

KIDWeb Command Reference – rev. 1.13

Telnet Commands supported by the KIDWeb module

This document will be updated with the latest information about the commands. Please check for updates that may contain new commands and/or more thorough explanation of the current commands. Each request for more information about a particular command will be fulfilled and this document updated.

Contact persons: Emil Popov and/or Dimitar Petrov – indsoft@einet.bg

Note1: All commands are case-INsensitive.

Note2: All numbers are entered in their HEX representation without the leading '0x', for example, 0x200 is entered as 200. The "b" command is an exception.

Note3: arguments enclosed in '[' ']' are optional, parameters enclosed in '{' '}' are mandatory.

Note4: All those commands can be used with both a telnet connection and with the ActiveX control supplied by Industrial Software ltd.

COMMAND	COMMAND DESCRIPTION
"h"	Deprecated as of 5.0.9
"b [baud]"	Sets/Displays the current CAN baud rate. If the argument is omitted, the current baud rate is displayed. These arguments are entered in decimal. Examples: "b" – display current CAN baud rate "b 125" – set the CAN baud rate to 125kbs Acceptable baud rates are: 1000, 500, 250, and 125
"s [node_id]"	Displays the status of one/all modules. If you specify a node_id, only the state of the device with this node_id will be displayed. If no argument is given, the status of all the modules will be displayed. One line of data is printed for every module. Examples: "s" – show the status of all devices "s 1E" – show the status of the device with node_id: 0x1E Output Format: <i>node_id_1 state<CR><LF></i> <i>node_id_2 state<CR><LF></i> <i>.</i> <i>.</i> <i>.</i> <i>node_id_n state<CR><LF></i> <i>O.K<CR><LF></i>
"dr {node_id} {index} {subindex}"	Dictionary Read operation. Displays the information, stored in the <i>index->subindex</i> of the module with the given <i>node_id</i> . The data is displayed in HEX
"du {node_id} {index} {subindex}"	Domain Upload operation. Reads a Domain object, as stored in the <i>index->subindex</i> . The data is displayed in HEX with spaces after every byte. Up to 70 bytes can be read. Example:

	<p>“du 1 2000 4” – read the domain object stored in index: 0x2000, subindex: 4 of the module with nodeid: 1</p> <p>Output Format: 30 31 32 33 34 31 32 33 34 35<CR><LF> O.K.<CR><LF></p>
<p>“<i>dw</i> {<i>node_id</i>} {<i>index</i>} {<i>subindex</i>} {<i>value</i>} {<i>length</i>}”</p>	<p>Dictionary Write operation. Writes <i>length</i> bytes with <i>value</i> to the <i>index</i>-><i>subindex</i> of the module with the given <i>node_id</i></p> <p>Output Format (error in entered values): <i>Err.</i><CR><LF></p> <p>Output Format (no error): <i>O.K</i><CR><LF></p>
<p>“<i>fs</i> {<i>node_id</i>}”</p>	<p>Store the current configuration parameters of the module with this <i>node_id</i> to its internal Flash memory. This command is only needed if you have changed some parameters of the module and you want it to remember this configuration.</p>
<p>“<i>fr</i> {<i>node_id</i>}”</p>	<p>Restore the Default configuration parameters of the module with this <i>node_id</i>. After this command, you do NOT need to issue an “fs” command.</p>
<p>“<i>tm</i> {<i>COB-ID</i>} {<i>length</i>} {<i>data</i>}”</p>	<p>Sends an arbitrary CAN message with the specified COB-ID, and data. The data can be from zero to 8 bytes.</p> <p>Example1: “tm 195 8 11223344aabbccdd” – will send a CAN message with COB-ID: 0x195, and with 8 bytes of data. The data that is going to be sent is 0x11, 0x22, 0xCC, 0xDD.</p> <p>Example2: “tm 212 2 A5A5” – will send two bytes of data (0xA5A5) with COB-ID: 0x212.</p> <p>Example3: “tm 13 0” – will send a message with COB-ID: 0x13 with no data.</p>
<p>“<i>mb</i> 100” / “<i>mb</i> 200”</p>	<p>“mb 100” instructs the KIDWeb to bring all the modules to Operational mode upon boot up. This is the DEFAULT behavior.</p> <p>“mb 200” instructs the KIDWeb NOT to bring all the modules to Operational mode upon boot up.</p> <p>NOTE: The “mb” command writes the parameter directly to the internal Flash memory, so it should be used with care. Always check the current state with the “im” command and only if necessary, use the “mb” command to change it.</p>
<p>“<i>im</i>”</p>	<p>The current configured behavior of the KIDWeb is displayed. If ‘1’ is returned, all the modules will be put into Operational mode upon boot up. If ZERO is returned, the</p>

