Tibbo Serial-over-IP Solutions: Hardware, Firmware, PC software

Table of Contents

Introduction 1

Legal Information 1

Hardware Manuals 3

Modules 3

EM100 Ethernet Module 3
I/O Pin Assignment and Pin Functions 4
Ethernet Port Lines 5
Serial Port and General-Purpose I/O Lines 5
LED Lines 6
Power, Reset, and Mode Selection Lines 7
Mechanical Dimensions 9
Specifications and EM100 Modifications 9
EM120 Ethernet Module 10
I/O Pin Assignment and Pin Functions 11
Ethernet Port Lines 11
Serial Port and General-Purpose I/O Lines 12
LED Lines 13
Power, Reset, and Mode Selection Lines 14
Mechanical Dimensions 15
Specifications and EM120 Modifications 15
EM200 Ethernet Module 16
I/O Pin Assignment and Pin Functions 17
Ethernet Port Lines 17
Serial Port and General-Purpose I/O Lines 19
LED Lines 20
Power, Reset, and Mode Selection Lines 21
Mechanical Dimensions 22
Specifications and EM200 Modifications 22
EM203A Ethernet-to-Serial Module 23
I/O Pin Assignment and Pin Functions 24
Serial Port and General-Purpose I/O Lines 25
Ethernet Lines and Jack/Magnetics Data 27
LED Lines 28
Power, Reset, and Mode Selection Lines 28
Onboard LEDs 30
Thermal considerations 30
Mechanical Dimensions 32
Specifications and Ordering Info 32
EM203 Ethernet-to-Serial Module 34
I/O Pin Assignment and Pin Functions 35
Serial Port and General-Purpose I/O Lines 36
Ethernet Lines and Jack/Magnetics Data 38
LED Lines 39
Power, Reset, and Mode Selection Lines 40
Onboard LEDs 41
Thermal considerations 42
Mechanical Dimensions 43
Specifications and Ordering Info 44
RJ203A Jack/Magnetics Module 45
Interface Pads 46
Interfacing the RJ203A to the DM9000A 46

Using the RJ203A With the EM203A Module ... 48
Mechanical Dimensions: RJ203A ... 49
Mechanical Dimensions: RJ203A+EM203A ... 50
Specifications and Ordering Info ... 51
RJ203 Jack/Magnetics Module ... 52
Interface Pads ... 52
Interfacing the RJ203 to the DM9000B ... 53
Using the RJ203 With the EM203 Module ... 54
Mechanical Dimensions: RJ203 ... 55
Mechanical Dimensions: RJ203+EM203 ... 56
Specifications and Ordering Info ... 57
Boards ... 57
EM100-EV Evaluation Board ... 58
Power Jack ... 58
Ethernet Port Pin Assignment ... 59
RS232 Port Pin Assignment ... 59
EM120/EM200-EV Evaluation Board ... 60
Power Jack ... 60
Ethernet Port Pin Assignment ... 61
RS232 Port Pin Assignment ... 61
Expansion Connector Pin Assignment ... 62
Device Servers and Controllers ... 63
DS100 Serial Device Server ... 63
DS100 Connectors and Controls ... 64
Power Jack ... 64
Ethernet Port Pin Assignment ... 64
Serial Port Pin Assignment and i/f Selection ... 65
Status LEDs ... 66
Setup Button ... 66
Specifications and DS100 modifications ... 67
DS203 Serial Device Server ... 68
DS203 Connectors and Controls ... 69
Power Jack ... 69
Ethernet Port Pin Assignment ... 70
RS232 Port Pin Assignment ... 70
Status LEDs ... 71
Setup Button ... 71
Specifications and DS203 Modifications ... 71
Kits and Accessories ... 72
WAS-P0004(B) DS-to-Device Serial Cable ... 74
WAS-1404 DS-to-Device Serial Cable ... 74
WAS-P0005(B) DS-to-PC Serial Cable ... 74
WAS-1455 DS-to-PC Serial Cable ... 75
WAS-1499 'Straight' Ethernet Cable ... 75
WAS-1498 'Crossover' Ethernet Cable ... 75
TB100 Terminal Block Adaptor ... 76
12VDC Power Adaptors ... 77
DMK100 DIN Rail/ Wall Mounting Kit ... 77
Firmware Manuals 80
Monitor (for Serial Firmware Upgrades) ... 80
Status LED Signals ... 81
DS Startup Procedure ... 82
Serial Upgrade Mode ... 82
Device Server Application Firmware (V3.34/V3.66) ... 83
Status LED Signals ... 84

Operation **86**
Ethernet Port and Network Communications 87
UDP Data 'Connections' 88
Broadcast UDP Communications 89
TCP Data Connections 89
DHCP 90
PPPoE [V3.54+] 91
Link Server Support 92
DS Powerup Procedure 92
Data Connection Establishment Procedure 94
Serial Port and Serial Communications 96
Opened and Closed States of the Serial Port 97
Data Routing 98
Serial-to-Ethernet Data Routing 98
Programming **100**
Serial Programming 100
Network Programming 101
Out-of-Band (UDP) Programming 102
Broadcast Out-of-Band (UDP) Programming 103
Inband (TCP) Programming 103
Command-Phase (TCP) Programming 104
Telnet TCP Programming [V3.50 and Above] 105
Authentication 106
Programming Priorities 107
Error Mode 108
Quick Initialization 109
Custom Profiles 109
Reference **109**
Commands, messages, and replies 110
Command & message description format 111
Login (L) command 111
Logout (O) command 112
Reboot (E) command 113
Initialize (I) command 114
Set Setting (S) command 115
Get Setting (G) command 115
Parameter (P) command 116
Echo (X) command 117
Status (U) command 120
Buzz (B) command 122
Reset Overflow Flags (R) command 123
Assign IP-address (A) command 123
Select In Broadcast Mode (W) command 124
Get Firmware Version (V) command 125
Jump To Netloader (N) command [Release3.0] 125
Set Programming Request Flag (N) command [Release 3.5] 126
Reset Upload Process (Q) command [Release 3.5] 127
Upload Data Block (D) command [Release 3.5] 128
Cable Status (C) command 129
Get My IP (T) command 129
Notification (J) message 129
Settings 131
Setting description format 131
Network Settings 132
Owner Name (ON) setting 133
Device Name (DN) setting 133
MAC-address (FE) setting 134
DHCP (DH) setting 135
IP-address (IP) setting 135

Port Number (PN) setting 136
dDNS Service Registration (DD) setting 137
dDNS Service IP-address (LI) setting 138
dDNS Service Port (LP) setting 138
LS Auto-registration (AR) setting 139
PPPoE Mode (PP) setting [V3.54+] 140
PPPoE Login Name (PL) setting [V3.54+] 141
PPPoE Login Password (PD) setting [V3.54+] 141
Gateway IP-address (GI) setting 142
Netmask (NM) setting 143
Password (PW) setting 144
Connection Settings 145
Connection Timeout (CT) setting 145
Transport Protocol (TP) setting 146
Broadcast UDP (BU) setting 147
Link Service Login (TL) setting 147
Inband Commands (IB) setting 148
Data Login (DL) setting 149
Retransmission Period (RP) setting 149
Routing Mode (RM) setting 150
Source IP Filtering (SF) setting 151
Connection Mode (CM) setting 152
Destination IP-address (DI) setting 153
Destination Port Number (DP) setting 154
Notification Destination (ND) setting 155
Serial Settings 156
Serial Interface (SI) setting 157
Flow Control (FC) setting 159
DTR Mode (DT) setting 160
DTR Startup Mode (DS) setting 160
Baudrate (BR) setting 161
Parity (PR) setting 162
Bits Per Byte (BB) setting 162
Soft Entry (SE) setting 163
Escape Character (EC) setting 165
On-the-fly Commands (RC) setting 165
On-the-fly Password (OP) setting 166
Notification Bitmask (NB) setting 167
Encapsulation Settings 168
Maximum Packet Length (ML) setting 169
Maximum Intercharacter Delay (MD) setting 169
Start On Any Character (SA) setting 170
Use Start Character (F1) setting 171
Start Character Code (S1) setting 171
Use Stop Character (U1) setting 172
Stop Character Code (E1) setting 173
Number Of Post Characters (P1) setting 173
Parameters and Instructions 174
Parameter & Instruction Description Format 174
On-the-fly (Network-Side) Parameters & Instructions 175
Flow Control (FC) parameter 175
Baudrate (BR) parameter 176
Parity (PR) parameter 176
Bits Per Byte (BB) parameter 177
Notification Bitmask (NB) parameter 177
Get I/O Pin Status (Gx) instruction 178
Set I/O Pin Status (Sx) instruction 179
Modem (Serial-Side) Parameters & Instructions 181
Transport Protocol (TP) parameter 181

Link Server Login (TL) parameter ... 182
Routing Mode (RM) parameter ... 182
Source IP Filtering (SF) parameter ... 182
Destination IP-address (DI) parameter ... 183
Destination Port Number (DP) parameter ... 184
Establish Connection (CE) instruction ... 184
Close Connection (CC) instruction ... 185
Abort Connection (CA) instruction ... 186
NetLoader (For Network Firmware Upgrades, V1.10) ... 187
Status LED Signals ... 187
MAC- and IP-address Under the NetLoader ... 188
NetLoader Communications ... 188
How the NetLoader is Started ... 189
Firmware Upload Procedure ... 189
Available Commands ... 190
Command Description Format ... 191
Echo (X) command ... 191
Select In Broadcast Mode (W) command ... 192
Start Over (Q) command ... 193
Data Block (D) command ... 193
Verify And Start Firmware (E) command ... 194
Get Firmware Version (V) command ... 195
Software Manuals 195
Device Server Toolkit (DST) Software for Windows (Release 3.9.82) ... 196
DS Manager ... 197
DS Status Icons ... 198
Access Modes ... 200
Auto-Discovery Access Mode ... 201
Broadcast Access ... 203
Troubleshooting (Auto-Discovery Mode) ... 204
Address Book Access Mode ... 205
Access Parameters for the Address Book Mode ... 207
Preparing the DS for Inband Access ... 208
Troubleshooting (Address Book Mode) ... 209
Serial Access Mode ... 212
Troubleshooting (Serial Access Mode) ... 212
Functions ... 213
Editing DS Settings (Settings Button) ... 213
Upgrading DS Firmware (Upgrade Button) ... 215
Initializing the DS (Initialize Button) ... 216
Monitoring DS Status (Routing Status Button) ... 217
"Buzzing" the DS (Buzz Button) ... 219
Changing IP-address (Change IP Button) ... 219
Finding a DS on the list (Find Button) ... 220
Editing the Address Book (Add, Remove, Edit Buttons) ... 220
Managing Address Book Groups (Groups button) ... 221
Warnings And Messages ... 222
DS Status Messages ... 222
Error Mode ... 222
Firmware Upgrade Mode ... 222
IP-address Not Obtained ... 223
No Response ... 223
No Status Info Available ... 224
Programming in Progress ... 224
Routing Buffer Overflow ... 225
Unknown Device ... 225

Unreachable IP-address ... 225
Warning and Error Messages ... 226
Broadcast Access Not Supported ... 226
Could Not Connect to the DS (Inband Access) ... 226
Data Link In Progress ... 227
DS Lost (After Changing IP) ... 227
DS Lost (After Entering NetLoader) ... 228
DS Lost (After Exiting NetLoader) ... 228
DS Lost (After Initialization) ... 229
Duplicate Address Book Entry ... 229
Error Mode (Initialization Required) ... 229
Failed to Start the NetLoader ... 230
Function Not Available (Upgrade Mode) ... 230
Function Not Supported ... 231
Incorrect Password ... 231
Initialization Failed ... 232
Initialization Not Allowed ... 232
Initialization Required ... 233
Input Login Password ... 233
Invalid Firmware File (or Comm Error) ... 233
Invalid Firmware Uploaded ... 234
Invalid IP-address ... 234
Network Firmware Upload Aborted ... 235
New Password Not Accepted ... 235
No Response From the DS (Network) ... 236
No Response From the DS (Serial) ... 236
Operation Cannot Be Executed ... 236
Out-of-Band (UDP) Access Required ... 237
Password Will Be Disabled ... 237
Potential DHCP Conflict ... 237
Power Up With Setup Button Pressed ... 238
Press Setup Button ... 239
Serial Firmware Upload Failed ... 239
Serial Upgrade Completed ... 239
Setting Description File Error ... 240
Unable to Find Setting Description Files ... 240
Unexpected NetLoader Error ... 240
Unreachable IP-address ... 241
Unable to Send a Broadcast ... 241
VSPD and VSP Manager ... 241
How VSP Works ... 242
VSP Manager ... 243
VSP Properties ... 244
VSP Name Selection ... 245
Transport Protocol ... 246
Additional Info on UDP and TCP Connections ... 247
On-the-fly Commands ... 248
When the VSP Sends On-the-fly Commands ... 249
Handling of RTS, CTS, DTR, and DSR Signals ... 251
Synchronization Issues for On-the-fly Commands ... 253
Disabled (With FF Escape) Mode of the VSP ... 255
Connection Timeout ... 256
Routing Mode ... 256
Connection Mode ... 256
Listening Port ... 257
Destination Modes ... 257
Single-destination Mode ... 258
Multi-destination Mode ... 259
Specify Destination By ... 263

Control lines Tab ... 263
Default serial settings Tab ... 264
Use WinSock for transport ... 265
Unsupported Features and Limitations ... 265
Warnings and Messages ... 266
Listening Port is In Use ... 266
No Start Characters defined ... 266
Port Substitution Required ... 266
VSP Is in Use ... 267
Port Monitor ... 267
Preferences Dialog (General Tab) ... 268
Preferences Dialog (Event Tab) ... 269
Data Dump Feature ... 270
Connection Wizard ... 271
Choosing the Wizard Job ... 272
VSP-to-DS Link ... 272
VSP Name Selection ... 273
Choosing the VSP Name ... 273
Target DS ... 274
Additional Info on Accessing the DS ... 276
Initiator of the Data Exchange ... 277
Netmask & Gateway for the DS ... 278
How the Wizard Decides Who Opens Connections ... 279
Transport Protocol & Listening Ports ... 281
On-the-fly Commands ... 283
Parameters for the Serial Port of the DS ... 285
Programming Inaccessible DS ... 286
Application-to-DS Link ... 287
Target DS ... 288
Additional Info on Accessing the DS ... 289
Initiator of The Data Exchange ... 290
Netmask & Gateway for the DS ... 291
How the Wizard Decides Who Opens Connections ... 292
Transport Protocol & Listening Ports ... 294
Parameters for the Serial port of the DS ... 297
Programming an Inaccessible DS ... 297
DS-to-DS Link ... 298
DS #1 (Must be Local) ... 299
DS #2 (Can be Local or Remote) ... 300
Additional Info on Accessing the DS ... 301
Initiator of the Data Exchange ... 302
Netmask & Gateway for the DS #1 ... 303
Netmask & Gateway for the DS #2 ... 304
How the Wizard Decides Who Opens the Connection ... 305
Transport Protocol & Listening Ports ... 307
Remote Exchange for RTS, CTS, DTR, DSR ... 309
Parameters for the Serial Port of the DS #1 ... 310
Parameters for the Serial Port of the DS #2 ... 311
Programming an Inaccessible DS ... 311
Finishing a Remote Job ... 312
Programming Method for the DS ... 313
Configuration Script File ... 314
Reviewing Setup Details ... 315
Final Screen ... 315
Warnings and Messages ... 316
Inband Access for Local DS ... 316
MAC-->IP Mapping Advised ... 317
DS #1 Must be Local ... 317
Non-zero Port Number Required ... 317

Port is in Use ... 317
VSP is Opened by Another Application ... 318
Different IP-address Required ... 318
Unable to Find Setting Description File ... 318
Invalid IP-address ... 318
The DS is Not Local ... 319
Avoid Data Characters with ASCII code FF ... 319
Unable to Send a Broadcast ... 320
DST Revision History ... 320
Virtual Serial Port Driver for Linux (VSPDL) ... 321
How VSP Works ... 322
VSPDL Device Files ... 323
Installation ... 323
Controlling VSPDL Operation ... 325
VSPDL Configuration File (vspd.conf) ... 325
Sample Configuration File ... 326
General VSPDL Configuration ... 327
Root Directory for the Daemon (<root> Section) ... 328
Path and Prefix for Device Files (<devprefix> Section) ... 328
Host to Bind By Default (<bind> Section) ... 328
Timeout for Basic I/O Operations (<timeout> Section) ... 328
VSPdaemon Event Logging Configuration (<log> Section) ... 329
VSP Configuration (<vsp> Section) ... 329
VSP Number (<vsp> Section) ... 330
Host and Port to Bind for the VSP (<bind> Section) ... 330
Bind Host (<host> Parameter) ... 330
Bind Port (<port> Parameter) ... 331
Connection Parameters (<connection> Section) ... 331
Routing Mode (<connection rmode>) ... 332
Transport Protocol (<connection proto>) ... 332
Additional Info on UDP and TCP Connections ... 333
Connection Mode (<connection conmode>) ... 334
Connection Timeout (<connection timeout>) ... 335
On-the-fly Commands (<connection onthefly>) ... 335
Password for On-the-fly Commands (<connection clogin>) ... 337
Data Login (<connection dlogin>) ... 337
Destination Device Parameters (<destination> section) ... 337
Destination IP-address (<ip> parameter) ... 338
Destination MAC-address (<mac> parameter) ... 338
Destination Data Port (<port> parameter) ... 339
Destination Command Port (<cport> parameter) ... 339
Outbound Packet Generation Options (<packets> section) ... 339
Maximum Packet Length (<maxlen> parameter) ... 340
Maximum Intercharacter Delay (<maxdelay> parameter) ... 341
Start on Any Character (<starton> parameter) ... 341
Start Character (<startchar> parameter) ... 341
Stop Character (<stopchar> parameter) ... 342
Event Logging Configuration for the VSP (<log> section) ... 342
Data Dump (<dump> section) ... 343

Application Notes 343

AN001. Customization Options in Our Products ... 343
AN002. Practical Advice on Integrating EM Module into Your Device ... 349
Example: PIC with EM203 ... 354
AN003. Time Delays When the DS is Opening TCP Connection to the PC ... 355
AN004. How to Send the Same Data to Several DS ... 358

AN005. Remotely Controlling I/O Lines on the DS ... 359
AN006. Using Device Server Toolkit with Windows Firewall (XP/SP2) ... 362
AN007. Installing and Configuring LinkServer ... 367
Preparing your Network (Router Configuration) ... 367
Downloading and Installing Software and Firmware ... 368
Creating Trial AuthKey ... 369
Initial LinkServer Configuration ... 371
Creating a User Account ... 372
Adding a DS as User ... 373
Configuring a Device Server ... 373
Configuring a Virtual Serial Port ... 376
Testing with HyperTerminal ... 377
AN008. Using HyperTerminal ... 379
What is it Good For? ... 380
Running HyperTerminal ... 381
Setting Correct Parameters on Startup ... 381
Establishing a Serial Connection with a Device Server ... 383
Establishing a Connection Through a Virtual Serial Port ... 384
Establishing a TCP/IP Connection with a Device Server ... 390
Setting Optional Parameters ... 396
Using HyperTerminal to Test a Connection ... 397
Two HyperTerminal Windows, one DS ... 398
Creating a Loopback for Testing ... 398
Sending Commands Using HyperTerminal or Telnet ... 399
Sending a Command (Command Format) ... 399
Serial Programming ... 400
Serial Parameters (Modem Commands) ... 400
Telnet Programming ... 403
Command-Phase TCP ... 404
Programming Exercise ... 405
AN009. WAN Basics ... 406
What Is a WAN ... 407
What Is a LAN ... 408
Subnets ... 409
Internal and External Addresses ... 410
Dynamic and Static Addresses ... 411
What Is a Gateway ... 412
Network Address Translation ... 413
How NAT Applies To Device Servers ... 416
Port Forwarding ... 418
AN010. Controlling the DS from the Serial Side ... 419
Benefits of Modem Commands ... 420
Issuing Commands ... 421
Establishing a Connection ... 422
Terminating a Connection ... 423
Finding Out Connection Status (X) ... 423
Finding Out Connection Details (U) ... 425
Exiting Serial Programming Mode ... 426
DSR/DTR ... 426
Real-World Example ... 427
AN011, Reading the Production Label ... 428
AN012, Creating an Integrated Power Supply ... 428
Update history 429

Introduction

Last update: 14OCT2008

Legal Information

Manual Update History

WELCOME to Tibbo Serial-over-IP Solutions Manual!

This Manual consists of four parts:

- **Hardware Manuals** describe the hardware of Tibbo Device Servers
- **Firmware Manuals** describe internal software (called "firmware") of Tibbo Device Servers
- **Software Manuals** describe available PC software
- **Application Notes** part is a collection of articles on practical use of Tibbo Device Servers

Legal Information

Please read and understand the following important legal information and disclaimer:

Tibbo Technology ("TIBBO") is a Taiwan corporation that designs and/or manufactures a number of hardware products, software products, and applications ("PRODUCTS"). TIBBO PRODUCT range includes BASIC-programmable devices ("PROGRAMMABLE DEVICES") that can run a variety of applications written in Tibbo BASIC ("BASIC APPLICATIONS").

As a precondition to your purchase and/or use of any TIBBO PRODUCT, including PROGRAMMABLE DEVICES, you acknowledge and agree to the following:

1. Tibbo does not have any branch office, affiliated company, or any other form of presence in any other jurisdiction. TIBBO customers, partners and distributors in Taiwan and other countries are independent commercial entities and TIBBO does not indemnify such customers, partners or distributors in any legal proceedings related to, nor accepts any liability for damages resulting from the creation, manufacture, importation, advertisement, resale, or use of any of its PRODUCTS.
2. TIBBO reserves the right to halt the production or availability of any of its PRODUCTS at any time and without prior notice. The availability of a particular PRODUCT in the past is not an indication of the future availability of this PRODUCT. The sale of the PRODUCT to you is solely at TIBBO's discretion and any such sale can be declined without explanation. If, for whatever reason, we are unable or unwilling to deliver the goods you have already paid for, we will offer a refund that shall not exceed original purchase price.
3. All specifications and information provided in this Manual are subject to change without prior notice. You agree that it is your responsibility to test all shipments of TIBBO PRODUCTS to determine suitability for your needs. TIBBO assumes no responsibility for any errors which may appear in this Manual, and does not

make any commitment to update the information contained herein.

4. TIBBO makes no warranty for the use of its PRODUCTS, other than that expressly contained in the Standard Warranty located on the Company's website. Your use of TIBBO PRODUCTS is at your sole risk. TIBBO PRODUCTS are provided on an "as is" and "as available" basis. TIBBO expressly disclaims the warranties of merchantability, future availability, fitness for a particular purpose and non-infringement. No advice or information, whether oral or written, obtained by you from TIBBO shall create any warranty not expressly stated in the Standard Warranty.
5. Many TIBBO PRODUCTS can and are routinely combined with other hardware or software, either supplied by TIBBO or any third party, to create a combinatorial product or system ("COMBINATORIAL PRODUCT/SYSTEM") where TIBBO PRODUCT constitutes only a portion of such COMBINATORIAL PRODUCT/ SYSTEM. Combining a particular TIBBO PRODUCT with other hardware or software, either supplied by TIBBO or any third party, may potentially create a COMBINATORIAL PRODUCT/SYSTEM that violates local rules, regulations, and/ or infringes an existing patent, trademark or copyright in a country where such combination has occurred or where the resulting COMBINATORIAL PRODUCT/ SYSTEM is manufactured, exported, or sold. TIBBO is not capable of monitoring any activities by its customers, partners or distributors aimed at creating any COMBINATORIAL PRODUCT/SYSTEM, does not provide advice on potential legal issues arising from creating such COMBINATORIAL PRODUCT/SYSTEM, nor explicitly recommends the use of any of its PRODUCTS in combination with any other hardware or software, either supplied by TIBBO or any third party.
6. Combining a particular PROGRAMMABLE DEVICE with a specific BASIC APPLICATION, either written by TIBBO or any third party, may potentially create an end product (”END PRODUCT”) that violates local rules, regulations, and/or infringes an existing patent, trademark or copyright in a country where such combination has occurred or where the resulting END PRODUCT is manufactured, exported, or sold. TIBBO is not capable of monitoring any activities by its customers, partners or distributors aimed at creating any END PRODUCT, does not provide advice on potential legal issues arising from creating such END PRODUCT, nor explicitly recommends the use of any of its PROGRAMMABLE DEVICES in combination with any BASIC APPLICATION, either written by TIBBO or any third party.
7. **Limitation of liability.** By using TIBBO PRODUCTS your expressly agree that TIBBO shall not be liable (to the fullest extent permitted by the law) to you for any direct, indirect, incidental, special, consequential or exemplary damages, including, but not limited to, damages for loss of profits, goodwill, or other intangible losses (even if TIBBO has been advised of the possibility of such damages) resulting from the use or the inability to use any TIBBO PRODUCT.
8. **Patent and copyright infringement.**

   A. By purchasing any TIBBO PRODUCT, you agree and acknowledge that the design of all aspects of any COMBINATORIAL PRODUCT/SYSTEM or END PRODUCT is solely your responsibility. You agree that TIBBO shall have no obligation to indemnify or defend you in the event that a third party asserts that your COMBINATORIAL PRODUCT/SYSTEM or END PRODUCT violates third party patents, copyrights, or other proprietary rights.

   B. You waive any right to cause TIBBO to defend or indemnify you or any of your customers in connection with a demand related to TIBBO PRODUCTS, including but not limited to any such right as may be imposed or implied by law, statute, or common law.

   C. If a demand or proceeding is brought against TIBBO based on an allegation that your COMBINATORIAL PRODUCT/SYSTEM or END PRODUCT violates a

patent, copyright, database right, trademark, or other intellectual property right, you shall defend such demand of proceeding and indemnify us and hold us harmless for, from and against all damages and costs awarded against us on the same basis and subject to the same conditions as were applicable to you.

9. "Tibbo" is a registered trademark of Tibbo Technology, Inc.
10. Terms and product names in this Manual may be trademarks of others.

Hardware Manuals

This part of documentation describes all hardware supplied by Tibbo.

All hardware manufactured by Tibbo is divided into three categories:

- **Ethernet-to-serial Modules**
- **Ethernet-to-serial Boards**
- **External Device Servers**

Additionally, Tibbo supplies a number of Accessories and Kits.

Modules

This part of documentation describes Ethernet-to-serial Modules for onboard installation supplied by Tibbo.

The following Modules are currently manufactured:

- **EM100** Ethernet-to-serial converter module
- **EM120** Ethernet-to-serial converter module
- **EM200** Ethernet-to-serial converter/ BASIC-programmable controller module
- **EM203A** Ethernet-to-serial converter/ BASIC-programmable controller module
- **EM203** Ethernet-to-serial converter/ BASIC-programmable controller module
- **RJ203A** Jack/Magnetics Module
- **RJ203** Jack/Magnetics Module

EM100 Ethernet Module

The EM100 is an Ethernet Module for onboard installation. Module hardware includes one 10BaseT Ethernet port (standard Ethernet magnetics are **integrated** into the Module), one serial port (CMOS-level) with a number of additional general-purpose I/O lines, and an internal processor, whose firmware acts as a bridge between the Ethernet and serial ports. Ethernet "side" of the Module connects directly to a standard RJ45 connector. Serial "side" interfaces directly to

the serial port pins of most microcontrollers, microprocessors, UARTs, etc.

From the hardware standpoint, the EM100 can be viewed as a universal platform suitable for running a variety of network and serial communications-related applications. It is the application firmware, not hardware that gives the EM100 most of its functionality.

The Application firmware EM100 is supplied with, currently in its 3rd generation ("Release3"), turns the EM100 into a Serial Device Server used to connect serial devices to the Ethernet (TCP/IP) networks.

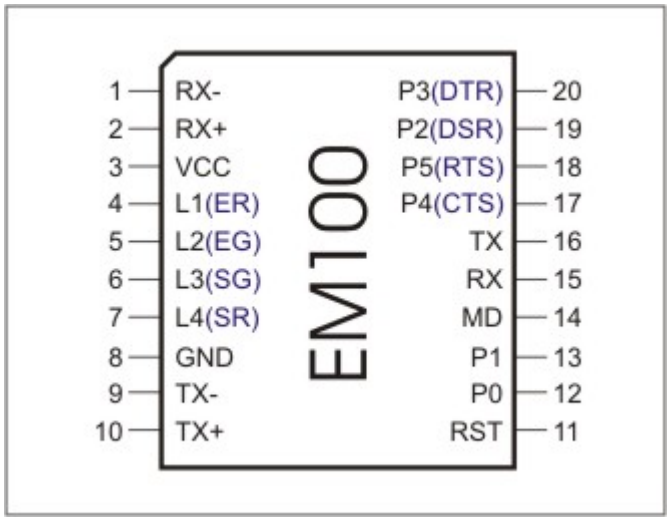
The application firmware of the EM100 can be upgraded through the module's serial port or Ethernet port*. Serial upgrades are facilitated by a so-called Monitor- a fixed "service" firmware inside the EM100. The Monitor itself cannot be upgraded. Network upgrades rely on the NetLoader firmware component that, like the application firmware itself, can be upgraded through the serial port of the EM100 (using the Monitor). The EM100 is supplied with the application firmware and the NetLoader already pre-loaded into the module.

Since most of the EM100's operation is defined by its firmware the major part of the EM100's functional description can be found in the Device Server Application Firmware Manual. This *EM100 Ethernet Module Manual* focuses on the hardware portion of the EM100.

** Network upgrades are only possible on the latest EM100-03 modification of the device (see specifications and EM100 modifications for details)*

I/O Pin Assignment and Pin Functions

EM100 pin assignment is shown below.



Click on the pin in the diagram above or one of the links provided below to learn more about EM100's I/O pins:

- Ethernet port lines
- Serial port and general-purpose I/O lines
- LED lines
- Power, reset, and mode selection lines

Ethernet Port Lines

#10	**TX+**	Output	Positive line of the differential output signal pair
#9	**TX-**	Output	Negative line of the differential output signal pair
#2	**RX+**	Input	Positive line of the differential input signal pair
#1	**RX-**	Input	Negative line of the differential input signal pair

Ethernet port of the EM100 is of 10BaseT type. The EM100 is compatible with all 10BaseT Ethernet hubs and also 99% of 100BaseT hubs. This is because most 100BaseT hubs are actually 100/10 machines that auto-detect the type of device connected to each port.

The EM100 is designed to attach directly to the RJ45 Ethernet connector. Standard magnetics circuitry (YCL part 20F001N) has been included onboard to provide a "glueless" interface to the Ethernet network.

It is important to make the PCB wire connections between the Ethernet port pins and the RJ45 as short as possible. Making the wires too long may cause the noise level generated by your PCB surpass the maximum radiated emission limits stipulated by FCC and CE regulations. Additionally, longer Ethernet lines on the PCB will make your board more susceptible to the damage from the ESD (electrostatic discharge).

Serial Port and General-Purpose I/O Lines

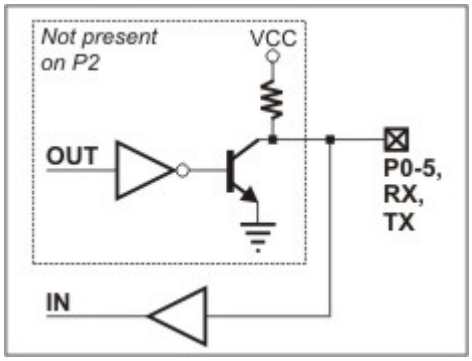
#16	**TX**	Output	Serial transmit line
#15	**RX**	Input	Serial receive line
#18	**P5** **(RTS/DIR)**	Input/output (output)	General-purpose input/output line Request to send output (full-duplex mode) Data direction control output (half-duplex mode)
#17	**P4** **(CTS/SEL)**	Input/output (input)	General-purpose input/output line Clear to send input Full-/half-duplex selection input
#20	**P3** **(DTR)**	Input/output (output)	General-purpose input/output line Data terminal ready output
#19	**P2** **(DSR)**	Input (input)	General-purpose input line Data set ready input
#13	**P1**	Input/output	General-purpose I/O line
#12	**P0**	Input/output	General-purpose I/O line

*Line functions defined by the application firmware are shown in **blue***

The EM100 features a serial port (RX, TX lines) and several general-purpose I/O lines (P0-P5). All of the above lines are of CMOS type. From the hardware point of view, all general-purpose I/O lines except P2 can serve as inputs or outputs. Line P2 can only work as an input. Maximum load current for each I/O line is 10mA.

Simplified structure of EM100's I/O lines is shown on the circuit diagram below. All

lines are "quasi-bidirectional" and can be viewed as open collector outputs with weak pull-up resistor. There is no explicit direction control. To "measure" an external signal applied to a pin the OUT line must first be set to HIGH. It is OK to drive the pin LOW externally when the pin outputs HIGH internally.



The application firmware of the EM100 maps certain serial port functions onto the general-purpose I/O pins- these functions are shown in blue in the table at the top of this topic. For example, P5 is a universal input/output but the application firmware can be set to turn this line into the RTS output of the serial port. Therefore, depending on your application you can view P5 as a general-purpose I/O line or specific control line of the serial port (RTS).

Being of CMOS type, the serial port and I/O lines of the EM100 can be connected directly to the serial port pins and I/O lines of most microcontrollers, microprocessors, etc. An interface IC* must be added to the EM100 externally if you want to connect the module to a "true" serial port (for example, COM port of the PC).

Logical signals on the serial port lines of the EM100 are active LOW. TX and RX lines are high when idle, start bit is LOW, stop bit is HIGH; LOW on CTS and RTS lines means "transmission allowed" and HIGH means "transmission not allowed". This is standard for CMOS-level serial ports and is exactly opposite to the signalling on the RS232 cables. Logical signals on the EM100 are inverted because standard interface ICs* invert the signals internally too.

As explained earlier, actual functionality of the I/O lines is firmware-dependent. See serial port and serial communications for details.

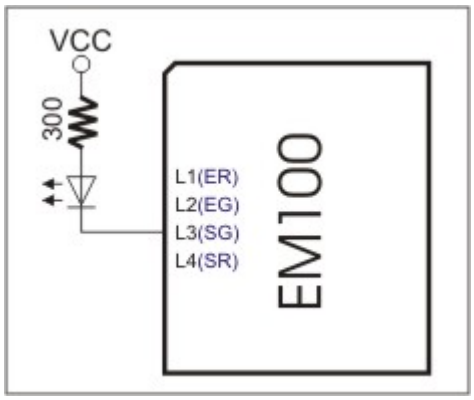
** Such as MAX232 for RS232, MAX485 for RS485, etc.*

LED Lines

#4	**L1 (ER)**	Output	LED output 1, Red Ethernet status LED
#5	**L2 (EG)**	Output	LED output 2, Green Ethernet status LED
#6	**L3 (SG)**	Output	LED output 3, Green status LED
#7	**L4 (SR)**	Output	LED output 4, Red status LED, Watchdog reset line

*Line functions defined by the application firmware are shown in **blue***

The EM100 has four LED control lines. All lines have the same internal structure and the LEDs should be connected to these lines as shown on the schematic diagram below. Maximum load for each line is 10mA.



The firmware of the EM100 assigns specific functions to these LED control lines- these functions are shown in blue in the table at the top of this topic.

ER and EG lines reflect the status of the Ethernet port. The EG LED is normally ON, and is temporarily turned off whenever the EM100 receives a network packet. The EG is normally OFF, and is temporarily turned on whenever a data collision is detected on the Ethernet*.

Additionally, ER line serves as a watchdog reset line. A very short (<10us) pulses are generated on this line at a rate of about 100Hz. When connected to the watchdog reset pin of external reset/watchdog IC, ER line keeps the watchdog "in check" preventing it from resetting the EM100. Watchdog reset pulses do not interfere with the main function of the line (that is, to indicate the status of Ethernet port). This is because the pulses are so short that they are not visible on the LED connected to the ER line.

The SR and SG LEDs display various status information depending on what firmware is running at the moment. Follow the links below to learn more about the behaviour of these LEDs under different conditions:

- SR/SG behavior in the monitor firmware.
- SR/SG behavior in the NetLoader.
- SR/SG behavior in the application firmware.

** Strictly speaking, the ER and EG lines are under firmware control. Their behavior is described here because they are always made to work as standard Ethernet status LEDs (like the ones found next to the RJ45 connector on the PC network cards).*

Power, Reset, and Mode Selection Lines

#3	**VCC**	Input	Positive power input, 5V nominal, +/- 5%, app. 40mA
#8	**GND**		Ground
#11	**RST**	Input	Reset, active high
#14	**MD (MD)**	Input	Mode selection pin

*Line functions defined by the application firmware are shown in **blue***

The EM100 should be powered from a stabilized DS power supply with output nominal voltage of 5V (+/- 5% tolerance). Current consumption of the EM100 is approximately 40mA.

Proper external reset is a must! Reset pulse should be an active HIGH. We strongly advise against using low-cost RC-networks and other unreliable methods of

generating reset pulse. Reset should be applied for as long as the power supply voltage is below 4.6V. We recommend using a dedicated reset IC with brownout detection, such as MAX810. Reset pulse length should be no less than 50ms, counting from the moment the power supply voltage exceeds 4.6V.

For added reliability, external reset IC with watchdog function may be selected. Watchdog reset pulses are provided on the ER line of the EM100.

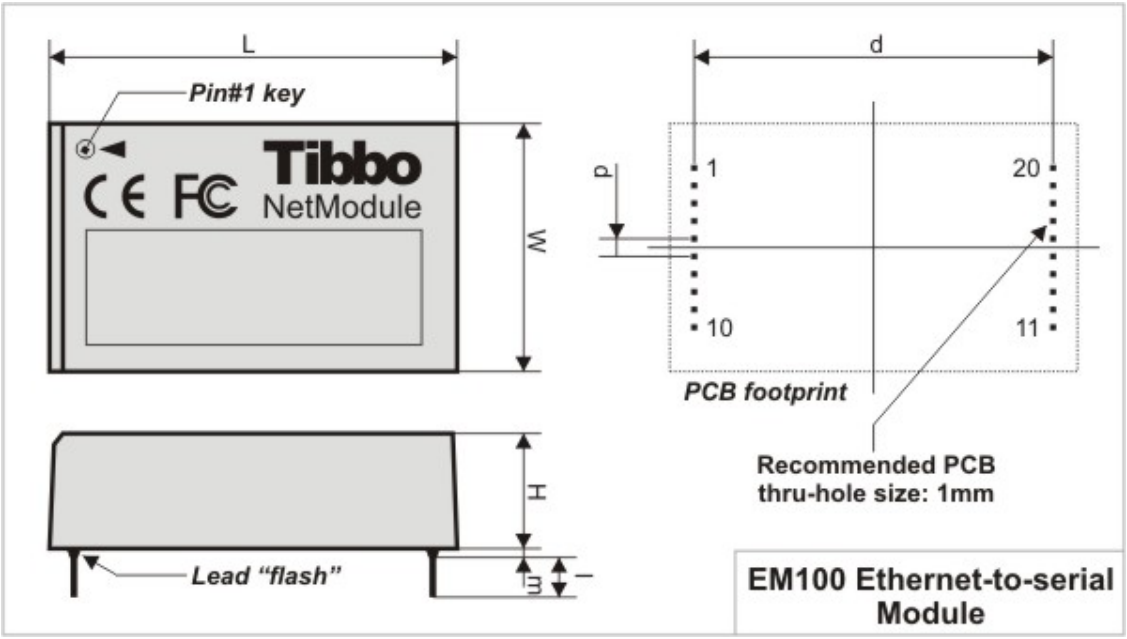
If the EM100 is used to serve as a communications co-processor in a larger system that has its own CPU it is also OK to control the RST line of the EM100 through a general-purpose I/O pin of the "host" microcontroller. I/O pins of many microcontrollers default to HIGH after the powerup and this means that the reset will be applied to the EM100 at the same time when the host microcontroller is reset. All the host microcontroller has to do is release the EM100 from reset at an appropriate time by switching the state of the I/O line to LOW.

The MD line of the EM100 is used to select the operating mode of the EM100 and/or its application firmware. The reason why the pin name is shown as MD(MD) is because the functionality of this pin is in part hardwired and in part depends on the application firmware:

- **Hardwired functionality.** When the EM100 powers up it verifies the state of the MD input. If the MD input is at HIGH the EM100 proceeds to verifying and running the application firmware loaded into its internal FLASH memory. If the MD input is at LOW the EM100 enters the serial upgrade mode. For more information see Monitor.
- **Application firmware-dependent functionality.** When the application firmware is already running the MD line is typically used to make the EM100 enter the serial programming mode. For more information see serial programming.

When the EM100 is used as a co-processor in a host system the MD line can be also controlled by the host microcontroller. Ability to control both the RST and DS lines allows the host microcontroller to switch between the operating modes of the EM100.

Mechanical Dimensions



L	Max.	46.2	Module length
W	Max.	28.0	Module width
H	Max.	13.0	Module height
I	Min.	4.5	Lead length
m	Max.	1.0	Lead "flash"
d	Aver.	40.6	Distance between lead rows
p	Aver.	2.0	Pin pitch

All dimensions are in millimeters

Specifications and EM100 Modifications

There are five different EM100 sub-models in circulation: EM100-00, EM100-01, EM100-02, EM100-03, and EM100-04. Currently, only the EM100-04 is being manufactured so the information on EM100-00...EM100-03 is provided for your reference only.

The EM100-00, EM100-01, and EM100-02 devices were basically the same, with only minor changes made to the internal hardware (such as bypass capacitors on the internal PCB, etc.). We will refer to all three modifications as the EM100-02.

The EM100-03 has extended functionality compared to the EM100-02. There are two notable differences:

- Memory size inside the device has been increased so the routing buffers of the EM100-03 are double the size of the buffers inside the EM100-02 (510 bytes in each direction vs. 255 bytes in each direction).
- Ability to upgrade the application firmware through the network was added (this is facilitated by the NetLoader firmware) to the EM100-03. The EM100-02 cannot run the NetLoader and cannot be upgraded through the network.

The EM100-04 is a RoHS-compliant version of the EM100-03. The EM100-04 and EM100-03 are identical in every other way.

Device specifications are presented in the table below.

Parameter	EM100-04
Ethernet interface	10BaseT Ethernet, standard magnetics built-in
Serial interface and I/O lines	CMOS-level; TX, RX, and 6 additional I/O lines with RTS, CTS, DTR, DSR implemented in application firmware
Routing buffers size	510 bytes x 2 (255 bytes x 2)
Maximum load current of I/O lines	10mA
Power requirements	DC 5V, +/- 5%, app. 40 mA
Operating temperature	-10 to +70 degrees C
Operating relative humidity	10-90%
Mechanical dimensions (excl. leads)	46.2x28x13mm
Packing	Tube, 10 modules/tube

Note: all specifications are for reference only. TIBBO assumes no responsibility for any errors which may appear in this Manual, and does not make any commitment to update the information contained herein.

EM120 Ethernet Module

The EM120 is an Ethernet Module for onboard installation. Module hardware includes one 10BaseT Ethernet port (standard Ethernet magnetics are **NOT integrated** into the Module), one serial port (CMOS-level) with a number of additional general-purpose I/O lines, and an internal processor, whose firmware acts as a bridge between the Ethernet and serial ports. Ethernet "side" of the Module connects directly to a standard Ethernet magnetics circuit (such as YCL-20F001N) or RJ45 connector with integrated magnetics. Serial "side" interfaces directly to the serial port pins of most microcontrollers, microprocessors, UARTs, etc.

From the hardware standpoint, the EM120 can be viewed as a universal platform suitable for running a variety of network and serial communications-related applications. It is the application firmware, not hardware that gives the EM120 most of its functionality.

The EM120 can run two distinctively different kinds of application firmware:

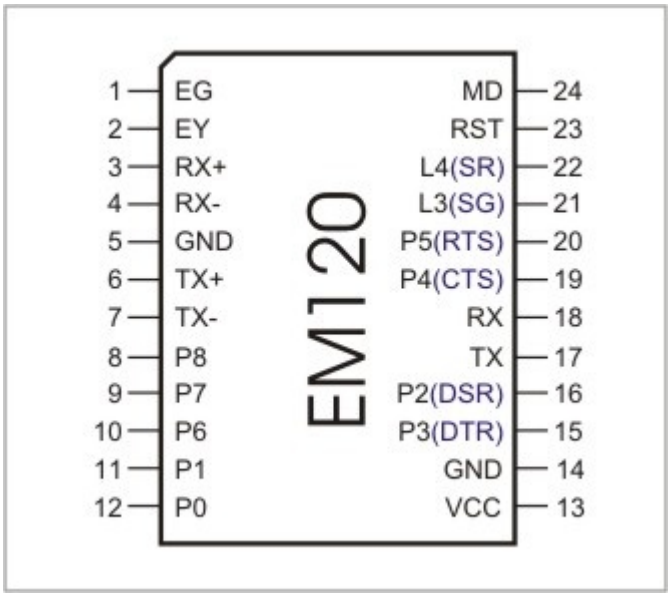
- The "serial device server" firmware, currently in its 3rd generation ("Release3"), turns the EM120 into a ready-to-work Serial Device Server that can connect almost any kind of serial device to the Ethernet (TCP/IP) network. This firmware has fixed functionality; you adjust the way the EM120 behaves by specifying the values of programmable parameters (settings) defined in this firmware. Functional description of the EM120 under the "serial device server" firmware can be found in the Device Server Application Firmware Manual. Also see Software Manuals for the information on PC software that works with devices running serial device server firmware.

- TiOS (Tibbo Operating System) firmware turns the EM120 into a BASIC-programmable controller. This controller can be used to create any kind of network and/or serial port-related control application. When running TiOS, the EM120 has no pre-defined functionality -- it simply executes your BASIC application. For more information see the following manuals: "Programmable Hardware Manual" and "TAIKO Manual".

The application firmware of the EM120 can be upgraded through the module's serial port or Ethernet port. Serial upgrades are facilitated by a so-called Monitor- a fixed "service" firmware inside the EM120. The Monitor cannot be upgraded. Network upgrades rely on the application firmware itself- there is a self upgrade algorithm that will be detailed later.

By default, the EM120 is supplied with "serial device server" firmware pre-loaded. If you wish to make the module run your BASIC application you need to upload TiOS firmware onto the Module. Visit Tibbo website to get the latest TiOS firmware.

I/O Pin Assignment and Pin Functions



Click on the pin in the diagram above or one of the links provided below to learn more about EM120's I/O pins:

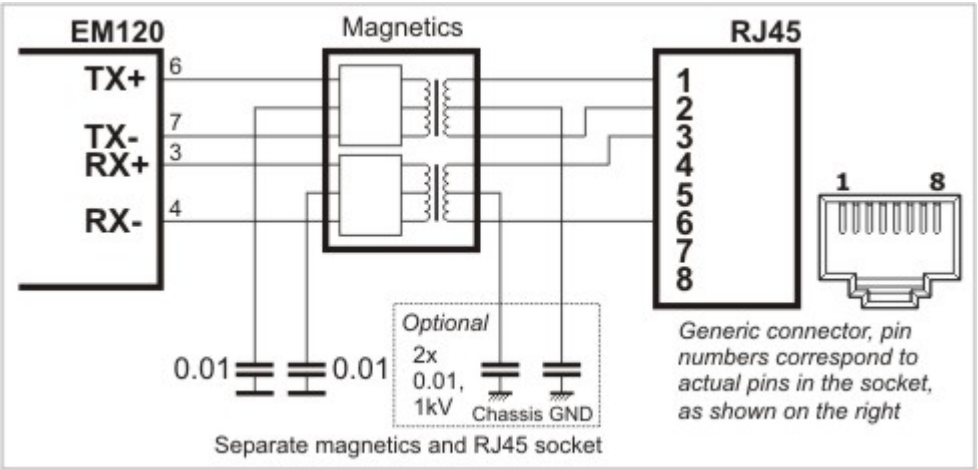
- Ethernet port lines
- Serial port and general-purpose I/O lines
- LED lines
- Power, reset, and mode selection lines

Ethernet Port Lines

#6	**TX+**	Output	Positive line of the differential output signal pair
#7	**TX-**	Output	Negative line of the differential output signal pair
#3	**RX+**	Input	Positive line of the differential input signal pair
#4	**RX-**	Input	Negative line of the differential input signal pair

Ethernet port of the EM120 is of 10BaseT type. Onboard electronics of the EM120 do not include Ethernet magnetics, so magnetic circuitry must be connected

externally. You can use either a standalone magnetics part (such as YCL-20F001N, schematic diagram shown below) or RJ45 connector with integrated magnetics.



It is important to make the PCB wire connections between the Ethernet port pins of the EM120 and external magnetics circuitry as short as possible. Making the wires too long may cause the noise level generated by your PCB surpass the maximum radiated emission limits stipulated by FCC and CE regulations. Additionally, longer Ethernet lines on the PCB will make your board more susceptible to the damage from the ESD (electrostatic discharge).

Serial Port and General-Purpose I/O Lines

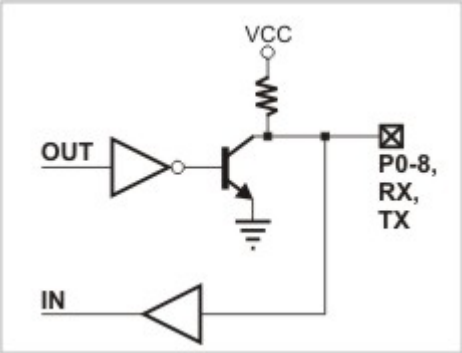
#8	**P8**	Input/output	General-purpose input/output line
#9	**P7**	Input/output	General-purpose input/output line
#10	**P6**	Input/output	General-purpose input/output line
#11	**P1**	Input/output	General-purpose input/output line
#12	**P0**	Input/output	General-purpose input/output line
#15	**P3 (DTR)**	Input/output (output)	General-purpose input/output line Data terminal ready output
#16	**P2 (DSR)**	Input/output (input)	General-purpose input/output line Data set ready input
#17	**TX**		Serial transmit line
#18	**RX**		Serial receive line
#19	**P4 (CTS/SEL)**	Input/output (input)	General-purpose input/output line Clear to send input Full-/half-duplex selection input
#20	**P5 (RTS/DIR)**	Input/output (output)	General-purpose input/output line Request to send output (full-duplex mode) Data direction control output (half-duplex mode)

*Line functions defined by the application firmware are shown in **blue***

The EM120 features a serial port (RX, TX lines) and several general-purpose I/O lines (P0-P8). All of the above lines are of CMOS type. From the hardware point of view, all general-purpose I/O lines can serve as inputs or outputs. Maximum load current for each I/O line is 10mA.

Simplified structure of EM120's I/O lines is shown on the circuit diagram below. All lines are "quasi-bidirectional" and can be viewed as open collector outputs with weak pull-up resistor. There is no explicit direction control. To "measure" an external signal applied to a pin the OUT line must first be set to HIGH. It is OK to

drive the pin LOW externally when the pin outputs HIGH internally.



The application firmware of the EM120 maps certain serial port functions onto the general-purpose I/O pins- these functions are shown in blue in the table at the top of this topic. For example, P5 is a universal input/output but the application firmware can be set to turn this line into the RTS output of the serial port. Therefore, depending on your application you can view P5 as a general-purpose I/O line or specific control line of the serial port (RTS).

Being of CMOS type, the serial port and I/O lines of the EM120 can be connected directly to the serial port pins and I/O lines of most microcontrollers, microprocessors, etc. An interface IC* must be added to the EM120 externally if you want to connect the module to a "true" serial port (for example, COM port of the PC).

Logical signals on the serial port lines of the EM120 are active LOW. TX and RX lines are high when idle, start bit is LOW, stop bit is HIGH; LOW on CTS and RTS lines means "transmission allowed" and HIGH means "transmission not allowed". This is standard for CMOS-level serial ports and is exactly opposite to the signalling on the RS232 cables. Logical signals on the EM120 are inverted because standard interface ICs* invert the signals internally too.

As explained earlier, actual functionality of the I/O lines is firmware-dependent. See serial port and serial communications for details.

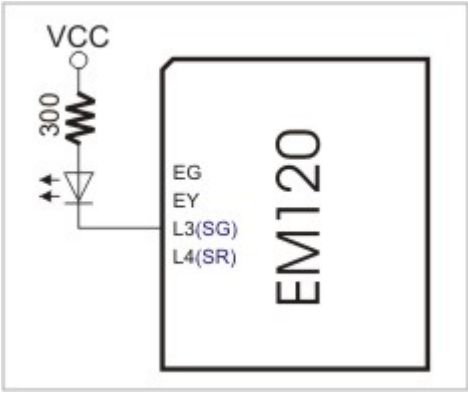
** Such as MAX232 for RS232, MAX485 for RS485, etc.*

LED Lines

#1	**EG**	Output	Green Ethernet status LED
#2	**EY**	Output	Yellow Ethernet status LED
#21	**L3 (SG)**	Output	LED output 3, Green status LED
#22	**L4 (SR)**	Output	LED output 4, Red status LED

*Line functions defined by the application firmware are shown in **blue***

The EM120 has four LED control lines. All lines have the same internal structure and the LEDs should be connected to these lines as shown on the schematic diagram below. Maximum load for each line is 10mA.



EG and EY lines reflect the status of the Ethernet port. The EG LED is normally ON, and is temporarily turned off whenever the EM120 receives a network packet. The EY is normally OFF, and is temporarily turned on whenever a data collision is detected on the Ethernet.

SG and SR lines are under firmware control and display various status information depending on what firmware is running at the moment. Follow the links below to learn more about the behaviour of these LEDs under different conditions:

- SR/SG behavior in the monitor firmware.
- SR/SG behavior in the application firmware.

Power, Reset, and Mode Selection Lines

#13	**VCC**	Input	Positive power input, 5V nominal, +/- 5%, app. 50mA
#5	**GND**		Ground
#14	**GND**		Ground
#23	**RST**	Input	Reset, active high
#24	**MD (MD)**	Input	Mode selection pin

*Line functions defined by the application firmware are shown in **blue***

The EM120 should be powered from a stabilized DS power supply with output nominal voltage of 5V (+/- 5% tolerance). Current consumption of the EM120 is approximately 50mA.

Proper external reset is a must! Reset pulse should be an active HIGH. We strongly advise against using low-cost RC-networks and other unreliable methods of generating reset pulse. Reset should be applied for as long as the power supply voltage is below 4.6V. We recommend using a dedicated reset IC with brownout detection, such as MAX810. Reset pulse length should be no less than 50ms, counting from the moment the power supply voltage exceeds 4.6V.

If the EM120 is used to serve as a communications co-processor in a larger system that has its own CPU it is also OK to control the RST line of the EM120 through a general-purpose I/O pin of the "host" microcontroller. I/O pins of many microcontrollers default to HIGH after the powerup and this means that the reset will be applied to the EM120 at the same time when the host microcontroller is reset. All the host microcontroller has to do is release the EM120 from reset at an appropriate time by switching the state of the I/O line to LOW.

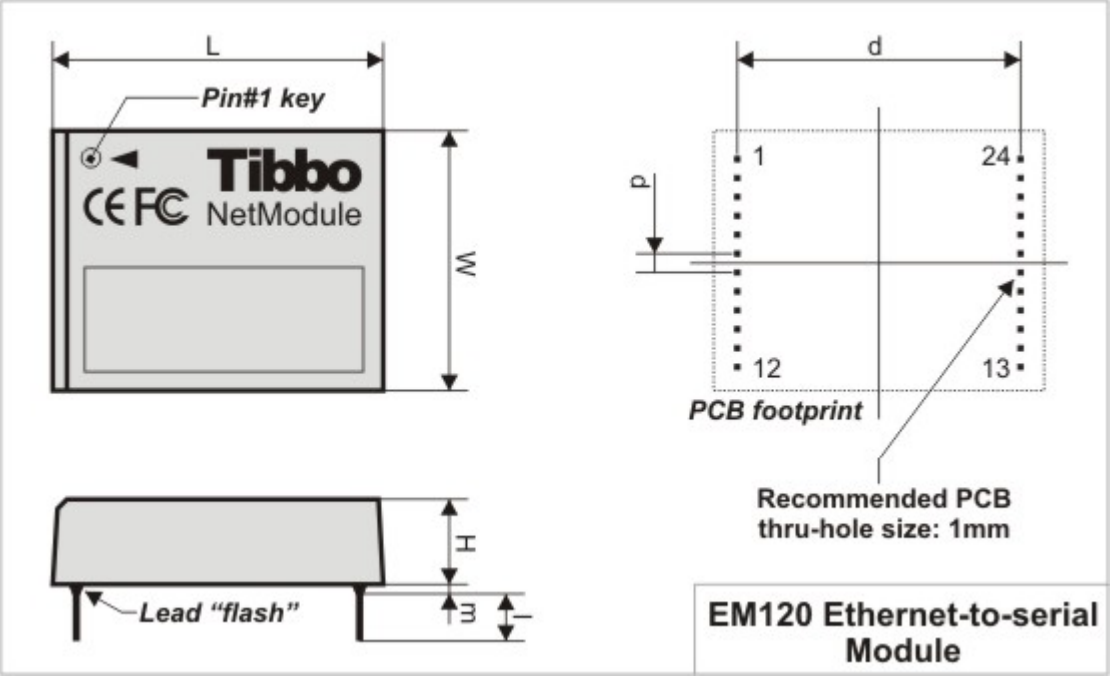
The MD line of the EM120 is used to select the operating mode of the EM120 and/or its application firmware. The reason why the pin name is shown as MD(MD) is because the functionality of this pin is in part hardwired and in part depends on

the application firmware:

- **Hardwired functionality.** When the EM120 powers up it verifies the state of the MD input. If the MD input is at HIGH the EM120 proceeds to verifying and running the application firmware loaded into its internal FLASH memory. If the MD input is at LOW the EM120 enters the serial upgrade mode. For more information see Monitor.
- **Application firmware-dependent functionality.** When the application firmware is already running the MD line is typically used to make the EM120 enter the serial programming mode. For more information see serial programming.

When the EM120 is used as a co-processor in a host system the MD line can be also controlled by the host microcontroller. Ability to control both the RST and DS lines allows the host microcontroller to switch between the operating modes of the EM120.

Mechanical Dimensions



L	Max.	35.0	Module length
W	Max.	27.5	Module width
H	Max.	9.1	Module height
I	Min.	5.0	Lead length
m	Max.	0.5	Lead "flash"
d	Aver.	30.0	Distance between lead rows
p	Aver.	2.0	Pin pitch

All dimensions are in millimeters

Specifications and EM120 Modifications

The EM120 has two sub-models in circulation- the EM120-00 and EM120-01. The EM120-01 is a RoHS-compliant version of the EM120-00. There are no other differences between these two versions. Currently, only the EM120-01 is being

manufactured.

Device specifications are presented in the table below.

Parameter	EM120-01
Ethernet interface	10BaseT Ethernet, magnetics **not** built-in
Serial interface and I/O lines	CMOS-level; TX, RX, and 9 additional I/O lines with RTS, CTS, DTR, DSR implemented in application firmware
Routing buffers size	12Kbytes x 2*
Maximum load current of I/O lines	10mA
Power requirements	DC 5V, +/- 5%, app. 50mA
Operating temperature	-10 to +70 degrees C
Operating relative humidity	10-90%
Mechanical dimensions (excl. leads)	App. 35x27.5x9.1mm
Packing	Tube, 10 modules/tube

** Maximum possible buffer size. Actual size may be smaller depending on how much RAM is "consumed" by the firmware*

Note: all specifications are for reference only. TIBBO assumes no responsibility for any errors which may appear in this Manual, and does not make any commitment to update the information contained herein.

EM200 Ethernet Module

The EM200 is an Ethernet Module for onboard installation. Module hardware includes one 100BaseT Ethernet port (standard Ethernet magnetics are **NOT integrated** into the Module), one serial port (CMOS-level) with a number of additional general-purpose I/O lines, and an internal processor, whose firmware acts as a bridge between the Ethernet and serial ports. Ethernet "side" of the Module connects directly to a standard Ethernet magnetics circuit (such as YCL-PH163112) or RJ45 connector with integrated magnetics. Serial "side" interfaces directly to the serial port pins of most microcontrollers, microprocessors, UARTs, etc.

From the hardware standpoint, the EM200 can be viewed as a universal platform suitable for running a variety of network and serial communications-related applications. It is the application firmware, not hardware that gives the EM200 most of its functionality.

The EM200 can run two distinctively different kinds of application firmware:

- The "serial device server" firmware, currently in its 3rd generation ("Release3"), turns the EM200 into a ready-to-work Serial Device Server that can connect almost any kind of serial device to the Ethernet (TCP/IP) network. This firmware has fixed functionality; you adjust the way the EM200 behaves by specifying the values of programmable parameters (settings) defined in this firmware. Functional description of the EM200 under the "serial device server" firmware can be found in the Device Server Application Firmware Manual. Also see Software

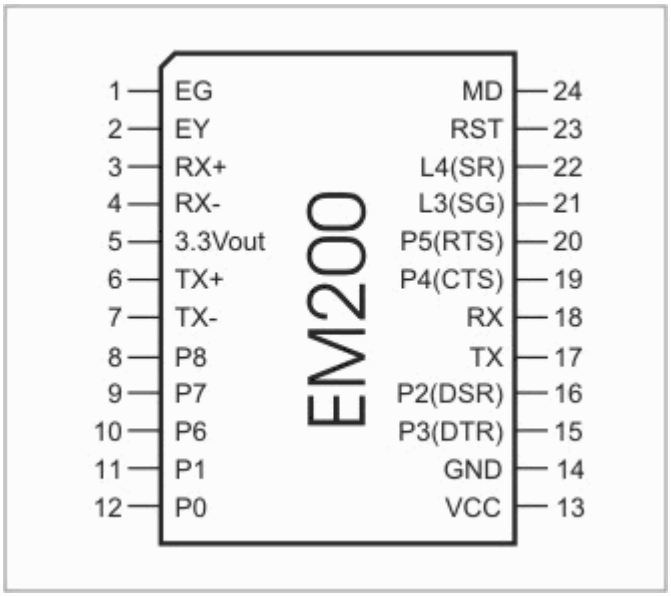
Manuals for the information on PC software that works with devices running serial device server firmware.

- TiOS (Tibbo Operating System) firmware turns the EM200 into a BASIC-programmable controller. This controller can be used to create any kind of network and/or serial port-related application. When running TiOS, the EM200 has no pre-defined functionality -- it simply executes your BASIC application. For more information see the following manuals: "Programmable Hardware Manual" and "TAIKO Manual".

The application firmware of the EM200 can be upgraded through the module's serial port or Ethernet port. Serial upgrades are facilitated by a so-called Monitor- a fixed "service" firmware inside the EM200. The Monitor cannot be upgraded. Network upgrades rely on the application firmware itself- there is a self upgrade algorithm that will be detailed later.

By default, the EM200 is supplied with "serial device server" firmware pre-loaded. If you wish to make the module run your BASIC application you need to upload TiOS firmware onto the Module. Visit Tibbo website to get the latest TiOS firmware.

I/O Pin Assignment and Pin Functions



Click on the pin in the diagram above or one of the links provided below to learn more about EM200's I/O pins:

- Ethernet port lines
- Serial port and general-purpose I/O lines
- LED lines
- Power, reset, and mode selection lines

Ethernet Port Lines

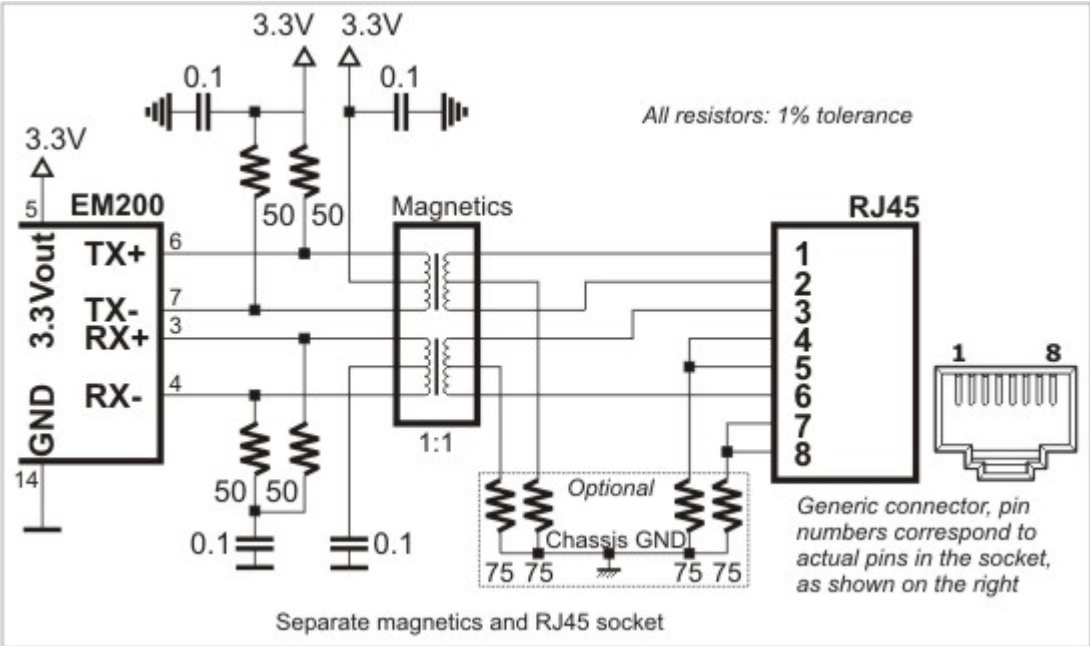
#6	**TX+**	Output	Positive line of the differential output signal pair
#7	**TX-**	Output	Negative line of the differential output signal pair

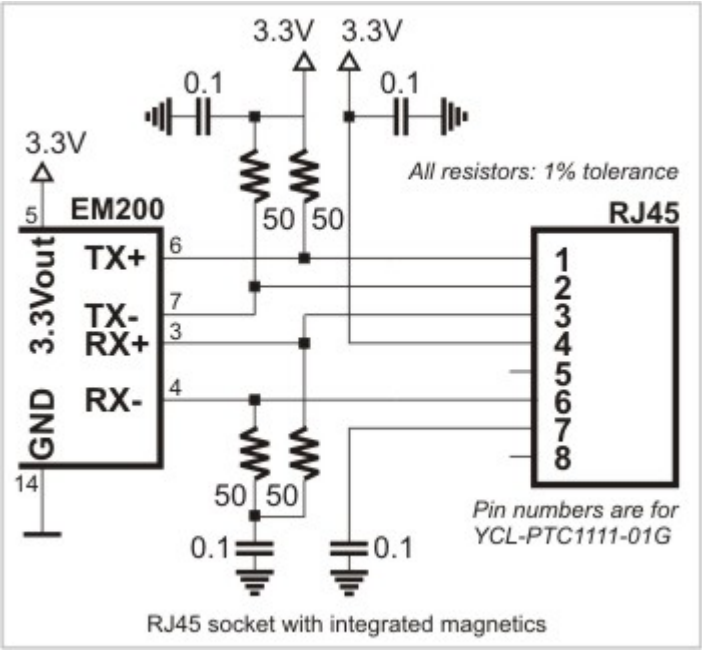
#3	**RX+**	Input	Positive line of the differential input signal pair
#4	**RX-**	Input	Negative line of the differential input signal pair
#5	**3.3Vout**	Output	"Clean" 3.3V power for magnetics circuitry

Ethernet port of the EM200 is of 100BaseT type. Onboard electronics of the EM200 do not include Ethernet magnetics, so magnetic circuitry must be connected externally. You can use either a standalone magnetics part (such as YCL-PH163112) or RJ45 connector with integrated magnetics (for example, YCL-PTC1111-01G). Diagrams below show circuit diagrams for both parts.

Please, note the following:

- The 3.3Vout is an *output* that provides clean power for the magnetics circuitry, which is very sensitive to noise.
- Do not combine 3.3Vout with the VCC (main power) pin. This is counter-productive and will cause FCC/CE certification issues.





It is important to make the PCB wire connections between the Ethernet port pins of the EM200 and external magnetics circuitry as short as possible. Making the wires too long may cause the noise level generated by your PCB surpass the maximum radiated emission limits stipulated by FCC and CE regulations. Additionally, longer Ethernet lines on the PCB will make your board more susceptible to the damage from the ESD (electrostatic discharge).

Serial Port and General-Purpose I/O Lines

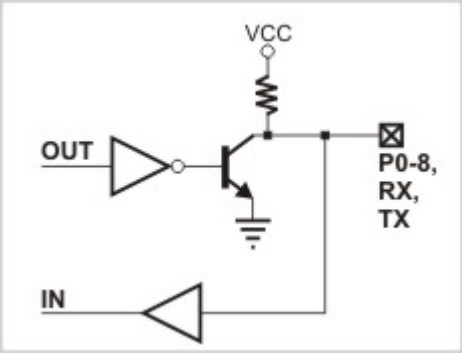
#8	**P8**	Input/output	General-purpose input/output line
#9	**P7**	Input/output	General-purpose input/output line
#10	**P6**	Input/output	General-purpose input/output line
#11	**P1**	Input/output	General-purpose input/output line
#12	**P0**	Input/output	General-purpose input/output line
#15	**P3 (DTR)**	Input/output (output)	General-purpose input/output line Data terminal ready output
#16	**P2 (DSR)**	Input/output (input)	General-purpose input/output line Data set ready input
#17	**TX**		Serial transmit line
#18	**RX**		Serial receive line
#19	**P4 (CTS/SEL)**	Input/output (input)	General-purpose input/output line Clear to send input Full-/half-duplex selection input
#20	**P5 (RTS/DIR)**	Input/output (output)	General-purpose input/output line Request to send output (full-duplex mode) Data direction control output (half-duplex mode)

*Line functions defined by the application firmware are shown in **blue***

The EM200 features a serial port (RX, TX lines) and several general-purpose I/O lines (P0-P8). All of the above lines are of CMOS type. From the hardware point of view, all general-purpose I/O lines can serve as inputs or outputs. Maximum load current for each I/O line is 10mA.

Simplified structure of EM200's I/O lines is shown on the circuit diagram below. All

lines are "quasi-bidirectional" and can be viewed as open collector outputs with weak pull-up resistor. There is no explicit direction control. To "measure" an external signal applied to a pin the OUT line must first be set to HIGH. It is OK to drive the pin LOW externally when the pin outputs HIGH internally.



Device server application firmware of the EM1000 maps certain serial port functions onto the general-purpose I/O pins- these functions are shown in blue in the table at the top of this topic. For example, P5 is a universal input/output but the application firmware can be set to turn this line into the RTS output of the serial port. Therefore, depending on your application you can view P5 as a general-purpose I/O line or specific control line of the serial port (RTS).

Being of CMOS type, the serial port and I/O lines of the EM200 can be connected directly to the serial port pins and I/O lines of most microcontrollers, microprocessors, etc. An interface IC* must be added to the EM200 externally if you want to connect the module to a "true" serial port (for example, COM port of the PC).

Logical signals on the serial port lines of the EM200 are active LOW. TX and RX lines are high when idle, start bit is LOW, stop bit is HIGH; LOW on CTS and RTS lines means "transmission allowed" and HIGH means "transmission not allowed". This is standard for CMOS-level serial ports and is exactly opposite to the signalling on the RS232 cables. Logical signals on the EM200 are inverted because standard interface ICs* invert the signals internally too.

As explained earlier, actual functionality of the I/O lines is firmware-dependent. See serial port and serial communications for details.

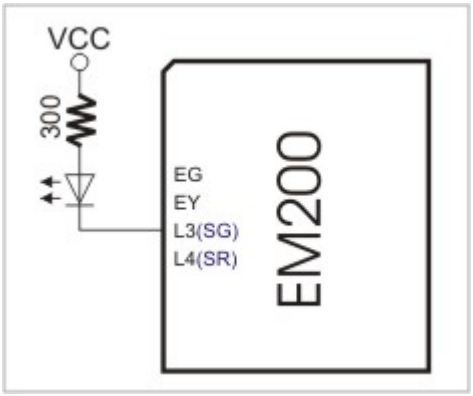
** Such as MAX232 for RS232, MAX485 for RS485, etc.*

LED Lines

#1	**EG**	Output	Green Ethernet status LED
#2	**EY**	Output	Yellow Ethernet status LED
#21	**L3 (SG)**	Output	LED output 3, Green status LED
#22	**L4 (SR)**	Output	LED output 4, Red status LED

*Line functions defined by the application firmware are shown in **blue***

The EM200 has four LED control lines. All lines have the same internal structure and the LEDs should be connected to these lines as shown on the schematic diagram below. Maximum load for each line is 10mA.



EG and EY lines reflect the status of the Ethernet port. The EG LED is switched on whenever a live Ethernet connection is detected by the EM200's Ethernet port. The EG LED is momentarily switched off whenever the EM200 receives a network packet. The EY shows whether current Ethernet link is of 10BaseT type (EY off) or 100BaseT (EY on).

SG and SR lines are under firmware control and display various status information depending on what firmware is running at the moment. Follow the links below to learn more about the behaviour of these LEDs under different conditions:

- SR/SG behavior in the monitor firmware.
- SR/SG behavior in the application firmware.

Power, Reset, and Mode Selection Lines

#13	**VCC**	Input	Positive power input, 5V nominal, +/- 5%, app. **220mA**
#14	**GND**		Ground
#23	**RST**	Input	Reset, active high
#24	**MD (MD)**	Input	Mode selection pin

*Line functions defined by the application firmware are shown in **blue***

The EM200 should be powered from a stabilized DS power supply with output nominal voltage of 5V (+/- 5% tolerance). Current consumption of the EM200 is approximately 220mA (in 100BaseT mode).

Proper external reset is a must! Reset pulse should be an active HIGH. We strongly advise against using low-cost RC-networks and other unreliable methods of generating reset pulse. Reset should be applied for as long as the power supply voltage is below 4.6V. We recommend using a dedicated reset IC with brownout detection, such as MAX810. Reset pulse length should be no less than 50ms, counting from the moment the power supply voltage exceeds 4.6V.

If the EM200 is used to serve as a communications co-processor in a larger system that has its own CPU it is also OK to control the RST line of the EM200 through a general-purpose I/O pin of the "host" microcontroller. I/O pins of many microcontrollers default to HIGH after the powerup and this means that the reset will be applied to the EM200 at the same time when the host microcontroller is reset. All the host microcontroller has to do is release the EM200 from reset at an appropriate time by switching the state of the I/O line to LOW.

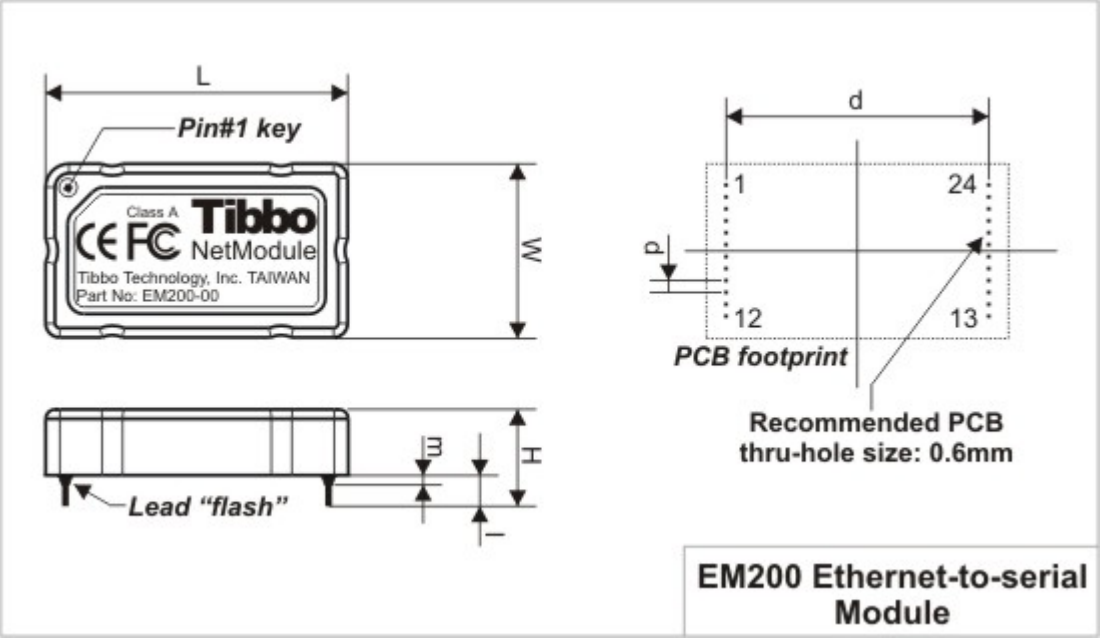
The MD line of the EM200 is used to select the operating mode of the EM200 and/or its application firmware. The reason why the pin name is shown as MD(MD) is because the functionality of this pin is in part hardwired and in part depends on

the application firmware:

- **Hardwired functionality.** When the EM200 powers up it verifies the state of the MD input. If the MD input is at HIGH the EM200 proceeds to verifying and running the application firmware loaded into its internal FLASH memory. If the MD input is at LOW the EM200 enters the serial upgrade mode. For more information see Monitor.
- **Application firmware-dependent functionality.** When the application firmware is already running the MD line is typically used to make the EM200 enter the serial programming mode. For more information see serial programming.

When the EM200 is used as a co-processor in a host system the MD line can be also controlled by the host microcontroller. Ability to control both the RST and DS lines allows the host microcontroller to switch between the operating modes of the EM200.

Mechanical Dimensions



L	Max.	32.1	Module length
W	Max.	18.5	Module width
H	Max.	7.3	Module height
I	Min.	2.2	Lead length
m	Max.	0.5	Lead "flash"
d	Aver.	28.0	Distance between lead rows
p	Aver.	1.27	Pin pitch

All dimensions are in millimeters

Specifications and EM200 Modifications

The EM200 has two sub-models in circulation- the EM200-00 and EM200-01. The EM200-01 is a RoHS-compliant version of the EM200-00. There are no other differences between these two versions. Currently, only the EM200-01 is being manufactured.

Device specifications are presented in the table below.

Parameter	EM200-01
Ethernet interface	10/100BaseT Ethernet, magnetics **not** built-in
Serial interface and I/O lines	CMOS-level; TX, RX, and 9 additional I/O lines with RTS, CTS, DTR, DSR implemented in application firmware
Routing buffers size	12Kbytes x 2*
Maximum load current of I/O lines	10mA
Power requirements	DC 5V, +/- 5%, app. 220mA
Device temperature during operation	+55 degrees C** (in 100BaseT mode)
Operating temperature	-10 to +70 degrees C
Operating relative humidity	10-90%
Mechanical dimensions (excl. leads)	App. 32.1x18.5x7.3mm
Packing	Tube, 10 modules/tube

** Maximum possible buffer size. Actual size may be smaller depending on how much RAM is "consumed" by the firmware*

*** As measured on top of the device*

Note: all specifications are for reference only. TIBBO assumes no responsibility for any errors which may appear in this Manual, and does not make any commitment to update the information contained herein.

EM203A Ethernet-to-Serial Module

The EM203A is an Ethernet module for onboard installation. Module hardware includes one 10/100BaseT Ethernet port (standard Ethernet magnetics are NOT integrated into the module), one CMOS-level serial port, and an internal processor, whose firmware acts as a bridge between the Ethernet and the serial port. The Ethernet "side" of the module connects to the RJ203A jack/magnetics module, a standard Ethernet magnetics circuit (such as YCL-PH163112) or RJ45 connector with integrated magnetics. Serial "side" interfaces directly to the serial port pins of most microcontrollers, microprocessors, UARTs, etc. The module additionally features four status LEDs onboard.

From the hardware standpoint, the EM203A can be viewed as a universal platform suitable for running a variety of applications. It is the application firmware, not the hardware that gives the EM203A most of its functionality. The EM203A is offered with two distinctively different kinds of application firmware:

- "Serial-to-Ethernet" firmware, currently in its 3rd generation ("Release3"), turns the EM203A into a ready-to-work serial-to-Ethernet converter that can connect almost any kind of serial device to the Ethernet (TCP/IP) network. This firmware has fixed functionality; you adjust the way the EM203A behaves by

specifying the values of programmable parameters (settings) defined in this firmware.

- TiOS (Tibbo Operating System) firmware turns the EM203A into a BASIC-programmable controller. When running TiOS, the EM203A has no pre-defined functionality -- it is your BASIC application that defines what the EM203A will do. TiOS and BASIC programming are covered in a separate Manual ("TAIKO Manual").

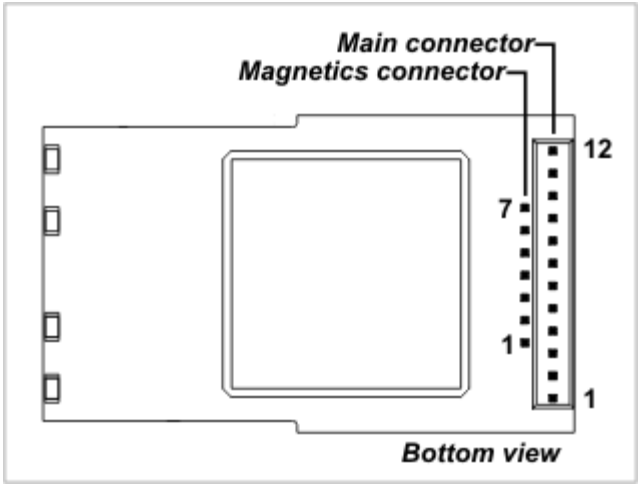
The application firmware of the EM203A can be upgraded through the module's serial port or Ethernet port. Serial upgrades are facilitated by a so-called Monitor- a fixed "service" firmware inside the EM203A. Network upgrades rely on the application firmware, that is, the firmware can "upgrade itself".

The EM203A is supplied with "serial device server" firmware pre-loaded. If you wish to receive the module with TiOS firmware, please specify option "P" device on your order. Alternatively, you can just load the TiOS firmware by yourself. Current firmware versions are posted on our website.

The EM203A module features a heat-conductive sticker. Protective paper of the sticker **MUST BE REMOVED** prior to installing the module onto the host PCB. More on this in the Thermal Considerations topic.

I/O Pin Assignment and Pin Functions

The EM203A has two connectors -- main connector and magnetics connector. Depending on the EM203A version, magnetics connector can be soldered facing up or down, as described in the Mechanical Dimensions topic.



Main connector

#1	**MD***	Input	Mode selection pin
#2	**RST**	Input	Reset, active high
#3	**P3** **DTR***	Input/output Output	General-purpose input/output line Data terminal ready output

#4	**P2** **DSR***	Input/output Input	General-purpose input/output line Data set ready input
#5**	**L3** **SG***	Output Output	LED output 3 Green status LED control line
#6**	**L4** **SR***	Output Output	LED output 4 Red status LED control line
#7**	**VCC**		Positive power input, 5V nominal, +/- 5%, app. 220mA
#8**	**GND**		Ground
#9	**RX**	Input	Serial receive line
#10	**TX**	Output	Serial transmit line
#11	**P4** **CTS/SEL***	Input/output Input	General-purpose input/output line Clear to send input; full-/half-duplex selection input
#12	**P5** **RTS/DIR***	Input/output Output	General-purpose input/output line Request to send output (full-duplex mode); data direction control output (half-duplex mode)

** Implemented in (supported through) firmware.*

*** For the EM203A device (without "D" option), these pins also extend upward so that they can potentially mate with the RJ203A module.*

Magnetics connector

#1	**RX+**	Input	Ethernet port, positive line of the differential input signal pair
#2	**RX-**	Input	Ethernet port, negative line of the differential input signal pair
#3	**AVCC**	Output	"Clean" 3.3V power output for magnetics circuitry
#4	**TX+**	Output	Ethernet port, positive line of the differential output signal pair
#5	**TX-**	Output	Ethernet port, negative line of the differential output signal pair
#6	**L1(EG)**	Output	Green Ethernet status LED control line.
#7	**L2(EY)**	Output	Yellow Ethernet status LED control line.

Serial Port and General-Purpose I/O Lines

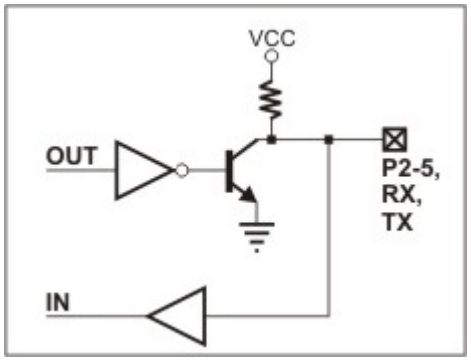
Main conn., #3	**P3** **DTR***	Input/output Output	General-purpose input/output line Data terminal ready output
Main conn., #4	**P2** **DSR***	Input/output Input	General-purpose input/output line Data set ready input
Main conn., #9	**RX**		Serial receive line
Main conn., #10	**TX**		Serial transmit line

Main conn., #11	**P4 CTS /SEL***	Input/ output Input	General-purpose input/output line Clear to send input; full-/half-duplex selection input
Main conn., #12	**P5 RTS /DIR***	Input/ output Output	General-purpose input/output line Request to send output (full-duplex mode); data direction control output (half-duplex mode)

* *Implemented in (supported through) firmware.*

The EM203A features a serial port (RX, TX lines) and several general-purpose I/O lines P2-P5 (there are no lines P0 and P1; line names were selected for naming compatibility with the EM100). All of the above lines are of CMOS type. From the hardware point of view, all general-purpose I/O lines can serve as inputs or outputs. Maximum load current for each I/O line is 10mA.

Simplified structure of EM203A's I/O lines is shown on the circuit diagram below. All lines are "quasi-bidirectional" and can be viewed as open collector outputs with weak pull-up resistor. There is no explicit direction control. To "measure" an external signal applied to a pin the OUT line must first be set to HIGH. It is OK to drive the pin LOW externally, while the pin outputs HIGH internally.



The serial-to-Ethernet firmware of the EM203A maps certain serial port functions onto the general-purpose I/O pins- these functions are shown in blue in the table above. For example, P5 is a universal input/output but the application firmware can be set to turn this line into the RTS output of the serial port. Therefore, depending on your application, you can view P5 as a general-purpose I/O line or specific control line of the serial port (RTS).

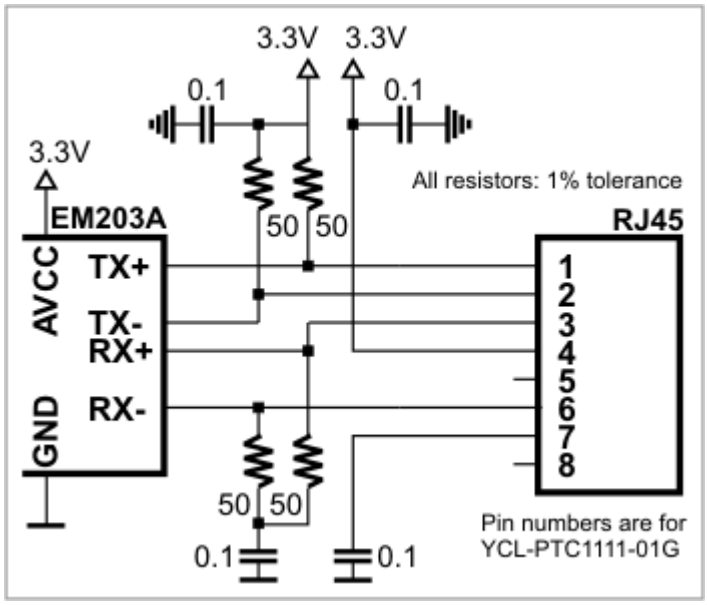
Being of CMOS type, the serial port and I/O lines of the EM203A can be connected directly to the serial port pins and I/O lines of most microcontrollers, microprocessors, etc. An interface IC, such as the MAX232, must be added to the EM203A externally if you want to connect the module to a "true" serial port (for example, COM port of the PC).

Logical signals on the serial port lines of the EM203A are active LOW. TX and RX lines are high when idle, start bit is LOW, stop bit is HIGH; LOW on CTS and RTS lines means "transmission allowed" and HIGH means "transmission not allowed". This is standard for CMOS-level serial ports and is exactly opposite to the signalling on the RS232 cables. Logical signals on the EM203A are inverted because standard RS232 ICs, such as the MAX232, invert the signals internally one more time.

As explained earlier, actual functionality of the I/O lines is firmware-dependent.

Ethernet Lines and Jack/Magnetics Data

Mag. conn., #1	**RX+**	Input	Ethernet port, positive line of the differential input signal pair
Mag. conn., #2	**RX-**	Input	Ethernet port, negative line of the differential input signal pair
Mag. conn., #9	**TX+**	Output	Ethernet port, positive line of the differential output signal pair
Mag. conn., #10	**TX-**	Output	Ethernet port, negative line of the differential output signal pair
Mag. conn., #11	**AVCC**	Output	"Clean" 3.3V power output for magnetics circuitry



Ethernet port of the EM203A is of 100/10BaseT type. Onboard electronics of the EM203A do not include Ethernet magnetics, so magnetics circuitry must be connected externally to pins TX+, TX-, RX+, RX-, and AVCC. The AVCC pin outputs clean power for the magnetics circuitry, which is very sensitive to noise. The voltage on the AVCC output is 3.3V.

Do not interconnect the AVCC and VCC pins! This will permanently damage the EM203A.

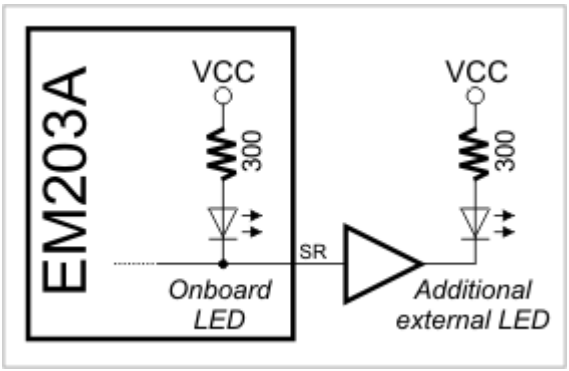
One of the options for implementing the Ethernet front-end is to use RJ203 jack/magnetics module.

LED Lines

Magnetics conn., #6	**L1(EG)**	Output	Green Ethernet status LED control line. The LED will be turned on when the EM203A links with the hub at 100Mb. The LED will be off if the link is established at 10Mb.
Magnetics conn., #7	**L2(EY)**	Output	Yellow Ethernet status LED control line. The LED will be turned on when "live" Ethernet cable is plugged into the Module. The LED will be temporarily switched off whenever an Ethernet packet is received.
Main conn., #5	**L3 SG***	Output Output	LED output 3 Green status LED
Main conn., #6	**L4 SR***	Output Output	LED output 4 Red status LED

* *Implemented in (supported through) firmware.*

The EM203A has four onboard LEDs and four control lines -- L1-L4 -- to connect external LEDs in parallel with the onboard ones. External LEDs should be connected through a TTL buffer element. This will reduce the load on the EM203A's internal circuit. Maximum load for each line without the buffer is 2mA.



The firmware of the EM203A uses L3 and L4 as "status LEDs" which display various status information depending on what firmware is running at the moment.

Power, Reset, and Mode Selection Lines

#7	**VCC**	Input	Positive power input, 5V nominal, +/- 5%, app. **230mA**
#8	**GND**		Ground
#2	**RST**	Input	Reset, active high
#1	**MD***	Input	Mode selection pin

* *Implemented in (supported through) firmware*

The EM203A should be powered from a stabilized DS power supply with output nominal voltage of 5V (+/- 5% tolerance). Current consumption of the EM203A is approximately 220mA (in 100BaseT mode).

Proper external reset is a must! Reset pulse should be active HIGH. We strongly advise against using low-cost RC-networks and other unreliable methods of generating reset. Reset should be applied for as long as the power supply voltage is below 4.6V. We recommend using a dedicated reset IC with brownout detection, such as the MAX810. Reset pulse length should be no less than 50ms, counting from the moment the power supply voltage exceeds 4.6V.

If the EM203A is used to serve as a communications co-processor in a larger system that has its own CPU it is also OK to control the RST line of the EM203A through a general-purpose I/O pin of the "host" microcontroller. I/O pins of many microcontrollers default to HIGH after the powerup and this means that the reset will be applied to the EM203A in parallel with the reset of the microcontroller. All the host microcontroller has to do is release the EM203A from reset at an appropriate time by switching the state of the I/O line to LOW.

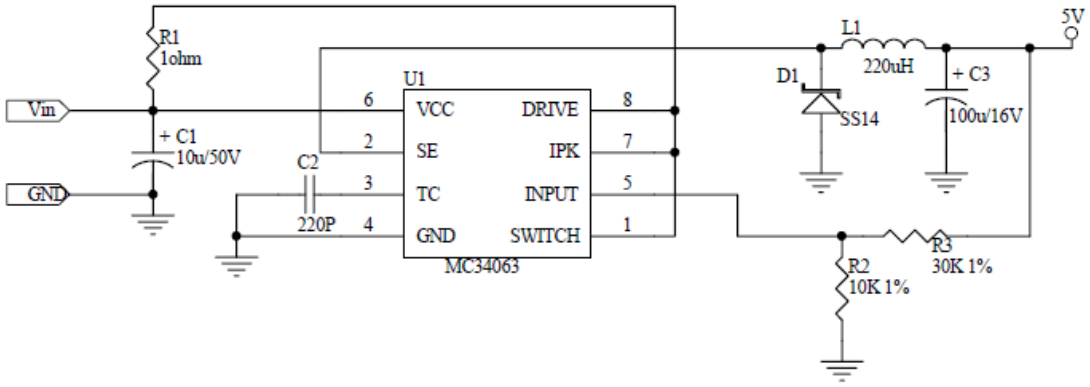
The MD line of the EM203A is used to select the operating mode of the EM203A and/or its application firmware. This pin has the following functionality:

- When the EM203A powers up it verifies the state of the MD input. If the MD input is at HIGH the EM203A proceeds to verifying and running the application firmware loaded into its internal FLASH memory. If the MD input is at LOW the EM203A enters the serial upgrade mode.
- When the serial-to-Ethernet firmware is already running the MD line is used to make the EM203A enter the serial programming mode.

When the EM203A is used as a co-processor in a host system the MD line can be also controlled by the host microcontroller. Ability to control both the RST and the MD lines will allow the host microcontroller to switch between the operating modes of the EM203A.

Power supply circuit

Many power supply circuits will work well. The one below is being used by Tibbo. The circuit can handle input voltages in the 9-24V range.



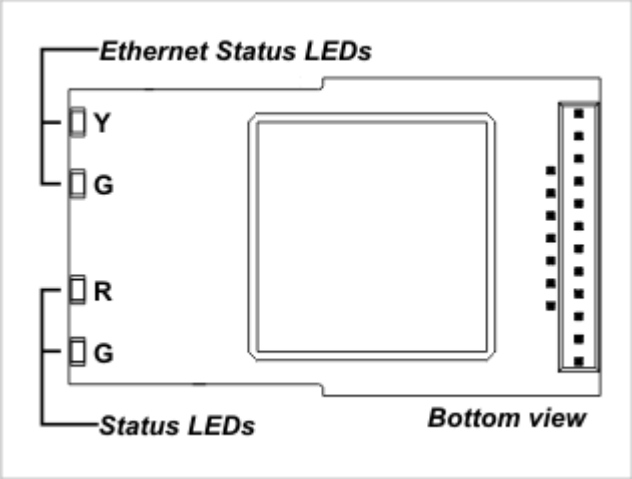
Notes:

- U1 is a popular power IC manufactured by ON SEMICONDUCTOR.
- C1 and C2 capacitors: Do not use SMD capacitors -- use regular through-hole aluminum capacitors. This really helps reduce noise produced by the power supply.
- This is an analog circuit, so layout matters. Apply reasonable "good layout" effort.

Ideally, one should use an oscilloscope to see what sort of "square wave" the power supply generates, both at low and high input voltages, as well as light and heavy loads. There are no recipes here -- just try and see what works for your circuit.

Onboard LEDs

The EM203A features four onboard status LEDs. The LEDs are strategically positioned on the edge of the module's board. Your product can have a small window or opening on its cover to make the LEDs of the EM203A visible from the outside. When the EM203A is used in combination with the RJ203A module, the status LEDs are visible through a transparent portion of the RJ203A's housing.



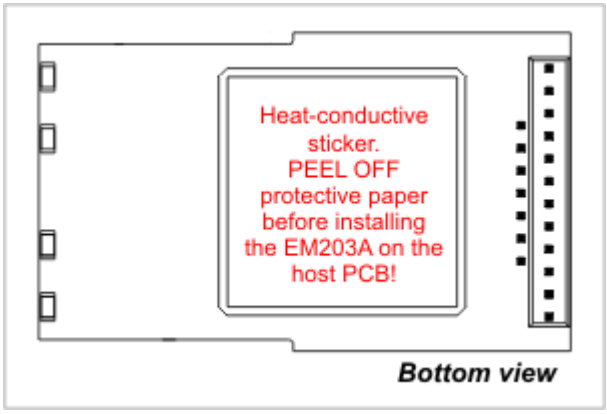
Onboard LEDs have the following function:

- Yellow Ethernet status LED is turned on when the EM203A links with the hub at 100Mb. The LED is off when the link is established at 10Mb.
- Green Ethernet status LED is turned on when "live" Ethernet cable is plugged into the Module. The LED is temporarily switched off whenever an Ethernet packet is received.
- Red and green status LEDs are under the control of EM203A's firmware.

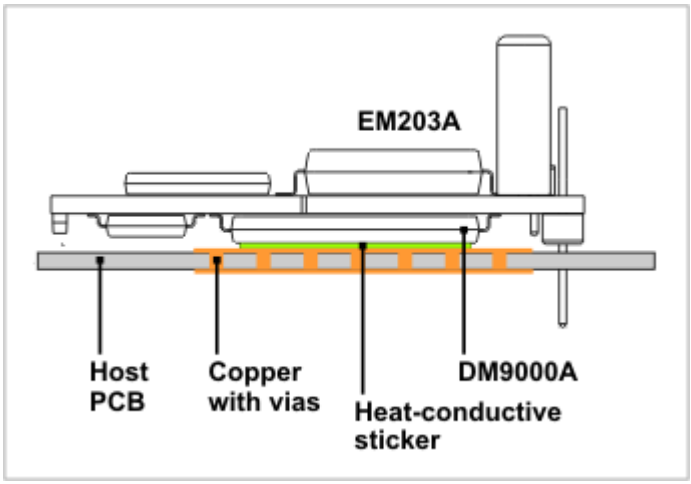
The EM203A also has four LED control lines that allow you to add external LEDs in parallel with the onboard ones.

