0x13	0x3F	0x0F

B2-B3: measured value is 0x0F3F

Copyright 2006 - 2007

3.8.4 CMD_IO IO_CNT_START

Layout:

BOT	op_len	CMD_IO	op_ref	node_id	**payload**	EOT
0x00	0x08	**0x0B**	X(1B)	X(2B)	X(4B)	0x04

Payload:

B0(0xX3)	B1(val)	B2(val)	B3(val)

Synopsis:

IO_CNT_START (0xX3): This command sets the pulse counter to the 24-bit value in B1B2B3 (little endian) and starts counting pulses on GP1.

Specifics:

The four most significant bits of first payload byte control additional parameters:

- If bit 5 in B0 is set, the 24-bit value in B1B2B3 is ignored and the counter starts (resumes) with its currently held value
- If bit 4 in B0 is set, the counter increments on rising pulse edges (or falling edges if not set)

Factory Settings or Defaults:

N/A

Valid Parameter Range:

As in Specifics

Power-up Value:

N/A

Results Returned:

Results are always returned, in usual format, with added op_rc field:

BOT	op_len	**CMD_IO**	op_ref	op_rc	node_id	**payload**	EOT
0x00	0x09	**0x0B**	X(1B)	X(1B)	X(2B)	X(4B)	0x04

Payload:

0xX3	B1(val)	B2(val)	B3(val)

with the payload as in the original command.

Error Codes:

RC_ERES **(0x02)**: error, the counter is already active

Notes:

None

Copyright 2006 - 2007

Examples:

Counter start:

Starts counting from 10 on rising edges:

```
00 08 0B CC 00 01 13 0A 00 00 04
```

operation	B1(val)	B2(val)	3(val)
0x13	0x0A	0x00	0x00

Returns:

```
00 09 0B CC 01 00 01 13 0A 00 00 04
```

operation	B1(val)	B2(val)	3(val)
0x13	0x0A	0x00	0x00

Starts counting from 200 on falling edges:

```
00 08 0B CC 25 00 03 C8 00 00 04
```

0xX3	B1(val)	B2(val)	B3(val)
0x03	0xC8	0x00	0x00

Returns:

```
00 09 0B CC 01 25 00 03 C8 00 00 04
```

operation	B1(val	B2(val)	B3(val)
0x03	0xC8	0x00	0x00

Copyright 2006 - 2007

3.8.5 CMD_IO IO_CMP_START

Layout:

BOT	op_len	CMD_IO	op_ref	node_id	**payload**	EOT
0x00	0x08	**0x0B**	X(1B)	X(2B)	X(4B)	0x04

Payload:

0x04	B1(val)	B2(val)	B3(val)

Synopsis:

IO_CMP_START (0x04): This command sets the comparator to the 24-bit value in B1B2B3 (little endian) and starts comparing. An event message is generated when the value in the comparator matches the value in the pulse counter.

Specifics:

None

Factory Settings or Defaults:

0 disables the comparator event messaging (factory default).

Valid Parameter Range:

Values in the range of the 24-bit register are valid.

Power-up Value:

N/A

Results Returned:

Results are always returned, in usual format, with added op_rc field:

BOT	op_len	CMD_IO	op_ref	op_rc	node_id	**payload**	EOT
0x00	0x09	**0x0B**	X(1B)	X(1B)	X(2B)	X(4B)	0x04

Payload:

0x04	B1(val)	B2(val)	B3(val)

with payload as in the original command.

Error Codes:

RC_ERES **(0x02):** is returned if the comparator is already active.

Notes:

The I/O event message contains information on both GP1 pulse counting and GP2 digital input events. See Section 2.90 for event message layout details.

Examples:

Set the comparator value to 40 and start comparing:

00 08 0B CC 00 01 04 **28 00 00** 04

operation	B1(val)	B2(val)	B3(val)
0x04	0x28	0x00	0x00

Returns:

00 09 0B CC 01 00 01 04 **28 00 00** 04

operation	B1(val)	B2(val)	B3(val)
0x04	0x28	0x00	0x00

Copyright 2006 - 2007

3.8.6 CMD_IO IO_NOT_START

Layout:

BOT	op_len	CMD_IO	op_ref	node_id	**payload**	EOT
0x00	0x08	**0x0B**	X(1B)	X(2B)	X(4B)	0x04

Payload:

0x05	B1(edge)	X(1B)	X(1B)

Synopsis:

IO_NOT_START (0x05): This command enables event messaging on GP2.