	modules will be left in their current mode when the KIDWeb boots up.
“gc”	Clears the group of COB-IDs that are monitored by the KIDWeb and sent over UDP. <i>For more information on this subject, read Appendix A</i>
“gas {COB-ID}”	Add (register) a single COB-ID to the list of monitored COB-IDs. <i>For more information on this subject, read Appendix A</i>
“gam {COB-ID1} [COB-ID2] ... [COB-IDn]”	Add (register) multiple COB-IDs to the list of monitored COB-IDs. <i>For more information on this subject, read Appendix A</i>
“gb {UDP_MODE}”	<p>Sets or Displays the current mode of the Cobid -> UDP sniffer. When executed without the UDP_MODE argument, “gb” displays 0 if the sniff packets are sent to the PC that registered the cobids, or 1 if the UDP packets are sent to the broadcast address. UDP_MODE should be 0 or 1. When the KidWEB module boots, this is set to 1 therefore the default is that UDP messages are sent as broadcasts. This command clears the list of monitored COB-IDs, so after its use, one should re-register the COB-IDs again.</p> <p>Output Format without argument: <i>UDP_SNIFF_BROADCAST = 1 O.K<CR><LF></i> <i>UDP_SNIFF_BROADCAST = 0 O.K<CR><LF></i></p> <p>Output Format with argument: <i>O.K<CR><LF></i></p>
“u {firmware_file}”	OBSOLETE! As of KIDWeb’s revision 3.4 use the command below. The new command syntax is backwards, but it is recommended to use the below syntax from now on for any KIDWeb device with firmware revision 3.4 and above.
“u {firmware_file} [node_id]”	<p>Upgrade the Firmware of the KIDWeb or a CAN module with the <i>firmware_file</i>. If <i>node_id</i> is specified as ZERO or is omitted, the KIDWeb’s firmware will be upgraded. If <i>node_id</i> is any number from 1 to 0x7F, the CAN module with the corresponding <i>node_id</i> will be upgraded.</p> <p>Note: the File containing the new revision should already be uploaded using FTP. This is valid for both upgrading the KIDWeb or a CAN module. The KIDWeb firmware filename should be in the format: BIOSXXX.BIN where XXX is the revision. The filename for upgrading a CAN module can be any .BIN file. When upgrading the KIDWeb, “O.K” is displayed and then the module restarts. When Upgrading a CAN module’s firmware, the response will be delayed until the module is upgraded, so a response may appear as much as 10-15sec after the command is issued.</p> <p>Note2: Module upgrade cannot be done when the module is in state STOPPED.</p> <p>Note3: When upgrading a CAN module, you cannot use a name starting with “BIOS”. This is so that somebody</p>