Thermal considerations

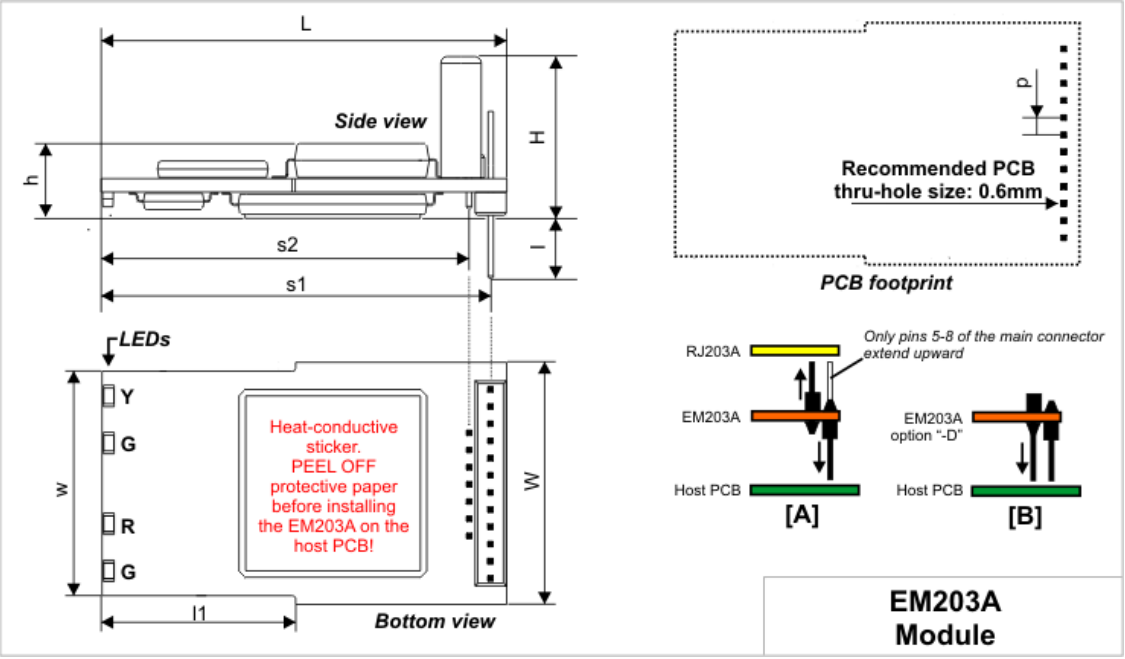
The DM9000A Ethernet controller of the EM203A can become very hot during normal module operation. To aid the module in dissipating excess heat, a special heat-conductive sticker is applied to the top of the DM9000A. Protective paper of the sticker **MUST BE REMOVED** prior to installing the module on the host PCB.



To further lower the operating temperature of the EM203A we advise you to arrange a copper area on the host PCB and in contact with the heat-conductive sticker. Best results are achieved when the copper area is larger, and also when two copper areas are provided on both sides of the host PCB and interconnected by a number of large vias.



Mechanical Dimensions

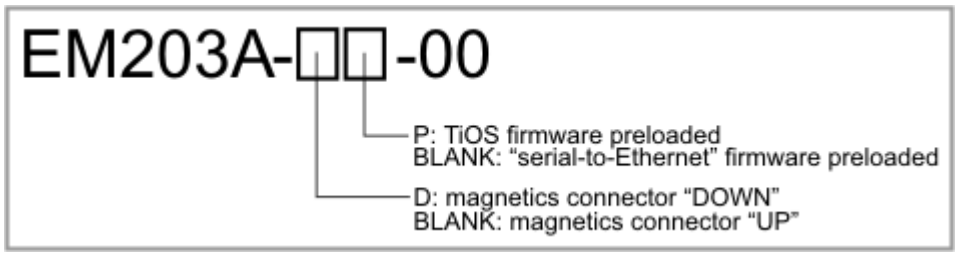


L	Max.	30.1	Length
W	Max.	18.1	Width
H	Max.	12.5	Height
l1	Aver.	14.4	Length of the narrower part of the board
w	Max.	16.7	Width at the narrower part of the board
h	Max.	5.5	Height excluding crystal oscillators
p	Aver.	1.27	Pin pitch
s1	Aver.	28.9	Distance from the edge of the board to the pins of the main connector
s2	Aver.	27.3	Distance from the edge of the board to the pins of the magnetics connector
l	Min.	4.0	Connector pin length

All dimensions are in millimeters

Specifications and Ordering Info

The EM203A has several versions available. Device numbering scheme is as follows:



Examples of valid model numbers

Model number	Description
EM203A-00	Connector configured to mate with the EM203A jack/magnetics module, the module will be supplied with the "serial-to-Ethernet" firmware
EM203A-D-00	Connectors configured to mate with the host PCB, the module will be supplied with the "serial-to-Ethernet" firmware
EM203A-DP-00	Connectors configured to mate with the host PCB, the module will be supplied with the TiOS firmware

Ordering the EM203A and RJ203A module combination

The EM203A can also be ordered in combination with the RJ203A module. To receive these two modules pre-assembled together, please specify "RJ203A+EM203A" on your order if you wish to receive the EM203A with the "serial-to-Ethernet" firmware preloaded, or "RJ203A+EM203A-P" if you wish to receive the EM203A with the TiOS firmware preloaded.

Specifications

Ethernet interface	10/100BaseT Ethernet, magnetics **not** built-in
Serial ports	1 port, CMOS-level
UART capabilities	Baudrates up to 115'200bps; none/even/odd/mark/space parity and 7/8 bits/character; full-duplex UART mode with optional flow control* and half-duplex UART mode with automatic direction control*; RX, TX, RTS*, CTS*, DTR*, and DSR* lines.
LEDs	x4 : red and green status LEDs*, yellow and green Ethernet status LEDs
Max. load current for each I/O line	10mA
Supported network protocols*	UDP*, TCP*, ICMP (ping)*, and DHCP*.
Nominal power supply voltage (VCC pin)	DC 5.0V, +/- 5%
Device reset	Proper reset must be provided externally, RST pin has active HIGH polarity
Max. operating current	220mA
Operating temperature	-20 to +70 degrees C
Operating relative humidity	10-90%
Mechanical dimensions (excl. leads)	30.1x18.1x12.5mm

Packaging	EM203A devices: tray, 30 modules/tray EM203A+RJ203A module combination: tube, 10 modules/tube

** Implemented in (supported through) firmware.*

Note: all specifications are for reference only. TIBBO assumes no responsibility for any errors which may appear in this Manual, and does not make any commitment to update the information contained herein.

EM203 Ethernet-to-Serial Module

The EM203 is an Ethernet module for onboard installation. Module hardware includes one auto-MDIX* 10/100BaseT Ethernet port (standard Ethernet magnetics are NOT integrated into the module), one CMOS-level serial port, and an internal processor, whose firmware acts as a bridge between the Ethernet and the serial port. The Ethernet "side" of the module connects to the RJ203 jack/magnetics module, a standard Ethernet magnetics circuit (such as YCL-PH163112) or RJ45 connector with integrated magnetics. Serial "side" interfaces directly to the serial port pins of most microcontrollers, microprocessors, UARTs, etc. The module additionally features four status LEDs onboard.

The EM203 device is an improved version of the EM203A module. The EM203 uses newer DM9000B Ethernet controller, which features auto-MDIX as well as fully digital PHY. Additionally, the EM203 has lower profile due to the removal of two large crystal oscillators which are present on the EM203A and are mounted vertically. The oscillators on the EM203 are tiny SMT components.

From the hardware standpoint, the EM203 can be viewed as a universal platform suitable for running a variety of applications. It is the application firmware, not the hardware that gives the EM203 most of its functionality. The EM203 is offered with two distinctively different kinds of application firmware (**version restrictions apply -- see below**):

- "Serial-to-Ethernet" firmware, currently in its 3rd generation ("Release3"), turns the EM203 into a ready-to-work serial-to-Ethernet converter that can connect almost any kind of serial device to the Ethernet (TCP/IP) network. This firmware has fixed functionality; you adjust the way the EM203 behaves by specifying the values of programmable parameters (settings) defined in this firmware.
- TiOS (Tibbo Operating System) firmware turns the EM203 into a BASIC-programmable controller. When running TiOS, the EM203 has no pre-defined functionality -- it is your BASIC application that defines what the EM203 will do. TiOS and BASIC programming are covered in a separate Manual ("TAIKO Manual").

Firmware version restrictions

The use of the newer DM9000B Ethernet controller required firmware changes which were made in **V3.70** of the "serial-to-Ethernet" firmware and **2.05.10** of the "TiOS" firmware. Earlier firmware versions will not run on the EM203! If you have to stick to the older firmware, please use the EM203A device instead.

The application firmware of the EM203 can be upgraded through the module's serial port or Ethernet port. Serial upgrades are facilitated by a so-called Monitor- a fixed "service" firmware inside the EM203. Network upgrades rely on the application firmware, that is, the firmware can "upgrade itself".

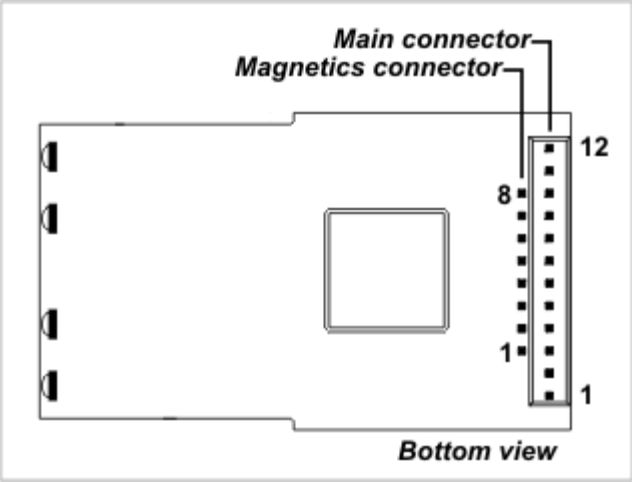
The EM203 is supplied with "serial device server" firmware pre-loaded. If you wish to receive the module with TiOS firmware, please specify option "P" device on your order. Alternatively, you can just load the TiOS firmware by yourself. Current firmware versions are posted on our website.

The EM203 module features a heat-conductive sticker. Protective paper of the sticker **MUST BE REMOVED** prior to installing the module onto the host PCB. More on this in the Thermal Considerations topic.

** Auto-MDIX means automatic detection of "straight" and "cross" cables.*

I/O Pin Assignment and Pin Functions

The EM203 has two connectors -- main connector and magnetics connector. Depending on the EM203 version, magnetics connector can be soldered facing up or down, as described in the Mechanical Dimensions topic.



Main connector

#1	**MD***	Input	Mode selection pin
#2	**RST**	Input	Reset, active high
#3	**P3** **DTR***	Input/output Output	General-purpose input/output line Data terminal ready output

#4	**P2** **DSR***	Input/output Input	General-purpose input/output line Data set ready input
#5	**L3** **SG***	Output Output	LED output 3 Green status LED control line
#6	**L4** **SR***	Output Output	LED output 4 Red status LED control line
#7	**VCC**		Positive power input, 5V nominal, +/- 5%, app. 220mA
#8	**GND**		Ground
#9	**RX**	Input	Serial receive line
#10	**TX**	Output	Serial transmit line
#11	**P4** **CTS/SEL***	Input/output Input	General-purpose input/output line Clear to send input; full-/half-duplex selection input
#12	**P5** **RTS/DIR***	Input/output Output	General-purpose input/output line Request to send output (full-duplex mode); data direction control output (half-duplex mode)

* *Implemented in (supported through) firmware.*

Magnetics connector

#1	**RX+**	Input	Ethernet port, positive line of the differential input signal pair
#2	**RX-**	Input	Ethernet port, negative line of the differential input signal pair
#3	**AVCC**	Output	"Clean" 1.8V power output for magnetics circuitry
#4	**---**	---	---
#5	**---**	---	---
#6	**GND**		Ground
#7	**TX+**	Output	Ethernet port, positive line of the differential output signal pair
#8	**TX-**	Output	Ethernet port, negative line of the differential output signal pair

Serial Port and General-Purpose I/O Lines

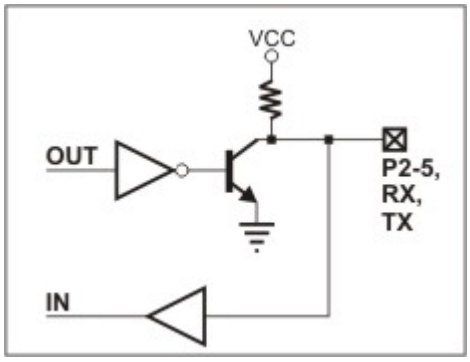
Main conn., #3	**P3** **DTR***	Input/output Output	General-purpose input/output line Data terminal ready output
Main conn., #4	**P2** **DSR***	Input/output Input	General-purpose input/output line Data set ready input
Main conn., #9	**RX**		Serial receive line
Main conn., #10	**TX**		Serial transmit line

Main conn., #11	**P4 CTS /SEL***	Input/ output	General-purpose input/output line Clear to send input; full-/half-duplex selection input
Main conn., #12	**P5 RTS /DIR***	Input/ output	General-purpose input/output line Request to send output (full-duplex mode); data direction control output (half-duplex mode)

* *Implemented in (supported through) firmware.*

The EM203 features a serial port (RX, TX lines) and several general-purpose I/O lines P2-P5 (there are no lines P0 and P1; line names were selected for naming compatibility with the EM100). All of the above lines are of CMOS type. From the hardware point of view, all general-purpose I/O lines can serve as inputs or outputs. Maximum load current for each I/O line is 10mA.

Simplified structure of EM203's I/O lines is shown on the circuit diagram below. All lines are "quasi-bidirectional" and can be viewed as open collector outputs with weak pull-up resistor. There is no explicit direction control. To "measure" an external signal applied to a pin the OUT line must first be set to HIGH. It is OK to drive the pin LOW externally, while the pin outputs HIGH internally.



The serial-to-Ethernet firmware of the EM203 maps certain serial port functions onto the general-purpose I/O pins- these functions are shown in blue in the table above. For example, P5 is a universal input/output but the application firmware can be set to turn this line into the RTS output of the serial port. Therefore, depending on your application, you can view P5 as a general-purpose I/O line or specific control line of the serial port (RTS).

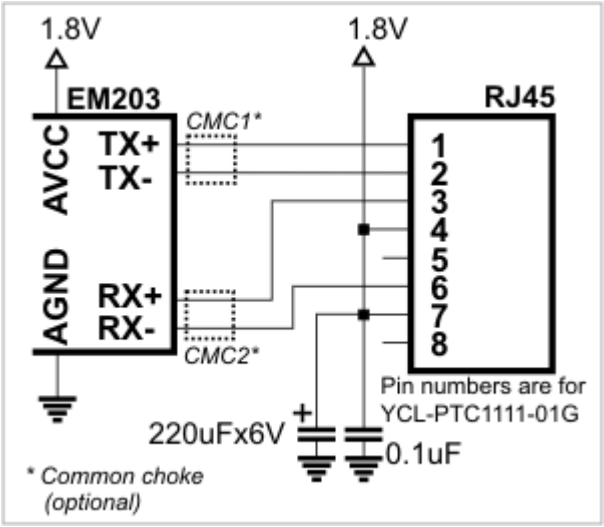
Being of CMOS type, the serial port and I/O lines of the EM203 can be connected directly to the serial port pins and I/O lines of most microcontrollers, microprocessors, etc. An interface IC, such as the MAX232, must be added to the EM203 externally if you want to connect the module to a "true" serial port (for example, COM port of the PC).

Logical signals on the serial port lines of the EM203 are active LOW. TX and RX lines are high when idle, start bit is LOW, stop bit is HIGH; LOW on CTS and RTS lines means "transmission allowed" and HIGH means "transmission not allowed". This is standard for CMOS-level serial ports and is exactly opposite to the signalling on the RS232 cables. Logical signals on the EM203 are inverted because standard RS232 ICs, such as the MAX232, invert the signals internally one more time.

As explained earlier, actual functionality of the I/O lines is firmware-dependent.

Ethernet Lines and Jack/Magnetics Data

Mag. conn., #1	**RX+**	Input	Ethernet port, positive line of the differential input signal pair
Mag. conn., #2	**RX-**	Input	Ethernet port, negative line of the differential input signal pair
Mag. conn., #9	**TX+**	Output	Ethernet port, positive line of the differential output signal pair
Mag. conn., #10	**TX-**	Output	Ethernet port, negative line of the differential output signal pair
Mag. conn., #11	**AVCC**	Output	"Clean" 1.8V power output for magnetics circuitry



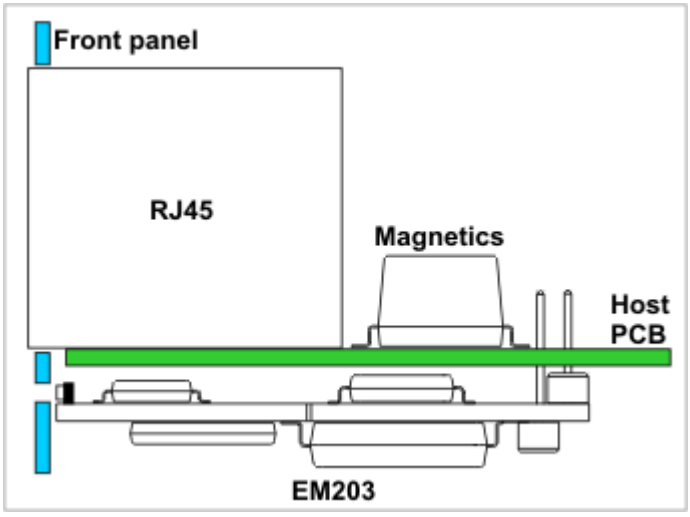
Ethernet port of the EM203 is of 10/100BaseT type. Onboard electronics of the EM203 do not include Ethernet magnetics, so magnetics circuitry must be connected externally to pins TX+, TX-, RX+, RX-, and AVCC. The AVCC pin outputs clean power for the magnetics circuitry, which is very sensitive to noise. The voltage on the AVCC output is 1.8V.

To reduce EMI you can optionally add common chokes into the TX and RX lines. Correctly selected common choke will have little negative impact on "useful" Ethernet signals, yet may substantially reduce unwanted EMI.

Do not interconnect the AVCC and VCC pins! This will permanently damage the EM203.

One possible (but not the only) way to place the EM203 on the host PCB is shown on the drawing below. The EM203 is installed on the bottom side of the host board, while a standard RJ45 jack and magnetics are placed on the top side of the board (alternatively, a jack with integrated magnetics can be used). In this configuration, the EM203 takes minimal board space and its LEDs are positioned close to the front panel of the host device. A small opening or window in the host device's housing

will allow the LEDs to be visible from the outside.



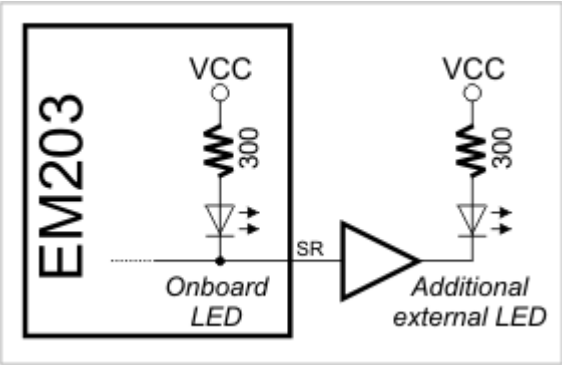
Another option is to combine the EM203 with the RJ203 jack/magnetics module.

LED Lines

Magnetics conn., #6	**L1(EG)**	Output	Green Ethernet status LED control line. The LED will be turned on when the EM203 links with the hub at 100Mb. The LED will be off if the link is established at 10Mb.
Magnetics conn., #7	**L2(EY)**	Output	Yellow Ethernet status LED control line. The LED will be turned on when "live" Ethernet cable is plugged into the Module. The LED will be temporarily switched off whenever an Ethernet packet is received.
Main conn., #5	**L3 SG***	Output Output	LED output 3 Green status LED
Main conn., #6	**L4 SR***	Output Output	LED output 4 Red status LED

** Implemented in (supported through) firmware.*

The EM203 has four onboard LEDs and four control lines -- L1-L4 -- to connect external LEDs in parallel with the onboard ones. External LEDs should be connected through a TTL buffer element. This will reduce the load on the EM203's internal circuit. Maximum load for each line without the buffer is 2mA.



The firmware of the EM203 uses L3 and L4 as "status LEDs" which display various status information depending on what firmware is running at the moment.

Power, Reset, and Mode Selection Lines

#7	**VCC**	Input	Positive power input, 5V nominal, +/- 5%, app. **230mA**
#8	**GND**		Ground
#2	**RST**	Input	Reset, active high
#1	**MD***	Input	Mode selection pin

* *Implemented in (supported through) firmware*

The EM203 should be powered from a stabilized DS power supply with output nominal voltage of 5V (+/- 5% tolerance). Current consumption of the EM203 is approximately 220mA (in 100BaseT mode).

Proper external reset is a must! Reset pulse should be active HIGH. We strongly advise against using low-cost RC-networks and other unreliable methods of generating reset. Reset should be applied for as long as the power supply voltage is below 4.6V. We recommend using a dedicated reset IC with brownout detection, such as the MAX810. Reset pulse length should be no less than 50ms, counting from the moment the power supply voltage exceeds 4.6V.

If the EM203 is used to serve as a communications co-processor in a larger system that has its own CPU it is also OK to control the RST line of the EM203 through a general-purpose I/O pin of the "host" microcontroller. I/O pins of many microcontrollers default to HIGH after the powerup and this means that the reset will be applied to the EM203 in parallel with the reset of the microcontroller. All the host microcontroller has to do is release the EM203 from reset at an appropriate time by switching the state of the I/O line to LOW.

The MD line of the EM203 is used to select the operating mode of the EM203 and/or its application firmware. This pin has the following functionality:

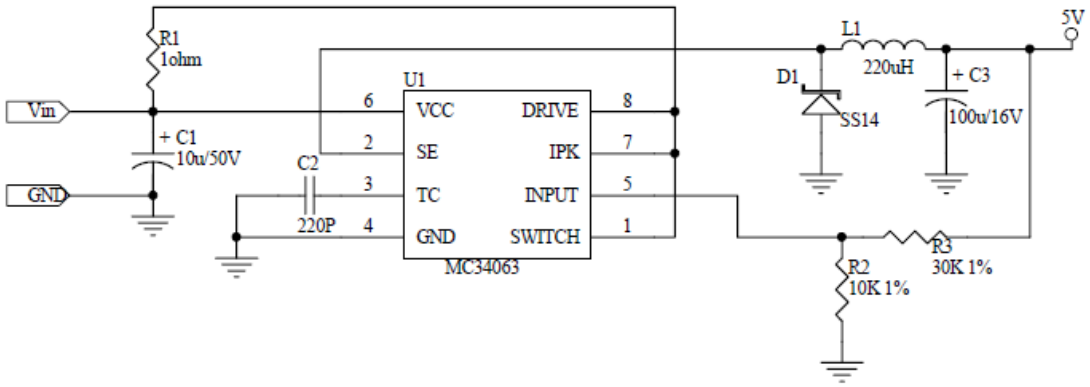
- When the EM203 powers up it verifies the state of the MD input. If the MD input is at HIGH the EM203 proceeds to verifying and running the application firmware loaded into its internal FLASH memory. If the MD input is at LOW the EM203 enters the serial upgrade mode.
- When the serial-to-Ethernet firmware is already running the MD line is used to make the EM203 enter the serial programming mode.

When the EM203 is used as a co-processor in a host system the MD line can be also controlled by the host microcontroller. Ability to control both the RST and the MD lines will allow the host microcontroller to switch between the operating modes

of the EM203.

Power supply circuit

Many power supply circuits will work well. The one below is being used by Tibbo. The circuit can handle input voltages in the 9-24V range.



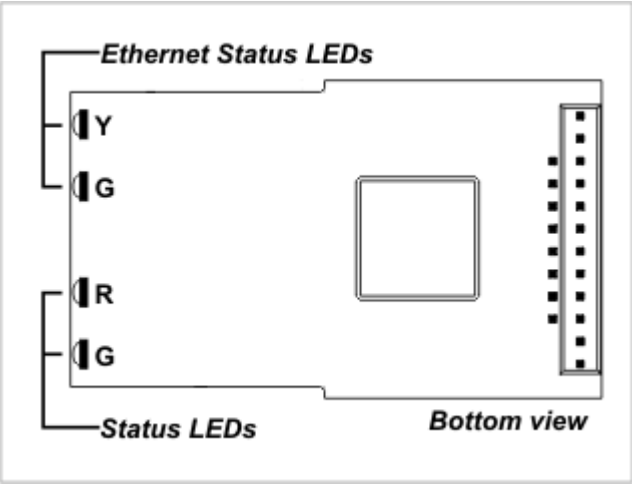
Notes:

- U1 is a popular power IC manufactured by ON SEMICONDUCTOR.
- C1 and C2 capacitors: Do not use SMD capacitors -- use regular through-hole aluminum capacitors. This really helps reduce noise produced by the power supply.
- This is an analog circuit, so layout matters. Apply reasonable "good layout" effort.

Ideally, one should use an oscilloscope to see what sort of "square wave" the power supply generates, both at low and high input voltages, as well as light and heavy loads. There are no recipes here -- just try and see what works for your circuit.

Onboard LEDs

The EM203 features four onboard status LEDs. The LEDs are strategically positioned on the edge of the module's board. Your product can have a small window or opening on its cover to make the LEDs of the EM203 visible from the outside (as shown on the drawing in the Ethernet Lines and Jack/Magnetics Data). When the EM203 is used in combination with the RJ203 module, the status LEDs are visible through a transparent portion of the RJ203's housing.



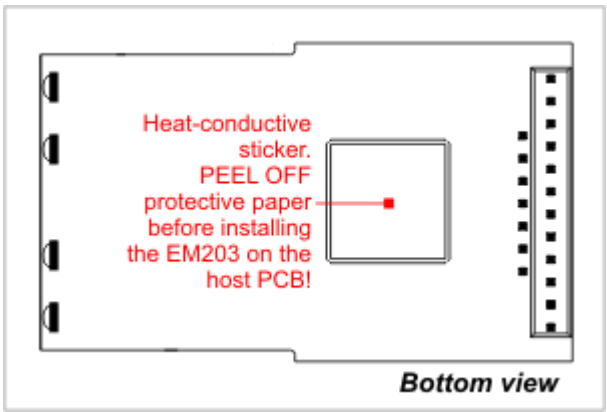
Onboard LEDs have the following function:

- Yellow Ethernet status LED is turned on when the EM203 links with the hub at 100Mb. The LED is off when the link is established at 10Mb.
- Green Ethernet status LED is turned on when "live" Ethernet cable is plugged into the Module. The LED is temporarily switched off whenever an Ethernet packet is received.
- Red and green status LEDs are under the control of EM203's firmware.

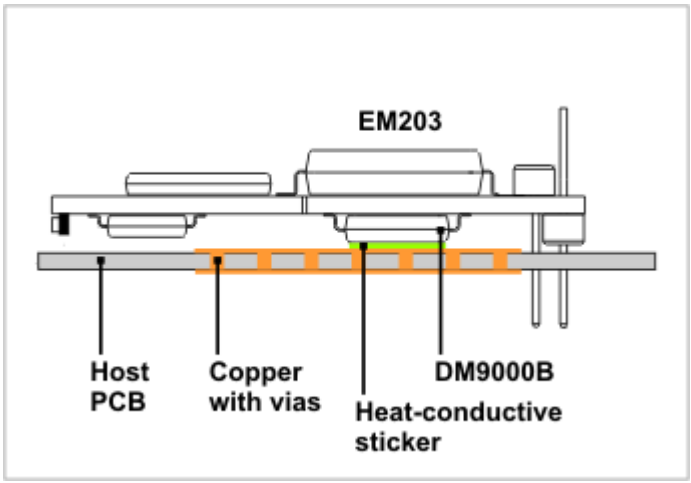
The EM203 also has four LED control lines that allow you to add external LEDs in parallel with the onboard ones.

Thermal considerations

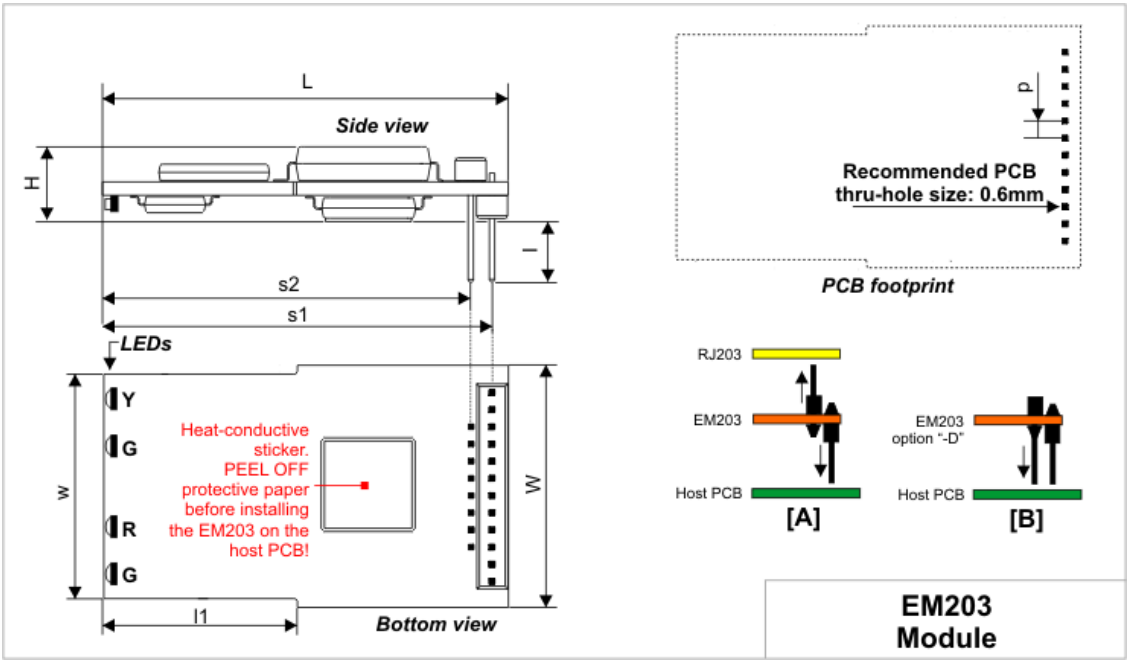
The DM9000B Ethernet controller of the EM203 can become very hot during normal module operation. To aid the module in dissipating excess heat, a special heat-conductive sticker is applied to the top of the DM9000B. Protective paper of the sticker **MUST BE REMOVED** prior to installing the module on the host PCB.



To further lower the operating temperature of the EM203 we advise you to arrange a copper area on the host PCB and in contact with the heat-conductive sticker. Best results are achieved when the copper area is larger, and also when two copper areas are provided on both sides of the host PCB and interconnected by a number of large vias.



Mechanical Dimensions



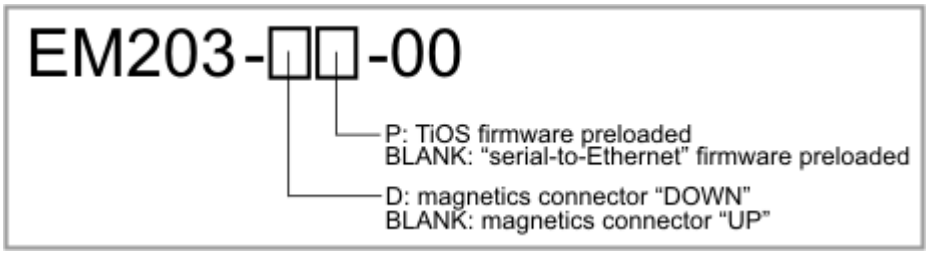
L	Max.	30.1	Length
W	Max.	18.1	Width
H	Max.	5.5	Height
l1	Aver.	14.4	Length of the narrower part of the board
w	Max.	16.7	Width at the narrower part of the board
p	Aver.	1.27	Pin pitch
s1	Aver.	28.9	Distance from the edge of the board to the pins of the main connector

s2	Aver.	27.3	Distance from the edge of the board to the pins of the magnetics connector
l	Min.	4.0	Connector pin length

All dimensions are in millimeters

Specifications and Ordering Info

The EM203 has several versions available. Device numbering scheme is as follows:



Examples of valid model numbers

Model number	Description
EM203-00	Connector configured to mate with the EM203 jack/magnetics module, the module will be supplied with the "serial-to-Ethernet" firmware
EM203-D-00	Connectors configured to mate with the host PCB, the module will be supplied with the "serial-to-Ethernet" firmware
EM203-DP-00	Connectors configured to mate with the host PCB, the module will be supplied with the TiOS firmware

Ordering the EM203 and RJ203 module combination

The EM203 can also be ordered in combination with the RJ203 module. To receive these two modules pre-assembled together, please specify "RJ203+EM203" on your order if you wish to receive the EM203 with the "serial-to-Ethernet" firmware preloaded, or "RJ203+EM203-P" if you wish to receive the EM203 with the TiOS firmware preloaded.

Specifications

Ethernet interface	10/100BaseT Ethernet, Auto-MDIX, magnetics **not** built-in
Serial ports	1 port, CMOS-level
UART capabilities	Baudrates up to 115'200bps; none/even/odd/mark/space parity and 7/8 bits/character; full-duplex UART mode with optional flow control* and half-duplex UART mode with automatic direction control*; RX, TX, RTS*, CTS*, DTR*, and DSR* lines.

LEDs	x4 : red and green status LEDs*, yellow and green Ethernet status LEDs
Max. load current for each I/O line	10mA
Supported network protocols*	UDP*, TCP*, ICMP (ping)*, and DHCP*.
Nominal power supply voltage (VCC pin)	DC 5.0V, +/- 5%
Device reset	Proper reset must be provided externally, RST pin has active HIGH polarity
Max. operating current	220mA
Operating temperature	-20 to +70 degrees C
Operating relative humidity	10-90%
Mechanical dimensions (excl. leads)	30.1x18.1x5.5mm
Packaging	EM203 devices: tray, 30 modules/tray EM203+RJ203 module combination: tube, 10 modules/tube

** Implemented in (supported through) firmware.*

Note: all specifications are for reference only. TIBBO assumes no responsibility for any errors which may appear in this Manual, and does not make any commitment to update the information contained herein.

RJ203A Jack/Magnetics Module

Patent pending

The RJ203 is an "Ethernet front-end" module that contains 10/100BaseT Ethernet magnetics and a standard RJ45 jack. Module's magnetics are designed to work with Davicom's DM9000A Ethernet controller.

Unique patent-pending design of the module minimizes module's footprint and allows you to put other components required on your host board under the RJ203, thus saving valuable host board space. Moreover, translucent housing of the RJ203's face enables you to place status LEDs directly on the host board and have these LEDs visible through the front face of the RJ203.

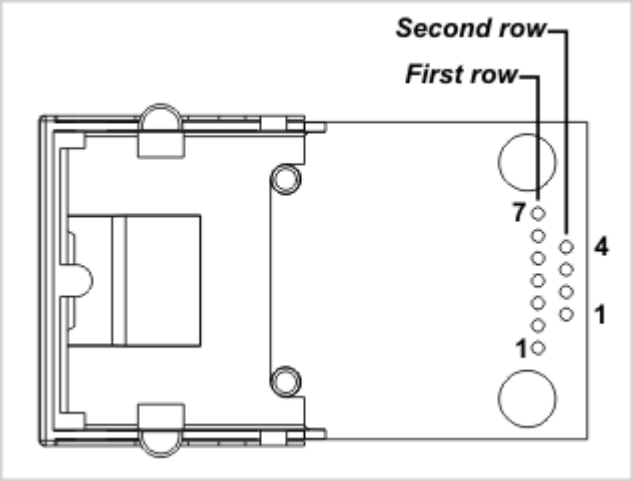
There are two ways in which you can utilize the RJ203 in your design:

- You can interface the RJ203 to the DM9000A IC located, together with the CPU and other necessary components, directly on your host PCB.

- Alternatively, you can use the RJ203 in combination with the EM203A Ethernet-to-serial module. The EM203A fits right "under" the RJ203, thus taking no additional space on the host PCB.

Interface Pads

The RJ203A has two rows of interface pads.



First pad row

#1	**RX+**	Output	Ethernet port, positive line of the differential input signal pair
#2	**RX-**	Output	Ethernet port, negative line of the differential input signal pair
#3	**AVCC**	Input	"Clean" 3.3V power output for magnetics circuitry
#4	**TX+**	Input	Ethernet port, positive line of the differential output signal pair
#5	**TX-**	Input	Ethernet port, negative line of the differential output signal pair
#6	**---**		*<unused>*
#7	**---**		*<unused>*

Second pad row

#1	**---**		*<unused>*
#2	**---**		*<unused>*
#3	**---**		*<unused>*
#4	**GND**		Ground

Interfacing the RJ203A to the DM9000A

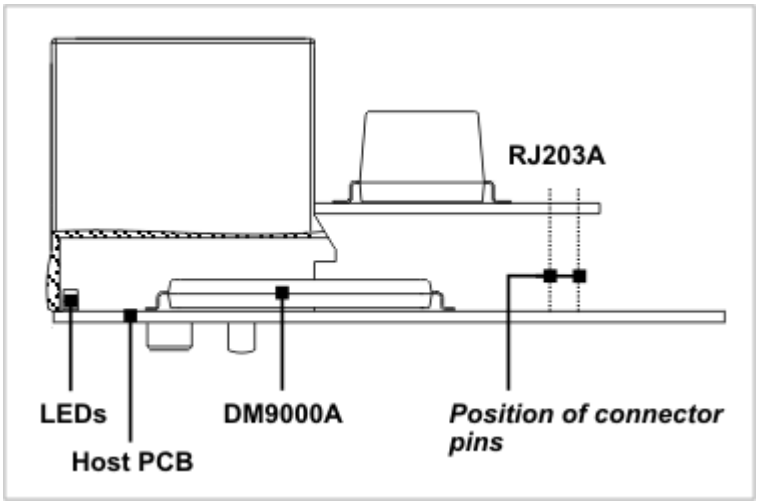
The RJ203A module interfaces directly to the DAVICOM's DM9000A Ethernet controller. The following table details the interconnection between the DM9000A and the interface pads of the RJ203A:

DM9000A	RJ203A
RXI- (#30)	RX- (first row, #2)
RXI+ (#29)	RX+ (first row, #1)
TX0- (#33)	TX- (first row, #5)
TX0+ (#34)	TX+ (first row, #4)

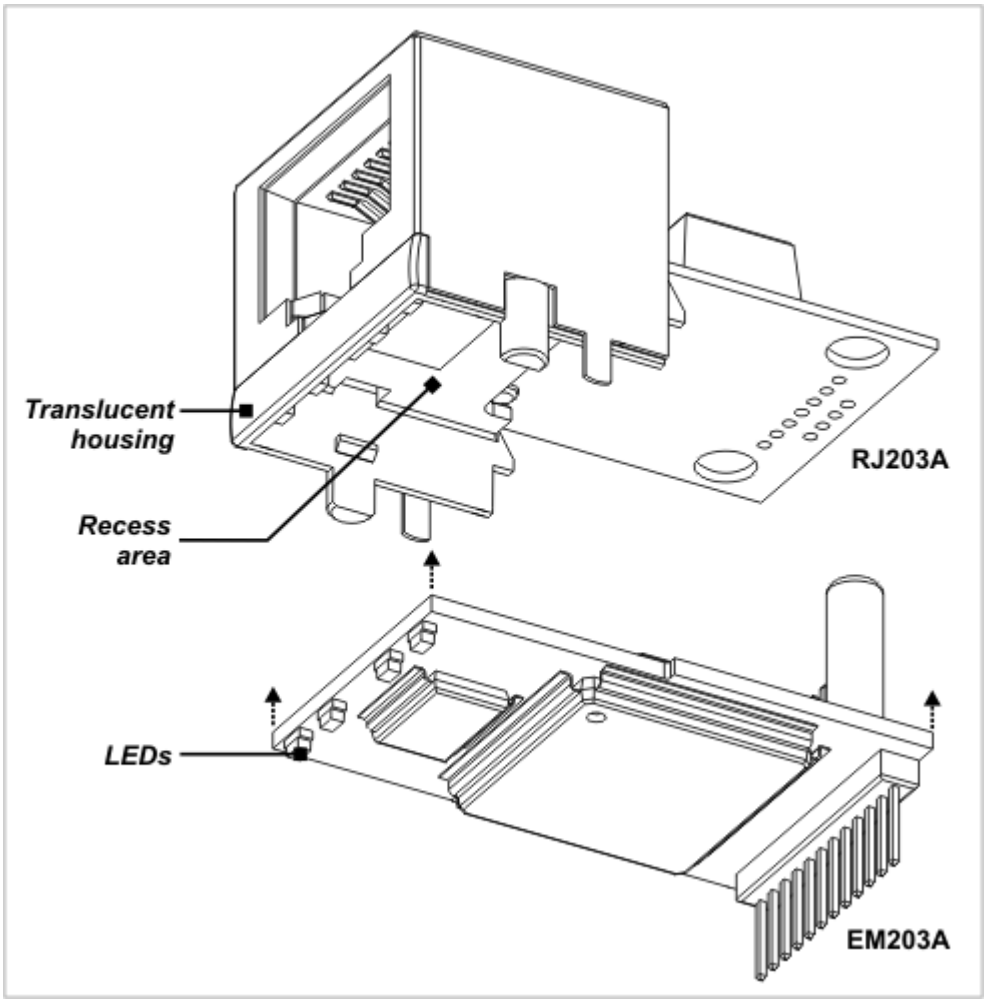
AVDD (#28)	AVCC (first row, #3)

Don't forget to connect grounds too!

To take full advantage of the unique space-saving design of the RJ203A, place the DM9000A (and/or other components as you see fit) under the module. The housing of the module has a substantial recess area under the RJ45 jack. This area can be utilized to accommodate various board components. Moreover, the housing of the RJ203A is made of a translucent material, so you can also place necessary status LEDs within the recess area and in the proximity to the front wall of the RJ203A. This way, your status LEDs will be visible through the translucent front face of the RJ203A. Four to six LEDs can easily fit along that front wall.



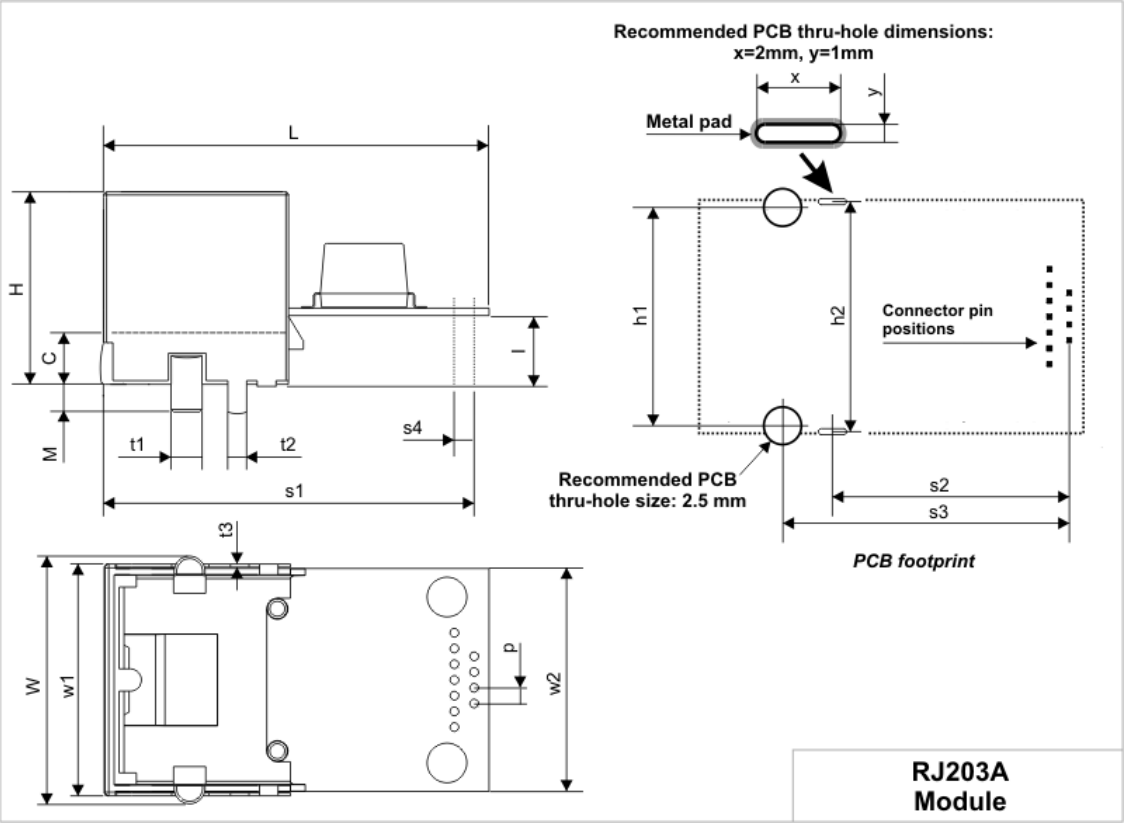
Using the RJ203A With the EM203A Module



The RJ203A can also be used in combination with the EM203A module. Connector pins of the EM203A are designed to mate with interface pads of the RJ203A. The EM203A module itself fits "under" the RJ203A and partially within the recess area provided by the RJ203A. This recess area is formed by a translucent housing of the RJ203A. When the EM203A is combined with the RJ203A, the status LEDs of the EM203A become positioned close to the translucent front wall of the RJ203As and remain visible through the front face of the RJ203A.

Additional information can be found in the Mechanical Dimensions: RJ203A +EM203A topic.

Mechanical Dimensions: RJ203A

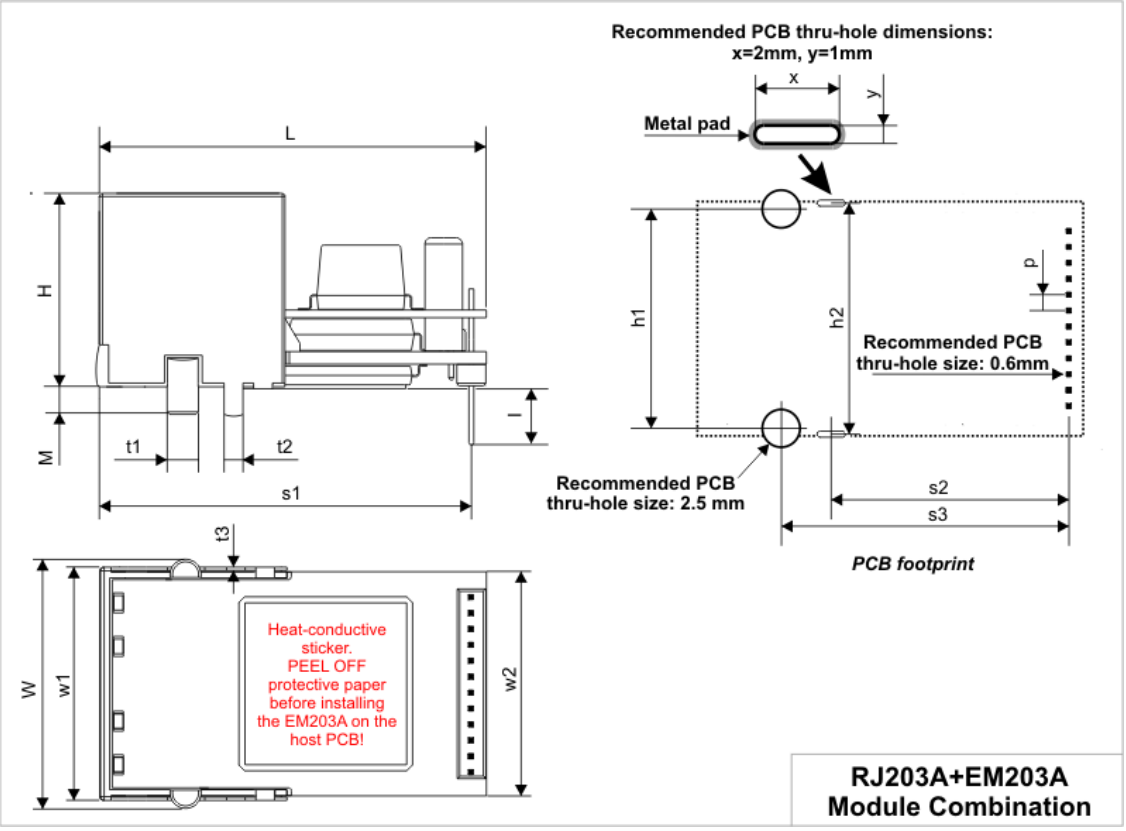


L	Max.	31.0	Length
W	Max.	20.0	Width
H	Max.	15.5	Height
I	Aver.	5.5	Clearance between the installation surface and the bottom of the RJ203A's board
w1	Max.	19.0	Width at the face excluding mounting stands
w2	Max.	18.1	Board width
M	Min.	1.9	Mounting stand and tail height
t1	Aver.	2.5	Mounting stand diameter
t2	Aver.	1.5	Solder tail width
t3	Aver.	0.25	Solder tail thickness
p	Aver.	1.27	Connector pad pitch
s1	Aver.	29.7	Distance from device face to the second pad row
s2	Aver.	19.0	Distance from the second pad row to the vertical centerline of solder tails
s3	Aver.	23.0	Distance from the second pad row to the vertical centerline of mounting stands

s4	Aver.	1.6	Distance between pad rows
h1	Aver.	17.5	Distance between the horizontal centerlines of mounting stands
h2	Aver.	18.5	Distance between the horizontal centerlines of solder tails
C	Min.	4.4	Clearance from the installation surface to the top wall of the recess area of the housing

All dimensions are in millimeters

Mechanical Dimensions: RJ203A+EM203A



L	Max.	31.0	Length
W	Max.	20.0	Width
H	Max.	15.5	Height
l	Min.	4.0	Connector pin length
w1	Max.	19.0	Width at the face excluding mounting stands
w2	Max.	18.1	Board width
M	Min.	1.9	Mounting stand and tail height
t1	Aver.	2.5	Mounting stand diameter
t2	Aver.	1.5	Solder tail width

t3	Aver.	0.25	Solder tail thickness
p	Aver.	1.27	Connector pin pitch
s1	Aver.	29.7	Distance from the face to the connector pins
s2	Aver.	19.0	Distance from connector pins to the vertical centerline of solder tails
s3	Aver.	23.0	Distance from connector pins to the vertical centerline of mounting stands
h1	Aver.	17.5	Distance between the horizontal centerlines of mounting stands
h2	Aver.	18.5	Distance between the horizontal centerlines of solder tails

All dimensions are in millimeters

Specifications and Ordering Info

The RJ203A has a single model available -- the RJ203A-00.

Ordering the EM203A and RJ203A module combination

The RJ203A can also be ordered in combination with the EM203A module. To receive these two modules pre-assembled together, please specify "RJ203A+EM203A" on your order if you wish to receive the EM203A with the "serial-to-Ethernet" firmware preloaded, or "RJ203A+EM203A-P" if you wish to receive the EM203A with the TiOS firmware preloaded.

Specifications

Jack type	Standard RJ45 Ethernet jack
Magnetics type	10/100BaseT, designed to work with DAVICOM DM9000A Ethernet controller
Operating temperature	-20 to +70 degrees C
Operating relative humidity	10-90%
Mechanical dimensions (excl. leads)	31.0x20.0x15.5 mm
Packaging	RJ203A devices: tray, 30 modules/tray EM203A+RJ203A module combination: tube, 10 modules/tube

Note: all specifications are for reference only. TIBBO assumes no responsibility for any errors which may appear in this Manual, and does not make any commitment to update the information contained herein.

RJ203 Jack/Magnetics Module

Patent pending

The RJ203 is an "Ethernet front-end" module that contains 10/100BaseT Ethernet magnetics and a standard RJ45 jack. Module's magnetics are designed to work with Davicom's DM9000B Ethernet controller.

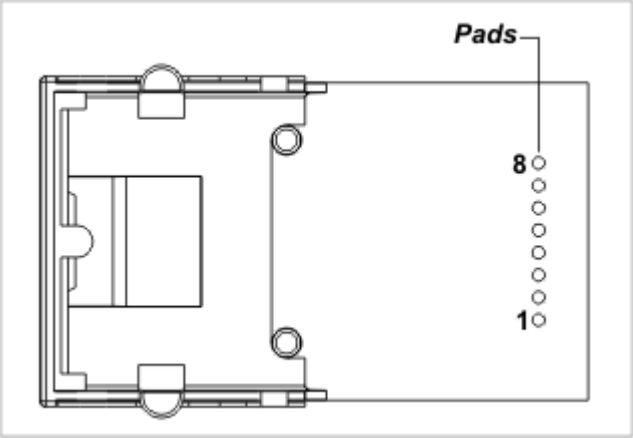
Unique patent-pending design of the module minimizes module's footprint and allows you to put other components required on your host board under the RJ203, thus saving valuable host board space. Moreover, translucent housing of the RJ203's face enables you to place status LEDs directly on the host board and have these LEDs visible through the front face of the RJ203.

There are two ways in which you can utilize the RJ203 in your design:

- You can interface the RJ203 to the DM9000B IC located, together with the CPU and other necessary components, directly on your host PCB.
- Alternatively, you can use the RJ203 in combination with the EM203 Ethernet-to-serial module. The EM203 fits right "under" the RJ203, thus taking no additional space on the host PCB.

Interface Pads

The RJ203 has a single row or interface pins.



#1	**RX+**	Output	Ethernet port, positive line of the differential input signal pair
#2	**RX-**	Output	Ethernet port, negative line of the differential input signal pair
#3	**AVCC**	Input	"Clean" 1.8V power output for magnetics circuitry
#4	**---**	---	---
#5	**---**	---	---
#6	**GND**		Ground

#7	**TX+**	Input	Ethernet port, positive line of the differential output signal pair
#8	**TX-**	Input	Ethernet port, negative line of the differential output signal pair

Interfacing the RJ203 to the DM9000B

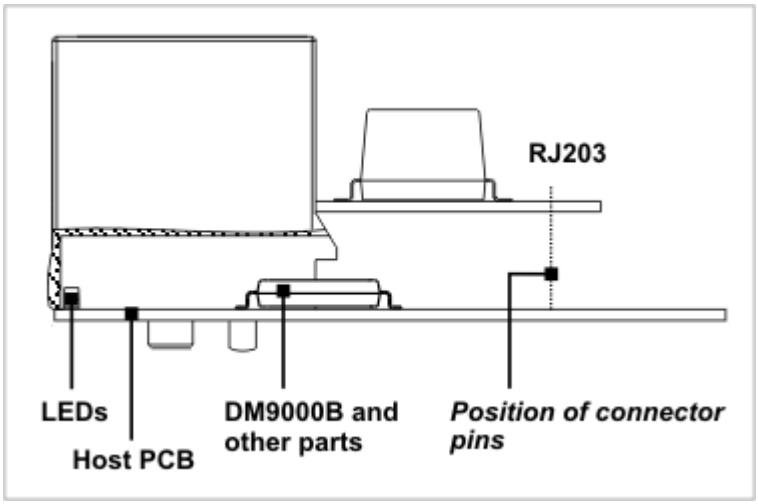
The RJ203 module interfaces directly to the DAVICOM's DM9000B Ethernet controller. The following table details the interconnection between the DM9000A and the interface pads of the RJ203:

DM9000B	**RJ203**
RX+ (#3)	RX+ (#1)
RX- (#4)	RX- (#2)
TX+ (#7)	TX+ (#7)
TX- (#8)	TX- (#8)
RXVDD (#2), TXVDD (#9)	AVCC (#3)

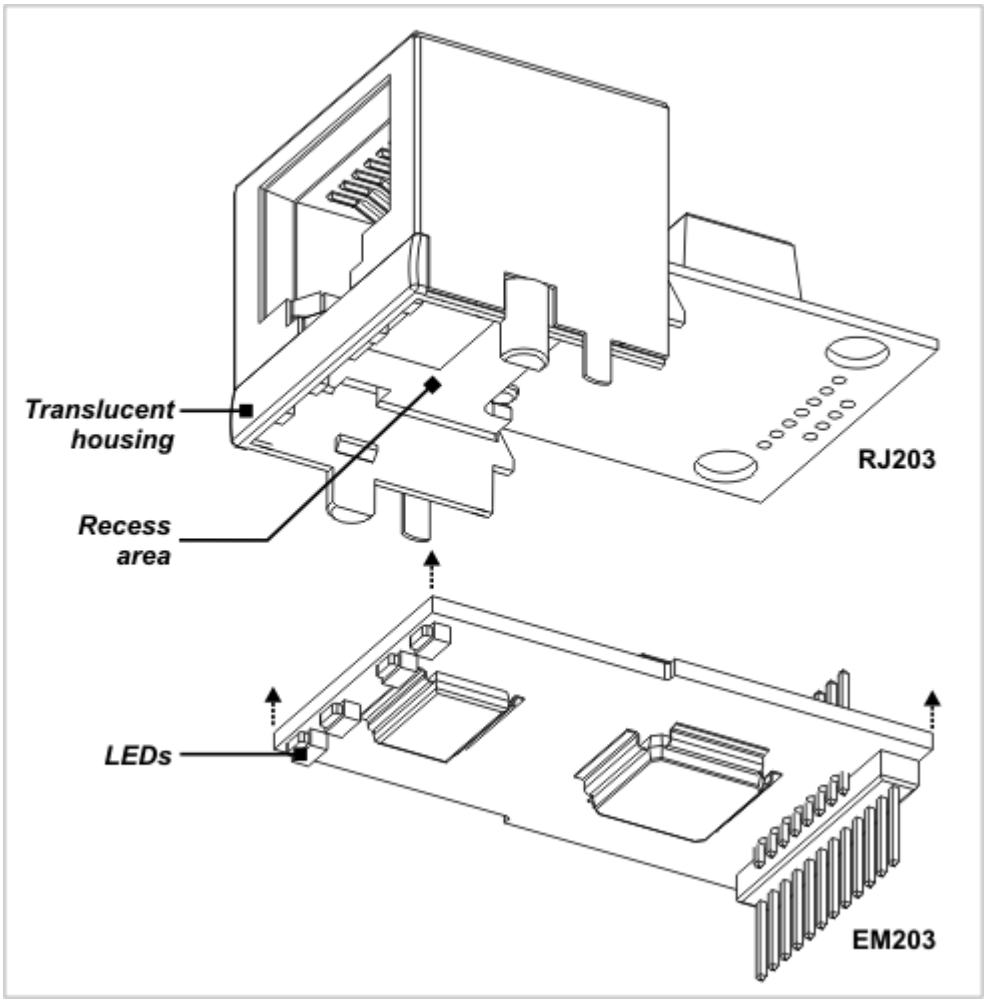
Don't forget to connect grounds too!

Additional passive components, such as resistors and capacitors must also be placed near the DM9000B and connected to RX and TX lines. For detailed information see the DM9000B datasheet.

To take full advantage of the unique space-saving design of the RJ203, place the DM9000B (and/or any other components as you see fit) under the module. The housing of the module has a substantial recess area under the RJ45 jack. This area can be utilized to accommodate various board components. Moreover, the housing of the RJ203 is made of a translucent material, so you can also place necessary status LEDs within the recess area and in the proximity to the front wall of the RJ203. This way, your status LEDs will be visible through the translucent front face of the RJ203. Four to six LEDs can easily fit along that front wall.



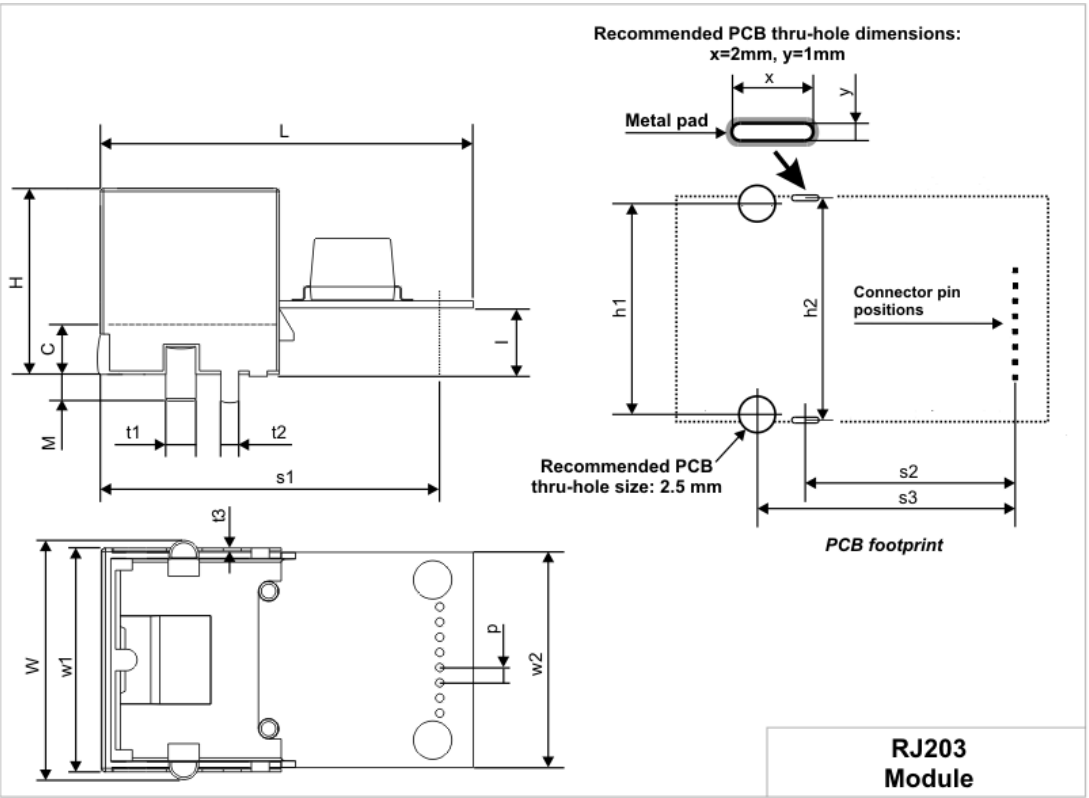
Using the RJ203 With the EM203 Module



The RJ203 can also be used in combination with the EM203 module. Connector pins of the EM203 are designed to mate with interface pads of the RJ203. The EM203 module itself fits "under" the RJ203 and partially within the recess area provided by the RJ203. This recess area is formed by a translucent housing of the RJ203. When the EM203 is combined with the RJ203, the status LEDs of the EM203 become positioned close to the translucent front wall of the RJ203s and remain visible through the front face of the RJ203.

Additional information can be found in the Mechanical Dimensions: RJ203+EM203 topic.

Mechanical Dimensions: RJ203

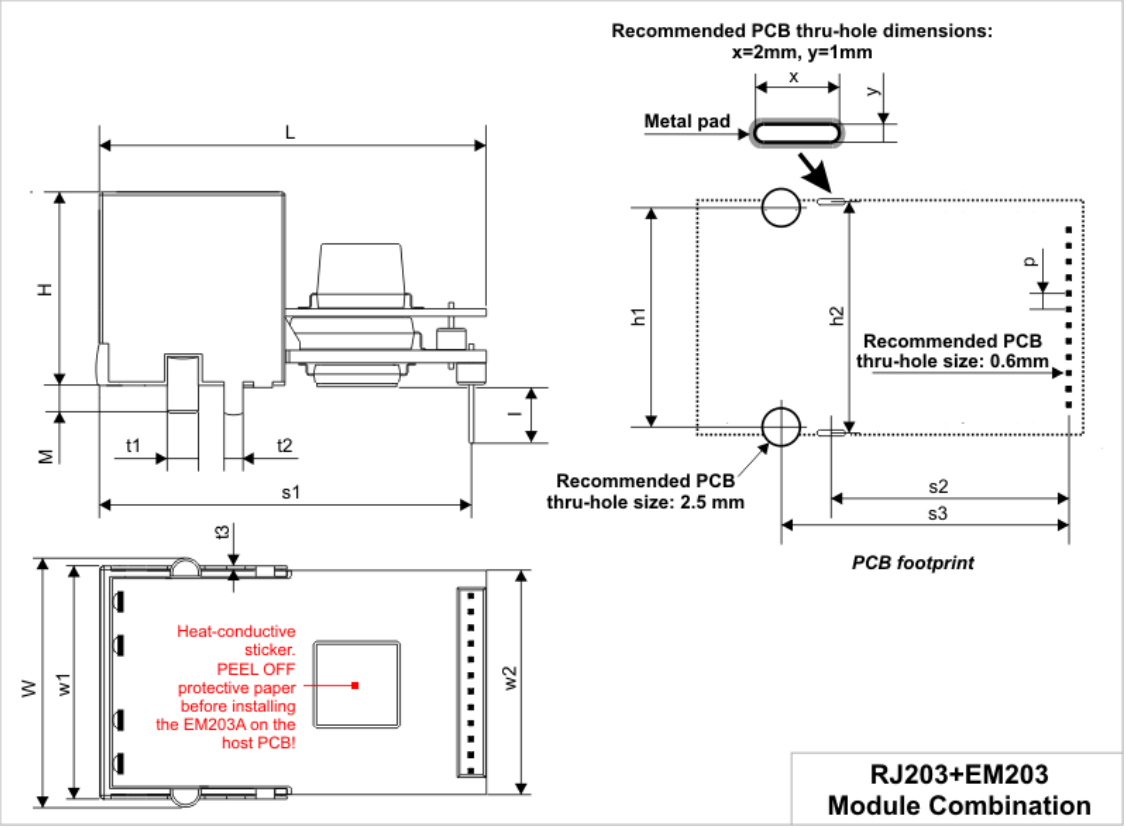


L	Max.	31.0	Length
W	Max.	20.0	Width
H	Max.	15.5	Height
I	Aver.	5.5	Clearance between the installation surface and the bottom of the RJ203's board
w1	Max.	19.0	Width at the face excluding mounting stands
w2	Max.	18.1	Board width
M	Min.	1.9	Mounting stand and tail height
t1	Aver.	2.5	Mounting stand diameter
t2	Aver.	1.5	Solder tail width
t3	Aver.	0.25	Solder tail thickness
p	Aver.	1.27	Connector pad pitch
s1	Aver.	28.1	Distance from device face to the pad row
s2	Aver.	17.4	Distance from the second pad row to the vertical centerline of solder tails
s3	Aver.	21.4	Distance from the second pad row to the vertical centerline of mounting stands

h1	Aver.	17.5	Distance between the horizontal centerlines of mounting stands
h2	Aver.	18.5	Distance between the horizontal centerlines of solder tails
C	Min.	4.4	Clearance from the installation surface to the top wall of the recess area of the housing

All dimensions are in millimeters

Mechanical Dimensions: RJ203+EM203



L	Max.	31.0	Length
W	Max.	20.0	Width
H	Max.	15.5	Height
l	Min.	4.0	Connector pin length
w1	Max.	19.0	Width at the face excluding mounting stands
w2	Max.	18.1	Board width
M	Min.	1.9	Mounting stand and tail height
t1	Aver.	2.5	Mounting stand diameter
t2	Aver.	1.5	Solder tail width
t3	Aver.	0.25	Solder tail thickness

p	Aver.	1.27	Connector pin pitch
s1	Aver.	29.7	Distance from the face to the connector pins
s2	Aver.	19.0	Distance from connector pins to the vertical centerline of solder tails
s3	Aver.	23.0	Distance from connector pins to the vertical centerline of mounting stands
h1	Aver.	17.5	Distance between the horizontal centerlines of mounting stands
h2	Aver.	18.5	Distance between the horizontal centerlines of solder tails

All dimensions are in millimeters

Specifications and Ordering Info

The RJ203 has a single model available -- the RJ203-00.

Ordering the EM203 and RJ203 module combination

The RJ203 can also be ordered in combination with the EM203 module. To receive these two modules pre-assembled together, please specify "RJ203+EM203" on your order if you wish to receive the EM203 with the "serial-to-Ethernet" firmware preloaded, or "RJ203+EM203-P" if you wish to receive the EM203 with the TiOS firmware preloaded.

Specifications

Jack type	Standard RJ45 Ethernet jack
Magnetics type	10/100BaseT, designed to work with DAVICOM DM9000B Ethernet controller
Operating temperature	-20 to +70 degrees C
Operating relative humidity	10-90%
Mechanical dimensions (excl. leads)	31.0x20.0x15.5 mm
Packaging	RJ203 devices: tray, 30 modules/tray EM203+RJ203 module combination: tube, 10 modules/tube

Note: all specifications are for reference only. TIBBO assumes no responsibility for any errors which may appear in this Manual, and does not make any commitment to update the information contained herein.

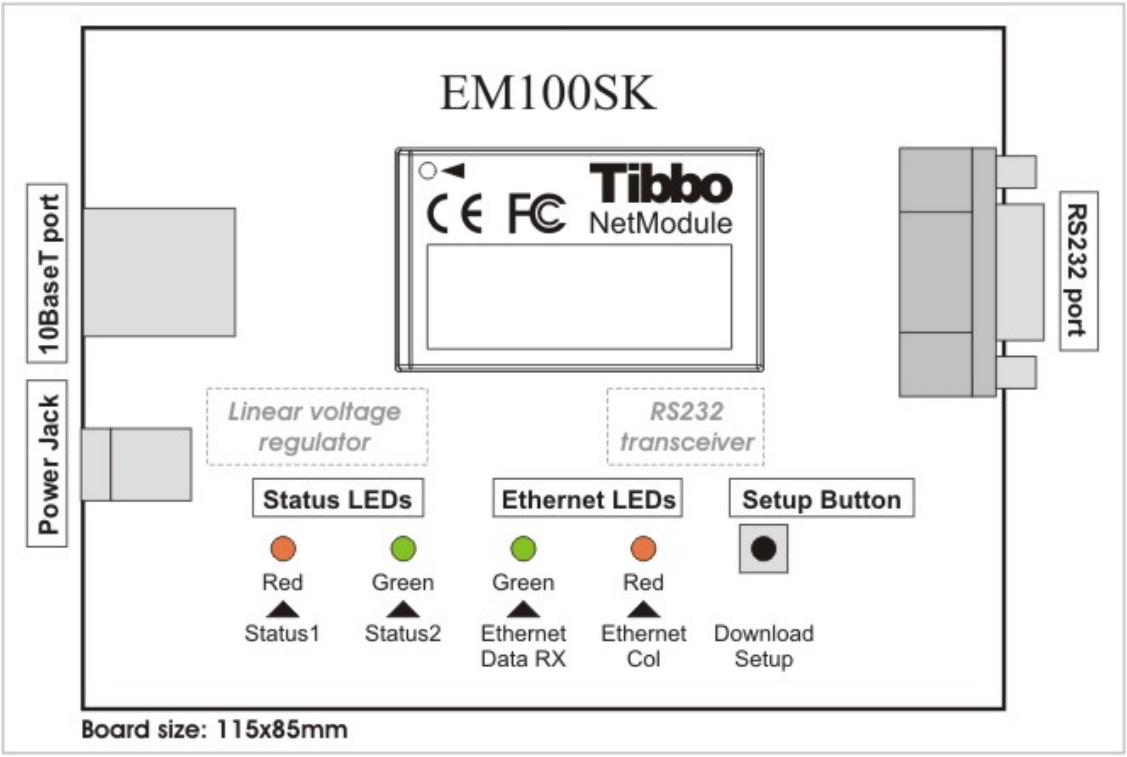
Boards

This part of documentation describes Ethernet-to-serial Boards supplied by Tibbo. These boards can be used for convenient evaluation and testing of Tibbo Ethernet-to-serial Modules. At the same time, some boards are also suitable for use in production devices as "faster implementation" alternatives to Modules.

The following Boards are currently manufactured:

- EM100-EV Evaluation Board
- EM120/EM200-EV Evaluation Board

EM100-EV Evaluation Board

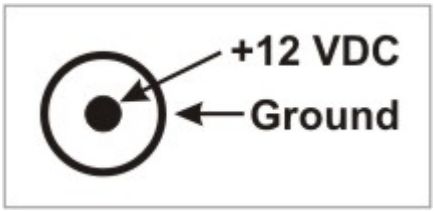


The EM100-EV Evaluation Board offers a convenient way of testing EM100 Ethernet Module. The board features the following components:

- A socket for EM100 Module installation
- Power jack and a linear voltage regulator circuitry (12VDC-->5VDC, adaptor current rating must be no less than 500mA)
- RJ45 connector
- DB9M RS232 connector and RS232 transceiver (supported signals are RX, TX, RTS, CTS)
- Setup button (connected to the MD line of the EM100)
- Ethernet LEDs and Status LEDs (connected to LED lines of the EM100)

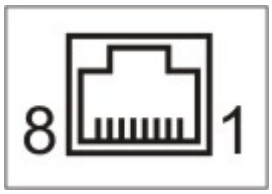
Power Jack

Power Jack of the EM100-EV accepts "large" power connectors with 5.5mm diameter. Use ARP-1014, ARP-1015A, or ARP-1018Apower adaptor supplied by Tibbo or similar adaptor. Nominal voltage is 12VDC and adaptor current rating should be at least 500mA. On the power jack, the ground is "on the outside", as shown on the figure below.



Ethernet Port Pin Assignment

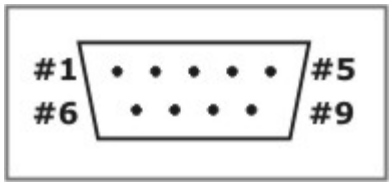
RJ45 Ethernet connector has the following pin assignment:



#1	**TX+**
#2	**TX-**
#3	**RX+**
#4	*<No connection>*
#5	*<No connection>*
#6	**RX-**
#7	*<No connection>*
#8	*<No connection>*

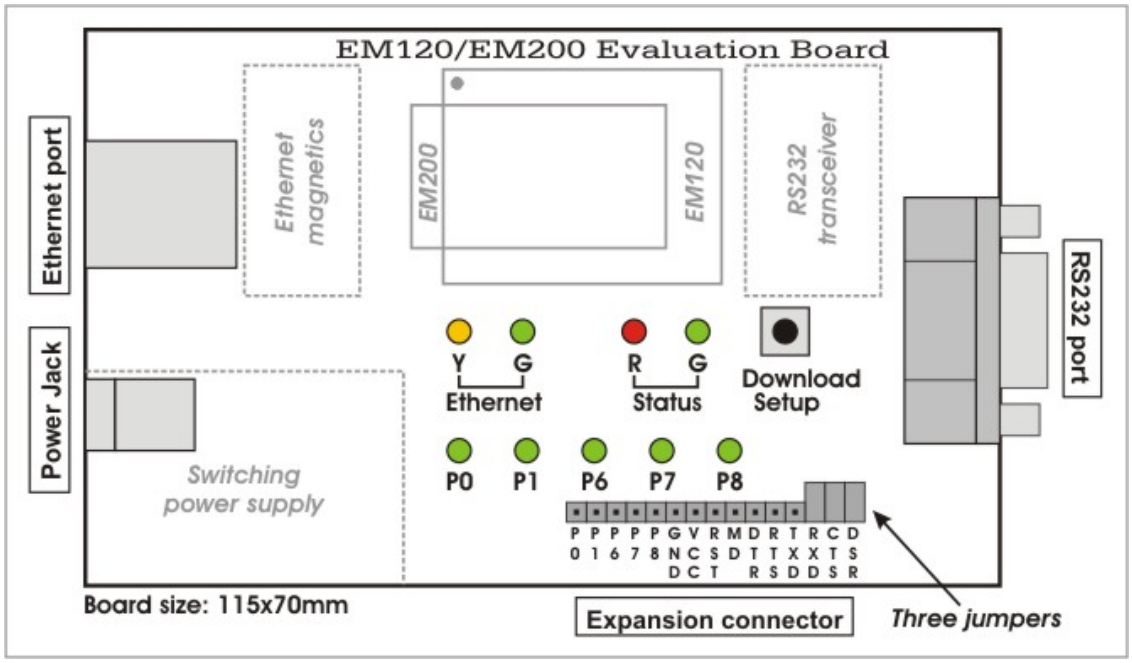
RS232 Port Pin Assignment

DB9M RS232 connector has the following pin assignment:



#1	*<No connection>*
#2	**RX** (input)
#3	**TX** (output)
#4	*<No connection>*
#5	**Ground**
#6	*<No connection>*
#7	**RTS** (output)
#8	**CTS** (input)
#9	*<No connection>*

EM120/EM200-EV Evaluation Board

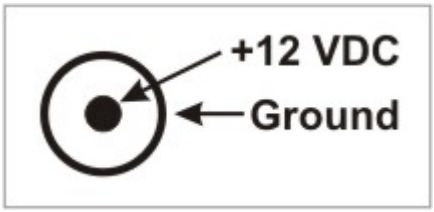


The EM120/200-EV Evaluation Board offers a convenient way of testing EM120 and EM200 Ethernet Modules. The board features the following components:

- A socket for EM120 or EM200 installation
- Power jack and a switching power regulator (12VDC-->5VDC, adaptor current rating must be no less than 500mA)
- RJ45 connector and 10/100BaseT Ethernet Magnetics (EM120 and EM200 do not have built-in magnetics)
- DB9M RS232 connector and RS232 transceiver (supported signals are RX, TX, RTS, CTS, DTR, DSR)
- Setup button (connected to the MD line of EM120/EM200)
- Two Ethernet LEDs and two status LEDs (connected to LED lines of EM120/200)
- Five additional LEDs connected to lines P0, P1, P6-8of the EM120/EM200
- 15-pin expansion connector provides access to EM120/EM200's serial and general-purpose I/O pins (therefore, all I/O lines on this connector are of TTL type)

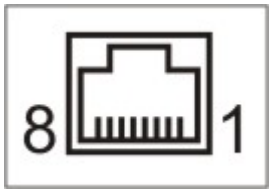
Power Jack

Power Jack of the EM120/EM200-EV accepts "large" power connectors with 5.5mm diameter. Use ARP-1014, ARP-1015A, or ARP-1018A power adaptor supplied by Tibbo or similar adaptor. Nominal voltage is 12VDC and adaptor current rating should be at least 500mA. On the power jack, the ground is "on the outside", as shown on the figure below.



Ethernet Port Pin Assignment

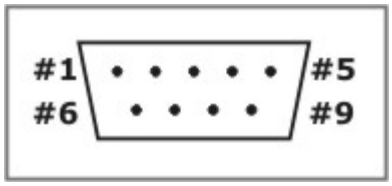
RJ45 Ethernet connector has the following pin assignment:



#1	**TX+**
#2	**TX-**
#3	**RX+**
#4	*<No connection>*
#5	*<No connection>*
#6	**RX-**
#7	*<No connection>*
#8	*<No connection>*

RS232 Port Pin Assignment

DB9M RS232 connector has the following pin assignment:



#1	*<No connection>*
#2	**RX** (input)
#3	**TX** (output)
#4	**DTR** (output)
#5	**Ground**
#6	**DSR** (input)
#7	**RTS** (output)
#8	**CTS** (input)
#9	*<No connection>*

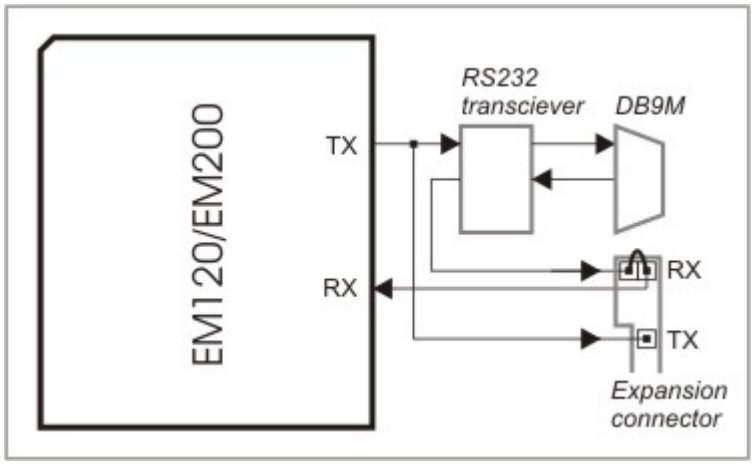
Expansion Connector Pin Assignment

15-pin expansion connector has the following pin assignment:

P0	Connected to pin P0 of EM120/EM200
P1	Connected to pin P1 of EM120/EM200
P6	Connected to pin P6 of EM120/EM200
P7	Connected to pin P7 of EM120/EM200
P8	Connected to pin P8 of EM120/EM200
GND	Ground
VCC	**+**5V from the EM120/EM200-EV board. Available "spare" current about 50mA
RST	Reset (active high) from the EM120/EM200-EV board. The signal is generated by an onboard reset IC. The same signal is applied to pin RST of EM120/EM200
MD	Connected to the download/setup button on the EM120/EM200-EV board. The signal is connected to pin MD of EM120/EM200
DTR	Connected to pin P3(DTR) of EM120/EM200
RTS	Connected to pin P5(RTS) of EM120/EM200
TX	Connected to pin TX of EM120/EM200
RX	Connected to pin RX of EM120/EM200
CTS	Connected to pin P4(CTS) of EM120/EM200
DSR	Connected to pin P2(DSR) of EM120/EM200

Output signals that are present both on the DB9M and expansion connectors (DTR, RTS, TX) need not be switched. So, for example, the TX (output) line from the EM120/EM200 is connected to the RS232 transceiver IC and to the expansion connector. For input signals (RX, CTS, DSR) there must be a way to disconnect the RS232 transceiver IC from the EM120/EM200. Three jumpers (combined with pins RX, CTS, DSR of the expansion connector) serve this purpose.

For example, when the RX jumper is closed the RX pin of the EM120/EM200 receives a signal from the RS232 transceiver. When the jumper is opened you can use the RX pin on the expansion connector to supply a TTL RX signal from your own external board. Figure below illustrates this.



Maximum load for all CMOS-type lines (P0, P1, ... RX, TX...) is 10mA.

Device Servers and Controllers

This part of documentation describes Ethernet-to-serial Servers (for external use) supplied by Tibbo.

The following devices are currently manufactured:

- DS100 Serial Device Server
- DS203 Serial Device Server

DS100 Serial Device Server

The DS100 is a Serial Device Server for external use. Device hardware includes one 10BaseT Ethernet port, one serial port and an internal processor that "glues" network and serial sides together. Internally, the DS100 is based on the EM100 Ethernet Module.

The DS100 is supplied in two modifications:

- **DS100R** with RS232 serial port.
- **DS100B** with universal RS232/422/485 serial port.

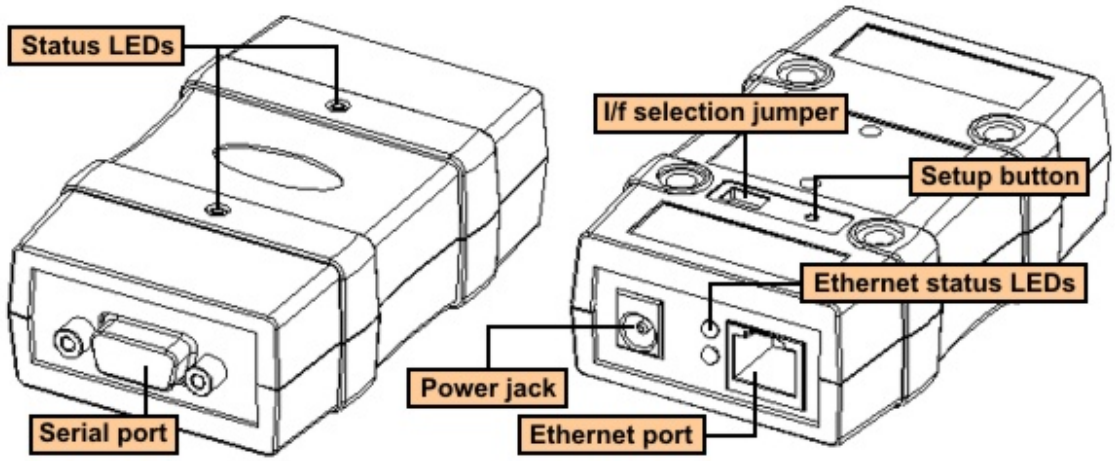
From the hardware standpoint, the DS100 can be viewed as a universal platform suitable for running a variety of network and serial communications-related applications. It is the application firmware, not the hardware that gives the DS100 most of its functionality. The firmware is currently in its 3rd generation ("Release3").

The application firmware of the DS100 can be upgraded through the device's serial port or Ethernet port. Serial upgrades are facilitated by a so-called Monitor- a fixed "service" firmware inside the DS100. The Monitor itself cannot be upgraded. Network upgrades rely on the NetLoader firmware component that, like the application firmware itself, can be upgraded through the serial port of the DS100 (using the Monitor). The DS100 is supplied with the application firmware and the NetLoader already pre-loaded.

Since most of the DS100's operation is defined by its firmware the major part of the functional description can be found in the Device Server Application Firmware Manual. This *DS100 Serial Device Server Manual* focuses on the hardware portion of the DS100.

** Network upgrades are only possible on the latest DS100-03 modification of the device (see specifications and EM100 modifications for details)*

DS100 Connectors and Controls

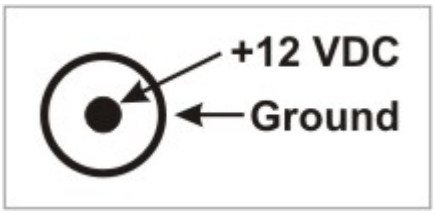


Click on the area on the picture above or one of the links provided below to learn more about the DS100:

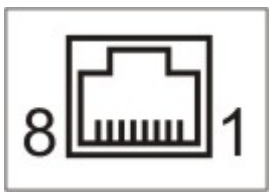
- Power Jack (input power is 12VDC, adaptor current rating must be no less than 500mA)
- Ethernet port pin assignment.
- Serial port pin assignment and interface selection.
- Status LEDs.
- Setup button.

Power Jack

Power Jack of the DS100 accepts "large" power connectors with 5.5mm diameter. Use ARP-1014, ARP-1015A, or ARP-1018A power adaptor supplied by Tibbo or similar adaptor. Nominal voltage is 12VDC and adaptor current rating should be at least 500mA. On the power jack, the ground is "on the outside", as shown on the figure below.



Ethernet Port Pin Assignment

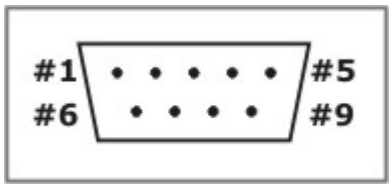


Ethernet port of the DS100 is of 10BaseT type. The DS100 is compatible with all 10BaseT Ethernet hubs and also 99% of 100BaseT hubs. This is because most 100BaseT hubs are actually 100/10 devices that auto-detect the type of device connected to each port.

Connector is of RJ45 type, pin assignment is as follows:

#1	**TX+**
#2	**TX-**
#3	**RX+**
#4	*<No connection>*
#5	*<No connection>*
#6	**RX-**
#7	*<No connection>*
#8	*<No connection>*

Serial Port Pin Assignment and i/f Selection



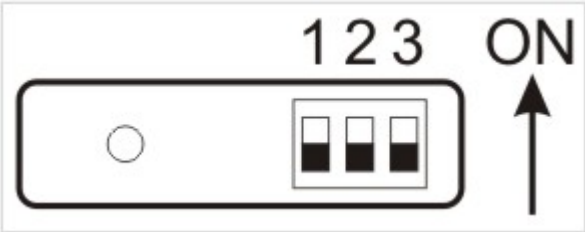
The serial port connector of the DS100 is of DB9M type. The DS100 is supplied in two models: the DS100R with RS232 port and DS100B with universal RS232/RS422/RS485 serial port. Notice, that there are no terminators (usually required at the ends of RS422 and RS485 buses) inside the DS100B. Termination circuits are present on the TB100 Terminal Block Adaptor that can be optionally supplied with the DS100B.

	DS100R	**DS100B**		
	RS232 (full-duplex op.)	**RS232 (full-duplex op.)**	**RS422 (full-duplex op.)**	**RS485 (half-duplex op.)**
#1	*<No connection>*	*<No connection>*	**RTS-** (output)	<No connection>
#2	**RX** (input)	**RX** (input)	**RX-** (input)	**RX-** (input)
#3	**TX** (output)	**TX** (output)	**TX+** (output)	**TX+** (output)
#4	*<No connection>*	**DTR** (output)	**TX-** (output)	**TX-** (output)
#5	**Ground**	**Ground**	**Ground**	**Ground**
#6	*<No connection>*	**DSR** (input)	**RX+** (input)	**RX+** (input)
#7	**RTS** (output)	**RTS** (output)	**RTS+** (output)	*<No connection>*
#8	**CTS** (input)	**CTS** (input)	**CTS+** (input)	*<No connection>*
#9	*<No connection>*	*<No connection>*	**CTS-** (input)	*<No connection>*

The difference between the RS422 and RS485 modes of the DS100B is not just in that there are no RTS+ and RTS- signals in the RS485 mode. Notice that the table above also details whether the serial port is running in the full-duplex or half-duplex mode when a particular interface is selected. When RS422 is selected the serial port is in the full-duplex mode and the TX+/TX- and RTS+/RTS- signal pairs are active at all times (i.e. output the data). When RS485 is selected the TX+/RX+ signal pair outputs the data only when the DS100 needs to send the data out through the serial port. The incoming data is ignored at this time. When the DS100 is not outputting the data the TX+/TX- signal pair is tri-stated and the DS100 is "listening" to the incoming data on the RX+/RX- signal pair. This allows you to arrange a 2-wire RS485 bus by externally connecting TX+ to the RX+ and

TX- to the RX- (this can be conveniently done by using TB100 Terminal Block Adaptor).

Interface selection for the DS100B is done through the DIP switches located on the bottom of the device, next to the setup button (DS100B only). Only switches 1 and 2 are used at the moment, switch 3 is reserved.



Interface	Switch 1	Switch 2
RS232	OFF	OFF
RS422	OFF	ON
RS485	ON	ON

If you change interface selection you need to power the DS100B off and back on again for the new selection to be recognized by the device. Also, for the interface selection to work you need to make sure that the **Serial Interface (SI) setting** of the application firmware is programmed to 2 (auto).

Status LEDs

The Green and Red status LEDs are located on the top of the DS100. The LEDs display various status information depending on what firmware is running at the moment. Follow the links below to learn more about the behaviour of these LEDs under different conditions:

- Status LED behavior in the monitor firmware
- Status LED behavior in the NetLoader
- Status LED behavior in the application firmware

There are also two Ethernet Status LEDs- Green and Red- located next to the RJ45 Ethernet connector. The Green LED is normally ON, and is temporarily turned off whenever the EM100 receives a network packet. The Red LED is normally OFF, and is turned on momentarily whenever a data collision is detected on the Ethernet.

Setup Button

The setup button is located on the bottom side of the DS100. The button can be pushed by a sharp tip of a pencil, pen, etc.

The Button is used to select an operating mode of the DS100:

- When the DS100 is powered up with the button pressed it enters a serial upgrade mode in which new application firmware file can be uploaded into the DS100 through its serial port. If the DS100 is powered up with the setup button not pressed the Device proceeds to running its current application firmware. This functionality is delivered by the Monitor firmware component of the DS100.
- When the application firmware is already running the setup button is used to make the DS100 enter the serial programming mode (hence, the name of a

button)*.

** Strictly speaking, this is a functionality that is defined by the application firmware, not the DS100 hardware.*

Specifications and DS100 modifications

There are two models of the DS100: the DS100R with RS232 serial interface and the DS100B with RS232/RS422/RS485 serial interface.

There are five different DS100R sub-models in circulation: DS100R-00, DS100R-01, DS100R-02, DS100R-03, and DS100R-04. Currently, only the DS100R-04 is being manufactured so the information on DS100R-00...DS100R-03 is provided for your reference only.

The DS100R-00, DS100R-01, and DS100R-02 devices were basically the same, with only minor changes made to the internal hardware (such as bypass capacitors on the internal PCB, etc.). We will refer to all three modifications as DS100R-02.

The DS100R-03 had extended functionality compared to the DS100R-02. There are two notable differences:

- Memory size inside the device has been increased so the routing buffers of the DS100R-03 are double the size of the buffers inside the DS100R-02 (510 bytes in each direction vs. 255 bytes in each direction).
- Ability to upgrade the application firmware through the network was added (this is facilitated by the NetLoader firmware) to the DS100R-03. The DS100R-02 cannot run the NetLoader and cannot be upgraded through the network.

The DS100R-04 is a RoHS-compliant version of the DS100R-03. The DS100R-04 and DS100R-03 are identical in every other way.

There are two DS100B sub-models in circulation: the DS100B-00 and the DS100B-01. The DS100B-01 is a RoHS-compliant version of the DS100B-00. The DS100B-01 and DS100B-00 are identical in every other way. On the functional side, both the DS100B-00 and the DS100B-01 had larger memory buffers and the possibility of firmware upgrades over the network from the very beginning of their production.

Device specifications are presented in the table below.

Parameter	**DS100R-04**	**DS100B-01**
Ethernet interface	10BaseT Ethernet	
Serial interface and I/O lines	RS232 (TX,RX,RTS,CTS)	RS232 (TX,RX,RTS,CTS,DTR,DSR) RS422 (TX+/-,RX+/-,RTS+/-,CTS+/-) RS485 (half-duplex, TX+/-,RX+/-)
Routing buffers size	510 bytes x 2 (255 bytes x 2)	
Power requirements	DC 12V, app. 100mA	
Operating temperature	-5 to +70 degrees C	
Operating relative humidity	10-90%	
Mechanical dimensions	95x57x30mm	
Carton dimensions ("bare" DS100)	130x100x65mm	

Gross weight ("bare" DS100)	170g.
Carton dimensions (DS100-KIT)	325x45x90mm
Gross weight (DS100-KIT)	950g.

Note: all specifications are for reference only. TIBBO assumes no responsibility for any errors which may appear in this Manual, and does not make any commitment to update the information contained herein.

DS203 Serial Device Server

The DS203 is a Serial Device Server for external use. Device hardware includes one Auto-MDIX* 10/100BaseT Ethernet port, one RS232 serial port and an internal processor that "glues" network and serial sides together.

From the hardware standpoint, the DS203 can be viewed as a universal platform suitable for running a variety of network and serial communications-related applications. It is the application firmware, not hardware that gives the DS203 most of its functionality. The DS203 can run two distinctively different kinds of application firmware (version restrictions apply -- see below):

"Serial-to-Ethernet" firmware, currently in its 3rd generation ("Release3"), turns the DS203 into a ready-to-work serial-to-Ethernet converter that can connect almost any kind of serial device to the Ethernet (TCP/IP) network. This firmware has fixed functionality; you adjust the way the DS203 behaves by specifying the values of programmable parameters (settings) defined in this firmware.

- TiOS (Tibbo Operating System) firmware turns the DS203 into a BASIC-programmable controller. When running TiOS, the DS203 has no pre-defined functionality -- it is your BASIC application that defines what the DS203 will do. TiOS and BASIC programming are covered in a separate Manual ("TAIKO Manual").

The application firmware of the DS203 can be upgraded through the device's serial port or Ethernet port. Serial upgrades are facilitated by a so-called Monitor- a fixed "service" firmware inside the DS203. Network upgrades rely on the application firmware itself- there is a self upgrade algorithm that will be detailed later.

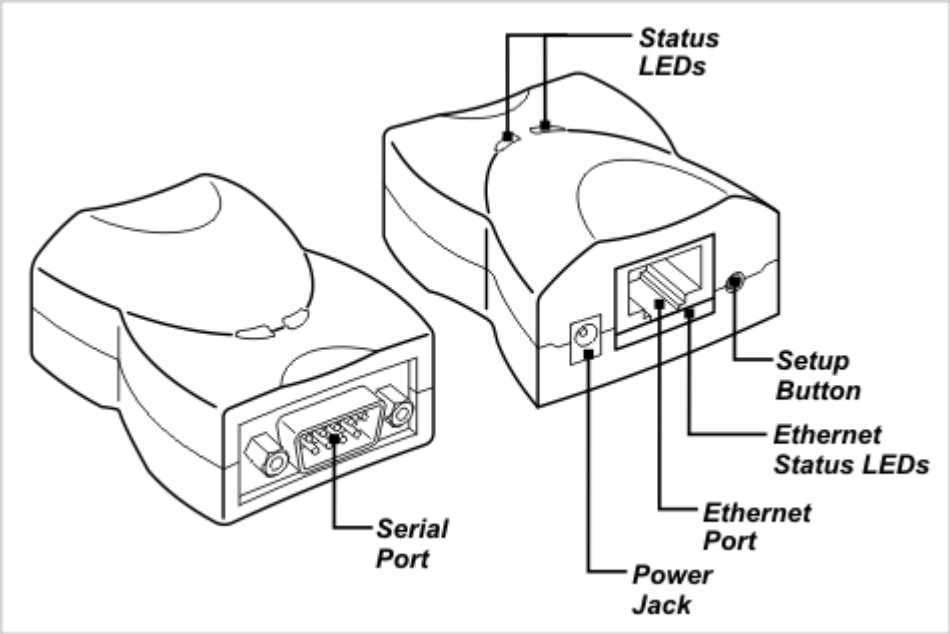
The DS203 is supplied with "serial device server" firmware pre-loaded. If you wish to receive the module with TiOS firmware, please specify option "P" device on your order. Alternatively, you can just load the TiOS firmware by yourself. Current firmware versions are posted on our website.

Firmware version restrictions

The DS203 uses newer DM9000B Ethernet controller (earlier Tibbo devices utilized an older DM9000A IC). Upgrade to the DM9000B required firmware changes which were made in **V3.70** of the "serial-to-Ethernet" firmware and **2.05.10** of the "TiOS" firmware. Earlier firmware versions will not run on this module! If you have to stick to the older firmware, please use the DS203A device instead.

** Auto-MDIX means automatic detection of "straight" and "cross" cables.*

DS203 Connectors and Controls

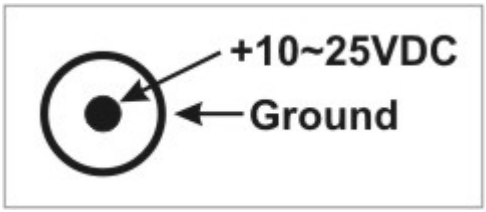


Click on one of the links provided below to learn more about the DS203:

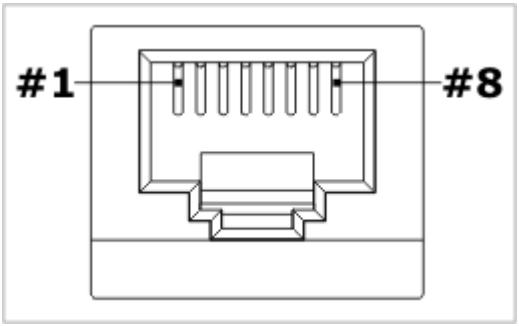
- Power Jack (input power is 10-25VDC, adaptor current rating must be no less than 500mA)
- Ethernet port pin assignment
- RS232 port pin assignment
- Status LEDs
- Setup button

Power Jack

Power Jack of the DS203 accepts "small" power connectors with 3.5mm diameter. Use ARP-P0011, ARP-P0012, or ARP-P0013 power adaptor supplied by Tibbo or similar adaptor. Acceptable power supply voltage range is 10-25VDC. Adaptor current rating should be at least 500mA. On the power jack, the ground is "on the outside", as shown on the figure below.