Specifics:

X(1B) - the value for this byte is not used

Factory Settings or Defaults:

N/A

Valid Parameter Range:

See Synopsis

Power-up Value:

N/A

Results Returned:

Results are always returned, in usual format, with added op_rc field:

BOT	op_len	CMD_IO	op_ref	op_rc	node_id	**payload**	EOT
0x00	0x09	**0x0B**	X(1B)	X(1B)	X(2B)	X(4B)	0x04

Payload:

0x05	B1(edge)	X(1B)	X(1B)

with payload as in the original command.

Error Codes:

RC_ERES **(0x02):** is returned if event messaging is already active.

Notes:

The I/O event message contains information on both GP1 pulse counting and GP2 digital input events.

Copyright 2006 - 2007

Examples:

Start event messaging on GP2:

```
00 08 0B CC 00 01 05 01 00 00 04
```

Start GP2 event messages on rising edges (0x01):

operation	B1(edge)	X(1B)	X(1B)
0x05	0x01	0xFF	0xFF

Returns:

```
00 09 0B CC 01 00 01 05 01 00 00 04
```

operation	B1(edge)	X(1B)	X(1B)
0x05	0x01	0xFF	0xFF

Since GP2 is high and notifications are requested on rising edges, an I/O event message is sent immediately, producing this message output:

```
00 09 FF 13 00 01 77 0B 00 00 01 04
```

B1(node_id)	B2(node_id)	B3(iO_state)	B4(val)
0x00	0x01	0x77	0x0B

B5(val)	B6(val)	B7(autoACK)
0x00	0x00	0x01

Node ID (B1B2): 0x0100

I/O state (B3): 0x77, or bit pattern 01110111b

For GP2

- b0 set: edge is rising
- b1 set: notification is on
- b2 set: an event is pending
- b3 clear: (unused)

For GP1

- b4 set: counter edge is rising
- b5 set: counter is active
- b6 set: comparator is active
- b7 clear: a comparator event not pending

Counter value is (B4B5B6) is 0x00000B

AutoACK is ON (B7 is 0x01).

Copyright 2006 - 2007

3.8.7 CMD_IO IO_STOP

Layout:

BOT	op_len	CMD_IO	op_ref	node_id	**payload**	EOT
0x00	0x08	**0x0B**	X(1B)	X(2B)	X(4B)	0x04

Payload:

0x06	B1(count)	B2(comp)	B3(notifier)

Synopsis:

IO_STOP (0x06): This command stops some or all of the pulse counter operations on GP1 according to the least significant bits in B1, B2 and B3.

Specifics:

B1 set: stop the counter
B2 set: stop the comparator
B3 set: stop the comparator event messages

Factory Settings or Defaults:

N/A

Valid Parameter Range:

As in Specifics

Power-up Value:

N/A

Results Returned:

Results are always returned, in usual format, with added op_rc field:

BOT	op_len	CMD_IO	op_ref	op_rc	node_id	**payload**	EOT
0x00	0x09	**0x0B**	X(1B)	X(1B)	X(2B)	X(4B)	0x04

Payload:

0x06	B1(count)	B2(comp)	B3(notifier)

with payload as in the original command.

Error Codes:

RC_ERES **(0x02):** is returned if requested functionality is already stopped, with appropriate field marked.

Notes:

None

Copyright 2006 - 2007

Examples:

Stop pulse counter operation:
Where the pulse counter and event messaging are active.

```
00 08 0B CC 25 00 06 01 00 00 04
```

operation	B1(count)	B2(comp)	B3(notifier)
0x06	0x01	0x00	0x00

Returns:

```
00 09 0B CC 01 25 00 06 01 00 00 04
```

operation	B1(count)	B2(comp)	B3(notifier)
0x06	0x01	0x00	0x00

Only the pulse counter is stopped.

Stop all pulse counter operations:
Where only event messaging is active.

```
00 08 0B CC 25 00 06 01 01 01 04
```

operation	B1(count)	B2(comp)	B3(notifier)
0x06	0x01	0x01	0x01

Returns:

```
00 09 0B CC 02 25 00 06 F1 F1 01 04
```

operation	B1(count)	B2(comp)	B3(event msg)
0x06	0xF1	0xF1	0x01

0x01 0x01 0x01 attempts to stop everything, and returns 0xF1 0xF1 0x01 with RC_ERES (0x02) if counter and comparator were not active.
Despite RC_ERES, the event messaging is stopped as requested.
0xF1 means that counter, or comparator, or event messaging was not active.