	<p>doesn't accidentally upgrade a module with the KIDWeb's firmware.</p> <p>Example: "u BIOS340.BIN" – upgrades the KIDWeb's firmware.</p> <p>Example2: "u BIOS340.BIN 0 " – same as above.</p> <p>Example3: "u ROLLER720.BIN 73 " – upgrade the firmware of the module with node_id 0x73</p> <p>Example4: "u FILE123.BIN 4A " – upgrade the firmware of the module with node_id 0x4A</p> <p>Output Format: <i>O.K<CR><LF> - no error</i> <i>Err:Not Valid Filename<CR><LF> - filename doesn't fulfill the requirements</i> <i>Err:No such File<CR><LF> - file was not found, maybe you forgot to upload it.</i> <i>Err: InitSDODown<CR><LF> - error during upgrade initialization, maybe the module with this node_id is not available or is in state STOPPED</i> <i>Err: SDODown<CR><LF> - error during data transfer – the module is left with no valid firmware (kernel not damaged)</i> <i>Err: SDO Close<CR><LF> - error during upgrade finalization – the module is left with no valid firmware (kernel not damaged)</i></p>
"is"	Displays the serial number of the KIDWeb Output Format: <i>10001<CR><LF></i>
"i"	Displays the IP address of the KIDWeb Output Format: <i>192.168.0.119<CR><LF></i>
"ie"	Displays the Ethernet (MAC) address of the KIDWeb Output Format: <i>00 00 0E 33 96 28<CR><LF></i>
"ic"	Displays the current IP connections present to the KIDWeb module. Output Format: <i>Command: "ic"<CR><LF></i> <i>192.168.200.2<CR><LF></i> <i>192.168.200.16<CR><LF></i>
"in"	Displays the name that has been given to the KIDWeb
"mi"	Displays the timestamp of the connection configuration file, or Zero if the file is not found. Output when No CONN.BIN file is found: <i>00 00 00 00<CR><LF></i> <i>O.K<CR><LF></i>

	<p>Normal output when CONN.BIN is present: <i>11 22 AA BB<CR><LF></i> <i>O.K<CR><LF></i></p> <p>Note: for more information regarding the CONN.BIN file and the inter-kidWeb connections, read <i>Appendix C</i></p>
“ms”	Management Start. This command sets all the devices on the CAN network in Operational mode (state 5)
“ms {node_id}”	Management Start Node. This command sets the device with the specified NODE_ID in Operational mode (state 5)
“mt”	Management Stop. This command sets all the devices on the CAN network in Stopped mode (state 4)
“mt {node_id}”	Management Stop Node. This command sets the device with the specified NODE_ID in Stopped mode (state 4)
“mr”	Management Reset. This command Resets all the devices on the CAN network
“mr {node_id}”	Management Reset Node. This command Resets the device with the specified NODE_ID
“mp”	This command sets all the devices on the CAN network in Preoperational mode (state 0x7F)
“mp {node_id}”	This command sets the device with the specified NODE_ID in Preoperational mode (state 0x7F)
“mg”	Stop Node Guarding
“mg 1”	Start Node Guarding
“mc”	Management Communications Reset for all modules
“mc {node_id}”	Management Communications Reset for the module with the specified node_id.
“r”	<p>Displays the current Firmware revision. Let’s suppose the revision is: Major: 3, Minor: 2 The output will be as follows.</p> <p>Output Format: <i>3.2<CR><LF></i></p>
“mm {INTERVAL}”	<p>This command instructs the KIDWeb to send the status of all the nodes currently on the CAN network in a broadcast UDP packet. The packet is sent according to the specified interval in seconds. The interval is also specified in HEX. <i>For a more thorough explanation of the UDP packet data, please look at Appendix B</i></p> <p>The interval is dependent on the staging mode. If the KIDWeb is in STAGING_OFF state, the interval is in 1 second increments. If however the KIDWeb is in STAGING_MONITOR or STAGING_CONTROLER state, than the interval is specified in 100ms increments. Example: if interval is 8, when the KIDWeb is in the default STAGING_OFF state, the UPD packets will be transmitted every 8 seconds, but if the mode is changed to STAGING_MONITOR or STAGING_CONTROLER, the messages will start appearing every 0.8 seconds.</p>
“mm”	This command stops the transmission of UDP packets containing the status of the nodes on the CAN network. <i>For a more thorough explanation of the UDP packet data, please look at Appendix B</i>
“mms”	Displays the current status of the