Ethernet Port Pin Assignment



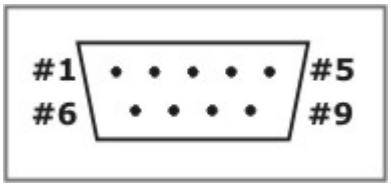
Ethernet port of the DS203 is of 10/100BaseT type.

Connector is of RJ45 type, pin assignment is as follows:

#1	**TX+**
#2	**TX-**
#3	**RX+**
#4	*<No connection>*
#5	*<No connection>*
#6	**RX-**
#7	*<No connection>*
#8	*<No connection>*

RS232 Port Pin Assignment

DB9M RS232 connector has the following pin assignment:



#1	*<No connection>*
#2	**RX** (input)
#3	**TX** (output)
#4	**DTR** (output)
#5	**Ground**
#6	**DSR** (input)
#7	**RTS** (output)
#8	**CTS** (input)
#9	*<No connection>*

Status LEDs

The DS203 has two status LEDs -- green and red. Status LEDs display various status information depending on what firmware is running at the moment. Follow the links below to learn more about the behaviour of these LEDs under different conditions:

- Status LED behavior in the monitor firmware
- Status LED behavior in the application firmware

There are also two Ethernet Status LEDs- Green and Yellow- located under the Ethernet jack:

- **Link/Data LED** (green) is turned on when "live" Ethernet cable is plugged into the DS203. The LED is temporarily switched off whenever an Ethernet packet is received.
- **100BaseT LED** (yellow) is turned on when the DS203 links with the hub at 100Mb. The LED is off when the link is established at 10Mb.

Setup Button

The setup button is located next to the Ethernet port of the DS203.

The Button is used to select an operating mode of the DS203:

- When the DS203 is powered up with the button pressed it enters a serial upgrade mode in which new application firmware file can be uploaded into the DS203 through its serial port. If the DS203 is powered up with the setup button not pressed the Device proceeds to running its current application firmware. This functionality is delivered by the Monitor firmware component of the DS203.
- When the application firmware is already running the setup button is used to make the DS203 enter the serial programming mode (hence, the name of a button)*.

** Strictly speaking, this is a functionality that is defined by the application firmware, not the DS203 hardware.*

Specifications and DS203 Modifications

The DS203 has a single sub-model circulation- "-00". Device numbering scheme is as follows:

Specifications

Ethernet interface	10/100BaseT Ethernet, Auto-MDIX
Serial interface and I/O lines	RS232; TX, RX, RTS, CTS, DTR, DSR*; baudrates up to 115'200bps; none/even/odd/mark/space parity and 7/8 bits/character; optional flow control*
Power requirements	DC 10-25V, use adaptor with current rating of at least 500mA
Operating temperature	-5 to +70 degrees C
Operating relative humidity	10-90%
Mechanical dimensions	60x47x30mm
Carton dimensions ("bare" DS203)	125x95x52mm
Gross weight ("bare" DS203)	110g
Carton dimensions (DS203-KIT)	325x45x90mm
Gross weight (DS203-KIT)	950g

** Implemented in (supported through) firmware.*

Note: all specifications are for reference only. TIBBO assumes no responsibility for any errors which may appear in this Manual, and does not make any commitment to update the information contained herein.

Kits and Accessories

Tibbo supplies the following Accessories:

- WAS-P0004(B) DS-to-device serial cable (replaces now obsolete WAS-1404)
- WAS-P0005(B) DS-to-PC serial cable (replaces now obsolete WAS-1455)
- WAS-1499 "straight" Ethernet cable (DS-to-hub cable)
- WAS-1498 "crossover" Ethernet cable (DS-to-device cable)
- TB100 Terminal Block Adaptor
- 12VDC/500mA Power Adaptors
- DMK100 DIN Rail/Wall Mounting Kit

Glossary definition: WAS stands for Wire Assembly. It is the prefix used for all Tibbo cables.

The following Starter Kits are available:

- **DS100R-SK Starter Kit.** The Kit includes all necessary parts for evaluation of the DS100R Serial Device Server:
  - DS100R Serial Device Server;
  - WAS-P0004(B) DS-to-device serial cable;
  - WAS-P0005(B) DS-to-PC serial cable;
  - WAS-1499 "straight" Ethernet cable;
  - WAS-1498 "crossover" Ethernet cable;

- 12VDC/500mA Power Adaptor.

- **DS203-SK Starter Kit.** The Kit includes all necessary parts for evaluation of the DS203 Serial Device Server:
  - DS203 Serial Device Server;
  - WAS-P0004(B) DS-to-device serial cable;
  - WAS-P0005(B) DS-to-PC serial cable;
  - WAS-1499 "straight" Ethernet cable;
  - WAS-1498 "crossover" Ethernet cable;
  - 12VDC/500mA Power Adaptor.

- **DS100B-SK Starter Kit.** The Kit includes all necessary parts for evaluation of the DS100B Serial Device Server:
  - DS100B Serial Device Server;
  - WAS-P0004(B) DS-to-device serial cable;
  - WAS-P0005(B) DS-to-PC serial cable;
  - WAS-1499 "straight" Ethernet cable;
  - WAS-1498 "crossover" Ethernet cable;
  - 12VDC/500mA Power Adaptor;
- TB100 Terminal Block Adaptor.

EM200-SK Starter Kit. The Kit includes all necessary parts for evaluation of the EM200 Ethernet-to-serial Module:

- EM120/EM200-EV Evaluation Board with one EM200 Module (installed on a socket);
- WAS-P0004(B) DS-to-device serial cable;
- WAS-P0005(B) DS-to-PC serial cable;
- WAS-1499 "straight" Ethernet cable;
- WAS-1498 "crossover" Ethernet cable;
- 12VDC/500mA Power Adaptor.

- **EM120-SK Starter Kit.** The Kit includes all necessary parts for evaluation of the EM120 Ethernet-to-serial Module:
  - EM120/EM200-EV Evaluation Board with one EM120 Module (installed on a socket);
  - WAS-P0004(B) DS-to-device serial cable;
  - WAS-P0005(B) DS-to-PC serial cable;
  - WAS-1499 "straight" Ethernet cable;
  - WAS-1498 "crossover" Ethernet cable;
  - 12VDC/500mA Power Adaptor.

- **EM100-SK Starter Kit.** The Kit includes all necessary parts for evaluation of the EM100 Ethernet-to-serial Module:

- EM100-EV Evaluation Board with one EM100 Module(installed on a socket);
- WAS-P0004(B) DS-to-device serial cable;
- WAS-P0005(B) DS-to-PC serial cable;
- WAS-1499 "straight" Ethernet cable;
- WAS-1498 "crossover" Ethernet cable;
- 12VDC/500mA Power Adaptor.

WAS-P0004(B) DS-to-Device Serial Cable

WAS-P0004(B) is a female-male serial cable that can be used to connect Tibbo Device Server or Board to the serial port of your device.

DB9M (Male)	DB9F (Female)
#2	#2
#3	#3
#4	#4
#5	#5
#6	#6
#7	#7
#8	#8

The cable is of **blue color**, approximately 1.5m long.

WAS-1404 DS-to-Device Serial Cable

This cable is now obsolete and superseded by WAS-P004(B).

WAS-1404 is a female-male serial cable that can be used to connect Tibbo Device Server or Evaluation Board to the serial port of your device.

DB9M (Male)	DB9F (Female)
#2	#2
#3	#3
#5	#5
#7	#7
#8	#8

The cable is of black color, approximately 1.5m long. Notice, that the cable doesn't have DTR/DSR lines!

WAS-P0005(B) DS-to-PC Serial Cable

WAS-P0005(B) is a female-female serial cable that can be used to connect Tibbo Device Server or Evaluation Board to the COM port of your PC.

DB9F (Female)	DB9F (Female)
#2	#3
#3	#2

#4	#6
#5	#5
#6	#4
#7	#8
#8	#7

The cable is of **green color**, approximately 1.5m long.

WAS-1455 DS-to-PC Serial Cable

This cable is now obsolete and superseded by WAS-P005(B).

WAS-1455 is a female-female cable that can be used to connect Tibbo Device Server or Evaluation Board to the COM port of your PC.

DB9F (Female)	DB9F (Female)
#2	#3
#3	#2
#5	#5
#7	#8
#8	#7

The cable is of white color, approximately 1.5m long. Notice, that the cable doesn't have DTR/DSR lines!

WAS-1499 'Straight' Ethernet Cable

WAS-1499 can be used to connect Tibbo Device Server or Evaluation Board to an Ethernet hub.

Side A	Side B
#1 (pair 1)	#1
#2 (pair 1)	#2
#3 (pair 2)	#3
#6 (pair 2)	#6

The cable is of blue color, approximately 1.5m long.

WAS-1498 'Crossover' Ethernet Cable

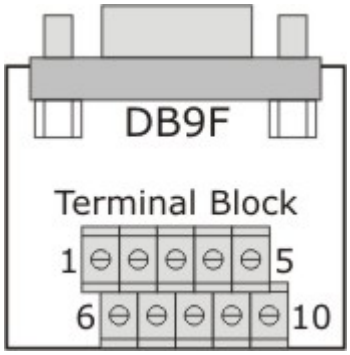
WAS-1498 can be used to connect Tibbo Device Server or Evaluation Board directly to some other Ethernet device (i.e. Ethernet port of the PC). This is a so called "crossover" cable that can interconnect two Ethernet devices without a hub.

Side A	Side B
#1 (pair 1)	#3
#2 (pair 1)	#6
#3 (pair 2)	#1
#6 (pair 2)	#2

The cable is of green color, approximately 1.5m long.

TB100 Terminal Block Adaptor

The TB100 Terminal Block Adaptor attaches to the DB9M serial port connector. The TB100 provides a convenient way of attaching wires in RS422 and RS485 systems. The wires are inserted into the terminal contacts and the terminals are then tightened using a screwdriver.



The following table details terminal block contact functions in case the terminal block is connected to the DS100:

	DS100R	DS100B		
	RS232 (full-duplex op.)	**RS232 (full-duplex op.)**	**RS422 (full-duplex op.)**	**RS485 (half-duplex op.)**
#2	*<No connection>*	*<No connection>*	**RTS-** (output)	<No connection>
#7	**RX** (input)	**RX** (input)	**RX-** (input)	**RX-** (input)
#8	**TX** (output)	**TX** (output)	**TX+** (output)	**TX+** (output)
#9	*<No connection>*	**DTR** (output)	**TX-** (output)	**TX-** (output)
#10	**Ground**	**Ground**	**Ground**	**Ground**
#6	*<No connection>*	**DSR** (input)	**RX+** (input)	**RX+** (input)
#1	**RTS** (output)	**RTS** (output)	**RTS+** (output)	*<No connection>*
#3	**CTS** (input)	**CTS** (input)	**CTS+** (input)	*<No connection>*
#4	*<No connection>*	*<No connection>*	**CTS-** (input)	*<No connection>*

As explained in serial port pin assignment the DS100B supports half-duplex RS485 communications, but the RX and TX line pairs on the DB9M connector of the DS100B are separated. In order to arrange a half-duplex two-wire RS485 bus you need to externally connect RX+ to TX+ and RX- to TX-. On the TB100 this is conveniently done by closing (putting to ON position) two switches- SW1 and SW2- located on the back of the TB100.

Additionally, the TB100 provides termination circuits typically needed at the end of the RS422 or RS485 bus. There are four identical terminators that can be switched on individually using switches located on the back of the TB100. The following table details which line pairs the terminators can be connected to:

SW3	**CTS+/CTS-**
SW4	**RTS+/RTS-**
SW5	**RX+/RX-**
SW6	**TX+/TX-**

Schematic diagram for one of the terminators is shown on figure below.

Notice, that if you are using RS485 half-duplex mode (SW1 and SW2 are closed) and you want to terminate the RS485 bus, then you only need to close either SW5 or SW6. Having both switches closed will effectively add two termination circuits to the bus!

12VDC Power Adaptors

The following adaptor models are now offered:

Products	Specifications	US	Europe	UK
DS100, DS100B, EM100-EV, EM120/200-EV	12VDC/0.5A, non-switching, "large" connector (5.5mm)	ARP-1014	ARP-1015A	ARP-1018A
DS203*	12VDC/0.5A, switching, "small" connector (3.5mm)	APR-P0011	APR-P0012	APR-P0013

**these devices were previously supplied with ARP-P0005, ARP-P0006, and ARP-P0007.*

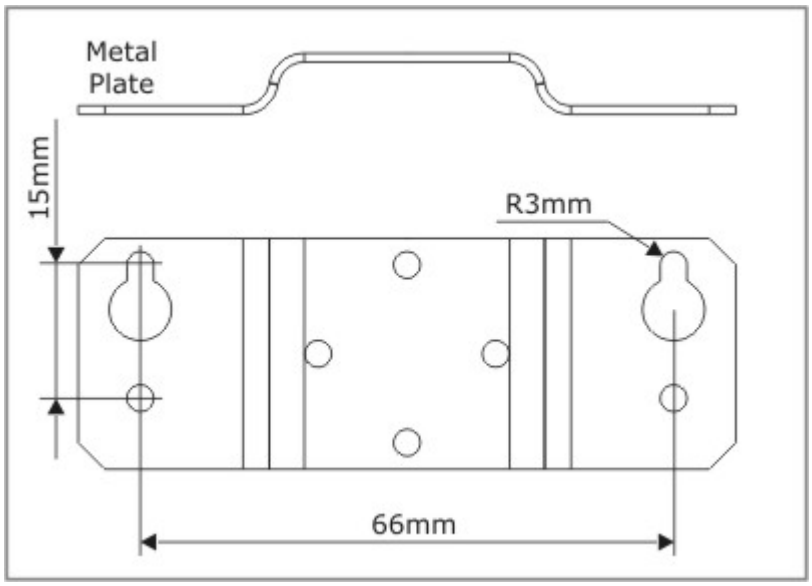
DMK100 DIN Rail/ Wall Mounting Kit

DMK100 DIN Rail/Wall Mounting Kit is used for wall or DIN-rail mounting of DS100 and DS203 Serial Device Servers. The kit includes:

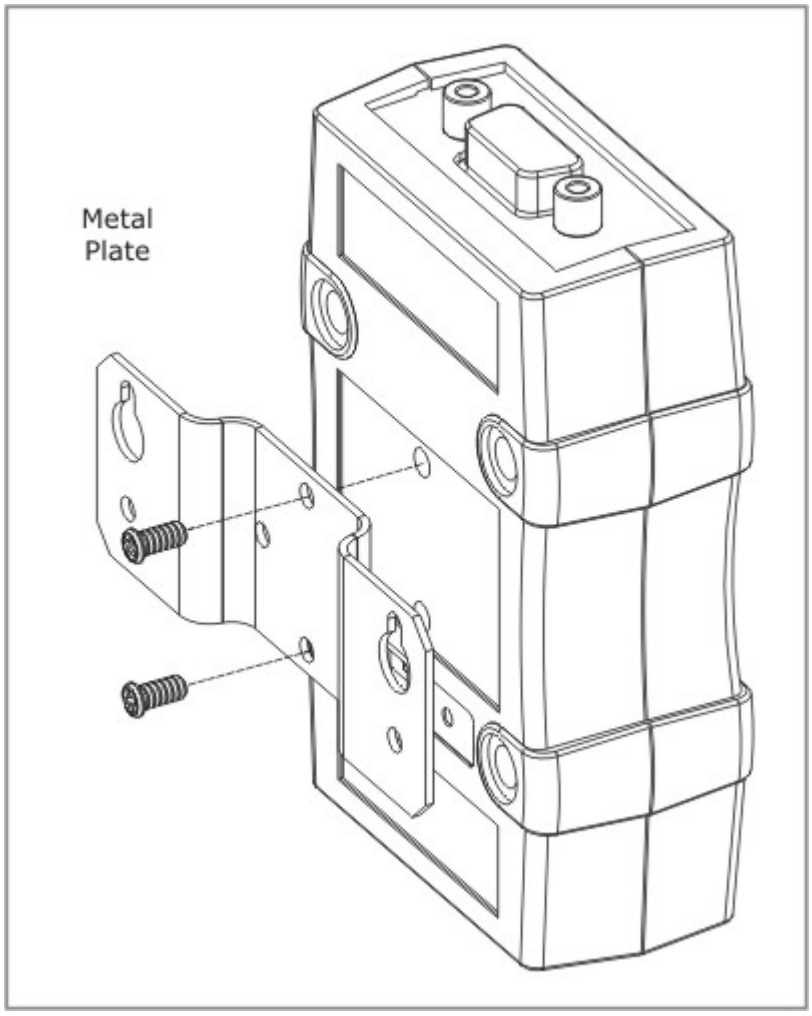
- Metal Plate x1
- DIN Rail Mounting Bracket x2
- Screw x6

Wall installation of the DS100/DS202

The Metal Plate has two pairs of mounting holes- one for "permanent" installation and another one for "easy removal".

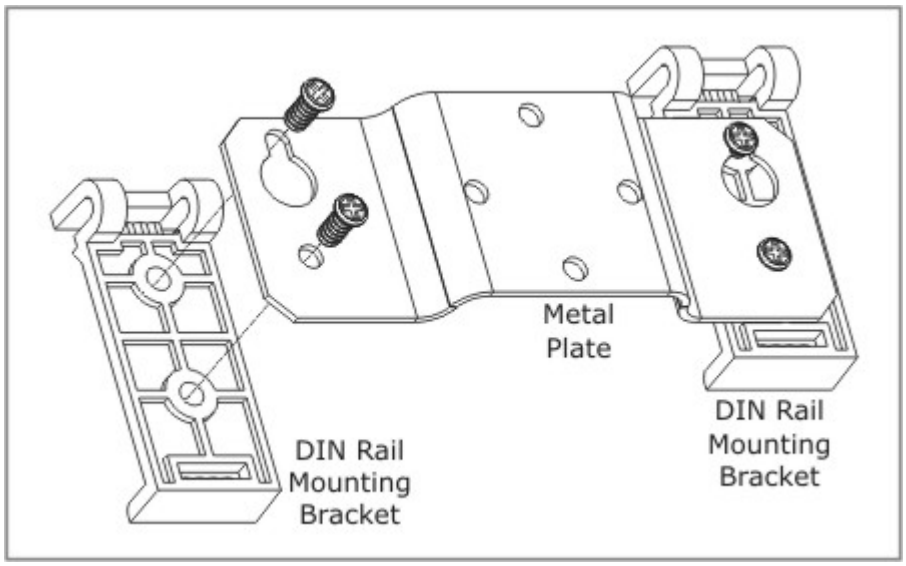


Attach the Metal plate to the DS100/DS202 as shown below.

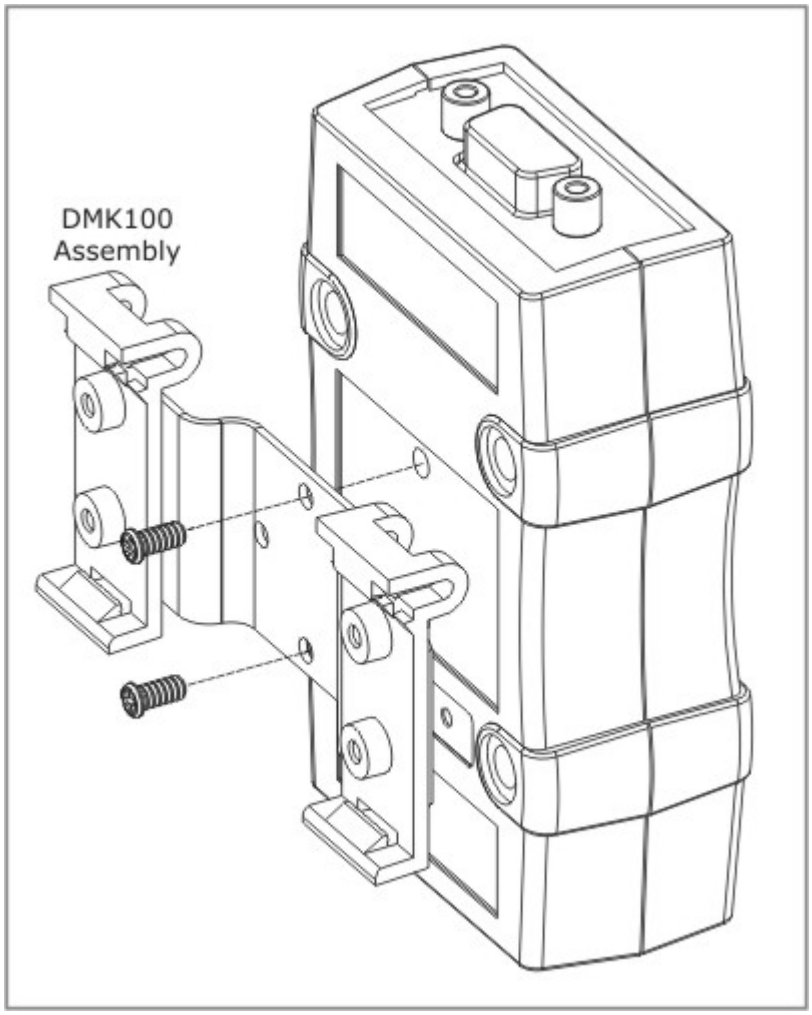


DIN Rail installation of the DS100/DS203

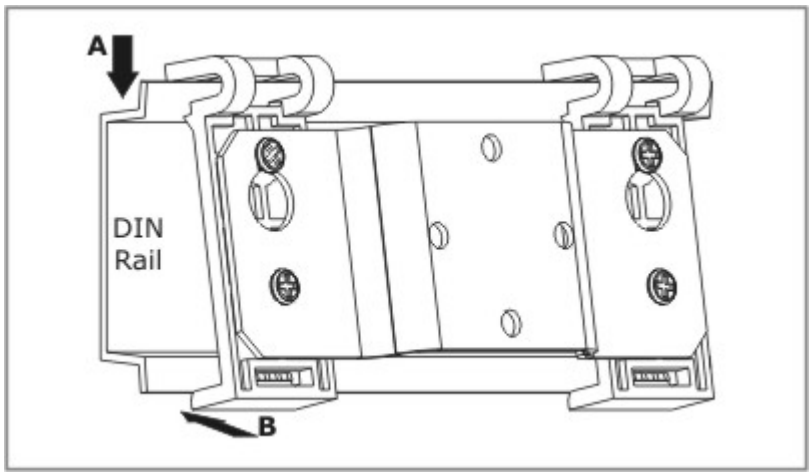
First, assemble the parts as shown on Figure below.



Next, attach the Metal Plate to the DS100/DS203:



Latch the DS100/DS203 onto the DIN Rail as shown below. **IMPORTANT!** Be careful when removing the DS100/DS203 from the DIN rail. Applying too much force may damage the Mounting Brackets or render them too loose.



Firmware Manuals

This part of the documentation describes all firmware components related to Tibbo Device Servers.

Three firmware components are currently available:

- **Monitor** is a "fixed" firmware that is responsible for starting the application firmware and firmware upgrades through the serial port of the Device Server.
- **Device Server Application Firmware** (currently in its **V3.0x/3.5x**) is what gives Tibbo Device Servers their functionality. The firmware can be uploaded into the Device Server through its serial port or through the network.
- **NetLoader** is a firmware component that facilitates application firmware upgrades through the network in the following devices: EM100-03, DS100R-03, DS100B-00. Like the application firmware itself, the NetLoader can be uploaded into the Device Server through the serial port (but not through the network- the NetLoader cannot network-upgrade itself!).

All EM100-03, DS100R-03, DS100B-00 devices come from the factory with the NetLoader already installed. Certain earlier devices- EM100-00/ -01/ -02, DS100-00/ -01/ -02- cannot be upgraded through the network at all. All other devices have "self-upgrading" firmware that doesn't require the NetLoader.

Monitor (for Serial Firmware Upgrades)

The Monitor is a fixed firmware component inside Tibbo Device Server (DS) that is responsible for booting up the DS, verifying and starting application firmware, and also performing application firmware upgrades via the serial port of the DS.

Status LED Signals

Tibbo Device Servers ("DS") feature two status LEDs*- Status Red (SR) and Status Green (SG)- that display various states of device operation. DS states are indicated by way of playing "LED patterns". Patterns in this topic are represented graphically in the following manner:

The pattern above means that both green and red status LEDs blink together three times. The following pattern means that red LED makes one long blink followed by two short ones:

When the Monitor is running status LEDs display a number of conditions:

 ...

Fast-blinking pattern means that neither application firmware, nor the NetLoader can be found in the FLASH memory of the DS. The way out of this situation is to upload application firmware and the NetLoader into the device via its serial port (see serial upgrade mode, also upgrading DS firmware).

Slow-blinking Green Status LED means that the serial upgrade was completed successfully.

Slow-blinking Red Status LED means that there was a timeout while waiting for the XMODEM data. If this happens right in the beginning of the serial upgrade then most probably this is caused by incorrect serial settings on the PC side, incorrect serial cable wiring, or incorrect XMODEM start procedure- XMODEM must be started on the PC first, and only then the DS is switched on (with he Setup button pressed).

Communications error. This pattern means that an error was detected on the protocol level in XMODEM communications. Most often this means that incorrect communications parameters are set on the PC side.

Firmware file is too big. This pattern means that the file you are trying to upload into the DS is too big. Check if you have selected a correct file.

FLASH failure. This pattern means that internal FLASH memory of the DS is malfunctioning.

DS Startup Procedure

The following describes what happens when the Device Server is powered up:

- After the powerup the Monitor starts running.
- First, the Monitor verifies whether the Setup button is pressed (on EM100, EM120, EM200, EM203(A)- whether the MD line is low). If yes then the Monitor goes into the firmware upgrade mode and is ready to receive new application firmware file through the serial port of the DS (see serial upgrade mode). If the DS has several serial ports, then the serial port #0 is always selected for firmware upgrade.
- If the Setup button is not pressed the Monitor verifies the presence and validity of the application firmware. If the firmware checks OK the control is passed to it and the DS starts running its application.
- If the application firmware is not present the following happens depending on the DS model:

For EM100-03, DS100R-03, DS100B-00 devices:	For all other devices:
• the Monitor verifies the presence and validity of the NetLoader. If the NetLoader checks out OK the control is passed to it and the DS starts waiting for the application firmware upload over the network. • If the NetLoader is not present or invalid the Monitor displays the "no firmware loaded" pattern (shown below) and halts.	The Monitor displays the "no firmware loaded" pattern (shown below) and halts.

This is a "fast-blinking" pattern that means that the DS doesn't have any firmware to run. Click here to see all available patterns.

Serial Upgrade Mode

In the serial upgrade mode the application firmware file can be uploaded into the Device Server through the serial port. The upload mechanism uses a simple and popular XMODEM protocol. Communications parameters are 38400-8-N-1.

The XMODEM upload is initiated by the Device Server side. When the DS boots up with the Setup button pressed (on EM100, EM120, EM200, EM203(A)- with the MD line pulled low) the Monitor enters the serial upgrade mode and sends the first character in the XMODEM exchange (SOH character, ASCII code 01). When the "sending side" receives this character it starts uploading the file into the DS.

Serial firmware upgrades can be performed from any communications software that supports XMODEM protocol, such as a *HyperTerminal* (supplied with *Windows* OS). Tibbo's own DS Manager (part of the Device Server Toolkit) also supports serial upgrades- see upgrading DS firmware.

A number of errors can be displayed by the Monitor while in the serial upgrade mode. These errors are displayed in the form of "patterns" that are "played" on the Red and Green Status LEDs of the DS. See status LED signals for details.

Device Server Application Firmware (V3.34/V3.66)

Application firmware is what turns Tibbo hardware into a fully-functional Device Server (DS). Tibbo Device Servers usually ship from the factory with the latest *officially released* application firmware pre-installed (unless an alternative version is requested by the Customer).

This Manual describes application firmware **V3.34/3.66 or older**. These two firmware versions are an upgrade to the previously available firmware V3.14/3.51, which is used as a "baseline" firmware for this Documentation. All changes that were made after V3.14/3.51 are clearly marked throughout this Documentation (like this: **[V3.24/3.54]**).

- V3.31 is a part of Release3 branch of application firmware. This firmware runs on first-generation Tibbo Device Servers. 3.1x is currently reaching its maturity so you won't see a lot of new features released within 3.1x branch in the future. V3.1x is available in "w", "s", and "sn" builds. Table below details which build should be used for which model of Tibbo Device Servers.
- V3.62 is a part of Release3.5 branch of application firmware. This firmware runs on second-generation Tibbo Device Servers. V3.5x is available in "r" and "d" builds. Table below details which build should be used for which models of Tibbo Device Servers.
- Certain new devices using the DM9000B Ethernet controller must be used with firmware **V3.70** or older. Loading earlier firmware into these device will make them inaccessible through the network.

As active firmware development continues, Release3.5 is gradually gaining numerous and powerful advantages over Release3. New features of Release3.5 that are not supported by earlier firmware versions are clearly marked throughout this Documentation (like this: **[V3.54]**).

Table below details firmware version/build compatibility:

First-generation devices (work with firmware V3.2*x* firmware)	
EM100-00, -01, -02; DS100R-00, -01, -02	**EM3.28W** (serial upgrades only). Tibbo no longer supports this hardware and V3.28 is the last version that will ever be available for these devices.
EM100-03; DS100R-03; DS100B*	**EM3.34S** (serial/network upgrades) **EM3.34SN** (serial upgrades)
Second-generation devices (work with firmware V3.5*x* firmware)	
EM120	**EM3.66R** (serial/network upgrades)
EM200; EM203A; DS203A	**EM3.66D** (serial/network upgrades)
EM203; DS203	**EM3.70D** (serial/network upgrades)

** These devices support network upgrades through a separate firmware component called the NetLoader. This component can only be uploaded into the DS through the serial port. "SN" build for serial upgrades contains application firmware **and** the NetLoader. "N" build for network upgrades only contains application firmware ("SN" build cannot be used for network upgrades because the NetLoader cannot upgrade itself).*

Status LED Signals

Tibbo Device Servers ("DS") feature two status LEDs*- Status Red (SR) and Status Green (SG)- that display various states of device operation. DS states are indicated by way of playing "LED patterns". Patterns in this topic are represented graphically in the following manner:

The pattern above means that both green and red status LEDs blink together three times. The following pattern means that red LED makes one long blink followed by two short ones:

Status LEDs of the DS display a number of conditions, some of which can exist at the same time. Because status LEDs can reflect only one condition at any given time there is a certain hierarchy that defines which conditions are more important and displayed in favor of other conditions.

Listed below are all patterns generated by this firmware**. Patterns are arranged in order of decreasing priority. For example, serial programming mode pattern is listed above the error mode pattern which means that if the DS is running in the serial programming mode and error mode at the same time it is the serial programming mode pattern that will be played by the status LEDs.

Most LED patterns correspond to states of various flags contained in response to the **Echo (X) command**. Where this is true a short note was added to reflect this fact (like this: *s flag*='S').

Patterns that are played once in response to a certain event:

Powerup pattern. This pattern is played once when the DS is switched on.

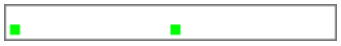
Buzz pattern. Both LEDs blink fast- this pattern is played when the DS receives the **Buzz (B) command**. This is used to identify a particular DS.

Patterns that are played repeatedly until replaced by another pattern:

Serial programming mode (*s flag*= 'S').

Error mode (*e flag*= 'E').

Ethernet port failure (*e flag*= 'N'). Indicates that the Ethernet port hardware is malfunctioning and network communications with the DS is not possible.



IP-address not obtained (i flag= '*'). Occurs at startup when **DHCP (DH) setting** is 1 (enabled) and the DS has not yet obtained its IP-address from the DHCP server.
[V3.54+]: PPPoE link is being established (details- *p flag*). Occurs at startup when **PPPoE**

Mode (PP) setting is 2 (on powerup).

[V3.54+]: PPPoE login failed (*p flag*= 'D'). Occurs at startup and means that either PPPoE login name and password (defined by **PPPoE Login Name (PL)** and **PPPoE Login Password (PD)** settings) are incorrect or PAP authentication protocol used by the DS is not supported by Access Concentrator.

Data connection is closed (*c flag*= '*'). This pattern means that no data connection (TCP or UDP) with any network host is currently established so the DS is idle.

Sending ARP (*c flag*= 'A'). Displayed when the DS is sending ARP requests to find out MAC-address of the destination network host with which the DS is about to establish a connection.
[V3.54+]: PPPoE link is being established (*p flag*= 'B'). This happens when **PPPoE Mode (PP) setting** is 1 (on connection) and the DS needs to create a PPPoE link in order to connect to remote network host.

TCP connection is being opened (*c flag*= 'O'). Indicates that TCP connection (either incoming or outgoing) is being established (i.e. SYN-SYN-ACK exchange is in progress).

TCP connection reset (rejected) by the network host (*c flag*= 'R'). Means that the TCP connection has been reset (using RST packet) by the network host to which the DS has tried to connect.

Link Server login in progress (*c flag*= 'L'). Means that the DS has already established TCP connection to the Link Server and is now attempting to login.

Link Server login failed (*c flag*= 'F'). Means that data connection to the Link Server could be established but the server has rejected this DS (because the data in the **Owner Name (ON)**, **Device Name (DN)**, or **Password (PW)**setting is incorrect or for some other reason).

Data connection is established or being closed (*c flag*= 'U' or 'C'). Means that data UDP "connection"or TCP connection is currently established or that TCP connection is being closed (i.e. FIN-ACK-FIN-ACK exchange is in progress).

When the data connection is established the following patterns are additionally displayed:

Data is being routed, no overruns detected (*E*

flag= '*' and *S flag*= '*'). This pattern is played when the data connection is established and the data is being routed through the DS;

Buffer overrun, no data routing (*E flag*= 'E' or *S flag*= 'S'). This pattern is displayed when the data connection is established and the routing buffer overrun has been detected (within the present data connection);

Buffer overrun + data routing. Data routing and overrun can be displayed at the same time.

** For modules such as EM100, EM120, EM200, EM203(A) there are two control lines to connect external status LEDs.*

*** Patterns related to the NetLoader and the Monitor are not shown here!*

Operation

The DS has two ports- *Ethernet port* and *serial port*. Typically, the serial port of the DS is connected to a certain serial device ("attached serial device"). Ethernet port links the DS to the Ethernet (TCP/IP) network (see figure below). Potentially, any network host (including another DS) on the network can communicate with the DS and the serial device "behind" it.

The job of the DS (i.e. the firmware described in this *Manual*) is to ensure transparent communications between the network host and the attached serial device by routing the data between the Ethernet and serial ports of the DS. Routing means that the data received into the Ethernet port is sent out via the serial port and vice versa. Routing is effected through two routing buffers- one for each routing direction*. In addition to routing the serial data itself, the DS allows the network host and the attached serial device to exchange the state of RTS, CTS, DTR, and DSR lines.

DS operation is governed by *settings*, *parameters*, and *instructions*:

Settings are *permanent functioning parameters* that are stored in the non-volatile memory (EEPROM) of the DS. Once programmed, they remain intact even when the DS is powered off. Many (but not all) settings require the DS to be rebooted for the new setting value to take effect. For example, the **Baudrate (BR) setting** defines the baudrate of the serial port. When the DS boots up it sets the baudrate of its serial port according to the value of this setting.

Parameters are *temporary overrides* for settings. Parameters are not saved into the EEPROM and have immediate effect (no rebooting required). For example, there is a **Baudrate (BR) parameter** that can be used to immediately change communications speed of the serial port (and override the permanent value defined by the **Baudrate (BR) setting**).

Instructions are used to make the DS perform a certain action. For example, **Establish Connection (CE) instruction** makes the DS establish a data connection with a specified network host.

Because parameters can override settings this *Manual* will sometimes make references to *current* values. For example, **current Baudrate (BR)** will mean, quite literally, the baudrate at the moment, regardless of what caused it to be such- a permanent value obtained from the setting or a temporary overriding parameter value.

Settings and parameters are manipulated and instructions are issued by sending

commands to (and getting replies from) the Ethernet or serial port of the DS- the process that will be referred to as programming.

** The size of routing buffers is hardware-dependent and is different for different models and modifications of Tibbo Device Servers. Routing buffer size can be verified using the **Status (U) command**.*

Ethernet Port and Network Communications

Just like any other Ethernet device, the DS has a unique *MAC-address* (defined by the **MAC-address (FE) setting**) and must be assigned a valid *IP-address* (defined by the **IP-address (IP) setting**) to function properly on the network. Unique MAC-address is preset during the production. Assigning a valid IP-address is the responsibility of the user. IP-address can be assigned manually or automatically, through DHCP (must be enabled through the **DHCP (DH) setting**).

To route the data between attached serial device and the network host the DS needs to establish and maintain a *data connection*. Depending on **current Transport Protocol [setting/ parameter]** the data connections can use TCP/IP or UDP/IP.

The DS only allows for a single data connection. This is because the serial port is not a shared media and cannot be controlled by more than a single network source at any given time. This is analogous to the COM port of the PC, which can only be opened by one program at a time.

The DS is capable of both accepting incoming connections (passive opens) and establishing outgoing connections of its own (active opens). Whether passive and/or active opens are allowed is defined by the **current Routing Mode (RM) [setting/ parameter]**.

Passive opens, when allowed, are accepted from any network host as long as correct transport protocol is used (must match **current Transport Protocol [setting/ parameter]**) and connection is made to a correct *data port* (defined by the **Data Port (DP) setting**). **[V3.24/3.54+]: Current Source IP Filtering [setting/parameter]** defines whether an incoming connection will be accepted from any network host or only the host whose IP-address matches the one specified by **current Destination IP-address (DI) [setting/parameter/instruction]**.

When performing an active open the DS attempts to connect to the **current Destination IP-address (DI) [setting/parameter/instruction]** and **current Destination Data Port (DP) [setting/parameter/instruction]**. When the destination is located on a different subnet, which is derived from comparing the destination IP-address, IP-address of the DS itself, and the *netmask* (defined by the **Netmask (NM) setting**), the DS connects through *default gateway* or **[V3.54+]** *PPPoE Access Concentrator*:

- When PPPoE is not used connection is made through default gateway defined by the **Gateway IP-address (GI) setting**
- **[V3.54+]:** When PPPoE is enabled (through **PPPoE Mode (PP) setting**) connection is made through PPPoE Access Concentrator. For more information see PPPoE.

Exactly when the DS attempts to establish an outgoing connection (when such connections are allowed) depends on the selected *connection mode* (see **Connection Mode (CM) setting**). Attached serial device can optionally control and monitor connection establishment and termination through a dedicated set of instructions known as *modem commands* (serial-side parameters and instructions), or through the *DSR line* manipulation (when enabled through the **Connection**

Mode (CM) setting) and *DTR line* monitoring (see **DTR mode (DT) setting**). Data connections can also be closed automatically, on timeout (see **Connection Timeout (CT) setting**).

In addition to "normal" data connections the DS supports data connections through the Link Server.

Besides the user-definable data port there are also two identical *UDP command ports* with numbers 65535 (FFFF Hex) and 32767 (8FFF Hex). Network programming of the DS is effected by sending programming commands (as UDP datagrams) to either port. Additionally, programming commands can be sent (under certain conditions) right within the data connection itself- see inband (TCP) commands and command-phase (TCP) commands.

Programming of the DS and data routing can be performed concurrently, these processes are completely independent from each other.

The DS also supports ARP and ICMP protocols- it can be PINGed just like any other network device.

UDP Data 'Connections'

This topic details UDP data connections (**current Transport Protocol (TP) [setting/ parameter]**= 0).

The notion of data connection is native to TCP/IP since this is a connection-based protocol. UDP/IP, however, is a connection-less protocol in which all packets (UDP datagrams) are independent from each other. How, then, the term "DS data connection" applies to the UDP transport protocol?

With UDP transport protocol true data connections (in the "TCP sense" of this term) are not possible (hence, quote marks around the word "connection"). The DS, however, attempts to mimic the behavior of TCP data connection whenever possible. Follows is the detailed description of UDP "connections" and their similarities and differences from TCP connections.

Incoming "connections"*. There is no connection establishment phase in UDP so an incoming UDP "connection" is considered to be "established" when the first UDP packet is received by the DS from the network host (on the port defined by the **Data Port Number (PN) setting**). Similarity with TCP is in that after having received the packet from the network host the DS knows who to send its own UDP packets to. **[V3.24/3.54]:** When **Current Source IP Filtering [setting/ parameter]** is 1 (enabled) the DS ignores UDP datagrams that arrive from any IP-address other than the one matching **current Destination IP-address (DI) [setting/parameter/instruction]** (unless Destination IP-address is set to 255.255.255.255- see broadcast UDP communications).

Outgoing "connections".** The DS establishes outgoing UDP connection by sending a UDP datagram to the targeted destination (defined by the **current Destination IP-address (DI) [setting/parameter/instruction]** and **current Destination Data Port (DP) [setting/parameter/instruction]**). If there is a data that the DS needs to send to the network host then the DS sends the first UDP datagram with (part of) this data. If there is no immediate data that needs to be sent to the network host then the DS sends the first UDP datagram of zero length. This happens when the **Connection Mode (CM) setting** is 0 (connect immediately). The purpose of this is to let the other side know the MAC-address, IP-address, and the data port of the DS.

Data transmission and destination switchover. Once the "connection" is established the DS and the network host can exchange the data. The difference

with TCP is that if another network host sends a datagram to the DS then the DS will interpret this as a new incoming "connection"*, forget about the IP-address of the first network host and start sending its own UDP datagrams to the IP-address of the second network host. In other words, the DS will always send its own UDP datagrams to the IP-address of the "most recent sender". The only exception is the case when the **current Destination IP-address** is 255.255.255.255- see broadcast UPD communications.

The situation with the port to which the DS will address its UDP datagrams is a little bit more complex and depends on the **current Routing Mode (RM) [setting / parameter]**:

- When the **current Routing Mode** is 0 (server) the DS sends its own UPD datagrams to the port number from which the most recent incoming data UDP packet was received. In other words, the DS always switches over not only to the IP-address of the most recent sender, but also to the port number from which the most recent UDP datagram was received.
- When the **current Routing Mode** is 1 (server/client) or 2 (client) the DS always sends its UDP datagrams to the port number specified by the **current Destination Data Port (DP) [setting/parameter/instruction]**. This means that even if the DS switches over to the IP-address of the most recent sender it will still address its own UDP datagrams to the data port specified by **current Destination Data Port**.

"Connection" termination. There is no connection termination phase in UDP so DS "terminates" its UDP connections by simply "forgetting" about them and the only event that can trigger UDP "connection" termination is connection timeout (defined by the **Connection Timeout (CT) setting**).

** Assuming that incoming connections are allowed (i.e. **current Routing Mode [setting/parameter]** is either "server", or "server/client") **AND [V3.24/3.54]: Current Source IP Filtering [setting/parameter]** is 0 (disabled)*

*** Assuming that outgoing connections are allowed (i.e. **current Routing Mode [setting/parameter]** is either "client", or "server/client").*

Broadcast UDP Communications

With UDP transport protocol (**current Transport Protocol (TP) [setting/ parameter]**= 0) it is possible to send and receive broadcast UDP datagrams.

Sending broadcast UDP datagrams. When **current Destination IP-address (DI) [setting/parameter]** is 255.255.255.255 the DS sends out its own UDP datagrams as link-level broadcasts i.e. with destination MAC-address of UDP datagrams set to 255.255.255.255.255.255. The DS continues to broadcast its data even if it receives an incoming UDP datagram. In other words, the DS doesn't switch over to the most recent sender as described in UDP data "connections".

Receiving broadcast UDP datagrams. Whether or not the DS will accept data UDP datagrams sent in the broadcast mode is defined by the **Broadcast UDP (BU) setting**.

TCP Data Connections

This Section details TCP data connections (**current Transport Protocol (TP) [setting/ parameter]**= 1).

TCP protocol implementation in the DS has the following differences from the standard:

TCP reconnects. When the data TCP connection between the DS and the network host is already established and another network host (with a different IP-address) attempts to connect to the data port of the DS this connection attempt is rejected (because the DS only allows for a single data connection at a time). However, if the second connection is attempted from the same IP-address (as the IP-address with which existing TCP connection is established) then the DS will "switchover" to this new connection. This means that the DS will forget the first connection and accept the second one. Such behavior is implemented to avoid DS stalling by hanged connection. Consider this scenario: the application on the network host has a TCP connection to the DS in progress. Suddenly, the application crashes- connection is left hanging because it wasn't properly terminated. When the application is re-launched it attempts to establish a TCP connection with the DS again. If the reconnect feature wasn't implemented the DS would have considered this to be an attempt to establish a second concurrent data connection and would have rejected it. Owing to reconnect feature the DS will recognize that this new connection attempt has originated from the same IP-address and switch over to this new connection. The downside of this arrangement is that two (or more) applications communicating with the DS from the same host can interfere with each other's connections- each new connection attempt will take over the existing one.

TCP retransmissions. Standard protocol implementation defines "exponential backoff" retransmit times whereas each subsequent retransmissions is attempted after a longer and longer periods of time. Such behaviour is not very suitable for real-time systems. For any situation that requires a TCP packet to be retransmitted the DS will perform six retransmission attempts at even intervals. These intervals are defined, in 0.5 second increments, by the **Retransmission Period (RP) setting**. Default value of this setting is 6 (3 seconds). This default retransmission period works fine on most networks. Setting it to higher value may improve DS operation on networks with significant response delays, such as those including GPRS segments.

DHCP

The DS supports Dynamic Host Configuration Protocol (DHCP). DHCP is used to automatically configure the following settings: **IP-address (IP)**, **Gateway IP-address (GI)**, and **Netmask (NM)**. For the DHCP to work there must be a *DHCP server* on the network where the DS is installed.

The DHCP is enabled and disabled through the **DHCP (DH) setting**. When **DHCP (DH) setting**is at 1 (enabled) the DS uses DHCP protocol to obtain its IP-address immediately upon startup. **The DS does not start its normal operation until it receives an IP-address from the DHCP server.** Gateway IP-address and netmask configuration is optional so the DS does not *require* these two parameters to be received from the DHCP server.

"IP-address not obtained" pattern is displayed by the status LED's of the DS while the IP-address is being configured through DHCP. This condition corresponds to the value '*' of the *i* flag returned by the **Echo (X) command**.

When requesting an IP-address for itself, the DS asks for a maximum possible *lease time* (this is the period of time during which the DS will be allowed to use the IP-address). The DS memorizes the lease time actually offered by the DHCP server and applies for a *lease extension* well before the lease expires. If the lease is extended the DS continues normal operation. If the lease is not extended the DS *reboots*.

While communicating with the DHCP server, the DS supplies its *name*. The name consists of the values of **Owner Name (ON)** and **Device Name (DN)** settings joined by hyphen. For example, the name might look like this:

abccorp-dev1

Names supplied by DHCP clients are usually supplied (by the DHCP server) to a local *DNS server* (DNS and DHCP servers often work together on the same physical server PC). This typically allows you to "see" your DS as a member of your local workgroup, for example:

abccorp-dev1.workgroup1.mainserver

When using DHCP the DS detects Ethernet cable disconnects and re-requests its IP-address from the DHCP server once the cable is plugged back in. The following outcomes are possible:

- If the DHCP server confirms that the IP-address that was leased to this DS can still be used the DS continues uninterrupted operation
- If the DHCP server informs the DS that the IP-address that was leased to this DS can no longer be used the DS reboots (naturally, this aborts any data communications in progress). During the boot process the DS will attempt to interact with the DHCP server repeatedly until the IP-address of the DS is properly configured
- If there is no reply from the DHCP server the DS repeats its request. If there is still no reply the DS reboots. During the boot process the DS will attempt to interact with the DHCP server repeatedly until the IP-address of the DS is properly configured.

PPPoE [V3.54+]

ATTENTION! This topic covers firmware functionality that is only supported in firmware **V3.54+** and, hence, only available on second-generation Devices (EM120, EM200, EM203(A), DS203).

PPPoE family of protocols is mostly used by ADSL modems. Some ADSl links (modems) are of "always on" type and do not require any link establishment. Other ADSL links require link establishment procedure based on PPPoE protocol that is very similar to that of old-style POTS dial-up modems. In fact, PPPoE means "PPP over Ethernet"- it is an Ethernet adaptation of an old PPP (point-to-point protocol) used with dial-up modems. If the DS is directly connected to such an ADSL modem it will need to establish a PPPoE link in order to connect to any remote network host (i.e. network host residing on a different network segment)*.

PPPoE only affect communications with remote network hosts, i.e. hosts that reside on a different network segment (local segment boundaries are defined by the **Netmask (NM) setting**). With PPPoE enabled, all communications with remote hosts are effected through PPPoE link and not default gateway (therefore, **Gateway IP-address (GI) setting** is irrelevant in this case).

PPPoE is enabled through **PPPoE Mode (PP) setting**. PPPoE protocol includes authentication phase (login). PPPoE login name and password are defined by **PPPoE Login Name (PL)** and **PPPoE Login Password (PD)** settings. Several authentication protocols are defined for PPPoE. Currently, the DS only supports authentication through a protocol called PAP.

The **PPPoE Mode (PP) setting** offers two options that define when PPPoE link is established. When this setting is at 1 (on connection) the link is established when the DS needs to open a data connection to a remote host. The link is terminated 30 seconds after the data connection is closed. When this setting is at 2 (on powerup) the PPPoE link is established during powerup procedure. The DS then attempts to maintain this link at all times and re-establishes the link if it is broken.

With any PPPoE link both sides have to agree on the IP-addresses that will be used on each side of the link. IP-address on the DS side is requested from and supplied by the Access Concentrator. This IP-address has nothing to do with the "main" IP-address of the DS which is defined by **IP-address (IP) setting** or supplied by DHCP server when **DHCP (DH) setting**= 1 (enabled). Just like on PC, a PPPoE link becomes a "virtual" network card (interface) of the DS. As such, it has its own IP-address. This does not affect operation of the DS in any way since it is, actually, not important which IP-address is used for PPPoE link- as long as the Access Concentrator is "satisfied" with this address.

The status of PPPoE link can be obtained via **Echo (X)** and **Status (U)** commands. Status LEDs of the DS also display PPPoE-related patterns.

** PPPoE is not required if the DS is connected to the ADSL modem through a router that supports PPPoE. In this case it is the job of this router to take care of PPPoE link management*.

Link Server Support

Link Server is a separate software package that makes working with remote Device Servers more convenient. With Link Server large distributed systems consisting of multiples Device Servers can easily be built. This section does not cover Link Server operation in details. For more information see dedicated Link Server documentation.

From the Device Server's point of view the Link server offers two distinct services and there are several settings and parameters that are related to these services:

- **Link Service:**
  - **Link Service (TL) setting** and **parameter**
  - **LS Auto-registration (AR) setting**
  - **Owner Name (ON) setting**
  - **Device Name (DN) setting**
  - **Password (PW) setting**
- **Dynamic DNS (dDNS) Service:**
  - **dDNS Service Registration (DD) setting**
  - **dDNS Service IP-address (LI) setting**
  - **dDNS Service Port (LP) setting**

DS Powerup Procedure

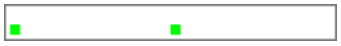
This topic details steps that the DS takes after the powerup. Shown on the left are status LED patterns that are displayed during each step of powerup procedure.

Step 1: powerup and initialization. The DS boots up and prepares for operation. At this time all

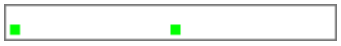
internal settings of the DS are verified and the hardware is checked.

If at least one of the settings is found to be invalid the DS enters the error mode. In this mode the DS is still accessible through the network but its functionality is limited.

Ethernet port malfunctions are also detected at this stage.



Step 2: IP-address configuration. When the **DHCP (DH) setting**is at 0 (disabled) the DS simply uses IP-address that is set in the **IP-address (IP) setting**, in which case IP-address configuration is instantaneous. When **DHCP setting** is at 1 (enabled) the DS attempts to obtain its IP-address using DHCP protocol. The DS stays on this step of powerup procedure until the IP-address is properly configured.



[V3.54+]: Step 3 (optional): PPPoE link establishment. If **PPPoE Mode (PP) setting**is at 2 (on powerup) the DS attempts to establish a PPPoE link. The DS will stay on this step of powerup procedure until the PPPoE link is successfully established.

PPPoE login failure. If PPPoE authentication fails (because the data in the **PPPoE Login Name (PL)** or **PPPoE Login Password (PD)**settings is incorrect or because Access Concentrator doesn't support PAP authentication protocol) the DS indefinitely continues attempts to establish the link.

<Several different patterns>

Step 4 (optional): registration at dDNS. If **dDNS Service Registration (DD) setting**is at 1 (enabled) the DS attempts to connect to the dDNS Service of the Link Server. Destination IP-address and port are specified by the **dDNS Service IP-address (LI)** and **dDNS Service Port (LP)** settings. At this point the DS goes through several different steps- just like for a normal outgoing TCP data connection.

dDNS login failure. dDNS login may fail (because the data in the **Owner Name (ON)**, **Device Name (DN)**, or **Password (PW)**settings is incorrect or for some other reason). When **LS Auto-registration (AR) setting** is at 1 (enabled) and the DS is informed that it is not currently registered on this Link Server it will attempt to register.

Procedure completed: idle state. At this point the DS is fully configured and operational. If **Routing Mode (RM) setting** is 1 (server/client) or 2 (client only) **AND Connection Mode (CM) setting** is 0 (immediately) the DS proceeds straight

to attempting to establish an outgoing data connection.

Data Connection Establishment Procedure

This topic details steps that the DS takes when it needs to establish an outgoing data connection (this is also known as "performing an active open") or after it accepts an incoming connection ("passive open").

Procedure for active opens. Exactly what triggers outgoing connection establishment is defined by the **Connection Mode (CM) setting**. Outgoing connections are not allowed when **current Routing Mode (RM) [setting/ parameter]** is 0 (server).

Procedure for passive opens. This procedure starts by another network host. Incoming connections are not allowed when **current Routing Mode (RM) [setting/parameter]** is 2 (client). **Source IP Filtering (SF) setting**defines who can connect to the DS. Incoming connection must match **current Transport Protocol (TP) [setting/parameter]** and **Port Number (PN)**.

Entire data connection establishment procedure is presented below. Shown on the left are status LED patterns that are displayed during each step of this procedure.

Active opens start here

Step 1: determining location of destination host. The DS first compares its own **IP-address (IP)** with **current Destination IP-address (DI) [setting/parameter]** and **Netmask (NM)**to determine whether the destination host is located on the same or different network segment. Depending on the outcome of this comparison and the data in **[V3.54+] PPPoE Mode (PP) setting** the DS may choose step **2a**, **2b**, **2c**, or **2d**.

Step 2a: destination IP is on the same network segment. The DS sends ARP request to this IP-address in order to "resolve" it into the MAC-address. This is done *each time* connection needs to be opened- the DS does not maintain "ARP cache".

Step 2b: destination is "remote", [V3.54+] PPPoE Mode= 0 (disabled). The DS sends ARP request to the default gateway specified by the **Gateway IP-address (GI) setting**. This is done *each time* connection needs to be opened- the DS does not maintain "ARP cache".

[V3.54+] Step 2c: destination is "remote", PPPoE Mode= 1 (on connection). The DS attempts to establish a PPPoE link.

[V3.54+] Step 2d: destination is "remote", PPPoE Mode= 2 (on powerup). The DS doesn't have to do anything on this step because PPPoE link has already been established during powerup procedure.

PPPoE link failure. PPPoE link establishment can fail (because the data in the **PPPoE Login Name (PL)** or **PPPoE Login Password (PD)**setting is incorrect or because Access Concentrator doesn't support PAP authentication protocol).

Step 3: establishing data connection. The DS opens a TCP connection or UDP "connection" to the remote host as defined by **current Transport Protocol (TP) [setting/parameter]** and **Destination Port (DP) setting**. This done, procedure continues- see **common portion of connection establishment**.

Connection rejection. In case of TCP transport protocol connection can be rejected by the destination network host which will be reflected by a corresponding LED pattern.

Passive opens start here

Step 1: accepting data connection. The DS accepts the data connection (provided that this connection can be accepted).

Common portion of connection establishment

Step 1 (optional): performing Link Server login. If **current Transport Protocol (TP) [setting/parameter]** is at 1 (TCP) **AND Link Server Login (TL) setting** is at 1 (enabled) the DS assumes that the destination host with which connection has just been established is a Link Server. The DS then initiates Link Service login sequence.

Link Server login failure. Link Server login may fail (because the data in the **Owner Name (ON)**, **Device Name (DN)**, or **Password (PW)**setting is incorrect or for some other reason). This will be reflected by a corresponding LED pattern. When **LS Auto-registration (AR) setting** is at 1 (enabled) and the DS is informed that it is not currently registered on this Link Server it will attempt to register.

Step 2 (optional): command portion of TCP connection. If **current Transport Protocol (TP) [setting/parameter]** is at 1 (TCP) **AND Data Login (DL) setting** is at 1 (enabled) the DS enters command-phase programming mode. Network host doesn't have access to the serial port of the DS yet but it can already send network commands to the DS. This step ends with network host using **Logout (O) command**.

Connection established. Network host has access to the serial port of the DS.

Serial Port and Serial Communications

Serial port of the DS has two modes of operation:

Data routing mode. Incoming serial data is routed to the Ethernet port and Ethernet data is routed to the serial port. It is in this mode that the DS performs its routing function. After the powerup the DS is running in the data routing mode.

Serial programming mode. In this mode the serial port is used for serial programming and all data received into the serial port is interpreted as programming commands.

Data connection with the network host can still be established and maintained while in the serial programming mode but the data received from the network host will be discarded and the data received into the serial port will be interpreted as commands. Therefore, data exchange with the network host and serial programming cannot be concurrent. DS can only perform one of the two at any given time. This is different from the network programming that can be performed concurrently with the data routing.

Operation of the serial port in the data routing mode is governed by several settings (see below), some of which (baudrate, parity, etc.) have corresponding parameters. These parameters are delivered to the DS via the *network* **Parameter (P) command** and are commonly known as on-the-fly commands (or, more officially, network-side parameters). On-the-fly commands provide a way for the network host to immediately change communications mode of the serial port without rebooting the DS. There are also network-side instructions (**Set I/O Pin Status (Sx)** and **Get I/O Pin Status (Gx)**) that are used to set and sense the state of RTS, CTS, DTR, DSR, and also additional P0 and P1 lines* of the DS.

Serial port of the DS has the following capabilities:

- Half-duplex or full-duplex operation as defined by the **Serial Interface (SI) setting**;
- Baudrates of up to 115200 bps as defined by the **current Baudrate (BR) [setting/ parameter]**;
- 7 or 8 bits/byte as defined by the **current Bits Per Byte (BB) [setting/ parameter]**;
- Several parity options as defined by the **current Parity (PR) [setting/ parameter]****;
- There are also several options related to the RTS, CTS, DTR, and DSR lines:
  - In full-duplex communications the RTS and CTS lines can be programmed to serve as flow control lines between the DS and attached serial device- this is defined by the **current Flow Control (FC) [setting/ parameter]**;
  - In half-duplex serial communications the RTS line serves as a direction control line (see **Serial Interface (SI) setting** for details). This is why this line is called "RTS/DIR";
  - The CTS line optionally serves as an additional function of selecting between full- and half-duplex communications (see "auto" selection of the **Serial Interface (SI) setting**).
  - DTR line can be programmed to reflect current data connection status- see the **DTR Mode (DT) setting**;
  - DSR line can be programmed to control data connection establishment and

termination- see the **Connection Mode (CM) setting**;

- Once again, the status of RTS and DTR lines can be controlled (set) by another network host using the **Set I/O Pin Status (Sx) instruction**, while the status of CTS and DSR lines can be monitored (polled) using **Get I/O Pin Status (Gx) instruction**;
- Additionally, the DS can be programmed to notify another network host of the state changes of its RTS, CTS, DTR, DSR, P0, and P1 lines*. Which lines are monitored for changes is defined by the **current Notification Bitmask (NB) [setting/parameter]**. Notifications are delivered in the form of **Notification (J) messages** to the port defined by the **Notification Destination (ND) setting**.

** Whether or not these lines are physically implemented depends on the DS model.*

*** There is not way to set the number of stop bits directly but the second stop-bit can be emulated by setting current Parity (PR) to 3 (mark).*

**** HI and LOW states are described with respect to the serial ports of DS100R, DS100B, DS203. For EM100, EM120, EM200, EM203(A) the signaling is exactly opposite.*

Opened and Closed States of the Serial Port

Depending on the current setup and data connection state the serial port of the DS may be opened or closed. When the serial port is opened it is ready to record incoming data into the serial-to-Ethernet buffer* when in the data routing mode or accept a serial command when in the serial programming mode. When the serial port is closed it ignores all incoming serial data except the escape sequences (when enabled through the **Soft Entry (SE) setting**).

The following details when the serial port is closed or opened:

- When the serial port is in the serial programming mode it is always **opened** (i.e. ready to accept programming commands);
- When the serial port is in the data routing mode:
  - While the IP-address is being obtained from the DHCP server** the port is **closed**;
  - After the IP-address has been obtained successfully:
    - If the **current Routing Mode (RM) [setting/ parameter]** is 0 (server):
      - If the data connection is not established the port is **closed**;
      - If data connection is established the port is **opened**;
    - If the **current Routing Mode (RM)** is 1 (server/client) or 2 (client):
      - If the **Connection Mode (CM) setting** is 0 (immediately) or 1 (on data or command) the port is **opened**;
      - If the **Connection Mode (CM) setting** is 2 (on command) or 3 (on command or DSR=HIGH):
        - If the data connection is not established the port is **closed**;
        - If the data connection is established the port is **opened**.

** Not all incoming data will necessarily be recorded into the buffer- see serial-to-Ethernet data routing*

*** Happens at startup when the **DHCP (DH) setting** is 1 (enabled)*

Data Routing3

Data routing between the Ethernet and serial ports of the DS is effected through two routing buffers*, one for each routing direction. Buffers are necessary because Ethernet and serial ports operate in fundamentally different ways and at different speeds. Ethernet port receives and sends the data in *packets* (groups) while the serial port sends and receives serial data *stream* where each data byte is independent. Here is how the DS transforms Ethernet packets into the serial stream and back:

Ethernet-to-serial data routing. The DS outputs the contents of arriving Ethernet data packets byte by byte via the serial port. The DS does not check or filter the contents of the data arriving from the network host.

Serial-to-Ethernet data routing. This requires grouping arriving serial data into Ethernet packets of suitable size. Several settings define what data is accepted into the buffer and when and how this data is combined into the Ethernet packets and sent out- see serial-to-Ethernet data routing for details.

Routing buffers are initialized (their data discarded) each time the data connection is closed.

** The size of routing buffers is hardware-dependent and is different for different models and modifications of Tibbo Device Servers. Routing buffer size can be verified using the **Status (U) command**.*

Serial-to-Ethernet Data Routing

The challenge of serial-to-Ethernet data routing is to choose how to group the serial data into the Ethernet packets of reasonable size. Carrying too little data in each packet increases network load while sending packets with too much data slows down the delivery of this data to the network host (because all this data needs to be accumulated in the buffer first). A dedicated group of packet encapsulation settings controls how and when the serial data is turned into the Ethernet packets and sent out to the network host.

When in the data routing mode the serial port of the DS treats all incoming serial data as a sequence of *serial data blocks**. In many cases serial traffic to and from the attached serial device is structured in some way (i.e. using some sort of data packets). Since it is only logical to apply the same division to the outbound network packets the DS can be programmed to recognize the beginning and the end of serial data blocks. This does not mean that the DS can only work with structured serial data- absolutely random data stream can simply be thought of as one continuous serial data block.

Serial data blocks begin when the *start condition* is detected and end when the *stop condition* is detected. After the start condition is detected the DS begins recording all incoming serial data into the serial-to-Ethernet routing buffer**. Thus, the start condition is said to *open* a data block. When the stop condition is detected the DS stops recording incoming serial data into the buffer (*closes* the

serial data block) and *commits* all the data in the serial-to-Ethernet routing buffer. Committed data is the data that the DS will attempt to route to the network host at the earliest possible opportunity. Before the data is committed the DS does not attempt to route it. Therefore, the number of committed bytes in the serial-to-Ethernet buffer may be smaller than the total number of bytes.

The DS ignores all data between the serial data blocks.

Besides the start and stop conditions there is also a *break* condition. When the break condition is detected the DS commits the data in the buffer but does not close the serial data block. Break conditions provide a way to subdivide large serial data blocks.

Follows is the description of available start, stop, and break conditions:

Start conditions. It is possible to make the DS either open a new serial data block after it receives any character (past the end of the previous data block) or a specific preset character- this is defined by the **Start On Any Character (SA) setting**. When this setting is 0 (disabled) then two other settings- **Use Start Character (F1)** and **Start Character Code (S1)** are used to enable and select specific start character. New serial data block is opened only if this start character is received. All characters between the end of the previous serial data block and the start character are ignored. Start characters received after the serial data block has been opened are treated as normal data characters and do not "reopen" the serial data block.

Stop condition is defined by the stop character which is enabled and selected via two separate settings- **Use Stop Character (U1)** and **Stop Character Code (E1)** . When the stop character is disabled no stop conditions are generated at all i.e. once opened, the serial data block never ends. Additionally, the **Number Of Post Characters (P1) setting** defines the number of characters past the stop character that will be counted as belonging to the same data block.

Break conditions are generated depending on two settings: **Maximum Intercharacter Delay (MD)** and **Maximum Data Length (ML)**. The first one defines the maximum time, in milliseconds, the DS will wait for the arrival of the next serial character. If this time is exceeded the break condition is generated. The second setting defines the number of bytes in the serial-to-Ethernet buffer, which, when exceeded, will generate a break condition.

Note, that all settings described above do not directly define the length of individual Ethernet packets generated by the DS, but only define when the data in the serial-to-Ethernet buffer is committed. Once the data is committed the DS will attempt to deliver this data to the network host as soon as possible but not necessarily in one "chunk".

Factory defaults for the packet generation settings provide a simple and usually acceptable schema: start on any character is enabled, stop-character is disabled, maximum intercharacter delay is set to 1 (10ms), and maximum data length is 255 bytes for UDP transport protocol and 127 bytes for TCP (see **Maximum Data Length (ML) setting** for explanation). This setup is universal and can handle random data of any size. All data is recorded into the serial-to-Ethernet buffer and committed either when the maximum size is reached or when there is a gap in the serial data arrival.

** The term "serial data blocks" was coined to avoid using the word "packets" that might cause a confusion with the network packets.*

*** The serial port must be opened for this to happen.*

Programming

The DS is programmed using programming commands that can be sent through the serial port of the DS or via the network. For each command the DS issues a reply.

All DS commands have the following format:

C.C.	Optional parameter(s)

- **C.C.** is the command code. Command code always consists of a single ASCII character. All available commands and their codes are listed in the command table at commands, messages, and replies.
- **Optional parameter(s)** field contains necessary data if required by the command.

All DS replies have the following format:

R.C.	Optional data

- **R.C.** is the reply code. Reply codes inform the sender of the command about the result of command execution. All available reply codes are listed in the reply code table at commands, messages, and replies.
- **Optional Data** field contains necessary data if requested by the command.

Example: here is a sample exchange between a certain device (performing the programming) and the DS. This device can be a network host (programming through the network) or attached serial device (programming through the serial port):

```
-->DS:    GIP
DS-->:    A192.168.100.90
```

In the above example programming device requests the IP-address of the DS. This is done by issuing the **Get Setting (G) command** with parameter "IP" (for **IP-address (IP) setting**). The DS replies with the **OK (A) reply code** indicating that command was processed successfully, followed by the current value of the **IP-address (IP) setting**, which is 192.168.100.90.

All commands and replies sent to the DS always have the same format, regardless of which interface is used. What is interface-dependent is the encapsulation that is used to mark the beginning and the end of commands (replies) and how these commands and replies are sent. For more information see serial programming and network programming.

Serial Programming

When the DS is powered up its serial port is running in the data routing mode (see serial port and serial communications). To enable DS programming via the serial port the latter must be switched into the *serial programming mode*.

Status LEDs of the DS are playing a *serial programming mode pattern* when the serial port of the DS is in the serial programming mode (click here to see all available patterns).

There are two methods of putting the serial port of the DS into the serial programming mode:

By pressing setup button (for DS100R, DS100B, DS203; for EM100, EM120,

EM200, EM203(A)- by pulling MD line low for at least 100 ms**.** This forces the DS to enter the serial programming mode with default communication parameters of 38400-8-N-1, *half-duplex mode* (regardless of the value of corresponding settings). This method always works and cannot be disabled.

By sending an escape sequence to the serial port (at a current baudrate). Once the escape sequence is recognized the DS will enter the serial programming mode at a *current* baudrate. Other communications parameters will still default to 8-N-1, *half-duplex mode*. For the escape sequence to work it must be previously enabled through the **Soft Entry (SE) setting**. Additionally, this setting provides two different escape sequences to choose from.

Notice, that while in the serial programming mode the serial port of the DS uses half-duplex mode of operation (suitable for RS485 communications). In this mode the RTS line provides direction control and the CTS line is unused. When the DS is waiting for the serial command to arrive the RTS line is HI*. When the DS outputs its reply to the serial command the RTS line is LOW* for as long as it takes to output this reply. Such behavior is intended to allow the RTS line to control the direction pin of RS485 interface ICs and RS232-to-RS485 converters.

Just because the DS is using the half-duplex operation in the serial programming mode doesn't mean that programming through RS232 or RS422 is not supported. If the actual hardware port on the DS is RS232 (RS422) then TX and RX lines should be used to send commands and receive replies while the RTS line should simply be ignored (i.e. RTS/CTS flow control must be disabled on the serial device connected to the DS). This was the rationale for choosing the half-duplex mode: it does not interfere with programming through RS232 or RS422 while making the DS also ready for programming via RS485.

All serial commands and replies use the following format:

STX	Command/reply	CR

STX (ASCII code 2) and **CR** (ASCII code 13) characters provide necessary encapsulation. All data before the STX and after the CR is ignored.
Command/reply field contents has been explained in programming.

Example: here is a sample exchange between a serial device and the DS. Special characters are represented as follows: STX- ☺, CR- ⏎.

Serial device-->DS: ☺**GIP**⏎
DS-->Serial device: ☺**A192.168.100.40**⏎

** HI and LOW states are described with respect to the serial ports of DS100R, DS100B, DS203. For EM100, EM120, EM200, EM203(A) the signaling is exactly opposite.*

Network Programming

Unlike serial programming that requires the serial port of the DS to be in the serial programming mode, in which the serial port cannot exchange data with the network host, network programming can continue concurrently with the data routing between the network host and attached serial device. Therefore, network programming is not a "mode" of the Ethernet port.

There are three methods of network programming:

Out-of-band UDP programming is effected by sending programming commands as UDP datagrams to one of the two *UDP command ports*- either 65535 or 32767*. This programming method is called out-of-band programming since commands are

sent outside of (separately from) the data connection. Programming through UDP command ports does not depend on any prior setup, always available and cannot be disabled.

Inband TCP programming is effected by sending programming commands within the TCP data connection itself, commands and replies are mixed right into the routing data stream (hence, the term "inband"). Inband programming is only possible when the **current Transport Protocol (TP) [setting/parameter]** is 1 (TCP) and **Inband (IB) setting** is 1 (enabled). Therefore, certain prior setup is needed for this programming method to work.

Command-phase TCP programming is also effected by sending programming commands within the data TCP connection. The difference with the inband programming is that the routing data and commands are not intermixed. Instead, the TCP connection itself is divided into two phases: *command-phase* in which programming can be performed (hence, the name of this programming method) and the *data phase* in which the data is routed. Command-phase TCP programming is only possible when the **current Transport Protocol (TP) [setting/parameter]** is 1 (TCP) and the **Data Login (DL) setting**is 1 (enabled). Therefore, certain prior setup is needed for this programming method to work.

"Telnet" TCP programming (**supported by firmware V3.50 and above**) is effected by establishing a TCP connection to the "telnet" port (port 23) of the DS. Such connection can be established regardless of whether **current Transport Protocol (TP) [setting/parameter]** is 0(UDP) or 1(TCP). Several TCP connections to port 23 can be established at the same time but only one of them will be at "logged in" state at any given time. With this new functionality, port 23 cannot be used as the data port of the DS.

** Existence of two identical command ports is historical. Originally, only one UDP port 65535 was allocated for programming. Later it was discovered that some routers can only forward traffic to port numbers 0-32767. Consequently, another command port at 32767 was introduced.*

Out-of-Band (UDP) Programming

With out-of-band UDP programming commands and replies are sent as UDP datagrams, one command or reply per datagram. As explained in network programming the DS accepts commands on two UDP ports- 65535 and 32767. Reply to a particular command is always sent to the IP-address and the port number from which this command was received.

Because each command and reply is sent in a separate UDP datagram no additional encapsulation (i.e. using STX and CR characters as in serial command/replies) is necessary.

Example: here is a sample exchange between the network host and the DS. Each line represents the data in a separate UDP datagram.

Host-->DS: **SBR4**
DS-->Host: **A**

One additional feature supported by the DS is the ability to receive and process *multiple* out-of-band commands. The network host can send up to eight such commands and the DS will receive and process them all one-by-one. This dramatically increases the speed with which the DS programming can be performed over the network.

Since delivery of UDP packets is not guaranteed and packets can also arrive

out-of-order the DS supports an optional command ID field. This field can be up to four ASCII characters long and contain any characters in the 0-9 and A-z range. ID field is added in the end of the command itself, '|' character (ASCII code 166) is used as a separator. When the DS receives a command that contains the ID field it issues the reply with the same ID field at the end. By sending commands with changing command ID the network host can match commands and replies.

Example: here is a sample exchange of two commands and replies between the network host and the DS. Each line represents the data in a separate UDP datagram. Replies arrive out of order but owing to the ID field the network host can still match each reply to its respective command.

Host-->DS: SBR4|1923
Host-->DS: SPR0|1924
DS-->Host: A|1924
DS-->Host: A|1923

Once again, the ID field is added at the end of all command data, which includes all command fields (mandatory or optional) described for individual commands. For example, **Parameter (P) command** can have an optional password field so command ID is added past this field (when it is present):

Host-->DS: PBR4/123pwd|1925

In the above example the password is "123pwd".

Broadcast Out-of-Band (UDP) Programming

Several DS commands are intended to be sent as broadcast UDP datagrams (column 'B' in the command table at commands, messages, and replies details which commands are accepted in broadcast packets). For example, the **Echo (X) command**, when sent as UDP broadcast, makes all locally attached DS send back a reply. This way it is possible to "auto-discover" all Device Servers on the local network segment and also receive their current status.

All other commands, not marked with a '+' in the 'B' column are supposed to address specific DS. When sent in the broadcast mode they are ignored by all Device Servers except the one that has been pre-selected using the **Select In Broadcast Mode (W) command**. This method provides a way to program the DS with invalid IP-address (address, that is incompatible with the network) when it is not possible to communicate with this DS in a "normal" way (i.e. by sending UDP datagrams to its IP-address). Once a certain DS has been pre-selected commands such as **Set Setting (S)** sent in the broadcast mode are accepted and processed by this DS as if they were regular non-broadcast datagrams.

Inband (TCP) Programming

Inband commands and replies are sent within the TCP data connection itself and can be mixed with the data being routed between the network host and attached serial device. As explained in network programming this method only works when the **current Transport Protocol (TP) [setting/parameter]** is 1 (TCP) and the **Inband (IB) setting** is 1 (enabled).

Since inband commands and replies are intermixed with the data a special escape character called Inband Escape Character (IEC) is needed to mark the beginning of the command (reply) in the data stream. IEC code is 255. When the **Inband (IB) setting** is 1 (enabled) and the DS receives a character with code 255 followed by any character *other than* 255, the DS considers this to be a beginning of the

inband command. If the network host wants to send a *data character* with code 255 it needs to send *two consecutive characters* with code 255. This will be interpreted by the DS as a *single* data character with code 255. Therefore, it is the responsibility of the network host to parse through the data it is sending to the DS and "double" all data characters with code 255.

When sending a reply back to the network host, the DS will also prefix this reply with the IEC. When sending a data byte with code 255 the DS will automatically double this character. It is the responsibility of the network host to parse through the data it receives from the DS and replace all double characters with code 255 with a single one.

All inband commands and replies use the following format:

IEC	STX	Command/reply	CR

IEC character followed by **STX** (ASCII code 2) mark the beginning of command (reply). **CR** (ASCII code 13) marks the end of command (reply). All characters before the IEC/STX and after the CR are considered to be a "regular" routing data. **Command/reply** field contents has been explained in programming.

Example: here is a sample exchange between the network host and the DS. Shown below is the data passed within a data TCP connection. Special characters are represented as follows: IEC- ◆, STX- ☺, CR- ⏎.

Host-->DS: ABC◆◆DEF◆☺GIP⏎GHIK
DS-->Host: LMN◆☺A192.168.100.40⏎XYZ

In the above exchange the network host is sending the following data string: "ABC◆DEFGHIK". There is a single character with code 255 (◆) but the DS has to double this character. Command inserted into this data stream is "GIP" (**Get Setting (G) command**, setting to be read is **IP-address (IP)**). Command is preceded with IEC (◆)followed by STX (☺) and ended by CR (⏎).

The DS sends the following data string to the network host: "LMNXYZ". At some point, when the reply to the command is ready, the DS inserts this reply into the data stream. Reply contains current IP-address of the DS.

Unlike out-of-band UDP programming, inband programming does not support multiple commands. It is not possible to send several commands and receive several replies back. The network host should only send the next command after having received a reply to the previous command.

Very important! Since inband commands are transmitted together with data execution of such commands by the DS can be delayed indefinitely if the data cannot be transmitted by (sent out of) the serial port of the DS. This will happen if **current Flow Control (FC) [setting/parameter]** is at 1 (enabled) and the CTS input of the DS is held in the "do not transmit" state. In this case the DS will not be sending out data and the inband command "embedded" within this data stream won't be processed (until all data before this command is finally transmitted).

Command-Phase (TCP) Programming

Another method of programming is called *command-phase* programming. As explained in network programming it only works when the **current Transport Protocol (TP) [setting/parameter]** is 1 (TCP) and the **Data Login (DL) setting** is 1 (enabled). In this mode, the TCP data connection between the network host and the DS is split into two phases: *command phase* and *data phase*.

When the **Data Login (DL) setting** is 1 (enabled) and the data TCP connection is

established the DS enters the command phase. In this phase the DS interprets all data sent by the network host as programming commands. Because no routing data can be transmitted in this phase an escape character (like the one used in the inband programming) is not needed.

All command-phase commands and replies use the following format:

STX	Command/reply	CR

Thus, command-phase commands and replies have the same format as those used in serial programming. **STX** (ASCII code 2) and **CR** (ASCII code 13) characters provide necessary encapsulation. All data before the STX and after the CR is ignored. **Command/reply** field contents has been explained in programming.

To switch the data connection from the command phase into the data phase the network host has to issue the **Logout (O) command**. After this command is sent* and accepted the DS switches into the data phase. From this moment on and until the TCP connection is terminated the network host can exchange the data with the attached serial device in the normal way.

Logout (O) command is only accepted after the network host logs in using **Login (L) command**, and this requires a valid password (if set in the **Password (PW) setting**). Therefore, setting **Data Login (DL)** to 1 (enabled) also enables network host authentication for data exchange with the attached serial device(hence, the name of this setting).

Example: here is a sample exchange that switches the DS into the data phase. Shown below is the data passed within a data TCP connection. Special characters are represented as follows: STX- ☺, CR- ⏎. Login password is "123pwd"

---TCP connection established, command phase---
Host-->DS: ☺**L123pwd**⏎ 'network host logs in
DS-->Host: ☺**A**⏎ 'OK
Host-->DS: ☺**O**⏎ 'exit into the data phase (no reply)
---data phase from this point and until the TCP connection is closed---

Command-phase programming should not be confused with inband programming. The first one is enabled by the **Data Login (DL) setting**, the second one- by the **Inband (IB) setting**. Both can be enabled at the same time! After the DS exits into the data phase inband commands can still be sent (this time, with proper escape character), provided that inband programming is enabled.

Command-phase programming is disabled automatically when **current Link Service Login (TL) [setting/parameter]** is 0 (disabled).

**Command-phase inband programming is the only programming method in which no reply is returned upon successful completion of the Logout (O) command. The DS simply switches into the data phase.*

Telnet TCP Programming [V3.50 and Above]

"Telnet" TCP programming is the new programming method that is **supported by firmware V3.50 or higher**. Telnet programming is called so because it is effected through a TCP connection to port 23 of the DS (This is a standard telnet port). Such connection can be established regardless of whether **current Transport Protocol (TP) [setting/parameter]** is 0(UDP) or 1(TCP).

Several TCP connections to port 23 can be established at the same time but only one of them will be at "logged in" state at any given time i.e. have a programming session in progress (see authentication and programming priorities).

Note, that port 23 cannot be used for programming if this port is assigned to be the data port (i.e **Port Number (PN)**= 23). In this case all TCP connections to port 23 will be interpreted as data connections, thus rendering "telnet" programming impossible. If the **Port Number** is set to any port other than 23, a TCP connection to port 23 will be interpreted as a programming connection.

All telnet commands and replies use the following format:

STX	Command/reply	CR

Thus, telnet commands and replies have the same format as those used in serial programming. **STX** (ASCII code 2) and **CR** (ASCII code 13) characters provide necessary encapsulation. All data before the STX and after the CR is ignored. **Command/reply** field contents has been explained in programming.

The replies from the DS will include an additional LF character trailing the CR at the end of the reply, just to improve readability on most terminal software. You don't have to send an LF when you send commands to the DS.

Authentication

Certain commands, when sent through the network, require authentication. To authenticate itself the network host must provide a password, that matches the one defined by the **Password (PW) setting**.

From the authentication standpoint, all commands can be divided into three groups:

Commands that do not require authentication. These commands can be sent at any time and by any network host.

Commands with immediate authentication. In these commands the password is supplied in the command body itself and authenticates this particular command. For some of these commands authentication is optional.

Commands that require prior login. These commands are only accepted after the network host has logged in using the **Login (L) command**. Login is performed once and is said to open a *programming session*.

The DS memorizes the source IP-address from which the **Login (L) command** is sent as well as the mode in which it is sent: out-of-band, inband, etc. Programming session must continue from the same IP-address and using the same way of command delivery. So, if the session was opened using out-of-band **Login (L) command** and the network host sends inband **Set Setting (S) command** (this command requires prior login) then this command is not considered to be a part of the opened programming session and is rejected.

Programming sessions are ended either by switching the DS off or using **Logout (O)** or **Reboot (E)** commands. There is also a two-minute programming session timeout: if no command (that requires prior login) is issued for two minutes the session is ended automatically. Inband, command-phase, and telnet-mode programming sessions are also closed when their TCP connection is closed.

The DS makes sure that only one programming session is opened at any given time- see programming priorities for details.

Command table at commands, messages, and replies details which commands require authentication (see 'L' and 'I' columns).

Sending **Login (L) command** to open a programming session is required even

when the DS is running in the error mode but sending login password in this case is not necessary.

Programming Priorities

The DS has five programming methods: serial, out-of-band UDP, inband TCP, command-phase TCP, and telnet programming. For the subject discussed below inband and command-phase programming are the same since they both take place within a data TCP connection. Therefore, four programming methods will be discussed: serial, out-of-band, inband/command-phase, and telnet.

Since all four programming methods can be used at the same time the DS maintains priority mechanism to avoid conflicts that might arise if attached serial device and the network host(s) attempted to program the DS at the same time.

Serial programming has the highest priority of all- any command sent to the DS via the serial interface (in the serial programming mode) is always accepted and processed*, regardless of whether any form of network programming is (has been) taking place at the time.

As explained in authentication, all network commands can be divided into those that do not require any authentication, commands that require immediate authentication, and commands that require prior login using **Login (L) command** (commands that need a *programming session* to be opened).

Network commands that do not require authentication can be sent at any time, using any method. For example, network **Echo (X) command** will be accepted and processed even when the DS is in the serial programming mode.

Network Commands that require immediate authentication can be sent at any time and using any method, as long as the DS is not in the serial programming mode. When the DS is in the serial programming mode these commands are rejected (**R reply code**).

For network commands that require prior login the following hierarchy of priorities is applied:

- Serial programming (highest priority)
- Out-of-band programming session
- Telnet programming session
- Inband or command-phase programming session (lowest priority)

The above should be understood as follows:

- Programming session using lower-priority programming method cannot start while higher-priority programming session is in progress. For example, programming session using out-of-band commands cannot start while the DS is in the serial programming mode. Programming session using TCP connection to telnet port cannot start while out-of-band session is in progress.
- Higher-priority programming can start at any time, even when lower-priority programming session was already in progress. In this case lower-priority programming session is aborted immediately. Thus, if out-of-band programming session was already opened and the DS enters a serial programming mode then out-of-band programming session is aborted.
- Programming session that is already opened cannot be interrupted by the same priority level programming session. The DS allows several simultaneous TCP

connections to telnet port 23. Only one of these connections, however, can carry a programming session at any given time. Attempt to login through another TCP connection to port 23 will be rejected for as long as the earlier session remains opened.

** As long as this command is allowed to be sent through the serial port at all. Command table at commands, messages, and replies details which commands can be issued through the serial port (see 'S' column).*

Error Mode3.4

The value of each DS setting (stored in the EEPROM) is protected by its own checksum. Every time the DS stores new setting value into the EEPROM it recalculates the checksum and saves it into the EEPROM along with the new setting data. Every time the DS reads out the value of a certain setting it verifies this setting's checksum. If the checksum error is detected (for any setting) the DS enters an *error mode*.

Status LEDs of the DS are playing an *error mode pattern* when the DS is in the error mode (unless the serial port of the DS is in the serial programming mode. Click here to see all available patterns).

Once entered, the error mode cannot be exited other then by rebooting the DS- either by power-cycling it or executing the **Reboot (E) command**.

DS operation in the error mode is characterized by the following:

- Status LEDs of the DS are displaying an error mode pattern (unless the DS is in the serial programming mode);
- Error status (***e*** flag) returned by the **Echo (X) command** is set to 'E';
- For every invalid setting the DS takes the default value* of this setting (which is fixed and does not depend on the EEPROM data) and assumes this default value as a run-time parameter. For example, if the **DHCP (DH) setting** is found to be invalid the DS will boot up with DHCP off because the default value of the **DHCP (DH) setting** is 0 (disabled);
- For two setting- **Routing Mode (RM) setting** and **Password (PW) setting**- the DS uses their default values in any case, even if these settings are *valid*. This means that:
  - The DS will be in the slave routing mode (default value of the **Routing Mode (RM) setting**);
  - Password authentication will be disabled (default value of the **Password (PW) setting** is <NULL>). Consequently:
    - No password is to be supplied in the **Login (L) command**, although this command itself must still be executed before sending commands that require prior authentication (such as the **Initialize (I) command**);
    - Login password doesn't need to be supplied in commands that require immediate authentication, such as on-the-fly (network side) **Parameter (P) command**, even if the **On-the-fly Password (OP) setting** is 1 (enabled).

The above also applies to two most important settings that define DS visibility on the network- the **MAC-address (FE) setting** and the **IP-address (IP) setting**. When invalid, actual values of these settings are substituted with default ones,

which are 0.1.2.3.4.5 and 192.168.100.40 respectively. This means that the DS will still be accessible through the network, but at default MAC and IP.

It is best to reinitialize the DS as soon as it is found to be in the error mode. This can be done through the **Initialize (I) command** or by using quick initialization.

** Default values can be changed through custom profile*

Quick Initialization

Quick initialization is a way to completely reinitialize the DS without using any commands. This feature is handy when the DS is running in the error mode and needs to be repaired.

Quick initialization provides the same result as the *serial* **Initialize (I) command**. See individual setting description for the information on how each settings will behave during the initialization (some settings will be initialized unconditionally and some- only when found to be invalid).

To quick-initialize the DS:

- Make sure the DS is powered;
- Press and release the Setup button* to enter the serial programming mode. Status LEDs will play the serial programming mode pattern;
- Wait for at least three seconds;
- Press and hold the Setup button for at least three seconds- both status LEDs will be switched off and this will indicate that the initialization has been started;
- Initialization takes about one second to complete. Initialization result will be shown by the status LEDs- green LED will be switched on for about 2 seconds indicating successful initialization. (If the red LED is switched on this means that the DS is malfunctioning).
- Switch the DS off and back on again to exit the error mode if necessary.

** For EM100, EM120, EM200, EM203(A) pull the MD line low*

Custom Profiles

Default setting values (i.e. values that settings assume after they are initialized through the **Initialize (I) command** or by using quick initialization) can be changed by adding a *custom profile* to the firmware file loaded into the DS.

Request more information on the subject via email.

Reference

Reference contains all necessary information on:

- **DS commands, messages and replies**. Commands are used to control the DS and can be issued through the network or through the serial port. The DS replies to the commands- reply codes indicate the result of command execution. Messages are different in that they are not replied to. For more information see programming.
- **DS settings**. Settings are *permanent functioning parameters* that are stored in

the non-volatile memory (EEPROM) of the DS. Once programmed, they remain intact even when the DS is powered off. Many (but not all) settings require the DS to be rebooted for the new setting value to take effect.

- **DS parameters and instructions**. Parameters are *temporary overrides* for settings. Parameters are not saved into the EEPROM and take immediate effect (no rebooting required). **Instructions** are used to make the DS perform a certain action.

Commands, messages, and replies

This section contains a reference for all DS commands and messages.

Commands are used to control the DS and can be issued through the network or through the serial port. The DS replies to the commands- reply codes indicate the result of command execution. Messages are different in that they are not replied to. For more information see programming.

Command and message description format can be found here.

Table below lists all available commands and messages:

C.C.	N	B	L	I	S	Description
L	+					Login command
O	+		+		+	Logout command
E	+		+		+	Reboot command
I	+		+		+	Initialize command
S	+		+		+	Set Setting command
G	+		+		+	Get Setting command
P	+			(+)	+	Parameter command
X	+	+			+	Echo command
U	+	+			+	Status command
B	+					Buzz command
R	+					Reset Overflow Flags command
A	+	+		+		Assign IP-address command
W	+	+				Select In Broadcast Mode command
V	+	+			+	Get Firmware Version command
N	+		+			Jump To NetLoader command
N	+		+			Set Programming Request Flag
Q	+		+			Reset Upload Process
D	+		+			Upload Data Block
C					+	Cable Status
T	+	+				Get My IP
J	+					Notification message

Notes:

- **C.C.**- command codes;
- **N**- '+' in this column indicates that command can be issued through the network;
- **B**- '+' in this column indicates that command can be issued in the broadcast mode (when sent through the network) without the need to pre-select a particular DS using **Select In Broadcast Mode (W) command** first;
- **L**- '+' in this column indicates that command, when issued through the network, requires prior login using the **Login (L) command**;

- **I**- '+' in this column indicates that command, when issued through the network, requires immediate authentication. '(+)' indicates that immediate authentication is optional;
- **S**- '+' in this column indicates that command can be issued through the serial port.

Listed below are all available reply codes:

R.C.	Description
A	**OK** (command completed successfully)
C	**Error** (incorrect command was issued)
R	**Rejected** (command was rejected by the DS)
D	**Denied** (access was denied by the DS)
F	**Failed** (command execution failed)
S	**Bad Sequence** (D command only)
O	**Out-of-range** (D command only)

Notes:

- **R.C.**- reply code.

Command & message description format

All commands in this section are described using the following format:

Function:	Command function in brief
Can be issued through:	Describes whether command can be issued through the network and/or serial port, or both; also lists additional conditions- whether or not prior login is required for network command, etc.
Command format:	Shows command syntax
Possible replies:	Lists all possible reply status codes that can be returned in response to this command
First introduced:	Describes whether this command has been available right from the "baseline" firmware version of 3.14/3.51 or was introduced in a later firmware release
See also:	Additional relevant links

Details

Additional information about the command.

Login (L) command

Description (see command description format info here)

Function:	Authenticates the network host and opens the network programming session
Can be issued through:	Network (broadcasts ignored*)
Command format:	**L*pp...p***, where ***pp...p*** is the login password, 0-8 characters long
Possible replies:	**A**, **R**, **D**

First introduced:	Earlier than "baseline" V3.14/V3.51
See also:	Authentication, Programming priorities, Error mode

Details

Login command is used to authenticate the network host, which is a necessary step to perform before attempting to execute certain commands over the network (such as **Set Setting (S)**. Check the "L" column of the command table at commands, messages, and replies to find out which commands require prior authentication.

Login password to supply in the **Login command** body is defined by the **Password (PW) setting**. Note, that to execute commands that require prior login you need to use the **Login command** even when the password is not set (i.e. it is <NULL>).

OK (A) reply code is returned if login is accepted by the DS. **Rejected (R) reply code** is returned if the DS is in the serial programming mode or a higher-priority programming session is already in progress (see programming priorities for details). **Denied (D) reply code** is returned if incorrect password is supplied.

Login command is said to open a programming session. For more information on programming sessions see authentication.

The DS memorizes the IP-address of the network host from which the **Login command** is sent. If subsequent commands (that require login) are sent from a different IP-address then these commands are replied to with the **Denied (D) reply code**.

Login password is not verified when the DS is running in the error mode. **Login command** should still be used as usual but it is not necessary to supply any password, leaving the ***pp...p*** field blank in this situation is OK.

** Without prior selection using Select In Broadcast Mode (W) command.*

Logout (O) command

Description (see command description format info here)

Function:	Network command: logs out the network host (ends the programming session); serial command: ends the serial programming mode
Can be issued through:	Network (broadcasts ignored*, login required); serial port
Command format:	**O**
Possible replies:	**A**, **D**
First introduced:	Earlier than "baseline" V3.14/V3.51
See also:	Authentication, Out-of-band (UDP) commands, Inband (TCP) commands, Command-phase (TCP) commands, Serial programming

Details

Logout command performs a different action depending on whether it was issued through the network or through the serial port.

When the **Logout command** is issued through the network it closes the programming session that was opened using the **Login (L) command**. **Denied (D) reply code** is returned if the programming session is not in progress or if it doesn't belong to the sender of the **Logout command**- the DS remembers the IP-address of the network host that opens the programming session and requires that all subsequent commands (that require prior login) are sent from the same IP.

OK (A) status code is returned if command is accepted, but only if this command was sent as an out-of-band (UDP) or inband (TCP) command. No reply is sent in case of command-phase (TCP) command, the DS simply switches into the data phase (**Denied (D) reply code** is still returned).

When the **Logout command** is issued through the serial port it makes the DS exit the serial programming mode (and into the data routing mode). Since no authentication is required for DS programming through the serial port the only possible reply code in this case is **OK (A)**.

Executing **Logout command** does not make the DS reboot. If it is necessary to reboot the DS (i.e. to make it reread updated setting values) the **Reboot (E) command** should be used instead.

** Without prior selection using Select In Broadcast Mode (W) command.*

Reboot (E) command

Description (see command description format info here)

Function:	Causes the DS to reboot
Can be issued through:	Network (broadcasts ignored*, login required); serial port
Command format:	**E**
Possible replies:	**D**
First introduced:	Earlier than "baseline" V3.14/V3.51
See also:	Authentication

Details

Reboot command causes the DS to reset (restart).

When issued through the network, this command requires prior login using the **Login (L) command** (programming session must be opened). **Denied (D) reply code** is returned if the programming session is not in progress or if it doesn't belong to the sender of the **Reboot command**- the DS remembers the IP-address of the network host that opens the programming session and requires that all subsequent commands (that require prior login) are sent from the same IP.

No reply code is returned if command is accepted- the DS simply reboots. This makes the DS "lose" any data connection that might be in progress so the network host must discard such a connection if it was established before.

Reboot command can be used to exit the serial programming mode or end the network programming session. This may be necessary, for instance, to make the DS reread new setting values that were programmed prior to the reboot. If it is necessary to end the programming without rebooting the DS the **Logout (O) command** should be used instead.

** Without prior selection using Select In Broadcast Mode (W) command.*

Initialize (I) command

Description (see command description format info here)

Function:	Initializes the settings of the DS
Can be issued through:	Network (broadcasts ignored*, login required); serial port
Command format:	**I**
Possible replies:	**A**, **D**, **F**
First introduced:	Earlier than "baseline" V3.14/V3.51
See also:	Authentication

Details

Initialize command restores the settings of the DS to their default factory values**. Some settings are initialized each time this command is executed. Other settings are initialized or left intact depending on the interface (network or serial) through which the **Initialize command** was issued and also on whether or not a particular setting contained a valid or invalid value.

All settings can be divided into three groups:

- Settings that are initialized unconditionally, no matter whether the **Initialize command** is issued through the network or through the serial port. For example, the **Routing Mode (RM) setting** is always reset to 0 (server) after the initialization.
- Settings that are initialized unconditionally for the serial **Initialize command**, but are only initialized in case they contained an invalid value (or found to be corrupted)*** for the network **Initialize command**. One of such settings is the **IP-address (IP)**. Since resetting the IP-address might possibly make the DS inaccessible through the network the risk is minimized by leaving the setting intact unless it is "bad".
- Settings that are only initialized when they contained an invalid value, no matter whether command was issued through the network or through the serial port. For example, the **MAC-address (FE) setting** is never altered unless found to be invalid. This is because the MAC-address is preset during the production and should not be altered unless absolutely necessary.

When issued through the network, the **Initialize command** requires prior login using the **Login (L) command** (programming session must be opened). **Denied (D) reply code** is returned if the programming session is not in progress or if it doesn't belong to the sender of the **Initialize command**- the DS remembers the IP-address of the network host that opens the programming session and requires that all subsequent commands (that require prior login) are sent from the same IP.

Failed (F) reply code is returned if the DS fails to reset one or more settings. This usually indicates a hardware malfunction (EEPROM failure).

** Without prior selection using Select In Broadcast Mode (W) command.*

*** Or the values defined by the Custom profile created by the User.*

**** Each setting's data is protected by an individual checksum.*

Set Setting (S) command

Description (see command description format info here)

Function:	Sets (writes) new setting value
Can be issued through:	Network (broadcasts ignored*, login required); serial port
Command format:	**S*ssvv...v***, where ***ss***: setting mnemonic, ***vv...v***: new setting value
Possible replies:	**A**, **D**, **C**, **F**
First introduced:	Earlier than "baseline" V3.14/V3.51
See also:	**Get Setting (G) command**

Details

Set Setting command assigns new values to the selected setting. ***ss*** is the setting mnemonic, i.e. "IP" for the **IP-address (IP) setting**.

Example: to set IP-address to 192.168.100.40 issue the following command**:

```
-->DS:  SIP192.168.100.40
DS-->:  A
```

When issued through the network, this command requires prior login using the **Login (L) command** (programming session must be opened). **Denied (D) reply code** is returned if the programming session is not in progress or if it doesn't belong to the sender of the **Set Setting command**- the DS remembers the IP-address of the network host that opens the programming session and requires that all subsequent commands (that require prior login) are sent from the same IP.

Error (C) reply code is returned if the setting mnemonic and/or supplied setting value is/are invalid. **Failed (F) reply code** is returned if the DS fails to write new setting value. This usually indicates a hardware malfunction (EEPROM failure).