Copyright 2006 - 2007

3.8.8 CMD_IO IO_CNT_GET

Layout:

BOT	op_len	CMD_IO	op_ref	node_id	**payload**	EOT
0x00	0x08	**0x0B**	X(1B)	X(2B)	X(4B)	0x04

Payload:

0x07	X(1B)	X(1B)	X(1B)

Synopsis:

IO_CNT_GET (0x07): This command gets the 24-bit value in GP1's pulse counter

Specifics:

Returns the 24-bit value in the pulse counter. In addition, bit 4 in the returned operations byte (first byte of the payload) encodes whether the counter is active or inactive.

X(1B) - value for this byte is not used.

Factory Settings or Defaults:

0

Valid Parameter Range:

As in Specifics

Power-up Value:

N/A

Results Returned:

Results are always returned, in usual format, with added op_rc field:

BOT	op_len	CMD_IO	op_ref	op_rc	node_id	**payload**	EOT
0x00	0x09	**0x0B**	X(1B)	X(1B)	X(2B)	X(4B)	0x04

Payload:

0x07	B1(value)	B2(value)	B3(value)

with bit 4 in the first byte of the payload encoding the counter active/inactive status.

Error Codes:

None

Notes:

None

Copyright 2006 - 2007

Examples:

Get the pulse counter value:

```
00 08 0B CC 00 01 07 FF FF FF 04
```

operation	X(1B)	X(1B)	X(1B)
0x07	0xFF	0xFF	0xFF

Returns:

```
00 09 0B CC 01 00 01 17 0C 00 00 04
```

operation	B1(value)	B2(value)	B3(value)
0x17	0x0C	0x00	0x00

The pulse counter value is 0x00000C.
The pulse counter is active (bit 4 in the first payload byte is set).

Get the pulse counter value:

```
00 08 0B CC 00 01 07 FF FF FF 04
```

operation	X(1B)	X(1B)	X(1B)
0x07	0xFF	0xFF	0xFF

Returns:

```
00 09 0B CC 01 00 01 07 0D 00 00 04
```

operation	B1(value)	B2(value)	B3(value)
0x07	0x0D	0x00	0x00

The pulse counter value is 0x00000D.
The counter is inactive (bit 4 in the first payload byte is not set).

Copyright 2006 - 2007

3.8.9 CMD_IO IO_CMP_GET

Layout:

BOT	op_len	CMD_IO	op_ref	node_id	**payload**	EOT
0x00	0x08	**0x0B**	X(1B)	X(2B)	X(4B)	0x04

Payload:

0x08	X(1B)	X(1B)	X(1B)

Synopsis:

IO_CMP_GET (0x08): This command gets the comparator's 24-bit value.

Specifics:

Bit 4 in the returned operations byte (first byte of the payload) encodes whether the comparator is active or inactive.

X(1B) - value for this byte is not used

Factory Settings or Defaults:

0 disables the comparator event messaging (factory default).

Valid Parameter Range:

Values in the range of the 24-bit register are valid.

Power-up Value:

N/A

Results Returned:

Results are always returned, in usual format, with added op_rc field:

BOT	op_len	CMD_IO	op_ref	op_rc	node_id	**payload**	EOT
0x00	0x09	**0x0B**	X(1B)	X(1B)	X(2B)	X(4B)	0x04

Payload:

0x08	B1(value)	B2(value)	B3(value)

with bit 4 in the first byte of the payload encoding the comparator active/inactive status.

Error Codes:

None

Notes:

None

Copyright 2006 - 2007

Examples:

Get the comparator's value:

```
00 08 0B CC 25 00 08 FF FF FF 04
```

operation	X(1B)	X(1B)	X(1B)
0x08	0xFF	0xFF	0xFF

Returns:

```
00 09 0B CC 01 25 00 18 0A 00 00 04
```

operation	B1(value)	B2(value)	B3(value)
0x18	0x0A	0x00	0x00

The comparator is set to 10 (0x00000A).
The comparator is active (bit 4 in the first payload byte is set).

Get the comparator's value:

```
00 08 0B CC 25 00 08 FF FF FF 04
```

operation	X(1B)	X(1B)	X(1B)
0x08	0xFF	0xFF	0xFF

Returns:

```
00 09 0B CC 01 25 00 08 0A 00 00 04
```

operation	B1(value)	B2(value)	B3(value)
0x08	0x0A	0x00	0x00

The comparator value is set to 10 (0x00000A)
The comparator is inactive (bit 4 in the first payload byte is clear).