	<p>Output Format for status monitoring every 15 seconds: <i>0F<CR><LF></i></p> <p>Output Format for status monitoring disabled: <i>00<CR><LF></i></p>
<i>“mo”</i>	<p>Displays the CAN and Ethernet overflow counters. Output Format: <i>O.K OVF Counters: 00 00 00 00 00 00 00 00</i> <i><CR><LF></i></p> <p>The 8 bytes have the following meaning: <i>First 4 Bytes</i> – CAN overflow counter <i>Second 4 Bytes</i> – Ethernet overflow counter Note: the counters are zeroed after every “mo” command</p>
<i>“me”</i>	<p>Displays the state of the CAN error counters. Output Format: <i>O.K Error Counters: 00 00 00 00 <CR><LF></i></p> <p>The 4 bytes have the following meaning: <i>Byte1</i> – receive error counter <i>Byte2</i> – transmit error counter <i>Byte3</i> – CAN state (0=Error Active, 1=warning, 2=Error Passive, 3=Bus OFF) <i>Byte4</i> – reserved</p>
<i>“wrm [node_id]”</i>	<p>KIDWEB2 ONLY STAGING_COMMAND! Displays the last reported state of a staging module. This is the state that gets preserved in NVRam. The command displays the state and tracking information for the 3 zones of the single module. Output Format: <i>0xSS 0xRR 0xTTTT 0xTTTT, 0xSS 0xRR 0xTTTT</i> <i>0xTTTT, 0xSS 0xRR 0xTTTT 0xTTTT<CR><LF></i> Where: 0xSS is zone status, 0xRR is reserved, 0xTTTT is a tracking identifier. Every zone has 2 tracking identifiers, but the spine zone’s second tracking is not used. In the output above, first the Spine zone is displayed, then the Left zone, and finally the Right zone. Note: node_id should be between 1 and 0x40</p>
<i>“wrz [zone_id]”</i>	<p>KIDWEB2 ONLY STAGING_COMMAND! The same as above but for a single zone. The zone_ids are calculated as node_id + X, where X is zero for the spine zones, 0x40 for the left zones and 0x80 for the right zones. Therefore if we want to see the status of the left zone on node_id 0x23, we need to execute “wrz 63” while the right zone for the same node_id would have zone_id = 0x23 + 0x80 = 0xA3 Output Format: <i>0xSS 0xRR 0xTTTT 0xTTTT<CR><LF></i></p>
<i>“wsg”</i>	<p>KIDWEB2 ONLY STAGING_COMMAND! Shows the current staging behavior. It can be: 0x00 - OFF – nothing staging specific is performed 0x01 - MONITOR – the KIDWeb will only monitor the staging data and store it in NVRAM 0x02 - CONTROLLER – the KIDWeb will monitor and acknowledge staging data. Output Format:</p>

	<p><i>0x02 O.K<CR><LF></i></p> <p>Note: the default behavior is 0x00 – OFF</p> <p>Note2: When the module starts, this setting is set to OFF</p>
<i>“wss {arg}”</i>	<p>KIDWEB2 ONLY STAGING_COMMAND! Sets the current staging behavior. If {arg} is omitted or invalid, it is assumed to be ZERO. When the command executes, if {arg} is reasonable, the KIDWeb outputs the new setting. Valid commands are:</p> <p>“wss” – set staging config to OFF</p> <p>“wss 0” – set staging config to OFF</p> <p>“wss 1” – set staging config to MONITOR</p> <p>“wss 2” – set staging config to CONTROLLER</p> <p>“wss 02” – set staging config to CONTROLLER</p> <p>Output Format:</p> <p><i>0x02 O.K<CR><LF></i></p> <p>The above means that the config was successfully changed to CONTROLLER.</p> <p>Output Format:</p> <p><i>Err.<CR><LF></i></p> <p>The above means that {arg} was an invalid number or greater than 2.</p> <p>Note: On startup, this setting is OFF, so for any application that needs this behavior, the user software must make sure it sets this setting after every KIDWeb reboot.</p>
<i>“fdelr”</i>	<p>KIDWEB2 ONLY Deletes and reinitiates the Nonvolatile RAM area.</p> <p>Output Format:</p> <p><i>O.K<CR><LF></i></p>
<i>“fdelf”</i>	<p>KIDWEB2 ONLY Deletes the file system</p> <p>Output Format:</p> <p><i>O.K<CR><LF></i></p>
<i>“fdeli”</i>	<p>KIDWEB2 ONLY Deletes the KIDWeb module info like IP address, baud rate, boot-up behavior, etc....</p> <p>Output Format:</p> <p><i>O.K<CR><LF></i></p>
<i>“n”</i>	<p>NOP. The KIDWeb will close its telnet connection after about 30 seconds of inactivity. You should send this command every 15 seconds or so in order to keep the connection alive.</p>