** Without prior selection using Select In Broadcast Mode (W) command.*

*** Encapsulation characters such as IEC, STX, CR are not shown.*

Get Setting (G) command

Description (see command description format info here)

Function:	Gets (reads) new setting value
Can be issued through:	Network (broadcasts ignored*, login required); serial port
Command format:	**S*ss***, where ***ss*** is the setting name
Possible replies:	**A*vv...v***, **D**, **C**, **F**, where ***vv...v*** is current setting value
First introduced:	Earlier than "baseline" V3.14/V3.51
See also:	---

Details

Get Setting command reads out current value of the selected setting. ***ss*** is the setting mnemonic, i.e. "IP" for the **IP-address (IP) setting**.

Example: to read current IP-address issue the following command**:

-->DS: **GIP**
DS-->: **A192.168.100.40**

When issued through the network, this command requires prior login using the **Login (L) command** (programming session must be opened). **Denied (D) reply code** is returned if the programming session is not in progress or if it doesn't belong to the sender of the **Get Setting command**- the DS remembers the IP-address of the network host that opens the programming session and requires that all subsequent commands (that require prior login) are sent from the same IP.

Error (C) reply code is returned if the setting mnemonic is invalid. **Failed (F) reply code** is returned if the DS fails to read out the setting value, setting value is not valid, or if the data is found to be corrupted***. This *may* indicate a hardware malfunction (EEPROM failure).

Invalid setting can be repaired using the **Set Setting (S) command** or (recommended) using the **Initialize (I) command** or quick initialization (this will initialize all other invalid settings).

** Without prior selection using Select In Broadcast Mode (W) command.*

*** Encapsulation characters such as IEC, STX, CR are not shown.*

**** Each setting's data is protected by an individual checksum.*

Parameter (P) command

Description (see command description format info here)

Function:	Sends parameter or instruction to the DS
Can be issued through:	Network (broadcasts ignored*, login not required, immediate authentication for this command can be enabled); serial port
Command format:	**P*<relevant data>*** *(see description of individual parameters and instructions for details)*
Possible replies:	**A**, **C**, **D**, **R**
First introduced:	Earlier than "baseline" V3.14/V3.51
See also:	Parameters and instructions, Authentication

Details

Command code 'P' (Parameter) serves as a "common entry point" for sending a variety of parameters and instructions.

Network **Parameter command** can be made to require immediate authentication. To enable this feature:

- **On-the-Fly Password (OP) setting** must be programmed to 1 (enabled);
- **Password (PW) setting** must contain a password (i.e. a string of non-zero length).

When two above conditions are met each network **Parameter command** sent to the DS must contain a password, which is to be added at the end of the command after a slash character.

Example: below is the parameter command that changes the baudrate. The password is "123pwd"**:

-->DS: **PBR4/123pwd**
DS-->: **A**

If no password or incorrect password is supplied when the password is expected the **Denied (D) reply code** is returned.

** Without prior selection using Select In Broadcast Mode (W) command.*

*** Encapsulation characters such as IEC, STX, CR, and the command ID field are not shown.*

Echo (X) command

Description (see command description format info here)

Function: Returns DS status information

Can be issued through: Network (broadcasts OK, login not required); serial port

Command format: **X*[v]***, where ***v*** is an optional "command version" parameter (decimal number up to 255, when omitted defaults to 1). **Command version is only recognized in firmware V3.54+**

Possible replies (network): **A*nnn.nnn.nnn.nnn.nnn.nnn/ppppp/mseic[p]/ES/oo...o/dd...d***, where
nnn.nnn.nnn.nnn.nnn.nnn- MAC-address of the DS;
ppppp- data port numberof the DS;
m- fixed to 'N' (means that the application firmware, not the NetLoaderis running);
s- programming mode: '*' (none), 'S' (serial), 'U' (out-of-band UDP), 'T' (inband TCP or command-phase TCP);
e- error status: '*' (no errors detected), 'E' (running in the error mode); 'N' (Ethernet port failure)
i- IP-address status: '*' (not obtained yet), 'I' (obtained via DHCP), 'M' (fixed, set manually);
c- data connection status: '*' (closed), 'A' (sending ARP **OR [V3.54+]** establishing PPP link), 'O' (being established), 'C' (TCP connection established or being closed), 'U' (UDP connection established), 'R' (reset by remote host), 'F' (Link Server login failed), 'L' (Link Server login in progress);
[V3.54+] ***p*- only returned when command version parameter of >1 is supplied**: '*' (PPPoE disabled), 'D' (PPPoE login denied), 'N' (PPPoE link not opened), 'B' (PPPoE link is being established), 'P' (PPPoE link established);
E- Ethernet-to-serial bufferoverflow: '*' (no overflow), 'E' (overflow detected);
S- serial-to-Ethernet bufferoverflow: '*' (no overflow), 'S' (overflow detected);
oo...o- owner name;
dd...d- device name.

Possible replies (serial): **A*mseic[p]/ES*** (see field description above)

First introduced: Earlier than "baseline" V3.14/V3.51, **functionality extended in V3.54**

See also: Network programming, Serial programming, PPPoE

Details

The primary use of the *network* **Echo command** is to auto-discover Device Servers on the network: when the network host sends this command in the broadcast mode, it collects the replies from all locally attached Device Servers (hence, the name of the command). Reply from each DS contains all necessary information (MAC-address, etc.) that is needed to continue communicating with each specific DS in a non-broadcast mode.

Information returned by the Echo command contains the following data:

- **MAC-address** is the most important field that can be used to uniquely identify each DS*! Besides, the MAC-address is used (and, therefore, must be known in advance) as a reference to the particular DS in such commands as **Assign IP-address (A)** and **Select in Broadcast Mode (W)**.
- **Data port number** field is read directly from the **Port Number (PN) setting** of the DS.
- Follow are several one-character flag fields that tell the network host about the present status of the DS:
  - ***m* flag** always returns 'N'. This is meant to indicate that the DS is running (this) application firmware. In contrast, when the NetLoader is running this flag shows 'L' (NetLoader also supports **Echo command**);
  - ***s* flag** shows whether or not any form of programming is in progress. 'S' is returned when the serial port of the DS is in the serial programming mode. If the network programming session is in progress the 'U' is returned for out-of-band (UDP) programming session and 'T' is returned for inband (TCP) or command-phase (TCP)programming sessions;
  - ***e* flag** indicates whether or not the DS is running in the error mode. Additionally, for serial **Echo command** this flag can be set to 'N' indicating hardware failure of the Ethernet port;
  - ***i* flag** reflects current IP-address status. It is useful when the DHCP is enabled (see the **DHCP (DH) setting**). The flag is set to '*' while the DS is trying to obtain the IP-address from the DHCP server. When this is done the flag is set to 'I'. When the DHCP is off this flag returns 'M' (for "manual");
  - ***c flag*** reflects the connection state;
  - **[V3.54+]** ***p flag*** reflects PPPoE link state. **This flag is only returned when optional command version parameter is supplied and is >1;**
  - ***E* and *S* flags** display routing buffer overflows. Both flags are reset automatically when the data connection is closed or aborted. The flags can also be reset manually through the **Reset Overflow Flags (R) command**.
- ***oo...o* and *dd...d*** fields return the data from the **Owner Name (ON)** and **Device Name (DN)**settings. Using meaningful names simplifies identification of a particular DS.

Optional command version parameter has been introduced in firmware **V3.54**. Command version is a decimal number (up to 255). When command version is omitted it is assumed to be 1. Earlier firmware releases do not support this parameter and will simply ignore it (result will be the same as having command

version set to 1). When command version of 2 or higher is supplied **Echo command** returns an additional ***p flag*** in the reply. This flag reflects the state of PPPoE link.

Example #1: supposing X command returns the following reply**:

-->DS: **X**
DS-->: **A0.150.30.213.55.74/1001/NS*IC/*S/BigCorp/Device1**

This means that the MAC-address of this DS is 0.150.30.213.55.74, data connections are accepted on port 1001, the DS is operating normally, has serial programming in progress. The IP-address of this DS was successfully obtained via the DHCP, TCP connection is currently established, serial-to-Ethernet buffer overrun has been detected (within current data connection). The owner name and device name of this DS is "BigCorp" and "Device1" respectively.

Example #2: here is a command with version parameter**:

-->DS: **X2**
DS-->: **A0.150.30.213.55.74/1001/N**MUP/**/BigCorp/Device1**

In this example the DS is operating normally, is not being programmed, has a fixes IP-address (DHCP is off), has UDP "connection" in progress and *PPPoE link established*.

It is noteworthy that reply to **Echo command** should not be parsed basing on the optional command version parameter. Remember that older firmware ignores this parameter so there is no guarantee that extended information (***p flag***) will be returned by the DS. Of course, it is always possible to process the reply basing on the firmware version of the DS but recommended solution is to simply check the presence of additional ASCII character before the *forward slash* that separates flags ***mseic****[**p**]* from flags ***E*** and ***S***.

Some comment should also be made about ***flag c***. Status 'A' of this flag now means that either ARP packets are being sent in order to discover the destination network host **OR** PPPoE link establishment is currently in progress. The usage of a single status to designate these two different processes is possible because they are never needed at the same time. When PPPoE is enabled (**PPPoE Mode (PP) setting** is either 1 or 2) and the destination host is on a different network segment the DS doesn't need an ARP but needs to establish a PPPoE link (for more information see PPPoE). In all other cases the DS doesn't need PPPoE but has to send ARP.

When issued through the serial port, the **Echo command** returns the same information minus the MAC-address, data port number, owner name, and device name fields. This is because these fields are only needed to discover and identify the DS on the network and are not required on the serial side. The primary application of the serial **Echo command** is for the attached serial device to inquire current data connection and routing status of the DS. This may be used by the serial device, in conjunction with the modem commands (serial-side parameters and instructions) to control and monitor data connection establishment and termination by the DS.

There is also a **Status (U) command** that returns additional information about the status of the DS.

** This is because each DS, like any other Ethernet device, has a unique MAC-address preset during the production.*

*** Encapsulation characters such as IEC, STX, CR, and the command ID field are not shown*

Status (U) command

Description (see command description format info here)

Function: Returns additional DS status information

Can be issued through: Network (broadcasts OK, login not required); serial port

Command format: **U***[**v**]*, where ***v*** is an optional "command version" parameter (decimal number up to 255, when omitted defaults to 1). **Command version is only recognized in firmware V3.54+**

Possible replies (network): **A*ddd.ddd.ddd.ddd/ppppp/eee/ttt/ccc/sss/fff/r/sdfpb/RCTS****[/iii.iii.iii.iii]*, where
ddd.ddd.ddd.ddd- IP-address of the network host with which the data connection is (was/ to be) established;
ppppp- data port number on the network host with which the data connection is (was/ to be) established;
eee- total number of characters in the Ethernet-to-serial buffer;
ttt- capacity of the Ethernet-to-serial buffer;
ccc- number of committed characters in the serial-to-Ethernet buffer;
sss- total number of characters in the serial-to-Ethernet buffer;
fff- capacity of the serial-to-Ethernet buffer;
r- current baudrate (same numbering is used as in the **Baudrate (BR) setting**);
s- serial port state: '*' (closed), 'O' (opened);
d- serial port mode: 'F' (full-duplex), 'H' (half-duplex);
f- flow control: '*' (disabled), 'R' (RTS/CTS flow control);
p- parity: '*' (none), 'E' (even), 'O' (odd), 'M' (mark), 'S' (space);
b- bits per byte: '7' (7 bits), '8' (8 bits);
R- current state of the RTS (output) line: '*' (LOW*), 'R' (HIGH*);
C- current state of the CTS (input) line: '*' (LOW*), 'C' (HIGH*);
T- current state of the DTR (output) line: '*' (LOW*), 'T' (HIGH*);
S- current state of the DSR (input) line: '*' (LOW*), 'S' (HIGH*);
[V3.54+] *iii.iii.iii.iii*- only returned when command version parameter of >1 is supplied - IP-address of the DS on the PPPoE link.

Possible replies (serial): **A*ddd.ddd.ddd.ddd/ppppp****[/iii.iii.iii.iii]* (see

field description above)

First introduced: Earlier than "baseline" V3.14/V3.51, **functionality extended in V3.54**

See also: Network programming, Serial programming, Data routing, Serial-to-Ethernet data routing, PPPoE

Details

Status command returns additional information about the status of the DS. In conjunction with the **Echo (X) command** it can be used to obtain extensive information about the state of the DS.

The following data is returned:

- ***ddd.ddd.ddd.ddd*** and ***ppppp*** **fields**. IP-address and port number of the network host with which the data connection is (was/ to be) established. IP-address in this field shows the following:
  - After the power-up the fields returns the IP-address and port defined by the **Destination IP-address (DI) setting** and **Destination Port Number (DP) setting**;
  - If these default values are overridden by the **Destination IP-address (DI) parameter**, **Destination Port Number (DP) parameter**, or **Establish Connection (CE) instruction**, then the fields show new overriding values;
  - While the data connection is established and after it is closed (aborted) the fields show the IP-address and port of the network host with which this connection is (was) established. Notice that this may be different from the above- if the DS has accepted an incoming connection.
- ***eee*** and ***ttt*** **fields** show the total number of data bytes in the Ethernet-to-serial buffer and the capacity of this buffer. Capacity information is included because the buffer size is different for different models of the DS.
- ***ccc***, ***sss***, and ***fff*** **fields** show the number of committed bytes in the serial-to-Ethernet buffer, total number of bytes in this buffer, and the buffer capacity. Again, buffer capacity is included because it differs depending on the DS model.
- ***r***, ***d***, ***f***, ***p***, and ***b*** **flags** show current serial port setup. This information is useful because on-the-fly commands (network-side parameters and instructions) can change serial port communications parameters at any time.
- ***d*** **flag** reflects whether the serial port is in the half-duplex or full-duplex mode. This data may be of interest when the **Serial Interface (SI) setting** is 2 (auto) and you need to verify what mode the DS has assumed at startup.
- ***R***, ***C***, ***T***, and ***S*** **flags** reflect current status of the RTS, CTS, DTR, and DSR lines of the serial port. This information may be useful in debugging communications problems between the DS and the attached serial device.
- **[V3.54+] *iii.iii.iii.iii field-* only returned when command version parameter of >1 is supplied**- IP-address of the DS on the PPPoE link. As explained in PPPoE the DS negotiates a separate and different IP-address for PPPoE communications.

Optional command version parameter has been introduced in firmware **V3.54**. Command version is a decimal number (up to 255). When command version is omitted it is assumed to be 1. Earlier firmware releases do not support this parameter and will simply ignore it (result will be the same as having command

version set to 1). When command version of 2 or higher is supplied **Status command** returns an additional ***iii.iii.iii.iii field***.

Example #1: supposing U command returns the following reply**:

-->DS: **U**
DS-->: **A192.168.100.90/37150/0/8192/0/3/8192/5/OFR*8/R*TS**

This means that the data connection is (was/ to be) established with the network host at 192.168.100.90, port number 37150. No data is currently in the Ethernet-to-serial buffer, buffer capacity is 8192 bytes. No committed data is in the serial-to-Ethernet buffer, there are 3 bytes of (uncommitted) data there, and the total capacity is 8192 bytes. The baudrate is 38400, serial port is opened, uses full-duplex mode. RTS/CTS flow control is enabled, parity is set to none, data is 8 bits/byte. RTS, DTR, and DSR lines are in the HIGH* state, CTS line is LOW*.

Example #2: here is a command with version parameter**:

-->DS: **U2**
DS-->: **A192.168.100.90/37150/0/8192/0/3/8192/5/OFR*8/R*TS/161.1.1.110**

In this reply an additional field is present. This field shows that current IP-address on the PPPoE link is 161.1.1.110

When issued through the serial port, the **Status command** returns less data. The primary use of the serial **Status command** is to let the attached serial device inquire the IP-address and port number of the network host with which the connection is (was/ to be) established (as well as current "PPPoE" IP-address). This may be used by the serial device, in conjunction with the modem commands (serial-side parameters and instructions) to control and monitor data connection establishment and termination by the DS.

** HI and LOW states are described with respect to the serial ports of DS100R, DS100B, DS203. For EM100, EM120, EM200, EM203(A) the signaling is exactly opposite.*

*** Encapsulation characters such as IEC, STX, CR, and the command ID field are not shown.*

Buzz (B) command

Description (see command description format info here)

Function:	Makes the status LEDs of the DS play a recognizable pattern
Can be issued through:	Network (broadcasts ignored*, login not required)
Command format:	**B**
Possible replies:	**A**
First introduced:	Earlier than "baseline" V3.14/V3.51
See also:	Status LED signals

Details

Buzz command, when received by the DS, makes the device "play" a recognizable fast-blinking pattern on its status LEDs. This can be used to match an IP-address to a physical DS.

** Without prior selection using Select In Broadcast Mode (W) command.*

Reset Overflow Flags (R) command

Description (see command description format info here)

Function:	Resets routing buffer overflow flags
Can be issued through:	Network (broadcasts ignored*, login not required)
Command format:	**R**
Possible replies:	**A**
First introduced:	Earlier than "baseline" V3.14/V3.51
See also:	Status LED signals, Routing buffers

Details

Reset Overflow Flags command clears routing buffer overflow flags. Buffer overflow condition is returned by the **Echo (X) command** (flags ***E*** and ***S***) and also displayed by the status LEDs of the DS. Overflow flags are cleared automatically when the data connection to the network host is closed or aborted or can also be reset manually using the **Reset Overflow Flags command**.

** Without prior selection using Select In Broadcast Mode (W) command.*

Assign IP-address (A) command

Description (see command description format info here)

Function:	Assigns new IP-address to the DS which is referenced by its MAC-address
Can be issued through:	Network (broadcasts OK, login not required)
Command format:	**A*mmm.mmm.mmm.mmm.mmm.mmm/pp...p/iii.iii.iii.iii***, where ***mmm.mmm.mmm.mmm.mmm.mmm***- MAC-address of the target DS; ***pp...p***- password (defined by the **Password (PW) setting**); ***iii.iii.iii.iii***- new IP-address to be assigned to the DS
Possible replies:	**A**, **D**, **C**, **F**
First introduced:	Earlier than "baseline" V3.14/V3.51
See also:	Authentication, Broadcast out-of-band commands

Details

Assign IP-address command is used to set the new IP-address of a certain DS over the network. Command should be sent in the broadcast mode, the target DS is referenced by its MAC-address (supplied in the command body). All locally attached devices receive the broadcast but only the DS with matching MAC-address reacts to it.

Assign IP-address is a command of immediate authentication typewhich means that the password (defined by the **Password (PW) setting**) is supplied in the

command body itself.

Example: if the MAC-address of the target DS is 0.150.30.213.55.74, the password is "pass1" and the new IP-address is to be 192.168.100.41 then the following command should be sent:

-->DS: **A0.150.30.213.55.74/pass1/192.168.100.41**
DS-->: **A**

Note: password field in this command should be present even if the **Password (PW) setting** is empty (<NULL>):

-->DS: **A0.150.30.213.55.74//192.168.100.41**

New IP-address is saved into the **IP-address (IP) setting**, just as if the **Set Setting (S) command**(i.e. "**SIP*iii.iii.iii.iii***") was executed. Differences with the **Set Setting (S) command** are in that the DS starts using the new IP-address immediately (no rebooting required) and that the target DS is referenced by its MAC-address.

Rejected (R) reply code is returned if the serial programming mode is in progress. **Denied (D) reply code** is returned if the password is incorrect. **Error (C) reply code** is returned if command structure is incorrect (a field or field separator is missing) or if the field data is wrong. **Failed (F) reply code** is returned when the DS failed to write new IP-address into the EEPROM. This usually indicates a hardware malfunction (EEPROM failure). Since this is a broadcast command no reply is returned if no DS on the network has the MAC-address specified in the command.

When the **Assign IP-address command** is issued while the DS has a data connection in progress this data connection is aborted. No packet (even RST in case of TCP data connection) is sent to the network host that was communicating with the DS.

Select In Broadcast Mode (W) command

Description (see command description format info here)

Function:	Selects the DS as the target in broadcast out-of-band UDP programming
Can be issued through:	Network (broadcasts OK, login not required)
Command format:	**W*mmm.mmm.mmm.mmm.mmm.mmm***, where ***mmm.mmm.mmm.mmm.mmm.mmm***-MAC-address of the target DS
Possible replies:	**A**
First introduced:	Earlier than "baseline" V3.14/V3.51
See also:	Broadcast out-of-band commands

Details

Select In Broadcast Mode command is used to pre-select a certain DS for subsequent programming via broadcast out-of-band (UDP) commands. Only a small portion of DS commands (such as **Echo (X)**) are accepted when sent in broadcast UDP datagrams. All other commands are only accepted if they address a specific DS. Such specific addressing normally involves sending UDP datagrams with the IP-address of the targeted DS as the destination (i.e. non-broadcast datagrams). This requires the IP-address of the DS to be configured reachable which is not always possible or convenient.

Select In Broadcast Mode command provides a way around this. Target DS, referenced by its MAC-address, is first pre-selected using this command. After that, all broadcast commands that are normally ignored when sent as broadcasts, are not ignored and processed by this pre-selected DS.

When **Select In Broadcast Mode command** is issued all devices whose MAC-addresses do not match the target MAC-address supplied in the command body de-select themselves. This means that to switch onto programming of another DS in the broadcast mode, you need to send the new **Select In Broadcast Mode command** with the new target MAC-address. This will pre-select a different DS while at the same time de-selecting the DS that was selected before. To de-select all DS on the network send **Select In Broadcast Mode command** with no MAC-address field.

This command only influences which DS responds when it is addressed using broadcast UDP commands. Command has no influence over any other form of programming that involves addressing the DS by its IP-address.

The only possible reply to this command is **OK (A)**. It is issued by the DS that has recognised its MAC-address in the command body. If no DS on the local network recognizes its MAC then there will be no reply received to this command.

Get Firmware Version (V) command

Description (see command description format info here)

Function:	Returns firmware version of this firmware
Can be issued through:	Network (broadcasts OK, login not required); serial port
Command format:	**V**
Possible replies:	**A*vv...v***, where ***vv...v*** is the version string
First introduced:	Earlier than "baseline" V3.14/V3.51
See also:	---

Details

Get Firmware Version command returns current firmware version string.

The version string is always encapsulated in '<' and '>', begins with the version number in **V*x.xx*** format, and possibly contains a small comment after a space.

Example:

-->DS: **V**
DS-->: **A<V3.20 R3 final>**

Jump To Netloader (N) command [Release3.0]

Description (see command description format info here)

Function:	Jumps to (launches) the NetLoader
Can be issued through:	Network (broadcasts ignored*, login required). UDP only, TCP/Telnet not supported.
Command format:	**N**

Possible replies:	**A, D, F**
First introduced:	Earlier than "baseline" V3.14, **not supported by Release3.5 firmware branch**
See also:	NetLoader

Details

This command is only used on first-generation Devices (i.e. it is implemented on Release3 firmware only).

Jump To NetLoader command verifies NetLoaderpresence and integrity, then launches it. NetLoader integrity is verified by calculating the checksum on its code and comparing this checksum with the stored one. The NetLoader is not launched if the checksum is found to be invalid.

This command requires prior login using the **Login (L) command** (programming session must be opened). **Denied (D) reply code** is returned if the programming session is not in progress or if it doesn't belong to the sender of the **Jump To NetLoader command**- the DS remembers the IP-address of the network host that opens the programming session and requires that all subsequent commands (that require prior login) are sent from the same IP.

Failed (F) reply code is returned if the NetLoader is found to be corrupted.

** Without prior selection using Select In Broadcast Mode (W) command.*

Set Programming Request Flag (N) command [Release 3.5]

Description (see command description format info here)

Function:	Sets a flag that will make the DS upgrade its firmware after the reboot
Can be issued through:	Network (broadcasts ignored*, login required). UDP only, TCP/Telnet not supported.
Command format:	**N**
Possible replies:	**A, D, F**
First introduced:	Earlier than "baseline" V3.51, **not supported by Release3 firmware branch**
See also:	NetLoader

Details

This command is only used on second-generation Devices (i.e. it is implemented on Release3.5 firmware only).

Set Programming Request Flag command programs a special word of data into the FLASH memory of the DS. When the DS reboots and the Monitor gains control it verifies the state of the flag. If the flag is set the Monitor copies new firmware file from the *data FLASH memory* of the DS into the *program FLASH memory* of the DS. After the programming is finished the Monitor resets the flag automatically. Before attempting to write into the program FLASH memory the Monitor verifies integrity of the data in the data FLASH. If the data is found to be corrupted the Monitor aborts the programming.

Set Programming Request Flag command is supposed to be used after the new application firmware file is uploaded into the data FLASH memory of the DS using

Reset Upload Process (Q) and **Upload Data Block (D)** commands. Once the flag is set the DS should be rebooted with **Reboot (E) command** (to start FLASH copying process).

This command requires prior login using the **Login (L) command** (programming session must be opened). **Denied (D) reply code** is returned if the programming session is not in progress or if it doesn't belong to the sender of the command- the DS remembers the IP-address of the network host that opens the programming session and requires that all subsequent commands (that require prior login) are sent from the same IP.

Failed (F) reply code is returned if the flag could not be set (written into the FLASH memory).

** Without prior selection using Select In Broadcast Mode (W) command.*

Reset Upload Process (Q) command [Release 3.5]

Description (see command description format info here)

Function:	Reset application firmware upload into the data FLASH memory of the DS
Can be issued through:	Network (broadcasts ignored*, login required). UDP only, TCP/Telnet not supported.
Command format:	**Q**
Possible replies:	**A, D**
First introduced:	Earlier than "baseline" V3.51, **not supported by Release3 firmware branch**
See also:	NetLoader

Details

This command is only used on second-generation Devices (i.e. it is implemented on Release3.5 firmware only).

Reset Upload Process command initializes application firmware file upload into the data FLASH memory of the DS. This command should always be used before upload itself, which is performed with **Upload Data Block (D) command**. To make the DS upgrade the contents of its program FLASH (i.e. copy the application from the data FLASH into the program FLASH) two other commands should be used: **Set Programming Request Flag (N)**, followed by **Reboot (E)**.

This command requires prior login using the **Login (L) command** (programming session must be opened). **Denied (D) reply code** is returned if the programming session is not in progress or if it doesn't belong to the sender of the command- the DS remembers the IP-address of the network host that opens the programming session and requires that all subsequent commands (that require prior login) are sent from the same IP.

** Without prior selection using Select In Broadcast Mode (W) command.*

Upload Data Block (D) command [Release 3.5]

Description (see command description format info here)

Function:	Uploads a 128-byte data block into the data FLASH memory of the DS
Can be issued through:	Network (broadcasts ignored*, login required). UDP only, TCP/Telnet not supported.
Command format:	**D*nnddd..dd***, where ***nn***: data block number in *binary format* (exactly 2 bytes), ***ddd..dd***- 128 bytes of data in *binary format*
Possible replies:	**A, D, C, S, 0, F**
First introduced:	Earlier than "baseline" V3.51, **not supported by Release3 firmware branch**
See also:	NetLoader

Details

This command is only used on second-generation Devices (i.e. it is implemented on Release3.5 firmware only).

Upload Data Block command sends a 128-byte block of data to the data FLASH memory of the DS. This command is used for application firmware upgrades through the network.

Upgrade starts with **Reset Upload Process (Q) command**. After that, **Upload Data Block** is used necessary number of times until entire application firmware file is uploaded block by block. The first command sent should have its ***nn*** field set to 0x00 0x00, next- 0x00 0x01, etc. Each command should supply exactly 128 bytes of data. If the file cannot be split into N full 128-byte blocks the last block should be padded with any data. Once entire file has been uploaded two additional commands should be used: **Set Programming Request Flag (N)**, followed by **Reboot (E)**. After the DS emerges from reset the Monitor will copy new firmware file from the data FLASH into the program FLASH.

This command requires prior login using the **Login (L) command** (programming session must be opened). **Denied (D) reply code** is returned if the programming session is not in progress or if it doesn't belong to the sender of the command- the DS remembers the IP-address of the network host that opens the programming session and requires that all subsequent commands (that require prior login) are sent from the same IP.

Error (C) reply code is returned if command length wasn't *exactly* 131 byte in length (command code + 2-byte block number + 128 bytes of data). **Bad Sequence (S) reply code** is issued if data blocks were not consecutive (for example, after block 3 came block 5). The DS replies with **Out-of-range (O) reply code** if file size has exceeded data FLASH capacity. **Failed (F) reply code** is returned if there was an error writing into the data FLASH. **OK (A) reply code** is returned when the block is received properly and all is right. This code is always followed by next block number -- 2 bytes in network byte order (High endian format).

Upload Data Block command is different from all other commands in that its fields are of binary type (all other commands are ASCII strings).

** Without prior selection using Select In Broadcast Mode (W) command.*

Cable Status (C) command

Description (see command description format info here)

Function:	Returns network cable status
Can be issued through:	Serial port
Command format:	**C**
Possible replies:	**AC**, **AD**
First introduced:	3.26, **not supported by 3.5 branch**
See also:	Status (U) command, Serial Programming

Details

Get Cable Status command returns current status of the network cable.

The return value **AC** means that the network cable is currently plugged in. The value **AD** means that the network cable is disconnected. Note that this does not indicate actual network connection status (see Status (U) command).

Get My IP (T) command

Description (see command description format info here)

Function:	Returns the IP address of the sender of this command
Can be issued through:	Network (broadcasts OK, login not required)
Command format:	+
Possible replies:	**A*ddd.ddd.ddd.ddd***, where ***ddd.ddd.ddd.ddd*** is the IP address of the sender of this command
First introduced:	**V3.32/3.63**
See also:	---

Details

Get My IP command is used to determine under which IP address the DS sees the sender of the command. This is not necessarily the same as the actual IP address of the sender (for example, there might be a router between the DS and the PC). The IP address returned by the command can then be used, for instance, to set the **Destination IP address (DI)** of the DS.

Note that the fact that the PC (or another command sender) can "reach" the DS does not automatically mean that the DS can "reach" (connect to) the PC. Network setups are not always symmetrical!

You can remember this command's mnemonic as the first character of "**T**ell me who I am".

Notification (J) message

Description (see command description format info here)

Function:	Reports I/O pin status change

Send through:	Network
Format:	***Jsss***, where ***sss*** is the byte value in the 0-255 range containing the status of all I/O lines of the DS
First introduced:	Earlier than "baseline" V3.14/V3.51
See also:	Serial port and serial communications

Details

Notification message in not a command, it is a message that the DS sends to the network host when one of the monitored I/O lines of the DS changes its state (for longer than 20milliseconds). The status of *all* lines is rolled into a single byte of data and sent out even when a *single* I/O line changes its state.

Which I/O lines are being monitored is defined by the **current Notification Bitmask (NB) [setting/ parameter]**.

Notification messages are only generated when the data connection is in progress and are sent to the network host with which this data connection is established. **Notification Destination (ND) setting** defines which port on this network host notifications are sent to.

How notifications are sent (out-of-band, inband, etc.) is defined by several factors:

- If the **Data Login (DL) setting** is 1 (enabled), **current Transport Protocol (TP) [setting/ parameter]** is 1 (TCP), and the data TCP connection is in the command phase then notifications are sent via this TCP connection (IEC character is not used). Otherwise...
- If the **Inband (IB) setting** is 1 (enabled), and the **current Transport Protocol (TP)** is 1 (TCP) then notifications are sent inside this TCP connection as inband messages (IEC character is used). Otherwise...
- Notifications are sent as out-of-band UDP datagrams.

The value in each **Notification message** should be interpreted as a collection of binary bits, with each position corresponding to a certain I/O line of the DS. Bit positions are exactly the same as those of the **Notification Bitmask (NB) Setting**. Bit values correspond to the states of the I/O lines of Modules. Line states on the RS232 connectors of Serial Device Servers and Boards that incorporate RS232 transceivers are inverted relative to the states reported in the **notification message**.

Example: supposing, the following Notification message is sent*:

J027

Decimal 27 converts to binary 00011011. This means that:

- For devices such as EM100: P2/DSR and P5/RTS lines are LOW; P0, P1, P2/DSR, and P4/CTS lines are HIGH;
- For devices such as DS100: DSR and RTS lines are HIGH; DTR and CTS lines are LOW.

Notification messages are not commands so they do not require any reply from the receiving end.

If it is the DS that receives a **Notification message** from another DS, then the following happens:

- If **current Flow Control (FC) [setting/ parameter]** on the receiving DS is 0 (disabled) and **current Serial Interface** is full-duplex then this DS will set its

RTS line according to the value of CTS bit (bit 4) supplied by the **Notification message**; otherwise the status of the RTS line will not be changed.

- If the **DTR Mode (DT) setting** on the receiving DS is 0 (idle) then this DS will set its DTR line according to the value of DSR bit (bit 2) supplied by the **Notification message**; otherwise the status of the DTR line will not be changed.

The above means that the **Notification message** links RTS-CTS and DTR-DSR signals on two communicating DS: when the CTS input on one end changes its status the RTS output on the other end changes its status accordingly (same with the DTR-DSR pair).

** Encapsulation characters such as IEC, STX, CR are not shown.*

Settings.2.4.2

This section contains a reference for all DS settings.

Settings are *permanent functioning parameters* that are stored in the non-volatile memory (EEPROM) of the DS. Once programmed, they remain intact even when the DS is powered off.

Setting description format can be found here.

All settings are divided into four groups:

- **Network settings** include basic set of parameters that define "networking environment" of the DS. For more information see Ethernet port and network communications.
- **Connection Settings** define how and in which fashion the DS establishes connections to and accepts connections from other hosts. For more information see Ethernet port and network communications
- **Serial settings** define the operation of the DS serial port. For more information see serial port and serial communications.
- **Encapsulation settings** define what incoming serial data is recorded into the serial-to-Ethernet routing buffer and when and how this data is combined into the network packets and sent to the network host. For more information see serial-to-Ethernet data routing.

Setting description format

All settings in this section are described using the following format:

Function:	Setting function in brief
Set (S) command format:	Syntax of the corresponding **Set (S) command** that is used to set new setting value
Get (G) command format:	Syntax of the corresponding **Get (S) command** that is used to read out current value of the setting
Init (I) command effect:	Explains under what additional conditions the setting is initialized when the **Initialize (I) command** is issued or quick initialization is launched. Some settings are always initialized, some are initialized only when invalid, etc.
Post-initialization value:	Shows factory initialization value that will be

	assigned to the setting after the initialization (factory initialization values may be overridden by the custom profile)
Change takes effect:	Explains when the new setting value takes effect. Changes to some settings have immediate effect, for some settings rebooting is required, etc.
Overriding parameter:	Certain settings have corresponding overriding parameters that can be supplied through the **Parameter (P) command**
Relevance conditions:	Some settings are relevant to the operation of the DS only when other settings have certain values
First introduced:	Describes whether this setting has been available right from the "baseline" firmware version of 3.14/3.51 or was introduced in a later firmware release
See also:	Additional relevant links

Details

Additional information about the setting.

Network Settings

Network settings include basic set of parameters that define "networking environment" of the DS. For more information see Ethernet port and network communications.

The following settings belong to this group:

Setting	Description
Owner Name (ON) setting	Defines the owner name identificator for the DS
Device Name (DN) setting	Defines the device name identificator for the DS
MAC-address (FE) setting	Defines MAC-address of the DS
DHCP (DH) setting	Enables/disables DHCP for the DS
IP-address (IP) setting	Defines the IP-address of the DS
Port Number (PN) setting	Defines the data port number of the DS
dDNS Service Registration (DD) setting [V3.24/3.54+]	Defines whether the DS will register its IP-address with dDNS Service of Link Server at startup
dDNS Service IP-address (LI) setting [V3.24/3.54+]	IP-address for dDNS registration
dDNS Service Port (LP) setting [V3.24/3.54+]	Port number for dDNS registration
PPPoE Mode (PP) setting [V3.54+]	Defines whether and when the DS will use PPPoE
LS Auto-registration (AR) setting [V3.24/3.54+]	Defines whether, if rejected by the Link Server, the DS will try to auto-register
PPPoE Login Name (PL) setting [V3.54+]	Defines login name for PPPoE Access Concentrator

PPPoE Login Password (PD) setting [V3.54+]	Defines login password for PPPoE Access Concentrator
Gateway IP-address (GI) setting	Defines the IP-address of the default gateway
Netmask (NM) setting	Defines the IP-address range for the local subnet
Password (PW) setting	Defines login password network programming

Owner Name (ON) setting

Description (see setting description format info here)

Function: Defines the owner name identificator for the DS

Set (S) command format: **SON*oo...o***, where ***oo...o*** is the name string, 0-8 characters long

Get (G) command format: **GON**

Init (I) command effect: Only initialized if invalid, through network command, serial command, or quick initialization

Post-initialization value: <NULL>

Change takes effect: Immediately

Overriding parameter: **---**

Relevance conditions: **---**

First introduced: Earlier than "baseline" V3.14/V3.51

See also: ---

Details

This setting, together with the **Device Name (DN) setting** forms a name identificator for the DS. Owner name and device name are returned by the **Echo (X) command**.

Owner Name also serves two other purposes:

- It is used to form device name supplied to the DHCP server
- It is also used for logins onto the Link Server

Device Name (DN) setting

Description (see setting description format info here)

Function: Defines the device name identificator for the DS

Set (S) command format: **SDN*dd...d***, where ***dd...d*** is the name string, 0-8 characters long

Get (G) command format: **GDN**

Init (I) command effect: Only initialized if invalid, through network command, serial command, or quick initialization

Post-initialization value: <NULL>

Change takes effect: Immediately

Overriding parameter: ---

Relevance conditions: ---

First introduced: Earlier than "baseline" V3.14/V3.51

See also: ---

Details

This setting, together with **Owner Name (ON) setting** forms a string identificator for the DS. Owner name and Device name are returned by the **Echo (X) command**.

Device Name also serves two other purposes:

- It is used to form device name supplied to the DHCP server
- It is also used for logins onto the Link Server

MAC-address (FE) setting

Description (see setting description format info here)

Function: Defines MAC-address of the DS

Set (S) command format: **SFE*xxx.xxx.xxx.xxx.xxx.xxx***, where ***xxx.xxx.xxx.xxx.xxx.xxx*** is the MAC-address in the dot-decimal notation (i.e. 0.2.3.4.120.240)

Get (G) command format: **GFE**

Init (I) command effect: Only initialized if invalid, through network command, serial command, or quick initialization

Post-initialization value: 0.1.2.3.4.5

Change takes effect: After reboot

Overriding parameter: ---

Relevance conditions: ---

First introduced: Earlier than "baseline" V3.14/V3.51

See also: Ethernet port and network communications

Details

Each DS is shipped from the factory with unique **MAC-address** already assigned to it. **DO NOT** change this address unless you have a good reason to do so. If you do change the address remember that the first digit of the address must be even!

Since the **MAC-address** of each DS is unique it may be used for device identification. It is returned by the **Echo (X) command** and also used to address a particular DS in the **Assign IP-address (A) command** and **Select in Broadcast Mode (W) command**.

This setting's mnemonic- "FE"- has to do with the previous name of the setting- "Factory Ethernet Address". Mnemonic was preserved to ensure compatibility with previous firmware versions.

DHCP (DH) setting

Description (see setting description format info here)

Function:	Enables/disables DHCP for the DS
Set (S) command format:	**SDH*x***, where ***x***: 0 (disabled), 1 (enabled)
Get (G) command format:	**GDH**
Init (I) command effect:	Initialized unconditionally, through network command, serial command, or quick initialization
Post-initialization value:	0 (disabled)
Change takes effect:	After reboot
Overriding parameter:	**---**
Relevance conditions:	**---**
First introduced:	Earlier than "baseline" V3.14/V3.51
See also:	DHCP

Details

This setting defines whether the DS will use a fixed IP-address, defined by the **IP-address (IP) setting** or will obtain its IP-address from the DHCP server at powerup. DHCP server must be present on the network for this to work. The DS will not start normal operation until it receives the IP-address.

IP-address obtained from the DHCP server is saved into the **IP-address (IP) setting** thus overwriting the previous setting value that might have been set manually.

IP-address status (obtained/ not obtained) of the DS is returned by the **Echo (X) command** and also displayed by status LEDs of the DS.

In addition to the IP-address configuration, the DHCP server is *usually* configured to provide default Gateway IP-address and netmask. If this is the case, then **Gateway IP-address (GI)** and **Netmask (NM)** settings will also be overwritten by the data from the DHCP server.

IP-address (IP) setting

Description (see setting description format info here)

Function:	Defines the IP-address of the DS
Set (S) command format:	**SIP*xxx.xxx.xxx.xxx***, where ***xxx.xxx.xxx.xxx*** is the IP-address in dot-decimal notation (i.e. 192.168.100.40)
Get (G) command format:	**GIP**
Init (I) command effect:	Initialized unconditionally through serial command or quick initialization, initialized only if invalid through network command
Post-initialization value:	1.0.0.1 (changed from 0.0.0.1 in **V3.34/3.66**)
Change takes effect:	After reboot
Overriding condition:	**---**

Relevance conditions: DHCP (DH) setting=0 (disabled)

First introduced: Earlier than "baseline" V3.14/V3.51

See also: Ethernet port and network communications

Details

IP-address must be compatible with the network on which the DS is installed. Many networks have DHCP server, in this case it is better to make the DS obtain the **IP-address** automatically on startup (this is enabled by programming the **DHCP (DH) setting** to 1 (enabled)).

When DHCP is activated the **IP-address** obtained from the DHCP server is saved into the **IP-address setting** thus overwriting older value that might have been set before.

Some **IP-addresses** are not valid in principle. Many devices and operating systems (including Windows) automatically discard network packets that refer to such incorrect IPs. The DS will allow such an **IP-address** to be saved into the EEPROM but will assume a modified address on startup:

Invalid IP-address	IP-address that the DS will actually use
x.x.x.0	x.x.x.1
x.x.x.255	x.x.x.1
>223.x.x.x	223.x.x.x

Example: if the **IP-address** is 224.168.100.255 then the DS will actually use 223.168.100.1. The EEPROM data will not be modified and **GIP** command will still return original data (224.168.100.255) but the actual **IP-address** used by the DS will be corrected according to the above rules.

The post-initialization value of this setting is 1.0.0.1. It used to be 0.0.0.1, and before that it used to be 127.0.0.1. All these changes were caused by increasing number of restrictions in *Windows* TCP stack and/or firewall. When *Windows Vista* was released it turned out that the DS Manager could not "see" Device Servers with the default IP address of 0.0.0.1. We have changed the IP to 1.0.0.1 and now everything works fine under *Vista*. Previous change from 127.0.0.1 to 0.0.0.1 was caused by the fact that *Windows XP* did not "approve" of 127.0.0.1 (earlier *Windows* versions did not have a problem with this IP).

Port Number (PN) setting

Description (see setting description format info here)

Function: Defines the data port number of the DS

Set (S) command format: **SPN*ppppp***, where ***ppppp*** is the port number in the 0-65534 range. Also, port 32767 cannot be used when **current Transport Protocol (TP)[setting/parameter]** is 0(UDP). Additionally, for **firmware 3.5x and above**, port 23 cannot be used when current Transport Protocol is 1(TCP)

Get (G) command format: **GPN**

Init (I) command effect: Initialized unconditionally, through network

	command, serial command, or quick initialization
Post-initialization value:	1001
Change takes effect:	After reboot
Overriding parameter:	**---**
Relevance conditions:	**---**
First introduced:	Earlier than "baseline" V3.14/V3.51
See also:	Ethernet port and network communications, Network programming

Details

This setting defines the *Data Port* on which the DS is accepting incoming data connections.

For UDP communications, outgoing UDP datagrams are also sent from this port. For TCP communications the DS accepts incoming data connections on the data port but establishes outgoing connections from a pool of ephemeral ports in the 10000-10255 range (port number is incremented with each new connection).

Port 65535 is excluded from the allowable range because port 65535 is a programming port of the DS- command UDP datagrams are sent to this port. Technically, this only affects UDP communications, TCP connections should still be able to use this port but the port 65535 was not allowed to be selected as the data port from the early versions of DS firmware so this restriction is now preserved for "historical" reasons.

Since programming UDP datagrams can now also be sent to port 32767, this port cannot be used for UDP data communications when **current Transport Protocol (TP)[setting/parameter]** is 0 (UDP). The DS will allow the **Port Number** to be programmed to 32767 but all data sent to this port in the UDP mode will be interpreted as programming commands. Using port 32767 for TCP communications won't cause any problems.

New telnet programming method introduced in firmware V3.5x uses telnet port 23 for DS programming as well. Any connection established to port 23 of the DS is interpreted by the DS as a programming connection. Therefore, this port cannot be used for data connections when the **current Transport Protocol (TP)** is 1(TCP). Older firmware does not use port 23 for programming so this restriction does not apply.

dDNS Service Registration (DD) setting

Description (see setting description format info here)

Function:	Defines whether the DS will register its IP-address with dDNS Service of the Link Server at powerup
Set (S) command format:	**SDD*x***, where ***x***: 0 (disabled), 1 (enabled)
Get (G) command format:	**GDD**
Init (I) command effect:	Initialized unconditionally, through network command, serial command, or quick initialization
Post-initialization value:	0 (disabled)
Change takes effect:	After reboot
Overriding parameter:	**---**

Relevance conditions: ---

First introduced: V3.24/3.54

See also: Ethernet port and network communications

Details

This setting defines whether the DS will register its IP-address with dDNS Service of the Link Server at powerup. For more information on dDNS see Link Server Documentation.

dDNS Service registration involves login onto the Link Server and uses the data from the following settings of the DS: **Owner Name (ON)**, **Device Name (DN)**, and **Password (PW) setting**). The password is supplied in the encrypted form so the registration process is secure.

dDNS Service IP-address (LI) setting

Description (see setting description format info here)

Function: IP-address for dDNS registration

Set (S) command format: **SLI*xxx.xxx.xxx.xxx***, where ***xxx.xxx.xxx.xxx*** is the IP-address in dot-decimal notation (i.e. 192.168.100.40)

Get (G) command format: **GLI**

Init (I) command effect: Initialized unconditionally, through network command, serial command, or quick initialization

Post-initialization value: 127.0.0.1

Change takes effect: After reboot

Overriding condition: ---

Relevance conditions: **dDNS Service Registration (DD) setting**= 1 (enabled)

First introduced: V3.24/3.54

See also: Ethernet port and network communications

Details

This setting defines the IP-address to which the DS will try to connect in order to register its IP-address with dDNS Service. For more information on dDNS see Link Server Documentation.

dDNS Service IP-address is irrelevant when **dDNS Service Registration (DD) setting**= 0 (disabled).

dDNS Service Port (LP) setting

Description (see setting description format info here)

Function: Port number for dDNS registration

Set (S) command format: **SLP*ppppp***, where ***ppppp*** is the port number in the 0-65535 range

Get (G) command format:	**GLP**
Init (I) command effect:	Initialized unconditionally, through network command, serial command, or quick initialization
Post-initialization value:	6450
Change takes effect:	After reboot
Overriding parameter:	**---**
Relevance conditions:	**dDNS Service Registration (DD) setting**= 1 (enabled)
First introduced:	**V3.24/3.54**
See also:	Ethernet port and network communications

Details

This setting defines the port number to which the DS will try to connect in order to register its IP-address with dDNS Service. For more information on dDNS see Link Server Documentation.

dDNS Service Port is irrelevant when **dDNS Service Registration (DD) setting**= 0 (disabled).

LS Auto-registration (AR) setting

Description (see setting description format info here)

Function:	Defines whether, if rejected by the Link Server the DS will attempt to auto-register
Set (S) command format:	**SAR*x***, where ***x***: 0 (disabled), 1 (enabled)
Get (G) command format:	**GAR**
Init (I) command effect:	Initialized unconditionally, through network command, serial command, or quick initialization
Post-initialization value:	0 (disabled)
Change takes effect:	After reboot
Overriding parameter:	**---**
Relevance conditions:	**dDNS Service Registration (DD) setting= 1** (enabled) **OR** **Current Link Service Login (TL) [setting/parameter]**= 1 (enabled)
First introduced:	**V3.24/3.54**
See also:	Ethernet port and network communications

Details

The DS logs onto the Link Server in two cases: when it needs to register its IP-address with the dDNS Service (**dDNS Service Registration (DD) setting= 1** (enabled)), or when the DS needs to communicate through the Link Service (**current Link Service Login (TL) [setting/parameter]**= 1 (enabled)).

When **LS Auto-registration** is set to 1 (enabled) the DS will attempt to register on the Link Server if, during login process, the DS is rejected and the reason for

this rejection is that this DS is not yet registered (i.e. Link Server does not recognize **Owner Name (ON)** and **Device Name (DN)**of this DS). Auto-registration is convenient because it allows the user to avoid manual editing of the list of client Device Servers on the Link Server.

WARNING! Auto-registration process involves sending DS password (defined by the Password (PW) setting) to the Link Server. In this particular case the password is sent "unprotected" (as is) which constitutes a potential security vulnerability.

PPPoE Mode (PP) setting [V3.54+]

Description (see setting description format info here)

Function:	Defines whether and when the DS will use PPPoE
Set (S) command format:	**SPP*x***, where ***x***: 0 (disabled), 1 (on connection), 2 (on powerup)
Get (G) command format:	**GPP**
Init (I) command effect:	Initialized unconditionally, through network command, serial command, or quick initialization
Post-initialization value:	0 (disabled)
Change takes effect:	After reboot
Overriding parameter:	**---**
Relevance conditions:	**---**
First introduced:	**V3.54**
See also:	PPPoE

Details

This setting enables/disables PPPoE and also defines when, if at all, PPPoE login will be performed. **PPPoE Mode** only affects communications with network hosts located on remote network segments (local segment boundaries are defined by the **Netmask (NM) setting**). When PPPoE is used all communications with remote network hosts are effected through PPPoE "channel".

The setting offers three options:

0 (disabled)	PPPoE is not used. When the DS needs to establish an outgoing connection to remote network host it sends the data to a default gateway as defined by the **Gateway IP-address (GI) setting**.
1 (on connection)	PPPoE is enabled. PPPoE login is performed when the DS needs to establish a data connection to the remote network host. All communications with remote hosts are effected through PPPoE link (Access Concentrator), so default gateway is not used in any way. PPPoE link is terminated app. 30 seconds after the data connection is closed.
2 (on powerup)	PPPoE is enabled. PPPoE login is performed at startup, after IP-address configuration is completed in case **DHCP (DH) setting** is 1 (enabled). The DS attempts to maintain PPPoE link at all times. If the DS detects that the PPPoE link is broken it reestablishes this link. All communications with remote hosts are effected through this PPPoE link (Access

Concentrator), so default gateway is not used in any way.

PPPoE authentication uses PAP protocol (this is the only authentication protocol currently supported). Login name and password for PPPoE Access Concentrator are defined by **PPPoE Login Name (PL)** and **PPPoE Login Password (PD)** settings.

Notice that PPPoE is only available in firmware **V3.54+**. This means that first-generation Devices (EM100-00/ -01/ -02, DS100-00/ -01/ -02) do not support PPPoE.

PPPoE Login Name (PL) setting [V3.54+]

Description (see setting description format info here)

Function:	Defines login name for PPPoE Access Concentrator
Set (S) command format:	**SPL*nn...n***, where ***nn...n*** is login name (0-20 characters long)
Get (G) command format:	**GPL**
Init (I) command effect:	Initialized unconditionally, through network command, serial command, or quick initialization
Post-initialization value:	<NULL>
Change takes effect:	After reboot
Overriding parameter:	**---**
Relevance conditions:	**PPPoE Mode (PP) setting**= 1 (on connection) or 2 (on powerup)
First introduced:	**V3.54**
See also:	PPPoE

Details

This setting defines login name for PPPoE Access Concentrator. Login name can be up to 20 characters long.

This setting is irrelevant when **PPPoE Mode (PP) setting** is 0 (disabled).

Notice that PPPoE is only available in firmware **V3.54+**. This means that first-generation Devices (EM100-00/ -01/ -02, DS100-00/ -01/ -02) do not support PPPoE.

PPPoE Login Password (PD) setting [V3.54+]

Description (see setting description format info here)

Function:	Defines login password for PPPoE Access Concentrator
Set (S) command format:	**SPD*pp...p***, where ***pp...p*** is login password (0-20 characters long)
Get (G) command format:	**GPD**
Init (I) command effect:	Initialized unconditionally, through network command, serial command, or quick initialization
Post-initialization value:	<NULL>

Change takes effect: After reboot

Overriding parameter: ---

Relevance conditions: **PPPoE Mode (PP) setting**= 1 (on connection) or 2 (on powerup)

First introduced: **V3.54**

See also: PPPoE

Details

This setting defines login password for PPPoE Access Concentrator. Login name can be up to 20 characters long.

This setting is irrelevant when **PPPoE Mode (PP) setting** is 0 (disabled).

Notice that PPPoE is only available in firmware **V3.54+**. This means that first-generation Devices (EM100-00/ -01/ -02, DS100-00/ -01/ -02) do not support PPPoE.

Gateway IP-address (GI) setting

Description (see Setting description format info here)

Function: Defines the IP-address of the default gateway

Set (S) command format: **SGI*xxx.xxx.xxx.xxx***, where ***xxx.xxx.xxx.xxx*** is the IP-address of the default gateway in dot-decimal notation (i.e. 192.168.100.1)

Get (G) command format: **GGI**

Init (I) command effect: Initialized unconditionally, through network command, serial command, or quick initialization

Post-initialization value: 0.0.0.1 (changed from 127.0.0.1 in **V3.30/3.62**)

Change takes effect: After reboot

Overriding condition: *May* be automatically updated by the gateway IP-address provided by the DHCP server (in case **DHCP (DH) setting** is 1 (enabled))

Relevance conditions: **Current Routing Mode (RM) [setting/ parameter]**= 1 (server/client) or 2 (client) **AND [V3.54+] PPPoE Mode (PP) setting**= 0 (disabled)

First introduced: Earlier than "baseline" V3.14/V3.51

See also: Ethernet port and network communications

Details

Gateway IP-address defines the IP-address of default gateway through which the DS will (attempt to) establish a connection to the destination network host at **current Destination IP-address (DI) [setting/ parameter/ instruction]** in case this host is not on the same subnet with the DS.

Whether or not the destination network host is on the local subnet is determined by comparing the **IP-address (IP) setting**, **current Destination IP-address (DI)**, and the **Netmask (NM) setting** (see this setting's description for details).

Gateway IP-address is irrelevant when the **current Routing Mode (RM) [**

setting/ parameter] is 0 (server) since in this mode outgoing connections are not allowed. **[V3.54+]** This setting is also irrelevant when PPPoE is used i.e. **PPPoE Mode (PP) setting** is 1 (on connection) or 2 (on powerup). This is because with PPPoE all communications with remote hosts go through PPPoE link (Access Concentrator), not default gateway.

When DHCP is activated (**DHCP (DH) setting** is 1(enabled)) the Gateway IP-address obtained from the DHCP server is saved into this setting thus overwriting older value that might have been set before. This only happens when the DHCP server is configured to provide gateway IP-address data.

Netmask (NM) setting

Description (see setting description format info here)

Function:	Defines the IP-address range for the local subnet
Set (S) command format:	**SNM*nn...n***, where ***nn...n*** is the netmask for the local subnet in dot-decimal notation (i.e. 255.255.255.0)
Get (G) command format:	**GNM**
Init (I) command effect:	Initialized unconditionally, through network command, serial command, or quick initialization
Post-initialization value:	0.0.0.0
Change takes effect:	After reboot
Overriding condition:	May be automatically updated by the netmask provided by the DHCP server (in case **DHCP (DH) setting** is 1 (enabled))
Relevance conditions:	**Current Routing Mode (RM) [setting/ parameter]**= 1 (server/client) or 2 (client)
First introduced:	Earlier than "baseline" V3.14/V3.51
See also:	Ethernet port and network communications

Details

Netmask defines the boundaries of a local subnet. When establishing an outgoing connection to the destination network host at **current Destination IP-address (DI) [setting/ parameter/ instruction]** the DS compares this address with the **Netmask** and its own **IP-address (IP)** to determine if the destination is on the local or foreign subnet. The DS will (attempt to) connect directly to the **current Destination IP-address (DI)** if the destination host is found to reside on the local subnet or to the **Gateway IP-address (GI)** if the destination is found to reside on a foreign subnet.

Comparison is done as follows:

- All four bytes of the **IP-address (IP)** are ANDed with four bytes of the **Netmask**.
- All four bytes of the **current Destination IP-address (DI)** are ANDed with four bytes of the **Netmask**.
- Results of previous two steps are compared: if they are equal then the destination is on the same subnet with the DS, if different- the destination is on the foreign subnet and the DS will be connecting through the gateway.

Example: supposing, the **IP-address (IP)** of the DS is 192.168.100.40 (C0.A8.64.28 in HEX representation), **current Destination IP-address (DI)** is

192.168.100.90 (C0.A8.64.5A in HEX) and the **Netmask** is 255.255.255.0 (FF.FF.FF.00 in HEX). Then:

- C0.A8.64.28 AND FF.FF.FF.00 will result in C0.A8.64.00
- C0.A8.64.5A AND FF.FF.FF.00 will result in C0.A8.64.00
- Resulting numbers are the same so the destination is on the same subnet

Here is another way of explaining how the Netmask works. When printed in binary representation, the Netmask always consists of a number of 1s on the left and the number of 0s on the right (for the example above the Netmask value is 1111111. 11111111. 11111111. 00000000). Positions with 1s (left side) represent the part in which the **current Destination IP-address (DI)** must match the **IP-address (IP)** of the DS to be considered local. Positions with 0s (right side) represent the range of IP-addresses belonging to the same subnet. If the Netmask is 255.255.255.0 then any IP-address that starts with 255.255.255 will be on the same subnet.

Netmask is irrelevant when the **Current Routing Mode (RM) [setting/ parameter]** is 0 (server) since in this mode outgoing connections are not allowed.

When DHCP is activated (**DHCP (DH) setting** is 1(enabled)) the Gateway IP-address obtained from the DHCP server is saved into this setting thus overwriting older value that might have been set before. This only happens when the DHCP server is configured to provide netmask data.

Password (PW) setting

Description (see setting description format info here)

Function:	Defines login password for network programming
Set (S) command format:	**SPW*pp...p***, where ***pp...p***: is the password string, 0-6 characters long
Get (G) command format:	**GPW**
Init (I) command effect:	Initialized unconditionally, through network command, serial command, or quick initialization
Post-initialization value:	<NULL>
Change takes effect:	Immediately
Overriding parameter:	---
Relevance conditions:	**---**
First introduced:	Earlier than "baseline" V3.14/V3.51
See also:	Authentication

Details

Certain network commands require authentication. To authenticate itself the network host must provide a password that matches the one defined by the **Password setting**.

Password setting also serves one other purpose- it is used for logins onto the Link Server.

Connection Settings

Connection settings define how and in which fashion the DS establishes connections to and accepts connections from other hosts. For more information see Ethernet port and network communications.

The following settings belong to this group:

Setting	Description
Connection Timeout (CT) setting	Defines connection timeout (in minutes)
Transport Protocol (TP) setting	Defines whether UDP or TCP protocol will be used for data connections
Broadcast UDP (BU) setting	Defines whether DS will accept or reject broadcast UDP datagrams
Link Service Login (TL) setting **[V3.24/3.54+]**	Enables/disables Link Service login procedure after a TCP data connection is established
Inband Commands (IB) setting	Defines whether inband command passing is enabled or disabled
Data Login (DL) setting	Enables or disables command-phase TCP programming
Routing Mode (RM) setting	Defines whether incoming and/or outgoing data connections are allowed
Source IP Filtering (SF) setting **[V3.24/3.54+]**	Defines whether the DS will accept incoming data connections from any network host or specific host only
Connection Mode (CM) setting	Defines conditions under which the DS will attempt to establish an outgoing connection to the remote host
Destination IP-address (DI) setting	Defines the IP-address of the destination network host to which the DS will attempt to connect to (by default)
Destination Port Number (DP) setting	Defines the port on the destination network host to which the DS will attempt to connect to (by default)
Notification Destination (ND) setting	Defines which UDP port the DS will send I/O line status change notifications to

Connection Timeout (CT) setting

Description (see setting description format info here)

Function: Defines data connection timeout

Set (S) command format: **SCT*ttt***, where ***ttt*** is connection timeout, 0-99 minutes; 0 means connection never times out

Get (G) command format: **GCT**

Init (I) command effect: Initialized unconditionally, through network command, serial command, or quick initialization

Post-initialization value:	5 (5 minutes)
Change takes effect:	After reboot
Overriding parameter:	---
Relevance conditions:	**---**
First introduced:	Earlier than "baseline" V3.14/V3.51
See also:	Ethernet port and network communications

Details

Connection Timeout defines after how many minutes an idle connection is terminated. Setting **Connection Timeout** to 0 disables automatic timeouts. Setting **Connection Timeout** to any value in the 1-99 range enables automatic timeouts: when no packets are transferred across a connection in either direction for a corresponding number of minutes the connection is terminated. The DS terminates TCP connections by sending an RST packet, while UDP "connections" are simply discarded (the other party is not informed in any way).

Connection Timeout prevents an idle ("hanged") connection from occupying the DS indefinitely thus keeping other network hosts from communicating with the DS. Note, that idle connection is defined as the one across which no *packets* are transferred for a period of time (not the one across which no *data* is transferred for a period of time). This provides a way of maintaining the connection even in the absence of data (this is known as "keepalive"). For TCP connections remote host can send empty ACK packets, for UDP "connection" remote host can send UDP datagrams of zero length.

Connection Timeout is relevant even when the **Connection Mode (CM) setting** is 0 (immediately). In this case, when timeout comes the DS terminates an existing connection to the network host and immediately opens a new one. This can be used to "auto-repair" hanged connections in systems where permanent connection to the network host must be maintained indefinitely.

Transport Protocol (TP) setting

Description (see setting description format info here)

Function:	Defines whether UDP or TCP protocol is used for data connections
Set (S) command format:	**STP*x***, where ***x***: 0 (UDP), 1 (TCP)
Get (G) command format:	**GTP**
Init (I) command effect:	Initialized unconditionally, through network command, serial command, or quick initialization
Post-initialization value:	0 (UDP)
Change takes effect:	After reboot
Overriding parameter:	**Transport Protocol (TP) Parameter**
Relevance conditions:	**---**
First introduced:	Earlier than "baseline" V3.14/V3.51
See also:	Ethernet port and network communications

Details

Transport Protocol defines which communications protocol- TCP/IP or UDP/IP will be used by the DS for exchanging data with the network host.

Some aspects of UDP and TCP implementation in the DS are different from standard or commonly used implementation. See UDP data "connections" and TCP data connections for more info on the subject.

Broadcast UDP (BU) setting

Description (see setting description format info here)

Function:	Defines whether DS will accept or reject broadcast UDP datagrams
Set (S) command format:	**SBU*x***, where ***x***: 0 (disabled), 1 (enabled)
Get (G) command format:	**GBU**
Init (I) command effect:	Initialized unconditionally, through network command, serial command, or quick initialization
Post-initialization value:	0 (disabled)
Change takes effect:	After reboot
Overriding parameter:	---
Relevance conditions:	**Current Transport Protocol (TP) [setting/ parameter]**= 0 (UDP)
First introduced:	Earlier than "baseline" V3.14/V3.51
See also:	Broadcast UDP communications

Details

When **Broadcast UDP** is 1 (enabled), the DS will accept and route the data received in broadcast UDP datagrams, as if these packets were addressed directly to this DS. Broadcast packets must still be addressed to the correct **Port Number (PN)**. DS will ignore broadcast UDP packets when **Broadcast UDP** is 0 (disabled).

This setting only allows/disallows the reception of broadcast UDP packets and has no influence over whether the DS can send out its own broadcast UDP datagrams or not. The DS can be made to send the broadcast packets by setting **current Destination IP-address (IP) [setting/ parameter/ instruction]** to 255.255.255.255.

This Setting is irrelevant when **current Transport Protocol (TP) [setting/ parameter]** is 1 (TCP) because TCP protocol cannot use broadcast packets to carry data.

Link Service Login (TL) setting

Description (see setting description format info here)

Function:	Enables/disables Link Service login procedure after a TCP data connection is established
Set (S) command format:	**STL*x***, where ***x***: 0 (disabled), 1 (enabled)
Get (G) command format:	**GTL**
Init (I) command effect:	Initialized unconditionally, through network

	command, serial command, or quick initialization
Post-initialization value:	0 (disabled)
Change takes effect:	After reboot
Overriding parameter:	**---**
Relevance conditions:	**Current Transport Protocol (TP) [setting/parameter]**= 1 (TCP)
First introduced:	**V3.24/3.54**
See also:	Ethernet port and network communications, Link Service

Details

This setting defines whether the DS will perform Link Service login procedure after a TCP data connection with another network host is established. For more information on Link Service see Link Server Documentation.

Link Service logins use the data from the following settings of the DS: **Owner Name (ON)**, **Device Name (DN)**, and **Password (PW) setting**). The password is supplied in the encrypted form so Link Service logins are secure.

Link Server Login is irrelevant when **Current Transport Protocol (TP) [setting/parameter]**= 0 (UDP) because UDP/IP cannot be used for communications with the Link Server.

Inband Commands (IB) setting

Description (see setting description format info here)

Function:	Enables or disables inband (TCP) programming
Set (S) command format:	**SIB*x***, where ***x***: 0 (disabled), 1 (enabled)
Get (G) command format:	**GIB**
Init (I) command effect:	Initialized unconditionally, through network command, serial command, or quick initialization
Post-initialization value:	0 (disabled)
Change takes effect:	After reboot
Overriding parameter:	---
Relevance conditions:	**Current Transport Protocol (TP) [setting/parameter]**= 1 (TCP) **AND** **current Link Service Login (TL) [setting/parameter]**= 0 (disabled)
First introduced:	Earlier than "baseline" V3.14/V3.51
See also:	Inband (TCP) commands

Details

Inband Commands setting defines whether inband (TCP) programming is enabled or disabled.

This Setting is irrelevant when the **current Transport Protocol (TP) [setting/parameter]** is 0 (UDP) because inband command passing is only possible when

TCP/IP is used for data connections between the network host and the DS.

Inband Commands setting is also irrelevant when **current Link Service Login (TL) [setting/parameter]**= 1 (enabled) because inband commands are *always* enabled when Link Service is used.

Data Login (DL) setting

Description (see setting description format info here)

Function:	Enables or disables command-phase (TCP) programming
Set (S) command format:	**SDL*x***, where ***x***: 0 (disabled), 1 (enabled)
Get (G) command format:	**GDL**
Init (I) command effect:	Initialized unconditionally, through network command, serial command, or quick initialization
Post-initialization value:	0 (disabled)
Change takes effect:	After reboot
Overriding parameter:	---
Relevance conditions:	**Current Transport Protocol (TP) [setting/parameter]**= 1 (TCP) AND **current Link Service Login (TL) [setting/parameter]**= 0 (disabled)
First introduced:	Earlier than "baseline" V3.14/V3.51
See also:	Command-phase (TCP) commands, Inband (TCP) commands

Details

Data Login defines whether command-phase TCP programming is enabled or disabled.

This setting is irrelevant when the **current Transport Protocol (TP) [setting/parameter]** is 0 (UDP) because command-phase programming is only possible when TCP/IP is used for data connections between the network host and the DS.

Command-phase programming is *disabled automatically* when **current Link Service Login (TL) [setting/parameter]** is 0 (disabled).

Retransmission Period (RP) setting

Description (see setting description format info here)

Function:	Defines the retransmission period for TCP packets
Set (S) command format:	**SRP*ttt***, where ***ttt*** is retransmission period in 0.5 second intervals, 1-255
Get (G) command format:	**GRP**
Init (I) command effect:	Initialized unconditionally, through network command, serial command, or quick initialization
Post-initialization value:	6 (3 seconds)
Change takes effect:	After reboot

Overriding parameter: ---

Relevance conditions: **Current Transport Protocol (TP) [setting/ parameter]**= 1 (TCP)

First introduced: **[V3.34/V3.66+]**

See also: TCP Data Connections

Details

Packet retransmission is a standard feature of TCP/IP protocol. Whenever a particular packet with data is lost the sender is supposed to retransmit this packet after a certain delay. Standard TCP implementations use variable delays that increase exponentially after each unsuccessful retry. Tibbo devices use a fixed delay specified, in 0.5 second intervals, by the **Retransmission Period**.

Default retransmission period works fine on most networks. Setting it to higher value may improve DS operation on networks with significant response delays, such as those including GPRS segments.

Retransmission Period is irrelevant when **Current Transport Protocol (TP) [setting/ parameter]** is 0 (UDP).

Routing Mode (RM) setting

Description (see setting description format info here)

Function: Defines whether incoming and/or outgoing data connections are allowed

Set (S) command format: **SRM*x***, where ***x***: 0 (server), 1 (server/client), 2 (client)

Get (G) command format: **GRM**

Init (I) command effect: Initialized unconditionally, through network command, serial command, or quick initialization

Post-initialization value: 0 (server)

Change takes effect: After reboot

Overriding parameter: **Routing Mode (RM) parameter**

Relevance conditions: **---**

First introduced: Earlier than "baseline" V3.14/V3.51

See also: Ethernet port and network communications, Error mode

Details

Routing Mode defines whether the DS will accept incoming connections (passive opens) and/or establish outgoing connections (perform active opens):

0 (server) Only incoming connections are accepted, the DS never attempts to establish an outgoing connection to the network host. There is no restriction on which network host can connect to the DS- connection from any IP-address will be accepted as long as the remote host is connecting to the correct **Port Number (PN)** using **current Transport Protocol (TP) [**

setting/ parameter].

1 (server/client)	Both incoming and outgoing connections are allowed*. Outgoing connections are established with **current Destination IP-address (DI) [setting/ parameter/ instruction]** and **current Destination Port (DP) [setting/ parameter/ instruction]**. Exactly when the DS attempts to establish an outgoing connection is defined by the **Connection Mode (CM) setting**.
2 (client)	Only outgoing connections are allowed, the DS rejects all incoming connections. This is a newly implemented routing mode.

Current Routing Mode of the DS is changed to 0 (server) if the DS enters the error mode(the value of the **Routing Mode setting** itself remains intact).

** Since the DS only allows for a single data connection at any given time this should be understood as "whichever comes first".*

Source IP Filtering (SF) setting

Description (see setting description format info here)

Function:	Defines whether the DS will accept incoming data connections from any network host (filtering disabled) or specific host only (filtering enabled)
Set (S) command format:	**SSF*x***, where ***x***: 0 (disabled), 1 (enabled)
Get (G) command format:	**GSF**
Init (I) command effect:	Initialized unconditionally, through network command, serial command, or quick initialization
Post-initialization value:	0 (disabled)
Change takes effect:	After reboot
Overriding param/instr:	**Source IP Filtering (SF) parameter**
Relevance conditions:	**Current Routing Mode (RM) [setting/ parameter]**= 0 (server) or 1 (server/client)
First introduced:	**V3.24/3.54**
See also:	Ethernet port and network communications

Details

When **Source IP Filtering** is 0 (disabled) the DS will accept an incoming data connection from any network host.

When **Source IP Filtering** is 1 (enabled) the DS will accept an incoming data connection only from host whose IP-address matches the one specified by **current Destination IP-address [setting/parameter]**.

Since out-of-band on-the-fly commands (network-side parameters issued using **Parameter (P) command**) can also be considered a part of a data connection the DS will reject any such command that comes from a "wrong" IP-address while source IP-address filtering is enabled.

This setting is irrelevant when **current Routing Mode (RM) [setting/parameter]** is 2 (client) because in this mode the DS won't accept incoming connections at

all.

Connection Mode (CM) setting

Description (see setting description format info here)

Function:	Defines conditions under which the DS will attempt to establish an outgoing connection to the remote host
Set (S) command format:	**SCM*x***, where ***x***: 0 (immediately), 1 (on data or command), 2 (on command), 3 (on command or DSR=HI*)
Get (G) command format:	**GCM**
Init (I) command effect:	Initialized unconditionally, through network command, serial command, or quick initialization
Post-initialization value:	1 (on data or command)
Change takes effect:	After reboot
Overriding parameter:	---
Relevance conditions:	**Current Routing Mode (RM) [setting/ parameter]**= 1 (server/client) **OR** 2 (client)
First introduced:	Earlier than "baseline" V3.14/V3.51
See also:	Ethernet port and network communications, Serial-to-Ethernet data routing, UDP data "connections"

Details

Connection Mode defines under which condition the DS attempts to establish an outgoing connection** to **current Destination IP-address (DI) [setting/ parameter/ instruction]** and **current Destination Port Number (DP) [setting/ parameter/ instruction]**:

0 (immediately)	The DS attempts to establish an outgoing connection right after the powerup***. The DS will also make this connection "persistent". If the connection is closed (aborted) by the network host the DS will (attempt to) establish it again. Connection timeout (defined by the **Connection Timeout (CT) setting**) still works in this mode: when the current connection times out the DS aborts it and immediately establishes a new connection. Such behavior "auto-repairs" hanged connections.
1 (on data or command)	The DS attempts to establish an outgoing connection when the first serial data is received into the serial port and committedOR when **Establish Connection (CE) instruction** is issued.
2 (on command)	The DS attempts to establish an outgoing connection only when **Establish Connection (CE) instruction** is issued.
3 (on command or DSR=HI)	The DS attempts to establish an outgoing

connection only when **Establish Connection (CE) instruction** is issued OR when the DSR line of the serial port is brought HI* (for at least 20ms).

Existing connection can always be terminated by using the **Close Connection (CC) instruction** or **Abort Connection (CA) instruction**. With **Connection Mode** 3 (on command or DSR=HI*) it is also possible to close the connection by bringing the DSR line LOW*.

Connection Mode is irrelevant when the **current Routing Mode (RM)** is 0 (server) since in this mode outgoing connections are not allowed at all.

** HI and LOW states are described with respect to the serial ports of DS100R, DS100B, DS203. For EM100, EM120, EM200, EM203(A) the signaling is exactly opposite.*

*** Since the DS only allows for a single data connection at a time all conditions described here only apply to a situation when no data connection is established yet.*

**** After the IP-address is obtained from the DHCP server if the **DHCP (DH) setting** is 1 (enabled).*

Destination IP-address (DI) setting

Description (see setting description format info here)

Function:	Defines the IP-address of the destination network host to which the DS will attempt to connect to (by default)
Set (S) command format:	**SDI*xxx.xxx.xxx.xxx***, where ***xxx.xxx.xxx.xxx*** is the IP-address of the destination in dot-decimal notation (i.e. 192.168.100.41)
Get (G) command format:	**GDI**
Init (I) command effect:	Initialized unconditionally, through network command, serial command, or quick initialization
Post-initialization value:	0.0.0.1 (changed from 127.0.0.1 in **V3.30/3.62**)
Change takes effect:	After reboot
Overriding param/instr:	**Destination IP-address (DI) parameter**, **Establish Connection (CE) instruction**
Relevance conditions:	For outgoing connections: **Current Routing Mode (RM) [setting/parameter]**= 1 (server/client) or 2 (client) **[V3.24/3.54+]** For incoming connections: **Current Routing Mode (RM) [setting/parameter]**= 0 (server) or 1 (server/client) **AND current Source IP Filtering (SF) [setting/parameter]**= 1 (enabled)
First introduced:	Earlier than "baseline" V3.14/V3.51, **functionality extended in V3.24/3.54**
See also:	Ethernet port and network communications

Details

Destination IP-address serves two purposes:

- It defines the IP-address of the network host to which the DS will attempt to establish an outgoing data connection. Exactly *when* the DS will attempt to establish such a connection is specified by the **Connection Mode (CM) setting**. Destination port the DS will attempt to connect to is specified by the **current Destination Port Number (DP) [setting/ parameter/ instruction]**.
- **[V3.24/3.54+]** This address also specifies the only network host from which an incoming data connection will be accepted when **current Source IP Filtering (SF) [setting/parameter]** is 1 (enabled).

Destination IP-address of 255.255.255.255 means "link-level broadcasts". This is a special case so the following considerations should be taken into account:

- For outgoing connections:
  - When **current Transport Protocol (TP) [setting/parameter]** is 0 (UDP) the DS will be sending out its own UDP datagrams as link-level broadcasts i.e. with destination MAC-address set to 255.255.255.255.255.255. Furthermore, there will be no destination switchover that happens when UDP datagram is received from a network host (therefore, the DS will keep sending its datagrams as broadcasts). For more information see UDP data "connections" and broadcast UDP communications.
  - When **current Transport Protocol (TP) [setting/parameter]** is 1 (TCP) the DS will not attempt to establish an outgoing connection at all. This is because TCP is strictly a point-to-point protocol and does not support broadcasting.
- **[V3.24/3.54+]** For incoming connections:
  - With **Destination IP-address** set to 255.255.255.255 the DS will accept incoming connections from *any* network host *even if* **current Source IP Filtering (SF) [setting/parameter]** is 1 (enabled).

Destination IP-address is irrelevant in the following cases:

- For outgoing connections:
  - When **current Routing Mode (RM) [setting/parameter]** is 0 (server) because the DS does not establish outgoing connections in this mode at all
- **[V3.24/3.54+]** For incoming connections:
  - When **current Routing Mode (RM) [setting/parameter]** is 2 (client) because the DS does not accept any incoming connections in this mode at all
  - When **current Source IP Filtering (SF) [setting/parameter]** is 0 (disabled).

Destination Port Number (DP) setting

Description (see Setting description format info here)

Function:	Defines the port on the destination network host to which the DS will attempt to connect to
Set (S) command format:	**SDP*ppppp***, where ***ppppp*** is the port number of the destination in the 0-65535 range
Get (G) command format:	**GDP**
Init (I) command effect:	Initialized unconditionally, through network command, serial command, or quick initialization

Post-initialization value:	1001
Change takes effect:	After reboot
Overriding param/instr:	**Destination Port Number (DP) Parameter**, **Establish Connection (CE) instruction**
Relevance conditions:	**Current Routing Mode (RM) [setting/ parameter]**= 1 (server/client) or 2 (client)
First introduced:	Earlier than "baseline" V3.14/V3.51
See also:	Ethernet port and network communications

Details

Destination Port Number defines the port of the network host to which the DS will attempt to establish an outgoing connection. When the DS will attempt to establish a connection to the destination host is defined by the **Connection Mode (CM) setting**. Destination IP-address the DS will attempt to connect to is defined by the **current Destination IP-address (DI) [setting/ parameter/ instruction]**.

Destination Port Number is irrelevant when the **current Routing Mode (RM) [setting/parameter]** is 0 (server) since in this mode outgoing connections are not allowed.

Notification Destination (ND) setting

Description (see setting description format info here)

Function:	Defines which UDP port the DS will send I/O line status change notifications to
Set (S) command format:	**SND*x***, where ***x***: 0 (last known port), 1 (port 65535)
Get (G) command format:	**GND**
Init (I) command effect:	Initialized unconditionally, through network command, serial command, or quick initialization
Post-initialization value:	0 (last known port)
Change takes effect:	After reboot
Overriding parameter:	---
Relevance conditions:	**Notification Bitmask (NB) setting**<>0 **AND** notifications are sent as out-of-band UDP datagrams (see **Notification Bitmask (NB) setting**for details)
First introduced:	Earlier than "baseline" V3.14/V3.51
See also:	Ethernet port and network communications, Serial port and serial communications

Details

Notification (J) messages are generated when one of the monitored I/O lines of the DS changes its state. Which I/O lines are monitored for changes is defined by the **Notification Bitmask (NB) setting**. Notifications are only sent when the data connection is established and are sent to the network host with which this connection is established. **Notification Destination** defines which UDP port on the

destination host notifications are sent to when notifications are being sent as UDP datagrams (out-of-band):

0 (last known port) Notifications are sent to the UDP port from which the most recent programming UDP datagram was received. Therefore, notifications are sent after at least one such datagram is received (since the data connection is established). This option is useful when the DS has to send notifications to the PC. Port number from which PC applications are sending their programming UDP datagrams are usually ephemeral which means that they are always changing. Therefore, the DS will wait for the first programming UDP datagram to arrive.

1 (port 65535) Notifications are always sent to the UDP port 65535. This option is useful when the DS is communicating with another DS. In this case the port to send notifications to is fixed and known so there is no need to wait for the programming UDP datagram to arrive.

Notification Destination is irrelevant when the **Notification Bitmask (NB)** is set to is 0 because this means that no I/O lines of the DS are monitored for changes (so there will be no notifications to generate). **Notification Destination** is also irrelevant when notifications are being sent inside the data TCP connection itself (see **Notification (J) message** for more information on when this happens).

Serial Settings

Serial settings define operation of the serial port of the DS. For more information see serial port and serial communications.

The following settings belong to this group:

Setting	Description
Serial Interface (SI) setting	Selects full-duplex or half-duplex mode for the serial port of the DS
Flow Control (FC) setting	Enables or disables hardware (RTS/CTS) flow control for the serial port of the DS
DTR Mode (DT) setting	Defines the function of the DTR line of the serial port of the DS
DTR Startup Mode (DS) setting **[V3.27/V3.57+]**	Defines the startup mode of the DTR line (High or Low)
Baudrate (BR) setting	Defines the baudrate of the serial port of the DS
Parity (PR) setting	Defines the parity mode of the serial port of the DS
Bits Per Byte (BB) setting	Defines the bits/byte mode of the serial port of the DS
Soft Entry (SE) setting	Disables or enabled serial programming mode entry through escape sequence and selects escape sequence type
Escape Character (EC) setting **[V3.24/3.54+]**	Defines ASCII code of character used in escape sequence
On-the-fly Commands (RC) setting	Enables or disables on-the-fly command processing by the DS
On-the-fly Password (OP) setting	Enables or disables password protection for on-the-fly commands

Notification Bitmask (NB) setting	Defines which I/O lines of the DS are monitored for changes

Serial Interface (SI) setting

Description (see setting description format info here)

Function:	Selects full-duplex or half-duplex mode for the serial port of the DS (when in the data routing mode)
Set (S) command format:	**SSI*x***, where ***x***: 0 (full-duplex), 1 (half-duplex), 2 (auto)
Get (G) command format:	**GSI**
Init (I) command effect:	Initialized unconditionally, through network command, serial command, or quick initialization
Post-initialization value:	2 (auto)
Change takes effect:	After reboot or exiting the serial programming mode
Overriding parameter:	---
Relevance conditions:	**---**
First introduced:	Earlier than "baseline" V3.14/V3.51
See also:	Serial port and serial communications

Details

Serial port of the DS can operate in full-duplex or half-duplex mode. "Full-duplex" and "half-duplex" here refers exclusively to the logical operation of the DS, not to the hardware implementation of the serial port, which depends on the DS model.

The setting only influences the operation of the serial port in the data routing mode. When the serial port is in the serial programming mode it is always using the half-duplex interface- see serial programming for details.

0 (full-duplex)	Suitable for RS232 and RS422 communications. RTS (output) and CTS (input) lines are used in a "normal" way i.e. for flow control between the DS and attached serial device (when **current Flow Control (FC) [setting/ parameter]**= 1 (enabled)), or signaling between the network host and attached serial device (through **Set I/O Pin Status (Sx) instructions**, **Get I/O Pin Status (Gx) instructions**, and **Notification (J) messages**).
1 (half-duplex)	Suitable for RS485 communications. In this mode the RTS line provides direction control and the CTS line is unused. When the Ethernet-to-serial buffer of the DS is empty (nothing to send out through the serial port) the RTS line is HI*. When there is some data to send out the RTS line is LOW* for as long as the data is being output. Such behavior is intended to allow the RTS line to control the direction pin of the RS485 interface ICs and RS232-to-RS485 converters.
2 (auto)	In this mode the DS selects full-duplex or half-duplex mode automatically, depending on the hardware. For

Ethernet Modules selection is done by interconnecting or not interconnecting a pair of I/O pins. External Device Servers carry necessary selection circuit internally. Table below details interface selection through "hardware".

Hardware selection of full-duplex/half-duplex mode on Tibbo Device Servers

When **Serial Interface** is at 2 (auto) the DS selects full-duplex or half-duplex mode for its serial port basing on "hardware". Table below details hardware-based mode selection.

DS Model	For full-duplex mode	For half-duplex mode	Comments
EM100	Leave CTS/SEL and ER/WS unconnected	Connect CTS/SEL to ER/WS**	
EM120 EM200	Leave CTS/SEL and SR unconnected	Connect CTS/SEL to SR**	
DS100 EM100-EV	Always selected	---	These products are based on the EM100; they only support RS232 i/f so CTS/SEL and ER/WS are left unconnected internally
DS100B	Depends on jumpers		This product is based on the EM100; jumpers "decide" whether CTS/SEL and ER/WS are interconnected
DS203 EM120/EM200-EV	Always selected	---	These products are based on the EM120 or EM200; they only support RS232 i/f so CTS/SEL and SR are left unconnected internally

** HI and LOW states are described with respect to the serial ports of DS100R, DS100B, DS203. For EM100, EM120, EM200, EM203(A) the signaling is exactly opposite.*

*** Whether or not these two lines are interconnected is tested once at powerup. Connecting or separating these line during device operation will not cause immediate change of selected interface mode.*

Flow Control (FC) setting

Description (see setting description format info here)

Function:	Enables or disables hardware (RTS/CTS) flow control for the serial port of the DS (when in the data routing mode)
Set (S) command format:	**SFC*x***, where ***x***: 0 (disabled), 1 (enabled)
Get (G) command format:	**GFC**
Init (I) command effect:	Initialized unconditionally, through network command, serial command, or quick initialization
Post-initialization value:	1 (enabled)
Change takes effect:	After reboot or exiting the serial programming mode
Overriding parameter:	**Flow Control (FC) parameter**
Relevance conditions:	**current Serial Interface (SI)**= 0 (full duplex)
First introduced:	Earlier than "baseline" V3.14/V3.51
See also:	Serial port and serial communications, Data routing

---

Details

Flow Control setting defines the behavior of the RTS and CTS lines of the DS, unless overridden by the **Flow Control (FC) parameter**.

When the **Flow Control** is 0 (disabled) the status of the CTS input is ignored and the RTS line is at HIGH* unless changed by the remote host. Remote host can control the RTS line through **Set I/O Pin Status (Sx) instructions** or **Notification (J) messages**.

When the **Flow Control** is 1 (enabled) the RTS (output) and CTS (input) lines are used to regulate the flow of data across the serial cable connecting the DS to the attached serial device.

RTS is used to regulate the flow of data from the attached serial device to the DS. When the serial-to-Ethernet buffer of the DS has free space the RTS is HIGH* and the serial device is free to send the data. When the buffer becomes (almost) full the DS sets the RTS line to LOW* thus telling the serial device to stop sending the data. The RTS is set to LOW* when there are less than 20 free bytes left in the serial-to-Ethernet buffer. Additionally, the RTS line is set to LOW* in all cases when the serial port is closed.

CTS is used to regulate the flow of data from the DS to the attached serial device. As long as the DS detects HIGH* on its CTS input it is free to send out the data to the attached serial device. To stop the DS from sending out the data, the serial device must set the CTS line to LOW*. The DS won't send out the data for as long as it detects LOW* on its CTS input.

When the **Flow Control** is 1 (enabled) remote host cannot remotely control the status of the RTS line. **Set I/O Pin Status instructions** addressing the RTS pin are ignored by the DS (although the **OK (A) status code** is still returned) and incoming **Notification (J) messages**also have no effect on the state of the RTS line.

Remote host can get current status of RTS and CTS lines at any time using the **Get I/O Pin Status (Gx) instruction**, regardless of whether the **Flow Control** is 0 (disabled) or 1 (enabled). Status change monitoring for RTS and CTS lines can also

be enabled and **Notification (J) messages** generated regardless of the value of the **Flow Control** setting.

Flow Control is irrelevant when the current serial interface is half-duplex.

** HIGH and LOW states are described with respect to the serial ports of DS100R, DS100B, DS203. For EM100, EM120, EM200, EM203(A) the signaling is exactly opposite.*

DTR Mode (DT) setting

Description (see setting description format info here)

Function:	Defines the function of the DTR line of the serial port of the DS
Set (S) command format:	**SDT*x***, where ***x***: 0 (idle), 1 (connection status)
Get (G) command format:	**GDT**
Init (I) command effect:	Initialized unconditionally, through network command, serial command, or quick initialization
Post-initialization value:	0 (idle)
Change takes effect:	After reboot
Overriding parameter:	---
Relevance conditions:	---
First introduced:	Earlier than "baseline" V3.14/V3.51
See also:	Ethernet port and network communications, Serial port and serial communications

Details

When the **DTR Mode** is 0 (idle) the DTR line is at HIGH* unless changed by the remote host. Remote host can control the DTR line through **Set I/O Pin Status (Sx) instructions** or **Notification (J) messages**.

When the **DTR Mode** is 1 (connection status) the DTR (output) line of the DS reflects current connection status: the DTR line is LOW* when no data connection is established at the moment and HIGH* when there is a data connection in progress.

Remote host can get current status of the DTR line at any time using the **Get I/O Pin Status (Gx) instruction**, regardless of whether the **DTR Mode** is 0 (idle) or 1 (connection status). Status change monitoring for the DTR line can also be enabled and **Notification (J) messages** generated regardless of the value of the **DTR Mode** setting.

** HIGH and LOW states are described with respect to the serial ports of DS100R, DS100B, DS203. For EM100, EM120, EM200, EM203(A) the signaling is exactly opposite.*

DTR Startup Mode (DS) setting

Description (see setting description format info here)

Function:	Defines High or Low state of DTR pin on Startup
Set (S) command format:	**SDSx**, where ***x***: 0 (LOW* on startup), 1 (HIGH* on startup)

Get (G) command format:	**GDS**
Init (I) command effect:	Initialized unconditionally, through network command, serial command, or quick initialization
Post-initialization value:	0
Change takes effect:	After reboot or exiting the serial programming mode
Overriding parameter:	---
Relevance conditions:	**DTR Mode (DT) setting** = 0 (idle)
First introduced:	**V3.27/V3.57**
See also:	Serial programming

Details

This setting defines the startup voltage of the DTR pin. By default, DTR is LOW* on startup. By changing this setting, you can have DTR set to HIGH* on startup.

DTR Startup Mode is irrelevant when the **DTR Mode (DT) setting** is 1 (connection status).

** HIGH and LOW states are described with respect to the serial ports of DS100R, DS100B, DS203. For EM100, EM120, EM200, EM203(A) the signaling is exactly opposite.*

Baudrate (BR) setting

Description (see setting description format info here)

Function:	Defines the baudrate of the serial port of the DS
Set (S) command format:	**SBR*x***, where ***x***: 0 (1200bps), 1 (2400bps), 2 (4800bps), 3 (9600bps), 4 (19200bps), 5 (38400bps), 6 (57600bps), 7 (115200bps), 8 (150bps), 9 (300bps), 10 (600bps), 11 (28800bps)
Get (G) command format:	**GBR**
Init (I) command effect:	Initialized unconditionally, through network command, serial command, or quick initialization
Post-initialization value:	5 (38400bps)
Change takes effect:	After reboot or exiting the serial programming mode
Overriding parameter:	**Baudrate (BR) parameter**
Relevance conditions:	---
First introduced:	Earlier than "baseline" V3.14/V3.51
See also:	Serial port and serial communications

Details

Baudrate setting defines the baudrate of the serial port of the DS.

This baudrate is used by the serial port in the data routing mode (unless overridden

by the **Baudrate (BR) parameter**) and also in the serial programming mode, but only if the serial programming mode was entered through the escape sequence. If the serial programming mode is entered by pressing the setup button the baudrate becomes 38400bps regardless of the value of the **Baudrate setting**.

Parity (PR) setting

Description (see setting description format info here)

Function:	Defines the parity mode of the serial port of the DS (when in the data routing mode)
Set (S) command format:	**SPR*x***, where ***x***: 0 (off), 1 (even), 2 (odd), 3 (mark), 4 (space)
Get (G) command format:	**GPR**
Init (I) command effect:	Initialized unconditionally, through network command, serial command, or quick initialization
Post-initialization value:	0 (off)
Change takes effect:	After reboot or exiting the serial programming mode
Overriding parameter:	**Parity (PR) parameter**
Relevance conditions:	---
First introduced:	Earlier than "baseline" V3.14/V3.51
See also:	Serial port and serial communications

Details

Parity setting defines the parity mode of the serial port of the DS, unless overridden by the **Parity (PR) parameter**.

DS hardware does not have an option of two stop bits. The way around this is to set the Parity to 3 (mark). Since this means that the parity bit will always be set to 1 and since the parity bit is always transmitted in front of the stop bit, this will have the same result as having two stop bits.

The DS transmits the serial data with the parity bit correctly set but does not verify correctness of parity bits in the received data.

Parity is always off when the serial port is in the serial programming mode.

Bits Per Byte (BB) setting

Description (see setting description format info here)

Function:	Defines the bits/byte mode of the serial port of the DS (when in the data routing mode)
Set (S) command format:	**SBB*x***, where ***x***: 0 (7 bits), 1 (8 bits)
Get (G) command format:	**GBB**
Init (I) command effect:	Initialized unconditionally, through network command, serial command, or quick initialization
Post-initialization value:	1 (8 bits)
Change takes effect:	After reboot or exiting the serial programming mode

Overriding parameter:	**Bits per byte (PR) parameter**
Relevance conditions:	---
First introduced:	Earlier than "baseline" V3.14/V3.51
See also:	Serial port and serial communications

Details

Bits per byte setting defines the bits/byte mode of the serial port of the DS, unless overridden by the **Bits per byte (BB) parameter**.

8 bits/byte are always used when the DS is in the serial programming mode.

Soft Entry (SE) setting

Description (see setting description format info here)

Function:	Disables or enabled serial programming mode entry through escape sequence and selects escape sequence type
Set (S) command format:	**SSE*x***, where ***x***: 0 (disabled), 1 (type 1), 2 (type 2)
Get (G) command format:	**GSE**
Init (I) command effect:	Initialized unconditionally, through network command, serial command, or quick initialization
Post-initialization value:	0 (disabled)
Change takes effect:	After reboot or exiting the serial programming mode
Overriding parameter:	---
Relevance conditions:	---
First introduced:	Earlier than "baseline" V3.14/V3.51
See also:	Serial programming, Data routing

Details

Soft Entry controls serial programming mode entry through escape sequence:

0 (disabled) Serial programming mode entry through escape sequence is disabled.

1 (type 1) Type 1 escape sequence consists of three consecutive escape characters (ASCII code of escape character is defined by the **Escape Character (EC) setting**). For the escape sequence to be recognized each of the escape characters must be preceded by a time gap of at least 100ms:

...previous data <--100ms--> **E.C.** <--100ms--> **E.C.** <--100ms--> **E.C.**

If the time gap before a certain escape character exceeds 100ms then this character is considered to be a part of the escape sequence and is not recorded into the serial-to-Ethernet buffer. If the time gap before a certain escape character is less than 100ms than this character is considered to be a normal data character and is saved into the serial-to-Ethernet buffer. Additionally, escape character

counter is reset and the escape sequence must be started again. The following example illustrates one important point (escape characters are shown as ■). Supposing, attached serial device sends the following string:

ABC<--100ms-->■<--100ms-->■■**DE**

First two escape characters is this example had correct time gap before them, so they were counted as a part of the escape sequence and not saved into the buffer. The third escape character did not have a correct time gap so it was interpreted as a data character and saved into the buffer. The following was routed to the network host:

ABC■DE

The side effect and the point this example illustrates is that first two escape characters were lost- they neither became a part of a successful escape sequence (because this sequence wasn't completed), nor were saved into the buffer.

2 (type 2) Type 2 escape sequence is not based on any timing. Escape sequence consists of escape character (defined by the **Escape Character (EC) setting**) followed by any character other than escape character. To send a *data* character whose ASCII code matches that of escape character the serial device needs to send this character *twice*. This will result in a single character being saved into the serial-to-Ethernet routing buffer. It is the responsibility of the serial device to parse through the data it sends to the DS and "double" all characters whose code matches that of escape character.

Example: the following sequence will make the DS enter the serial programming mode (that is, if current escape character is not 'D'):

ABC■D

In the sequence below two consequtive escape characters will be interpreted as data (data routed to the network host will contain only one such character- the DS will automatically eliminate the second one):

ABC■■

It should be noted that attached serial device should parse the data and double certain characters only for the data it sends to the DS. **Reverse operation is not needed** for the data being received by the serial device from the DS. Characters with ASCII code matching that of escape character arrive from the DS in a normal way- i.e. they are not "doubled".

Considerations Regarding RTS Line Status

Notice, that serial escape sequence works even when the serial port of the DS is closed (i.e. when the **Current Routing Mode (RM) [setting/ parameter]** is 0 (Slave) and the data connection is not established). "Closed" merely means that the serial port is not accepting any data into its serial-to-network buffer. The serial port is still listening for escape sequences even at this time.

When the DS is running with the **Flow Control (FC) setting** set to 1 (Local) and its serial port is "closed", the RTS line of the DS will be in the disabled state (thus indicating to attached serial device that transmission is not allowed). Therefore, if the serial device needs to transmit an escape sequence it must **ignore the state of RTS signal** at that time.

Further behaviour of the RTS line while *in* serial programming mode is documented under Serial Programming.

Escape Character (EC) setting

Description (see setting description format info here)

Function:	Defines ASCII code of character used in escape sequence
Set (S) command format:	**SEC*ccc***, where ***ccc*** is the ASCII code of escape character
Get (G) command format:	**GEC**
Init (I) command effect:	Initialized unconditionally, through network command, serial command, or quick initialization
Post-initialization value:	1
Change takes effect:	After reboot or exiting the serial programming mode
Overriding parameter:	---
Relevance conditions:	**Soft Entry (SE) setting**= 1 (type 1) or 2 (type 2)
First introduced:	**V3.24/3.54**
See also:	Serial programming

Details

This setting defines the ASCII code of character used in escape sequence. Which escape sequence, if any, is used is defined by the **Soft Entry (SE) setting**.

Escape Character is irrelevant when the **Soft Entry (SE) setting** is 0 (disabled).

On-the-fly Commands (RC) setting

Description (see setting description format info here)

Function:	Enables or disables on-the-fly command processing by the DS
Set (S) command format:	**SRC*x***, where ***x***: 0 (disabled), 1 (enabled)
Get (G) command format:	**GRC**
Init (I) command effect:	Initialized unconditionally, through network command, serial command, or quick initialization
Post-initialization value:	0 (disabled)
Change takes effect:	After reboot
Overriding parameter:	---
Relevance conditions:	---
First introduced:	Earlier than "baseline" V3.14/V3.51
See also:	Serial port and serial communications, Authentication, On-the-fly (network-side)

Parameters and instructions

Details

When **On-the-fly Commands setting** is 1 (enabled) the DS will accept on-the-fly commands (network-side parameters and instructions) from the network host. When **On-the-fly Commands setting** is 0 (disabled) the DS will reject on-the-fly commands from the host (**Denied (D) reply code** will be returned).

On-the-fly commands do not require prior authentication through **Login (L) command**. Password protection specifically for on-the-fly commands can be enabled through the **On-the-fly Password (OP) setting**.

On-the-fly commands provide a way of remotely controlling the serial port of the DS, hence, the name of this setting- "Remote Control".