Copyright 2006 - 2007

3.8.10 CMD_IO IO_STATE

Layout:

BOT	op_len	CMD_IO	op_ref	node_id	**payload**	EOT
0x00	0x08	**0x0B**	X(1B)	X(2B)	X(4B)	0x04

Payload:

0x09	X(1B)	X(1B)	X(1B)

Synopsis:

IO_STATE (0x09): This command gets I/O states encoded in byte B1.

Specifics:

The I/O states for GP1 and GP2 are is returned in the bit pattern of B1:

For GP2

b0 set: send event message on a rising edge
b1 set: event messaging is on
b2 set: an event message is pending
b3 is 0

For GP1

b4 set: count pulses on rising edges
b5 set: pulse counter is active
b6 set: comparator is active
b7 set: a comparator event message is pending

X(1B) - the value in these bytes are not used

Factory Settings or Defaults:

0x00

Valid Parameter Range:

0x00 to 0xFF

Power-up Value:

N/A

Results Returned:

Results are always returned, in usual format, with added op_rc field:

BOT	op_len	CMD_IO	op_ref	op_rc	node_id	**payload**	EOT
0x00	0x09	**0x0B**	X(1B)	X(1B)	X(2B)	X(4B)	0x04

Payload:

0x09	B1(state)	X(1B)	X(1B))

with payload as in the original command.

Copyright 2006 - 2007

Error Codes:

None

Notes:

None

Examples:

Get I/O states:

```
00 08 0B CC 00 01 09 FF FF FF 04
```

operation	X(1B)	X(1B)	X(1B)
0x09	0xFF	0xFF	0xFF

Returns:

```
00 09 0B CC 01 00 01 09 70 FF FF 04
```

operation	B1(state)	X(1B)	X(1B)
0x09	0x70	0xFF	0xFF

0x70, or bit pattern 0111 0000

For GP2

b0 clear: send event message on a falling edge
b1 clear: event messaging is OFF
b2 clear: no event message pending
b3 clear (not used)

For GP1

b4 set: counting pulse rising edges
b5 set: pulse counting is active
b6 set: comparator is active
b7 clear: no comparator event message pending

Copyright 2006 - 2007

3.8.11 CMD_IO IO_CREG_GET

Layout:

BOT	op_len	CMD_IO	op_ref	node_id	**payload**	EOT
0x00	0x08	**0x0B**	X(1B)	X(2B)	X(4B)	0x04

Payload:

0x0A	X(1B)	X(1B)	X(1B)

Synopsis:

IO_CREG_GET (0x0A): This command gets the 24-bit offset register (CREG) value.

Specifics:

X(1B) - the value in this byte is not used

Factory Settings or Defaults:

0x000000

Valid Parameter Range:

0x000000 to 0x0FFFFF

Power-up Value:

N/A

Results Returned:

Results are always returned, in usual format, with added op_rc field:

BOT	op_len	CMD_IO	op_ref	op_rc	node_id	**payload**	EOT
0x00	0x09	0x0B	X(1B)	X(1B)	X(2B)	X(4B)	0x04

Payload:

0x0A	B1(value)	B2(value)	B3(value)

with the payload containing the 24-bit offset register value.

Error Codes:

None

Notes:

None

Copyright 2006 - 2007

Examples:

Get the offset register (CREG) value:

```
00 08 0B CC 00 01 0A FF FF FF 04
```

operation	X(1B)	X(1B)	X(1B)
0x0A	0xFF	0xFF	0xFF

Returns:

```
00 09 0B CC 01 00 01 0A 0B 00 00 04
```

operation	B1(value)	B2(value)	B3(value)
0x0A	0x0B	0x00	0x00

The offset register is set to 11 (0x00000B).

Copyright 2006 - 2007

3.8.12 CMD_IO IO_CREG_SET

Layout:

BOT	op_len	CMD_IO	op_ref	node_id	**payload**	EOT
0x00	0x08	**0x0B**	X(1B)	X(2B)	X(4B)	0x04

Payload:

0x0B	B1(val)	B2(val)	B3(val)

Synopsis:

IO_CREG_SET (0x0B): This command sets the 24-bit offset register (CREG) value.