Appendix A: Monitoring of CAN messages using UDP

As of revision 4.1.2, there is a telnet command that sets if the UDP packets should be sent to the broadcast address or only to the PC that last registered a COB-ID for monitoring. The correct procedure for using the “gb” command is as follows:

1. “gb 0” or “gb 1” to tell the KIDWeb whether it should send messages as broadcasts or to one host only. When using those commands, the KIDWeb automatically clears the list of monitored COB-IDs.
2. Register all the COB-IDs that need to be monitored with the “gas” or “gam” commands.

As of revision 3.2, the KIDWeb can keep a list of COB-IDs that if encountered on the CAN side, should be sent on the Ethernet side. This enables applications that have access to the Ethernet side of the KIDWeb to ‘register’ the COB-IDs that they are interested in and then listen for broadcast

UDP messages from KIDWeb. The KIDWeb on his behalf is responsible to gather the messages with those COB-IDs and send the message over UDP. The KIDWeb will send broadcast UDP packets with a destination port set to 0x7777 (30583 decimal). The source port in the packet is 0x4444 (17476 decimal) although this should not be relevant to most implementations. The COB-IDs that may be registered in the KIDWeb are from 0x181 to 0x57F (CANOpen PDO range). Any combination in the above range may be used (up to 1023 COB-IDs).

Command Syntax: “**gc**” – (**G**roup **C**lear) clears the list of registered COB_IDs

Command Syntax: “**gas COB_ID**” – (**G**roup **A**dd **S**ingle) instructs the KIDWeb to register this cob-id in its internal tables, and upon reception of a message with this cob-id, to transmit the message over UDP.

Example 1: “gas 232” – instruct the KIDWeb to monitor all the messages with COB-ID: 0x232

Command Syntax: “**gam COB_ID_1 COB_ID_2 COB_ID_n**” – (**G**roup **A**dd **M**ultiple) same as the above, but registers multiple COB-IDs with a single command.

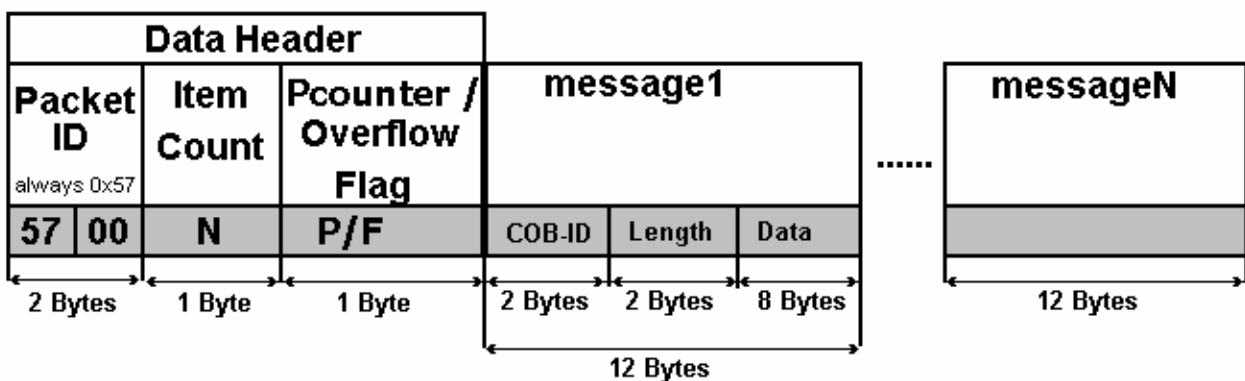
Example 2: “gam 182 300 575” – instruct the KidWEB to monitor all the messages with COB-IDs: 0x180, 0x300, 0x575