On-the-fly Password (OP) setting

Description (see setting description format info here)

Function:	Enables or disables password protection for on-the-fly commands
Set (S) command format:	**SOP*x***, where ***x***: 0 (disabled), 1 (enabled)
Get (G) command format:	**GOP**
Init (I) command effect:	Initialized unconditionally, through network command, serial command, or quick initialization
Post-initialization value:	0 (disabled)
Change takes effect:	After reboot
Overriding parameter:	---
Relevance conditions:	**On-the-fly Commands (RC) setting** = 1 (enabled)
First introduced:	Earlier than "baseline" V3.14/V3.51
See also:	Serial port and serial communications, Authentication

Details

On-the-fly commands (network-side parameters and instructions) do not require prior authentication through **Login (L) command**. **On-the-fly Password** enables password authentication specifically for on-the-fly commands, i.e. for the **Parameter (P) command** that carries them.

When **On-the-fly Password** is 1 (enabled) and the **Password (PW) setting** is set (not <NULL>) the network host must supply a valid password with every **Parameter (P) command** it sends (see this command's description for more information on how to do this).

On-the-fly Password is irrelevant when the **On-the-fly Commands (RC) setting** is 0 (disabled) because in this case the DS doesn't accept on-the-fly commands at all.

Notification Bitmask (NB) setting

Description (see setting description format info here)

Function: Defines which I/O lines of the DS are monitored for changes

Set (S) command format: **SNB*bbb***, where ***bbb*** is a "collection" of bit flags each of which disables (when 0) or enables (when 1) status change monitoring for a specific I/O line of the DS. Allowable value range is 0-255.

Get (G) command format: **GNB**

Init (I) command effect: Initialized unconditionally, through network command, serial command, or quick initialization

Post-initialization value: 0 (status change monitoring is disabled for all I/O lines)

Change takes effect: After reboot

Overriding parameter: ---

Relevance conditions: **---**

First introduced: Earlier than "baseline" V3.14/V3.51

See also: Serial port and serial communications, Notification Destination (ND) setting

---

Details

Notification Bitmask defines which I/O lines are monitored for changes. When the line being monitored changes its state a **Notification (J) message** is generated and sent to the network host with which the DS has an established data connection.

Notification Bitmask is a byte value each bit of which disables (when 0) or enables (when 1) status change monitoring for a specific I/O line. Bit assignment is as follows:

Bit position	DS100R, EM100-EV	DS100B, DS203, EM120/EM200-EV*
0 (LSB)	*<Not implemented>*	*<Not implemented>*
1	*<Not implemented>*	*<Not implemented>*
2	*<Not implemented>*	DSR (input)
3	*<Not implemented>*	DTR (output)**
4	CTS (input)	CTS (input)
5	RTS (output)**	RTS (output)**
6	*<Not implemented>*	*<Not implemented>*
7 (MSB)	*<Not implemented>*	*<Not implemented>*

Bit position	EM100	EM120, EM200
0 (LSB)	P0	*<Not implemented>*
1	P1	*<Not implemented>*
2	P2/DSR(input)***	P2/DSR(input)***
3	P3/DTR(output)***	P3/DTR(output)***
4	P4/CTS(input)***	P4/CTS(input)***
5	P5/RTS(output)***	P5/RTS(output)***
6	*<Not implemented>*	P6
7 (MSB)	*<Not implemented>*	P7

** This data is for the case when you are using RS232 FB9M connector of the EM120/EM200-EV. If you are using expansion connector (three jumpers removed) then use I/O data for the EM120 and EM200 Modules*

*** From firmware standpoint, these are general-purpose I/O lines. These I/O lines, however, are connected to the CMOS inputs of RS232 (RS422, RS485) transceivers and that dictates that the lines can only be used as outputs. Therefore, it is meaningless to enable notifications for such lines*

**** These are general-purpose input/output pins. Application firmware uses these pins to implement specific serial port functionality (shown in blue) and this defines "logical" direction of the pins*

Example: if **Notification Bitmask** is set to 20 then status change monitoring is enabled for lines DSR and CTS (binary representation of this value is 00010100, bits 2 and 4 are set, corresponding I/O lines are DSR and CTS).

Be sure to keep the mask bit for unimplemented and reserved lines at 0.

For the line status change to be detected, the new status must be preserved for at least 20 ms (milliseconds). Shorter "pulses" are ignored.

Note, that there is no bit for line P8 found on EM120 and EM200 Modules.

Encapsulation Settings

Encapsulation settings define what incoming serial data is recorded into the serial-to-Ethernet routing buffer and when and how this data is combined into the network packets and sent to the network host. For more information see serial-to-Ethernet data routing.

The following settings belong to this group:

Setting	Description
Maximum Packet Length (ML) setting	Defines after how many new bytes recorded into the serial-to-Ethernet buffer a break condition will be generated
Maximum Intercharacter Delay (MD) setting	Defines the maximum time gap between two consecutive characters recorded into the serial-to-Ethernet buffer, exceeding which will trigger a break condition
Start On Any Character (SA) setting	Defines whether the (next) data block can be opened by a specific character or any character
Use Start Character (F1) setting	Enables or disables the use of the start character
Start Character Code (S1) setting	Defines ASCII code of the start character
Use Stop Character (U1) setting	Enables or disables the use of the stop character
Stop Character Code (E1) setting	Defines ASCII code of the stop character

Number of Post Characters (P1) setting	Defines the number of post characters for the stop character

Maximum Packet Length (ML) setting

Description (see setting description format info here)

Function: Defines after how many new bytes recorded into the serial-to-Ethernet buffer a break condition will be generated

Set (S) command format: **SML*lll***, where ***lll*** is the number of new bytes in the 1-255 range

Get (G) command format: **GML**

Init (I) command effect: Initialized unconditionally, through network command, serial command, or quick initialization

Post-initialization value: 255 (bytes)

Change takes effect: After reboot or exiting the serial programming mode

Overriding parameter: ---

Relevance conditions: **Current Transport Protocol (TP) [setting/ parameter]**= 0 (UDP)

First introduced: Earlier than "baseline" V3.14/V3.51

See also: Serial-to-Ethernet data routing

Details

Maximum Packet Length defines after how many new bytes recorded into the serial-to-Ethernet buffer a break condition will be generated. Break conditions are related to routing the data in the serial-to-Ethernet direction. See serial-to-Ethernet data routing for more information.

Maximum Packet Length is irrelevant when **current Transport Protocol (TP) [setting/ parameter]** is 1 (TCP) because in this mode the DS ignores the value of the **Maximum Packet Length setting** and uses the run-time parameter of 127 bytes.

Maximum Intercharacter Delay (MD) setting

Description (see setting description format info here)

Function: Defines the maximum time gap between two consecutive characters recorded into the serial-to-Ethernet buffer, exceeding which will trigger a break condition

Set (S) command format: **SMD*ddd***, where ***ddd*** is the time gap in the 0-255 range (0-2550 ms), value of 0 deactivates break condition generation on time gaps

Get (G) command format: **GMD**

Init (I) command effect: Initialized unconditionally, through network command, serial command, or quick initialization

Post-initialization value:	1 (10 ms)
Change takes effect:	After reboot or exiting the serial programming mode
Overriding parameter:	---
Relevance conditions:	---
First introduced:	Earlier than "baseline" V3.14/V3.51
See also:	Serial-to-Ethernet data routing

Details

Maximum Intercharacter Delay defines the maximum time gap between two consecutive characters recorded into the serial-to-Ethernet buffer, exceeding which will trigger a break condition. Break conditions are related to routing the data in the serial-to-Ethernet direction. See serial-to-Ethernet data routing for more information.

Intercharacter delay can be set in 10 ms increments. Delay tracking is disabled when **Maximum Intercharacter Delay** is set to 0.

Intercharacter delay is not counted when the **Flow Control (FC) setting** is 1 (enabled) and the DS is holding the RTS line in the LOW* state thus indicating to the attached serial device that the DS is not ready to receive the data. This is done to prevent the DS from generating the break conditions on time gaps caused by the DS itself.

** HIGH and LOW states are described with respect to the serial ports of DS100R, DS100B, DS203. For EM100, EM120, EM200, EM203(A) the signaling is exactly opposite.*

Start On Any Character (SA) setting

Description (see setting description format info here)

Function:	Defines whether the (next) serial data block can be opened by a specific character or any character
Set (S) command format:	**SSA*x***, where ***x***: 0 (disabled), 1 (enabled)
Get (G) command format:	**GSA**
Init (I) command effect:	Initialized unconditionally, through network command, serial command, or quick initialization
Post-initialization value:	1 (enabled)
Change takes effect:	After reboot or exiting the serial programming mode
Overriding parameter:	---
Relevance conditions:	---
First introduced:	Earlier than "baseline" V3.14/V3.51
See also:	Serial-to-Ethernet data routing

Details

When **Start On Any Character** is 1 (enabled) then any character received into the serial port past the end of the previous serial data block will open a new serial data block. When the **Start On Any Character** is 0 (disabled) then only a selected specific character defined by the **Start Character Code (S1) setting** (must be enabled through the **Use Start Character (F1) setting**) will be able to open the new serial data block. All data between the end of the previous serial data block and this start character will be ignored.

Serial data blocks are related to routing the data in the serial-to-Ethernet direction. For more information see serial-to-Ethernet data routing.

Use Start Character (F1) setting

Description (see setting description format info here)

Function:	Enables or disables the use of the start character
Set (S) command format:	**SF1*x***, where ***x***: 0 (disabled), 1 (enabled)
Get (G) command format:	**GF1**
Init (I) command effect:	Initialized unconditionally, through network command, serial command, or quick initialization
Post-initialization value:	0 (disabled)
Change takes effect:	After reboot or exiting the serial programming mode
Overriding parameter:	---
Relevance conditions:	**Start On Any Character (SA) setting= 0** (disabled)
First introduced:	Earlier than "baseline" V3.14/V3.51
See also:	Serial-to-Ethernet data routing

Details

This setting enables the use of specific character code as the start character that can open the serial data block. Start character code itself is defined by the **Start Character Code (S1) setting**. Serial data blocks are related to routing the data in the serial-to-Ethernet direction. See serial-to-Ethernet data routing for more information.

Use Start Character is irrelevant when the **Start on Any Character (SA) setting** is 1 (enabled) because in this case the new serial data block is opened by any character and there is no need to set a specific start character.

Care should be taken not to disable **Use Start Character** and **Start on Any Character (SA) setting** at the same time. No data will ever be accepted into the serial port of the DS in this case!

Start Character Code (S1) setting

Description (see setting description format info here)

Function:	Defines ASCII code of the start character
Set (S) command format:	**SS1*ccc***, where ***ccc*** is the ASCII code of the start character in the 0-255 range

Get (G) command format:	**GS1**
Init (I) command effect:	Initialized unconditionally, through network command, serial command, or quick initialization
Post-initialization value:	0
Change takes effect:	After reboot or exiting the serial programming mode
Overriding parameter:	---
Relevance conditions:	**Start On Any Character (SA) setting= 0** (disabled) **AND Use Start Character (F1) setting**= 1 (enabled)
First introduced:	Earlier than "baseline" V3.14/V3.51
See also:	Serial-to-Ethernet data routing

Details

This setting defines the ASCII code of the start character that can open the serial data block. Start character must be enabled separately through the **Use Start Character (F1) setting**. Serial data blocks are related to routing the data in the serial-to-Ethernet direction. See serial-to-Ethernet data routing for more information.

Start Character Code is irrelevant when the **Start on Any Character (SA) setting** is 1 (enabled) because in this case the new serial data block is opened by any character and there is no need to set a specific start character.

Use Stop Character (U1) setting

Description (see setting description format info here)

Function:	Enables or disables the use of the stop character
Set (S) command format:	**SU1*x***, where ***x***: 0 (disabled), 1 (enabled)
Get (G) command format:	**GU1**
Init (I) command effect:	Initialized unconditionally, through network command, serial command, or quick initialization
Post-initialization value:	0 (disabled)
Change takes effect:	After reboot or exiting the serial programming mode
Overriding parameter:	---
Relevance conditions:	---
First introduced:	Earlier than "baseline" V3.14/V3.51
See also:	Serial-to-Ethernet data routing

Details

This setting enables the use of specific character code as the stop character that can close the serial data block. Stop character code itself is defined by the **Stop Character Code (S1) setting**. It is also possible to have the serial data block closed not on the stop character itself but after a predefined number of characters after the stop character. This is done through the **Number Of Post Characters**

(P1) setting.

Serial data blocks are related to routing the data in the serial-to-Ethernet direction. See serial-to-Ethernet data routing for more information.

If **Use Stop Character** is 0 (disabled) the serial data block, once opened, never closes.

Stop Character Code (E1) setting

Description (see setting description format info here)

Function:	Defines ASCII code of the stop character
Set (S) command format:	**SE1*ccc***, where ***ccc*** is the ASCII code of the stop character in the 0-255 range
Get (G) command format:	**GE1**
Init (I) command effect:	Initialized unconditionally, through network command, serial command, or quick initialization
Post-initialization value:	0
Change takes effect:	After reboot or exiting the serial programming mode
Overriding parameter:	---
Relevance conditions:	**Use Stop Character (U1) setting**= 1 (enabled)
First introduced:	Earlier than "baseline" V3.14/V3.51
See also:	Serial-to-Ethernet data routing

Details

This setting defines the ASCII code of the stop character that can close the serial data block. The usage of the stop character must be enabled separately through the **Use Stop Character (U1) setting**. It is also possible to have the serial data block closed not on the stop character itself but after a predefined number of characters after the stop character. This is done through the **Number Of Post Characters (P1) setting**.

Serial data blocks are related to routing the data in the serial-to-Ethernet direction. See serial-to-Ethernet data routing for more information.

Number Of Post Characters (P1) setting

Description (see setting description format info here)

Function:	Defines the number of post characters for the stop character
Set (S) command format:	**SP1*x***, where ***x***: is the number of post characters in the 0-15 range
Get (G) command format:	**GP1**
Init (I) command effect:	Initialized unconditionally, through network command, serial command, or quick initialization
Post-initialization value:	0
Change takes effect:	After reboot or exiting the serial programming

	mode
Overriding parameter:	---
Relevance conditions:	**Use Stop Character (U1) setting**= 1 (enabled)
First introduced:	Earlier than "baseline" V3.14/V3.51
See also:	Serial-to-Ethernet data routing, Stop Character Code (E1) setting

Details

Number Of Post Characters defines the number of characters received past the stop character that will still be counted as belonging to the same serial data block. Serial data blocks are related to routing the data in the serial-to-Ethernet direction. See serial-to-Ethernet data routing for more information.

Number of Post Characters is irrelevant when the **Use Stop Character (U1) setting** is 0 (disabled) because in this case the stop character is disabled.

Parameters and Instructions

This section contains a reference for all DS parameters and instructions.

Parameters are *temporary overrides* for settings. Parameters are not saved into the EEPROM and take immediate effect (no rebooting required). **Instructions** are used to make the DS perform a certain action.

Parameter and instruction description format can be found here.

All parameters and instructions can be divided into two groups:

- **On-the-fly (network-side) parameters and instructions** are delivered to the DS via the *network* **Parameter (P) command** and are used to change communications mode of the serial port of the DS without rebooting (i.e. "on-the-fly") and also set and sense the state of the I/O lines of the DS. For more information see serial port and serial communications;
- **Modem (serial-side) parameters and instructions** are delivered to the DS via the *serial* **Parameter (P) command** and are used to control data connection establishment and termination by the DS. For more information see Ethernet port and network communications.

Parameter & Instruction Description Format

All parameters and instructions in this section are described using the following format:

Function:	Parameter (instruction) function in brief
Parameter (P) cmd format:	Syntax of the **Parameter (P) command** that is used to send this parameter (instruction)
Possible replies:	Lists all possible reply status codes that can be returned in response to the **Parameter (P) command** carrying this parameter (instruction)
Relevance conditions:	Some parameters (instructions) are relevant to the operation of the DS only when other settings (or their overriding parameters) have certain values
First introduced:	Describes whether this parameter (instruction) has

been available right from the "baseline" firmware version of 3.14/3.51 or was introduced in a later firmware release

See also: Additional relevant links

Details

Additional information about the parameter (instruction).

On-the-fly (Network-Side) Parameters & Instructions

On-the-fly (network-side) parameters and instructions are delivered to the DS via the *network* **Parameter (P) command** and are used to change communications mode of the serial port of the DS without rebooting (i.e. "on-the-fly") and also set and sense the state of the I/O lines of the DS. For more information see serial port and serial communications.

The following parameters and instructions belong to this group:

Parameter/ instruction	Description
Flow Control (FC) parameter	Overrides **Flow Control (FC) setting**
Baudrate (BR) parameter	Overrides **Baudrate (BR) setting**
Parity (PR) parameter	Overrides **Parity (PR) setting**
Bits Per Byte (BB) parameter	Overrides **Bits Per Byte (BB) setting**
Notification Bitmask (NB) parameter	Overrides **Notification Bitmask (NB) setting**
Get I/O Pin Status (Gx) instruction	Reads the status of a certain I/O line of the DS
Set I/O Pin Status (Sx) instruction	Sets the status of a certain I/O line of the DS

Flow Control (FC) parameter

Description (see parameter description format info here)

Function: Overrides **Flow Control (FC) setting**

Parameter (P) cmd format: **PFC*x***, where ***x***: 0 (disabled), 1 (enabled)

Possible replies: **A**, **C**, **D**, **R**

Relevance conditions: **current Serial Interface (SI)**= 0 (full duplex)

First introduced: Earlier than "baseline" V3.14/V3.51

See also: Serial port and serial communications

Details

Flow Control parameter overrides the **Flow Control (FC) setting**.

Error (C) reply code is returned if the data supplied in the command is invalid.
Denied (D) reply code is returned if:

- **On-the-fly Commands (RC) setting** is 0 (disabled).
- If no password or incorrect password is supplied while the **On-the-fly Password (OP) setting** is 1 (enabled) and the password is set (value of the **Password (PW) setting** is not <NULL>).

Rejected (R) reply code is returned if this command is sent while the DS is in the

serial programming mode.

Baudrate (BR) parameter

Description (see parameter description format info here)

Function:	Overrides **Baudrate (BR) setting**
Parameter (P) cmd format:	**PBR*x***, where ***x***: 0 (1200bps), 1 (2400bps), 2 (4800bps), 3 (9600bps), 4 (19200bps), 5 (38400bps), 6 (57600bps), 7 (115200bps), 8 (150bps), 9 (300bps), 10 (600bps), 11 (28800bps)
Possible replies:	**A**, **C**, **D**, **R**
Relevance conditions:	**---**
First introduced:	Earlier than "baseline" V3.14/V3.51
See also:	Serial port and serial communications

Details

Baudrate parameter overrides the **Baudrate (BR) setting**.

Error (C) reply code is returned if the data supplied in the command is invalid.
Denied (D) reply code is returned if:

- **On-the-fly Commands (RC) setting** is 0 (disabled).
- If no password or incorrect password is supplied while the **On-the-fly Password (OP) setting** is 1 (enabled) and the password is set (value of the **Password (PW) setting** is not <NULL>).

Rejected (R) reply code is returned if this command is sent while the DS is in the serial programming mode.

Parity (PR) parameter

Description (see parameter description format info here)

Function:	Overrides **Parity (PR) setting**
Parameter (P) cmd format:	**PPR*x***, where ***x***: 0 (off), 1 (even), 2 (odd), 3 (mark), 4 (space)
Possible replies:	**A**, **C**, **D**, **R**
Relevance conditions:	**---**
First introduced:	Earlier than "baseline" V3.14/V3.51
See also:	Serial port and serial communications

Details

Parity parameter overrides the **Parity (PR) setting**.

Error (C) reply code is returned if the data supplied in the command is invalid.
Denied (D) reply code is returned if:

- **On-the-fly Commands (RC) setting** is 0 (disabled).
- If no password or incorrect password is supplied while the **On-the-fly Password (OP) setting** is 1 (enabled) and the password is set (value of the **Password**

(PW) setting is not <NULL>).

Rejected (R) reply code is returned if this command is sent while the DS is in the serial programming mode.

Bits Per Byte (BB) parameter

Description (see parameter description format info here)

Function:	Overrides **Bits Per Byte (BB) setting**
Parameter (P) cmd format:	**PBB*x***, where ***x***: 0 (7 bits), 1 (8 bits)
Possible replies:	**A**, **C**, **D**, **R**
Relevance conditions:	**---**
First introduced:	Earlier than "baseline" V3.14/V3.51
See also:	Serial port and serial communications

Details

Bits Per Byte parameter overrides the **Bits Per Byte (BB) setting**.

Error (C) reply code is returned if the data supplied in the command is invalid.
Denied (D) reply code is returned if:

- **On-the-fly Commands (RC) setting** is 0 (disabled).
- If no password or incorrect password is supplied while the **On-the-fly Password (OP) setting** is 1 (enabled) and the password is set (value of the **Password (PW) setting** is not <NULL>).

Rejected (R) reply code is returned if this command is sent while the DS is in the serial programming mode.

Notification Bitmask (NB) parameter

Description (see parameter description format info here)

Function:	Overrides **Notification Bitmask (NB) setting**
Parameter (P) cmd format:	**PNB*bbb***, where ***bbb*** is a "collection" of bit flags each of which disables (when 0) or enables (when 1) status change monitoring for a specific I/O line of the DS. Allowable value range is 0-255
Possible replies:	**A**, **C**, **D**, **R**
Relevance conditions:	**---**
First introduced:	Earlier than "baseline" V3.14/V3.51
See also:	Serial port and serial communications

Details

Notification Bitmask parameter overrides the **Notification Bitmask (NB) setting**. See **Notification Bitmask (NB) setting** for the bit position assignment within the ***bbb*** value.

Error (C) reply code is returned if the data supplied in the command is invalid.
Denied (D) reply code is returned if:

- **On-the-fly Commands (RC) setting** is 0 (disabled).

- If no password or incorrect password is supplied while the **On-the-fly Password (OP) setting** is 1 (enabled) and the password is set (value of the **Password (PW) setting** is not <NULL>).

Rejected (R) reply code is returned if this command is sent while the DS is in the serial programming mode.

Get I/O Pin Status (Gx) instruction

Description (see instruction description format info here)

Function: Reads the status of a certain I/O line of the DS

Parameter (P) cmd format: **PG*x***, where ***x*** is the I/O line number

Possible replies: **A*s***, **C**, **D**, **R**, where ***s*** is the state of I/O line (0 or 1)

Relevance conditions: **---**

First introduced: Earlier than "baseline" V3.14/V3.51

See also: Serial port and serial communications

Details

Get I/O Pin Status instruction returns the status of the DS I/O line specified by the ***x*** parameter:

Value of *x*	DS100R, EM100-EV	DS100B, DS203, EM120/EM200-EV*
0	*<Not implemented>*	*<Not implemented>*
1	*<Not implemented>*	*<Not implemented>*
2	*<Not implemented>*	DSR (input)**
3	*<Not implemented>*	DTR (output)**
4	CTS (input)	CTS (input)**
5	RTS (output)**	RTS (output)**
6	*<Not implemented>*	*<Not implemented>*
7	*<Not implemented>*	*<Not implemented>*
8	*<Not implemented>*	*<Not implemented>*

Value of *x*	EM100	EM120, EM200
0	P0	*<Not implemented>*
1	P1	*<Not implemented>*
2	P2/DSR(input)***	P2/DSR(input)***
3	P3/DTR(output)***	P3/DTR(output)***
4	P4/CTS(input)***	P4/CTS(input)***
5	P5/RTS(output)***	P5/RTS(output)***
6	*<Not implemented>*	P6
7	*<Not implemented>*	P7
8	*<Not implemented>*	P8

** This data is for the case when you are using RS232 FB9M connector of the EM120/EM200-EV. If you are using expansion connector (three jumpers removed) then use I/O data for the EM120 and EM200 Modules*

*** From firmware standpoint, these are general-purpose lines of the Ethernet Modules that can be used as inputs. These I/O lines, however, are connected to the CMOS inputs of RS232 (RS422, RS485) transceivers and that dictates that the lines*

can only be used as outputs.

**** These are general-purpose input/output pins. Application firmware uses these pins to implement specific serial port functionality (shown in blue) and this defines "logical" direction of the pins*

I/O line status (***s***) returned by the **Get I/O Pin Status instruction** indicates current status of the I/O lines of Modules. For Serial Device Servers and Boards that incorporate RS232 transceivers actual line status on the RS232 connector actual line status is exactly opposite to the value of ***s***: if ***s***=0 then the line is at HIGH, if ***s***=1- the line is at LOW. Notice, that not only inputs, but also outputs can be monitored using this command.

Error (C) reply code is returned if the data supplied in the command is invalid.
Denied (D) reply code is returned if:

- **On-the-fly Commands (RC) setting** is 0 (disabled).
- If no password or incorrect password is supplied while the **On-the-fly Password (OP) setting** is 1 (enabled) and the password is set (value of the **Password (PW) setting** is not <NULL>).

Rejected (R) reply code is returned if this command is sent while the DS is in the serial programming mode.

I/O line status can be set using **Set I/O Pin Status (Sx) instruction**.

Set I/O Pin Status (Sx) instruction

Description (see instruction description format info here)

Function:	Sets the status of a certain I/O line of the DS
Parameter (P) cmd format:	**PS*xs***, where ***x*** is the I/O line number and ***s*** is the desired status of the I/O line (0 or 1)
Possible replies:	**A**, **C**, **D**, **R**
Relevance conditions:	**---**
First introduced:	Earlier than "baseline" V3.14/V3.51
See also:	Serial port and serial communications

Details

Sets I/O Pin Status instruction allows the network host to remotely set the status of the DS I/O line. Parameter ***x*** specifies the I/O line:

Value of *x*	DS100R, EM100-EV	DS100B, DS203, EM120/EM200-EV*
0	*<Not implemented>*	*<Not implemented>*
1	*<Not implemented>*	*<Not implemented>*
2	*<Not implemented>*	DSR (input)**
3	*<Not implemented>*	DTR (output)
4	CTS (input)**	CTS (input)**
5	RTS (output)	RTS (output)
6	*<Not implemented>*	*<Not implemented>*
7	*<Not implemented>*	*<Not implemented>*
8	*<Not implemented>*	*<Not implemented>*

Value of *x*	EM100	EM120, EM200
0	P0	*<Not implemented>*
1	P1	*<Not implemented>*
2	*<Cannot be set>***	P2/DSR(input)****
3	P3/DTR(output)****	P3/DTR(output)****
4	P4/CTS(input)****	P4/CTS(input)****
5	P5/RTS(output)****	P5/RTS(output)****
6	*<Not implemented>*	P6
7	*<Not implemented>*	P7
8	*<Not implemented>*	P8

** This data is for the case when you are using RS232 FB9M connector of the EM120/EM200-EV. If you are using expansion connector (three jumpers removed) then use I/O data for the EM120 and EM200 Modules*

*** From firmware standpoint, these are general-purpose lines of the Ethernet Modules that can be used as outputs. These I/O lines, however, are connected to the CMOS outputs of RS232 (RS422, RS485) transceivers and that dictates that the lines must only be used as inputs.*

**** Hardware limitation*

***** These are general-purpose input/output pins. Application firmware uses these pins to implement specific serial port functionality (shown in blue) and this defines "logical" direction of the pins*

Desired I/O line state (***s***) corresponds to the status of I/O lines of Modules. For Serial Device Servers and Boards that incorporate RS232 transceivers actual line status on the RS232 connector is exactly opposite to the value of ***s***: if ***s***=0 then the line will be set to HIGH, if ***s***=1- the line will be set to LOW.

Error (C) reply code is returned if the data supplied in the command is invalid. **Denied (D) reply code** is returned if:

- **On-the-fly Commands (RC) setting** is 0 (disabled).
- If no password or incorrect password is supplied while the **On-the-fly Password (OP) setting** is 1 (enabled) and the password is set (value of the **Password (PW) setting** is not <NULL>).

Rejected (R) reply code is returned if this command is sent while the DS is in the serial programming mode.

There are several cases when the **Set I/O Pin Status (Sx) instruction** is ignored by the DS:

- When the **current Flow Control (FC) [setting/ parameter]** is 1 (enabled) or **current Serial Interface (SI)** is 1 (half-duplex) the DS ignores **SP5*s*** commands (i.e. commands that attempt to set the status of the RTS line). This is because in this case the RTS line is under the internal control of the DS.
- When the **DTR Mode (DT) setting** is 1 (connection status) the DS ignores **SP3*s*** commands (i.e. commands that attempt to set the status of the DTR line). This is because in this case the DTR line is under the internal control of the DS.

Note, that in all above cases the DS still returns **OK (A) reply code** but the commands are discarded internally.

Modem (Serial-Side) Parameters & Instructions

Modem (serial-side) parameters and instructions are delivered to the DS via the *serial* **Parameter (P) command** and are used to control data connection establishment and termination by the DS. For more information see Ethernet port and network communications.

The following parameters and instructions belong to this group:

Parameter/ instruction	Description
Transport Protocol (TP) parameter	Overrides **Transport Protocol (TP) setting**
Link Server Login (TL) parameter [V3.24/3.54+]	Overrides **LS Login (TL) setting**
Routing Mode (RM) parameter	Overrides **Routing Mode (RM) setting**
Source IP Filtering (SF) parameter	Overrides **Source IP Filtering (SF) setting**
Destination IP-address (DI) parameter	Overrides **Destination IP-address (DI) setting**
Destination Port Number (DP) parameter	Overrides **Destination Port Number (PN) setting**
Establish Connection (CE) instruction	Makes the DS establish data connection with the network host
Close Connection (CC) instruction	Makes the DS close data connection with the network host, reset buffer overflow flags
Abort Connection (CA) instruction	Makes the DS abort data connection with the network host, reset buffer overflow flags

Transport Protocol (TP) parameter

Description (see parameter description format info here)

Function:	Overrides **Transport Protocol (TP) setting**
Parameter (P) cmd format:	**PTP*x***, where ***x***: 0 (UDP), 1 (TCP)
Possible replies:	**A**, **C**, **R**
Relevance conditions:	**---**
First introduced:	Earlier than "baseline" V3.14/V3.51
See also:	Ethernet port and network communications

Details

Transport Protocol parameter overrides the **Transport Protocol (TP) setting**.

Error (C) reply code is returned if the data supplied in the command is invalid.
Rejected (R) reply code is returned if:

- Command is issued while the IP-address of the DS is not properly configured (**DHCP (DH) setting** is 1 (enabled) and the DS hasn't yet obtained the IP-address from the DHCP server).
- Current data connection state is *not* "closed". Transport protocol cannot be changed while the connection is in progress. Close (or shut) the connection using the **Close Connection (CC) instruction** or **Abort Connection (CA) instruction**

first. Current connection status can be verified using **Echo (X) command** (***c*** flag).

Link Server Login (TL) parameter

Description (see parameter description format info here)

Function:	Overrides **Link Server Login (TL) setting**
Parameter (P) cmd format:	**PTL*x***, where ***x***: 0 (disabled), 1 (enabled)
Possible replies:	**A**, **C**
Relevance conditions:	**Current Transport Protocol (TP) [setting/ parameter]**= 1 (TCP)
First introduced:	**V3.24/3.54**
See also:	Ethernet port and network communications

Details

Link Server Login parameter overrides the **Link Server Login (TL) setting**.

Error (C) reply code is returned if the data supplied in the command is invalid.

Routing Mode (RM) parameter

Description (see parameter description format info here)

Function:	Overrides **Routing Mode (RM) setting**
Parameter (P) cmd format:	**PRM*x***, where ***x***: 0 (server), 1 (server/client), 2 (client)
Possible replies:	**A**, **C**, **R**
Relevance conditions:	**---**
First introduced:	Earlier than "baseline" V3.14/V3.51
See also:	Ethernet port and network communications

Details

Routing Mode parameter overrides the **Routing Mode (RM) setting**.

Error (C) reply code is returned if the data supplied in the command is invalid. **Rejected (R) reply code** is returned if this command is issued while the IP-address of the DS is not properly configured (**DHCP (DH) setting** is 1 (enabled) and the DS hasn't yet obtained the IP-address from the DHCP server).

Source IP Filtering (SF) parameter

Description (see parameter description format info here)

Function:	Overrides **Source IP Filtering (SF) setting**
Parameter (P) cmd format:	**PSF*x***, where ***x***: 0 (disabled), 1 (enabled)
Possible replies:	**A**, **C**
Relevance conditions:	**Current Routing Mode (RM) [setting/ parameter]**= 0 (server) or 1 (server/client)

First introduced: V3.24/3.54

See also: Ethernet port and network communications

Details

Source IP Filtering parameter overrides the **Source IP Filtering (SF) setting**.

Error (C) reply code is returned if the data supplied in the command is invalid.

Destination IP-address (DI) parameter

Description (see parameter description format info here)

Function: Overrides **Destination IP-address (DI) setting**

Parameter (P) cmd format: **PDI*xxx.xxx.xxx.xxx***, where ***xxx.xxx.xxx.xxx*** is the IP-address of the destination in dot-decimal notation (i.e. 192.168.100.41)

Possible replies: **A**, **C**, **R**

Relevance conditions: For outgoing connections:
Current Routing Mode (RM) [setting/parameter]= 1 (server/client) or 2 (client)

[V3.24/3.54] For incoming connections:
Current Routing Mode (RM) [setting/parameter]= 0 (server) or 1 (server/client)
AND current Source IP Filtering (SF) [setting/parameter]= 1 (enabled)

First introduced: Earlier than "baseline" V3.14/V3.51, **functionality extended in V3.24/3.54**

See also: Ethernet port and network communications, **Establish Connection (CE) instruction**

Details

Destination IP-address parameter overrides the **Destination IP-address (DI) setting**. Notice that this setting's functionality has been extended in **V3.24/3.54**.

Error (C) reply code is returned if the data supplied in the command is invalid.
Rejected (R) reply code is returned if:

- Command is issued while the IP-address of the DS is not properly configured (**DHCP (DH) setting** is 1 (enabled) and the DS hasn't yet obtained the IP-address from the DHCP server).
- Current data connection state is *not* "closed". Destination IP-address cannot be changed while the data connection is in progress. Close (or shut) the connection using the **Close Connection (CC) instruction** or **Abort Connection (CA) instruction** first. Current connection status can be verified through **Echo (X) command** (***c*** flag).

Destination Port Number (DP) parameter

Description (see parameter description format info here)

Function:	Overrides **Destination Port Number (PN) setting**
Parameter (P) cmd format:	**PDP*ppppp***, where ***ppppp*** is the port number of the destination in the 0-65535 range
Possible replies:	**A**, **C**, **R**
Relevance conditions:	**Current Routing Mode (RM) [setting/ parameter]**= 1 (server/client) or 2 (client)
First introduced:	Earlier than "baseline" V3.14/V3.51
See also:	Ethernet port and network communications, **Establish Connection (CE) instruction**

Details

Destination Port Number parameter overrides the **Destination Port Number (DI) setting**.

Error (C) reply code is returned if the data supplied in the command is invalid.
Rejected (R) reply code is returned if:

- Command is issued while the IP-address of the DS is not properly configured (**DHCP (DH) setting**is 1 (enabled) and the DS hasn't yet obtained the IP-address from the DHCP server).
- Current data connection state is *not* "closed". Destination port number cannot be changed while the data connection is in progress. Close (or shut) the connection using the **Close Connection (CC) instruction** or **Abort Connection (CA) instruction** first. Current connection status can be verified through **Echo (X) command** (***c*** flag).

Establish Connection (CE) instruction

Description (see instruction description format info here)

Function:	Makes the DS establish the data connection with the network host
Parameter (P) cmd format:	**PCE** or **PCE*xxx.xxx.xxx.xxx/ppppp***, where ***xxx.xxx.xxx.xxx*** is the IP-address of the destination host; ***ppppp*** is the port number on the destination in the 0-65535 range
Possible replies:	**A**, **C**, **R**
Relevance conditions:	**Current Routing Mode (RM) [setting/ parameter]**= 1 (server/client) or 2 (client)
First introduced:	Earlier than "baseline" V3.14/V3.51
See also:	Ethernet port and network communications

Details

Establish Connection instruction makes the DS establish an outgoing connection

with the network host. Which transport protocol is used (UDP or TCP) is defined by the **current Transport Protocol (TP) [setting/ parameter]**.

This command has two syntax options: with and without destination IP-address and destination port number fields. Both fields must be either supplied or not supplied, it is not possible to have only one of the fields in the command body.

When destination IP-address and destination port number fields are supplied they are interpreted exactly as data fields in **Destination IP-address (DI) parameter** and **Destination Port Number (DP) parameter** i.e. override the **Destination IP-address (DI) setting**and **Destination Port Number (DP) setting**. This means that sending **PCE*xxx.xxx.xxx.xxx/ppppp*** command is equivalent to sending **PDI*xxx.xxx.xxx.xxx***, then **PDP*ppppp***, followed by the **PCE** command.

When destination fields are not provided the DS will use the following destination IP-address and port number to connect to:

- If, since the last time the **PDI** (**Destination IP-address (DI) parameter**) and **PDP** (**Destination Port Number (DP) parameter**) commands were issued the DS hasn't received any incoming connections from other network hosts then the DS will attempt to connect to the destination IP-address and port number defined by these recent **PDI** and **PDP** commands;
- If (since the most recent **PDI** and **PDP** commands) the DS has accepted an incoming data connection then the DS will *reread* the values of **Destination IP-address (DI) setting** and **Destination Port Number (DP) setting** and attempt to connect to the destination host defined by these settings.

TCP data connections are established in a normal way by initiating the SYN-SYN-ACK exchange with the network host. UDP data "connections" are established by sending a UDP datagram of zero length.

OK (A) reply code is returned if command is accepted. This does not mean that the data connection has been established already, only that the DS has accepted the command. Actual connection status can be verified at any time using the **Echo (X) command** (see ***c*** flag). Actual IP-address and port number of the destination host with which the DS has a data connection can be verified using the **Status (U) command**.

Error (C) reply code is returned if the data supplied in the command is invalid (or only one of the data fields is present). **Rejected (R) reply code** is returned if:

- Command is issued while the IP-address of the DS is not properly configured (**DHCP (DH) setting** is 1 (enabled) and the DS hasn't yet obtained the IP-address from the DHCP server);
- Current data connection state is *not* "closed". Close (or shut) the connection first using the **Close Connection (CC) instruction** or **Abort Connection (CA) instruction**;
- **Current Routing Mode (RM) [setting/ parameter]** is 0 (server), because in this state the DS is not allowed to establish outgoing connections.

Close Connection (CC) instruction

Description (see instruction description format info here)

Function:	Makes the DS close the data connection with the network host, reset buffer overflow flags
Parameter (P) cmd format:	**PCC**

Possible replies: **A**

Relevance conditions: **---**

First introduced: Earlier than "baseline" V3.14/V3.51

See also: Ethernet port and network communications, **Establish Connection (CE) instruction**.

Details

Close Connection instruction makes the DS close current data connection (if existed) and also reset routing buffers overflow flags (overflows are displayed by the status LEDs, overflow status is also returned by the **Echo (X) command**- see ***S*** and ***E*** flags).

As a result of **Close Connection instruction** execution UDP data "connections" are simply discarded. TCP data connections are properly shut down (using FIN-ACK-FIN-ACK sequence).

Close Connection instruction can be used at any time, even if the data connection is already closed and if the data connection was not established by the DS itself (i.e. it was a passive open).

OK (A) reply code is returned if command is accepted. This does not mean that the connection has been shut down already, only that the DS has accepted the command. Actual connection status can be verified at any time using the **Echo (X) command** (see ***c*** flag).

Another instruction- **Abort Connection (CA)**- can be used to abort the TCP connection (using the RST packet). For UDP data "connections" there is no difference between the **Close Connection instruction** and the **Abort Connection (CA) instruction**.

Abort Connection (CA) instruction

Description (see instruction description format info here)

Function: Makes the DS abort the data connection with the network host, reset buffer overflow flags

Parameter (P) cmd format: **PCA**

Possible replies: **A**

Relevance conditions: **---**

First introduced: Earlier than "baseline" V3.14/V3.51

See also: Ethernet port and network communications, **Establish Connection (CE) instruction**.

Details

Abort Connection instruction makes the DS end current data connection (if existed) and also resets routing buffers overflow flags (overflows are displayed by the status LEDs, overflow status also returned by the **Echo (X) command**- see ***S*** and ***E*** flags).

As a result of **Abort Connection instruction** execution UDP data "connections" are simply discarded. TCP data connections are reset (using RST packet).

Abort Connection instruction can be used at any time, even if the data

connection is already closed and if the data connection was not established by the DS itself (i.e. it was a passive open).

OK (A) reply code is returned if command is accepted. This does not mean that the connection has been reset already, only that the DS has accepted the command. Actual connection status can be verified at any time using the **Echo (X) command** (see ***c*** flag).

Another instruction- **Close Connection (CC)**- can be used to properly shut down the TCP connection (using the FIN-ACK-FIN-ACK sequence). For UDP data "connections" there is no difference between the **Abort Connection instruction** and the **Close Connection (CC) instruction**.

NetLoader (For Network Firmware Upgrades, V1.10)

The NetLoader is a separate firmware component that facilitated application firmware upgrades over the network in the following devices: EM100-03, DS100R-03, DS100B-00. Without the NetLoader the only way to upgrade the application firmware of these devices would be through the serial port. Serial upgrades are made possible by the Monitor, which is a "fixed" component that is always present.

The NetLoader is not used in newer EM120, EM200, EM203(A), DS203 device(s). The firmware of these devices has the ability to self-upgrade itself, so NetLoader is not required.

From the Monitor's point of view the NetLoader is just another application firmware. Therefore, the NetLoader can be loaded into the DS through the serial port of the DS. The NetLoader is written in such a way that it can coexist with an actual application firmware so both can be present in the FLASH memory of the DS at the same time.

To facilitate smooth interaction between the application firmware and the NetLoader the following provisions are made:

- The application firmware can verify the presence of the NetLoader and "jump" (pass control) to the NetLoader when needed. This way the application firmware can start its "self-upgrade".
- The NetLoader can receive new firmware file from the network host over the network, save this file into the FLASH memory of the DS, verify the presence of and a validity of the application firmware and "jump" (pass control) to this firmware.

To the User the above looks like a fully automatic self-upgrade of the application firmware.

Status LED Signals

Status LED signaling of the NetLoader is very simple. When the NetLoader is entered the Green Status LED is switched on. The LED remains on at all times except when the data block is being programmed into the FLASH memory. The Green Status LED is off for the duration of the data block programming. Visually this makes the Green Status LED blink during the firmware file upload into the DS.

MAC- and IP-address Under the NetLoader

When the NetLoader is started it attempts to read out the values of the **IP-address (IP) setting** and **MAC-address (FE) setting** used by the application firmware. If the readout is successful the NetLoader uses current values of these settings and this means that the DS continues running under the same IP and MAC that was just used by the application firmware.

Although all efforts have been taken to make sure that the DS continues running under the NetLoader using the same IP- and MAC-address there are few cases when the NetLoader may not be able to read out one or both of the above settings successfully (for example, when the data in the EEPROM memory that keeps setting values was corrupted).

The following happens in this situation:

- When the NetLoader is unable to read out the **IP-address (IP) setting** it assumes a default IP-address of 127.0.0.1.
- When the NetLoader is unable to read out the **MAC-address (FE) setting** it assumes a default MAC-address of 0.1.2.3.4.5.

NetLoader Communications

Communications with the NetLoader is effected by sending programming commands as UDP datagrams to port 65535 of the DS. For each command the NetLoader sends a reply, also in the form of a UDP datagram. Reply to a particular command is always sent to the IP-address and the port number from which this command was received.

Because each command and reply is sent in its own UDP datagram no additional encapsulation is necessary.

All NetLoader commands have the following format:

C.C.	Optional parameter(s)

- **C.C.** is the command code. Command code always consists of a single ASCII character. All available commands and their codes are listed in the command table at available commands.
- **Optional parameter(s)** field contains necessary data if required by the command.

All NetLoader replies have the following format:

R.C.	Optional data

- **R.C.** is the reply code. Reply codes always consist of a single ASCII character and inform the sender of the command about the result of command execution.
- **Optional Data** field contains necessary data (if any).

Example: here is a sample exchange between the network host and the DS running NetLoader. Each line represents the data in a separate UDP datagram.

Host-->DS(NetLoader): V
DS(NetLoader)-->Host: A<NV1.10 Netloader>

This method of exchanging commands and replies is exactly the same as the one

used by the application firmware of the DS where it is called out-of-band (UDP) programming. UDP communications with the application firmware and the NetLoader are the same except for the following small differences:

- Latest versions of the application firmware accept UDP commands on two ports- 65535 and 32767. The NetLoader can only accept UDP commands on port 32767.
- To speed up command execution the application firmware accepts multiple UDP commands (this is based on using additional command ID field). The NetLoader doesn't supports this feature and all commands should be sent strictly one-by-one i.e. the network host should not intentionally send the next command without first receiving (waiting for) a reply to the previous command.

Except for the above two differences UDP communications with the NetLoader is the same as UDP communications with the application firmware. Just like the application firmware, the NetLoader supports broadcadt UDP programming, which is initiated by the **Select In Broadcast Mode (W) command**. There are other commands that have similar or identical "parallel" commands in the application firmware- **Echo (X) command**, **Verify and Start Firmware (E) command**, **Get Firmware Version (V) command**.

UDP programming is called (in the Application Firmware Manual) out-of-band (UDP) programming because there is also inband (TCP) programming that relies on TCP communications with the DS. Because of size constraints the NetLoader doesn't support TCP communications and the only available method of exchanging commands and replies is the one based on UDP commands.

How the NetLoader is Started

The NetLoader is started in one of the two ways:

- If the application firmware is not loaded or corrupted the Monitor starts the NetLoader automatically when the DS is powered up (see DS startup procedure).
- When the application firmware is running remote host can make the application firmware jump (pass control) to the NetLoader by sending a **Jump To Netloader (N) command**.

In both cases the presence and validity of the NetLoader is verified before the control is passed to it.

The network host can verify whether the application firmware or the NetLoader is running by sending the **Echo (X) command [application firmware/ NetLoader]**. This command is supported by both firmware components but returns different replies. Specifically, there is an ***m*** (mode) flag that is set to 'N' by the application firmware but to 'L' by the NetLoader. _

Firmware Upload Procedure

The network host should use the following procedure in order to upload an application firmware file into the DS:

- First, it is necessary to verify if the NetLoader is already running. This is done by sending the **Echo (X) command [application firmware/ NetLoader]** and checking the ***m*** flag in the reply data (see how the NetLoader is started). If the NetLoader is not started yet then the following steps should be taken:

  - The network host should login (if not logged in yet) using the **Login (L) command**;
  - Next, the network host should send **Jump To Netloader (N) command**. This will start the NetLoader
  - It is better to verify that the NetLoader is, indeed, running, by sending another **Echo (X) command** and checking the status of the **m** flag.
- Once the NetLoader is started the network host should send **Start Over (Q) command** to reset the data upload process.
- Then, the network host should issue as many **Data Block (D) commands** as necessary to upload the entire firmware file.
- When the upload is completed the host should send the **Verify and Start Firmware (E) command** to start the newly loaded firmware. The host should wait about 2 seconds (to allow the firmware to start) before proceeding to the next step.
- As the last step of the procedure it is better to issue the **Echo (X) command** again to verify that the new firmware is, indeed, running.

The above procedure is, of course, simplified, as it does not take into account all kind of possible errors that can potentially arise practically on every step of the firmware upload. The "entity" performing the upgrade should carefully analyse the result of every step in the above procedure and correctly react to different error codes returned by the DS.

Also, the procedure above doesn't show the use of the **Select In Broadcast Mode (W) command** that provides a way to program the DS whose IP-address is unreachable (see broadcadt UDP programming for explanation). The NetLoader also support a similar command (see **Select In Broadcast Mode (W) command**). Important point to realize is that since the NetLoader is a separate firmware component the **Select In Broadcast Mode (W) command** should be used twice:

- The **W command** should be used to first time before sending the **Login (L) command** and **Jump To Netloader (N) command**. This will pre-select the DS for broadcast access while still in the application firmware.*
- The **W command** should be used the second time when the NetLoader starts, before starting the file upload (**Start Over (Q) command**, **Data Block (D) commands**, and **Verify and Start Firmware (E) command**)*.

** These commands should, of course, be sent as UDP broadcasts in this case.*

Available Commands

Table below lists all available NetLoader commands:

C.C.	B	Description
X	+	**Echo command**
W	+	**Select In Broadcast Mode command**
S		**Start Over command**
D		**Data Block command**
E		**Verify And Start Firmware command**
V		**Get Firmware Version command**

Notes:

- **C.C.**- command codes.

B- '+' in this column indicates that command can be issued in the broadcast mode (when sent through the network) without the need to pre-select a particular DS using **Select In Broadcast Mode (W) command** first.

Command Description Format

All commands in this section are described using the following format:

Function:	Command function in brief
Command format:	Shows command syntax
Possible replies:	Lists all possible reply status codes that can be returned in response to this command
See also:	Additional relevant links

Details

Additional information about the command.

Echo (X) command

Description (see command description format info here)

Function:	Returns NetLoader status
Command format:	**X**
Possible replies:	***Annn.nnn.nnn.nnn.nnn.nnn/ppppp/m***, where ***nnn.nnn.nnn.nnn.nnn.nnn***- MAC-address of the DS; ***ppppp***- fixed at "65535"; ***m***- fixed to 'L' (means that the NetLoader, not the application firmware is running).
See also:	How the NetLoader is started, **Echo (X) command**in the application firmware

Details

The primary use of the **Echo command** is to auto-discover Device Servers on the network and to verify that the NetLoader is running. When the network host sends this command in the broadcast mode, it collects the replies from all locally attached Device Servers (hence, the name of the command). Reply from each DS contains all necessary information (MAC-address, etc.) that is needed to continue communicating with each specific DS in a non-broadcast mode. This command has its counterpart in the application firmware (see **Echo (X) command**).

Information returned by the Echo command contains the following data:

- **MAC-address** is the most important field that can be used to uniquely identify each DS! Besides, the MAC-address is used (and, therefore, must be known in advance) as a reference to the particular DS in the **Select In Broadcast Mode (W) command**.
- **Data port number** field is fixed at 65535. The NetLoader doesn't use any data ports so this field is provided for format compatibility with the **Echo (X) command** in the application firmware.
- ***m* flag** always returns 'L'. This is meant to indicate that the NetLoader is running. In contrast, when the application firmware is running this flag shows

'N'.

** This is because each DS, like any other Ethernet device, has a unique MAC-address preset during the production.*

Select In Broadcast Mode (W) command

Description (see command description format info here)

Function:	Selects the DS as the target in broadcast communications
Command format:	**A*mmm.mmm.mmm.mmm.mmm.mmm***, where ***mmm.mmm.mmm.mmm.mmm.mmm***- MAC-address of the target DS
Possible replies:	**A*vv...v***, where ***vv...v*** is the version string
See also:	Firmware upload procedure, Broadcast out-of-band (UDP) programming, **Select In Broadcast Mode (W) command** of the **application firmware**

Details

Select In Broadcast Mode command has exactly the same function as the **Select In Broadcast Mode (W) command** of the **application firmware**.

This command is used to pre-select a certain DS for subsequent firmware upload. Only few NetLoader commands (such as **Echo (X)**) are accepted when sent in broadcast UDP datagrams. All other commands are only accepted if they address a specific DS. Such specific addressing normally involves sending UDP datagrams with the IP-address of the targeted DS as the destination (i.e. non-broadcast datagrams). This requires the IP-address of the DS to be reachable.

Select In Broadcast Mode command provides a way around this. Target DS, referenced by its MAC-address, is first pre-selected using this command. After that, all broadcast commands that are normally ignored when sent as broadcasts, are not ignored and processed by this pre-selected DS.

When **Select In Broadcast Mode command** is issued all devices whose MAC-addresses do not match the target MAC-address supplied in the command body de-select themselves. This means that to switch onto working with another DS in the broadcast mode you need to send the new **Select In Broadcast Mode command** with the new target MAC-address. This will pre-select a different DS while at the same time de-selecting the DS that was selected before. To de-select all DS on the network send **Select In Broadcast Mode command** with no MAC-address field.

This command only influences which DS responds when it is addressed using broadcast UDP commands. Command has no influence over commands sent using normal IP addressing.

The only possible reply to this command is **OK (A)**. It is issued by the DS that has recognised its MAC-address in the command body. If no DS on the local network recognizes its MAC then there will be no reply received to this command.

Select In Broadcast Mode command should be issued (when necessary) right after the NetLoader is started as all subsequent commands, such as **Start Over (Q)**, **Data Block (D)**, and **Verify And Start Firmware (E)** won't work when send as broadcasts without prior pre-selection.

It is important to understand that **W command** should be used within the NetLoader even if it has already been used in the application firmware before the NetLoader was started. This is because the application firmware and the NetLoader are separate firmware components and the pre-selection made in the application firmware will not be "passed" to the NetLoader.

Start Over (Q) command

Description (see command description format info here)

Function:	Initializes (reset) firmware file upload
Command format:	**Q**
Possible replies:	**A**
See also:	Firmware upload procedure

Details

Start Over command is used to reset the firmware upload. This resents internal data block counter of the NetLoader and the first **Data Block (D) command** will be expected to carry data block 0.

This command is recommended to be used after the NetLoader is started to make sure that the upload process is properly initialized.

Data Block (D) command

Description (see command description format info here)

Function:	Supplies one 128-byte block of firmware file to the NetLoader
Command format:	**D*nndd...d***, where ***nn***- block number (binary format, two bytes, high byte first); ***dd...d***- 128 bytes of file data.
Possible replies:	**A*nn***, **C*nn***, **S*nn***, **O*nn***, **I*nn***, **F*nn***, where ***nn*** is the number of the next expected data block (binary format, two bytes, high byte first)
See also:	Firmware upload procedure

Details

Data Block command carries a 128-byte block of application firmware file. Entire application firmware file is to be uploaded using multiple Data Block commands and the NetLoader expects that the firmware file size will consist of an integer number of 128-byte blocks.

The data blocks must be sent in consecutive order i.e. the first block (block 0) must carry first 128 bytes of the firmware file, block 1 should supply next 128 bytes, etc. Because NetLoader communications is based on UDP commands and UDP packets can get lost each data block is sent with its own block number. This way the NetLoader can make sure that all data blocks received are in sequence.

The ***nn*** field in the command body carries the block number. The block number is supplied in a binary form. Since block numbers can exceed 255 two bytes are needed for the block number. High byte goes first so, for example, to denote block

#300 the first two bytes should be 01H 2CH. Block numbers start from 0.

Following two block number bytes are 128 bytes of firmware file data. Again, the data is supplied in a binary form, that is, a part of the firmware file is "cut out" and attached to the command.

Like all other commands, the **Data Block command** returns an appropriate reply status code. The difference is that all reply status codes are followed by the number of the next expected data block. Next data block is present even in replies indicating that the command was not processed successfully. In this case the next expected data block simply remains the same (is not incremented). If the command was completed successfully the ***nn*** field in the reply will "point at" the next data block to be sent. For example, if the ***nn*** field in the command was 300 (01H 2CH) and the command was completed successfully the reply will have the ***nn*** field of 301 (01H 2DH). If command failed the ***nn*** field in the reply will still be at 300 (01H 2DH).

OK (A) reply status code is returned if the data block was programmed successfully *or* if the Data Block command has carried a data block that has already been programmed. That is, if the NetLoader expects the block number to be 10 and the network host supplies block 5 the NetLoader will reply with **A code** and discard this data block.

Error (C) reply status code is returned if the total length of ***nn*** and ***dd...d*** fields in the command is not 130 bytes.

Sequence Error (S) reply code is returned if the NetLoader receives the data block with ***nn*** field greater than the one expected (for example, block 7 when block 6 was expected).

Out-of-range (O) reply code is returned if the total size of the firmware file received by the NetLoader exceeds the size detected in the beginning of the file upload. Total file size information is contained in the firmware file itself (in block 1). For example, if the file size was expected to be 40064 bytes then the total number of data blocks had to be 40064/128=313. Therefore, the range of expected data block numbers is 0...312. The NetLoader will return the **O code** if any data block with the number greater than 312 is received.

Invalid Data (I) reply code is retuned if the NetLoader detects that the file data sent by the network host is invalid. This can only be generated for data blocks 0 and 1. These blocks contain certain "service" information (such as the total file size- see above) that the NetLoader checks to make sure that the firmware file being uploaded is acceptable.

Failed (F) reply code is returned if the NetLoader fails to program the data block into the FLASH memory of the DS.

We recommend that the network host sends the **Start Over (Q) command** before beginning file upload with the **Data Block commands**.

Verify And Start Firmware (E) command

Description (see command description format info here)

Function:	Verifies and starts the application firmware
Command format:	**E**
Possible replies:	**I**
See also:	Firmware upload procedure, **Reboot (E)**

command of the application firmware

Details

Verify And Start Firmware command instructs the NetLoader to verify the checksum of the application firmware and, if OK, to start application firmware execution. This command is similar to the **Reboot (E) command** of the application firmware in that it ends the execution of the current firmware (NetLoader).

Invalid (I) reply code is returned if the checksum is wrong and the NetLoader continues running. No reply is returned if the checksup is OK, the NetLoader simply passes control to the application firmware.

This command should be used as the last step of the firmware upload through the network.

Get Firmware Version (V) command

Description (see command description format info here)

Function:	Returns the version of this NetLoader
Command format:	**V**
Possible replies:	**A*vv...v***, where ***vv...v*** is the version string
See also:	**Get Firmware Version (V) command** in the application firmware

---

Details

Get Firmware Version command returns the version of this NetLoader. This command is similar to the **Get Firmware Version (V) command** of the application firmware.

The version string is always encapsulated in '<' and '>', begins with the version number in **NV*x.xx*** format, and possibly contains a small comment after a space. "N" in front stands for "NetLoader".

Example:

-->DS: **V**
DS-->: **A<NV1.10 Netloader>**

Software Manuals

This part of the documentation describes all PC software supplied by Tibbo.

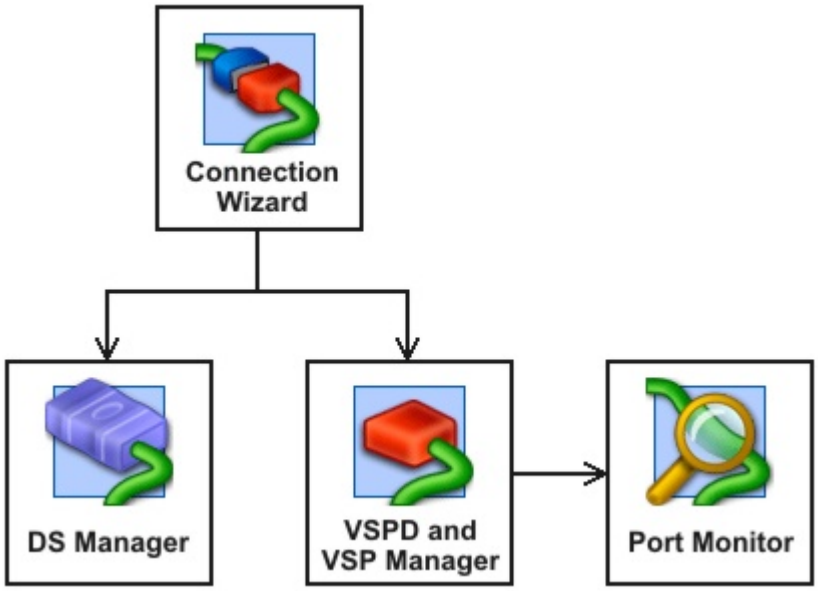
At the moment, two software packages are available:

- Device Server Toolkit (DST) software for Windows (revision history).
- Virtual Serial Port Driver for LINUX (VSPDL).

Device Server Toolkit (DST) Software for Windows (Release 3.9.82)

Looking for Device Server Toolkit (DST) revision history? Click here

Tibbo Device Server Toolkit (*DST*) for *Windows* is supplied free of charge with all Device Servers manufactured by Tibbo. The *DST* runs under all versions of *Windows** starting from *Windows 98*. At the time of writing this includes *Windows 98*, *Me*, *NT(SP4)*, *2000*, *XP*, *2003 (server)*.



The Device Server Toolkit includes five components:

- **Device Server Manager** (*DS Manager*) is used to locate, setup, manage, monitor, and upgrade Tibbo Device Servers. The *DS Manager* works with all Tibbo Device Servers, regardless of their model number and internal firmware version.
- **Virtual Serial Port Driver*** (*VSPD*) powers **Virtual Serial Ports** (*VSPs*) that are used to network-enable legacy serial systems. To any *Windows* application the *VSP* "looks and feels" just like any standard COM port, while in reality the *VSP* is transparently routing the data between the application and the serial port of the Device Server.
- **Virtual Serial Port Manager** (*VSP Manager*) is used to add, delete, and setup *VSPs*. The *VSPD* and the *VSP Manager* are different entities (one is a *driver*, another one- a *setup utility*). This *Manual* offers combined description of the two and all *VSPD* features are explained "through" the *VSP Manager* and its dialogs.
- **Port Monitor** is a supplementary program that logs the activity of *VSPs* on your system. Every action of the *VSP* (port opening, port closing, establishing connection with a particular Device Server, etc.) is recorded by the *Port Monitor*. This makes the *Monitor* an indispensable debugging tool that can be used to pinpoint *VSP*-related problems.
- **Connection Wizard** assists you in setting up typical data links involving Tibbo Device Servers and *Virtual Serial Ports*. Although the description of the *Connection Wizard* comes last in this *Manual* this is the program you should probably try out *first*. Chances are, this will be *the only* program of the *DST* you'll ever use!

** Windows is a registered trademark of Microsoft Corporation.*

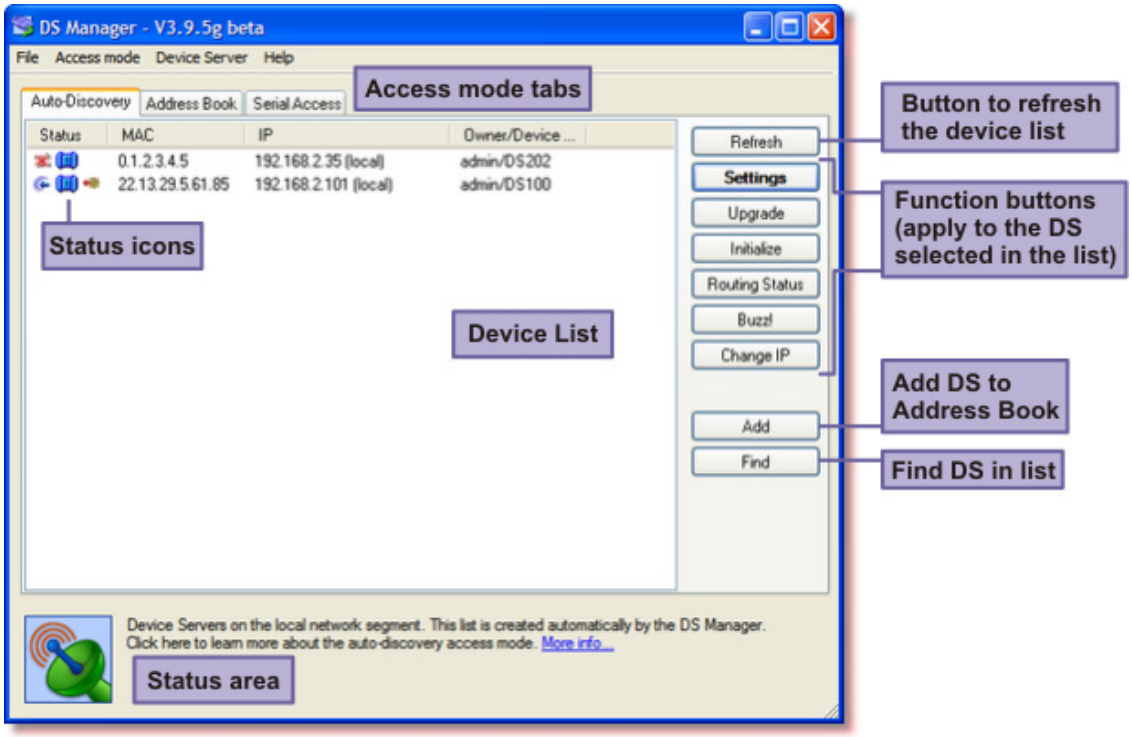
** Also known as "COM redirector".*

DS Manager

Device Server Manager (*DS Manager*) is a part of the Device Server Toolkit.

The *DS Manager* is used to locate, setup, manage, monitor, and upgrade Tibbo Device Servers ("DS").

Shown below is the screenshot of the *DS Manager's main window*. Click on the picture area to jump to the related topic or select the topic from the list under the screenshot.



The *main window* has the following areas and controls:

- **Access mode tabs** provide a selection of three different access modes (auto-discovery, address book, and serial). Access modes define how the *DS Manager* locates and accesses the Device Servers.
- **Device list**. The data shown in the *device list* depends on the selected access mode. The status icons displayed in the auto-discovery and address book access modes reflect the current state of each DS in the list.
- **Status area.** When you select a different access mode this area displays a brief mode introduction. Single-clicking on a particular DS in the *device list* displays

extended information about the current status of this DS (auto-discovery and address book access modes only).

- **Refresh button** is used to refresh the *device list* (auto-discovery and address book access modes only)
- Actions of the following *function buttons* apply to a single DS (or COM port to which this DS is connected) selected in the *device list*:
  - **Settings button** opens the *settings dialog* that allows you to view and edit the settings of a particular DS;
  - **Upgrade button** launches the upgrade of the internal DS firmware;
  - **Initialize button** is used to initialize the settings of the DS to their default values;
  - **Routing Status button** (auto-discovery and address book access modes only) opens the *routing status dialog* that displays additional information on the current DS state (data connection state, amount of data in the routing buffers, serial port setup, etc.);
  - **Buzz button** (auto-discovery and address book access modes only) causes the status LEDs of the DS to "play" a fast-blinking pattern. This can be used to match a particular entry in the *device list* to an actual physical device;
  - **Change IP button** (auto-discovery mode only) is used to assign new IP-address to the DS over the network (using MAC-address of the DS as a reference).
- **Add, Remove, Edit and Groups buttons** are used to edit the address book (address book is a manually created list of Device Servers in the address book access mode) and to manage device groups. *Add button* is visible both in the auto-discovery and address book access modes, while *Remove, Edit* and *Groups* buttons are only visible in the address book access mode.
- **Find button** is used to find a DS on the list, and is visible both in the auto-discovery and address book access modes.

DS Status Icons

Status field of the *device list* (available in the auto-discovery and address book access modes) displays a **status icon** for each DS. The status icon reflects current DS state and is updated each time the *Refresh button* is pressed. Listed below are all DS states that may be reflected in the *device list*.

The following two states can only be displayed in the address book mode:

--- **When no icon is displayed** for an address book entry then this means that there was no response to the PING sent by the *DS Manager*. This indicates that there is no device whatsoever at the IP-address specified by the address book entry. For more information see troubleshooting (address book mode).

Unknown device. This icon means that there was a response to the PING sent by the *DS Manager* (which proves that there is at least *some* device at the specified IP-address) but that the *DS Manager* cannot make sure that this is really a Tibbo Device Server. For more information see troubleshooting (address book mode).

Note. The reason why the above two states can only be encountered in the address book access mode is because in the auto-discovery mode the device list only displays Tibbo Device Servers that have actually responded to the refresh.

All remaining icons may be displayed both in the auto-discovery and address book access modes.

The status icon consists of three parts:

- **The central part** depicts the DS and reflects its general status and well-being:


No status info available. The DS is running old firmware (2.xx or older) and the status information cannot be obtained remotely.


Normal state. The DS is online and appears to function properly.


Error mode. The DS is running in the error mode and requires initialization;


IP-address not obtained. The DS is online but hasn't yet obtained its IP-address from the DHCP server (when the **DHCP (DH) setting** is 1 (enabled)). In this state the DS is not performing its data routing function. If the DS is also in the error mode (see above) at the same time then it is the error mode status that will be shown by the icon;


Firmware upload mode. The DS is in the firmware upload mode and is ready to accept new firmware file. If the DS enters this mode right after the powerup then this means that no firmware is loaded or that the firmware is corrupted.

- **Left part** of the icon shows current data connection status:

--- **Idle.** No data connection is established, the DS is idle (so no icon is displayed);


ARP. The DS is sending ARP requests in order to find the MAC-address of the destination network host (or gateway) before attempting to establish a data connection;


Opening. TCP data connection is being established. This icon cannot be displayed for the UDP transport protocol since there is no connection establishment phase for UDP data "connections";


Established (or being closed), no overrun. TCP data connection or UDP UDP data "connection" is established or TCP connection is being closed (there is not connection closing phase for UDP). Routing buffer overflow is not detected (within current data connection).


Established (or being closed), overrun detected. Same as the above but routing buffer (Ethernet-to-serial and/or serial-to-Ethernet) overflow has been detected.


Reset. TCP data connection has been reset by the network host (not the DS itself). This icon cannot be displayed for UDP data "connections".

- **Right part** of the icon displays current programming status:

--- **No programming.** The serial port of the DS is not in the serial programming mode and no network programming session* is opened;


Programming in progress. Either the serial port of the DS is in the serial programming mode or network programming session* has been opened.

Central, left, and right icon parts described above are combined into a single status icon.

Example: the following "combination" icon means that the DS is running in the error mode, data connection is currently established (but no overrun has been detected), and some form of programming (either serial or network) is in progress:

In addition to different states described above the whole status icon can be displayed in full color or grayed (see sample icons below). This only applies to local Device Servers and the auto-discovery mode).

Full color. This means that the *DS Manager* can communicate with the DS using "normal" IP addressing.

Grayed****.** When the status icon is grayed then this means that the *DS Manager* can see the DS but cannot communicate with the DS using normal IP-addressing. Full details on what this means are provided in the following topics: broadcast access, troubleshooting (auto-discovery mode).

Additional (and more full) status information for the selected DS (i.e. the DS whose line is highlighted in the *device list*) is displayed in the *status area* below the *device list*. For example, while the status icon may show that some sort of programming is in progress the *status area* message will detail that the "UDP network programming session is in progress". Each status message has a clickable link that opens a corresponding help topic (all such topics can be found at DS status messages).

Programmer's info:

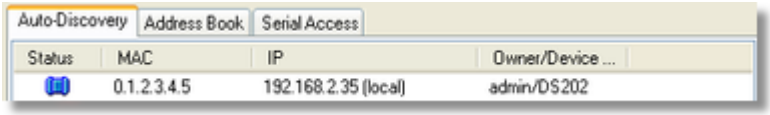
The *DS Manager* collects DS status information (and detects DS presence) using the **Echo (X) command**.

** More info on network programming sessions can be found in the following topics: authentication, programming priorities.*

*** Actually, "watered down" as colors can still be distinguished.*

Access Modes

The *DS Manager* has three fundamental modes of operation, called *access modes*, that define how it finds, connects to and works with Device Servers. Desired access mode is selected from the *access mode drop-down box*, located at the top of the *main window* of the *DS Manager* (see figure below):



Available access modes are:

- **Auto-Discovery** In this mode the *DS Manager* automatically finds all local Device Servers* and displays them in the *device list* (*remote* Device Servers located behind one or several routers cannot be auto-discovered). Device list is updated each time *Refresh button* is pressed, only devices that have actually responded to the refresh appear in the list.
- **Address Book** In this mode the list of available Device Servers- called "the address book"- is created manually by the User and the *DS Manager* does not attempt to locate Device Servers automatically. Device list displays all entries from the address book at all times, even if corresponding Device Servers are not online. Actual availability of each Device Server is reflected in the status icon for this DS. In this mode the *DS Manager* can communicate both with local and remote Device Servers (i.e. located behind the routers on different network segments)**. Since local Device Servers can easily be accessed using the auto-discovery mode the primary use of the address book mode is to access remote Device Servers.
- **Serial Access** In this mode the *DS Manager* communicates with the DS connected (by its serial port) to the COM port of the PC. Device list displays all available physical COM ports of the PC and the *refresh button* is hidden. Since the DS is accessed through its serial port it must be in the serial programming mode for this access method to work. Therefore, the User must press the setup button*** to make the DS enter the serial programming mode before the *DS Manager* will be able to access the DS.

Note:

Status LEDs of the DS are playing a *serial programming mode pattern* (shown on the left) when the serial port of the DS is in the serial programming mode (click here to see all available patterns).

** I.e. the Device Servers located on the same network segment. The definition of the network segment implies that there are only network hubs (and no routers, bridges, firewalls, etc.) between the PC and all other devices on the segment.*

*** Ability to access remote Device Servers also depends on the configuration of the network, routers, and Device Servers.*

**** On EM100, EM120, EM200, EM203(A)- pull the MD line LOW for at least 100ms.*

Auto-Discovery Access Mode

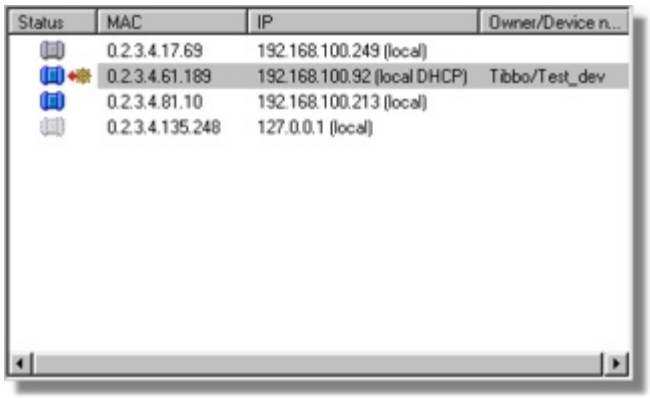
Can't find your Device Server in the *device list* of the auto-discovery mode?
The status icon is grayed indicating that the IP-address of this DS is "unreachable"?
See troubleshooting!

Auto-discovery access mode is selected by choosing "Local Device Servers (Auto-discovery by broadcast)" from the *access mode drop-down box*, located at the top of the *DS Manager's main window*.

In this mode the *DS Manager* finds all local Device Servers* automatically. Device Servers located behind the routers (remote Device Servers) cannot be auto-discovered by the *DS Manager* because broadcast UDP datagrams used to find the Device Servers on the network cannot pass through routers. Address book access mode should be used to work with remote Device Servers.

All discovered Device Servers are displayed in the *device list*. Each DS is uniquely identified by its MAC-address which is different for every DS manufactured. The *device list* will correctly display all local Device Servers even if some of them have the same IP-address, as well as devices whose IP-address is unreachable. Correctly configured IP-address is *not required* for the *DS Manager* to be able to access local Device Servers in the auto-discovery mode (see broadcast access for more information).

Device list is updated each time *Refresh button* is pressed. Only Device Servers that have replied to the *DS Manager's* request appear in the list, the *DS Manager* does not "remember" previous refresh results. Status area under the *device list* displays extended status information for the DS selected in the *list*.



In the auto-discovery mode the *device list* has the following fields:

- **DS status icon**. See DS status icons for more information.
- **MAC-address** of the DS. It is the MAC, not the IP-address, that uniquely identifies each DS in the list.
- **IP-address** of the DS. Comment next to the IP-address shows that this device is local (same for all devices in the list since auto-discovery mode only shows local Device Servers). If the DS is running with DHCP enabled the note "DHCP" is also displayed next to the IP-address (as in line 2 of the screenshot above). As you can see from the screenshot, correct IP-address is not required for the DS to be discovered. IP-address in line 4 is invalid yet the *DS Manager* can "see" this DS as well.
- **Owner name and device name** fields displays the data from the **Owner Name (ON)** and **Device Name (DN)** settings of the DS. The purpose of these settings is to simplify distinguishing between the Device Servers in the *device list*.

Device list of the auto-discovery mode is sorted by MAC-address, then by IP-address. You can change the sorting order by clicking on any of the headers.

Programmer's info:

When the *Refresh button* is pressed the *DS Manager* sends out a broadcast **Echo (X) command**. This command reaches all local Device Servers, even the ones with unreachable IP-address. Devices shown in the *device list* are those that have replied to the broadcast

** I.e. the Device Servers located on the same network segment. The definition of the network segment implies that there are only network hubs (and no routers, bridges, firewalls, etc.) between the PC and all other devices on the segment.*

Broadcast Access

In the auto-discovery access mode the *DS Manager* can detect and access all local Device Servers, even those whose IP-address is "unreachable" (incompatible with the network settings of the PC, conflicts with the same IP-address set on another Ethernet device, etc.).

Device Servers with unreachable IP-address are accessed using broadcast communications (broadcasts do not depend on the IP-address of the device in any way)*. The downside of this method is that broadcasts cannot pass through the routers so only local Device Servers can be accessed this way (hence, the method is only used in the auto-discovery mode).

Devices with unreachable IP-address are displayed in the *device list* with their status icon "grayed" (see sample icon on the left). For complete list of all icons please turn to DS status icons.

Note, that conditions that cause the IP-address of the DS to appear to be unreachable may be temporary. See troubleshooting (auto-discovery mode) for details.

Programmer's info:

The *DS Manager* determines which Device Servers can be accessed using "normal" IP-addressing (and which- cannot) by performing the following procedure:

- When the *Refresh button* is pressed (in the auto-discovery mode) the *DS Manager* sends out a broadcast **Echo (X) command**. This command reaches all local Device Servers, even the ones with unreachable IP-address. Devices shown in the *device list* are those that have replied to the broadcast.
- Next, the *DS Manager* sends out non-broadcast **Echo (X) commands** directly addressing each DS in the *device list* (so as many commands are sent as there are devices detected in the previous step). If there is a reply to a command addressing a particular DS then the *DS Manager* considers the IP-address of this device to be reachable. Otherwise, the *DS Manager* marks this DS as having an unreachable IP-address and accesses it using broadcasts.

The ability to access and program Device Servers using broadcast communications is delivered through the use of the **Select In Broadcast Mode (W) command**. You can also read about how this works in the following topic: Broadcast out-of-band commands.

** Broadcast programming is not supported by Device Servers with the older firmware (earlier than V3.xx). Such Devices must be assigned a reachable IP-address before they can be accessed by the DS Manager.*

Troubleshooting (Auto-Discovery Mode)

This topic provides a list of hints that can help you figure out why you cannot "find" your Device Server in the device list of the *DS Manager* or why the DS is shown as having an unreachable IP-address (status icon is "grayed"). The topic only covers the auto-discovery access mode. See troubleshooting (address book mode) and troubleshooting (COM mode) for hints on other two access modes.

Here is why you may not be able to see the DS in the *device list*:

- The DS may be switched off or not connected to the network*.
- The DS may not be on the local network segment. In the auto-discovery access mode the *DS Manager* can only "see" Device Servers on the same network segment as your PC (local Device Servers). Remote Device Servers located behind routers (firewalls, etc.) cannot be accessed. Use address book access mode to work with remote Device Servers.
- Firewall software on your PC may be preventing communications between the *DS Manager* and Device Servers. This communications relies on UDP traffic between port 65534 of the PC and port 65535 of the Device Servers. Make sure that your firewall software does not ban this traffic! Our experience shows that many Users are not aware of the firewall software installed on their own PCs! Some firewalls come as part of larger "protection" suites (anti-virus, anti-intrusion, etc. programs). Some operating systems, such as *Windows XP*, include the firewall software too!
- If your DS is running an older firmware (V2.xx) then it will not be accessible through the network and visible to the *DS Manager* in the following cases:

When the DS is in the serial programming mode (status LEDs of the DS "play" the pattern as shown on the left)**;

When the DS is in the error mode (status LEDs of the DS "play" the pattern as shown on the left)**.

This *does not* apply to the Device Servers running DS firmware V3.xx or higher. In this firware the DS is visible on the network at all times.

- If there are many devices shown in the *device list* the DS you are looking for may actually be in the list! A very useful buzz feature allows you to match *device list* entries to their actual physical Device Servers. Use this function and you will probably "find" your DS (buzz feature is only supported by DS firmware V3.xx or higher).

Here is why the DS may be shown in the device list as having an unreachable IP-address:

- Because the IP-address of the DS is really unreachable. This means that it is outside of the IP-address range defined for the subnet to which your PC is connected. For example, if the IP-address of your PC is 192.168.100.30 and the netmask is 255.255.255.0 then your local subnet has the IP-address range from 192.168.100.1 to 192.168.100.254***. So, if you connect the DS to the local network segment and set its IP-address to 192.168.100.40 then the *DS Manager* will be able to access this DS. If you set the IP-address to 192.168.10.40 then your PC will decide that this IP-address is on some other subnet and will relate all communications (related to this IP) to the "default gateway". This means that network packets addressing this DS won't even be released into the network to which the DS is physically connected.

- Because the *DS Manager* has cached a different MAC-address for the IP-address currently assigned to the DS in question. This can happen if you connect different Device Servers one by one to the same network segment and then assign the same IP-address to each Device Server. This is often reported by our Users that want to pre-program or test several Device Servers. What happens here is that after the first DS is detected/accessed by the *DS Manager* the PC finds out the MAC-address that corresponds to the IP-address of the DS****. If you quickly connect another DS in place of the first one and assign the same IP-address to it the PC won't bother to "resolve" the IP-address again and will attempt to communicate with the "cached" MAC-address instead. Since MACs are unique for each DS the *DS Manager* won't be able to communicate with the new DS using "normal" IP-addressing. This problem will be resolved automatically after IP-MAC mapping data kept by the PC expires (about 20 minutes later). If your task is to pre-program/test several Device Servers then you don't have to wait until this happens- the *DS Manager* automatically chooses broadcast access for Device Servers that (temporarily) cannot be addressed using their IP-address.
- Because the IP-address of the DS is the same as the IP-address of some other device on the network segment. The best way to find out if this is so is to switch the DS off and try to PING the IP-address in question. If you still get PING replies then this means there is some other network device that is using this IP!
- Because of the firewall software installed on your PC. We have seen cases when the *DS Manager* could discover the Device Servers (broadcast UDP communications to port 65535 were allowed) but could not address these devices using normal IP addressing (non-broadcast UDP communications to port 65535 were blocked). When this is the cause of the problem all of your devices will be seen in the *DS Manager* with grayed icons.

** Really sorry for this "hint"... it's just that this is the root cause of many problems!*

*** See status LED signals for the full list of all available patterns.*

**** .0 and .255 are not allowed in principle.*

***** MAC-address is an actual physical network address used by all Ethernet devices. When connecting to the target device with a specified IP-address the "connectee" first finds out what MAC-address corresponds to the targeted IP-address. This way IP-addresses are mapped onto the MAC-addresses. This process is called "address resolution" and is performed using a special Address Resolution Protocol (ARP). Mapping information is memorized ("cached") by the PC and is remembered for about 20 minutes. The information is reused until it expires. This means that instead of using the ARP again the PC simply uses cached data instead.*

Address Book Access Mode

Can't access your Device Server in the address book mode?
The device list shows "no response" or "unknown device" status?
See troubleshooting!

Address book access mode is selected by choosing the "Address Book" tab in *DS Manager's main window*.

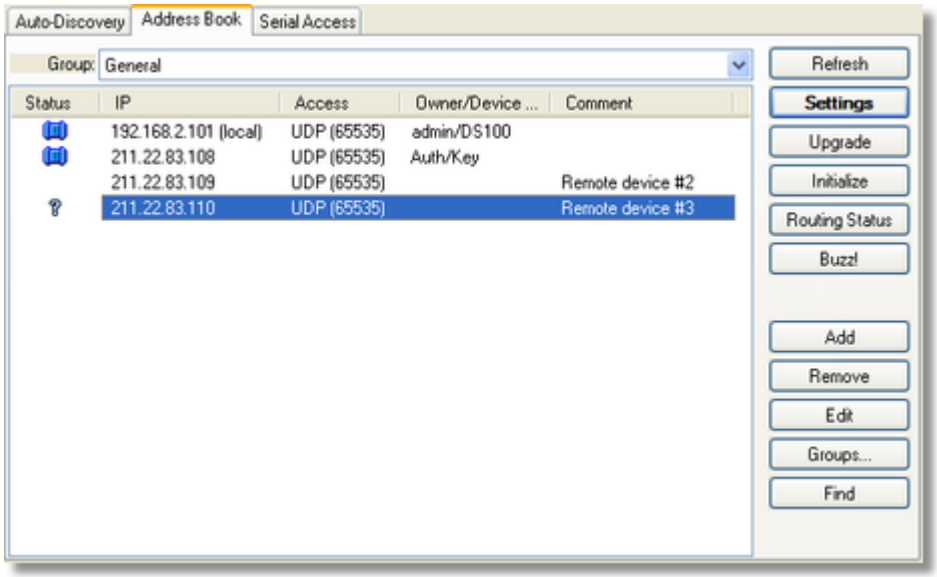
In this mode the *DS Manager* displays a manually created list ("the address book") of Device Servers. Each entry in the list specifies the IP-address (and port number) at which the DS is (supposed to be) located or through which the DS can be

accessed. The address book is edited using three buttons - Add, Remove, and Edit. You can also divide the book into Device Groups, which can be managed via the Groups button.

Address book mode allows the *DS Manager* to communicate both with *local* and *remote* Device Servers (i.e. located behind the routers)*. Since local Device Servers can easily be accessed using the auto-discovery access modethe primary use of the address book mode is to access remote Device Servers.

Modern routers offer a bewildering array of routing arrangements and security (firewall) options. To adjust for all this complexity the address book mode provides a flexibility of defining access parameters separately for each entry in the list. This includes, apart from the IP-address, the access method and the access port number.

In the address book mode the *device list* always displays all entries from the currently active group in the address book (this is different from the auto-discovery mode which only shows online devices). Each time *Refresh button* is pressed the *DS Manager* verifies which Device Servers are online and reflects this by displaying an appropriate icon for each entry in the list. *Status area* under the *device list* displays extended status information for the DS selected in the *list*.



In the address book mode the *device list* has the following fields:

- **Group listbox** allows you to select and manage Address Book device groups. See Managing Address Book Groups (Groups button).
- **DS status icon**. See DS status icons for more information. Since the address book consists of *addresses*, not *actual devices*, there is no guarantee that there is definitely a functioning DS at each specified location. On the sample screenshot above no device is found at location specified by line 3 (no status icon displayed) and unknown device (not a Tibbo Device Server) is found at location specified by line 4 (is displayed).
- **IP-address** of the DS itself or the forwarding IP-address on the router through which this DS can be accessed (this depends on the router setup**). Comment next to the IP-address shows whether the DS is local (otherwise it is remote). If the DS is running with DHCP enabled the note "DHCP" is also displayed next to the IP-address.
- **Access method and programming port number** of the DS itself or the forwarding port on the router through which this DS can be accessed (this

depends on the router setup**). Access parameters for the address book mode topic provides detailed info on the subject.

- **Owner name and device name** fields displays the data from the **Owner Name (ON)** and **Device Name (DN)** settings of the DS. The purpose of these settings is to simplify distinguishing between the Device Servers in the *device list*.
- **Comment field** (unlike owner name and device name) is associated with the address book entry and is stored on the PC.

Device list of the address book mode is not sorted and is kept in the original order of entry. You can sort the list by clicking on the field headers (Status, IP, Access, etc).

Programmer's info:

In the address book access mode broadcast commands are not used. Clicking *Refresh button* makes the DS "poll" each address book entry individually. Polling consists of sending a PING and (if there is a reply to the PING) an **Echo (X) command**. If there is no reply to the PING no response status is displayed for this DS. If there is a reply to the PING but no reply to the **Echo (X) command** then the *DS Manager* assumes that this is an unknown device.

Notice that address book entries with inband access method are not polled at all when you click *Refresh* (see access parameters for the address book mode).

** Ability to access remote Device Servers also depends on the configuration of the network, routers, and Device Servers.*

*** Here we touch on a very complicated subject. Modern routers offer a bewildering array of setup options. See AN009. WAN Basics for further details.*

Access Parameters for the Address Book Mode

Modern routers offer a bewildering array of routing arrangements and security (firewall) options*. To adjust for all this complexity the address book mode provides a flexibility of defining access parameters separately for each entry in the list. This includes, apart from the IP-address, an **access method** and an **access (command) port number**.
Both parameters are set in the address book entry *dialog* (see editing the address book).

The access method defines how the *DS Manager* communicates with the DS:

- When **out-of-band UDP access method** is selected the *DS Manager* communicates with the DS using command UDP datagrams sent to **command port** 65535 (or 32767) of the DS. Out-of-band UDP commands is a primary way of network programming that requires no preliminary setup on the DS side. In other words, it always works as long as the routers and firewall do not ban UDP traffic to port 65535 (32767). Therefore, we suggest you to always use this access method whenever possible. The *DS Manager* allows you to input any port number but actually only two UDP command ports are provided on the DS- 65535 and 32767 (either one can be used).
- When **inband TCP access method** is selected the *DS Manager* communicates with the DS by sending commands through the TCP connection established to the data port of the DS (this is the port number defined by the **Data Port Number (PN) setting**). Inband TCP commandsprovide a secondary method of network programming that can be used in situations when out-of-band UDP access is impossible. You can use any access port number- this is an additional

flexibility- just make sure that you program the same port number on the DS and in the address book entry. There are several limitations associated with inband access:

- For "inband" address book entries the *DS Manager* does not automatically verify DS presence when the *Refresh button* is pressed. Therefore, after the *Refresh button* is clicked all Device Servers for which the inband access is defined initially appear in the list with the ? icon. If you want to check if such a DS is accessible then attempt to execute the desired action: press Settings button or Buzz button and see the result (Upgrade, Initialize, and Routing Statusfunctions are not available for "inband" address book entries). At this point the *DS Manager* will attempt to establish a TCP connection to the DS and find out if the DS is accessible. This will be reflected in the *device list*.
- For the inband access to work the DS must be preset in a certain way first- read preparing the DS for inband access for step-by-step instructions.
- It is not possible to access the DS using inband access method when the DS is already engaged in a data TCP connection with another application and/or network host. Since inband commands are, by definition, the commands that are passed within the data connection itself the *DS Manager* actually establishes a data TCP connection with the DS in order to program it (this is why this connection is made to the data port of the DS, not the command port). The DS only allows for a single data connection at a time, so if it is already engaged in a data connection with another application and/or network host the *DS Manager* will be rejected.
- When telnet TCP access method is selected the *DS Manager* performs programming through a TCP connection established to port 23 of the DS (see telnet TCP programming). Unlike inband access method, this does not require any prior setup of the DS side but will only work with newer Device Servers that run on **firmware V3.50 or higher**.

** Here we touch on a very complicated subject. Modern routers offer a bewildering array of setup options. We will attempt to cover this in details in our upcoming white papers.*

Preparing the DS for Inband Access

As explained in access parameters for the address book mode a certain pre-programming must be made on the DS side before this DS can be accessed using inband access method. Inband access method is usually used for remote Device Servers at locations with which out-of-band communications are not possible. Therefore, it may be not possible to pre-program the DS for inband access when it is already installed in its intended remote location!

There are two ways in which the pre-programming can be done:

- Use the auto-discovery access mode to pre-program the DS from a PC that is located on the same network segment as this DS. Alternatively, you can temporarily connect the DS to a local network segment, pre-program it, then return it to its intended remote location.
- Use the COM access mode. Connect the serial port of the DS to the COM port of the PC and make the pre-programming through the serial port.

Here is what you need to do:

- Select the DS:
  - If the pre-programming is to be done through the network select the

auto-discovery access mode and double-click on the DS in the *device list*- this will open the *settings dialog*. You don't have to change the IP-address of the DS- in this mode the DS Manager will be able to access even Device Servers with unreachable IP-address;

- If the pre-programming is to be done through the serial port select the COM access mode and double-click on the COM port (to which the DS is connected) in the *device list*. You will be asked to press the setup button* and then the *settings dialog* will be opened.

- In the *settings dialog* make the following choices:
  - Set desired IP-address for this DS (**IP-address (IP) setting**). Once the DS is installed in its intended remote location you won't be able to access it unless it already has a suitable and known IP-address;
  - Set desired data port number (**Port Number (PN) setting**). Once the DS is installed in its intended remote location you won't be able to access it unless it already has a suitable and known port number;
  - Select 1 (TCP) for the **Transport Protocol (TP) setting**. Inband access is only possible when the TCP transport protocol is selected.
  - Select 0 (server) or 1 (server/client) for the **Routing Mode (RM) setting**. The DS won't accept data connections when the **Routing Mode** is 2 (client) and the inband programming is performed through the data connection.
  - Select 1 (enabled) for the **Inband Commands (IB) setting**.
- Press *OK* to close the *settings dialog* and reboot the DS.

Note on DHCP servers

Procedure above includes the setup of the IP-address. This should only be done if the remote network segment doesn't have DHCP service. If there is a DHCP service then it is better to run with the **DHCP (DH) setting** at 1 (enabled) and let the DHCP server handle the IP-address of the DS automatically.

The problem here is that the DHCP server may change the IP-address of the DS in the future and you will not know about this! If you want to prevent this from happening then set the IP-address of the DS manually and configure this IP-address as manually managed on the DHCP server's side (most DHCP servers provide an option to exclude certain IPs from automatic management).

** On EM100, EM120, EM200, EM203(A)- pull the MD line LOW for at least 100ms.*

Troubleshooting (Address Book Mode)

This topic provides a list of hints that can help you figure out why the *device list* of the address book mode shows a particular DS (address book entry) with "no response" or "unknown device" status. See troubleshooting (auto-discovery mode) and troubleshooting (COM mode) for hints on other two access modes.

In the address book access mode you do not encounter a situation when you cannot "see" some entries in the *device list*. This is because this list is created manually by you and all address book entries are displayed at all times, even when corresponding Device Servers cannot be reached. Instead, you often have to solve the problem of why the *DS Manager* cannot contact the DS specified in the particular address book entry.

The problem can manifest itself in two different ways:

--- The *DS Manager* indicates that there is no response from the IP-address specified in the address book entry (no status icon is displayed at all).

The *DS Manager* indicates that there *is* some kind of device at the specified IP-address but it doesn't seem to be a Tibbo Device Server (the question mark is displayed).

"No response" status is displayed when the *DS Manager* cannot PING the IP-address specified in the address book entry.

Here is why you may get "no response" status:

- The DS may be switched off or not connected to the network*.
- Incorrect IP-address may be specified for this address book entry. Think carefully what IP-address you input. Depending on the router setup you may need to input the IP-address of the DS itself or the access (forwarding) IP-address on the router**.
- Firewall software on your PC or on the router may be set to disallow PINGs (a very common situation on many networks). For the *DS Manager* to work you must have PING (ICMP) traffic enabled. Attention here! Our experience shows that many Users are not aware of the firewall software installed on their own PCs! Some firewalls come as part of larger "protection" suites (anti-virus, anti-intrusion, etc. programs). Some operating systems, such as *Windows XP*, include the firewall software too.
- If your DS is running an older firmware (V2.xx) then it will not be accessible through the network and won't respond to PINGs in the following cases:

When the DS is in the serial programming mode (status LEDs of the DS "play" the pattern as shown on the left)**;

When the DS is in the error mode (status LEDs of the DS "play" the pattern as shown on the left)**.

This *does not* apply to the Device Servers running DS firmware V3.xx or higher. In this firware the DS is visible on the network at all times.

"Unknown device" status is displayed when there is no reply to the echo command sent by the *DS Manager*. This command is unique to Tibbo Device Servers and is used to identify them on the network (and also collect status information). Depending on the access method specified for this address book entry the echo command may be sent as out-of-band UDP commandor inband TCP command.

Here is why you may get "unknown device" status:

- **If out-of-band (UDP) access method** is selected for this address book entry:
  - In certain cases this may be because the DS is switched off or not connected to the network. It is true that if the *DS Manager* displays the icon this means that it has already got the reply to the PING request but in some router modes the PING reply actually comes from the router, not the DS itself**. Therefore, do make sure that the DS is online;
  - Incorrect programming port number may be specified for this address book entry. Think carefully what port number you input. Depending on the router setup you may need to input the command port number on the DS itself (i.e. 65535 or 32767, you can use either one) or the access (forwarding) port on the router**;

- Firewall software on your PC or on the router may be set to disallow UDP traffic (a very common situation on many networks). For the *DS Manager* to work you must enable UDP traffic to the command port specified in this address book entry. In some router setups** UDP traffic through the router is very unreliable or completely impossible. In this case you may need to use inband access method which relies on TCP protocol.

- **If inband TCP access method** is selected for this address book entry:
  - First of all remember, that for "inband" address book entries the *DS Manager* does not automatically verify DS presence when the *Refresh button* is pressed. Therefore, after the *Refresh button* is clicked all Device Servers for which the inband access is defined initially appear in the list with the ? icon. If you want to check if such a DS is accessible then attempt to execute the desired action: press Settings button or Buzz button and see the result (upgrade, initialize, and routing status functions are not available for "inband" address book entries). At this point the *DS Manager* will attempt to establish a TCP connection to the DS and find out if the DS is accessible. This will be reflected in the *device list*.
  - Incorrect programming port number may be specified for this address book entry. Think carefully what port number you input. Depending on the router setup you may need to input the data port number on the DS itself (this is the port number specified by the **Data Port Number (PN) setting** of the DS) or the access (forwarding) port on the router**;
  - The DS may not be configured for inband programming- read preparing the DS for inband access for step-by-step instructions.
  - The DS may be already engaged in a data TCP connection with another application and/or network host. Since inband commands are, by definition, the commands that are passed within the data connection itself the *DS Manager* actually establishes a data TCP connection with the DS in order to program it (this is why this connection is made to the data port of the DS, not the command port). The DS only allows for a single data connection at a time, so if it is already engaged in a data connection with another application and/or network host the *DS Manager* will be rejected***.

- **If telnet TCP access method** is selected for this address book entry:
  - Telnet TCP access method is only supported by newer Device Servers running firmware V3.50 or higher. Make sure you are not attempting to use this access method for the Device Server that doesn't support it;
  - Incorrect programming port number may be specified for this address book entry. Think carefully what port number you input. Depending on the router setup you may need a standard telnet port of 23 or some other access (forwarding) port on the router**.

** Really sorry for this "hint"... it's just that this is the root cause of the problem in many cases!*

*** Here we touch on a very complicated subject. Modern routers offer a bewildering array of setup options. We will attempt to cover this in details in our upcoming white papers.*

**** We understand that this is a limitation and are working to remove it.*

Serial Access Mode

Can't access your Device Server through the serial port?
See troubleshooting!