Specifics:

None

Factory Settings or Defaults:

0x000000

Valid Parameter Range:

0x000000 to 0x0FFFFF

Power-up Value:

N/A

Results Returned:

Results are always returned, in usual format, with added op_rc field:

BOT	op_len	CMD_IO	op_ref	op_rc	node_id	**payload**	EOT
0x00	0x09	**0x0B**	X(1B)	X(1B)	X(2B)	X(4B)	0x04

Payload:

0x0B	B1(val)	B2(val)	B3(val)

with payload as in the original command.

Error Codes:

None

Notes:

The counter's offset register (CREG) allows useful operations without operator's involvement, as it can automatically update the counter's or comparator's event message trigger thresholds, etc.

Copyright 2006 - 2007

Examples:

Set the offset register (CREG)vale to 0x00000B:

```
00 08 0B CC 00 01 0B 0B 00 00 04
```

operation	B1(val)	B2(val)	B3(val)
0x0B	0x0B	0x00	0x00

Returns:

```
00 09 0B CC 01 00 01 0B 0B 00 00 04
```

operation	B1(val)	B2(val)	B3(val)
0x0B	0x0B	0x00	0x00

The offset register's value is set to 0x00000B.

Copyright 2006 - 2007

3.8.13 CMD_IO IO_FLAGS_GET

Layout:

BOT	op_len	CMD_IO	op_ref	node_id	**payload**	EOT
0x00	0x08	**0x0B**	X(1B)	X(2B)	X(4B)	0x04

Payload:

0x0C	X(1B)	X(1B)	X(1B)

Synopsis:

IO_FLAGS_GET (0x0C): This command gets the pulse counter's nonvolatile memory backup timer in B1 and the offset register mode control flags in B2 (mode action triggered by a match between the value in the comparator and the pulse counter match).

Specifics:

B1 holds the backup timer period, 0 to 63, increments of 15 minutes
B2 operation modes:

0x00: do nothing with the offset register value
0x01: add the offset register value (CREG) to the comparator when triggered
0x02: subtract the offset register value (CREG) from the pulse counter when triggered
0x03: subtract the comparator value from the counter when triggered

X(1B) the value for this byte is not used.

Factory Settings or Defaults:

The default counter nonvolatile memory backup time is 0 seconds (0 disables backup)
The default operation mode is 0x00 (do nothing)

Valid Parameter Range:

B1: 0 to 63 in increments of 15 minutes
B2: 0, 1, 2, 3

Power-up Value:

Defaults retrieved from program memory.

Results Returned:

Results are always returned, in usual format, with added op_rc field:

BOT	op_len	CMD_IO	op_ref	op_rc	node_id	**payload**	EOT
0x00	0x09	**0x0B**	X(1B)	X(1B)	X(2B)	X(4B)	0x04

Payload:

0x0C	B1(timer)	B2(operation)	X(1B)

and the payload carrying timer and mode data.

Copyright 2006 - 2007

Error Codes:

None

Notes:

None

Examples:

Get the pulse counter I/O flags:

```
00 08 0B CC 25 00 0C FF FF FF 04
```

operation	X(1B)	X(1B)	X(1B)
0x0C	0xFF	0xFF	0xFF

Returns:

```
00 09 0B CC 01 25 00 0C 00 00 FF 04
```

operation	B1(timer)	B2(operation)	X(1B)
0x0C	0x00	0x00	0xFF

B1: 0x00 - timer
B2: 0x00 - operation (do nothing)

The pulse counter nonvolatile memory backup is disabled.
The offset register value is not used in any way when a comparator trigger occurs.

Copyright 2006 - 2007

3.8.14 CMD_IO IO_FLAGS_SET

Layout:

BOT	op_len	CMD_IO	op_ref	node_id	**payload**	EOT
0x00	0x08	**0x0B**	X(1B)	X(2B)	X(4B)	0x04

Payload:

0x0D	B1(timer)	B2(flag)	X(1B)

Synopsis:

IO_FLAGS_SET (0x0D): This command sets the pulse counter's nonvolatile memory backup timer and the offset register mode control flags (mode action triggered by a match between the value in the comparator and the pulse counter match).

Specifics:

B1 holds the backup timer period, 0 to 63 increments of 15 minutes

B2 holds the offset register modes control flags:

0x00: do nothing with the offset register value

0x01: add the offset register value (CREG) to the comparator when triggered

0x02: subtract the offset register value (CREG) from the pulse counter when triggered

0x03: subtract the comparator value from the counter when triggered

X(1B) - the value in this byte is not used.