Note: if the messages are encountered relatively rarely, the KIDWeb will transmit a UDP packet for every message. If however the a couple of messages arrive in a very short time, and have been registered for monitoring, the KIDWeb will collect and send them in a single UDP packet. If too many packets are received on the CAN side too quickly, the KIDWeb will rise the OVERFLOW flag in the UDP packet and transmit as much data as it was able to collect (currently up to 40 messages can be gathered and sent on a single UDP packet).

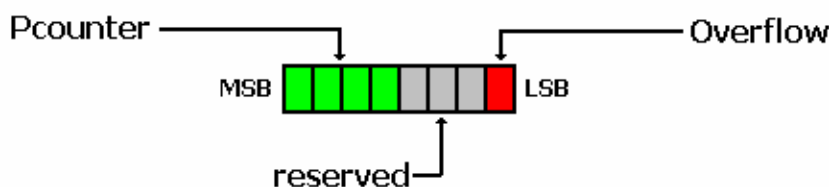
The PacketCounter/Overflow is one byte. The least significant bit represents the overflow flag, while the high nibble (4bits) holds the packet counter. The packet counter is incremented every time a new packet is sent. When it reaches 15, it overflows and starts from 0 again. If the CAN side of the module is not under heavy load, the KIDWeb will retransmit the last packet twice. In this case, the PCounter will NOT change. This can be used to identify if a packet is a retransmit, or if it holds new information. If the PCounter is different from the last packet you received, the packet holds new data that was collected from the CAN network.