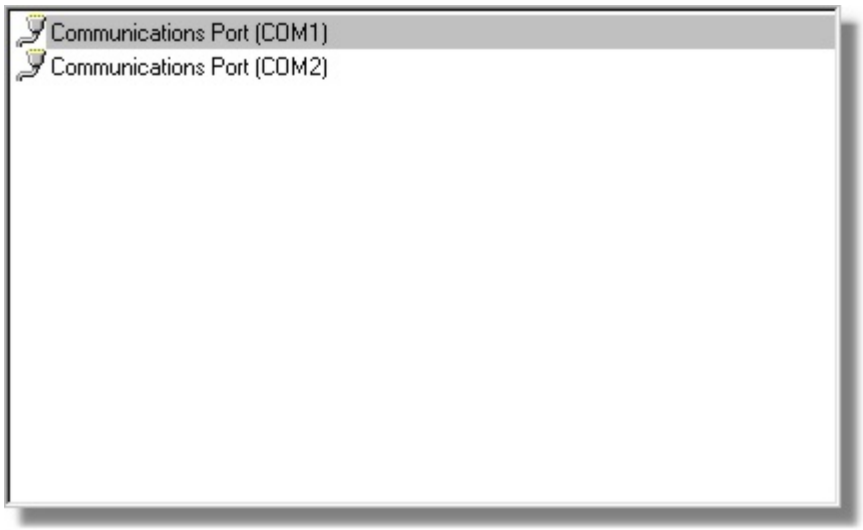
COM mode is selected by choosing "Device Servers attached to the COM ports" from the access mode drop-down box, located at the top of the *DS Manager's* main window. In the COM access mode the DS is accessed through its serial port. Therefore, you must connect the serial port of the DS to the unused COM of your PC using a DS-to-PC cable (like WAS-1455 supplied by Tibbo).

In the COM access mode the *device list* displays all physical COM ports of your PC. The *DS Manager* is not able to determine which COM the DS is connected to so you have to make a correct selection by yourself.

For the *DS Manager* to access the DS the latter must be in the serial programming mode. To put the DS into this mode press the setup button*.

Note:

Status LEDs of the DS are playing a *serial programming mode pattern* (shown on the left) when the serial port of the DS is in the serial programming mode (click here to see all available patterns).



** On EM100, EM120, EM200, EM203(A)- pull the MD line LOW for at least 100ms.*

Troubleshooting (Serial Access Mode)

This topic provides a list of hints that can help you figure out why you cannot access the DS through the serial port. See troubleshooting (auto-discovery mode) and troubleshooting (address book mode) for hints on other two access modes.

Here are the reasons why the *DS Manager* may not able to access the DS in the COM access mode:

- The DS may be switched off*.
- Incorrect cable is used to connect the serial port of the DS to the COM port of the PC. Use WAS-1455 supplied by Tibbo!

- Wrong COM port is selected in the *device list*.
- The serial port of the DS is not in the serial programming mode. To put the DS into this mode press the setup button**.

Note:

Status LEDs of the DS are playing a *serial programming mode pattern* (shown on the left) when the serial port of the DS is in the serial programming mode (click here to see all available patterns).

** Really sorry for this "hint"... it's just that this is the root cause of the problem in many cases!*

*** On EM100, EM120, EM200, EM203(A)- pull the MD line low for at least 100ms.*

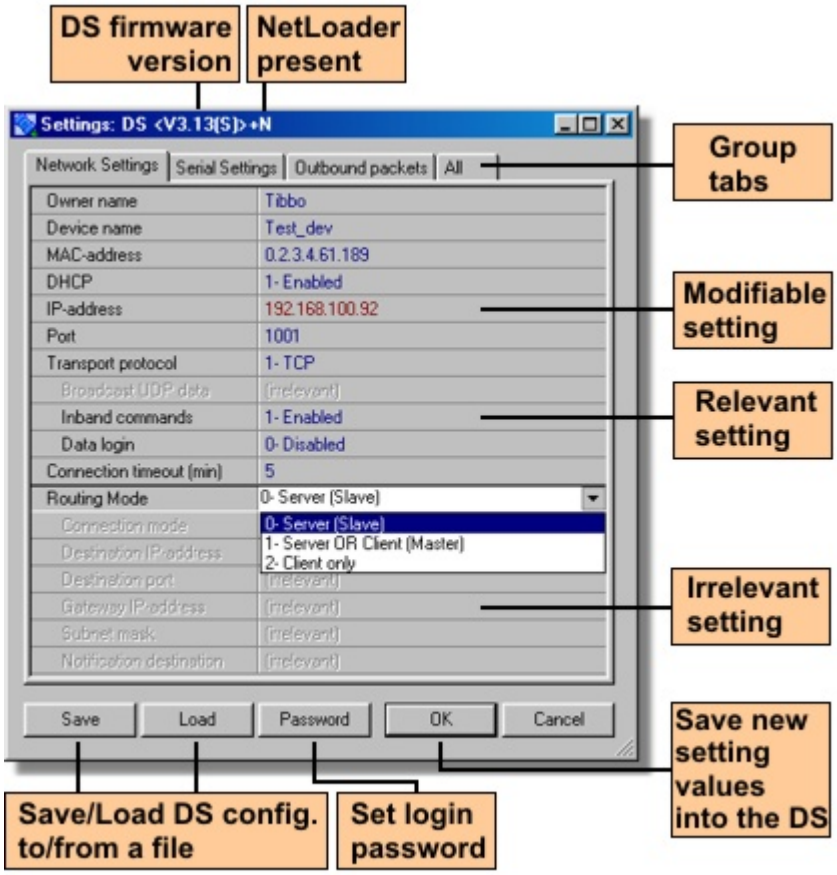
Functions 1.3

This section lists all "functions" (features) of the *DS Manager*. These functions are associated with the *function buttons* located in the *DS Manager's* main window.

Editing DS Settings (Settings Button)

Internal settings of DS are viewed and edited in the *settings dialog*. This function is available in all access modes.

To open the *settings dialog* either double-click on the DS in the *device list* (COM port to which the DS is connected if you are in the COM access mode). Alternatively, you can single-click on the DS (COM port) in the *device list* and then press the *Settings button*.



When the *settings dialog* is opened it reads out current values of all settings available on the DS and displays them in a *setting table*. All settings are divided into groups and each group is placed on its own *group tab*. *Dialog caption* displays the version of firmware running on this DS. If "+N" is displayed next to the version number then this means that the NetLoader is present and the firmware of this DS can be upgraded through the network.

Settings dialog can correctly display the settings for Device Servers with different application firmware versions- correct setting table will be displayed for each supported firmware version. The list of settings available in different firmware versions comes from the **.sdf* files ("setting definition files") located in the */sdf* subfolder of the DST's program folder.

The only setting that is not shown in the table is the **Password (PW) setting**. This setting defines the login password that will have to be entered when you want to access the DS through the network (i.e. open the *settings dialog*, upgradeor initialize the DS, or change the IP-address). Click *Password button* to set the password (edit the **Password (PW) setting**).

Relevance (or irrelevance) of certain settings depends on the current values of other settings. Setting reference section describes which settings "depend" on which other settings. Additionally, some settings can be relevant but their value can be modified automatically by the DS.

The settings displayed in the *table* may be in one of the following three states:

- **Relevant**- setting name and its value appear in solid black. This means that the setting can be edited.
- **Irrelevant**- setting name is grayed and the line "*[irrelevant]*" is displayed instead of the setting value. This means that the setting is currently irrelevant

and cannot be edited. To edit such a setting make it relevant first by adjusting the values of other settings. For example, the **Gateway IP-address (GI) setting** is shown as irrelevant on the screenshot above because the **Routing Mode (RM) setting** is 0 (server). Select the **Routing Mode** of 1 (server/client) or 2 (client) and the **Gateway IP-address** will become relevant.

- **Modifiable**- setting name is shown in solid black and the setting value is shown in dark red. This means that the setting can be edited but the new value can be automatically overwritten by the DS. For example, the **IP-address (IP) setting** is relevant at all times but when the **DHCP (DH) setting** is 1 (enabled) the DS will overwrite original IP-address with the (new) address supplied by the DHCP server.

The *settings dialog* also lets you save current values of DS settings into a file (click *Save button*) and load setting values from file (click *Load button*). Configuration files have the *.ds* extension.

Programmer's info:

The *DS Manager* reads and writes the values of settings using the **Get Setting (G)** and **Set Setting (S)** commands.

In case of network access (auto-discovery and address book modes) the *DS Manager* has to login first (open the network programming session) and this is done using the **Login (L) command**. Serial programming (COM access mode) doesn't require this step.

Network programming sessions and serial programming are ended either with **Reboot (E) command** (this makes the DS "recognize" new setting values) or with **Logout (O) command** (no reboot).

Upgrading DS Firmware (Upgrade Button)

Internal firmware of the DS can be upgraded in the field. This function is available in all access modes with the following limitations:

- Some earlier Tibbo devices (EM100-00/ -01/ -02, DS100-00/ -01/ -02) did not support network upgrades so their firmware can only be upgraded through the serial port (COM access mode).
- All other devices do support network upgrades but such an upgrade is only possible if a functioning (albeit older) firmware file is already found on the device. This implies that loading a wrong file into the device renders all future network upgrades impossible until a correct upgrade is performed through the serial port.
- Additionally, for the network upgrade to work the *DS Manager* must be able to access the DS using an out-of-band UDP access method. Therefore, Device Servers defined by the "inband" address book entries (in the address book mode) cannot be upgraded through the network (access methods are described in access parameters for the address book mode).

To upgrade the DS firmware:

- Select (single-click) the DS you want to upgrade in the *device list* (or COM port to which the DS is connected in the COM access mode) and click *Upgrade button*. *Upgrading the Device Server dialog* will appear.
- Browse to a file you want to upload into the DS and click OK. Be sure to select correct firmware file- this depends on the DS model and also the way upgrade is done (through the network or through the serial port).
- What happens next depends on the access mode:
  - For network upgrades (auto-discovery and address book modes) the upgrade will start automatically. After the upgrade is finished the *DS Manager* will reboot the DS. The *DS Manager* will also make sure that the DS starts running with the newly loaded firmware and let you know if the DS enters the error mode, which means that its settings must be initialized;
  - For serial upgrades (COM access mode) the procedure is more "manual". You will be asked to switch the DS off, press the setup button*, and switch it back on while keeping the button pressed. Upgrade will start after that. At the end of the upgrade you will be asked to reboot the DS manually (switch it off and on). You will have to also manually check if the DS requires initialization.

Programmer's info:

The *DS Manager* verifies the NetLoader presence and passes control to it using the **Jump To Netloader (N) command**.

** On EM100, EM120, EM200, EM203(A)- pull the MD line LOW for at least 100ms.*

Initializing the DS (Initialize Button)

Internal settings of the DS can be initialized. Initialization restores all settings to their default values*. This function can be used to return the DS to the "known setup" and/or "repair" the DS when one or more settings are found to be invalid (i.e. the DS is in the error mode). This function is available in all access modes with the following limitation:

- When you are working in the address book mode the *DS Manager* won't let you initialize the DS for which inband access method is set. This is because initialization would have disabled inband access** and this would have made further interactions with this DS impossible. If you need to initialize such a DS then connect it temporarily to the same network segment as your PC (and use auto-discovery mode) or to the COM port of your PC (and use COM access mode), and then perform initialization.

To initialize the settings of the DS select (single-click) the DS you want to upgrade in the *device list* (or COM port to which the DS is connected in the COM access mode) and click *Initialize button*.

Initialization results differ depending on the access mode. Some settings are always initialized when the initialization command is issued through the serial port (COM access mode) but are only initialized if were invalid when the initialization command is issued through the network (auto-discovery and address book access modes). Some settings are never initialized (no matter what the access mode is) unless found to be invalid.

Example: the **IP-address (IP) setting** is only affected by the initialization if the DS is accessed through the serial port. When you initialize the DS through the network this setting won't be initialized unless it was invalid. This is done to make sure that the DS can still be accessed through the network after the initialization.

Setting reference provides complete information on each setting of the DS including default factory values and conditions under which a particular setting will be initialized.

Programmer's info:

This function relies on the **Initialize (I) command**.

** These are either default factory values or default values defined by the user (if custom profile is added to the firmware).*

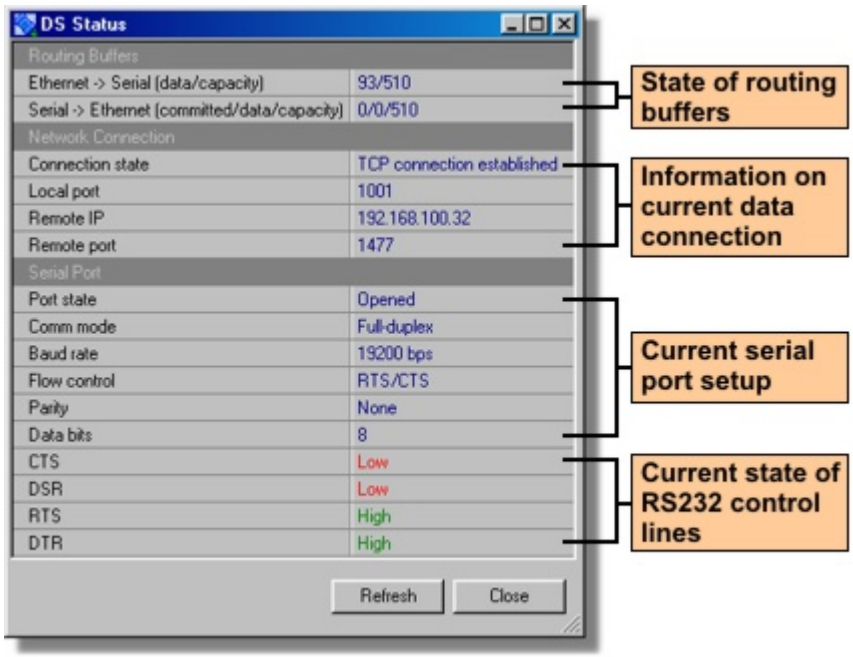
*** Default factory value for the **Inband (IB) setting** is 0 (disabled) and **Transport Protocol (TP) setting** is 0 (UDP). Inband access must be enabled and the transport protocol must be TCP for the inband access to work.*

Monitoring DS Status (Routing Status Button)

DS operation can be monitored remotely, using a *routing status dialog*. To open the *dialog* click the *Routing status* button. This feature is a very useful debugging tool that lets you determine the DS state at any moment. The information in the *dialog* is updated each time *Refresh button* is pressed (this is the button in the *dialog*, not in the *main window*).

This function is available in the auto-discovery and address book modes with the following limitations:

- When you are working in the address book mode you won't be able to use this feature on Device Servers for which inband access method is set. Routing status feature requires out-of-band access! This is because inband access implies sending commands through the TCP data connection and the DS only allows for a single data connection at a time. Since the feature is created mainly to monitor the DS while it is routing the data (i.e. has a data connection with some other network host/application), the only available data connection should not be occupied by the *DS Manager*.
- *Routing status feature* is not supported by the Device Servers running older firmware (earlier than V3.xx).



***Routing status dialog* displays the following information:**

- **Current state of routing buffers.** Routing buffers are used as a temporary storage for the data being routed between the Ethernet and serial ports of the DS. The *routing status dialog* reports the number of committed bytes (for serial-->Ethernet buffer), total number of bytes in each buffer ("data"), and the capacity of the buffers. Read serial-to-Ethernet data routing topic to learn what "committed" means. Capacity information is provided because different DS models have buffers of different size.
- **Information on current data connection.** This partially doubles the data reflected by the status icons (displayed in the *device list*) and the status areaof the *main window*. Additional information provided by the *routing mode dialog* includes the IP-address and the port number of the network host with which the DS has been/is/will be in a data connection. Here is what these two fields show:
  - After power-up the fields show the IP-address and port defined by the **Destination IP-address (DI) setting** and **Destination Port Number (DP) setting**;
  - If these default values are overridden by serial-side parameters and instructions (a.k.a. "modem commands") then the fields show new overriding values;
  - While the data connection is established and after it is closed (aborted) the fields show the IP-address and port of the network host with which this connection is (was) established. Notice that this may be different from the above- if the DS has accepted an incoming connection!
- **Current serial port setup.** This data may be of interest because it will not necessarily match the serial port setup defined by the serial port-related DS settings. The values of settings can be overridden by the network-side parameters (a.k.a. "on-the-fly commands")*. Therefore, the *routing status dialog* can be used to verify actual current serial port setup.
- **Current state of RS232 control lines.** The state of control lines is displayed in the correct polarity for DS100R, DS100B, DS203. For EM100, EM120, EM200, EM203(A) the state is exactly opposite. So, if the state of RTS line (output) is shown as HIGH then this means that the DS is ready to receive the data from

attached serial device.

Programmer's info:

The *DS Manager* uses two commands- **Echo (X)**and **Status (U)**- to read out the current status of the DS.

** On-the-fly commands are used, for instance, by Virtual Serial Ports (VSPs) to change serial port configuration of the DS. This way PC application that has opened the VSP can change the setup of the DS serial port as needed.*

"Buzzing" the DS (Buzz Button)

The *buzz feature* allows you to match an entry in the *device list* to an actual physical DS. The feature is available in the auto-discovery and address book modes with the following limitation:

- The *buzz feature* is not supported by the Device Servers running older firmware (earlier than V3.xx).

Clicking on the *Buzz button* causes the DS (selected in the *device list*) to play a fast-blinking pattern on its status LEDs (shown at the left). This way you can easily identify which device a particular *device list* entry corresponds to.

Programmer's info:

This function relies on the **Buzz (B) command**.

Changing IP-address (Change IP Button)

The *DS Manager* provides a way to assign a new IP-address to the DS over the network. This is done by sending a broadcast command that refers to the target DS by its MAC-address*. All Device Servers on a local network segment will receive the broadcast but only the one whose MAC-address matches the one specified in the broadcast will react.

The feature will work even if the DS had an unreachable or invalid IP-address. The disadvantage is that, since the broadcasts cannot pass through the routers, the feature can only be used with local Device Servers and, therefore, is limited to the auto-discovery access mode.

To assign a new IP-address to the DS click the *Change IP button*, input the new IP-address and click OK.

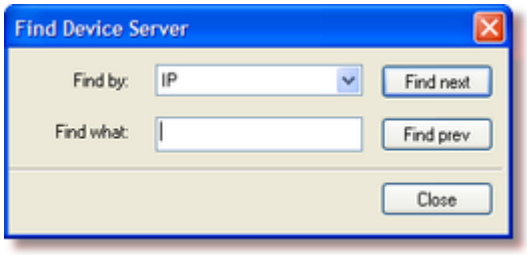
Programmer's info:

This function relies on the **Assign IP-address (A) command**.

**MAC-addresses are unique for all Device Servers (and, in fact, all Ethernet devices).*

Finding a DS on the list (Find Button)

The *Find* button is used to locate a specific Device Server in a long list. Clicking it pops up the *Find Device Server* dialog:



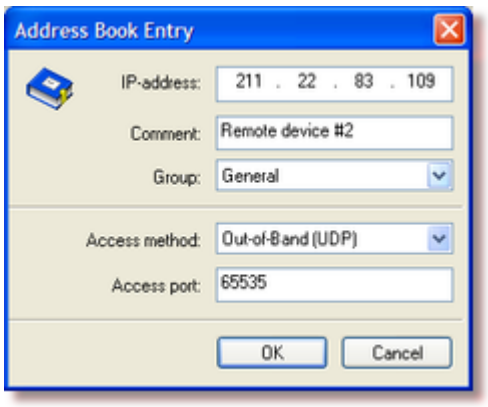
The *dialog* has the following fields:

- **Find by:** Allows you to select by which field to search. In auto-discovery mode, you can search by IP, MAC, Owner Name or Device Name. In address book mode, you can search by IP, Owner Name, Device Name or Comment.
- **Find what:** Once you select the field by which you wish to search, start typing your search string in the **Find what** textbox. Searching is 'live' - the first record matching your criteria is immediately highlighted. If you wish to see the next record matching your criteria, press ENTER or **Find next**.

Editing the Address Book (Add, Remove, Edit Buttons)

Add, *Remove*, and *Edit buttons* are used to edit the address book. Address book is a fixed list of Device Servers displayed in the address book access mode. *Remove* and *Edit buttons* are only visible in the address book access mode, while the *Add button* is also available in the auto-discovery mode. This allows you to add the DS "found" in the auto-discovery mode to a permanent list in the address book mode.

Clicking the *Add, Edit* or *Remove* buttons brings up the A*ddress Book Entry dialog*.



The *dialog* has the following fields:

- **IP-address.** This is the IP-address of the DS itself or the forwarding IP-address on the router through which this DS can be accessed (this depends on the router setup*).
- **Comment.** This comment is stored on the PC (not in the DS) and can contain any useful information (i.e. "Remote device #1").
- **Group.** This listbox allows you to select a group for this DS. Dividing devices into

groups can make life easier when managing an Address Book with hundreds of devices. For more information, see Managing Address Book Groups (Groups button).

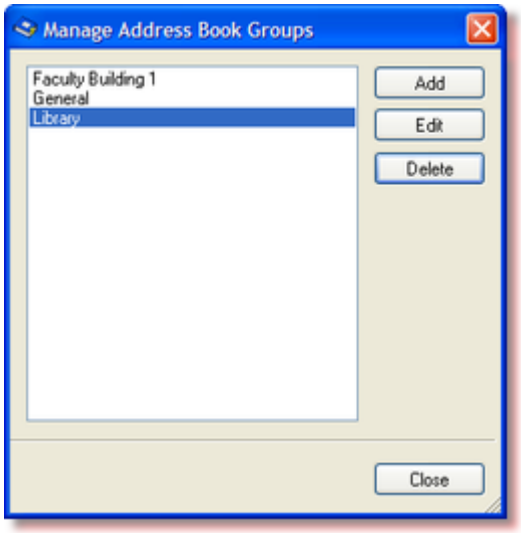
- **Access method.** The *DS Manager* can access a particular DS using out-of-band (UDP), inband (TCP), or telnet (TCP)access method. See access parameters for the address book mode topic for more information.
- **Access port.** This is the access port on the DS to which the *DS Manager* will send programming commands. Depending on the router setup this may be a port on the DS itself or the forwarding port on the router*. The port number on the DS is different depending on the selected access method. See access parameters for the address book mode topic for more information.

** Here we touch on a very complicated subject. Modern routers offer a bewildering array of setup options. For further details, see* AN009. WAN Basics.

Managing Address Book Groups (Groups button)

The Address Book can be subdivided into arbitrary device groups. Such subdivision can ease administration when working with hundreds or thousands of Device Servers in a WAN scenario. You can create a scope of just 10 or 15 devices, instead of endlessly scrolling through a huge list to find the DS you need.

This is the *Manage Address Book Groups* Dialog:



It allows you to create, edit and delete groups (the buttons are self-explanatory). A group has just one attribute -- a name.

Once you create a group, you can assign Device Servers to it using the Address Book Entry dialog. You can rename a group after you have filled it with devices. The devices will still be mapped to that group.

Warnings And Messages

This reference section contains the following information:

- **Additional information on the status messages** displayed in the *status area* of the *DS Manager's main window*.
- **Additional information on the warning and error messages** displayed by the *DS Manager*.

DS Status Messages

This section provides additional information on the status messages displayed in the *status area* of the *DS Manager's main window*.

The list of messages is arranged in the alphabetical order.

Description

Message text: Device Server is running in the error mode and must be reinitialized

Corresponding status icon: 

May appear in: Auto-discovery and address book access modes

Details

This status means that some settings on this DS are invalid and require initialization. The DS should not be allowed to operate in this condition and must be reinitialized as soon as possible.

Programmer's info:

The *DS Manager* collects DS status information (and detects DS presence) using the **Echo (X) command**. Whether or not the DS is in the error mode is determined from the state of the ***e*** flag in the command response.

Description

Message text: Device Server is running in the firmware upgrade mode and is ready to accept new firmware file

Corresponding status icon: 

May appear in: Auto-discovery and address book access modes

Details

This status means that the DS is running the NetLoader. NetLoader is a separate firmware component that facilitates application firmware upgrades over the network. Normally, this icon is displayed during the network firmware upgrade. After the network upgrade is finished the *DS Manager* reboots the DS and after that the DS is supposed to start running the newly loaded firmware.

If the DS enters the firmware upgrade mode unexpectedly and if this mode "persists" then this means that DS firmware is corrupted, or that incorrect firmware file was uploaded, or that the upload was incomplete.

Programmer's info:

The *DS Manager* collects DS status information (and detects DS presence) using the **Echo (X) command**. This command is supported both by the application firmware and by the NetLoader. Whether the DS is running its application firmware or is in the firmware upgrade mode is determined from the ***m*** flag in the command response (it is 'N' for the application firmware and 'L' for the NetLoader).

Description

Message text:	Device Server has not yet obtained its IP-address from the DHCP server
Corresponding status icon:	
May appear in:	Auto-discovery and address book access modes

Details

This status means that the DS is running with DHCP enabled (**DHCP (DH) setting** is 1 (enabled)) and the DS hasn't yet received its IP-address from the DHCP server. The DS attempts to configure its IP-address after the powerup and won't perform its routing function until IP configuration is completed. Typically, IP configuration takes only 1-2 seconds to complete. If the DS is "stuck" in this state for much longer then this means that DHCP service may not be available on your network.

Programmer's info:

The *DS Manager* collects DS status information (and detects DS presence) using the **Echo (X) command**. IP configuration status is determined from the state of the ***i*** flag in the command response.

Description

Message text:	No response received from this IP-address (and port)
Corresponding status icon:	--- (no icon is displayed for this status)
May appear in:	Address book access mode

Details

See troubleshooting (address book mode) for a list of hints on why you might be getting no reply for this address book entry.

Programmer's info:

Description

Message text: Device Server did not return the status information. This means that this DS is running an older firmware version

Corresponding status icon: 

May appear in: Auto-discovery and address book access modes

Details

In the V3.xx firmware the DS returns its status information along with the response to the echo request, sent by the *DS Manager* during the refresh. In older firmware versions (V2.xx) the response from the DS does not contain any status information. You are recommended to upgrade your firmware to the latest version.

Programmer's info:

The *DS Manager* collects status information (and detects DS presence) using the **Echo (X) command**. In the old firmware versions this command only returned the first two fields (***nnn.nnn.nnn.nnn.nnn.nnn*** and ***ppppp***).

Description

Message text: Out-of-band (UDP) network programming session is in progress;
Inband (TCP) network programming session is in progress;
The DS is in the serial programming mode

Corresponding status icon: 

May appear in: Auto-discovery and address book access modes

Details

Three separate status messages are displayed for three possible forms of DS programming: serial, out-of-band (UDP), and inband (TCP). Serial programming is considered to be in progress whenever the serial port of the DS is in the serial programming mode. Out-of-band (UDP) or inband (TCP)programming session is considered to be in progress after the network host has logged in using a corresponding access method (out-of-band or inband)- see authentication*. Inband (TCP) access method is only used in the address book mode (see access parameters for the address book mode).

All three forms of programming are mutually exclusive- see programming priorities.

Programmer's info:

The *DS Manager* collects DS status information (and detects DS presence) using the **Echo (X) command**. Whether or not any form of programming is in progress is determined from the state of the ***s*** flag in the command response. Network logins are performed using the **Login (L) command**.

** Just sending commands that do not require prior login does not constitute a programming session.*

Description

Message text: Ethernet-to-serial buffer overflow;
Serial-to-Ethernet buffer overflow;
Serial-to-Ethernet and Ethernet-to-serial buffer overflow

Corresponding status icon:  (overflow condition is displayed only in conjunction with "connection established" icon)

May appear in: Auto-discovery and address book access modes

Details

These messages are displayed when one or both routing buffer overflow is detected. Routing buffers are used for temporary data storage when routing data between the Ethernet and serial ports of the DS.

In general, do the following to avoid overflows:

- For Ethernet-to-serial buffer: using TCP/IP transport protocol (see **Transport Protocol (TP) setting**) guarantees that this buffer never overflows.
- For serial-to-Ethernet buffer: using RTS/CTS flow control (see **Flow Control (FC) setting**) guarantees that this buffer never overflows*.

Programmer's info:

The DS Manager collects DS status information (and detects DS presence) using the **Echo (X) command**. Buffer condition is determined from the state of the ***E*** and ***S*** flags in the command response.

** Naturally, attached serial device must also support RTS/CTS flow control for this to work.*

Description

Message text: Although there is SOME device at this IP-address it appears that this is not a Device Server

Corresponding status icon: 

May appear in: Address book access mode

Details

See troubleshooting (address book mode) for a list of hints on why you might be getting this status for an address book entry.

Programmer's info:

Description

Message text: Current IP-address is unreachable with your network

Corresponding status icon:  (The whole icon is grayed)

May appear in: Auto-discovery access mode

Details

This status means that the *DS Manager* has detected the DS on the local network segment but is unable to communicate with this DS by using a "normal" IP addressing. See troubleshooting (auto-discovery mode) for a list of hints on why this could happen.

Programmer's info:

Warning and Error Messages

This section provides additional information on the warning and error messages displayed by the *DS Manager*.

The list of messages is arranged in the alphabetical order.

Description

Message text: The IP-address of this DS is unreachable and the DS can only be accessed in the broadcast mode. The firmware on this Device Server is outdated and does not support broadcast access. Assign a compatible IP-address or upgradeto firmware V3.xx or higher (recommended)

May appear in: Auto-discovery access mode; Settings, Initialize, Buzzfunctions

---

Details

This message means that the *DS Manager* cannot communicate with the DS using a normal IP-addressing. Possible reasons for why this could happen are outlined in troubleshooting (auto-discovery mode).

The *DS Manager* can access local Device Servers even when normal IP communications is impossible. This is done through a so called broadcast access. This functionality, however, is only supported by DS firmware V3.xx or higher.

The way out of this situation is to make normal IP communications possible or/and to upgrade the DS firmware to V3.xx or higher. Depending on how old the DS firmware is you may not be able to upgrade the firmware over the network and will need to do this through the serial port of the DS (in the COM access mode).

Programmer's info:

Broadcast access is based on the **Select In Broadcast Mode (W) command** that was only introduced in firmware V3.xx.

Description

Message text: Inband (TCP) access method is defined for this Device Server and the *DS Manager* was unable

to establish a TCP connection to the DS

May appear in: Address book access mode; Settings, Initialize, Buzzfunctions

---

Details

When inband (TCP) access method is specified for a particular address book entry and the *DS Manager* needs to access the DS it must establish a TCP connection to this DS first. This message indicates that TCP connection could not be established. Read troubleshooting (address book mode) for hints on what might be causing the problem.

Programmer's info:

Description

Message text: This Device Server has a data connection in progress. Performing requested operation will abort this connection. Continue still?

May appear in: Auto-discovery and address bookaccess modes; Settings, Upgrade, Initializefunctions

---

Details

Programmer's info:

The *DS Manager* verifies whether the DS is engaged in a data connection by sending the **Echo (X) command** and analysing the status of the ***c*** flag in the reply. If this flag is not '*' then this means that the data connection is not closed (is in progress).

Description

Message text: IP-address was changed successfully but the *DS Manager* cannot locate this Device Server on the network

May appear in: Auto-discovery access mode; Change IP function

---

Details

After having changed the IP-address the *DS Manager* makes sure that the DS is online and using the IP-address that was just assigned. This message appears when the DS has confirmed that command was accepted but later could not be found among local Device Servers. This situation is not normal and may indicate that the new IP-address is (for some reason) blocked by your PC or network equipment.

Note, that this is more serious than the case when the IP-address turns out to be unreachable (this situation is reported by the unreachable IP-address status message). If this was the case the *DS Manager* would still be able to detect the DS on the network. Watch out for some special firewall settings of your PC or similar

reasons why communications with a certain IP-address might be blocked (also see troubleshooting (auto-discovery mode)).

Programmer's info:

IP-address is changed using **Assign IP-address (A) command**. This message means that **OK (A) status code** was actually received for this command but the DS could not be found during subsequent refresh operation.

Description

Message text:	*DS Manager* cannot locate the Device Server after making it enter the firmware upgrade mode. This may be because the IP-address (or MAC-address) of this Device Server has changed
May appear in:	Auto-discovery and address bookaccess modes; Upgrade function

---

Details

Firmware upgrades over the network are facilitated by a separate firmware component called the NetLoader. To upgrade the application firmware over the network the *DS Manager* makes the DS switch to the NetLoader first.

When control is passed to the NetLoader, it attempts to read the values of **IP-address (IP)** and **MAC-address (FE)** settings and use these values. This way the switchover to the NetLoader is "seamless" and the DS is still accessible at the same IP and MAC as when it was running the application firmware.

In selected cases, the readout of those two settings can fail (for instance, when one or both of these settings is/are invalid). In this case the DS will replace the invalid value(s) with a default one: 127.0.0.1 for the IP-address, 0.1.2.3.4.5 for the MAC-address.

If you are accessing the DS using the address book mode and the DS assumes these default values you may not be able to access it (at least temporarily). We recommend you to connect the DS to the same network segment with your PC and use the auto-discovery access mode.

Programmer's info:

Description

Message text:	Upgrade completed successfully but the *DS Manager* was unable to locate this Device Server on the network after rebooting it
May appear in:	Auto-discovery and address bookaccess modes; Upgrade function

---

Details

The cause of this message is similar to that of the DS lost (after entering NetLoader) message (but "in reverse"). After the NetLoader has finished working and the DS was rebooted its IP-address and/or MAC-address have probably

changed. To the *DS Manager* this will look like "disappearance" of the DS from the network.

To "find" the DS connect it to the same network segment as the PC and run the *DS Manager* in the auto-discovery access mode. To find out which device in the list is the one you are looking for either disconnect all other Device Servers from the network (this will leave only one DS in the list) or use the Buzz function to locate the DS.

Programmer's info:

Description

Message text: Initialization was completed successfully but the *DS Manager* cannot locate this Device Server on the network

May appear in: Auto-discovery access mode; Initialize function

---

Details

After having performed the initialization the *DS Manager* makes sure that the DS is online. Network initialization does not change the IP-address unless the **IP-address (IP) setting** were found to be invalid. The default factory value for this setting is 127.0.0.1*, so see if there is a DS now that has this IP-address. If so then this may be your "lost" Device Server!

Programmer's info:

** Unless another default value has been defined through a custom profile.*

Description

Message text: Such address book entry already exists

May appear in: Auto-discovery and address bookaccess modes; Add or Edit function

---

Details

Each DS in the address book is identified by the combination of the IP-address, access port number, and the access method (see access parameters for the address book mode). This message means that your input matches another address book entry that already exists.

Programmer's info:

Description

Message text: Device Server is currently running in the error mode, which means that some of its settings are invalid. You are recommended to initialize the Device Server first. Continue still?

May appear in: Auto-discovery and address bookaccess modes; Settings function

---

Details

This message appears when you attempt to open the settings dialog while the DS is in the error mode. Because some settings are invalid the *DS Manager* is unable to read out their values and display them in the *settings dialog*.

Once the error mode is detected it is better to initialize the DS as soon as possible. Continuing DS operation is this state may lead to incorrect (unexpected) DS behavior and also exposes the DS to unauthorized access- password protection is disabled when the DS is in the error mode!

Programmer's info:

The *DS Manager* verifies whether the DS is running in the error mode by sending the **Echo (X) command** and analysing the status of the ***e*** flag in the reply.

Description

Message text: Failed to put the Device Server into the firmware upgrade mode. NetLoader may not be installed

May appear in: Auto-discovery and address bookaccess modes; Upgrade function

---

Details

Firmware upgrades through the network are facilitated by a separate firmware component called the NetLoader. When you click Upgrade button the *DS Manager* instructs the application firmware of the DS to pass the control to the NetLoader. Before doing this application firmware checks if the NetLoader is actually loaded. This message is displayed when the NetLoader appears to be absent.

Here is why the NetLoader may be missing:

- If your DS model is EM100-00/ -01/ -02, DS100-00/ -01/ -02 then this DS cannot be upgraded through the network in principle. The only way to upgrade such a DS is through the serial port (in the COM access mode).
- For EM100-03, DS100R-03, DS100B-00 network upgrades are possible, but only if the NetLoader is installed. When the NetLoader is not present the only way to upgrade the DS is through the serial port (in the COM access mode). If you choose the upgrade file that has "SN" in its name then you will be able to upgrade the firmware and install the NetLoader at the same time! From this moment on you will be able to perform network upgrades as well. Firmware download page at <%WEB%> provides complete info on what file to choose.

Programmer's info:

The *DS Manager* instructs the application firmware of the DS to switch to the NetLoader by sending **Jump To NetLoader (N) command**. This message is displayed when **Failed (F) reply code** is returned.

Description

Message text: This Device Server is currently running in the firmware upgrade mode. Requested function is

not available at this time

May appear in: Auto-discovery and address bookaccess modes; Settings, Initialize, Routing Status, Buzz, Change IPfunctions

---

Details

Network upgrade mode is a separate mode of operation facilitated by an independent firmware component called the NetLoader. The NetLoader allows you to upgrade the main application firmware of the DS over the network. Naturally, when the NetLoader is running the DS is not executing the application firmware and all functions related to the "normal" operation of the DS are not available. The only available function in this mode is Upgrade.

Programmer's info:

Description

Message text: The firmware on this Device Server is outdated and does not support this function. You are recommended to upgrade to firmware V3.xx or higher

May appear in: Auto-discovery and address bookaccess modes; Upgrade, Routing Status, Buzz functions

---

Details

This message means that the firmware on a particular DS you are accessing is outdated (earlier than V3.xx). The function you have requested requires firmware V3.xx or higher.

We recommend you to upgrade to V3.xx (or higher) firmware. This will allow you to fully use all the features of the *DS Manager*. Depending on how old the firmware is you may not even be able to upgrade the firmware over the network (and this is one of the reasons why you may be reading this message). In this case upgrade the DS firmware via the serial port (in the COM access mode).

Programmer's info:

Description

Message text: You have entered an incorrect login password. Please, try again

May appear in: Auto-discovery and address bookaccess modes; Settings, Upgrade, Initialize, Change IPfunctions

---

Details

Login password is the one you have set by clicking on the *Password button* in the *settings dialog*. You cannot access the DS through the network without this password.

If you have forgotten the password then you can either:

- Access the DS in the COM access mode (this does not require password), or...
- Quick-initialize the DS- this will erase the password (only supported by firmware V3.xx or higher).

Programmer's info:

Login password is the one defined by the **Password (PW) setting** of the DS. Also see authentication topic.

Description

Message text:	This Device Server is (still) running in the error mode. Press *Retry* to repeat the initialization. If the problem persists the Device Server may be malfunctioning
May appear in:	Auto-discovery and address bookaccess modes; Initialize function

---

Details

This message appears when the DS initialization fails. Initialization restores all DS settings to their default values and, therefore, "repairs" invalid settings that caused the DS to enter the error mode. It is not normal that the DS is still in the error mode after the initialization and may indicate DS hardware failure.

Programmer's info:

The *DS Manager* verifies whether the DS is running in the error mode by sending the **Echo (X) command** and analysing the status of the ***e*** flag in the reply.

Description

Message text:	Inband (TCP) access method is selected for this Device Server. Initialization is not allowed since this would disable inband access and subsequent communications with the *DS Manager*
May appear in:	Address book access mode; Initializefunction

Details

This message is only shown when inband (TCP) access method is selected for a particular address book entry. Inband TCP access to the DS is only possible when the **Inband (IB) setting** is 1 (enabled) and the **Transport Protocol (TP) setting** is 1 (TCP). Default value of these settings is 0*. Therefore, initialization would result in the inability to access this DS using inband communications!

To initialize this DS find some other way to communicate with it:

- Temporarily connect the DS to the same network segment with your PC and initialize the DS in the auto-discovery access mode (this mode always uses out-of-band (UDP) access method).
- Temporarily connect the serial port of the DS to the COM port of your PC and initialize it in the COM access mode.

Programmer's info:

** Unless another default value has been defined through a custom profile (the DS Manager won't detect this and will still disallow the device to be initialized while accessing using inband commands).*

Description

Message text:	This Device Server is running in the error mode. This means that newly loaded firmware has some new (different) settings that need to be initialized. Do you want to do this right now?
May appear in:	Auto-discovery and address bookaccess modes; Upgrade function

Details

Newly loaded firmware may have new or different settings that have never been initialized on this particular device. After the DS has started running the newly loaded application firmware it has detected that these settings contain invalid values and entered the error mode. You are recommended to initialize the DS as soon as possible.

Programmer's info:

The *DS Manager* verifies whether the DS is in the error mode by sending the **Echo (X) command** and analysing the status of the ***e*** flag in the reply.

Description

Message text:	Input login password for this Device Server
May appear in:	Auto-discovery and address bookaccess modes; Settings, Upgrade, Initialize, Change IPfunctions

Details

Login password is the one you have set by clicking on the *Password button* in the *settings dialog*. You cannot access the DS through the network without this password.

If you have forgotten the password then you can either:

- Access the DS in the COM access mode (this does not require password), or...
- Quick-initialize the DS- this will erase the password (only supported by firmware V3.xx or higher).

Programmer's info:

Login password is the one defined by the **Password (PW) setting** of the DS. Also see authentication topic.

Description

Message text:	Firmware upgrade has failed. This may be because you are trying to upload an invalid file or because of communications error

May appear in: COM access mode; Upgradefunction

Details

This message means that either you tried to upload an invalid file or that there was a communications error. Check the file you are trying to upload, your serial connection (cable, baudrate, etc.) and try again.

The firmware file you are supposed to upload into the DS depends on the DS model number and also on the way you are performing the upgrade- through the network (auto-discovery and address book modes) or through the serial port (COM access mode). Firmware download page at <%WEB%> provides complete info on what file to choose. Be sure you understand this information and select a correct firmware file.

Programmer's info:

Description

Message text: After firmware upload and reboot the Device Server has still entered the firmware upgrade mode. This means that you have downloaded an invalid file or that the upload was incomplete

May appear in: Auto-discovery and address bookaccess modes; Upgrade function

---

Details

When the DS powers up its internal "operating system" verifies if the application firmware is loaded and correct (this is done by verifying the checksum). If the firmware is found to be corrupted the operating system checks if the NetLoader is present. When this is so, control is passed to the NetLoader so you have a chance to upload correct application firmware.

When this message is displayed this means that the firmware upload process has completed successfully and the DS was rebooted with the intention to launch the newly loaded application firmware but has emerged from reboot in the firmware upgrade mode again.

If you are sure that the firmware upgrade has completed successfully then you must have uploaded a wrong file. Firmware download page at <%WEB%> provides complete info on what file to choose.

Programmer's info:

Description

Message text: This IP-address is invalid. Setting it will result in the inability to access the Device Server through the network

May appear in: Auto-discovery access mode; Change IP function

---

Details

Certain IP-addresses are invalid in principle and should never be used. Many devices and operating systems (including *Windows*) automatically discard network packets that refer to such invalid IPs. Assigning an invalid IP to the DS can make it inaccessible over the network.

Here is the list of IP-addresses that should not be used:

- **x.x.x.0** (i.e. 0 in the last number, as in 192.168.100.0).
- **x.x.x.255** (i.e. 255 in the last number, as in 192.168.100.255).
- **>223.x.x.x** (i.e. a number that is more than 223 in the first number, as in 224.168.100.40).

Latest firmware versions of the DS prevent such invalid IPs from being used- the DS will automatically assume a modified and correct IP-address if invalid one is set (see **IP-address (IP) setting** for more information). Older firmware versions did not have this protection so the *DS Manager* itself also prevents invalid IP-addresses from being set.

Programmer's info:

IP-address is changed using the **Assign IP-address (A) command**.

Description

Message text:	Firmware upgrade was aborted by the Device Server. This may be because of communications error or because you are trying to upload a wrong file
May appear in:	Auto-discovery and address bookaccess modes; Upgrade function

---

Details

This message indicates that the *DS manager* has aborted the upload of a firmware file. Typically, this happens when the DS has detected an error in the data being uploaded. This means that you are trying to upload an incorrect firmware file.

The firmware file you are supposed to upload into the DS depends on the DS model number and also on the way you are performing the upgrade- through the network (auto-discovery and address book modes) or through the serial port (COM access mode). Firmware download page at <%WEB%> provides complete info on what file to choose. Be sure you understand this information and select a correct firmware file.

Programmer's info:

Description

Message text:	Password and re-typed password do not match
May appear in:	All access modes, Settings function

Details

When you attempt to set new login password the *DS Manager* asks you to input the same password twice. This is to make sure that you know what password you have

entered. This message appears when the password you have entered the first time does not match the password you have entered the second time.

Programmer's info:

Login password is the one defined by the **Password (PW) setting** of the DS. Also see authentication topic.

Description

Message text: There was no response from this Device Server. Make sure it is online, connected to the network, and (for address book access mode) is being accessed correctly

May appear in: Auto-discovery and address bookaccess modes; Settings, Upgrade, Initialize, Routing Status, Buzz, Change IPfunctions

---

Details

This message indicates that the *DS Manager* has attempted to access the DS (using out-of-band (UDP) programming commands) but did not get any reply. The reasons for this depend on the selected access mode: see troubleshooting (auto-discovery mode) and troubleshooting (address book mode).

Programmer's info:

Description

Message text: There was no response from the Device Server. Make sure it is powered, its serial port is connected to the COM port of this PC, and the Device Server is in the serial programming mode

May appear in: COM access mode; Settings, Initializefunctions

---

Details

This message indicates that the *DS Manager* has attempted to access the DS through the serial connection (serial port of the DS to the COM port of the PC) but did not get any reply back. See troubleshooting (COM access mode) for the list of hints on why this might happen.

Programmer's info:

Description

Message text: Requested operation cannot be completed because the Device Server is in the serial programming mode or the network programming session is in progress

May appear in: Auto-discovery and address bookaccess modes; Settings, Upgrade, Initialize, Change IPfunctions

---

Details

When you click Settings, Upgrade, Initialize, or Change IP button the *DS Manager* needs to login before it can execute requested operation (see authentication). Login is required even if the login password is not set and is only accepted if no programming with the same priority level is already in progress.

Programmer's info:

Before executing any of the operations listed above the *DS Manager* attempts to login using the **Login (L) command**. This message is displayed when the response to this command is **Rejected (R)**.

Description

Message text: This function requires out-of-band (UDP) access to the Device Server. Inband (TCP) access is currently selected for this Device Server so the function cannot be used

May appear in: Address book access mode; Upgrade, Routing Statusfunctions

---

Details

This message means that the inband (TCP) access method is selected for this particular address book entry. It is not possible to use the Upgrade and Routing status functions with "inband" address book entries (see function topics for an explanation).

Programmer's info:

Description

Message text: Password for this Device Server will be disabled

May appear in: All access modes, Settings function

Details

You have pressed *OK* without entering any password string. This means that no password will be set and the password protection for this DS will be disabled.

Programmer's info:

Login password is the one defined by the **Password (PW) setting** of the DS. Also see authentication topic.

Description

Message text: This Device Server has DHCP enabled. New IP-address may work but the use of this new IP will neither be authorized, nor noted by the

	DHCP server. Do you still want to continue?
May appear in:	Auto-discovery access mode; Change IP function

Details

This message is shown when the Change IP function is used while the IP-address of the DS appears to be configured through the DHCP (which means that **DHCP (DH) setting** is 1 (enabled) and IP-address appears to have been successfully received from the DHCP server i.e. the DS is not in the IP-address not obtained state)*.

Technically, the DS will work fine when you assign an IP-address manually (while the network has the DHCP service) as long as you find an unused IP-address**. The problem is that the DHCP server will not know that you have occupied this IP and may assign it to some other device in the future***.

Finally, keep in mind that the IP-address you set manually may be changed next time the DS reboots. This is because, when the **DHCP (DH) setting** is 1 (enabled), the DS negotiates its IP-address with the DHCP server each time it powers up or reboots and the DHCP server may assign a different IP-address at this time. This new IP will be saved into the **IP-address (IP) setting** thus overwriting the value you may have set.

Programmer's info:

The *DS Manager* verifies whether the IP-address of the DS is configured through the DHCP by sending the **Echo (X) command** and analysing the status of the ***i*** flag in the reply. If the flag is set to 'I' then this means that the DHCP is enabled and the DHCP server is actually present on the network.

** In other words, the DS Manager does not detect the presence of DHCP server directly but instead relies on the "indirect evidence" from the DS. Unfortunately, this means that the DS Manager won't be able to alert you when the DHCP server is actually present but **DHCP (DH) setting** is 0 (disabled) on the DS.*

*** This can always be done by PINGing different IPs on the local subnet. No reply means that the IP is probably unused.*

**** Unless you have specifically banned the DHCP server from using this IP-address (this can usually be done).*

Description

Message text:	Power the Device Server off, press and hold the setup button*, and power the Device Server back on while keeping the button pressed. The upgrade process will start after that
May appear in:	COM access mode; Upgradefunction

Details

Powering the DS up while keeping the setup button pressed puts the DS into the serial upgrade mode.

Programmer's info:

** On EM100, EM120, EM200, EM203(A)- pull the MD line LOW and power up while*

holding this line LOW.

Description

Message text: Press the setup button* on the Device Server

May appear in: COM access mode; Settings, Initializefunction

Details

The DS can be programmed through the serial port only when the latter is in the serial programming mode. To switch the serial port into the serial programming mode press the setup button*.

Programmer's info:

** On EM100, EM120, EM200, EM203(A)- pull the MD line LOW for at least 100ms.*

Description

Message text: Communications error was detected during the serial firmware upload. Firmware upgrade has failed

May appear in: COM access mode; Upgradefunction

Details

This message means that there was some communications error while transferring the firmware file from the PC to the DS. The message does not mean that the file itself was found to be invalid. Check your serial connection (cable, baudrate, etc.) and try again.

Programmer's info:

Description

Message text: Serial upgrade completed successfully. Switch the Device Server off and back on to start normal operation. DS initializationmay be required after the upgrade

May appear in: COM access mode; Upgradefunction

Details

After the serial upgrade the DS is not able to reboot by itself. You need to power it off and back on again to test the newly uploaded firmware. Since the new firmware can have new or different settings that were not present on a previous one the DS may require initialization (watch out for the error mode after the DS reboots).

Programmer's info:

Description

Message text: Setting description file contains error(s)

May appear in: All access modes, Settings function

Details

Setting description files (SDFs) contain the list of all settings found on a specific firmware version of the DS. During the installation SDFs are copied into the */SDF* subfolder inside the target installation folder of the Device Server Toolkit (DST). This message indicates that the SDF file required to correctly display the settings *dialog* for a particular DS firmware appears to contain invalid information. This problem can be corrected by reinstalling the software.

Programmer's info:

Description

Message text: Corrupted installation: unable to find SDF files

May appear in: All access modes, Settings function

Details

Setting description files (SDFs) contain the list of all settings found on a specific firmware version of the DS. During the installation SDFs are copied into the */SDF* folder inside the target installation folder of the Device Server Toolkit (DST). This message indicates that the */SDF* folder is empty. This problem can be corrected by reinstalling the software.

Programmer's info:

Description

Message text: Unexpected NetLoader error. Firmware upgrade has failed

May appear in: Auto-discovery and address bookaccess modes; Upgrade function

Details

The NetLoader is a separate firmware component that facilitates application firmware upgrades through the network. This message indicates that the NetLoader was entered but firmware upload could not be started because of an unexpected reply from the DS.

This message should not appear under normal circumstances. Please, contact us if the problem persists!

Programmer's info:

Description

Message text: New IP-address is unreachable

May appear in: Auto-discovery access mode; Initialize function

---

Details

After assigning new IP-address the *DS Manager* performs an automatic refresh and verifies that the DS is still online. This message appears when the DS is detected to be online but the *DS Manager* is unable to communicate with this DS using a "normal" IP addressing (see broadcast access for details).There are actually several reasons why this may be so- see troubleshooting (auto-discovery mode).

Programmer's info:

Description

Message text: Unable to send an auto-discovery broadcast because a local port from which this broadcast packet is supposed to be sent is currently in use by another program

May appear in: Auto-discovery access mode

Details

In the auto-discovery access mode the *DS Manager* is finding all locally connected Device Servers by sending an **Echo (X) command** as UDP broadcast. All Devices that receive this broadcast reply to it and the *DS Manager* is building a list of available Devices basing on received replies.

This message means that the *DS Manager* is unable to send the broadcast because local port from which the broadcast is supposed to be sent is currently opened by some other program.

This situation is extremely rare. If you encounter it please contact Tibbo for instructions on how to change the port from which the *DS Manager* is sending the broadcasts! By default, the port number used for the purpose is 65534.

Programmer's info:

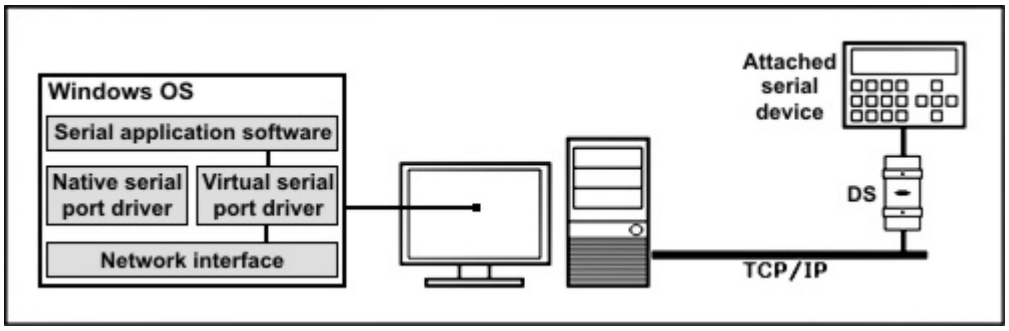
VSPD and VSP Manager

Virtual Serial Port Driver (*VSPD*) and **Virtual Serial Port Manager** (*VSP Manager*) are parts of the Device Server Toolkit (*DST*).

The *VSPD* powers **Virtual Serial Ports** (*VSPs*) that emulate "real" COM ports under *Windows OS*. The *VSP Manager* is used to add, delete, and setup *VSPs*.

The *VSPD* is an "engine" that powers **Virtual Serial Ports** (*VSPs*). To any Windows application the *VSP* "looks and feels" just like a "normal" COM port. In reality, the *VSP* transparently reroutes all data sent by the application to the Tibbo Device Server ("DS") and the serial device behind it ("attached serial device"). Likewise,

all the data sent by the serial device is received by the DS and routed to the *VSP*, which, in turn, passes this data to the application. Both the software application on the PC and the serial device communicate with each other just as if they were interconnected by a "normal" serial cable, without knowing that there is a network in between. This allows you to network-enable your existing serial system without changing the serial device itself or its PC software.



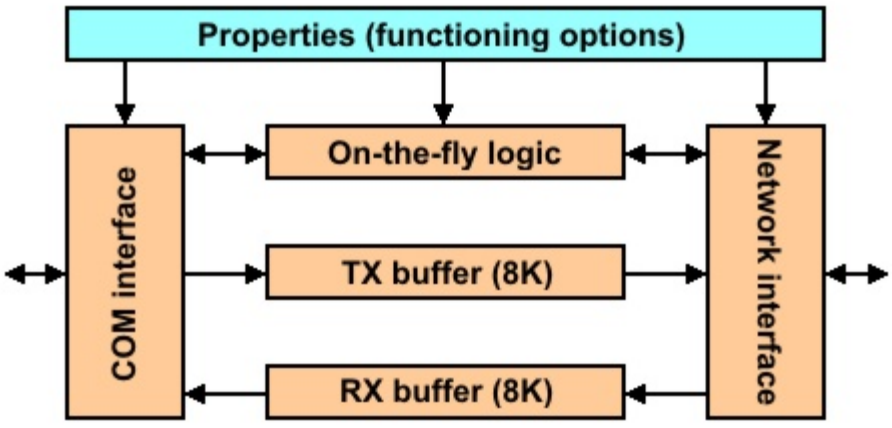
The *VSPD* is a driver and is running in the very "guts" of the *Windows OS*. Its presence is only manifested by the *VSPs* available to *Windows* applications.

The VSP Manager is an application that allows you to add, remove, and setup *VSPs* . The *VSPD* and the *VSP Manager* are different entities (one is a *driver*, another one- a *setup utility*). This *Manual* offers combined description of the two and all *VSPD* features are explained "through" the *VSP Manager*.

Closely related to the work of the *VSPD* is the Port Monitor, another member of the *DST*, that is used to log the activity of the *VSPs*.

How VSP Works

When designing the *VSPD* we have attempted to emulate the work of a standard serial port driver as closely as possible. All system calls supported by the COM driver were carefully ported into the *VSPD* and the behavior of the *VSPs* closely mimics that of COMs.



Shown above is a *VSP* block diagram:

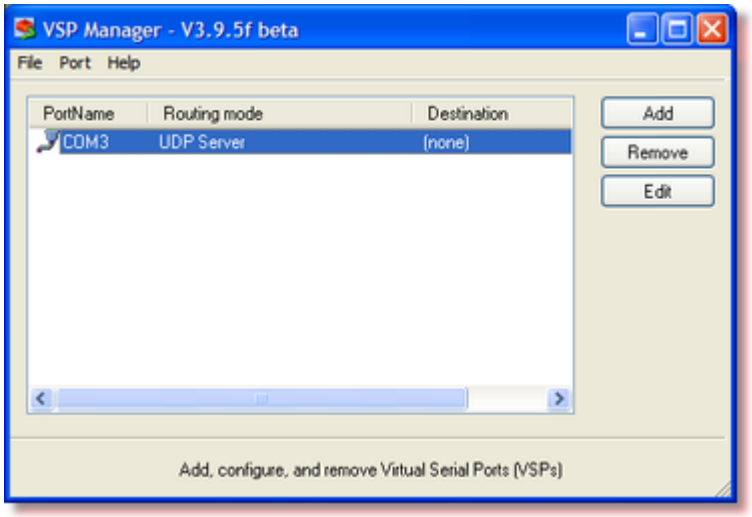
- **COM interface** presents an application interface, compatible with the standard COM port driver. *Windows* applications are not able to tell the difference between a *VSP* and a regular COM port.
- **TX and RX buffers** (8KBytes each) are used to pass the data between the application and the DS. It is noteworthy that *VSP* operation is fully asynchronous, and this is *different* from the operation of a standard COM. When

the application "writes" the data into the *VSP* the data is stored into the TX buffer and the control is returned to the application immediately, not when this data is actually sent out (as is the case with COM ports).

- **Network interface** communicates (through the TCP/IP network) with the target DS. The *VSP* transparently establishes and accepts data connections with/from the DS as needed.
- **On-the-fly logic** (when enabled) is responsible for adjusting communications parameters of the serial port on the DS to the requirements of the application. For example, if the application wants the serial port to run at 19200 bps a special on-the-fly command will be sent to the DS telling it to change the serial port baudrate to 19200. Thus, the DS serial port functions just like the COM and the PC!
- ***VSP* properties** define different aspects of *VSP* operation. The properties are stored in the *system registry* and edited using the VSP Manager.

VSP Manager

VSP Manager is used to add, remove, and setup *VSPs*. It looks like this:

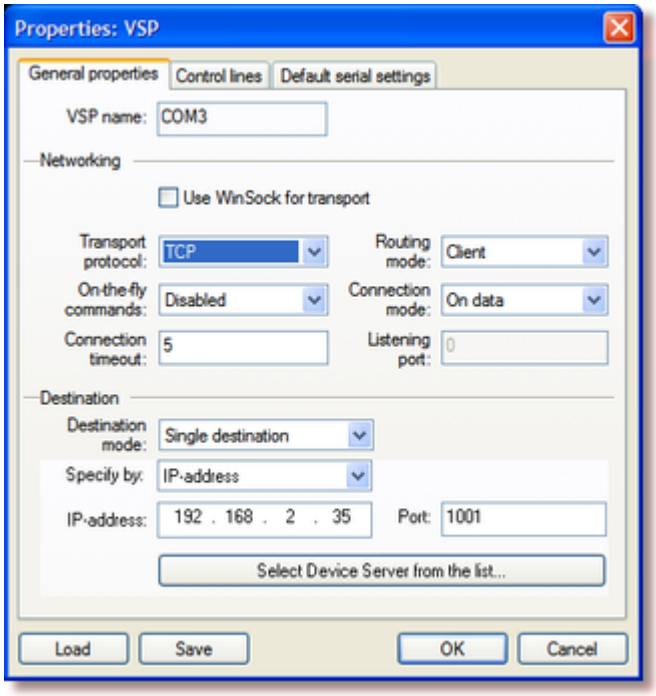


The main window has the following areas and controls:

- ***VSP* list** shows all *VSPs* currently found on your PC.
- **Add, Remove, and Edit buttons** are used to add *VSPs*, delete *VSPs*, and edit *VSP* properties. Clicking *Add* or *Edit* brings up the *VSP properties dialog* shown on the screenshot above. You can also open the *dialog* by double-clicking on the *VSP* in the *VSP list*.
- **File menu** contains **Import** and **Export** commands which allow you to load or save a list of VSPs and their configurations using external plain-text files. This is useful for migrating VSP configurations between computers, or for troubleshooting (sending a configuration to a support engineer so he could recreate it).
- **Port menu** contains **Add, Remove and Edit** commands (see above).
- **Help menu** allows you to access the document you are now reading (the online help) or the About box for Tibbo Device Server Toolkit.

VSP Properties

The VSP Properties window contains the following tabs:



- **General properties tab** (shown on the screenshot) provides a set of "main" controls that guide *VSP* operation. For more information select the topic from the list belowJA.
- **Control lines tab** offers additional options for control line inputs CTS, DSR, and DCD. You can choose to receive input line status updates from the DS, "fix" these inputs at high or low, or "connect" CTS to RTS and DSR to DTR.
- **Default serial settings tab** exists purely for compatibility with regular COM ports. Under *Windows*, each COM port has a set of default parameters associated with it, and so does *VSP*. These parameters *do not affect* operation of the *VSP* in any way.

The General Properties tab (shown above) has the following areas and controls:

- **Use WinSock for transport checkbox** sets whether TDI (Transport Device Interface, the default interface) is used for transport, or WinSock.
- **VSP name drop-down box** selects the port name that will be associated with this *VSP* (i.e. "COM3", etc.).
- **Transport protocol drop-down box** defines which transport protocol- TCP/IP or UDP/IP will be used for data communications with the DS.
- **On-the-fly commands drop-down box** disables or enables the generation of on-the-fly commands, used to adjust the serial port parameters on the DS as needed by the application software (that uses the *VSP*). There is also a choice of how the on-the-fly commands will be sent when enabled (as out-of-band or inband commands).
- **Connection timeout parameter** defines after how many minutes of inactivity (no data transmitted across the data connection between the *VSP* and the DS) the current data connection will be aborted.

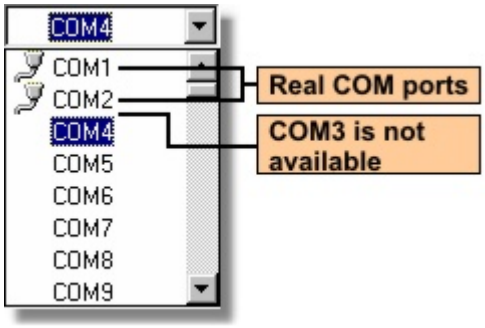
- **Routing mode drop-down box** defines whether the *VSP* will accept incoming connections (passive opens) and/or establish outgoing connections (perform active opens).
- **Connection mode drop-down box** defines when the *VSP* will attempt to establish an outgoing connection to the destination: right after the *VSP* is opened by the application or when the application sends the first data.
- **Listening port parameter** defines the listening port that will be associated with this *VSP*. The *VSP* will be accepting incoming connections (passive opens) on this port (when allowed by the routing mode).
- **Destination mode drop-down box** offers two destination modes that define how the *VSP* will choose its destination DS: a simple single destination mode that targets one DS and a more complex multi-destination mode that makes the *VSP* switch between several Device Servers basing on the outgoing data sent by the application.
- **Specify by drop-down box** defines how the address of the destination is to be specified - by IP address, by MAC address or by DNS hostname.

VSP Name Selection

***VSP* name drop-down box** selects the port name that will be associated with this *VSP*.

The number of ports you can have under *Windows* is virtually unlimited. The *VSP Manager* lets you create any port in the range between COM1 and COM255*.

There are no rules on what name to choose, just make sure that you pick the name that can be selected in the application software you plan to use this *VSP* with. Most programs provide a limited selection of ports (typically, up to COM2 or COM4). Therefore, choosing "COM100" wouldn't be suitable as you will not be able to select this *VSP* in such software.



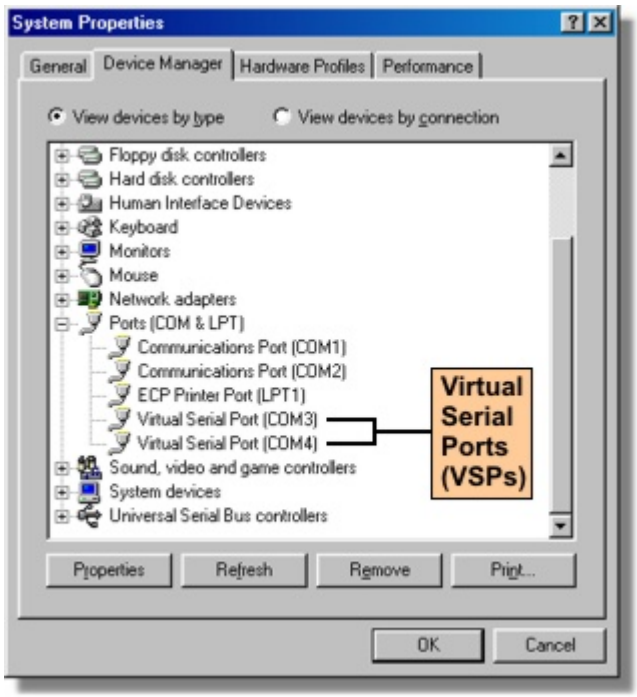
The *VSP name drop-down box* shows the list of available port names. On the screenshot above COM3 is not listed- this happens when the *VSP* with this name already exists. Notice, however, that ports COM1 and COM2 are not excluded from the list and are marked with icons identifying them as "real" COMs**. Even though these port names are "occupied" the *VSP Manager* can still "grab" them. For example, if you select COM1 the *VSP Manager* will automatically substitute a standard COM port driver with the *VSPD* and assign the name "COM1" to this *VSP*. From this moment on the COM1 will cease working as a regular COM and will start working as a *VSP*. Substitution may be necessary when you are dealing with an old application software that only provides a choice of COM1 and COM2 which are usually occupied by real COMs.

When you delete the *VSP* that substituted a standard COM port the *VSP Manager*

will attempt to restore the original driver. This *mostly* works but on some systems you may encounter problems (especially under *Windows ME*). We have conducted an extensive "research" into the port substitution, only to conclude that it just doesn't work reliably on all systems!

We recommend that you do not use port substitution unless absolutely necessary!

All *VSPs* appear under the *Ports section* of the *device manager's device list* (*Control Panel--> System Properties--> Device Manager*)***:



** We could extend the number even further but feel that the current range is sufficient for all practical purposes.*

*** This is system-dependent. Some modern PCs don't have any real COMs.*

**** VSP properties cannot be edited from within the device manager*

Transport Protocol

Transport protocol drop-down box selects which communications protocol- TCP/IP or UDP/IP will be used by the *VSP* for data communications with the DS.

For the data connection with the DS to work the same transport protocol must be selected on the DS side- see the **Transport Protocol (TP) setting**.

Unless you have a specific reason why the UDP should be used (very rare!) we recommend you to stick to the TCP/IP. The TCP/IP is a "reliable delivery" protocol that makes sure that no data is lost "in transit" between the *VSP* and the DS. The UDP, on the contrary, does not guarantee data delivery.

Some considerations and additional info on the TCP and UDP implementation in the *VSP* can be found in the next topic.

Additional Info on UDP and TCP Connections

UDP data connections

The notion of data connection is native to TCP/IP since this is a connection-based protocol. UDP/IP, however, is a connection-less protocol in which all packets (UDP datagrams) are independent from each other. How, then, the term "data connection" applies to the UDP transport protocol?

With UDP transport protocol true data connections (in the "TCP sense" of this term) are not possible (hence, parenthesis around the word "connection"). The *VSP*, however, attempts to mimic the behavior of TCP data connection whenever possible. Follows is the detailed description of UDP "connections" and their similarities and differences with TCP connections.

Incoming "connections"*. There is no connection establishment phase in UDP so an incoming UDP "connection" is considered to be "established" when the first UDP packet is received by the *VSP* (on the listening port). Similarity with TCP is in that after having received the packet from the DS the *VSP* knows who to send its own UDP packets to.

Outgoing "connections".** The *VSP* establishes outgoing UDP connection by sending a UDP datagram to the targeted destination. If there is a data that needs to be transmitted the *VSP* sends the first UDP datagram with (a part of) this data. If there is no immediate data that needs to be transmitted to the DS the *VSP* sends the first UDP datagram of zero length (this happens when the connection mode is set to "connect immediately"). The purpose of this is to let the other side know the IP-address of the *VSP* (PC it is running on), as well as the data port currently used by the *VSP*.

Data transmission and destination switchover. Once the "connection" is established the *VSP* and the DS exchange the data using UDP datagrams. The difference with TCP is that if another DS sends a datagram to the *VSP*, then the *VSP* will interpret this as a new incoming connection*, forget about the first DS and start sending its own UDP datagrams to the second one. In other words, the *VSP* will always communicate with the "most recent sender". Such behavior is not possible in TCP, in which a third party cannot interfere with an existing connection.

"Connection" termination. There is no connection termination phase in UDP so *VSP* "terminates" its UDP connections by forgetting about them and the only event that can trigger UDP "connection" termination (except for the closing of the *VSP*) is connection timeout.

Local port used by the *VSP* depends on the selected routing mode:

- **In the server routing mode** the *VSP* sends the UDP datagrams from an "automatic" port selected by the OS;
- **In the server/client and client routing modes** the *VSP* sends and receives UDP datagrams through the port, defined by the listening port parameter.

TCP data connections

Only one connection at a time. TCP protocol stack on the PC is capable of supporting thousands of concurrent TCP connections but the *VSPD* strictly enforces that only a single TCP connection exists for each *VSP* at any time. This is because the serial port is not a shared media and allowing, say, two Device Servers to connect to the same *VSP* would have created a data chaos***. Allowing only a single connection at a time follows a "serial port culture" of "one serial port- one application"! If the *VSP* is already engaged in a data connection with the DS and

another DS attempts to establish a connection to this *VSP* then this DS will be rejected.

Separate ports for outgoing and incoming connections. The *VSP* establishes its outgoing connections** from an "automatic" port selected by the OS. Each time such outgoing connection is established the source port number on the *VSP* side will be different. The *VSP* accepts incoming connections* on a fixed listening port, whose number is defined by the listening port parameter. When the incoming connections are not allowed the listening port is closed.

** Assuming that incoming connections are allowed (i.e. the routing mode is either "server", or "server/client").*

*** Assuming that outgoing connections are allowed (i.e. the routing mode is either "client", or "server/client").*

**** What if both Device Servers started sending the data to the VSP at the same time? Then the PC application using the VSP would have received a mix of data consisting of an input from both Device Servers! And if two Device Servers were connected to the same VSP and the PC application needed to send out the data then which DS of the two the VSP would have to send this data to?*

On-the-fly Commands

On-the-fly commands are used to change the serial port configuration of the DS as needed (i.e. "on the fly"). Serial port configuration made through the on-the-fly commands overrides the permanent one, defined by the serial port settings of the DS. The difference between the changes made using on-the-fly commands and changes made through altering DS settings is that, unlike serial settings, on-the-fly commands have immediate effect and do not require the DS to be rebooted in order for the new values to be recognized.

With on-the-fly commands enabled, the serial port of the DS is always setup as required by the PC application that communicates with this DS through the *VSP*. When the PC application opens the *VSP* (or some communications parameters are changed) the application informs the *VSP* about required changes* and the *VSP* relates this information to the DS by sending on-the-fly commands.

Additionally, on-the-fly commands are used by the *VSP* to control the RTS and DTR outputs of the DS serial port. The status of the CTS and DSR input of the DS serial port can be passed to the *VSP too*- this is done using so-called "notification messages". For more information see handling of RTS, CTS, DTR, and DSR signals.

On-the-fly commands drop-down box provides four choices:

Disabled	On-the-fly commands are not sent at all, so the serial port of the DS will use "permanent" serial port configuration defined by the serial port settings. In this mode it doesn't matter what serial port parameters are set in the PC software application- the DS will not be aware of them!
Out-of-band	On-the-fly commands are enabled and sent in the form of out-of-band (UDP) commands. **On-the-fly commands (RC) setting** of the DS must be programmed to 1 (enabled) for the out-of-band on-the-fly commands to be accepted.
Inband	On-the-fly commands are enabled and sent in the form of inband (TCP) commands. **On-the-fly commands (RC) setting** of the DS must be programmed to 1

(enabled) for the on-the-fly commands to be accepted. Additionally, there are some other programming steps that must be performed before the DS will recognize inband commands- see preparing the DS for inband access.

Disabled (w FF esc.) On-the-fly commands are not sent, but the *VSP* treats all incoming and outgoing data as if *inband* mode was used (i.e. it doubles all "escape" characters (ASCII code 255) in the data sent by the application and expects all escape characters to be doubled in the data stream sent by the DS). See disabled (with FF escape) mode of the VSP for details.

In general, we recommend you to keep on-the-fly commands enabled (unless there are some special reasons preventing you from doing so). Enabling on-the-fly commands keeps the serial port setup of the DS "in sync" with the requirements of the software application using the *VSP*.

As for choosing between out-of-band and inband modes, follow these recommendations:

Out-of-band commands work most of the time, especially when the PC (running *VSP*) and the DS are located on the same network segment**. Out-of-band commands may not work very well or not work at all for the remote Device Servers located behind the routers, firewalls, etc***. This is because:

- Routers are known to "drop" UDP datagrams (on which out-of-band on-the-fly commands are based) under heavy network traffic.
- UDP traffic is banned by the firewalls of many networks (hence, out-of-band on-the-fly commands cannot be used at all). If you want out-of-band on-the-fly commands to work then your network must allow UDP traffic to port 65535!

If you encounter one of the above situations then you should use inband on-the-fly commands or not use on-the-fly commands at all!

There is one other reason why out-of-band commands may not be suitable- this is when on-the-fly commands must be synchronized with the data sent by the *VSP*. For more information see synchronization issues.

On-the-fly command-related activity of the *VSP* is best observed using the Port Monitor, as all on-the-fly commands as well as the result of their execution are logged- see next section for details.

** This is standard for Windows COM ports.*

*** The definition of the network segment implies that there are only network hubs (and no routers, bridges, firewalls, etc.) between the PC and all other devices on the segment.*

**** Here we touch on a very complicated subject. Modern routers offer a bewildering array of setup options. We will attempt to cover this in details in our upcoming white papers.*

When the VSP Sends On-the-fly Commands

This topic details when and what on-the-fly commands the *VSP* sends to the DS.

At certain times the *VSP* sends an entire "parameter block" of required communications parameters to the DS:

- Parameter block is sent when the *VSP* is just opened, except when the routing

mode is "server".

- Additionally, parameter block is sent each time the data connection is established (no matter whether this was an incoming or outgoing connection).

Parameter block includes all communications parameters needed by the serial port, such as the baudrate, parity, etc.

In addition to sending parameter blocks the *VSP* also sends individual commands whenever some parameter is changed within the application. For example, if the User chooses a different baudrate (while the application is already running and the *VSP* is opened) the *VSP* won't send out entire parameter block again but will instead send just the on-the-fly command to change the baudrate of the DS.

Such individual on-the-fly commands are sent immediately after the serial communications parameters are changed in the software application, except for the case when the routing mode is "server" and no data connection is established at the moment. In this case the *VSP* will have to wait until it receives an incoming connection, and this will trigger an entire parameter block to be sent, as described above.

Here is an example log from the Port Monitor detailing one "cycle" of the *VSP* operation (out-of-band on-the-fly commands are enabled, routing mode is not "server"):

```
--- application is started  ---
12/15/03 14:22:47 - COM3 (INFO): Port opened
--- beginning of parameter block (this block is sent because the VSP has been opened) ---
12/15/03 14:22:47 - COM3 (INFO): "On-the-Fly" command for 192.168.100.92: get DSR pin status...success
12/15/03 14:22:47 - COM3 (INFO): "On-the-Fly" command for 192.168.100.92: get CTS pin status...success
12/15/03 14:22:47 - COM3 (INFO): "On-the-Fly" command for 192.168.100.92: set DTR to high...success
12/15/03 14:22:47 - COM3 (INFO): "On-the-Fly" command for 192.168.100.92: set RTS to high...success
12/15/03 14:22:47 - COM3 (INFO): "On-the-Fly" command for 192.168.100.92: set baud rate to 38400 bps...success
12/15/03 14:22:47 - COM3 (INFO): "On-the-Fly" command for 192.168.100.92: set parity to none...success
12/15/03 14:22:47 - COM3 (INFO): "On-the-Fly" command for 192.168.100.92: set data bits to 8 bits...success
12/15/03 14:22:47 - COM3 (INFO): "On-the-Fly" command for 192.168.100.92: set flowcontrol to RTS/CTS...success
12/15/03 14:22:47 - COM3 (INFO): "On-the-Fly" command for 192.168.100.92: enabling line change notification for
DSR...success
--- end of parameter block ---
---
--- connection is established (i.e. because the application sends data) ---
12/15/03 14:22:54 - COM3 (INFO): Established TCP connection with node 192.168.100.92:1001
--- beginning of parameter block (this block is sent because the connection has been established) ---
12/15/03 14:22:54 - COM3 (INFO): "On-the-Fly" command for 192.168.100.92: get DSR pin status...success
12/15/03 14:22:54 - COM3 (INFO): "On-the-Fly" command for 192.168.100.92: get CTS pin status...success
12/15/03 14:22:54 - COM3 (INFO): "On-the-Fly" command for 192.168.100.92: set DTR to high...success
12/15/03 14:22:54 - COM3 (INFO): "On-the-Fly" command for 192.168.100.92: set RTS to high...success
12/15/03 14:22:54 - COM3 (INFO): "On-the-Fly" command for 192.168.100.92: set baud rate to 38400 bps...success
12/15/03 14:22:54 - COM3 (INFO): "On-the-Fly" command for 192.168.100.92: set parity to none...success
12/15/03 14:22:54 - COM3 (INFO): "On-the-Fly" command for 192.168.100.92: set data bits to 8 bits...success
12/15/03 14:22:54 - COM3 (INFO): "On-the-Fly" command for 192.168.100.92: set flow control to
RTS/CTS...success
12/15/03 14:22:54 - COM3 (INFO): "On-the-Fly" command for 192.168.100.92: enabling line change notification for
DSR...success
--- end of parameter block ---
---
--- the following commands are sent in response to the User changing the baudrate within the application ---
12/15/03 14:22:56 - COM3 (INFO): "On-the-Fly" command for 192.168.100.92: set baud rate to 19200 bps...success
12/15/03 14:22:58 - COM3 (INFO): "On-the-Fly" command for 192.168.100.92: set baud rate to 9600 bps...success
12/15/03 14:22:59 - COM3 (INFO): "On-the-Fly" command for 192.168.100.92: set baud rate to 19200 bps...success
12/15/03 14:23:00 - COM3 (INFO): "On-the-Fly" command for 192.168.100.92: set baud rate to 9600 bps...success
--- application is closed ---
12/15/03 14:23:02 - COM3 (INFO): TCP connection closed
12/15/03 14:23:02 - COM3 (INFO): Port closed
```

A slightly different behavior is observed when the inband mode is selected for on-the-fly commands. Since inband commands are passed within the TCP data connection, the *VSP* establishes such a connection in case it needs to send on-the-fly command(s) and no connection is established at the moment. Again,

this is only done when the routing mode is not "server".

Here is another output, this time for inband on-the-fly commands (routing mode is not "server"):

```
--- application is opened ---
12/17/03 09:30:13 - COM3 (INFO): Port opened
--- TCP connection is established because the VSP needs to send a parameter block ---
12/17/03 09:30:13 - COM3 (INFO): Established TCP connection with node 192.168.100.92:1001
--- beginning of parameter block ---
12/17/03 09:30:13 - COM3 (INFO): "On-the-Fly" command for 192.168.100.92: get DSR pin status...success
12/17/03 09:30:13 - COM3 (INFO): "On-the-Fly" command for 192.168.100.92: get CTS pin status...success
12/17/03 09:30:13 - COM3 (INFO): "On-the-Fly" command for 192.168.100.92: set DTR to high...success
12/17/03 09:30:13 - COM3 (INFO): "On-the-Fly" command for 192.168.100.92: set RTS to high...success
12/17/03 09:30:13 - COM3 (INFO): "On-the-Fly" command for 192.168.100.92: set baud rate to 38400 bps...success
12/17/03 09:30:13 - COM3 (INFO): "On-the-Fly" command for 192.168.100.92: set parity to none...success
12/17/03 09:30:13 - COM3 (INFO): "On-the-Fly" command for 192.168.100.92: set data bits to 8 bits...success
12/17/03 09:30:13 - COM3 (INFO): "On-the-Fly" command for 192.168.100.92: set flowcontrol to none...success
12/17/03 09:30:14 - COM3 (INFO): "On-the-Fly" command for 192.168.100.92: (unknown)...success
12/17/03 09:30:14 - COM3 (INFO): Line status change notification: DSR:low CTS:low
--- end of parameter block ---
---
--- connection is aborted because no data is being transmitted across this connection
12/17/03 09:31:14 - COM3 (INFO): TCP connection aborted by remote node
---
--- flow control mode must be changed, and the VSP establishes the TCP connection again
12/17/03 09:34:00 - COM3 (INFO): Established TCP connection with node 192.168.100.92:1001
12/17/03 09:34:00 - COM3 (INFO): "On-the-Fly" command for 192.168.100.92: set flowcontrol to
RTS/CTS...success
```

The *VSP* employs a retry mechanisms that makes up to 3 additional attempts in case no reply is received from the DS. If, after 4 attempts there is still no reply the *VSP* disables on-the-fly command generation:

```
12/16/03 10:55:58 - COM3 (WARNING): "On-the-Fly" command for 192.168.100.92: set baud rate to 9600
bps...timed out, still trying...
12/16/03 10:56:01 - COM3 (WARNING): "On-the-Fly" command for 192.168.100.92: set baud rate to 9600
bps...timed out, still trying...
12/16/03 10:56:04 - COM3 (WARNING): "On-the-Fly" command for 192.168.100.92: set baud rate to 9600
bps...timed out, still trying...
12/16/03 10:56:04 - COM3 (ERROR): "On-the-Fly" command for 192.168.100.92: set baud rate to 9600
bps...timed out
12/16/03 10:56:05 - COM3 (INFO): "On-the-Fly" commands disabled (until port opened next time)
```

No additional attempts are made if the DS denies or rejects any on-the-fly command:

```
12/16/03 11:03:33 - COM3 (ERROR): "On-the-Fly" command for 192.168.100.92: get DSR pin status...access
denied
12/16/03 11:03:34 - COM3 (INFO): "On-the-Fly" commands disabled (until port opened next time)
```

Handling of RTS, CTS, DTR, and DSR Signals

The status of RTS, CTS, DTR, and DSR lines can be exchanged between the *VSP* and the DS. This means that the *VSP* can remotely control the state of RTS and DTR outputs of the serial port on the DS while the DS can notify the *VSP* of the state changes on the CTS and DSR inputs of its serial port. This topic provides detailed information on the subject.

RTS and DTR outputs of the DS serial port

Whenever the application requires the state of the RTS or DTR line to be changed the *VSP* sends an appropriate on-the-fly command to the DS. Here is how this is reflected in the Port Monitor:

```
12/16/03 09:46:24 - COM3 (INFO): "On-the-Fly" command for 192.168.100.92: set RTS to low...success
12/16/03 09:46:27 - COM3 (INFO): "On-the-Fly" command for 192.168.100.92: set RTS to high...success
```

```
12/16/03 10:05:06 - COM3 (INFO): "On-the-Fly" command for 192.168.100.92: set DTR to low...success
12/16/03 10:05:06 - COM3 (INFO): "On-the-Fly" command for 192.168.100.92: set DTR to high...success
```

The DS always replies to each such command with the **A (OK) status code***. Whether or not the DS *actually processes* the command depends on the current DS setup:

If the application is running with RTS/CTS flow control disabled (which means the *VSP* has sent a **Flow Control (FC) parameter** of 0) the DS does process all RTS-related on-the-fly commands. If the application is running with RTS/CTS flow control enabled (the *VSP* has sent a **Flow Control (FC) parameter** of 1) the DS ignores all RTS-related on-the-fly commands (but still replies with **A**). This is because in this mode the DS controls the RTS line on its own and the function of the RTS line is to regulate the flow of data from the attached serial device into the serial port of the DS (also see **Flow Control (FC) setting**).

As for the DTR line, whether or not the DS actually processes DTR-related on-the-fly commands depends on the **DTR Mode (DT) setting**. If this setting is 0 (idle) the DS does process all DTR-related on-the-fly commands. If this setting is 1 (connection mode) the DS ignores all such commands. This is because in this mode the DS controls the DTR line on its own and the function of the DTR line is to reflect current status of the data connection.

CTS and DSR inputs of the DS serial port

Changes in the states of CTS and DTR inputs of the DS serial port are delivered to the *VSP* through **Notification (J) messages**. For the notifications to work the DS must first be informed the status change of which lines should be reported to the *VSP*. This is done through the **Notification Bitmask (NB) parameter**. Depending on the flow control mode selected by the application the *VSP* sets the bitmask in one of the two ways:

- If the application is running with RTS/CTS flow control disabled, the *VSP* programs the DS to react to the changes of both the CTS and DSR lines.
- If the application is running with RTS/CTS flow control enabled, the *VSP* programs the DS to react to the changes of the DSR line only. This is because when the flow control is enabled the CTS line is handled on the DS "level"- it is used to regulate the flow of serial data from the DS into the serial device.

Since the bitmask depends on the selected flow control it is re-programmed each time the application changes the flow control mode:

```
--- application disables RTS/CTS flow control so the bitmask is set to include the CTS line ---
12/16/03 10:10:18 - COM3 (INFO): "On-the-Fly" command for 192.168.100.92: set flowcontrol to none...success
12/16/03 10:10:18 - COM3 (INFO): "On-the-Fly" command for 192.168.100.92: enabling line change notification for
DSR, CTS...success
---
--- application enables RTS/CTS flow control so the bitmask is set to exclude the CTS line ---
12/16/03 10:10:20 - COM3 (INFO): "On-the-Fly" command for 192.168.100.92: set flowcontrol to RTS/CTS...success
12/16/03 10:10:20 - COM3 (INFO): "On-the-Fly" command for 192.168.100.92: enabling line change notification for
DSR...success
```

Additional considerations concerning notification messages

Just as on-the-fly commands can be sent using out-of-band or inband access method, notification messages can also be sent out-of-band or inband. When the **Inband (IB) setting**of the DS is programmed to 1 (enabled) the DS sends inband notifications (otherwise, the DS sends out-of-band notifications).

For the inband notifications to work no additional DS setup is required (past the pre-programming needed to make the DS accept inband commands in principle- see preparing the DS for inband access).

For the out-of-band notifications to work properly the **Notification Destination**

(ND) setting of the DS must be programmed to 0 (last known port). In this case the DS will send its notifications to the port from which it last received an on-the-fly command. Because the *VSP* sends on-the-fly commands from an "automatic" port the DS has no way of knowing beforehand what this port number will be. Therefore, the DS won't send notifications unless it receives at least one on-the-fly command from the *VSP*. This is not a limitation since on-the-fly commands are generated by the *VSP* as soon as it is opened and/or connection is established (see when the VSP sends on-the-fly commands).

There are certain limitations associated with notification messages:

- Notifications are only generated when the data connection is established between the *VSP* and the DS. This is different from on-the-fly commands which can be sent by the *VSP* at any time, even when the data connection is not established yet.
- The change in the state of the CTS or DSR input of the DS serial port is recognized with a 20 millisecond delay. That is, the line must change the state and remain in this new state for at least 20 milliseconds for this change to be detected and notification to be sent to the *VSP*.

* *That is, if on-the-fly commands are enabled in principle.*

Synchronization Issues for On-the-fly Commands

This topic details a subtle difference between the timing of out-of-band and inband on-the-fly commands (relative to the data being sent by the *VSP*).

In certain serial systems transmission of data by the PC is synchronized with toggling of RTS or DTR outputs (or with changing of the serial port setup). Consider the following two examples:

Example 1: The RTS line is used to mark the beginning and the end of the data transmission by the PC.

The diagram of such data transmission is shown below. When the PC wants to send out the data it does the following:

- Sets the RTS line to HIGH.
- Transmits the data.
- Sets the RTS line to LOW.

The positive pulse on the RTS line is said to envelope or encapsulate the data.

Example 2: *mark* and *space* parity bits are used by the serial device to distinguish between the *address* and *data* bytes transmitted by the PC.

In most RS485 systems (where there are multiple serial devices attached to the serial bus) the PC needs to select a particular "node" it wants to communicate with by transmitting the address of this node. Such systems often use the parity bit to

distinguish between the node address and the actual data, i.e. (parity= *mark* means an address byte, parity= *space* means a is data byte, or vise versa).

Since a standard COM driver does not allow the application software to control the parity bit directly, such software uses the following algorithm to transmit the node address followed by the data:

- Change the parity to *mark*.
- Send the node address.
- Change the parity to *space*.
- Send the data for this node.

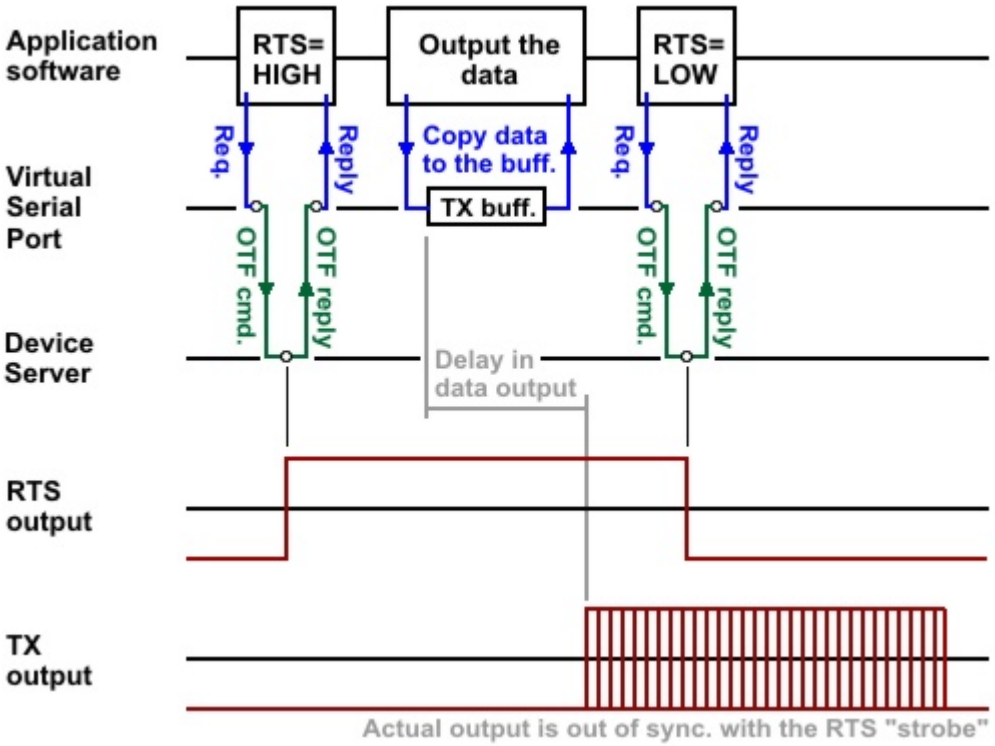
The point here is that changes in the parity mode must be synchronized with the data being sent out.

How the use of *VSP* affects synchronization

The operation of the *VSP* is more asynchronous than that of a standard serial port. When the application sends out the data the *VSP* just puts this data into the Tx buffer and returns control back to the application. To the application this means that the data has been sent out. Obviously, this is not so and there will be a certain delay before this serial data actually comes out of the serial port on the DS side.

Out-of-band commands (such as the command that tells the DS to change the state of the RTS line) are sent separately from the data. For example, when the application tells the *VSP* to change the state of the RTS line the *VSP* sends a required out-of-band command immediately.

For the example 1 the above means that the pulse on the RTS line will be completely out of sync with the serial data itself (the diagram below illustrates this). For the example 2 this means that the switching of the parity mode will occur at wrong (and unpredictable) times and, therefore, there is no guarantee that the address byte will be output with mark parity bit and the data byte will be output with space parity bit!



The solution

Fortunately, there is a solution to this problem- use inband mode when trying to achieve full synchronization between the Tx data and on-the-fly commands! Because inband on-the-fly commands are mixed into the data stream itself there is no chance that these commands will somehow "pass over" the data. The DS will receive the data and corresponding on-the-fly commands in exactly the same order as generated by the application.

Disabled (With FF Escape) Mode of the VSP

Some comments should be made on this mode of the *VSP*. The mode exists so that you could prevent the *VSP* from sending on-the-fly commands to the DS while at the same be able to program the DS from the DS Manager using inband access method.

Inband programming is usually used when out-of-bandaccess is not possible. For the inband programming to work the DS must be configured for inband access. In this case the DS treats all data characters with ASCII code 255 (FF Hex) in a special way: all ASCII code 255 characters sent from the *VSP* to the DS must be doubled, or the DS will mistake these data characters for a beginning of inband commands (see inband commands for details). In the *inband on-the-fly* mode the *VSP* does this "doubling" but it also sends on-the-fly commands. If you don't want the on-the-fly commands to be sent then use the *disabled (with FF escape)* on-the-fly mode- the *VSP* will refrain from sending any on-the-fly commands but will still "double" all outgoing FF characters. Likewise, the *VSP* will correctly handle all "double" FF characters sent by the DS.

Connection Timeout

Connection timeout parameter sets the timeout (in minutes) for the data connections between the *VSP* and the DS. If no data is transmitted across the data connection (TCP or UDP) for a specified number of minutes the *VSP* aborts the connection.

Setting connection timeout parameter to 0 disables the feature and the connection is never closed on timeout.

Notice, that this feature will work even when the connection mode is set to "immediately". When the timeout comes the *VSP* closes the connection first and immediately reopens it (timeout counter is reloaded). This provides additional reliability since hanged connections are automatically "repaired".

Connection timeout parameter works like the **Connection Timeout (CT) setting** of the DS.

Routing Mode

Routing mode defines whether the *VSP* will accept incoming connections (passive opens) and/or establish outgoing connections (perform active opens).

***Routing Mode drop-down box* provides three choices:**

Server	Only incoming connections are accepted, the *VSP* never attempts to establish an outgoing connection to the DS. There is no restriction on which DS can connect to the *VSP*- connection from any IP-address will be accepted as long as the DS is connecting to the correct listening port using correct transport protocol.
Server/client	Both incoming and outgoing connections are allowed. Outgoing connections are established with the destination, specified in the destination section of the *dialog*. Exactly when the *VSP* attempts to establish an outgoing connection is defined by the selected connection mode. Incoming connections are accepted from any IP-address, just like with the server routing mode.
Client	Only outgoing connections are allowed, the *VSP* rejects all incoming connections.

Routing mode option works like the **Routing Mode (RM) setting** on the DS.

Connection Mode

Connection mode defines when the *VSP* attempts to establish an outgoing connection to the destination, specified in the destination section of the *dialog*.

***Connection mode drop-down box* provides two choices:**

Immediately	The *VSP* attempts to establish an outgoing connection right after it is opened by the application. The *VSP* also tries to make this connection "persistent". If the connection is aborted by the DS, the *VSP* will (attempt to) re-establish it immediately. Connection timeout (defined by the connection timeout parameter) still works in this mode: when the current connection times out the *VSP* aborts it and immediately establishes a new connection. Such behavior

	"auto-repairs" hanged connections.
On data	The *VSP* attempts to establish an outgoing connection when the first "serial" data (since the *VSP* was opened or previous connection was closed/aborted) is sent by the PC application into the *VSP*.

Connection mode option is irrelevant when the routing mode is "server", since in this mode outgoing connections are not allowed in principle.

Connection mode option works like the **Connection Mode (CM) setting** of the DS (modes 0 and 1 only).

Listening Port

Listening port parameter defines the listening port number that will be associated with this *VSP*.

Listening port usage is slightly different for TCP/IP and UDP/IP transport protocols:

- **For TCP/IP transport protocol** this parameter defines a listening port on which incoming connections are to be accepted. Listening port is closed and irrelevant when the routing mode is "client", since in this mode incoming connections are not allowed in principle. Outgoing connections, when allowed (i.e. in the "client" and "server/client" routing modes), are established from an "automatic" port, so the listening port number has nothing to do with this.
- **For UDP/IP transport protocol** the situation is as follows:
  - When the routing mode is "client" (incoming "connections" are not allowed) the listening port parameter is irrelevant and the *VSP* sends it own UDP datagrams from an "automatic" port.
  - When the routing mode is "server" or "client/server" the listening port parameter defines the listening port on which incoming UDP datagrams are accepted and is also used as the port from which outgoing UDP datagrams are sent.

Listening port parameter is similar to the **Port Number (PN) setting** of the DS.

Destination Modes

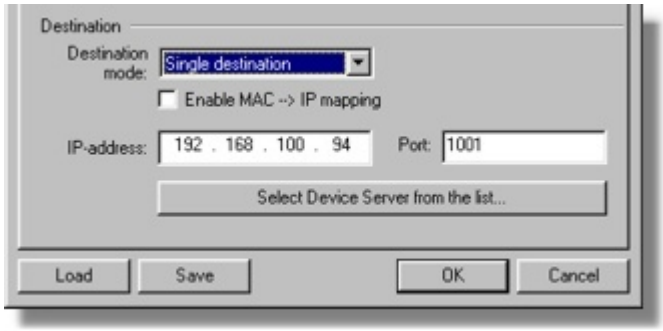
The *VSP* has two "destination" modes that define which destination the *VSP* will attempt to connect to*:

- **In the single-destination mode** there is only one destination specified either by its IP-address or MAC-address.
- **In the multi-destination mode** the *VSP* maintains a table of destinations. The *VSP* switches between these destinations basing on the data sent by the application. Multi-destination mode is a "logical" equivalent of a "multi-node" RS485 network.

** When outgoing connections are allowed in principle, i.e. in the client or client/server routing mode.*

Single-destination Mode

Single-destination mode is selected by choosing "single destination" from the *destination mode drop-down box* (see the screenshot below). Single destination mode allows you to specify one particular DS to which the *VSP* will be establishing its outgoing connections (when allowed by the routing mode and according to the connection mode option).



The destination DS can be specified either by its IP-address, or its MAC-address:

- When *Enable MAC-->IP mapping optionbox* is not checked the DS is specified by its IP-address.
- When *Enable MAC-->IP mapping optionbox* is checked the DS is specified by its MAC-address. This option exists to make the VSP-to-DS communications independent of the IP-address of the DS. If the DS is running with DHCP enabled (**DHCP (DH) setting** is 1 (enabled)) the IP-address may eventually change but the MAC-address will remain the same.

With mapping enabled, the *VSP* "discovers" current IP-address of the DS each time it needs to communicate with this DS. The process is illustrated by the following log in the Port Monitor:

```
12/17/03 17:11:00 - COM3 (INFO): Port opened
12/17/03 17:11:06 - COM3 (INFO): MAC --> IP mapping for MAC 0.2.3.4.61.189...ok, mapped to 192.168.100.92
12/17/03 17:11:06 - COM3 (INFO): Established TCP connection with node 192.168.100.92:1001
12/17/03 17:11:29 - COM3 (INFO): TCP connection closed
12/17/03 17:11:29 - COM3 (INFO): Port closed
--- IP-address of the DS has changed ---
12/17/03 17:17:03 - COM3 (INFO): Port opened
12/17/03 17:17:05 - COM3 (INFO): MAC --> IP mapping for MAC 0.2.3.4.61.189...ok, mapped to 192.168.100.91
12/17/03 17:17:05 - COM3 (INFO): Established TCP connection with node 192.168.100.91:1001
```

In the above example the IP-address of the DS has changed between the "communications sessions" but the *VSP* was able to connect to the same DS by using the MAC-->IP mapping.