Factory Settings or Defaults:

The default nonvolatile memory backup time for the counter is 0 seconds (0 disables backup)

The default operation mode is 0x00 **(**do nothing)

Valid Parameter Range:

Backup timer, 0 to 63 increments of 15 minutes.

Flags, 0 to 3

Power-up Value:

Default retrieved from program memory.

Results Returned:

Results are always returned, in usual format, with added op_rc field:

BOT	op_len	CMD_IO	op_ref	op_rc	node_id	**payload**	EOT
0x00	0x09	**0x0B**	X(1B)	X(1B)	X(2B)	X(4B)	0x04

Payload:

0x0D	B1(timer)	B2(flag)	X(1B)

with the payload as in the original command.

Copyright 2006 - 2007

Error Codes:

RC_EVA **(0x07):** error, bad value

RC_ERES **(0x02):** is returned if B1 is greater than 63 or B2 is greater than 3.

Notes:

IO_FLAGS allow useful operations without operator's involvement, as they can automatically update the counter's or comparator's event message trigger thresholds, etc.

Examples:

Set I/O flags:

```
00 08 0B CC 25 00 0D 10 03 FF 04
```

operation	B1(timer)	B2(flag)	X(1B)
0x0D	0x10	0x03	0xFF

Returns:

```
00 09 0B CC 01 25 00 0D 10 03 FF 04
```

operation	B1(timer)	B2(flag)	X(1B)
0x0D	0x10	0x03	0xFF

B1: 0x10 - nonvolatile memory backup frequency 16 second

B2: 0x03 - subtract the comparator value from the counter

Copyright 2006 - 2007

3.8.15 CMD_IOACK

Layout:

BOT	op_len	**CMD_IOACK**	op_ref	node_id	**payload**	EOT
0x00	0x08	**0x0C**	X(1B)	X(2B)	X(4B)	0x04

Payload:

B1(io_state)	B2(counter)	B3(counter)	B4(counter)

Synopsis:

CMD_IOACK (0x0C): This command acknowledges an I/O event message.

Specifics:

This command acknowledges an I/O event message being sent by the target node. The four bytes must be the same as in the I/O event message being acknowledged. Note that this functionality is automatic if PAR_AUTOACK is set.

If another event occurs while the first one is still unacknowledged, retries for the first event are cancelled and reporting of the new event starts its cycle.

Factory Settings or Defaults:

N/A

Valid Parameter Range:

See Specifics

Power-up Value:

N/A

Results Returned:

Results are always returned, in usual format, with added op_rc field:

BOT	op_len	**CMD_IOACK**	op_ref	op_rc	node_id	**payload**	EOT
0x00	0x09	**0x0C**	X(1B)	X(1B)	X(2B)	X(4B)	0x04

Payload:

B1(io_state)	B2(counter)	B3(counter)	B4(counter)

with the payload as in the original command.

Error Codes:

RC_OK **(0x00):** success
RC_EMEM **(0x0B):** error, no memory to send the message.

Notes:

None

Copyright 2006 - 2007

Examples:

PAR_AUTOACK on the master node is set to 0.

An I/O event message is transmitted:

The event message received and output:

```
00 09 FF 13 25 00 17 07 00 00 00 04
```

Send an I/O event message acknowledgment:

```
00 08 0C AA 25 00 17 07 00 00 04
```

B1(io_state)	B2(counter)	B3(counter)	B4(counter)
0x17	0x07	0x00	0x00

Returns:
None

Further event message retransmissions are stopped.

Copyright 2006 - 2007

3.8.16 CMD_DAT

Layout:

BOT	op_len	**CMD_DAT**	op_ref	node_id	**payload**	EOT
0x00	**X(nB+4)**	**0x0D**	X(1B)	X(2B)	X(nB)	0x04

Payload:

B1()	B2()	...	Bn()

Synopsis:

CMD_DAT (0x0D): This command outputs the string of bytes in the payload at the addressed node's serial port.

Specifics:

None

Factory Settings or Defaults:

N/A

Valid Parameter Range:

Maximum length is 64 bytes

Power-up Value:

N/A

Results Returned:

Results are not returned. If the most significant bit of op_rc is set, a msg_datAck is returned to the issuer; msg_datAck format:

BOT	op_len	**msg_output**	msg_type	**msg_dat**	EOT
0x00	0x05	**0xFF**	**0x16**	**X(3B)**	0x04

Message:

B1(node_id)	B2(node_id)	B3(operation_ref)

Error Codes:

RC_EMEM **(0x0B):** error, no memory to send the message.