UDP packet structure for CAN monitoring over UDP

UDP Data Format All numbers are in HEX

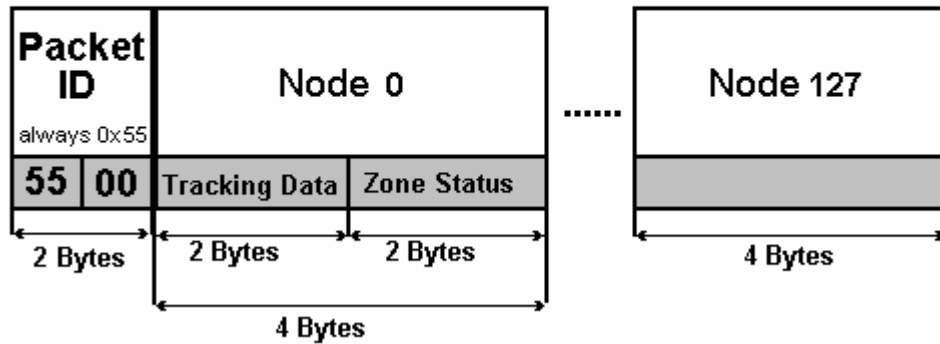


PCounter and Overflow BYTE structure



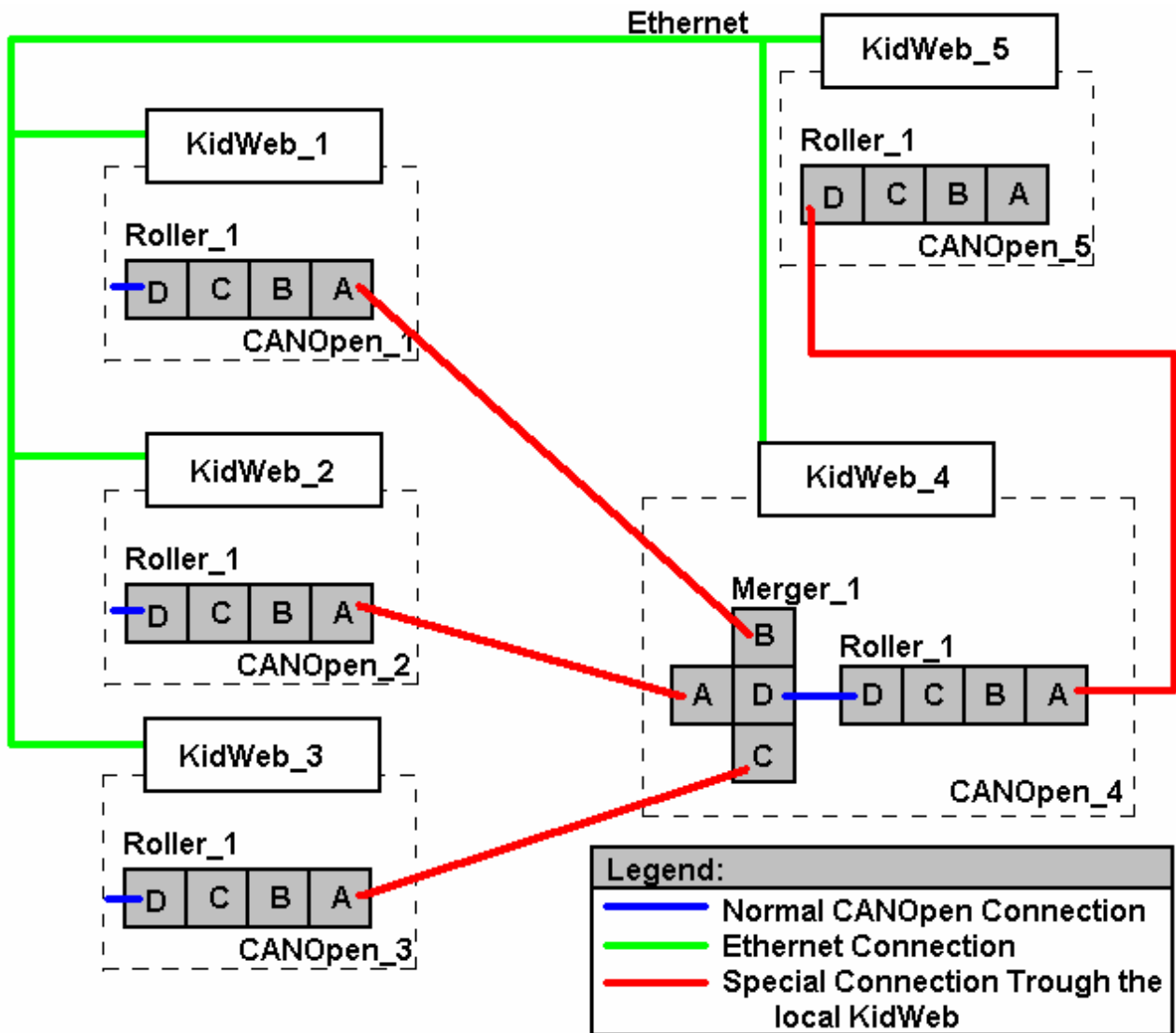
Appendix B: UDP broadcast packets of monitoring the status of the modules on the CAN side.