Because the *VSP* is using broadcast UDP communications to "find" the DS with matching MAC-address, the MAC-->IP mapping feature only works for local Device Servers, i.e. devices located on the same network segment* with the PC (running *VSP*).

Also specified in the destination section of the *VSP properties dialog* is the *port number* on the DS. This must be the same number as the one specified in the **Port Number (DP) setting** of the destination DS.

Instead of entering the destination data manually, you can press the *Select Device Server from the list... button* and choose the DS from the list. The *button* brings up the *dialog*, similar to the *main window* of the DS Manager. In fact, it is the *DS Manager*, with the following two differences:

- There is an additional *Select button*. To select a particular DS as a destination for the *VSP* single-click on the DS in the *device list* and press *Select*. Alternatively, you can double-click on the DS in the *device list*.
- There is no COM access mode available in the access mode drop-down box. This is because you are choosing a network destination for the *VSP* so the target DS has to be selected from the list of devices accessible through the network!

When you select the DS from the list the *VSP Manager* selects or deselects the MAC-->IP mapping automatically:

- Mapping is deselected if the DS is not local (no matter the DS is running with DHCP enabled or disabled).
- Mapping is deselected if the DS is local but is running with DHCP disabled.
- Mapping is selected if the DS is local and is running with DHCP enabled.

Additionally, when you select the DS from the list the destination port is set automatically to the one, defined by the **Port Number (DP) setting** of the selected DS.

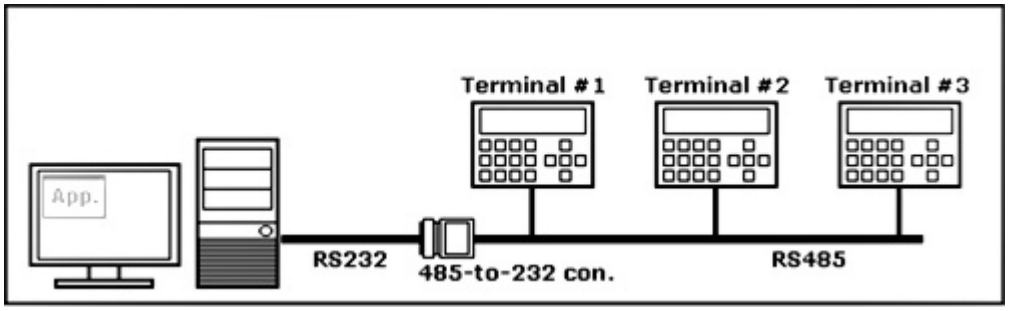
** The definition of the network segment implies that there are only network hubs (and no routers, bridges, firewalls, etc.) between the PC and all other devices on the segment. Broadcast messages cannot pass through the routers and cannot reach the Device Servers located behind those routers.*

Multi-destination Mode

Multi-destination mode allows you to communicate with *several* Device Servers (and the serial devices behind them) through a *single VSP*. To understand the usefulness of this feature consider the following example:

Example

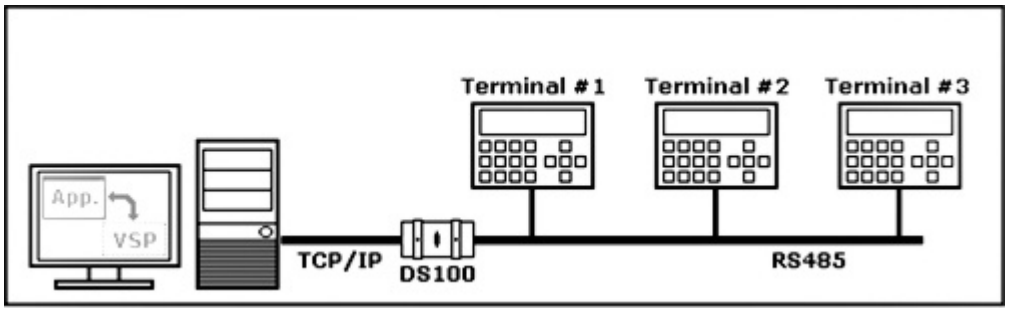
Let's assume there is a multi-drop (RS485) network with several serial data terminals connected to it (see diagram below). The RS485 "bus" is attached to the PC's COM port via a 485-to-232 converter. An application software on the PC (App.) can address each terminal individually by sending a formatted command that contains a "node address" of the terminal. All terminals on the RS485 bus receive the command but only the one with a matching node address will respond to the command. This is a very common way of multi-drop communications. The PC acts as a "master" and the terminals act as "slaves".



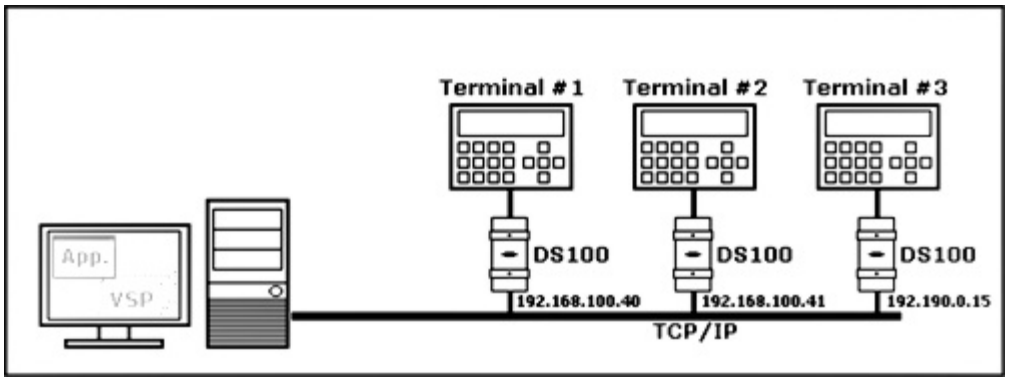
The formatted command typically has a start character (STX, ASCII code 2 in this example), followed by the node address (transmitted, for example, as two ASCII characters representing the number, i.e. "01", "02", etc.). The address characters are followed by the command contents and some sort of end character (for example, CR, ASCII code 13):

STX	Addr1	Addr2	Command contents	CR

Now let's suppose that you need to network-enable this system. The simplest way to do so would be to connect the RS485 bus to the DS, create a *VSP* on the PC and let the application software access the terminals through the *VSP* and the DS.



This will work, but only if the distance between the terminals is small. If the terminals are to be located far away from each other the RS485 network connecting them would have become very long thus defying the purpose of network-enabling this system! Much better solution would be to connect each terminal individually to its own DS (see figure below). This way there would be no RS485 network whatsoever.



The problem here is that each DS (and hence, each terminal) will now have its own IP-address and this means that a single *VSP* will have to switch between these IPs as needed (remember that on the original RS485 system the application software communicated with all terminals on the RS485 bus through a single COM port!).

The solution

The multi-destination mode of the *VSP* allows you to define several destination Device Servers and switch between these destinations basing on the outgoing data stream sent by the PC application. As explained above, the multi-drop RS485 system (almost) always has some form of node addressing and the *VSP* can be programmed to filter out this addressing information and automatically switch between the destinations.

The data in the outgoing data stream the *VSP* reacts to is divided into two portions:

- **Prefix string**- this is a fixed string that signals to the *VSP* that the actual node address will follow. Prefix can be any string of (almost) any length but it cannot be NULL (empty). Only one prefix string can be defined for each *VSP*.
- **Switch string**- this is the data that immediately follows the prefix. The switch string is different for each destination listed in the *destination table*. Switch strings can also be of (almost) any length but cannot be NULL (empty).

For the example used in this topic the prefix string is a single character with ASCII code 2 (STX) and the destination table looks like this:

Switch string	Destination IP-address
"01"	192.168.100.40
"02"	192.168.100.41
"03"	192.190.0.15

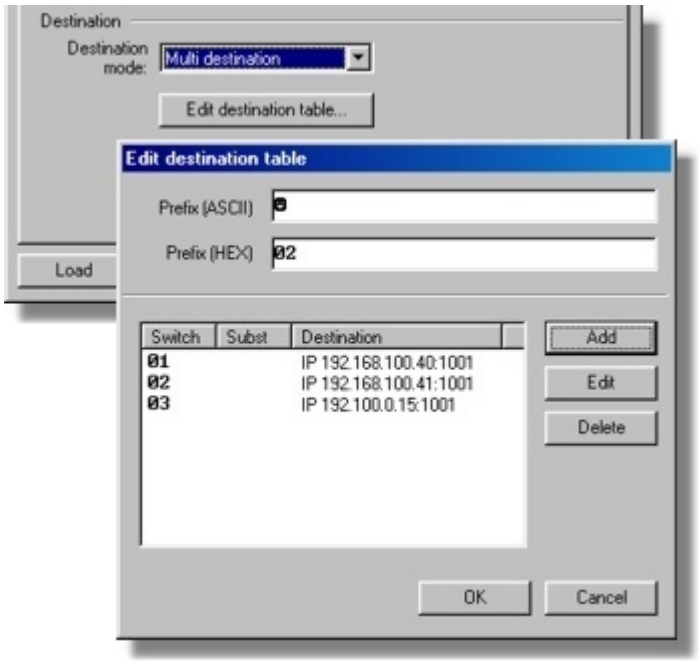
Once the *VSP* detects the prefix string in the data stream sent by the application software it starts checking if the subsequent data will match one of the defined switch strings. When the match is detected the *VSP* closes current data connection (if any) and switches to the new destination, corresponding to the detected switch string.

Here is how this is reflected in the log of the Port Monitor:

```
12/18/03 10:39:49 - COM3 (INFO): Port opened
12/18/03 10:40:04 - COM3 (INFO): Switching to 192.168.100.40:1001 ("01")...
12/18/03 10:40:04 - COM3 (INFO): Established TCP connection with node 192.168.100.40:1001
12/18/03 10:40:15 - COM3 (INFO): TCP connection closed
12/18/03 10:40:15 - COM3 (INFO): Switching to 192.168.100.41:1001 ("02")...
12/18/03 10:40:15 - COM3 (INFO): Established TCP connection with node 192.168.100.41:1001
```

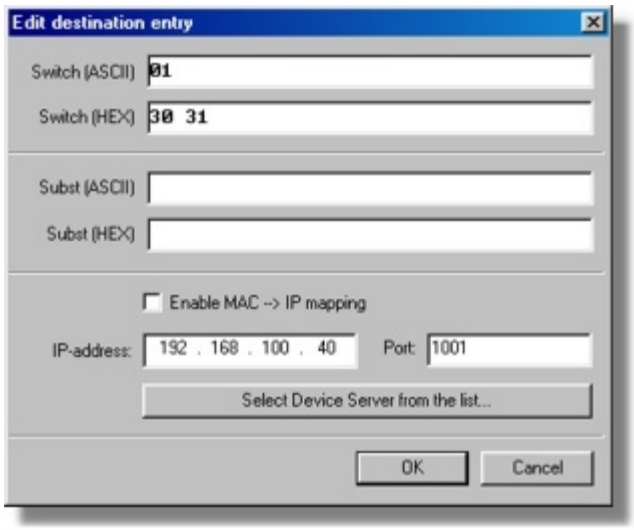
Setting up the multi-destination VSP

To switch into the multi-destination mode and create the list of destinations select "multi-destination" from the *destination mode drop-down box* and click *Edit destination table... button- Edit destination table dialog* will open (shown on the screenshot below). The multi-destination mode is only available when the routing mode is "client" and when the connection mode is "on data". "Server/client" routing mode is not allowed because the *VSP* is supposed to act as a master and establish all connections by itself. "Connect immediately" mode is not allowed because the *VSP* needs to receive a switch string from the application in order to know which destination to connect to.



The prefix string is entered into the *Prefix (ASCII)* or *Prefix (Hex) textboxes*. These textboxes are synchronized with each other. *Prefix (ASCII) texbox* records the characters as you type them. For example, if you press <A> key this will be interpreted as ASCII character 'A'. The *Prefix (Hex) texbox* will reflect the ASCII HEX code of this character- 41. Typing "02" (ASCII code of the STX character) into the *Prefix (Hex) textbox* will cause the image of this character (smiley face) to appear in the *Prefix (ASCII)* textbox. Special characters, such as STX, can also be entered in the Prefix *(ASCII) textbox* using the <CTRL> key (<CTRL>+<B> for STX).

Available destinations are entered into the *destination table*- there are *Add*, *Edit*, and *Delete buttons*. Pressing *Add* or *Edit button* opens the *Edit destination entry dialog* (see the screenshot below).



Switch (ASCII) and *Switch (Hex) textboxes* are used to enter the switch string for a particular destination.

Substs (ASCII) and *Subst (Hex) textboxes* provide additional functionality that haven't been mentioned yet. If the data is entered into those fields then the *VSP*

will automatically substitute the switch string encountered in the data stream sent by the application with this substitution string.

For example, if the switch string is "01" and the substitution string is "00" then the following will happen (STX character is represented as ☺):

Application-->VSP: **☺01ABC**
VSP-->DS: **☺00ABC**

The destination portion of the *Edit destination entry dialog* is similar to that displayed in the *VSP properties dialog* in the single-destination mode. The destination DS can be defined by its IP-address or by its MAC-address, depending on whether the *MAC-->IP mapping option* is enabled or not. Destination port must match the one specified by the **Port Number (DP) setting** of the destination DS. Instead of typing in the data manually, you can press the *Select Device Server from the list... button* and choose the DS from the list.

How multi-destination errors are handled

- If, after the *VSP* is opened, the application starts sending "random" data without sending a prefix and a switch string first, or if the data after the prefix does not match any switch string, the *VSP* will be discarding this data until a valid prefix and switch are detected.
- If, after a valid prefix and switch have already been detected and the *VSP* has established a connection with a certain destination DS, the *VSP* receives the prefix string that is not followed by any valid switch string, then the *VSP* will keep sending the data across the existing data connection.

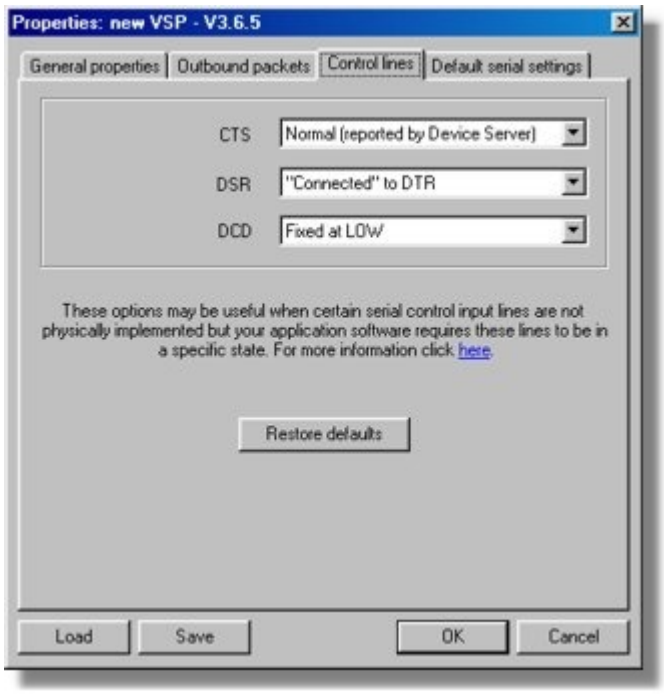
Specify Destination By

The **Specify by** drop-dox box allows you to select one of three methods to specify the VSP's destination:

- **IP-address:** Select the destination of the VSP by specifying its IP address. When using this option, the IP address should be static and not be dynamically assigned by DHCP.
- **Host name:** Specify the destination by DNS host name. The destination should be registered in DNS for this to work. Registration in DNS can be performed manually (using your DNS server administration interface) or using an interface between DHCP and DNS.
- **MAC-address:** Select the destination by specifying its MAC address. This is useful for destinations which are located in the current (local) network segment, whose IP address may change (i.e, when the IP address is dynamically assigned by DHCP).

Control lines Tab

Control lines tab allows you to define the "operating mode" for CTS, DSR, and DCD "inputs" of the *VSP*. CTS, DSR, and DCD are actually real physical input lines found on serial ports/cables (they exist alongside RTS and DTR outputs). It is a standard COM port functionality to be able to report the status of CTS, DSR, and DCD inputs to the application software that is working with this COM port.



The following options are available for each input line:

- **Normal (reported by Device Server).** In this mode the application software working with the *VSP* is updated (or queries) the status of actual physical lines on the serial port of the DS*.
- **Fixed at HIGH.** When this option is selected for a particular line this line appears to always be at HIGH ("enabled"), no matter what actual state of the corresponding physical line on the Device Server's serial port is.
- **Fixed at LOW.** When this option is selected for a particular line this line appears to always be at LOW ("disabled"), no matter what actual state of the corresponding physical line on the Device Server's serial port is.
- **Connected to RTS (DTR).** When this option is selected for CTS input it appears to be connected to the RTS output. When this option is selected for DSR input it appears to be connected to the DTR output. To the application software, this really looks like there is a wire interconnecting CTS and RTS (or DTR and DSR) lines. DCD line doesn't have a "pair" so this option is not available for DCD.

Fixed at HIGH, *fixed at LOW*, and *connected to RTS (DTR)* options are handy when certain serial control input lines are not physically implemented but the application software requires these lines to be in a specific state. For example, DTR and DSR lines are not implemented on the DS100R Device Server, yet some application software might wait for the DSR input to become HIGH as an indication that the serial device is ready to accept the data. To make sure that the software receives the state it is looking for the DSR line can be set to *fixed at HIGH*.

** This functionality depends on so-called notifications that are automatically generated by the DS and sent to the VSP and also on-the-fly commands (network-side parameters) generated by the VSP.*

Default serial settings Tab

Default serial settings tab allows you to define a set of communications parameters associated with this VSP. This *tab* exists purely for compatibility with regular COM ports. Under *Windows*, each COM port has a set of default parameters associated

with it, and so does *VSP*. These parameters *do not affect* operation of the *VSP* in any way.

Use WinSock for transport

Use WinSock for transport checkbox enables or disables working via WinSock.

On some PCs, a program called WinSock Proxy (WSP) Client is installed. This client modifies the way Windows handles IP routing.

When working with such a PC with the VSP in default mode (with the **Use WinSock for transport checkbox** unchecked), the VSPD will not be able to connect to remote IP addresses (addresses beyond the scope of the current subnet), but will be able to connect to local addresses.

When something like this happens, you should mark this checkbox and thus instruct the VSPD to use WinSock for transport, rather than the default mechanism (called TDI, Transport Device Interface).

Unsupported Features and Limitations

This topic describes existing limitations of the *VSP*.

- **Password protection for on-the-fly commands.** On the DS side this feature is enabled through the **On-the-fly Password (OP) setting**. The *VSP* does not support this yet so the password protection for on-the-fly commands must be disabled on the DS.
- **Password protection for data connections.** On the DS side this feature is

enabled through the **Data Login (DL) setting**. The *VSP* doesn't support data logins yet and so the password protection for data connections must be disabled on the DS.

Warnings and Messages

This reference section contains additional information on the warning and error messages displayed by the *VSP Manager*.

Listening Port is In Use

Description

Message text: Listening port associated with this *VSP* is already in use, possibly by another *VSP*. Set another listening port number and try again

---

Details

Listening port is the port number the *VSP* is accepting incoming connections on*. This message appears when the *VSP* attempts to open this port (which usually happens when the *VSP* is opened) and the port is already in use by another application (possibly, another *VSP*).

To solve this problem select another number for the listening port and try again.

** When incoming connections are allowed by the routing mode.*

No Start Characters defined

Description

Message text: You haven't defined any start-characters. Close still?

Details

This message appears when you are closing the *start/stop characters dialog* and you haven't defined any start characters while the *start on any character option* is disabled. This means that there are no conditions whatsoever that will open the data block and no data will ever be accepted from the application.

Port Substitution Required

Description

Message text: You have chosen the name that is currently assigned to a standard COM port. The COM driver for this port will be substituted with the *VSPD*. Are you sure you want to continue?

Details

This message appears when you select the port name for the *VSP* that is currently in use by the real COM port. Typically, these are "COM1" and "COM2" port names.

For more information on the subject see VSP name selection.

VSP Is in Use

Description

Message text: This *VSP* is currently opened by another application. Port properties cannot be altered at this time

This *VSP* is currently opened by another application and cannot be deleted

---

Details

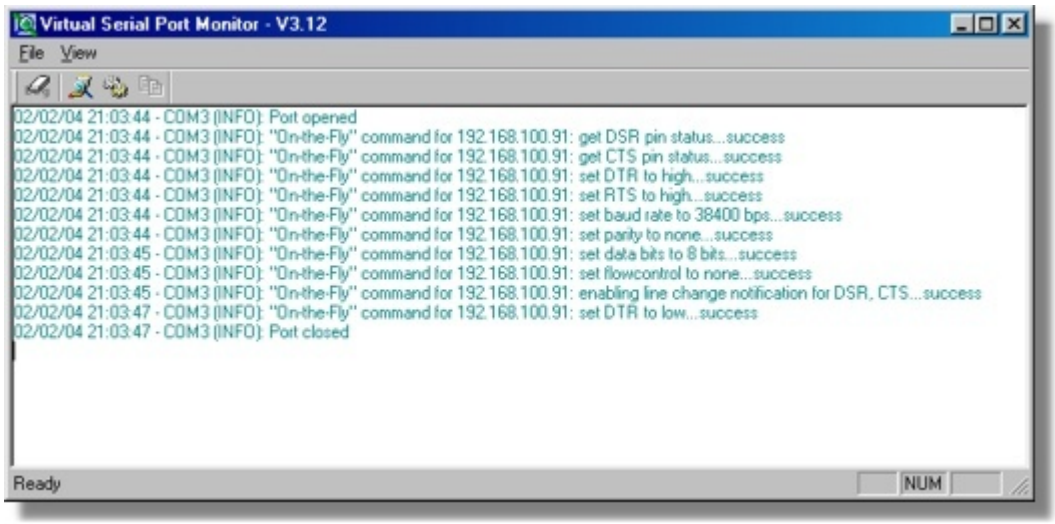
You can only edit *VSP* properties or delete a *VSP* when it is not currently opened (i.e. not in use by another application).

Port Monitor

Port Monitor is a parts of the Device Server Toolkit.

The *Port Monitor* is used to log the activity of Virtual Serial Ports (VSPs) installed on your *Windows* system.
When the *Port Monitor* is running its icon appears in the system tray, as shown on the screenshot below.

Double-clicking on the *Port Monitor's* icon in the system tray brings up the *main window*:



The *main window* displays the log of registered events. The simplest way to observe the work of the *Port Monitor* is to open the *VSP* from some application program and check what kind of data appears in the *Monitor* (sample output is shown on the screenshot above).

You can erase all data in the *Port Monitor's window* by choosing *File--> Clear log* from the *main menu* or clicking *clear monitor log* button (shown below).

You can also copy the log data into the clipboard by first selecting the desired part of the data in the *Port Monitor's window* and then choosing *File--> Copy* from the *main menu* or clicking *copy button* (shown below).

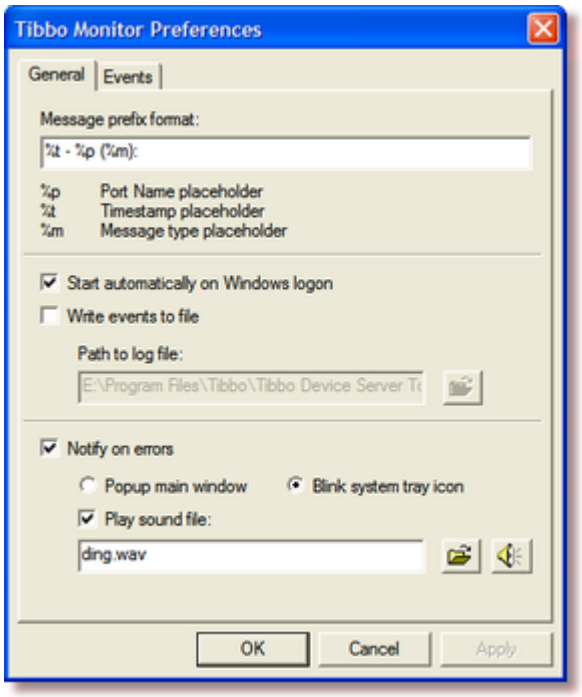
All events logged by the *Monitor* can be divided into informational, warning, and error events. *Preferences dialog* (general and event *tabs*) defines which events are logged, what format is used to displays those events, and what action the *Monitor* will take when an error event is encountered.

The *Monitor* also features a dump function which can be used to track the data passing through a particular *VSP*.

Preferences Dialog (General Tab)

General tab of the *preferences dialog* (shown on the screenshot below) offers a number of options that define how the *Port Monitor* logs events. To display the tab choose *File-->Preferences* from the *main menu* of the *Port Monitor* or click *Preferences button* (shown below).

 *Preferences button*



- **Event prefix format** defines how each event in the log will look like. There are three fields that can be added to each event line:
  - **Port name placeholder (%p).** Adding **%p** to the format string will make the *Monitor* print the *VSP* name (i.e. "COM3", "COM4") for which the event was generated.
  - **Timestamp placeholder (%t).** Adding **%t** to the format string will make the *Monitor* print current date and time stamp for each event line (i.e. "12/30/03 14:57:13").
  - **Message type placeholder (%m).** Adding **%m** to the format string will make the *Monitor* print the type of event (i.e. "INFO", "WARNING", "ERROR")

for each event line.

You can combine the placeholders, for example: "*%t- %p (%m):*". This will make the *Wizard* print all events like this:

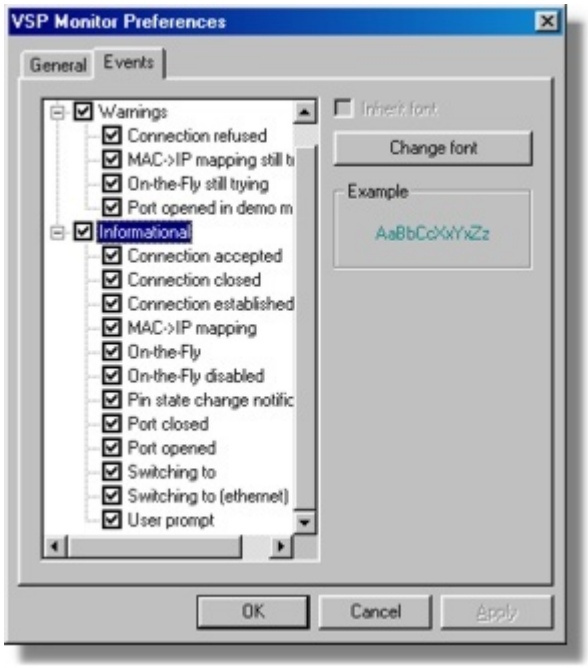
```
12/30/03 15:06:51 - COM4 (INFO): Port opened
12/30/03 15:06:52 - COM3 (INFO): Port closed
```

- **Start automatically on Windows logon option**, which is enabled by default, starts the monitor application when Windows boots.
- **Write events to file option**, when enabled, allows you to choose a file to which the *Monitor* will save all logged events. The *Monitor* will create a plain text file that can be viewed using any text editor.
- **Notify on error option**, when enabled, allows you to choose what the *Monitor* should do if any error events are logged. You have two choices:
  - **Popup main window.** The *main window* of the *Port Monitor* will be displayed automatically whenever an error event is registered.
  - **Blink system tray icon.** The *main window* will not be opened but the *Monitor's* icon in the *system tray* will have a blinking exclamation point.
  - Additionally, you can enable **play sound file option** and choice the sound that will be generated each time an error event is logged.

Preferences Dialog (Event Tab)

Events tab of the *preferences dialog* (shown on the screenshot below) allows you to enable/disable logging of and set the font for individual events and event groups. To display the tab choose *File-->Preferences* from the *main menu* of the *Port Monitor* (or click *Preferences button*), then click *events tab*.

 Preferences button



When the *Monitor* is installed all events are enabled and three different font types are selected for the informational, warning, and error events:

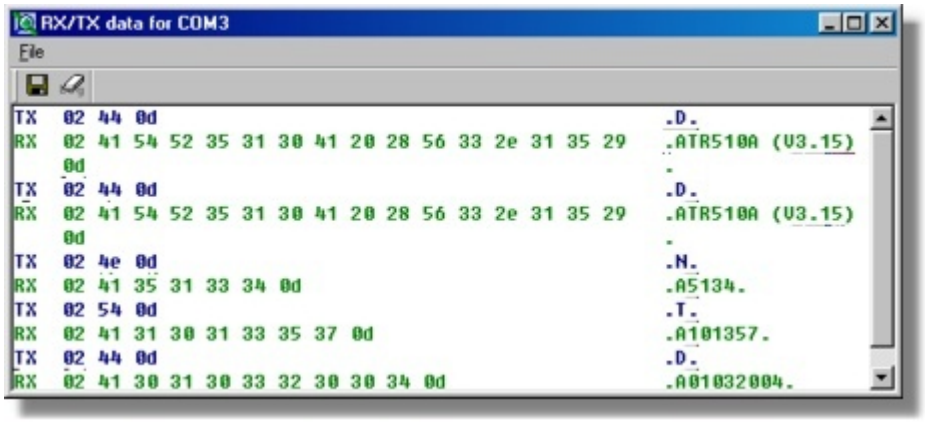
12/30/03 15:44:45 - COM3 (INFO): Port opened **Y---- info event**
12/30/03 15:44:48 - COM3 (WARNING): "On-the-Fly" command for 192.168.100.95: get DSR pin status...timed out, still trying... **Y---- warning event**
12/30/03 15:44:54 - COM3 (ERROR): "On-the-Fly" command for 192.168.100.95: get DSR pin status...timed out Y---- error event

To enable/disable entire event group click on the option box next to the group name. To change the font for the entire group select the group name in the list and press *Change font button*.

To enable/disable individual event "expand" the event group (by clicking on the "+" sign or double-clicking on the group name), then click on the option box next to the event name. To change the font for an individual event select this event in the list, deselect the *inherit font option*, then press *Change font button*.

Data Dump Feature

The *Port Monitor* can be used to capture the "serial data" passing through the *VSP*. The data is displayed in the *Hex dump window*, both in the HEX and ASCII format, as shown on the screenshot below:



To see the hex dump, click *View--> RX/TX Data* from the *main menu* of the *Port Monitor*- the *choose VSP to monitor dialog* will open. Select *desired VSP* and click OK.

The only two functions available in the *Hex dump window* are *save* (to save all captured data into a data file) and *clear* (to erase all data).

The Data Dump feature works using the Tibbo Service, which is an auxiliary service providing several functions for the VSP and VSP Monitor. If this service isn't running, the Data Dump feature will not work.

Connection Wizard

Connection Wizard is a part of the Device Server Toolkit.

Connection Wizard assists you in setting up typical data links involving Tibbo Device Servers and *Virtual Serial Ports (VSPs)*.

Shown below is the *welcome screen* of the *Connection Wizard*.

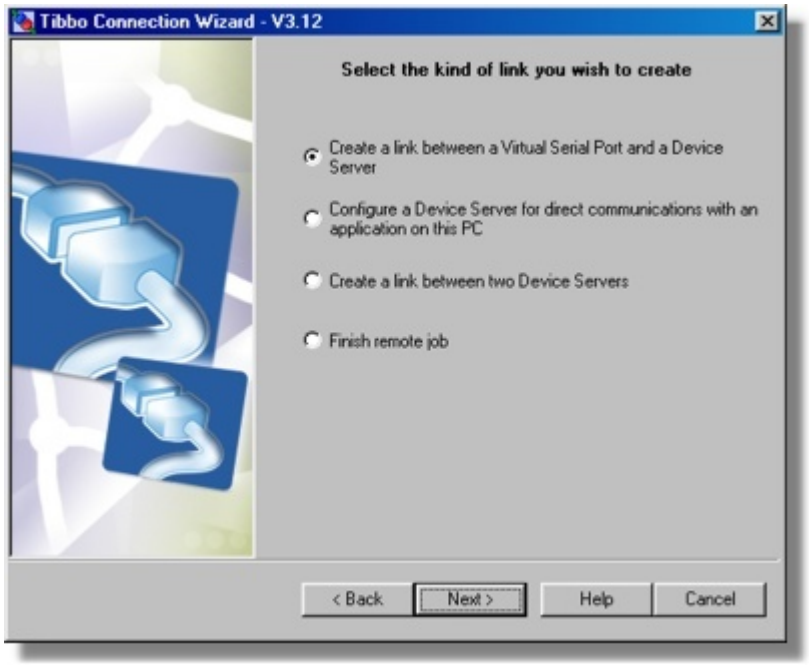
Connection Wizard offers a selection of jobs that help you setup several kinds of typical data links involving Tibbo Device Servers and *Virtual Serial Ports* (*VSPs*). For example, a VSP-to-DS job helps you create a *VSP* on the PC, and then configure this *VSP* and the Device Server (DS) of your choice to work with each other.

Creating a VSP-to-DS link through the *Wizard* produces the same effect as when you manually set up the *VSP* using the VSP Manager and then configure the DS with the DS Manager. The difference is that after you have answered several simple questions the *Wizard* does all necessary adjustments automatically. As a result you save tremendously in terms of time and debugging effort spent on setting up this data link. After you have familiarized yourself with the *Wizard* you can expect to complete typical data links in under 1 minute! Even when the link you need to create is not exactly the one this *Wizard* can help you with, you still can run the closest *Wizard* job, then make remaining setup changes manually.

This *Manual* follows the actual "flow" of *Wizard steps*. Pressing *Next* in the *welcome screen* of the *Wizard* moves you to the next step, on which you choose the kind of data link you want to create. Likewise, the next topic of this *Manual* (choosing the Wizard job) details this next step.

Choosing the Wizard Job

On this step you choose the kind of job you want the *Connection Wizard* to perform.



Four choices are currently available:

- **Create a link between a Virtual Serial Port and a Device Server**
- **Configure a Device Server for direct communications with an application on this PC**
- **Create a link between two Device Servers**
- **Finish remote job**

VSP-to-DS Link

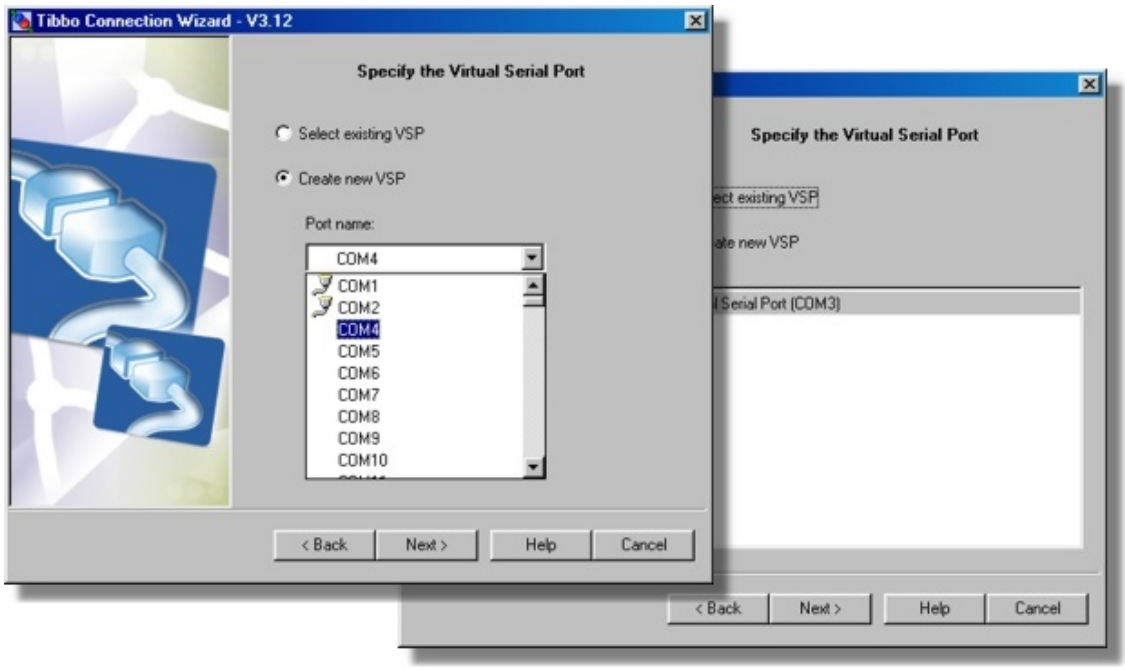
This job is selected by choosing *create a link between a Virtual Serial Port and a Device Server* from the list of available jobs.

When performing this job the *Connection Wizard* creates a *VSP* (on the PC the *Wizard* is running on) and then configures this *VSP* and the DS of your choice to work with each other. Use this job if you want to make your "serial" PC software* communicate with the serial device through the *VSP* and the DS. This way your PC software and the serial device can communicate with each other as if they were still interconnected by a normal serial cable. This is a fast and economical way of network-enabling a legacy serial system that you are unwilling or not able to modify- both the PC software and the serial device will require no modification whatsoever to work through the network.

** I.e. the software that can only communicate through COM ports. If your software supports communications through TCP/IP networks use direct-to-DS connection.*

VSP Name Selection

On this step you select the name of the *VSP* that will be configured on the PC side of the VSP-to-DS connection. *VSP* names look like the numbers of regular COMs, i.e. COM1, COM2, COM3.



- Choosing *select existing VSP option* displays the list of *VSPs* that are already found on your system (i.e. the ones that have been created before).
- Choosing *create new VSP option* displays the list of unused *VSP* names (shown above). The number of ports you can have in *Windows* is virtually unlimited- the *Connection Wizard* lets you create any port in the range between COM1 and COM255*.

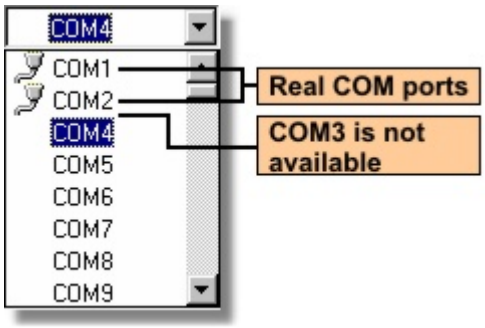
There are no rules on what *VSP* name to choose but some considerations are provided here.

** We could extend the number even further but feel that the current range is sufficient for all practical purposes.*

Choosing the VSP Name

This topic provides additional information on choosing a proper *VSP* name.

There are no rules on what *VSP* name to choose, just make sure that you pick the name that can be selected in the application software you plan to use with this *VSP* . Most programs provide a limited selection of ports (typically, up to COM2 or COM4). Therefore, choosing "COM100" won't be suitable as you will not be able to select this *VSP* in such a program.



On the screenshot above COM3 is not listed- this is because the *VSP* with this name already exists. Notice, however, that ports COM1 and COM2 are not excluded from the list and are marked with icons identifying them as "real" COMs*. This is because even though these port numbers are "occupied" the *Connection Wizard* can still "grab" them. For example, if you select COM1 the *Wizard* will automatically reassign COM1 to be a *VSP* (from this moment on COM1 will cease working as a normal COM port and will start working as a *VSP*). Substitution may be necessary when you are dealing with an old application software that only provides a choice of COM1 and COM2 which are usually occupied by real COMs.

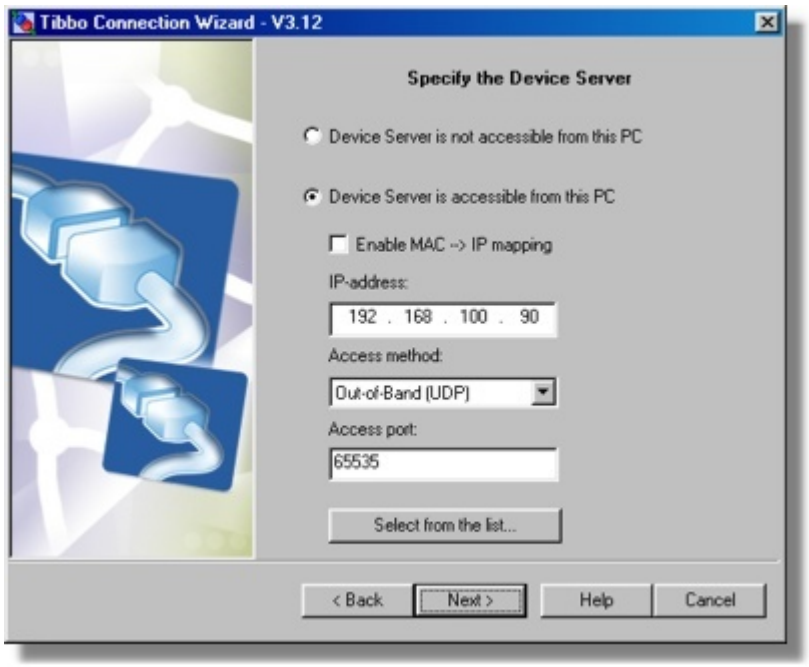
When you delete the *VSP* (this can be done through the VSP Manager) that substituted a standard COM port the *VSP Manager* will attempt to restore the original driver (make this port become a normal COM again). This mostly works but on some systems you may encounter problems (especially under *Windows ME*). We have conducted an extensive "research" into the port substitution, only to conclude that it just won't work reliably on all systems!

We recommend that you do not use port substitution unless absolutely necessary!

** This is system-dependent. Some modern PCs don't have any real COMs.*

Target DS

On this step you select the DS that the *VSP* (selected on the previous step) will be communicating with.



In most cases you complete this step as follows:

- Select *Device Server is accessible from this PC*.
- Click *Select DS button* and choose the DS from the list. The *button* brings up the *dialog*, similar to the *main window* of the DS Manager. In fact, it is the *DS Manager*, with the following two differences:
  - There is an additional *Select button*. To select a particular DS single-click on the DS in the *device list* (in the auto-discoveryor address book access mode) and press *Select*. Alternatively, you can double-click on the DS in the *device list*.
  - The COM access mode is not available when the *DS Manager* is called from within *Wizard*. This is because you are choosing a network destination for the *VSP* so the DS has to be selected from the list of devices, accessible through the network!
- Click *Next*.

This step provides a choice of several access options for the target DS- read additional info on accessing the DS for more information.

When you click *Next* the *Wizard* attempts to contact the target DS and verifies if this DS is local or remote. The DS of your choice must be available on the network, or the *Wizard* won't allow you to continue. Moreover, the IP-address of this DS must be reachable*. If everything is OK the *Wizard* reads out all current setting values of this DS.

What if the DS is not visible from the PC

If the PC and the DS are located on different network segments it may well be that the DS is not "visible" from this PC but the PC is visible from the DS. This is normal as many networks are not symmetrical. For example, if the PC has a public (real) IP-address and the DS is located on a firewalled network segment then the DS can connect to the PC (*VSP*), but the PC (*VSP*) cannot connect to the DS.

To specify that the DS is not visible from this PC select the *Device Server is not accessible from this PC* option and click *Next*.

There are two implications of this choice:

- In this VSP-to-DS connection it is the DS that will always be connecting to the *VSP*, since *VSP* cannot connect to the DS.
- The *Wizard* won't be able to setup the DS through the network (this is because it cannot "see" the DS). You will have a chance to setup this inaccessible DS through its serial port or through a temporary network connection. You will be given a choice to do this within current Wizard job or generate a *configuration script* and finish the setup later, using the finish remote job feature of the *Connection Wizard*.

** The DS Manager and the Connection Wizard have the ability to "see" local Device Servers even if the IP-address of these devices is unreachable (this is done by using broadcast communications- read about this here). The reason why the Wizard demands that the Device Server has a proper IP-address is because the VSP will have to establish a real data connection with this DS and that requires a reachable IP-address.*

Additional Info on Accessing the DS

This topic provides additional information on the access options available for the target DS.

The target DS can be selected from the list (using *Select DS button*) or its address and access parameters can be input manually. The first option is usually more convenient and requires less manual input and decisions on your behalf.

Specifying the target address

The target DS can be specified either by its IP-address or MAC-address:

- When *Enable MAC-->IP mapping option* is not checked the DS is specified by its IP-address.
- When *Enable MAC-->IP mapping option* is checked the DS is specified by its MAC-address. If the DS is running with DHCP enabled its IP-address may eventually change but the MAC-address will remain the same. This option exists to make the communications with this DS "IP-address independent": each time the *VSP* needs to connect to the DS it will find out the current IP-address of the DS with the target MAC-address, and only then access this IP-address. The MAC-->IP mapping feature only works for local Device Servers. If the target DS is remote it can only be specified by its IP-address.

When you choose the DS from the list (through the *Select DS* button) the *Wizard* selects or deselects the mapping option automatically:

- Mapping is deselected if the DS is not local (no matter the DS is running with DHCP enabled or disabled).
- Mapping is deselected if the DS is local but is running with DHCP disabled (**DHCP (DH) setting** is 0).
- Mapping is selected if the DS is local and is running with DHCP enabled (**DHCP (DH) setting** is 1).

Specifying the access method and port

Besides the IP (or MAC) you need to specify the *access method* and *access port* for this DS. Access method is the way the *Wizard* will access the DS in order to program it. Access port is the port to which the *Wizard* will send its programming

commands. These options are not directly related to how the *VSP* will be accessing the DS, they just specify how the *Wizard* is going to access the DS.

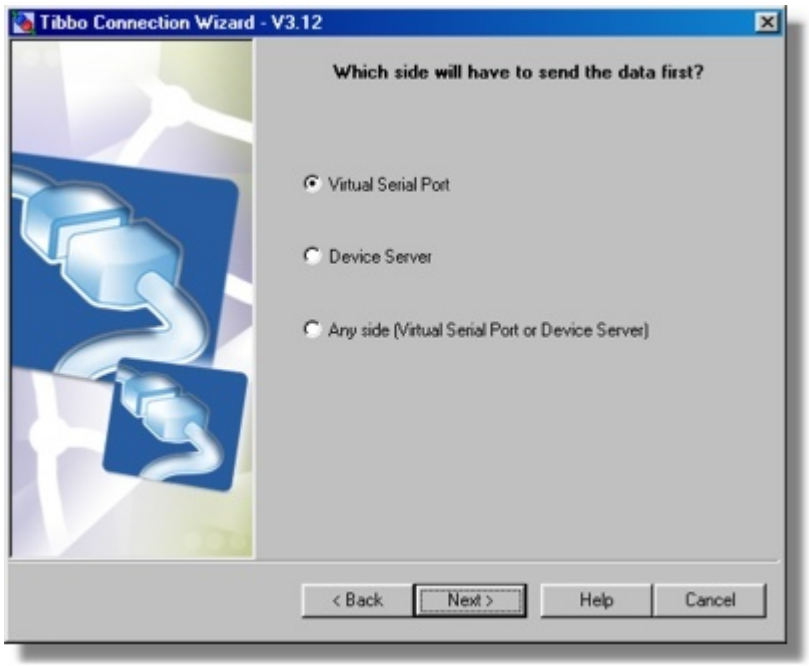
There are two access methods:

- **When out-of-band (UDP) access method** is selected the *Wizard* communicates with the DS using command UDP datagrams sent to the command port 65535 of the DS. Out-of-band commands offer a primary way of network programming that requires no preliminary setup on the DS side. This access method is suitable for all local Device Servers. It will also work with remote Device Servers unless the UDP traffic is banned by the network routers (firewalls, etc).
- **When inband (TCP) access method** is selected the *Wizard* communicates with the DS by sending commands through the TCP connection established to the data port of the DS (this is the number defined by the **Port Number (PN) setting** of the DS). Inband (TCP) commands provide a secondary method of network programming that can be used in situations when out-of-band UDP access is impossible (for remote Device Servers). The downside is that a certain pre-programming must be done on the DS before it can be accessed using inband access method- read preparing the DS for inband access for step-by-step instructions.

When you select the DS from the address book (through the *Select DS* button) the *Wizard* copies the access method and access port from the address book entry.

Initiator of the Data Exchange

On this step you select which side of the VSP-to-DS connection will be the first to send the data.



Notice, that asking "who will be the first to send the data?" is not the same as asking "which side will be establishing the data connection to the other side?"* When making your choice imagine that there is no network at all and the serial device is attached to the COM port of the PC directly. Now, which side sends the

data first?

There are three choices:

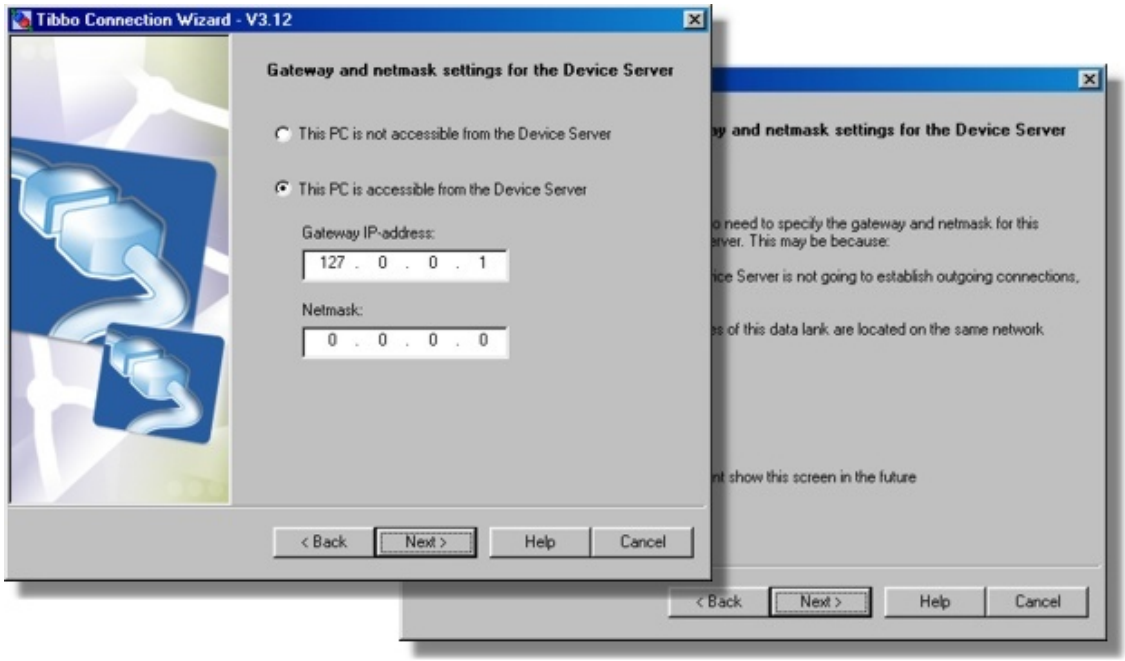
- **Virtual Serial Port.** Select this option if it is the application software that will be sending the first data. This is typical for the kind of serial devices that we, at Tibbo, came to call "slave terminals". Such terminals communicate with the PC software using some sort of command-reply communication protocol and it is the PC software that usually plays the role of a "master": the terminal won't send any data unless the PC sends a command first (and this is the "first data"!).
- **Device Server.** Select this option if it is the serial device that will be sending the first data. This is typical for the kind of serial devices that we call "scanners" (readers)**. This class of serial devices differs from "slave terminals" in that they send out the data spontaneously, without any prior prompt from the PC software.
- **Any side.** In some systems the data may first come from any side. For example, there are some "active terminals" out there that can send out the data spontaneously and at the same time respond to the commands from PC. Also choose this option if you are not sure which side of your system sends the data first.

** Although these two questions are connected- see how the Wizard decides which side open the connection.*

*** Meaning barcode scanners or magnetic card readers.*

Netmask & Gateway for the DS

This step comes in two options: you are either requested to input the gateway and netmask information that the DS needs to connect to this PC (left screenshot) or you are shown a screen informing you that the entry of this data is not necessary (right screenshot).



The gateway and netmask need only be set when two conditions are observed:

- The DS may need to (or must) establish data connections to the PC (this is determined by the *Wizard*).
- The DS and the PC are located on different network segments (there is at least one router between the PC and the DS).

Setting gateway and netmask information for the DS that only accepts incoming connections is not necessary*. If the DS is local there are no routers between this DS and the PC, so the gateway and netmask are not necessary even if the DS has to connect to the PC.

How to set the gateway and netmask

There is no straight way of finding out what data to enter. You need to obtain this information from somebody with a knowledge of your network's topology. Here is one hint, though.

If the IP-address of the target DS is configured through DHCP (**DHCP (DH) setting** is 1 (enabled)), then the DHCP server might have configured this Device Server's gateway and netmask as well. When you reach this step of the *Wizard* the *Gateway IP-address* and the *Netmask fields* are filled with the data from the **Gateway IP-address (GI)** and **Netmask (NM)**settings of the target DS**. This data may already be correct!

What if the PC is not visible from the DS

When the PC and the DS are located on different network segments it may well be that the PC is not "visible" from this DS. This is normal as many networks are not symmetrical. For example, if the DS has a public (real) IP-address and the PC is located on a firewalled network segment then the PC (*VSP*) can connect to the DS, but the DS cannot connect to the PC (*VSP*).

To provide for this situation the option box at the top of this step's screen allows you to specify that *this PC is not accessible from the Device Server*. Choose the option if this is so and if the option is available.

The option to specify that the PC is not accessible from the DS is disabled (not allowed to be chosen) when you have previously specified (at the target DS step) that the DS is not accessible from the PC. The reason is obvious: at least one side of the connection must be able to "see" the other side!

After you have completed this step the *Wizard* has enough information to decide which side of the VSP-to-DS link will be responsible for establishing the data connections.

** We found that some people hold a passionate belief that these parameters are necessary in any case. This is not true! If your network device doesn't have to establish outgoing connections you never have to bother about the gateway and netmask!*

*** Unless you have specified that "the DS is inaccessible from this PC" at the target DS step of the Wizard, in which case the Wizard cannot obtain this information.*

How the Wizard Decides Who Opens Connections

When you click *Next* on the netmask and gateway step the *Wizard* has enough information to decide which side of the VSP-to-DS connection will be responsible for establishing data connections with the other side.

In doing this, the *Wizard* decides on the future values of the **Routing Mode (RM)**

and **Connection Mode (CM)** settings of the DS, as well as the Routing Mode and Connection Modeproperties of the *VSP*.

In general, the *Wizard* tries to follow the natural flow of data between the application and the serial device. In the initiator of the data exchange step you have already specified which side sends the data first, so the *Wizard* will have this side open a data connection to the other side as soon as the first data needs to be sent from this side of the connection.

For example, if you are dealing with the "slave terminal" type of serial device, then the first data is always sent by the application (and the *VSP*). Therefore, the *Wizard* will set the Routing Mode property of the *VSP* to "client" and the **Routing Mode (RM) setting**of the DS to "slave".

Lines 1-3 of the following table illustrate what's just been said. Notice that the **Connection Mode (CM) setting** is always set to "on data". As the first data is received (by one side of the link) it triggers an attempt to open a data connection with the other side of the link:

	Which side can "see" the other side of the connection	**Which side sends the first data**	**Routing Mode on the *VSP***	**Connection Mode on the *VSP***	**Routing Mode on the DS**	**Connection Mode on the DS**
1	Both sides can see each other	Application	Client	On data	Server	---
2	Both sides can see each other	Serial device	Server	---	Client (server/client)	On data
3	Both sides can see each other	Any side	Server/client	On data	Server/client	On data
4	Only PC can see the DS	Application	Client	On data	Server	---
5	Only PC can see the DS	Serial device	Client	Immediately	Server	---
6	Only PC can see the DS	Any side	Client	Immediately	Server	---
7	Only DS can see the PC	Application	Server	---	Client	Immediately
8	Only DS can see the PC	Serial device	Server	---	Client	On data

9	Only DS can see the PC	Any side	Server	---	Client	Immediately

The situation becomes more complicated when only one side of the connection can "see" the other side. Consider the case in line 5 of the table. The serial device has to send the data first, but it cannot "see" the *VSP*. Therefore, even when the DS receives the first data into its serial port it will be unable to establish a connection to the *VSP* and send this data!

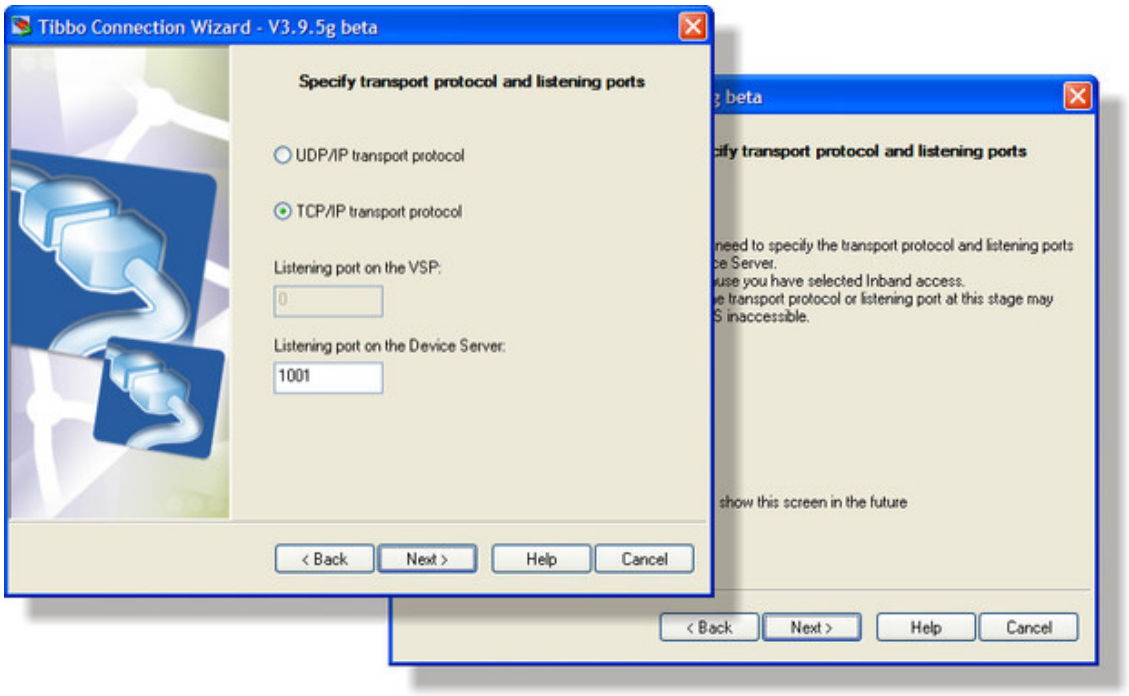
The way out is to have the *VSP* connect to the DS as soon as the *VSP* is opened (so, connection mode for the *VSP* is set to "immediately"). This way, by the time the DS needs to send the first data to the *VSP* the connection is already established. This kind of connections are called "reverse".

A special comment should be made on line 2 of the table, where the Routing Mode on the DS is designated to be either "client" or "server/client". The *Wizard* chooses "client" if DS programming is effected using out-of-band access method (you have selected this on the target DS step of the *Wizard*). If inband access method was selected the *Wizard* will set the mode to "server/client". This is because choosing "client" would cause it DS to reject all incoming connections in the future and this means that you wouldn't be able to program the DS using inband access in the future!

** This is the case when you have specified that "the Device Server is not accessible from this PC" on the target DS step, or that "the PC is not accessible from this Device Server" on the netmask and gateway for the DS step of the Wizard.*

Transport Protocol & Listening Ports

On this step you select the protocol that will be used for exchanging the data between the *VSP* and the DS. You also choose the listening port number on the side(s) of the link that will be receiving incoming connections from the other side.



In general, we recommend you to stick to the TCP/IP, unless you have a good reason (which is very rare!) to use UDP/IP protocol.

There is one case when the UDP/IP selection is not available and the TCP protocol is pre-selected for you- this is when you have specified an inband access method on the target DS step of the *Wizard*. Since inband access requires a TCP/IP data connection with the DS you must use TCP/IP transport protocol!

Under What Circumstances Transport Protocol & Port Selection Is Disabled

As noted above, when you use Inband access mode for the wizard, you must use TCP/IP as the transport protocol. Also, port selection will be disabled in this case. At this stage of the Connection Wizard you are already in communication with a DS. The communication is done via the data transport channel of the DS (as opposed to a separate command channel, like in Telnet or out-of-band mode).

The DS is usually far away (somewhere on a WAN), otherwise you would not use inband access to reach it. Thus, there are all sorts of firewalls and gateways between yourself and the DS. Firewalls only allow traffic on certain ports to go through, and UDP packets are often dropped on WANs.

If you change the protocol to UDP now, or change the listening port on the DS, you may render it completely inaccessible. The change *will* occur, because the Wizard is in communication with the DS (on the proper port which you configured in the beginning). But at the end of the Wizard run, you'll have an unpleasant surprise - the DS may suddenly disappear.

Thus, when selecting Inband Access mode in the beginning of the Wizard, you cannot later change the Transport Protocol or Listening Port.

Listening ports

This screen also provides an option of entering the port number on the listening side(s) of the VSP-to-DS link. By now the *Wizard* has already decided which side opens the connections. Connecting side needs to know the number of the listening port on the other side. For example, if it is the *VSP* that will always be connecting

to the DS, then you only have to specify the listening port on the DS side. Consequently, the *listening port on the DS textbox* will be enabled, and the *listening port on the PC textbox* will be disabled. If both sides will need to establish the connection, then you have to specify the listening ports on both sides too, and both textboxes will be active.

There is one case when the listening port on the DS side is fixed and pre-selected for you- this is when you have specified an inband access method for this DS. In this case you have already specified the listening port (as the access port) on the target DS stepof the *Wizard* (listening port and the access port are the same for inband mode).

Once the listening port on one side is known, the Wizard sets the destination port on the other side accordingly. If the *VSP* will need to connect to the DS the destination port on the *VSP* will be the same, as the value of the **Port Number (PN) setting** on the DS side. Likewise, if the DS will have to connect to the *VSP* the **Destination Port (DP) setting** on the DS will be the same as the local port number on the *VSP*.

How the listening port numbers are chosen

In most cases the choice of listening ports is automatic so when you get to this step suggested port numbers are already entered by the *Wizard*.

On the *VSP* side the number is calculated as 998+ the number of COM. For example, if you are dealing with COM3 then the *Wizard* will suggest the number of 998+3=1001*. Tying the listening port number to the *VSP* name provides a simple and effective way of making sure that all *VSP*s on your system use a different listening port (listening port cannot be shared between the *VSP*s).

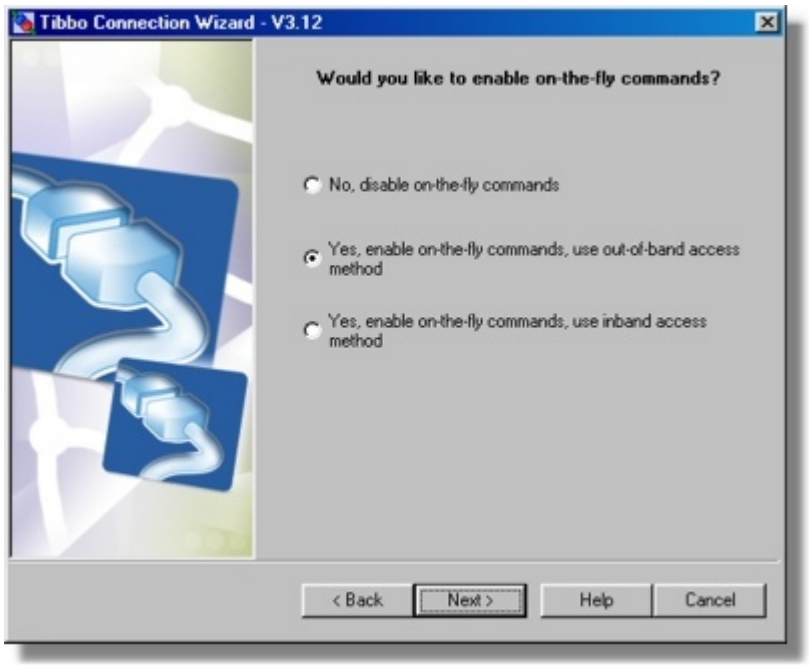
On the DS side you can choose any port of your liking, except the 65535 (65535 is a special command port). What the *Wizard* is showing you by default is the default value of the **Port Number (PN) setting** of the DS (1001).

The only reason to change suggested port numbers is if your network's firewall bans most of the traffic so only specific ports are opened for communications. In this case you may need to adjust the port numbers to the requirements of your network.

** There is no any special reason for this, we cannot even remember why we start from 998.*

On-the-fly Commands

On this step you decide whether the PC application will be controlling the setup of the serial port of the DS.



The VSP-to-DS connection can be arranged in such a way that the serial port of the DS is always setup as required by the "serial" application. For example, if the application that opens the *VSP* selects the baudrate of 9600 bps, then the baudrate of the DS serial port is immediately set to this baudrate as well.

This functionality is delivered through the so-called on-the-fly commands (or, more officially, network-side parameters). Just like programming commands, on-the-fly commands can be sent using out-of-band or inband access method, so you have a total of three choices:

- **No, disable on-the-fly commands**. On-the-fly commands are not sent at all, the DS is not aware of the serial port settings required by the application and is running with "permanent" serial parameters defined through the serial settings. For example, even if the application requires 9600 bps, but the **Baudrate (BR) setting** of the DS is programmed to 5 (38400 bps), the serial port of the DS will be running at 38400.
- **Yes, enable on-the-fly commands, use out-of-band access method.** On-the-fly commands will be sent as UDP datagrams to the command port 65535 of the DS. This access method is suitable for all local Device Servers. It will also work with remote Device Servers unless the UDP traffic is banned by the network routers (firewalls, etc). We suggest that you select this option whenever possible.
- **Yes, enable inband on-the-fly commands, use inband access method.** On-the-fly commands will be sent through the TCP connection established to the data port of the DS (this is the port number defined by the **Data Port Number (PN) setting**). This method can be used in situations when out-of-band on-the-fly commands cannot get through.

In certain cases the *Wizard* will automatically disallow the out-of-band option:

- If you have specified inband access on the target DS step of the *Wizard* then the *Wizard* will assume that this is because out-of-band access is impossible. Since the *VSP* will be running on the same PC as this *Wizard*, the same constraints (probably) apply to the *VSP* as well, so *Wizard* disallows out-of-band access.

- If the *Wizard* has decided that the *VSP* is to run in the server routing mode then the *VSP* cannot sent out-of-band commands too. This is because sending out-of-band commands would be like establishing outgoing UDP "connections" to the DS, and selected routing mode doesn't allow this. On the contrary, inband on-the-fly commands are OK- after the DS connects to the *VSP* the latter can send these commands across this existing data connection.

If you choose to disable on-the-fly commands the *Wizard* will select either "disabled" or "disabled (with FF escape)" on-the-fly mode on the *VSP* side depending on how the DS is being accessed for programming (you have selected this on the target DS step of the *Wizard*). See disabled (with FF escape) mode for details.

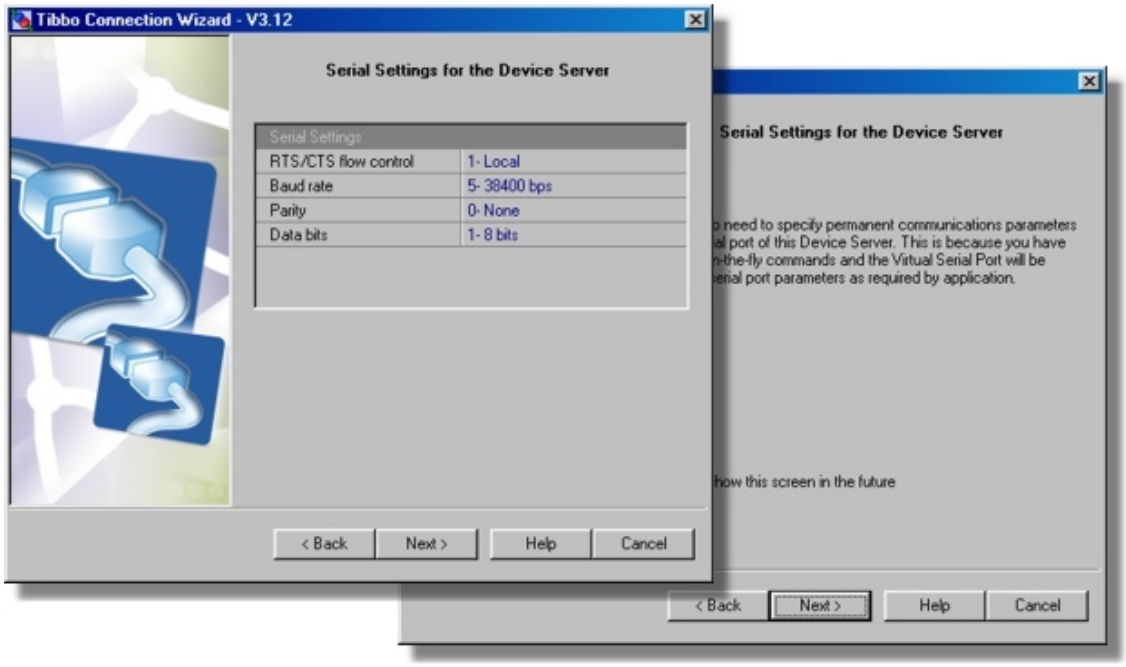
Enabling on-the-fly commands also has another effect: it allows the status of RTS, CTS, DTR, and DSR lines to be transmitted between the *VSP* and the DS.

More information regarding on-the-fly commands

- Serial port and serial communications (*Firmware Manual*)
- On-the-fly (network-side) parameters and instructions (*Firmware Manual*)
- On-the-fly commands (*VSPD and VSP Manager Manual*)

Parameters for the Serial Port of the DS

This step comes in two options: you are either requested to input communication parameters for the serial port of the DS (right screenshot) or you are shown a screen informing you that this information is not required (left screenshot).



Values entered on this screen are saved into the serial settings of the DS. Once set, they remain unchanged even after the DS is switched off. The serial port of the DS will always use these values unless temporary overriding parameters are received from the *VSP* (in the form of the on-the-fly commands- if you have enabled them on the previous step).

The screen with permanent serial port values is shown in two cases:

- You have disabled on-the-fly commands on the previous step of the *Wizard*. This means that the DS won't be aware of the serial parameters required by the application and will use the permanent values you are to set now.
- You have enabled on-the-fly commands, but the Wizard has decided earlier that the **Routing Mode (RM) setting** of the DS is to be either 2 (client) or 1 (server/client). This deserves an explanation- see below for details.

The empty screen is shown when on-the-fly commands are enabled and the **Routing Mode (RM) setting** of the DS is to be 0 (server). Follows is the explanation on why this is so.

When the "permanent" serial port parameters are relevant or irrelevant

When the DS is in the server **Routing Mode** its serial port is not opened until an incoming data connection is received from the network host (*VSP* in our case)*. When on-the-fly commands are enabled the *VSP* sends on-the-fly commands to the DS as soon as the *VSP* is opened**. Therefore, the serial port of the DS is configured with on-the-fly commands right when it is opened. This means, that the "permanent" values defined by the serial settings are never actually used.

It's a different story when the **Routing Mode** of the DS is server/client or client. In this case the serial port of the DS is opened right from the moment the DS is powered up*. This means, that the serial port is opened *before* the *VSP* is opened and the on-the-fly commands are sent to the DS. Therefore, there is a chance that the DS will receive some serial data before the serial port is configured by the *VSP*. This is why the parameters defined by the serial settings are not irrelevant***!

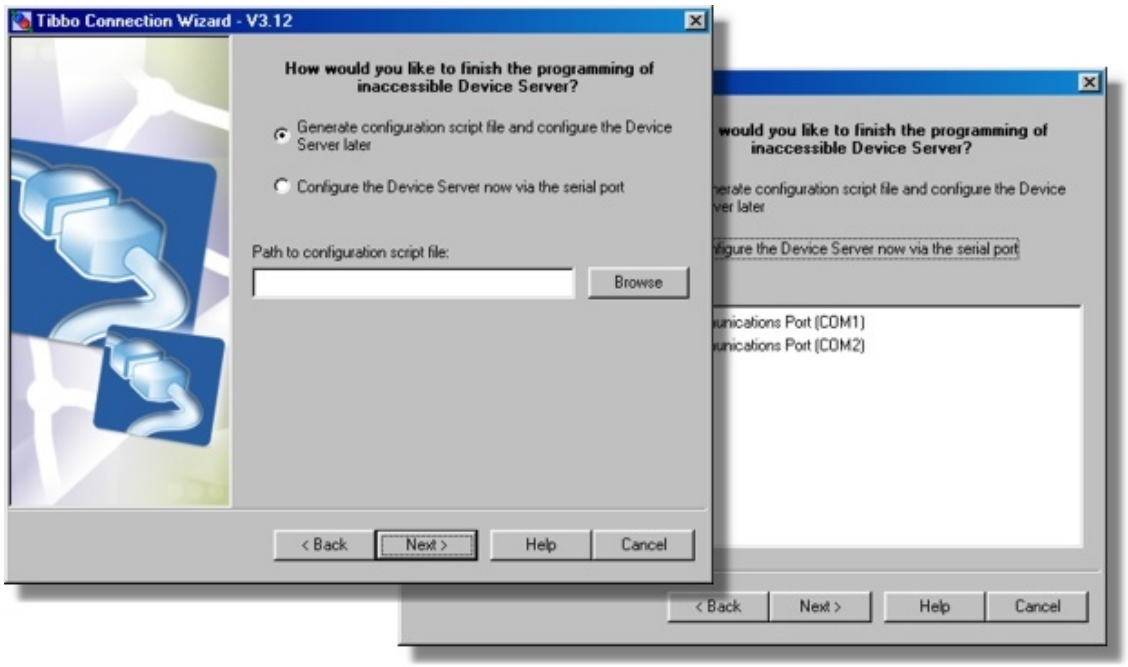
** This is described in the opened and closed states of the serial port topic.*

*** Unless the VSP is in the server routing mode but this cannot be so for the case in question: if the DS is in the server routing mode then the VSP has to be in the client or client/server mode!*

**** Of course, when the VSP is opened it will still send the on-the-fly commands and the data received before that can simply be ignored by the application. In this case you can still say that it doesn't matter what the permanent settings of the serial port are.*

Programming Inaccessible DS

If this screen is shown then you have specified on the target DS step step that the *Device Server is not accessible from this PC*. Now you have to decide how you will setup this DS.



There are two choices:

- **Generate configuration script file and configure the Device Server later**. The script file can later be fed into the *Connection Wizard*, possibly running on a *different* PC (see finishing remote job). Choose this option if you cannot physically bring the DS to this PC for programming. Input the path and filename for the script file and press *Next*.
- **Configure the Device Server now via the serial port**. Choose this option if you can temporary bring the DS to this PC for programming. Connect the serial port of the DS to the unused COM port of your PC (using WAS-1455 or similar cable), make sure the DS is powered up and press the setup button* (this will put the DS into the serial programming mode). Click *Next* to continue (the *Wizard* will read out current setting values of the DS).

Note:

 Status LEDs of the DS are playing a *serial programming mode pattern* (shown on the left) when the serial port of the DS is in the serial programming mode (click here to see all available patterns).

** On EM100, EM120, EM200, EM203(A)- pull the MD line low for at least 100ms.*

Application-to-DS Link

This job is selected by choosing *configure a Device Server for direct communications with an application on this PC* from the list of available jobs.

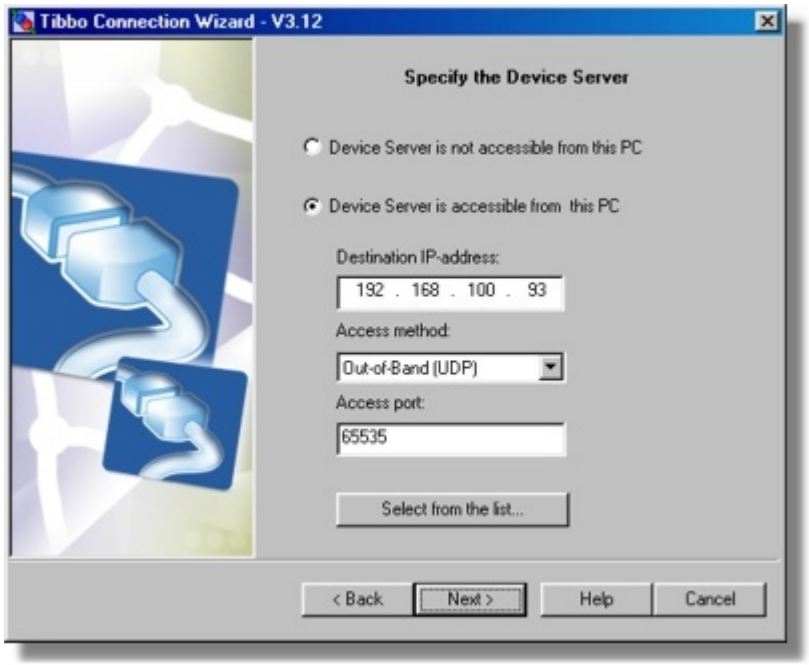
When performing this job the *Connection Wizard* configures the DS of your choice to work with the "network" software application* on this PC. Use this job if you want to make your PC application communicate directly with the DS and the serial device behind it.

When working on this job the *Wizard* can only setup the DS side of the link. It is your responsibility to setup the software as needed (expected by the *Wizard*).

** I.e. an application that is capable of communications through the TCP/IP networks. If your application can only communicate through COM ports use the VSP-to-DS connection instead.*

Target DS

On this step you select the DS that the application on this PC will be communicating with. Notice, that the *Wizard* can only setup the DS side of the link. It is your responsibility to correctly setup the PC application!



In most cases you complete this step as follows:

- Select *Device Server is accessible from this PC*.
- Click *Select DS button* and choose the DS from the list. The *button* brings up the *dialog*, similar to the *main window* of the DS Manager. In fact, it is the *DS Manager*, with the following two differences:
  - There is an additional *Select button*. To select a particular DS single-click on the DS in the *device list* (in the auto-discoveryor address book access mode) and press *Select*. Alternatively, you can double-click on the DS in the *device list*.
  - The COM access mode is not available when the *DS Manager* is called from within *Wizard*. This is because you are choosing a network destination so the DS has to be selected from the list of devices, accessible through the network!
- Click *Next*.

This step provides a choice of several access options for the target DS- read additional info on accessing the DS for more information.

When you click *Next* the *Wizard* attempts to contact the target DS and verifies if this DS is local or remote. The DS of your choice must be available on the network, or the *Wizard* won't allow you to continue. Moreover, the IP-address of this DS

must be reachable*. If everything is OK the *Wizard* reads out all current setting values of this DS.

What if the DS is not visible from the PC

If the PC and the DS are located on different network segments it may well be that the DS is not "visible" from this PC but the PC is visible from the DS. This is normal as many networks are not symmetrical. For example, if the PC has a public (real) IP-address and the DS is located on a firewalled network segment then the DS can connect to the PC (application) , but the PC (application) cannot connect to the DS.

To specify that the DS is not visible from this PC select the *Device Server is not accessible from this PC* option and click *Next*.

There are two implications of this choice:

- In this application-to-DS connection it is the DS that will always be connecting to the application, since the application cannot connect to the DS.
- The *Wizard* won't be able to setup the DS through the network (this is because it cannot "see" the DS). You will have a chance to setup this inaccessible DS through its serial port or through a temporary network connection. You will be given a choice to do this within current Wizard job or generate a *configuration script* and finish the setup later, using the finish remote job feature of the *Connection Wizard*.

** The DS Manager and the Connection Wizard have the ability to "see" local Device Servers even if the IP-address of these devices is unreachable (this is done by using broadcast communications- read about this here). The reason why the Wizard demands that the Device Server has a proper IP-address is because the PC application will have to establish a real data connection with this DS and that **requires** a reachable IP-address.*

Additional Info on Accessing the DS

This topic provides additional information on the access options for the target DS.

The target DS can be selected from the list (using *Select DS button*) or its IP-address and access parameters can be input manually. The first option is usually more convenient and requires less manual input and decisions on your behalf.

Besides the IP (or MAC) you need to specify the *access method* and *access port* for this DS. Access method is the way the *Wizard* will access the DS in order to program it. Access port is the port to which the *Wizard* will send its programming commands. These options are not directly related to how the application will be accessing the DS, they just specify how the *Wizard* is going to access the DS.

There are two access methods:

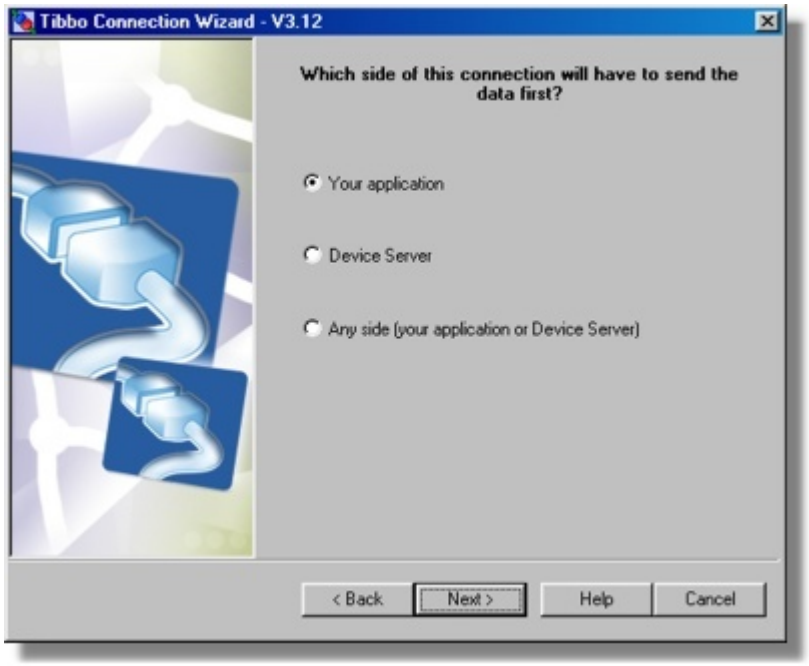
- **When out-of-band (UDP) access method** is selected the *Wizard* communicates with the DS using command UDP datagrams sent to the command port 65535 of the DS. Out-of-band commands offer a primary way of network programming that requires no preliminary setup on the DS side. This access method is suitable for all local Device Servers. It will also work with remote Device Servers unless the UDP traffic is banned by the network routers (firewalls, etc).
- **When inband (TCP) access method** is selected the *Wizard* communicates with the DS by sending commands through the TCP connection established to the data port of the DS (this is the number defined by the **Port Number (PN)**

setting of the DS). Inband (TCP) commands provide a secondary method of network programming that can be used in situations when out-of-band UDP access is impossible (for remote Device Servers). The downside is that a certain pre-programming must be done on the DS before it can be accessed using inband access method- read preparing the DS for inband access for step-by-step instructions.

When you select the DS from the address book (through the *Select DS* button) the *Wizard* copies the access method and access port from the address book entry.

Initiator of The Data Exchange

On this step you select which side of the application-to-DS connection will be the first to send the data.



Notice, that asking "who will be the first to send the data?" is not the same as asking "which side will be establishing the data connection to the other side?"*.

There are three choices:

- **Your application.** Select this option if it is the application software that will be sending the first data. This is typical for the kind of serial devices that we, as Tibbo, came to call "slave terminals". Such terminals communicate with the PC software using some sort of command-reply communication protocol and it is the PC that usually plays the role of a "master": the terminal won't send any data unless the PC sends a command first (and this is the "first data"!).
- **Device Server.** Select this option if it is the serial device that will be sending the first data. This is typical for the kind of serial devices that we call "scanners" (readers)**. This class of serial devices differs from "slave terminals" in that they send out the data spontaneously, without any prior prompt from the PC software.
- **Any side.** In some systems the data may first come from any side. For example, there are some "active terminals" out there that can send out the data spontaneously and at the same time respond to the commands from PC. Also choose this option if you are not sure which side of your system sends the data

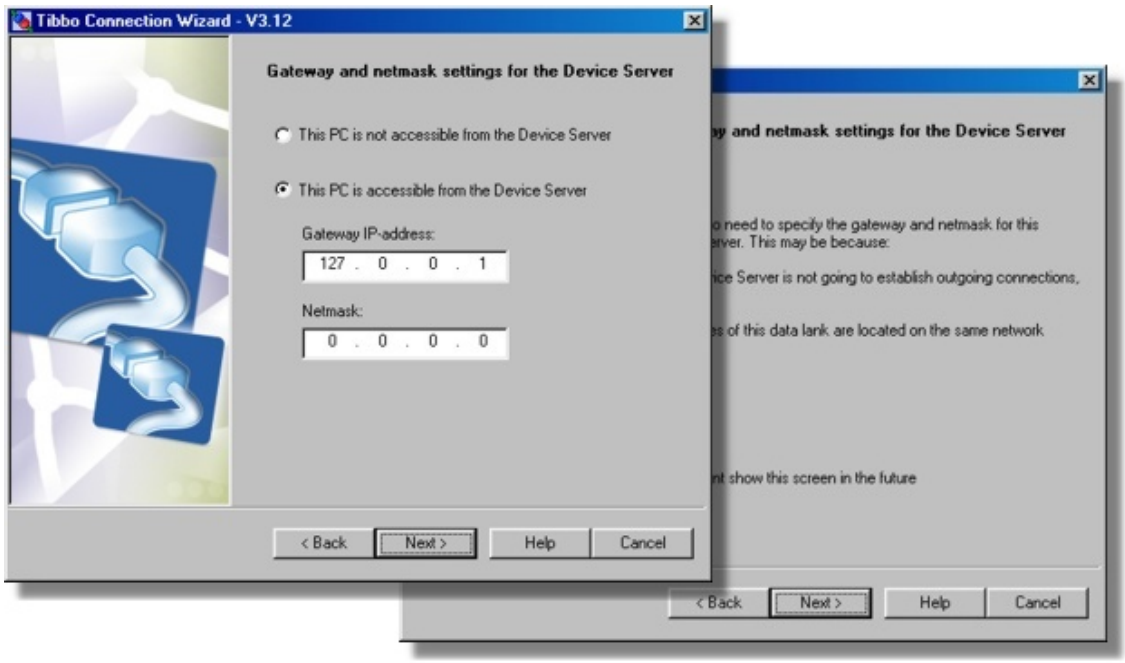
first.

** Although these two questions are connected- see how the Wizard decides which side open the connection.*

*** Meaning barcode scanners or magnetic card readers.*

Netmask & Gateway for the DS

This step comes in two options: you are either requested to input the gateway and netmask information that the DS needs to connect to this PC (right screenshot) or you are shown a screen informing you that the entry of this data is not necessary (left screenshot).



The gateway and netmask need only be set when two conditions are observed:

- The DS may need to (or must) establish data connections to the PC (this is determined by the *Wizard*).
- The DS and the PC are located on different network segments (there is at least one router between the PC and the DS).

Setting gateway and netmask information for the DS that only accepts incoming connections is not necessary*. If the DS is local there are no routers between this DS and the PC, so the gateway and netmask are not necessary even if the DS has to connect to the PC.

How to set the gateway and netmask

There is no straight way of finding out what data to enter. You need to obtain this information from somebody with a knowledge of your network's topology. Here is one hint, though.

If the IP-address of the target DS is configured through DHCP (**DHCP (DH) setting** is 1 (enabled)), then the DHCP server might have configured this Device Server's gateway and netmask as well. When you reach this step of the *Connection*

Wizard the *Gateway IP-address* and the *Netmask fields* are filled with the data from the **Gateway IP-address (GI)** and **Netmask (NM)** settings of the target DS**. This data may already be correct!

What if the PC is not visible from the DS

When the PC and the DS are located on different network segments it may well be that the PC is not "visible" from this DS. This is normal as many networks are not symmetrical. For example, if the DS has a public (real) IP-address and the PC is located on a firewalled network segment then the PC (application) can connect to the DS, but the DS cannot connect to the PC (application).

To provide for this situation the option box at the top of this step's screen allows you to specify that *this PC is not accessible from the DS*. Choose the option if this is so and if the option is available.

The option to specify that the PC is not accessible from the DS is disabled (not allowed to be chosen) when you have previously specified (at the target DS step) that the DS is not accessible from the PC. The reason is obvious: at least one side of the connection must be able to "see" the other side!

After you have completed this step the *Wizard* has enough information to decide which side of the application-to-DS link will be responsible for establishing the data connections.

** We found that some people hold a passionate belief that these parameters are necessary in any case. This is not true! If your network device doesn't have to establish outgoing connections you never have to bother about the gateway and netmask!*

*** Unless you have specified that "the DS is inaccessible through the network" at the target DS step of the Wizard, in which case the Wizard cannot obtain this information.*

How the Wizard Decides Who Opens Connections

When you click *Next* on the netmask and gateway step the *Wizard* has enough information to decide which side of the application-to-DS connection will be responsible for establishing data connections with the other side.

In doing this, the *Wizard* decides on the future values of the **Routing Mode (RM)** and **Connection Mode (CM)** settings of the DS. With VSP-to-DS connections, the *Wizard* would also be able to set the routingand connection modes on the *VSP* side. For the application-to-DS connection, the *Wizard* can only setup the DS side. It is your responsibility to set the application side as expected by the *Wizard*. For the discussion below we will assume that the application also has "imaginary" routing and connection options, similar in function to the ones found in *VSP*. Here is what this means:

***VSP* properties**	**Corresponding expected application behavior**
Server routing mode	Once the application is opened it is ready to accept incoming connections. The application never establishes outgoing connections.
Server/client routing mode	Once the application is opened it is ready to accept incoming connections and establish outgoing connections to the specified IP-address and port number (whichever happens first)

Client routing mode	The application only establishes outgoing connections to the specified IP-address and port number. Incoming connections are not accepted
On data connection mode	The application establishes an outgoing connection when it has the data to send (that is, if outgoing connections are allowed)
Immediate connection mode	The application establishes an outgoing connection as soon as it is started, even if there is no data to transmit (that is, if outgoing connections are allowed)

In general, the *Wizard* tries to follow the natural flow of data between the application and the serial device. In the initiator of the data exchange step you have already specified which side sends the data first, so the *Wizard* will have this side open a data connection to the other side as soon as the first data needs to be sent from this side of the connection.

For example, if you are dealing with the "scanner" type of serial device, then the first data is always sent by the serial device. Therefore, the *Wizard* will program the **Routing Mode (RM) setting** of the DS to "client". Consequently, the DS will be opening the data connections to the PC application so the application needs to be ready to accept those connections (in *VSP* terms this is the server routing mode).

Lines 1-3 of the following table illustrate what's just been said. Notice that the **Connection Mode (CM) setting** is always set to "on data". As the first data is received (by one side of the link) it triggers an attempt to open a data connection with the other side of the link:

	Which side can "see" the other side of the connection	**Which side sends the first data**	**Routing Mode on the appl**.**	**Connection Mode on the appl**.**	**Routing Mode on the DS**	**Connection Mode on the DS**
1	Both sides can see each other	Application	Client	On data	Server	---
2	Both sides can see each other	Serial device	Server	---	Client (server/client)	On data
3	Both sides can see each other	Any side	Server/client	On data	Server/client	On data
4	Only PC can see the DS	Application	Client	On data	Server	---
5	Only PC can see the DS	Serial device	Client	Immediately	Server	---
6	Only PC can see the DS	Any side	Client	Immediately	Server	---

7	Only DS can see the PC	Application	Server	---	Client	Immediately
8	Only DS can see the PC	Serial device	Server	---	Client	On data
9	Only DS can see the PC	Any side	Server	---	Client	Immediately

The situation becomes more complicated when only one side of the connection can "see" the other side. Consider the case in line 7 of the table. The application has to send the data first, but it cannot "see" the DS. Therefore, even when the application needs to transmit the data it will be unable to establish a connection to the DS!

The way out is to have the DS connect to the PC (application) as soon as the DS is powered (so, the **Connection Mode (CM) setting** is 0 (immediately)). This way, by the time the application needs to send the first data to the DS the connection is already established. This kind of connections are called "reverse".

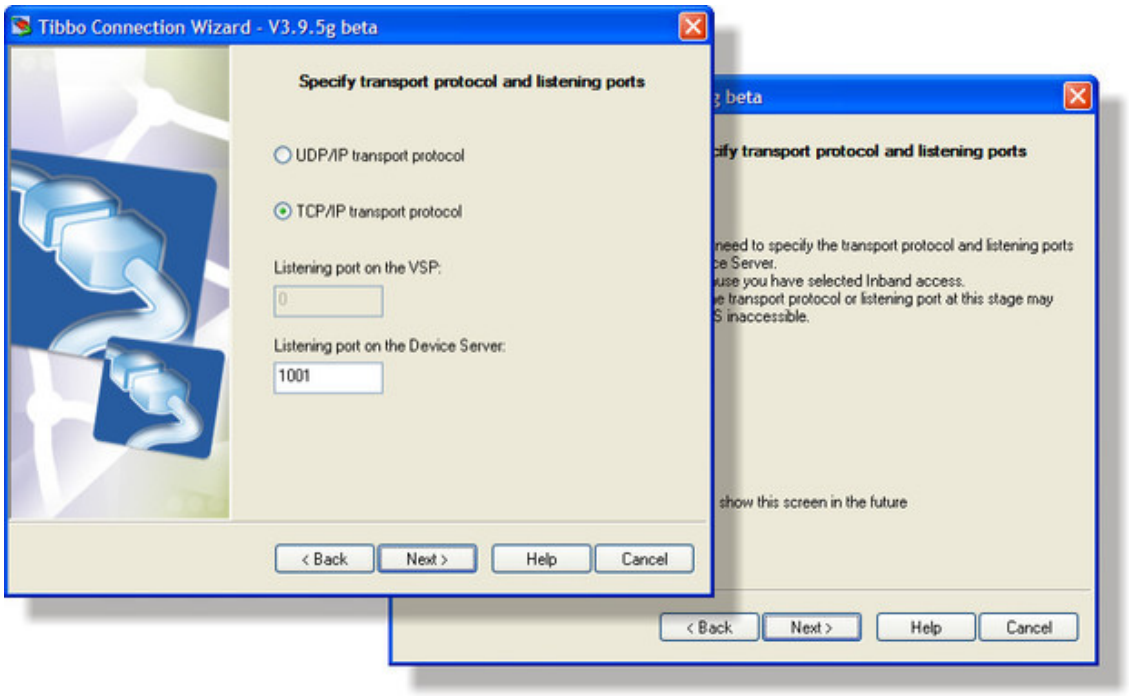
A special comment should be made on line 2 of the table, where the Routing Mode on the DS is designated to be either "client" or "server/client". The *Wizard* chooses "client" if DS programming is effected using out-of-band access method (you have selected this on the target DS step of the *Wizard*). If inband access method was selected the *Wizard* will set the mode to "server/client". This is because choosing "client" would cause it DS to reject all incoming connections in the future and this means that you wouldn't be able to program the DS using inband access in the future!

** This is the case when you have specified that "the Device Server is not accessible from this PC" on the target DS step, or that "the PC is not accessible from this Device Server" on the netmask and gateway for the DS step of the Wizard.*

*** "Imaginary" options of the PC application.*

Transport Protocol & Listening Ports

On this step you select the protocol that will be used for exchanging the data between the application and the DS. You also choose the listening port number on the side(s) of the link that will be receiving incoming connections from the other side.



In general, we recommend you to stick to the TCP/IP, unless the application requires that the UDP/IP protocol is used. Since the *Wizard* can only setup the transport protocol on the DS side of this connection, it is your responsibility to setup the application side accordingly.

There is one case when the UDP/IP selection is not available and the TCP protocol is pre-selected for you- this is when you have specified an inband access method on the target DS step of the *Wizard*. Since inband access requires a TCP/IP data connection with the DS you must use TCP/IP transport protocol!

Under What Circumstances Transport Protocol & Port Selection Is Disabled

As noted above, when you use Inband access mode for the wizard, you must use TCP/IP as the transport protocol. Also, port selection will be disabled in this case. At this stage of the Connection Wizard you are already in communication with a DS. The communication is done via the data transport channel of the DS (as opposed to a separate command channel, like in Telnet or out-of-band mode).

The DS is usually far away (somewhere on a WAN), otherwise you would not use inband access to reach it. Thus, there are all sorts of firewalls and gateways between yourself and the DS. Firewalls only allow traffic on certain ports to go through, and UDP packets are often dropped on WANs.

If you change the protocol to UDP now, or change the listening port on the DS, you may render it completely inaccessible. The change *will* occur, because the Wizard is in communication with the DS (on the proper port which you configured in the beginning). But at the end of the Wizard run, you'll have an unpleasant surprise - the DS may suddenly disappear.

Thus, when selecting Inband Access mode in the beginning of the Wizard, you cannot later change the Transport Protocol or Listening Port.

Listening ports

This screen also provides an option of entering the port number on the listening side(s) of application-to-DS link. By now the *Wizard* has already decided which side

opens the connections. Connecting side needs to know the number of the listening port on the other side. For example, if it is the PC application that will always be connecting to the DS, then you only have to specify the listening port on the DS side. Consequently, the *listening port on the DS textbox* will be enabled, and the *listening port on the PC textbox* will be disabled. If both sides will need to establish the connection, then you have to specify the listening ports on both sides too, and both textboxes will be active.

There is one case when the listening port on the DS side is fixed and pre-selected for you- this is when you have specified an inband access method for this DS. In this case you have already specified the listening port (as the access port) on the target DS stepof the *Wizard* (listening port and the access port are the same for inband mode).

Once again, proper setup of the application side is your responsibility. For example, if you choose the listening port on the DS to be 1001 the *Wizard* will program this number into the **Port Number (PN) setting** on the DS automatically, and your job will be to set the destination port number on the application side to 1001. Likewise, if you choose the listening port on the application side to be 3500, then the *Wizard* will automatically set the **Destination Port (DP) setting** of the DS to 3500, but you need to make sure that the same listening number is actually selected in the application.

How the listening port numbers are chosen

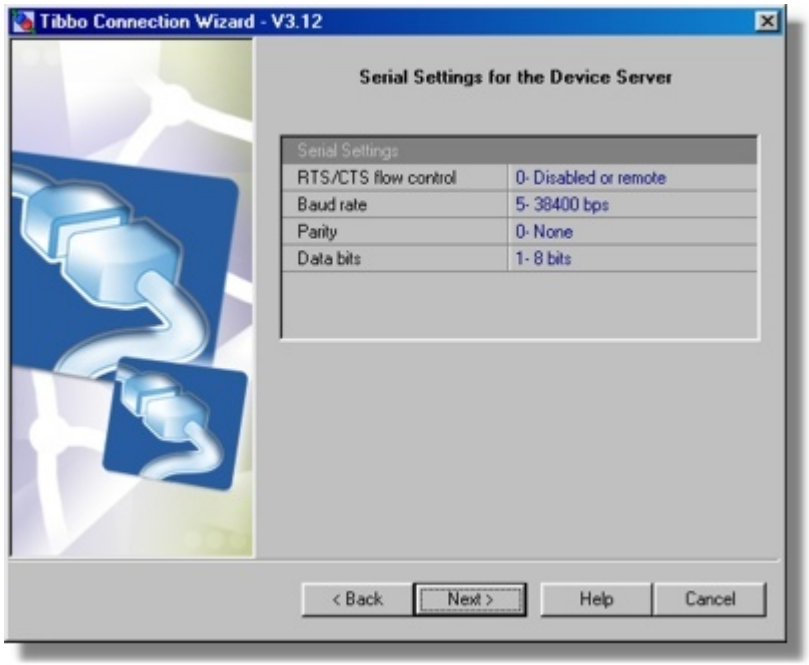
You can choose any port numbers of your liking. The only limitations are:

- PC applications often have fixed destination and/or listening ports so the DS settings may need to be adjusted to suit the requirements of your application.
- Certain firewall or router setup may require you to use specific port numbers for communications between the application and the DS.