Notes:

None

Copyright 2006 - 2007

Examples:

PAR_MODE (0x23) on node 0x0025 is set to 0 to invoke data mode serial port operation.

Send a string of bytes to node ID 0x0025:
`00 0A 0D AA 25 00 **AB CD 66 77 EF 77** 04`
(the most significant bit of op_rc is set)

Returns:
The message is output and a msg_datAck sent back:
`00 05 FF **16 25 00** A1 04`

B1(node_id)	B2(node_id)	B3(operation_ref)
0x25	0x00	0xAA

Data output on node 0x0025:
`AB CD 66 77 EF 77`

Send a string of bytes: `AB 0D 66 77 EF 77` from the master to node 0x0025 (with 0x0D - carriage return in the string):

`00 0A **0D** AA 25 00 AB **0D** 66 77 EF 77 04`
(The most significant bit of op_rc is set, 0x0D carriage return in the string)

Returns:
The message is output and a msg_datAck sent back:
`00 05 FF **16 25 00 AA** 04`

B1(dest_node_id)	B2(dest_node_id)	B3(operation_ref)
0x25	0x00	0xAA

Data output on node 0x0025:
`AB 0D 66 77 EF 77`

Send a string of bytes: `66 77` from the master to node 0x0025
`00 06 0D **11** 25 00 66 77 04`
(The most significant bit of op_rc is not set)

Returns:
None

Data output on node 0x0025:
`66 77`

Copyright 2006 - 2007

Send a string of bytes from node 0x0025:
String: `66 77 68 01 02`

Returns:
None

Message output only on the master in msg_data format:
`00 0B FF` **`15`** `25 00 B0` **`66 77 68 01 02`** `01 04`
msg_data:

B1(sr_node_id)	B2(sr_node_id)	B3(data_ref)	B4(data)	B5(data)
0x25	0x00	0xB0	0x66	0x77

…

B6(data)	B7(data)	B8(data)	Bn+1(autoACK)
0x68	0x01	0x02	0x01

0x0025: source node ID
B0: data reference
66 77 68 01 02: string of bytes (data)
X01: auto-acknowledgment (ON)

Send a string of bytes (44 bytes) from node 0x0025:

`66 77 BB 68 01 24 36 36 77 66 77 BB 68 66 77 BB 68 66 77 BB 01 02`
`01 02 01 02 66 77 BB 68 01 02 BB 68 01 02 30 32 66 77 BB 68 01 02`

Message output only on the master in msg_data format:

`00 32 FF 15 25 00 92` **`66 77 BB 68 01 24 36 36 77 66 77 BB 68 66 77`**
`BB 68 66 77 BB 01 02 01 02 01 02 66 77 BB 68 01 02 BB 68 01 02 30`
`32 66 77 BB 68 01 02` `01 04`

x0025: source node ID
0xBB: data reference

`66 77 BB 68 01 24 36 36 77 66 77 BB 68 66 77 B 68 66 77 BB 01 02 01`
`02 01 02 66 77 BB 68 01 02 BB 68 01 02 30 32 66 77 BB 68 01 02:`

string of bytes (data)
0x01: autoACK (ON)

Copyright 2006 - 2007

Send a string of bytes from node 0x0025:
(0x0D - carriage return was put in the string of data)

String:
```
00 04 0D 11 25 00 04
```

Returns:
None

Two messages are output only on the master:

```
00 09 FF 15 25 00 84 00 04 0D 01 04
00 0A FF 15 25 00 85 11 25 00 04 01 04
```

Note:
0x0D was put in the string sent by the node 0x0025. The 0x0D carriage return creates two message outputs at the master, each in msg_data format:

First message:
```
00 09 FF 15 25 00 84 00 04 0D 01 04
```

0x0025:	source node ID
0x84:	data reference
0x00 0x04 0x0D:	string of bytes (data)
0x01:	Auto-acknowledgment (ON)

Second message:
```
00 0A FF 15 25 00 85 11 25 00 04 01 04
```

0x0025:	source node ID
0x85:	data reference
0x11 0x25 0x00 0x04:	string of bytes (data)
0x01:	Auto-acknowledgment (ON)

Copyright 2006 - 2007

Send a string of bytes from the node 0x0035 to node 0x0025:

```
00 0A 0D AA 25 00 AB 0D 66 77 EF 77 04
```
(The most significant bit of op_rc is set)