UDP Data Format
All numbers are in HEX



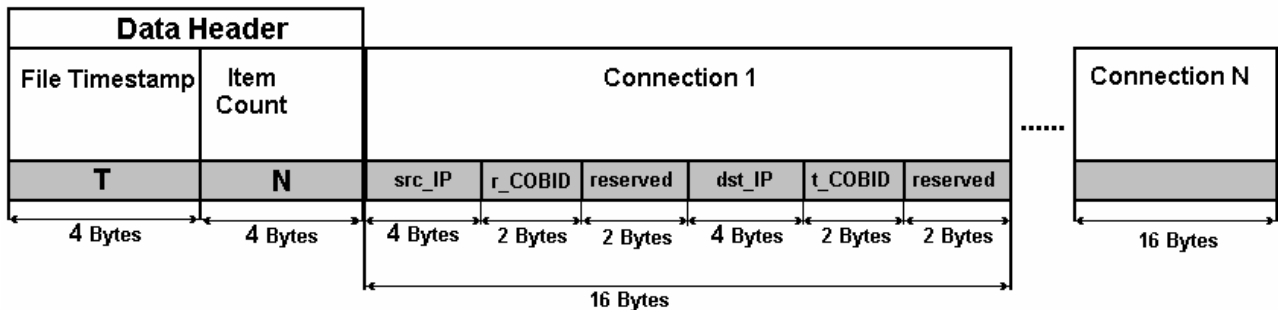
Appendix C: Inter-KidWeb Connections

As of revision 3.3 of the KIDWEB firmware and rev. 2.1 of “RollOn CANOpen Configuration Tool” there is the concept of connecting modules across KIDWeb devices. The idea is this: one part of a conveyor system can be connected to one KIDWeb, while another part can be connected to a different KIDWeb. The two or more separate CANOpen networks can now communicate among each other through their respective KIDWeb. The connection can be established as long as the KIDWeb devices are all connected to the same Ethernet. The above described configuration may be depicted in the diagram below.



The Special Connections (depicted as thick RED lines) are described in a special CONN.BIN file and the file is downloaded to all the KIDWeb devices that take part in the connection scheme. The CONN.BIN file folds all the connections and one and the same file is downloaded to all the KIDWebs. The file also holds a timestamp, to help distinguish between KIDWebs with old and new configurations. The format of the file is shown below.

CONN.BIN Internal Structure

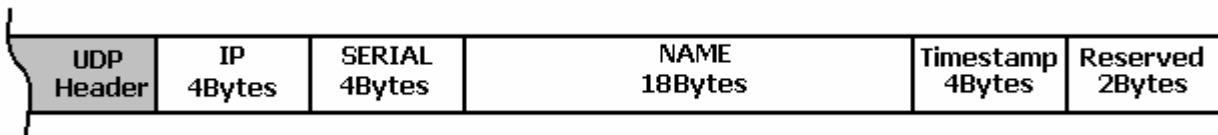


Dealing with IP addresses

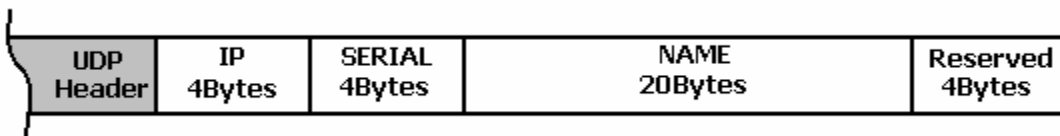
Each KIDWeb device comes with its own IP and MAC address. The MAC address is unique and cannot be changed. The IP address identifies the KIDWeb on the Ethernet and is vital that it is set unique on the network. Changing the IP address of the KIDWeb is a two step process. First, a “discovery” packet is sent on the network. All the KIDWeb devices on the network respond to this message with their serial number and their IP. The second step of setting the IP of a device involves sending a special packet to this device.

The discover packet is a UDP broadcast packet (destination address 255.255.255.255) with a source port 0x3333 and destination port 0x2222. The packet must contain 12 bytes of data, all holding the value of 0xFF.

Every KIDWeb module that hears this packet responds with a unicast packet. The packet contains the KIDWeb’s IP address, serial number, name, and the timestamp of the connection file if one is present.



Setting the IP and or name of a KIDWeb. The SET-IP packet is the same as the discover packet with respect to destination address and port numbers. The format is as follows:



This time, the PC fills in the desired IP address, serial number and name. The IP address and name are the new values for the KIDWeb and the serial number field holds the serial number of the target KIDWeb. This allows the packet to be a broadcast again since the PC and KIDWeb may not be in the same logical network. Of all the KIDWebs that hear this SET-IP packet, only the one that sees its own serial number in the packet will handle the packet. All other KIDWebs will discard the packet without any consecutive actions taken. The KIDWeb that recognizes its serial number and handles the data, will respond to the PC with its new IP address and name just as if it has received a discover packet.

The IP address is always coded in BIG ENDIAN format, while the serial number and timestamp are in LITTLE ENDIAN.