Returns:
The message is output and a msg_datAck sent back:
```
00 05 FF 16 25 00 AA 04
```

B1(dest_node_id)	B2(dest_node_id)	B3(operation_ref)
0x25	0x00	0xAA

Data output on node 0x0025: `AB 0D 66 77 EF 77`

Send a string of bytes from the master to node 0x0035:
```
00 0A 0D AA 35 00 AB 0D 66 77 EF 77 04
```
(The most significant bit of op_rc is set)

Returns:
The message is output and a msg_datAck sent back:
```
00 05 FF 16 35 00 AA 04
```

msg_dataAck:

B1(dest_node_id)	B2(dest_node_id)	B3(operation_ref)
0x35	0x00	0xAA

On node 0x0035:
Message is output in msg_data format:
```
00 0C FF 15 00 01 AA AB 0D 66 77 EF 77 01 04
```

Msg_data:

B1(sr_node_id)	B2(sr_node_id)	B3(data_ref)	B4(data)	B5(data)
0x00	0x01	0xAB	0x0D	0x66

…

B6(data)	B7(data)	B8(data)	Bn+1(autoACK)
0x77	0xEF	0x77	0x01

Copyright 2006 - 2007

3.8.17 CMD_DATACK

Layout:

BOT	op_len	**CMD_DATACK**	op_ref	node_id	**payload**	EOT
0x00	**0x05**	**0x0E**	X(1B)	X(2B)	**X(1B)**	0x04

Payload:

B1(data_ref)

Synopsis:

CMD_DATACK (0x0E): This command sends an acknowledgment to a data message.

Specifics:

This command acknowledges a data message being sent by the target node. The data reference byte *op_ref* must be the same as in the data message being acknowledged. Note that this functionality is automatic if PAR_AUTOACK is set.

Factory Settings or Defaults:

N/A

Valid Parameter Range:

See specifics

Power-up Value:

N/A

Results Returned:

None

Error Codes:

RC_EMEM **(0x0B):** error, no memory to send the message.

Notes:

None

Examples:

PAR_AUTOACK on the master is set to 0.
PAR_MODE on node 0x0025 is set to 0 (data mode)
String of bytes `66 77 68 01 02` is sent by node 0x0025

The message output in msg_data format:

```
00 0B FF 15 25 00 B5 66 77 68 01 02 00 04
```

With AutoACK off, the message will be retransmitted after the retry timeout period.

Copyright 2006 - 2007

Send acknowledgment with CMD_DATACK:

```
00 05 0E AA 25 00 B5 04
```

B1(data_ref)
0xB5

The data reference byte (0xB5) must be the same as in the msg_data format being acknowledged

Returns: None

The further retransmission of the message is stopped.

Copyright 2006 - 2007

4 Glossary

Attribute (ATTR)
Read only data items are called attributes.

Beacon
Beacons are messages that are sent by the master node that contain information of interest to field nodes.

Binding
Binding is the process of a field node joining a network.

BOT
Beginning of transmission byte, 0x00.

Command Mode
A master or field node that is able to receive commands.

Data Mode / Logger
Node with UART interface set in “data mode”, to input / output raw strings of bytes into the network.

EOT
End of transmission byte, 0x04.

ESN
Equipment serial number, 4 bytes, little endian format, a unique constant assigned to each node.

Field Node
A node that is configured not to be the master but a node in the mesh network. Also called and end node.

Little endian
Little endian is a term that describe the order in which a sequence of bytes are stored in computer memory. Little endian is an order in which the "little end" (least significant value in the sequence) is stored first.

Master Node
In a given network, a single node acts as the master node that provides the gateway to outside communications. Also called a coordinator node.

Neighborhood
The field nodes that are one hop from the field node being asked.

Copyright 2006 - 2007

Network ID
Identifier of a network. Messages are sent between nodes with the same network ID.

Node ID
The 2 byte address (logical identifier) of a node, unique within a network (2 bytes).

NVM
Nonvolatile memory (EEPROM or Flash Information Memory).

Parameter (PAR)
Configurable data items are called parameters.

Payload
The part of the message that contains data.

Response Message
A message sent in response to a command.

Return Code (RC)
The code that indicates the success of the command. Also called an error code.

RSSI
Received Strength Signal Indicator

Source Node
The node where a command or message is initiated.

Target Node
Command's target node ID, usually destination of a message generated by the command.

TARP
Tiny **A**d-hoc **R**outing **P**rotocol

Copyright 2006 - 2007