The DS can be adjusted to use any destination and local port numbers. The only limitation is that the **Port Number (PN) setting** cannot be programmed to 65535 (this is a special command port number).

Parameters for the Serial port of the DS

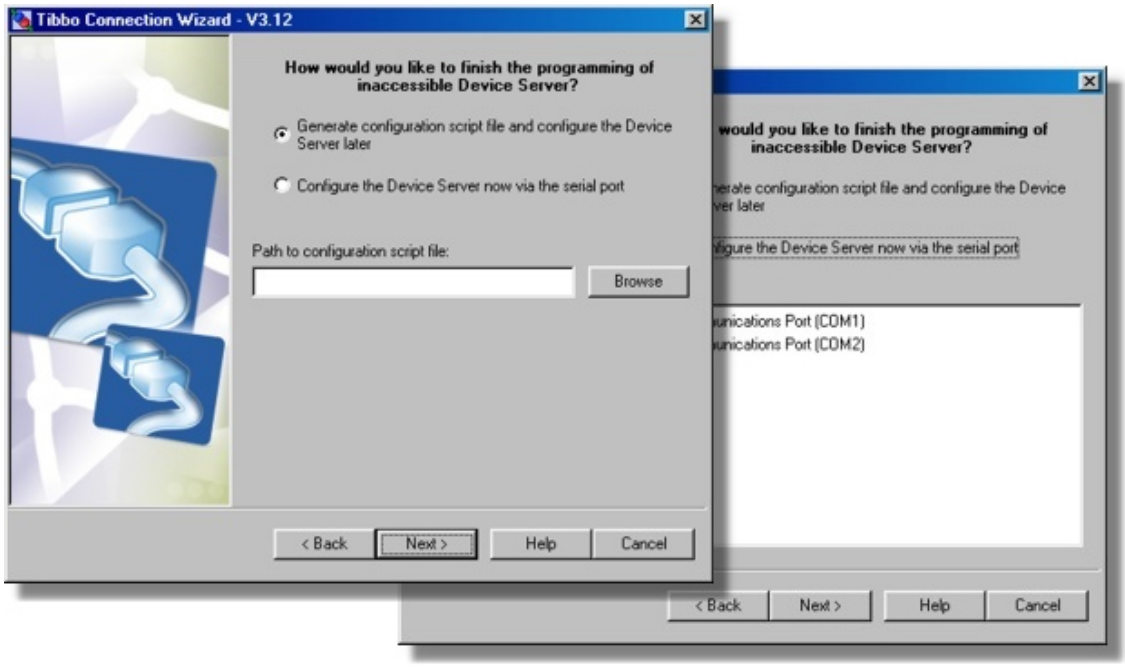
On this step you select communication parameters for the serial port of the DS.



Unlike with the VSP-to-DS link, the serial port parameters of the DS, involved in the application-to-DS link, cannot be adjusted via on-the-fly commands. You have to define fixed parameters that will be used by the DS at all times.

Programming an Inaccessible DS

If this screen is shown then you have specified on the target DS step step that the *Device Server is not accessible from this PC*. Now you have to decide how you will setup this DS.



There are two choices:

- **Generate configuration script** (left screenshot). The script file can later be fed into the *Connection Wizard*, possibly running on a *different* PC (see finishing remote job). Choose this option if you cannot physically bring the DS to this PC for programming. Input the path and filename for the script file (left screenshot) and press *Next*.
- **Configure the DS through the serial port** (right screenshot). Choose this option if you can temporary bring the DS to this PC for programming. Connect the serial port of the DS to the unused COM port of your PC (using WAS-1455or similar cable), make sure the DS is powered up and press the setup button* (this will put the DS into the serial programming mode). Click *Next* to continue (the *Wizard* will read out current setting values of the DS).

Note:

 Status LEDs of the DS are playing a *serial programming mode pattern* (shown on the left) when the serial port of the DS is in the serial programming mode (click here to see all available patterns).

** On EM100, EM120, EM200, EM203(A)- pull the MD line low for at least 100ms.*

DS-to-DS Link

This job is selected by choosing *create a link between two Device Servers* from the list of available jobs.

When performing this job the *Connection Wizard* configures two Device Servers of your choice to communicate with each other. This effectively creates a "virtual serial cable" that interconnects the serial ports of these two Device Servers. Not only the serial data, but also the state of the RTS, CTS, DTR, and DSR signals can be seamlessly transmitted across this virtual cable!

There are two possible applications for the DS-to-DS link:

- To interconnect two non-PC devices (shown on the diagram above).
- To interconnect a serial device with a non-*Windows* PC software. Tibbo Virtual Serial Portscan only work under *Windows* so if you have a case where you need to network-enable a system involving a non-*Windows* software (i.e. DOS, etc.) you can do this by using a second DS located on the PC side.

In this case the software still communicates through a real COM port, but the DS on the PC side connects this COM to the network. This is effectively the same kind of "virtual serial cable" connection. Device Servers do not know (nor care) what kind of serial devices are communicating through them.

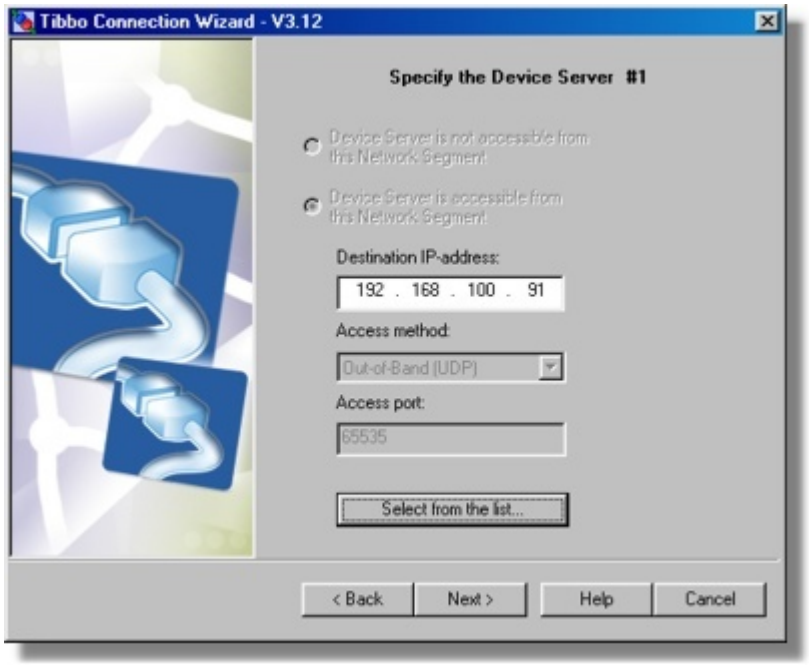
To setup the DS-to-DS link through the *Wizard* you need to use a *Windows* PC the *Wizard* will run on (shown on both diagrams as "configuration" PC). Once the link is created this PC is no longer needed as it is not directly participating in the data link.

One limitation that the *Wizard* applies to the DS-to-DS link is that at least one of the Device Servers in the link must reside on the same network segment* as the configuration PC. The other DS may be located anywhere (be local or remote).

** I.e. there must be no routers (firewalls, etc.) between the configuration PC and the DS.*

DS #1 (Must be Local)

On this step you select the first DS in the DS-to-DS link (it is called the "DS #1").



As explained earlier, the DS #1 must be local, i.e. it must be located on the same network segment as the PC on which this *Wizard* is running (DS #2 can be local or remote). Because of that, the accessibility option is fixed at *Device Server is assessable from this network segment* (if the DS is local it is definitely must be accessible).

Several access options that exist for the DS are also disabled because they are only necessary when you are dealing with the remote DS and the DS #1 cannot be remote.

The only editable option is the IP-address of the DS. You can input this IP-address manually or select the DS from the list:

- Click *Select DS button* and choose the DS from the list. The *button* brings up the *dialog*, similar to the *main window* of the DS Manager. In fact, it is the *DS Manager*, with the following two differences:
  - There is an additional *Select button*. To select a particular DS single-click on the DS in the *device list* (in the auto-discoveryaccess mode) and press *Select*. Alternatively, you can double-click on the DS in the *device list*.
  - The address book and COM access mode are not available. The DS #1 must be local and so you must be able to find it in the auto-discovery mode.
- Click *Next*.

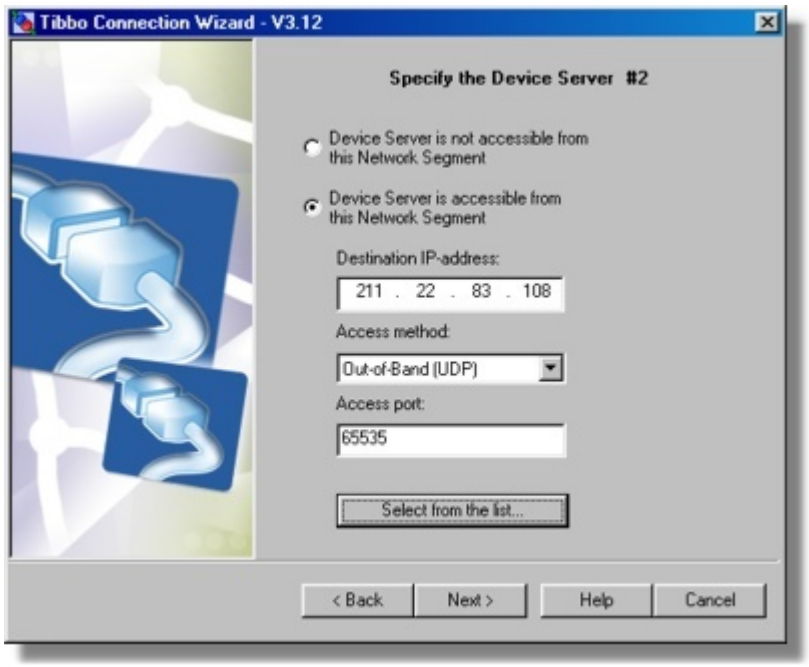
When you click *Next* the *Wizard* attempts to contact the target DS and makes sure that the DS is local and reachable*. If everything is OK the *Wizard* reads out all current setting values of this DS.

** The DS Manager and the Connection Wizard have the ability to "see" local Device Servers even if the IP-address of these devices is unreachable (this is done by using broadcast communications- read about this here). The reason why the Wizard demands that the DS #1 has a proper IP-address is because the DS #2 will have to establish a real data connection to this DS and that requires a reachable*

IP-address.

DS #2 (Can be Local or Remote)

On this step you select the second DS in the DS-to-DS link (it is called the "DS #2").



The DS #2 can be located anywhere, i.e. be local or remote.

In most cases you complete this step as follows:

- Select *Device Server is accessible from this network segment*.
- Click *Select DS button* and choose the DS from the list. The *button* brings up the *dialog*, similar to the *main window* of the DS Manager. In fact, it is the *DS Manager*, with the following two differences:
  - There is an additional *Select button*. To select a particular DS single-click on the DS in the *device list* (in the auto-discoveryor address book access mode) and press *Select*. Alternatively, you can double-click on the DS in the *device list*.
  - The COM access mode is not available when the *DS Manager* is called from within *Wizard*. This is because you are choosing a network destination for the *VSP* so the DS has to be selected from the list of devices, accessible through the network!
- Click *Next*.

This step provides a choice of several access options for the DS #2- read additional info on accessing the DS for more information.

When you click *Next* the *Wizard* attempts to contact the target DS and verifies if this DS is local or remote. The DS of your choice must be available on the network, or the *Wizard* won't allow you to continue. Moreover, the IP-address of this DS must be reachable*. If everything is OK the *Wizard* reads out all current setting values of this DS.

What if the DS #2 is not visible from the PC

If the PC and the DS #2 are located on different network segments it may well be that the DS #2 is not "visible" from this network segment (and hence, from the DS #1) but the DS#1 is visible from the DS #2. This is normal as many networks are not symmetrical. For example, if the DS #1 has a public (real) IP-address and the DS #2 is located on a firewalled network segment then the DS #2 can connect to the DS #1, but the DS#1 (and this PC) cannot connect to the DS #2.

To specify that the DS is not visible from this PC select the *Device Server is not accessible from this network segment* option and click *Next*.

There are two implications of this choice:

- In this DS-to-DS connection it is the DS #2 that will always be connecting to the DS #1, since DS #1 cannot connect to the DS #2.
- The *Wizard* won't be able to setup the DS through the network (this is because it cannot "see" the DS). You will have a chance to setup this inaccessible DS through its serial port or through a temporary network connection. You will be given a choice to do this within current Wizard job or generate a *configuration script* and finish the setup later, using the finish remote job feature of the *Connection Wizard*.

** The DS Manager and the Connection Wizard have the ability to "see" local Device Servers even if the IP-address of these devices is unreachable (this is done by using broadcast communications- read about this here). The reason why the Wizard demands that the DS #2 has a proper IP-address is because the DS #1 will probably have to establish a real data connection with this DS and that requires a reachable IP-address.*

Additional Info on Accessing the DS

This topic provides additional information on the access options for the DS #2.

The DS #2 can be selected from the list (using *Select DS button*) or its IP-address and access parameters can be input manually. The first option is usually more convenient and requires less manual input and decisions on your behalf.

Besides the IP (or MAC) you need to specify the *access method* and *access port* for this DS. Access method is the way the *Wizard* will access the DS #2 in order to program it. Access port is the port to which the *Wizard* will send its programming commands. These options are not directly related to how the application will be accessing the DS, they just specify how the *Wizard* is going to access the DS.

There are two access methods:

- **When out-of-band (UDP) access method** is selected the *Wizard* communicates with the DS using command UDP datagrams sent to the command port 65535 of the DS. Out-of-band commands offer a primary way of network programming that requires no preliminary setup on the DS side. This access method is suitable for all local Device Servers. It will also work with remote Device Servers unless the UDP traffic is banned by the network routers (firewalls, etc).
- **When inband (TCP) access method** is selected the *Wizard* communicates with the DS by sending commands through the TCP connection established to the data port of the DS (this is the number defined by the **Port Number (PN) setting** of the DS). Inband (TCP) commands provide a secondary method of network programming that can be used in situations when out-of-band UDP access is impossible (for remote Device Servers). The downside is that a certain

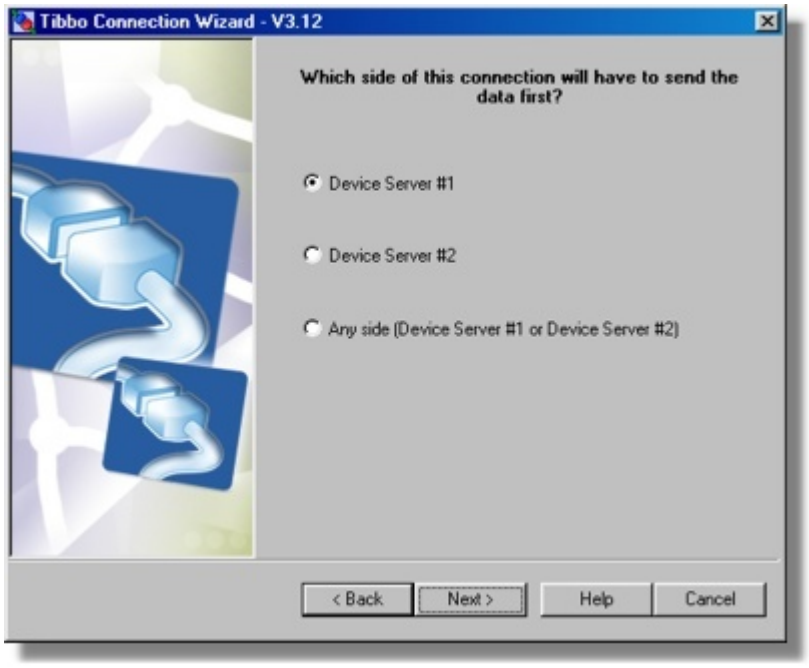
pre-programming must be done on the DS before it can be accessed using inband access method- read preparing the DS for inband access for step-by-step instructions.

Inband access method is usually utilized when out-of-band access is not possible (i.e. because of restrictions on the network). To allow future programming of the DS #2 the *Wizard* will preserve inband access in the DS #2 and this means that the **Inband (IB) setting** of the DS will be kept at 1 (enabled). Because in this mode ASCII characters with code 255 (FF) are handled in a special way the **Inband (IB) setting** of the DS #1 will also be set to 1 (enabled). This will ensure correct transmission of FF characters between the Device Servers.

When you select the DS #2 from the address book (through the *Select DS* button) the *Wizard* copies the access method and access port from the address book entry.

Initiator of the Data Exchange

On this step you select which side of the DS-to-DS connection will be the first to send the data.



k

Notice, that asking "who will be the first to send the data?" is not the same as asking "which side will be establishing the data connection to the other side?"* When making your choice imagine that there is no network at all and that two serial devices are interconnected via the serial cable. Now, which side sends the data first?

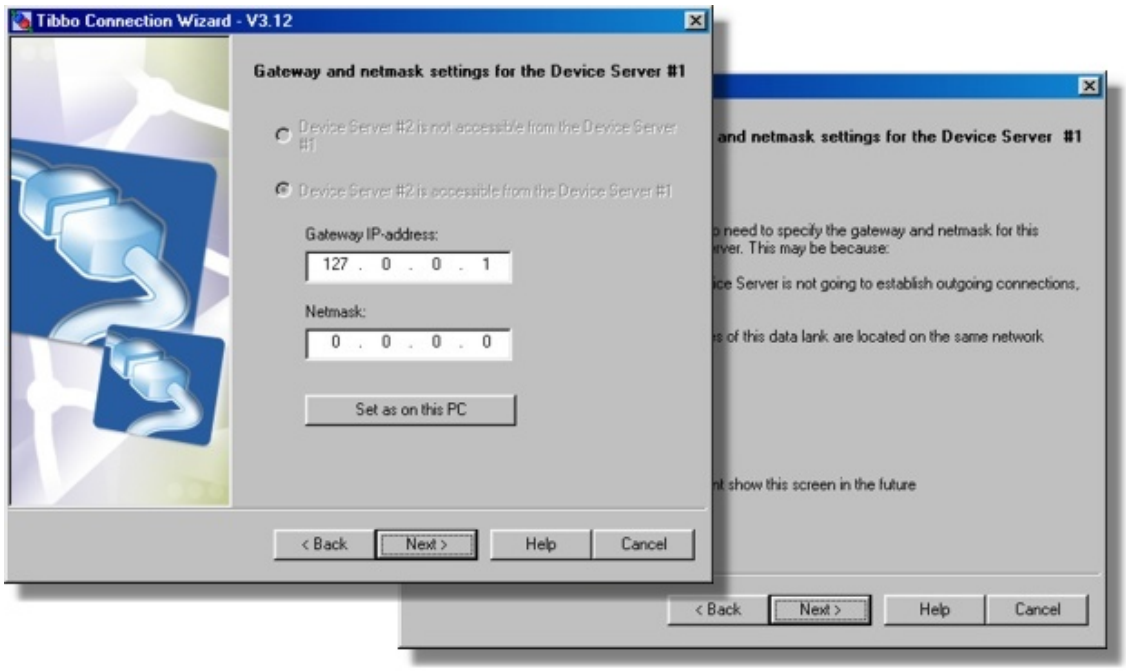
There are three choices:

- **Device Server #1.** Select this option if the first data will be sent by the serial device connected to the DS #1.
- **Device Server #2.** Select this option if the first data will be sent by the serial device connected to the DS #2.
- **Any side.** In some systems the first data may come from any side. Also choose this option if you are not sure which side of your system sends the data first.

** Although these two questions are connected- see [how the Wizard decides which side open the connection](#).*

Netmask & Gateway for the DS #1

This step comes in two options: you are either requested to input the gateway and netmask information that the DS #1 needs to connect to the DS #2 (right screenshot) or you are shown a screen informing you that the entry of this data is not necessary (left screenshot).



The gateway and netmask need only be set when two conditions are observed:

- The DS #1 may need to (or must) establish data connections to the DS #2.
- The DS #1 and the DS #2 are located on different network segments (there is at least one router between them).

Setting gateway and netmask information for the DS that only accepts incoming connections is not necessary*. If both Device Servers are located on the same network segment, the gateway and netmask are not necessary even if the DS #1 has to connect to the DS #2.

How to set the gateway and netmask

Since the DS #1 is located on the same network segment with this PC the gateway and netmask values that need to be set for this DS are probably the same as the ones set for the PC. Clicking *Set as on the PC button* copies the gateway and netmask information from the PC.

If the IP-address of the DS #1 is configured through DHCP (**DHCP (DH) setting** is 1 (enabled)), then the DHCP server might have configured this Device Server's gateway and netmask as well. When you reach this step of the *Connection Wizard* the *Gateway IP-address* and the *Netmask fields* are filled with the data from the **Gateway IP-address (GI)** and **Netmask (NM)** settings of the target DS. This data may already be correct!

What if the DS #2 is not visible from the DS #1

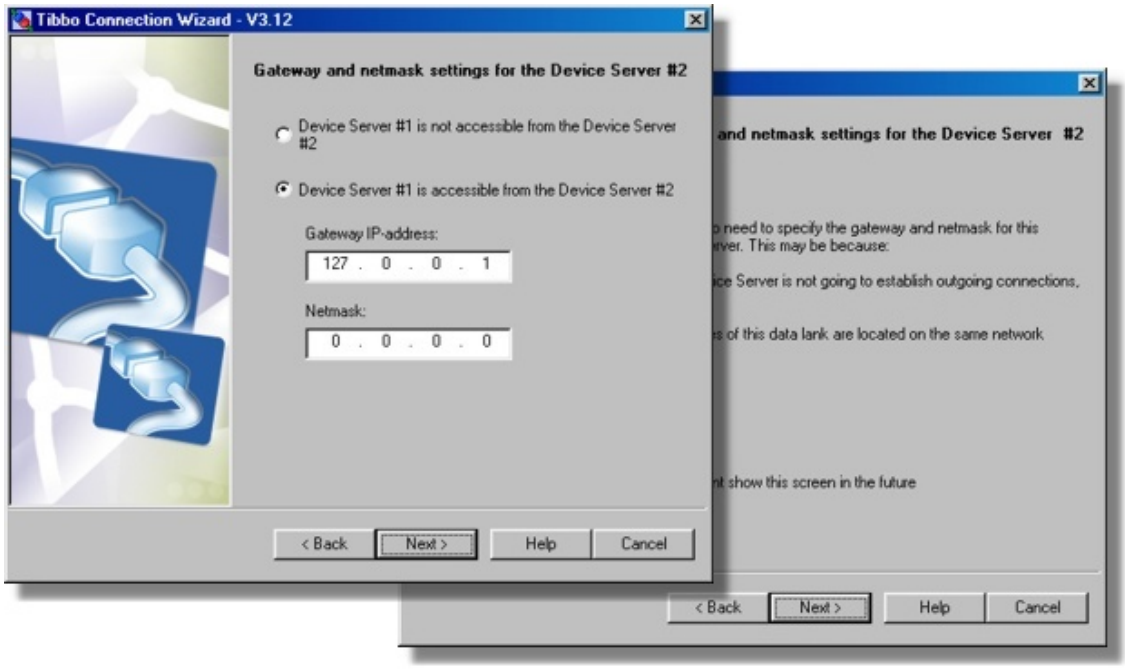
When the DS #1 and the DS #2 are located on different network segments it may well be that the DS #2 is not "visible" from the DS #1. This is normal as many networks are not symmetrical. Since the DS #1 and this PC are located on the same network segment the visibility of the DS #2 is the same for the PC and the DS #1. This is why the options at the top of the screen are disabled and cannot be changed:

- The *DS #2 is accessible from the DS #1* is selected if you have specified that the *Device Server #2 is accessible from this network segment* on the DS #2 step of the *Wizard*.
- The *DS #1 is not accessible from the DS #1* is selected if you have specified that the *Device Server is not accessible from this network segment* on the DS #2 step of the *Wizard*.

** We found that some people hold a passionate belief that these parameters are necessary in any case. This is not true! If your network device doesn't have to establish outgoing connections you never have to bother about the gateway and netmask!*

Netmask & Gateway for the DS #2

This step comes in two options: you are either requested to input the gateway and netmask information that the DS #2 needs to connect to the DS #1 (right screenshot) or you are shown a screen informing you that the entry of this data is not necessary (left screenshot).



The gateway and netmask need only be set when two conditions are observed:

- The DS #2 may need to (or must) establish data connections to the DS #1.
- The DS #2 and the DS #1 are located on different network segments (there is at least one router between them).

Setting gateway and netmask information for the DS that only accepts incoming

connections is not necessary*. If both Device Servers are on the same network segment, the gateway and netmask are not necessary even if the DS #2 has to connect to the DS #1.

How to set the gateway and netmask

There is no straight way of finding out what data to enter. You need to obtain this information from somebody with a knowledge of your network's topology. Here is one hint, though.

If the IP-address of the DS #2 is configured through DHCP (**DHCP (DH) setting** is 1 (enabled)), then the DHCP server might have configured this Device Server's gateway and netmask as well. When you reach this step of the *Connection Wizard* the *Gateway IP-address* and the *Netmask fields* are filled with the data from the **Gateway IP-address (GI)** and **Netmask (NM)** settings of the DS #2**. This data may already be correct!

What if the DS #1 is not visible from the DS #2

When two Device Servers are located on different network segments it may well be that the DS #1 is not "visible" from the DS #2. This is normal as many networks are not symmetrical. For example, if the DS #2 has a public (real) IP-address and the DS #1 is located on a firewalled network segment then the DS #1 can connect to the DS #2, but the DS #2 cannot connect to the DS #1.

To provide for this situation the option box at the top of this step's screen allows you to specify that the *DS #1 is not accessible from the DS #2*. Choose the option if this is so and if the option is available.

The option to specify that the DS #1 is not accessible from the DS #2 is disabled (not allowed to be chosen) when you have previously specified (at the DS #2 step) that the DS #2 is not accessible from the DS #1. The reason is obvious: at least one side of the connection must be able to "see" the other side!

After you have completed this step the *Wizard* has enough information to decide which side of the DS-to-DS link will be responsible for establishing the data connections.

** We found that some people hold a passionate belief that these parameters are necessary in any case. This is not true! If your network device doesn't have to establish outgoing connections you never have to bother about the gateway and netmask!*

*** Unless you have specified that "the Device Server is not accessible from this network segment" at the DS #2 step of the Wizard, in which case the Wizard cannot obtain this information.*

How the Wizard Decides Who Opens the Connection

When you click *Next* on the netmask and gateway for the DS #2 step the *Wizard* has enough information to decide which side of the DS-to-DS connection will be responsible for establishing data connections with the other side.

In doing this, the *Wizard* decides on the future values of the **Routing Mode (RM)** and **Connection Mode (CM)** settings of both Device Servers.

In general, the *Wizard* tries to follow the natural flow of data between the Device Servers. In the initiator of the data exchange step you have already specified which side sends the data first, so the *Wizard* will have this side open a data connection to the other side as soon as the first data needs to be sent.

For example, if the first data is supposed to come from the DS #1 side the *Wizard*

will program the **Routing Mode** of the DS #1 to 2 (client) and the **Routing Mode** of the DS #2 to 0 (server).

Lines 1-3 of the following table illustrate what's just been said. Notice that the **Connection Mode (CM) setting** is always set to "on data". As the first data is received (by one side of the link) it triggers an attempt to open a data connection with the other side of the link:

	Which side can "see" the other side of the connection	Which side sends the first data	Routing Mode on the DS #1	Connection Mode on the DS #1	Routing Mode on the DS #2	Connection Mode on the DS #2
1	Both sides can see each other	DS #1	Client	On data	Server	---
2	Both sides can see each other	DS #2	Server	---	Client (server/client)	On data
3	Both sides can see each other	Any side	Server/client	On data	Server/client	On data
4	Only DS #1 can see the DS #2	DS #1	Client	On data	Server	---
5	Only DS #1 can see the DS #2	DS #2	Client	Immediately	Server	---
6	Only DS #1 can see the DS #2	Any side	Client	Immediately	Server	---
7	Only DS #2 can see the DS #1	DS #1	Server	---	Client	Immediately
8	Only DS #2 can see the DS #1	DS #2	Server	---	Client	On data
9	Only DS #2 can see the DS #1	Any side	Server	---	Client	Immediately

The situation becomes more complicated when only one side of the connection can "see" the other side*. Consider the case in line 5 of the table. The DS #2 has to send the data first, but it cannot "see" the DS #1. Therefore, even when the DS #2 receives the first data into its serial port it will be unable to establish a connection to the DS #1 and send this data!

The way out is to have the DS #1 connect to the DS #2 as soon as the DS #1 is

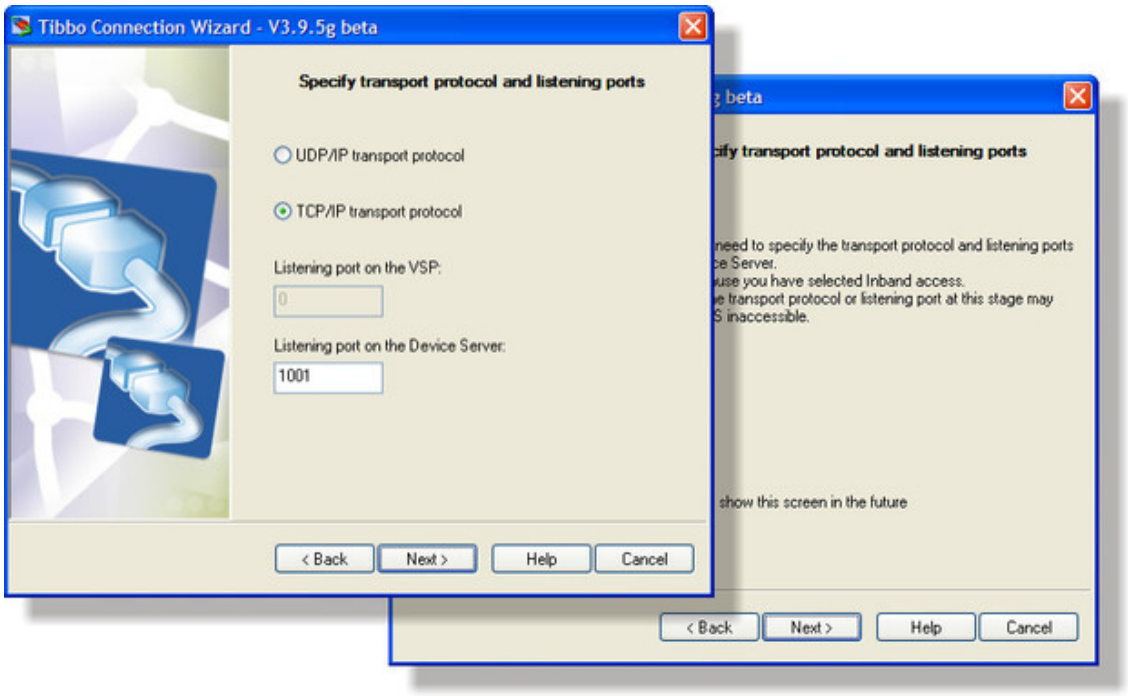
powered up (so, the **Connection Mode (CM) setting** of this DS is to be 0 (immediately)). This way, by the time the DS #2 needs to send the first data to the DS #1 the connection is already established. This kind of connections are called "reverse".

A special comment should be made on line 2 of the table, where the Routing Mode on the DS #2 is designated to be either "client" or "server/client". The *Wizard* chooses "client" if DS #2 programming is effected using out-of-band access method (you have selected this on the DS #2 step of the *Wizard*). If inband access method was selected the *Wizard* will set the mode to "server/client". This is because choosing "client" would cause it DS to reject all incoming connections in the future and this means that you wouldn't be able to program the DS using inband access in the future!

** This is the case when you have specified that the "Device Server is not accessible from this network segment" on the DS #2 step, or that the "DS #1 is not accessible from DS #2" on the netmask and gateway for the DS #2 step of the Wizard.*

Transport Protocol & Listening Ports

On this step you select the protocol that will be used for exchanging the data between the Device Servers. You also choose the listening port number of the side(s) of the link that will be receiving incoming connections.



In general, we recommend you to stick to the TCP/IP, unless you have a good reason (which is very rare!) why you have to use UDP/IP protocol.

There is one case when the UDP/IP selection is not available and the TCP protocol is pre-selected for you- this is when you have specified an inband access method on the DS #2 step of the *Wizard*. Since inband access requires a TCP/IP data connection with the DS you must use TCP/IP transport protocol, or the configuration PC won't be able to access the DS #2 in the future.

Under What Circumstances Transport Protocol & Port Selection Is Disabled

As noted above, when you use Inband access mode for the wizard, you must use TCP/IP as the transport protocol. Also, port selection will be disabled in this case. At this stage of the Connection Wizard you are already in communication with a DS. The communication is done via the data transport channel of the DS (as opposed to a separate command channel, like in Telnet or out-of-band mode).

The DS is usually far away (somewhere on a WAN), otherwise you would not use inband access to reach it. Thus, there are all sorts of firewalls and gateways between yourself and the DS. Firewalls only allow traffic on certain ports to go through, and UDP packets are often dropped on WANs.

If you change the protocol to UDP now, or change the listening port on the DS, you may render it completely inaccessible. The change *will* occur, because the Wizard is in communication with the DS (on the proper port which you configured in the beginning). But at the end of the Wizard run, you'll have an unpleasant surprise - the DS may suddenly disappear.

Thus, when selecting Inband Access mode in the beginning of the Wizard, you cannot later change the Transport Protocol or Listening Port.

Listening ports

This screen also provides an option of entering the port number on the listening side(s) of the DS-to-DS link. By now the *Wizard* has already decided which side opens the connections. Connecting side needs to know the number of the listening port on the other side. For example, if it is the DS #1 that will always be connecting to the DS #2, then you only have to specify the listening port on the DS #1 side. Consequently, the *listening port on the DS #2 textbox* will be enabled, and the *listening port on the DS #1 textbox* will be disabled. If both sides will need to establish the connection, then you have to specify the listening ports on both sides too, and both textboxes will be active.

There is one case when the listening port on the DS #2 side is fixed and pre-selected for you- this is when you have specified an inband access method for this DS. In this case you have already specified the listening port (as the access port) on the DS #2 step of the *Wizard* (listening port and the access port are the same for inband mode).

Once the listening port on one side is known, the *Wizard* sets the destination port on the other side accordingly. If the DS #1 will need to connect to the DS #2 the **Destination Port (DP) setting** on the DS #1 will be the same, as the value of the **Port Number (PN) setting** on the DS #2 side. Likewise, if the DS #2 will have to connect to the DS #1 the **Destination Port (DP) setting** on the DS #2 will be the same as the **Port Number (PN) setting** on the DS #1.
ration for both connection directions, so the **Destination Port** on one side will point at the **Porn Number** on the other side.

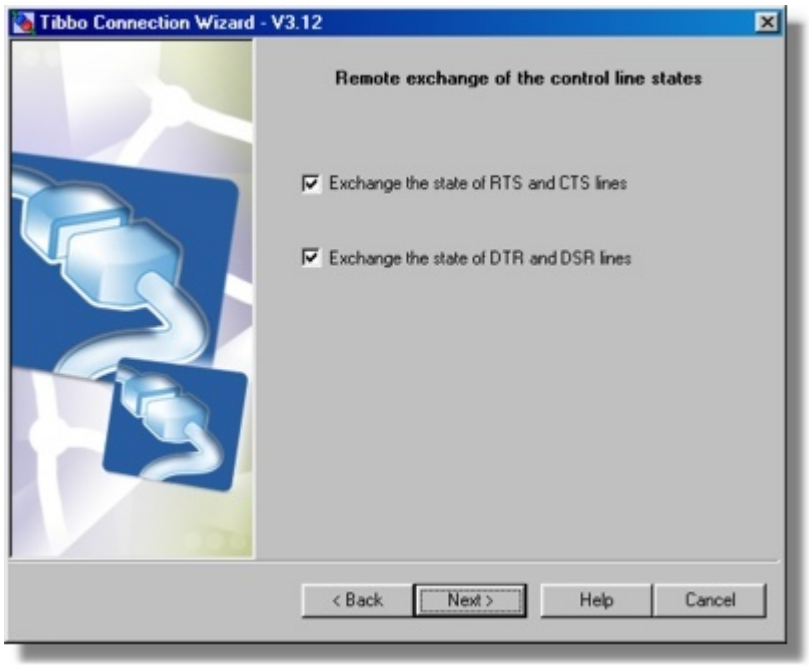
How the listening port numbers are chosen

You can choose any port numbers of your liking, except the 65535. This is because 65535 is a special command port. What the *Wizard* is showing you by default is the default value of the **Port Number (PN) setting** of the DS on the receiving end of the connection (1001).

The only reason to change suggested port numbers is if your network's firewall bans most of the traffic so only specific ports are opened for communications. In this case you may need to adjust the port numbers to the requirements of your network.

Remote Exchange for RTS, CTS, DTR, DSR

On this step you select whether the status of RTS, CTS, DTR, and DSR lines will be exchanged between the Device Servers.



When this feature is enabled, a change in the state of the CTS (DSR) input on one DS causes a corresponding change on the RTS (DTR) output of the other DS. In effect, this achieves compete emulation of the serial cable over the network: not only the serial data is seamlessly transmitted between the serial ports of two Device Servers, but also the control lines appear to be interconnected.

Exchange of signal states is enabled separately for the RTS/CTS and DTR/DSR signal pairs (there are two checkboxes- *exchange the state of RTS and CTS lines* and *exchange the state of DTR and DSR lines*).

If you enable the status exchange for the RTS and CTS lines you won't be able to enable RTS/CTS flow control for the serial ports of the DS (on parameters for the serial port of the DS #1 and parameters for the serial port of the DS #2 *Wizard* steps.

Control line status exchange and the connection mode of the DS

Exchange of the control line states is done through **Notification (J) Messages** sent between the Device Servers. Notification messages are sent only when the data connection is already established. This means that if there is no data connection between the Device Servers remote exchange of control line states doesn't work too. This may be a problem on some serial system that may need to exchange the state of control lines even before any data is sent.

To avoid this problem the *Wizard* will program the **Connection Mode (CM) setting** of one of the Device Servers to 0 (connect immediately) whenever control line status exchange is enabled for at least one of the signal pairs (RTS/CTS or DTR/DSR). "Connect immediately" option makes the DS establish a data connection with its destination as soon as this DS is switched on. Connection is then maintained at all times and control line status exchange will also work at all times, even when no data is transmitted across the serial connection.

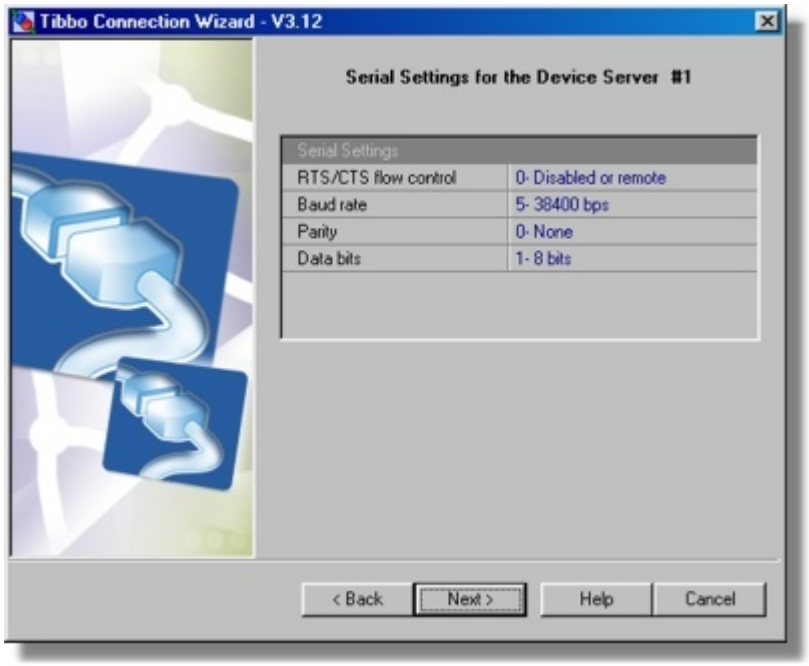
Notice, that the decision on the **Connection Mode** of both Device Servers has

already been taken before and the *Wizard* might have already decided that "connect immediately" option must be enabled on one of the Device Servers (this is needed for "reverse" connections). If this is so then the "connect immediately" option will stay, no matter whether the control line status exchange is enabled on this step or not.

If the *Wizard* haven't previously decided that the "connect immediately" option is necessary for one of the Device Servers and you have opted to enable control line status exchange for at least one signal pair (RTS/CTS or DTR/DSR) then the *Wizard* will decide that the **Connection Mode (CM) setting** of the DS #1 is to be 0 (connect immediately).

Parameters for the Serial Port of the DS #1

On this step you select communication parameters for the serial port of the DS #1.

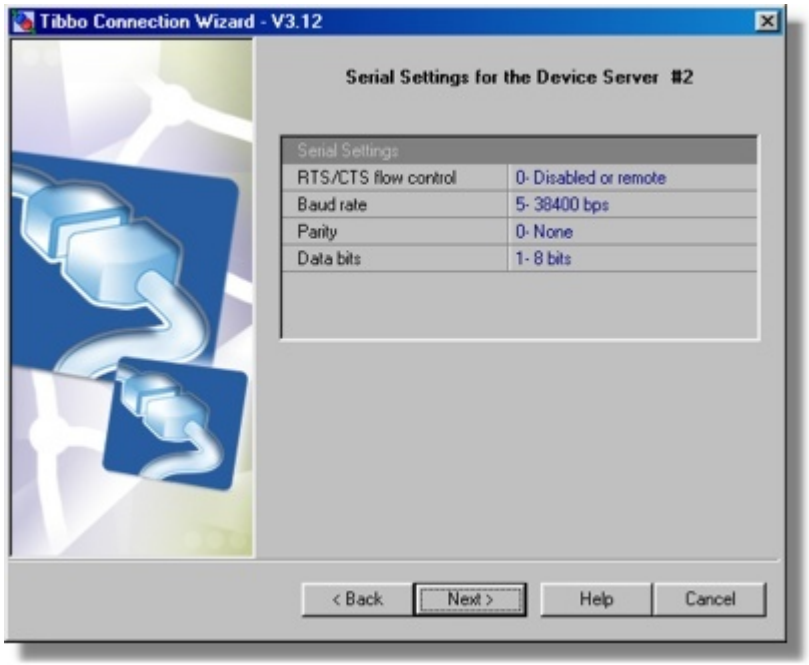


Unlike with the VSP-to-DS link, the serial port parameters of the DS, involved in the DS-to-DS link, cannot be adjusted via on-the-fly commands. You have to define fixed parameters that will be used by this DS at all times.

If you have previously enabled status exchange for the RTS/CTS signal pair then the RTS/CTS flow control option will be pre-set and fixed at *disabled or remote*. This is because the remote exchange and "local" flow control are mutually exclusive options.

Parameters for the Serial Port of the DS #2

On this step you select communication parameters for the serial port of the DS #2.

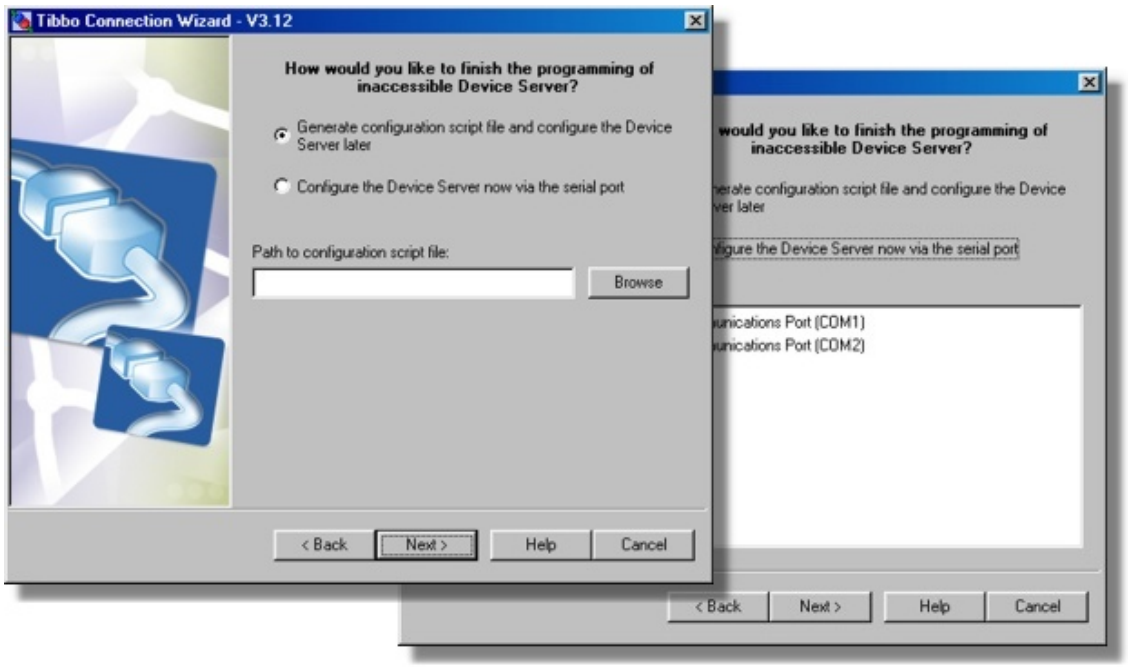


Unlike with the VSP-to-DS link, the serial port parameters of the DS, involved in the DS-to-DS link, cannot be adjusted via on-the-fly commands. You have to define fixed parameters that will be used by this DS at all times.

If you have previously enabled status exchange for the RTS/CTS signal pair then the RTS/CTS flow control option will be pre-set and fixed at *disabled or remote*. This is because the remote exchange and "local" flow control are mutually exclusive options.

Programming an Inaccessible DS

If this screen is shown then you have specified on the DS #2 step that the *Device Server is not accessible from this network segment*. Now you have to decide how you will setup this DS.



There are two choices:

- **Generate configuration script** (left screenshot). The script file can later be fed into the *Connection Wizard*, possibly running on a *different* PC (see finishing remote job). Choose this option if you cannot physically bring the DS to this PC for programming. Input the path and filename for the script file (left screenshot) and press *Next*.
- **Configure the DS through the serial port** (right screenshot). Choose this option if you can temporary bring the DS to this PC for programming. Connect the serial port of the DS to the unused COM port of your PC (with the WAS-1455 or similar cable), make sure the DS is powered up and press the setup button* (this will put the DS into the serial programming mode). Click *Next* to continue (the *Wizard* will read out current setting values of the DS and proceed to the next step).

Note:

Status LEDs of the DS are playing a *serial programming mode pattern* (shown on the left) when the serial port of the DS is in the serial programming mode (click here to see all available patterns).

** On EM100, EM120, EM200, EM203(A)- pull the MD line low for at least 100ms.*

Finishing a Remote Job

This job is selected by choosing "Finish remote job" from the list of available jobs. This feature allows you to finish the programming of the DS that was marked as inaccessible from the "programming" PC when running one of the three connection jobs.

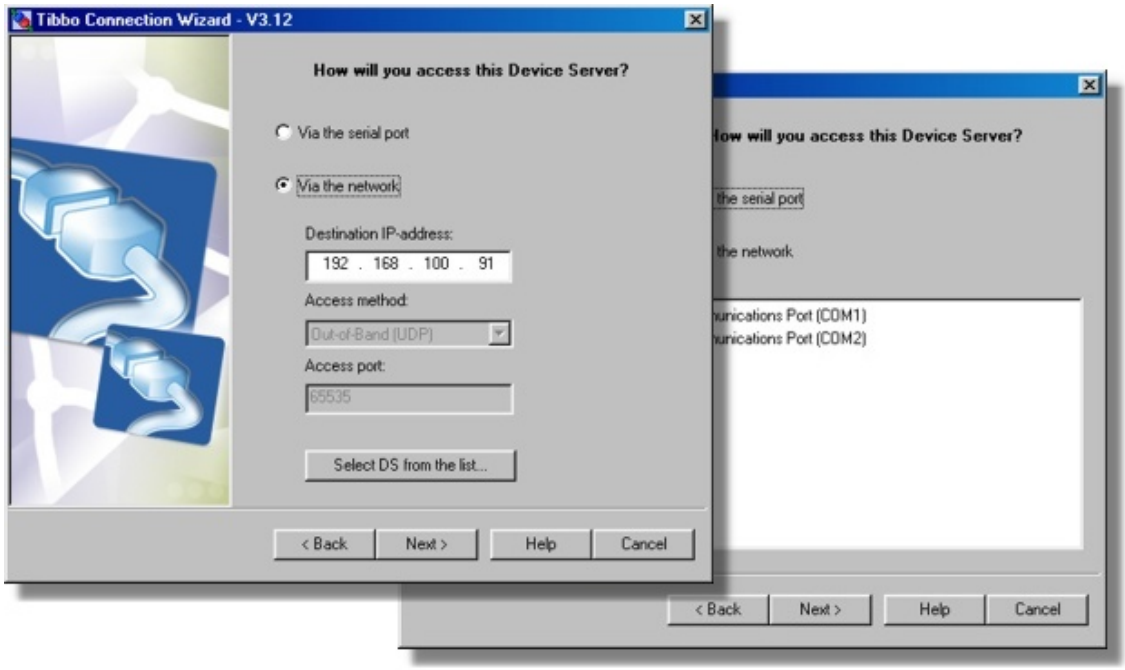
Inaccessibility of the DS from the "programming" PC is not an uncommon situation in wide-area networks (WANs). Such WANs are often not symmetrical i.e. only one side of the data link can "see" the other side. For example, in the VSP-to-DS link

the DS may be able to "see" (and connect to) the PC (*VSP*) but the PC (*VSP*) won't be able to "see" the DS. The *Wizard* easily deals with this kind of networks by correctly identifying which side of the link will be responsible for establishing data connections with the other side.

The problem, however, is that if the DS is not visible from the PC, then the *Wizard* cannot configure it through the network as well! The way out is either to program the DS via its serial port immediately or generate configuration script file that can be used later (you have been offered this choice on the programming inaccessible DS step* of the *Wizard*). "Later" programming of the DS is done by choosing Finish remote job on the Wizard job step of the *Wizard*.

Programming Method for the DS

On this screen you select how the *Wizard* will program the DS.



There are two options to choose from:

- **Through the serial port** (right screenshot). Choose this option if you can temporary bring the DS to this PC for programming. Connect the serial port of the DS to the unused COM port of your PC (with the WAS-1455 or similar cable), make sure the DS is powered up and press the setup button* (this will put the DS into the serial programming mode). Click *Next* to continue (the *Wizard* will read out current setting values of the DS and proceed to the next step).

Status LEDs of the DS are playing a *serial programming mode pattern* (shown on the left) when the serial port of the DS is in the serial programming mode (click here to see all available patterns).

- **Through a temporary network connection** (left screenshot). Choose this option if this PC resides on the same network segment with the DS or if you can temporarily connect the DS to the same network segment with this PC. In any case the DS and the PC must be on the same network segment.

Programming the DS through a temporary network connection

Since the DS has to be local the access options are disabled because they are only necessary when you are dealing with the remote DS.

The only editable option is the IP-address of the DS. You can input this IP-address manually or select the DS from the list:

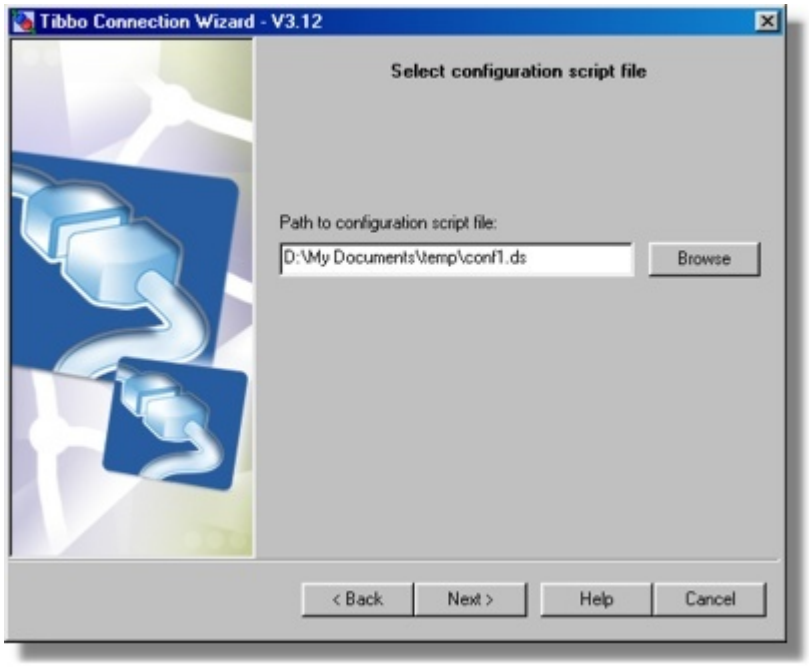
- Click *Select DS button* and choose the DS from the list. The *button* brings up the dialog, similar to the *main window* of the DS Manager. In fact, it is the *DS Manager*, with the following two differences:
  - There is an additional *Select button*. To select a particular DS single-click on the DS in the *device list* (in the auto-discoveryaccess mode) and press *Select*. Alternatively, you can double-click on the DS in the *device list*.
  - The address book and COM access mode are not available. The DS #1 must be local and so you must be able to find it in the auto-discovery mode.
- Click *Next*.

When you click *Next* the *Wizard* attempts to contact the target DS. If everything is OK the *Wizard* reads out all current setting values of this DS.

** On EM100, EM120, EM200, EM203(A)- pull the MD line low for at least 100ms.*

Configuration Script File

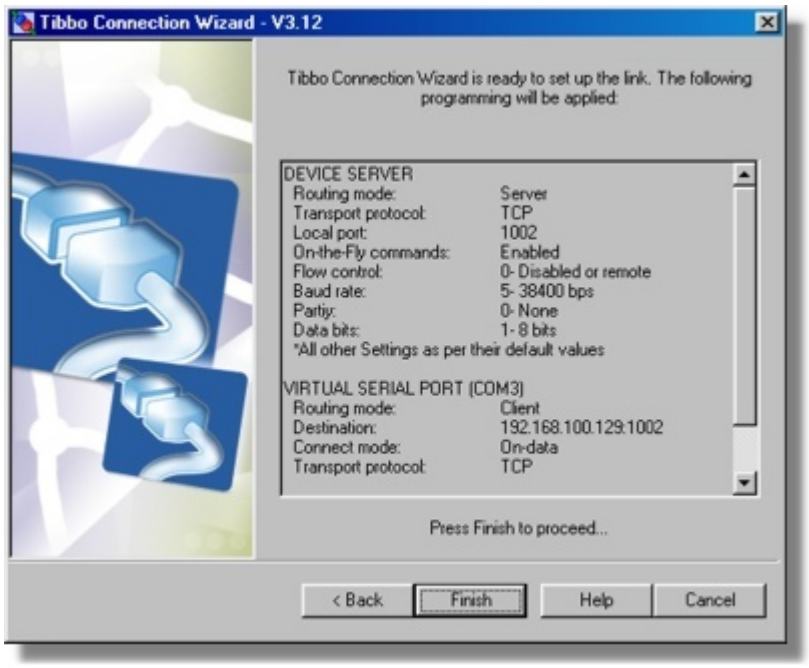
On this screen you choose the configuration script file that will be used to setup the DS.



Browse to a file of your choice and click *Next*.

Reviewing Setup Details

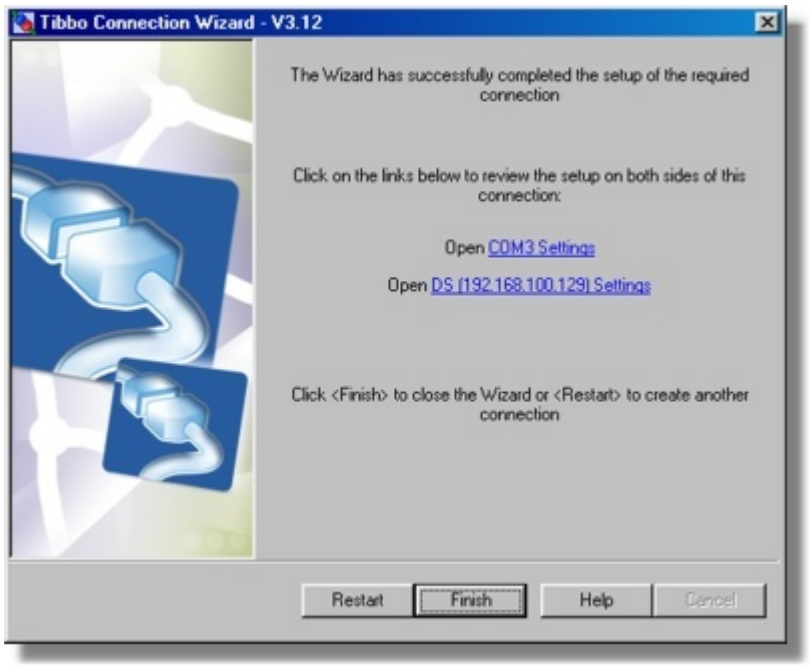
On this step you can review the setup this *Wizard* is about to apply to both sides of the connection. There is nothing to select or decide on here- the step is provided for your information only. Screenshot below shows the configuration of the *VSP* and the target DS for the VSP-to-DS link.



Once you click *Next* the *Wizard* starts configuring the link.

Final Screen

If you got to this screen then this means that the *Wizard* has successfully configured your connection. Clicking on the internet-style links in the middle of the screen opens configuration dialogs for the "participants" of this connection.



For example, if you've just finished the VSP-to-DS job then you will see two links as shown on the screenshot above. The first one opens the VSP properties dialog, the second one- DS settings dialog. This provides a comfortable way of reviewing the setup and making changes (if necessary) right from inside the *Wizard*.

Click *Finish* if you want to close the *Wizard*. Click *Restart* to run another *Wizard* job.

Warnings and Messages

This reference section contains additional information on the warning and error messages displayed by the *Connection Wizard*.

Inband Access for Local DS

Description

Message text: It makes no sense to use inband (TCP) access method for local Device Servers. Do you still wish the *Connection Wizard* to use it?

Details

This message was displayed because the *Connection Wizard* has determined that the DS you've specified is local (i.e. located on the same network segment as this PC) yet you have opted to program this DS using inband (TCP) access method.

The primary method of programming Device Servers is by using out-of-band (UDP) access method. This method always works and should be used for all local Device Servers. Inband programming is meant as an alternative method for remote Device Servers that cannot be accessed using out-of-band method.

MAC-->IP Mapping Advised

Description

Message text: It is recommended to use the MAC --> IP mapping option because this Device Server is running with DHCP enabled and its IP-address can change in the future. Do you still wish to reference this DS by its IP-address?

Details

MAC-->IP mapping option was created as a means to avoid communication problems with Device Servers that run with **DHCP (DH) setting** programmed to 1 (enabled). The IP-address of such Device Servers is configured by the DHCP server and there is no guarantee that this IP-address won't change in the future. When this happens the *VSP* won't be able to communicate with the DS since the target IP-address specified for this *VSP* will no longer belong to this DS.

The solution is to reference the DS by its MAC-address, which is done by enabling the MAC-->IP mapping option. More on the subject can be found here: additional info on accessing the DS.

If you choose to not enable the mapping the data connection will work, but only until the DHCP server assigns a different IP-address to the DS.

DS #1 Must be Local

Description

Message text: Device Server #1 must be local (located on the same network segment with this PC)

Details

This is a limitation imposed by the *Connection Wizard*. When you are creating a link between two Device Servers the *Wizard* requires that at least one of the Servers is located on the same network segment as the PC on which the *Wizard* is running. Another DS may be local or remote.

Non-zero Port Number Required

Description

Message text: Listening port number cannot be zero

Details

When you choose a port number for the listening port on the *VSP* side you can input any number in the 1-65535 range. Port 0 is a special number that instructs the PC to choose an actual port automatically. Since the port number of the listening port must be fixed and known you are not allowed to input "0".

Port is in Use

Description

Message text: This port is in use. Choose another port number

Details

A port cannot be used by several applications simultaneously. This message means that the port is already opened by another application (possibly, another *VSP*!). Select a different port number and try again.

VSP is Opened by Another Application

Description

Message text: This *VSP* is currently opened by another application. Close this application first to allow the *Wizard* to change *VSP* properties

Details

When the *VSP* is opened by another application its properties are locked and cannot be changed.

Different IP-address Required

Description

Message text: You cannot select the same Device Server twice

Details

This message is displayed in the cause of the DS-to-DS connection job when you select the same DS on both DS #1 and DS #2steps of the *Wizard*.

Unable to Find Setting Description File

Description

Message text: Unable to find serial settings definition file (wizserial.sdf)

Details

Wizserial.sdf contains the list of settings related to the serial port of the DS. During the installation this file is copied into the */SDF* folder inside the target installation folder of the *Device Server Toolkit (DST).* This message indicates that there is no such file in the */SDF* folder. This problem can be corrected by reinstalling the software.

Invalid IP-address

Description

Message text: You cannot select this Device Server because its IP-address is unreachable

Details

This message is displayed when the *Connection Wizard* has determined that local DS of your choice cannot be communicated with using normal IP-addressing.

As explained in broadcast access, a local Device Server can be accessed for programming even if its IP-address is not configured properly. Therefore, the *Connection Wizard* itself would be able to access and program any local Device Server, even if the IP-address of this DS is unreachable. Data communications with the DS, however, require the IP-address to be reachable. For example, the *VSP* won't be able to connect to the DS with a "wrong" IP-address. This is why the *Wizard* demands that you assign a proper IP-address to the DS before continuing with the connection job.

There are several reasons why the IP-address of the Device Server may (appear to) be unreachable. See troubleshooting (auto-discovery mode) for more information.

The DS is Not Local

Description

Message text: This Device Server is not local. Is this correct?

Details

This message means that the *Wizard* has verified if the target Device Server is located on the same network segment with this PC and has determined that the DS is not local, i.e. located on some other network segment and there is at least one router (firewall, bridge, etc.) between this PC and the DS.

Understanding whether the target DS is local or remote is important for several *Wizard* steps that will follow.

If you believe that the DS is actually local press *Retry* to make the *Wizard* re-check the location of the DS.

Avoid Data Characters with ASCII code FF

Description

Message text: You must make sure that your application won't send any characters with ASCII code 255 (FF).

Details

This message is displayed when you have specified inband access method for the target DS. Inband access method is usually utilized when out-of-band access is not possible (i.e. because of restrictions on the network). To allow future programming of the target DS the *Wizard* will preserve inband access in the DS and this means that the **Inband (IB) setting** of the DS will be kept at 1 (enabled).

When the **Inband (IB) setting** of the DS is at 1 (enabled) the DS can accept commands sent by the remote host across the data connection and ASCII character with code 255 (FF Hex) is used as an "escape" character that signifies the beginning of a command and two FF characters need to be sent to transmit one data FF character (see inband programming for details).

If your application sends any FF characters the DS will interpret this as a beginning of an inband command, which is obviously a problem as all subsequent data will

not be routed to the serial port of the DS. There are three solutions to this problem:

- Find a way to program the DS using out-of-band access method. In this case the *Wizard* will disable inband commands on the DS side and all FF characters will be interpreted by the DS in a "normal" way.
- Find a way to avoid sending ASCII characters with code FF between your application and the DS (your serial device).
- Alter how your application handles all FF characters in a way compatible with the inband mode. This means that:
  - Whenever your application needs to send ASCII character FF it actually sends two consecutive FF characters;
  - Whenever your application receives two FF characters it interprets this as a single data character with code FF.

Unable to Send a Broadcast

Description

Message text: Unable to send an auto-discovery broadcast because a local port from which this broadcast packet is supposed to be sent is currently in use by another program

---

Details

After you have selected a particular DS in the *Connection Wizard* the latter attempts to detect whether this DS is local or remote. This is done by sending an **Echo (X) command** as UDP broadcast. Only locally attached Device Servers will receive the broadcast, as broadcast packets cannot pass through the routers. Therefore, if the DS you have selected replies to the broadcast then this DS is local, otherwise it is remote.

This message means that the *Wizard* is unable to send the broadcast because local port from which the broadcast is supposed to be sent is currently opened by some other program.

This situation is extremely rare. If you encounter it please contact Tibbo for instructions on how to change the port from which the *Wizard* is sending the broadcasts! By default, the port number used for the purpose is 65534.

DST Revision History

This topic briefly describes *DST software* genesis from V3.56 (considered to be the "baseline" version for this revision history). This log is provided for your reference only; Tibbo is not obliged to elaborate on or explain detailed meaning of any item listed below.

Release V3.9.82 [the latest published and documented version]

- New design for DS Manager (tabs)
- VSPD architecture significantly changed for better emulation
- Added support for working with Winsock Proxy server
- Optimizations in flow of Connection Wizard.
- VSPD now registers with the serial PNP device enumerator of Windows. Now serial PNP devices connected to the DS are recognized correctly.

- It was possible to run the DS Manager, VSP Manager, and/or Connection Wizard at the same time (wrong!) on slower PCs. Now corrected. Only one of those components can run at any given time.
- In the Connection Wizard, it was possible to create several identical VSPs by rapidly clicking on FINISH button.
- DS firmware upgrade from the DS Manager could crash when performed in the background.

Release V3.6.6

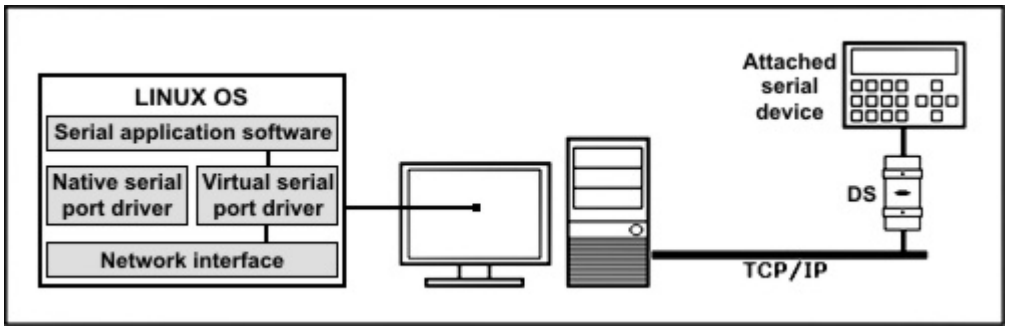
- New feature in VSP Manager. The Manager is now based on a *COM object*. You can interact with this object from within your application software. This provides a way for a full control programmatic control over creation, setup, and deletion of *Virtual Serial Ports*.
- New feature: additional options for serial (port control) lines CTS, DSR, and DCD (see serial lines tab).
- New feature: default serial property page (see default serial settings tab).
- Bug correction in VSPD. In previous *VSPD* versions some special characters (such as XOFF, ESC) when sent by a DOS-based program (running under *Windows NT/2000/XP*) were incorrectly handled by the *VSP*. The problem has now been corrected.
- Bug correction: *VSP* properties could not be opened from within *Windows Device Manager*. This problem has now been corrected. Under *Windows98* you will see default serial property page only. Under *Windows NT/2000/XP* you will see all *VSP* properties- just like with *VSP Manager*.
- Several other minor bugs and problems have been corrected.

Release V3.56 [baseline for this revision history]

Virtual Serial Port Driver for Linux (VSPDL)

Virtual Serial Port Driver for LINUX (*VSPDL*) powers **Virtual Serial Ports** (*VSPs*) that emulate "real" serial ports under *LINUX OS*.

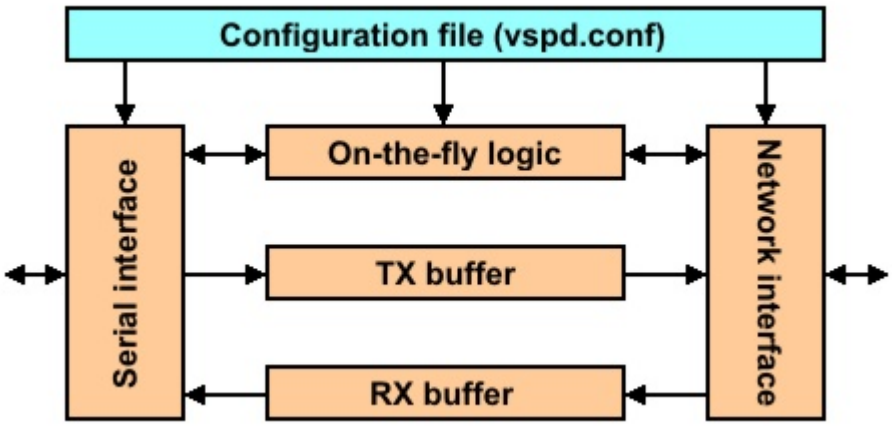
The *VSPDL* is an "engine" that powers **Virtual Serial Ports** (*VSPs*). To any LINUX application the *VSP* "looks and feels" just like a "normal" serial port. In reality, the *VSP* transparently reroutes all data sent by the application to the Tibbo Device Server ("DS") and the serial device behind it ("attached serial device"). Likewise, all the data sent by the serial device is received by the DS and routed to the *VSP*, which, in turn, passes this data to the application. Both the software application on the PC and the serial device communicate with each other just as if they were interconnected by a "normal" serial cable, without knowing that there is a network in between. This allows you to network-enable your existing serial system without changing the serial device itself or its PC software.



VSPDL operation is defined by configuration file named vspd.conf. This manual explains the features of the VSPDL through the structure and parameters of the vspd.conf.

How VSP Works

When designing the *VSPDL* we have attempted to emulate the work of a standard serial port as closely as possible. All system calls supported by the serial port driver were carefully ported into the *VSPDL* and the behavior of the *VSPs* closely mimics that of "real" serial ports.



Shown above is a *VSP* block diagram:

- **Serial interface** presents an application interface, compatible with the standard serial port driver. LINUX applications are not able to tell the difference between a *VSP* and a "real" COM port.
- **TX and RX buffers** are used to pass the data between the application and the DS.
- **Network interface** communicates (through the TCP/IP network) with the target DS. The *VSP* transparently establishes and accepts data connections with/from the DS as needed.
- **On-the-fly logic** (when enabled) is responsible for adjusting communications parameters of the serial port on the DS to the requirements of the application. For example, if the application wants the serial port to run at 19200 bps a special on-the-fly command will be sent to the DS telling it to change the serial port baudrate to 19200. Thus, the DS serial port functions just like the serial port and the PC!
- **Configuration file** (vspd.conf) define different aspects of *VSPDL* operation.

VSPDL Device Files

Two types of device files are created by the VSPDL:

- **/dev/vsps*n*** are virtual serial port device files (***n*** is the port number, so actual names look like this: */vsps0*, */vsps1*, etc.). These are the files that your "serial" application should open for input/output
- **/dev/vspd*n*** or **/dev/vsp/vspd*n*** (depending on your LINUX kernel version) are system files created by the VSPDL for its internal needs (***n*** is the port number, so actual names look like this: */vsps0*, */vsps1*, etc.). Do not attempt to access these files.

Installation

The VSPDL is supplied in two different versions:

- **TAR** archive.
- **RPM** package.

To install VSPDL you will need the following:

- **LINUX system** (kernel 2.2.x-2.6.x).
- **GNU C** compiler.
- **Linux kernel** headers/sources of the Linux kernel you are running now.
- **make-3.74** or above.
- **binutils-2.7.0** or above.
- **bash**
- **libc.so.6**
- **libexpat.so.0**
- **libgcc_s.so.1**
- **libm.so.6**
- **libstdc++.so.5**

Further, you will need either

- **gunzip-1.2.4** or above to process TAR archive, or
- **RPM tool** of your choice to process RPM package.

The Virtual Serial Port Driver for LINUX (VSPDL) distribution includes the following:

- **VSPModule** - Linux kernel module that creates Virtual Serial Ports (VSPs) on your system (in *lib/** of your distribution).
- **VSPDaemon** - interacts with the VSPModule, facilitates data transfers between the VSPs and the network (in *sbin/** of your distribution).
- **Documentation** - current documentation set in HTML format (in *man/** of your distribution).
- **Sample configuration files** (in *etc/** of your distribution).
- **VSPModule/VSPDaemon startup scripts** (in *bin/vspm*, *bin/vspd* of your

distribution).

To install from TAR archive:

- Login from console as a superuser (root).
- Copy the TAR archive into a temporary directory (i.e. */root/vsptmp*):
  - `#mkdir /root/vsptmp`
  - `#cp ./vspd-1.00.tar.gz /root/vsptmp`
  - `#cd /root/vsptmp`
- Unpack the TAR archive and go to the destination directory:
  - `#tar xvf ./vspd-1.00.tar.gz -z`
  - `#cd ./vspd-1.00-dist`
- Install the VSPDL into */usr/local/vspd* : `#./install.sh /usr/local/vspd` (the shell script will install all files automatically):
  - The script will check system dependencies for VSPModule compilation;
  - The script will then compile the VSPModule;
  - Finally, the script will install all files into their target location (*/usr/local/vspd*) and necessary startup shell scripts (vspd and vspm)- into */etc/rc.d/init.d/*.

To install from RPM archive:

- Login from console as a superuser (root).
- Go to the directory containing RPM archive and execute the following command: `#rpm --install ./vspd-X.YY-linux.i586.rpm` (you can also use any other tool of your choice to install RMP).
  - RPM subsystem will check package dependencies and let you know if something is missing;
  - If all dependencies check out successfully an installation directory will be created (normally, */usr/local/vspd-X.YY-dist*);
  - Next, the VSPModule will be compiled and installed into the package installation directory (normally, to */usr/local/vspd*);
  - Finally, the RPM subsystem will check the distribution installation directory and package installation directory to see if both were successfully created.

If the VSPDL installs successfully, you will have the following:

- In */etc/rc.d/init.d/vspd* and */etc/rc.d/init.d/vspm* - standard UNIX startup scripts for VSPDaemon and VSPModule.
- In */usr/local/vspd/bin/** - VSPDL services (startup and VSPTty).
- In */usr/local/vspd/etc/** - VSPDL configuration files.
- In */usr/local/vspd/lib/vspm.ko* - VSPModule.
- In */usr/local/vspd/sbin/vspd* - VSPDaemon.
- In */usr/local/vspd/man/** - VSPD documentation.

Warning regarding LATEST Linux KERNELS (2.6.x, x>= 13): If your system can't load VSPModule and shows a "module not found" error after VSPDL installation, you have to add a string into /lib/modules/<kernel-name>/modules.dep:
`/usr/local/vspd/lib/vspm.ko`
and loader will have possibility to load VSPDModule.

Controlling VSPDL Operation

Use the following commands to start the VSPDL:

- `#/etc/rc.d/init.d/vspm start` (must report **[OK]** if started successfully).
- `#/etc/rc.d/init.d/vspd start`.

As explained in how VSP works, the VSPLD configuration is stored in the configuration file vspd.conf. After the file is edited and changes saved you must "update" driver configuration by using the following command:

```
/etc/rc.d/init.d/vspd reloadconf
```

This will make the VSPDL re-read the file and start using new configuration. Configuration can be updated even when the VSPDL is working i.e. some VSPs are opened and in use.

VSPDL Configuration File (vspd.conf)

Functioning parameters of the VSPDL, and also configurations of individual VSPs are all defined by a single configuration file *vspd.conf*. The file usually resides at */usr/local/vsp/etc/*. You must be logged in as a superuser to be able to edit this file. Editing can be done with any text editor of your choice.

After the file is edited and changes saved you must "update" driver configuration by using the following command:

```
/etc/rc.d/init.d/vspd reloadconf
```

This will make the VSPDL re-read the file and start using new configuration. Configuration can be updated even when the VSPDL is working i.e. some VSPs are opened and in use.

Vspd.conf has the following structure* (see a sample file in the next topic for actual configuration example):

```
<!-- **** VIRTUAL SERIAL PORT DRIVER FOR LINUX (VSPDL) CONFIGURATION FILE **** -->
<vspdconfig>

  <!-- ==================== GENERAL CONFIGURATION ==================== -->

    <!-- Root directory for the daemon -->
    <root_dir="directory"/>

    <!-- Path and prefix for device files -->
    <devprefix value="value"/>

    <!-- Host to bind by default -->
    <bind_host="IP_address"/>

    <!-- Timeout for basic I/O operations -->
    <timeout [exec="exec(100)"] [priority="priority(10)"]/>

    <!-- VSPdaemon event logging configuration -->
    [<log type="syslog|pipe|file" level="EMR|ALR|CRT|ERR|WRN|NTC|INF|DBG" path="path
```

```
"/>]
    ...
    ...
    [<log type="syslog|pipe|file" level="EMR|ALR|CRT|ERR|WRN|NTC|INF|DBG" path="path
"/>]

  <!-- ======= CONFIGURATION OF INDIVIDUAL VIRTUAL SERIAL PORTS (VSPs) ======= -->

    <!-- -------------------- VSP0 CONFIGURATION -------------------- -->

    <!-- VSP number -->
    <vsp num="0">

        <!-- HOST AND PORT TO BIND -->
        <bind host="[IP_address]" port="port"/>

        <!-- Connection parameters -->
        <connection [rmode="client|server|server/client"] [proto="udp|tcp"] [conmode="
ondata|immediately"]
        [timeout="timeout(5)"] [onthefly="outofband|inband|disabled"] [clogin="pwd"] [
dlogin="pwd"] />

        <!-- Destination device parameters -->
        <destination ip="IP_address"|mac="MAC_address" [port="port(1001)"] [cport="c
port(65535)"] />

        <!-- Outbound packet generation options -->
        <packets [maxlen="len(255)"] [maxdelay="delay(0)"] [starton="any|char"] [
startchar="hex"] [stopchar="hex"] />

        <!-- Event logging configuration for this VSP -->
        [<log type="syslog|pipe|file" level="EMR|ALR|CRT|ERR|WRN|NTC|INF|DBG" path="
path"/>]
        ...
        ...
        [<log type="syslog|pipe|file" level="EMR|ALR|CRT|ERR|WRN|NTC|INF|DBG" path="
path"/>]

        <!-- Data dump section -->
        [<dump port="no|yes" path="path"/>]

    </vsp>
    <!-- -------------------- END OF VSP0 CONFIGURATION -------------------- -->


    <!-- -------------------- VSP1 CONFIGURATION -------------------- -->
        <vsp num="1">
        ...
        ...
        </vsp>
    <!-- -------------------- END OF VSP1 CONFIGURATION -------------------- -->

</vspdconfig>
<!-- **** END OF VIRTUAL SERIAL PORT DRIVER FOR LINUX (VSPDL) CONFIGURATION FILE ****
-->
```

** Some comments found in the actual vspd.conf that comes with the driver were omitted. Default values for optional parameters are shown in **purple***

Sample Configuration File

Here is a sample configuration file (some optional parameters are at default values and were included for clarity):

```
<!-- **** VIRTUAL SERIAL PORT DRIVER FOR LINUX (VSPDL) CONFIGURATION FILE **** -->
<vspdconfig>

  <!-- ==================== GENERAL CONFIGURATION ==================== -->

    <!-- Root directory for the daemon -->
    <root dir="/usr/local/vsp"/>
```

```
    <!-- Path and prefix for device files -->
    <devprefix value="/dev/vsp/vspd"/>

    <!-- Host to bind by default -->
    <bind host="192.168.100.40"/>

    <!-- Timeout for basic I/O operations -->
    <timeout exec="100" priority="10"/>

    <!-- VSPdaemon event logging configuration -->
    <log type="file" level="EMR" path="var/vspd.log"/>
    <log type="file" level="ALR" path="var/vspd.log"/>
    <log type="file" level="CRT" path="var/vspd.log"/>
    <log type="file" level="ERR" path="var/vspd.log"/>
    <log type="file" level="WRN" path="var/vspd.log"/>
    <log type="file" level="NTC" path="var/vspd.log"/>
    <log type="file" level="INF" path="var/vspd.log"/>


  <!-- ======= CONFIGURATION OF INDIVIDUAL VIRTUAL SERIAL PORTS (VSPs) ======= -->

    <!-- ==================== VSP0 CONFIGURATION ==================== -->

    <!-- VSP number -->
    <vsp num="0">

        <!-- HOST AND PORT TO BIND -->
        <bind host="192.168.100.40" port="3500"/>

        <!-- Connection parameters -->
        <connection rmode="server/client" proto="tcp" conmode="ondata" timeout="2"
onthefly="outofband" clogin="pwd" dlogin="pwd"/>

        <!-- Destination device parameters -->
        <destination ip="192.168.100.90" port="1001" cport="32767"/>

        <!-- Outbound packet generation options -->
        <packets maxlen="255" maxdelay="0" starton="any"/>

        <!-- Event logging configuration for this VSP -->
         <log type="file" level="EMR" path="var/vspd.log"/>
         <log type="file" level="ALR" path="var/vspd.log"/>
         <log type="file" level="CRT" path="var/vspd.log"/>
         <log type="file" level="ERR" path="var/vspd.log"/>
         <log type="file" level="WRN" path="var/vspd.log"/>
         <log type="file" level="NTC" path="var/vspd.log"/>
         <log type="file" level="INF" path="var/vspd.log"/>

    </vsp>
    <!-- ==================== END OF VSP0 CONFIGURATION ==================== -->

</vspdconfig>
<!-- **** END OF VIRTUAL SERIAL PORT DRIVER FOR LINUX (VSPDL) CONFIGURATION FILE ****
-->
```

General VSPDL Configuration

General configuration options define the configuration of the VSPDL as a whole. The format for the VSPDL configuration header is as follows (**purple** marks default values that will be used if corresponding parameter is omitted):

```
<!-- **** VIRTUAL SERIAL PORT DRIVER FOR LINUX (VSPDL) CONFIGURATION FILE **** -->
<vspdconfig>

  <!-- ==================== GENERAL CONFIGURATION ==================== -->

    <!-- Root directory for the daemon -->
    <root dir="directory"/>

    <!-- Path and prefix for device files -->
    <devprefix value="value"/>
```

```
    <!-- Host to bind by default -->
    <bind host="IP_address"/>

    <!-- Timeout for basic I/O operations -->
    <timeout [exec="exec(100)"] [priority="priority(10)"]/>

    <!-- VSPdaemon event logging configuration -->
    [<log type="syslog|pipe|file" level="EMR|ALR|CRT|ERR|WRN|NTC|INF|DBG" path="path
"/>]
    ...
    ...
    [<log type="syslog|pipe|file" level="EMR|ALR|CRT|ERR|WRN|NTC|INF|DBG" path="path
"/>]

  <!-- ======= CONFIGURATION OF INDIVIDUAL VIRTUAL SERIAL PORTS (VSPs) ======= -->


</vspdconfig>
<!-- **** END OF VIRTUAL SERIAL PORT DRIVER FOR LINUX (VSPDL) CONFIGURATION FILE ****
-->
```

Root Directory for the Daemon (Yroot> Section)

<Root> section defines root directory for the VSPdaemon. We do not recommend you changing the default value of dir parameter unless necessary (and you know what you are doing). The section has the following syntax:

```
<root dir="/usr/local/vsp/"/>
```

Path and Prefix for Device Files (Ydevprefix> Section)

<Devprefix> section defines the prefix for the device name used by the daemon. For example, if prefix is "/dev/vsp/vspd" then the daemon will access device 0 as /dev/vsp/vspF0. We do not recommend you changing the value parameter unless necessary (and you know what you are doing). The section has the following syntax:

```
<devprefix value="/dev/vsp/vspd"/>
```

Host to Bind By Default (Ybind> Section)

<Bind> section selects the IP-address to which the VSPDL will "bind". LINUX servers often have several network interfaces (with each interface having its own IP-address) and it is necessary to specify which interface the VSPs will work on. All VSPs will bind to the same IP-address, except for those whose configuration specifies a different bind host.

The section has the following syntax:

```
<bind host="IP_address"/>
```

Timeout for Basic I/O Operations (Ytimeout> Section)

We do not recommend you changing this section unless absolutely necessary (and you know what you are doing). The section has the following syntax (**purple** marks default values that will be used if corresponding parameter is omitted):

```
<timeout exec="exec(100)" priority="priority(10)"/>
```

Exec parameter specifies a sleep timeout in milliseconds, which is performed after all VSPs has been polled. This value can be decreased if you have many VSPs and experience problems with VSPDL performance. Setting **exec** to 0 causes your PC to poll the ports at maximum possible speed. This leads to high CPU utilization by the VSPLD but may be necessary if you want to achive best possible performance in a system with a large number of virtual ports.

Priority parameter specifies priority level for the VSPdaemon process.

VSPdaemon Event Logging Configuration (Ylog> Section)

<Log> section describes the logging options for events generated by the VSPdaemon. Notice, that <log> section can be included into the VSP configuration itself (i.e. inside `<VSP> .... </VSP>`) thus defining individual logging options for this particular VSP.

The section has the following syntax:

```
[<log [type="syslog|pipe|file"] level="EMR|ALR|ERR|WRN|NTC|INF|DBG" [
path="path"]/>]
```

Type parameter defines the output destination for the event. If **type** is set to "syslog", all events are logged to a syslog facility "USER". If omitted, this parameter defaults to "syslog".

Level parameter defines the type of event for which this entry is created. Standard level names are used.

Path parameter is only needed when type="pipe" or "file" and defines the file to which events will be saved or a program that will accept events into its STDIN.

Because each entry in the format shown above specifies logging only for one type of events (**level**) the configuration file can have several entries, for example:

```
<log type="file" level="EMR" path="var/dev.0.log"/>
<log type="file" level="ALR" path="var/dev.0.log"/>
<log type="file" level="CRT" path="var/dev.0.log"/>
<log type="file" level="ERR" path="var/dev.0.log"/>
<log type="file" level="WRN" path="var/dev.0.log"/>
<log type="file" level="NTC" path="var/dev.0.log"/>
<log type="file" level="INF" path="var/dev.0.log"/>
```

In general, we recommend you to keep the list of logged events like the one shown above. Logging "DBG" events is not necessary.

VSP Configuration (Yvsp> Section)

VSP configuration options define individual functioning parameters for each VSP. Each VSP has to be described in the following format (**purple** marks default values that will be used if corresponding parameter is omitted):

```
<!-- -------------------- VSP0 CONFIGURATION -------------------- -->

<!-- VSP number -->
<vsp num="0">

    <!-- HOST AND PORT TO BIND -->
    <bind host="ip_address" port="port"/>

    <!-- Connection parameters -->
    <connection [rmode="client|server|server/client"] [proto="udp|tcp"] [conmode="
```

```
ondata|immediately"]
        [timeout="timeout(5)"] [onthefly="outofband|inband|disabled"] [clogin="pwd"] [
dlogin="pwd"] />

        <!-- Destination device parameters -->
        <destination ip="IP_address"|mac="MAC_address" [port="port(1001)"] [cport="c
port(65535)"] />

        <!-- Outbound packet generation options -->
        <packets [maxlen="len(255)"] [maxdelay="delay(0)"] [starton="any|char"] [
startchar="hex"] [stopchar="hex"] />

        <!-- Event logging configuration for this VSP -->
        [<log type="syslog|pipe|file" level="EMR|ALR|CRT|ERR|WRN|NTC|INF|DBG" path="
path"/>]
        ...
        ...
        [<log type="syslog|pipe|file" level="EMR|ALR|CRT|ERR|WRN|NTC|INF|DBG" path="
path"/>]

        <!-- Data dump section -->
        [<dump port="no|yes" path="path"/>]

    </vsp>
    <!-- -------------------- END OF VSP0 CONFIGURATION -------------------- -->
```

Click on the links above to jump to parameter group and individual parameter description topics.

VSP Number (Yvsp> Section)

Syntax:	<**vsp num**="*num*"]/>, where *num* is the port number in the 0-127 range
Default value (if omitted):	this parameter is mandatory and must be present
Relevance conditions:	**---**
See also:	VSP name selection

Num (VSP number) parameter defines the number for the VSP. The number can be in the 0-127 range.

This parameter is similar to the port name of the *VSP* for *Windows*.

Host and Port to Bind for the VSP (Ybind> Section)

<Bind> section defines the IP-address and the port number of the VSP on its side. The section has the following syntax (**purple** marks default values that will be used if corresponding parameter is omitted):

```
<bind host="[IP_address]" port="port"/>
```

Click on the links above to jump to individual parameter description topics.

Bind Host (Yhost> Parameter)

Syntax:	<**bind** ... **host**="[*IP_address*]"/>
Default value (if omitted):	Parameter is mandatory; if the value is NULL (i.e. host="") then the value of "general" bind

parameter will be used

Relevance conditions: ---

See also:

Host parameter selects the IP-address to which this VSP will "bind". LINUX servers often have several network interfaces (with each interface having its own IP-address) and it may be necessary to specify which interface this particular VSP will work on.

This parameter must not be omitted. Defining Host="" makes the VSP bind to the "general" bind IP-address.

Bind Port (Yport> Parameter)

Syntax: <**bind** ... **port**="[*port*]"/>

Default value (if omitted): Parameter is mandatory

Relevance conditions: rmode="server" or "server/client"

See also: **Port Number (PN) setting** (DS), listening port (*VSP* for *Windows*)

Port parameter specifies the port number on the PC that will be associated with this VSP. This port serves as a "listening port" for incoming data connections, when such connections are allowed by the routing mode of the VSP. Additionally, in case of UDP transport protocol and server/client routing mode, the port is also used to send outgoing UDP datagrams. For more details see additional info on UDP and TCP connections.

This parameter must be specified and cannot be omitted.

Port parameter works like the **Port Number (PN) setting** of the DS and listening portoption of the *VSP* for *Windows*.

Connection Parameters (Yconnection> Section)

<Connection> section specifies several different parameters that define how VSP will communicate and interact with the DS. The section has the following syntax (**purple** marks default values that will be used if corresponding parameter is omitted):

<**connection** [**rmode**="***client***|*server*|*server/client*"] [**proto**="***udp***|*tcp*"] [**conmode**="***ondata***|*immediately*"] [**timeout**="*tout(****5****)*"] [**onthefly**="***outofband***|*inband*|*disabled*"] [**clogin**="*pwd*"] [**dlogin**="*pwd*"]/>

Click on the links above to jump to individual parameter description topics.

Routing Mode (Yconnection rmode>)

Syntax: <connection ... [rmode="*server|client|server/client*"]/>

Default value (if omitted): "server"

Relevance conditions: ---

See also: **Routing Mode (RM) setting** (DS), Routing Mode option (*VSP* for *Windows*)

Rmode (routing mode) parameter defines whether the *VSP* will accept incoming connections (passive opens) and/or establish outgoing connections (perform active opens).

The following choices are available:

"server" (default) Only incoming connections are accepted, the *VSP* never attempts to establish an outgoing connection to the DS. There is no restriction on which DS can connect to the *VSP*- connection from any IP-address will be accepted as long as the DS is connecting to the correct bind port using correct transport protocol.

"client" Only outgoing connections are allowed, the *VSP* rejects all incoming connections.

"server/client" Both incoming and outgoing connections are allowed. Outgoing connections are established with the destination, specified in the destination section of the configuration file. Exactly when the *VSP* attempts to establish an outgoing connection is defined by the selected connection mode. Incoming connections are accepted from any IP-address, just like with the server routing mode.

Rmode parameter works like the **Routing Mode (RM) setting** on the DS and the Routing Mode option of the *VSP* for *Windows*.

Transport Protocol (Yconnection proto>)

Syntax: <connection ... [proto="*udp|tcp*"]/>

Default value (if omitted): "udp"

Relevance conditions: ---

See also: **Transport Protocol (TP) setting** (DS), transport protocol (*VSP* for *Windows*)

Proto (transport protocol) selects which communications protocol- TCP/IP or UDP/IP will be used by the *VSP* for data communications with the DS. Omitting the parameter selects the UDP/IP protocol.

For the data connection with the DS to work the same transport protocol must be selected on the DS side- see the **Transport Protocol (TP) setting**. This parameter has the same function as the transport protocol option of the *VSP* for *Windows*.

Unless you have a specific reason why the UDP should be used (very rare!) we recommend you to stick to the TCP/IP. The TCP/IP is a "reliable delivery" protocol that makes sure that no data is lost "in transit" between the *VSP* and the DS. The

UDP, on the contrary, does not guarantee data delivery.

Some considerations and additional info on the TCP and UDP implementation in the *VSP* can be found in the next topic.

UDP data connections

The notion of data connection is native to TCP/IP since this is a connection-based protocol. UDP/IP, however, is a connection-less protocol in which all packets (UDP datagrams) are independent from each other. How, then, the term "data connection" applies to the UDP transport protocol?

With UDP transport protocol true data connections (in the "TCP sense" of this term) are not possible (hence, parenthesis around the word "connection"). The *VSP*, however, attempts to mimic the behavior of TCP data connection whenever possible. Follows is the detailed description of UDP "connections" and their similarities and differences with TCP connections.

Incoming "connections"*. There is no connection establishment phase in UDP so an incoming UDP "connection" is considered to be "established" when the first UDP packet is received by the *VSP* (on the bind port). Similarity with TCP is in that after having received the packet from the DS the *VSP* knows who to send its own UDP packets to.

Outgoing "connections".** The *VSP* establishes outgoing UDP connection by sending a UDP datagram to the targeted destination. If there is a data that needs to be transmitted the *VSP* sends the first UDP datagram with (a part of) this data. If there is no immediate data that needs to be transmitted to the DS the *VSP* sends the first UDP datagram of zero length (this happens when the connection mode is set to "immediately"). The purpose of this is to let the other side know the IP-address of the *VSP* (PC it is running on), as well as the data port currently used by the *VSP*.

Data transmission and destination switchover. Once the "connection" is established the *VSP* and the DS exchange the data using UDP datagrams. The difference with TCP is that if another DS sends a datagram to the *VSP*, then the *VSP* will interpret this as a new incoming connection*, forget about the first DS and start sending its own UDP datagrams to the second one. In other words, the *VSP* will always communicate with the "most recent sender". Such behavior is not possible in TCP, in which a third party cannot interfere with an existing connection.

"Connection" termination. There is no connection termination phase in UDP so *VSP* "terminates" its UDP connections by forgetting about them and the only event that can trigger UDP "connection" termination (except for the closing of the *VSP*) is connection timeout.

Local port used by the *VSP* depends on the selected routing mode:

- **In the server routing mode** the *VSP* sends the UDP datagrams from an "automatic" port selected by the OS;
- **In the server/client and client routing modes** the *VSP* sends and receives UDP datagrams through the port, defined by the bind port parameter.

TCP data connections

Only one connection at a time. TCP protocol stack on the PC is capable of supporting thousands of concurrent TCP connections but the *VSPDL* strictly enforces that only a single TCP connection exists for each *VSP* at any time. This is because the serial port is not a shared media and allowing, say, two Device Servers to connect to the same *VSP* would have created a data chaos***. Allowing

only a single connection at a time follows a "serial port culture" of "one serial port-one application"! If the *VSP* is already engaged in a data connection with the DS and another DS attempts to establish a connection to this *VSP* then this DS will be rejected.

Separate ports for outgoing and incoming connections. The *VSP* establishes its outgoing connections** from an "automatic" port selected by the OS. Each time such outgoing connection is established the source port number on the *VSP* side will be different. The *VSP* accepts incoming connections* on a fixed listening port, whose number is defined by the bind port parameter. When the incoming connections are not allowed the bind port is closed.

** Assuming that incoming connections are allowed (i.e. the routing mode is either "server", or "server/client").*

*** Assuming that outgoing connections are allowed (i.e. the routing mode is either "client", or "server/client").*

**** What if both Device Servers started sending the data to the VSP at the same time? Then the PC application using the VSP would have received a mix of data consisting of an input from both Device Servers! And if two Device Servers were connected to the same VSP and the PC application needed to send out the data then which DS of the two the VSP would have to send this data to?*

Connection Mode (Yconnection conmode>)

Syntax:	<**connection** ... [**conmode**="*ondata*|*immediately*"]/>
Default value (if omitted):	"ondata"
Relevance conditions:	rmode= "client" or "server/client"
See also:	**Connection Mode (CM) setting** (DS), connection mode (*VSP* for *Windows*)

Conmode (connection mode) parameter defines when the *VSP* attempts to establish an outgoing connection to the destination, specified in the destination section of the configuration file.

***Connection mode drop-down box* provides two choices:**

"ondata" (default)	The *VSP* attempts to establish an outgoing connection when the first "serial" data (since the *VSP* was opened or previous connection was closed/aborted) is sent by the PC application into the *VSP*.
"immediately"	The *VSP* attempts to establish an outgoing connection right after it is opened by the application. The *VSP* also tries to make this connection "persistent". If the connection is aborted by the DS, the *VSP* will (attempt to) re-establish it immediately. Connection timeout still works in this mode: when the current connection times out the *VSP* aborts it and immediately establishes a new connection. Such behavior "auto-repairs" hanged connections.

Connection mode option is irrelevant when the routing mode is "server", since in

this mode outgoing connections are not allowed in principle.

Connection mode option works like the **Connection Mode (CM) setting** of the DS (modes 0 and 1 only) and connection mode option of the *VSP* for *Windows*.

Connection Timeout (Yconnection timeout>)

Syntax:	<**connection** ... [**timeout**="*tout*"]/>, where *tout*- connection timeout in minutes ("0" means "never timeout"); maximum value is 65535
Default value (if omitted):	5 minutes
Relevance conditions:	**---**
See also:	**Connection Timeout (CT) setting** (DS), connection timeout (*VSP* for *Windows*)

Timeout (connection timeout) parameter sets the timeout (in minutes) for the data connections between the *VSP* and the DS. If no data is transmitted across the data connection (TCP or UDP) for a specified number of minutes the *VSP* aborts the connection.

Setting connection timeout parameter to 0 disables the feature and connection never times out. If omitted, this parameter defaults to 5 minutes.

Notice, that this feature will work even when the connection mode is set to "immediately". When the timeout comes the *VSP* closes the connection first and immediately reopens it (timeout counter is reloaded). This provides additional reliability since hanged connections are automatically "repaired".

Connection timeout parameter works like the **Connection Timeout (CT) setting** of the DS and the connection timeout option of the *VSP* for *Windows*.

On-the-fly Commands (Yconnection onthefly>)

Syntax:	<**connection** ... [**onthefly**="*outofband*|*inband*|*disabled*"]/>
Default value (if omitted):	"outofband"
Relevance conditions:	**---**
See also:	**Connection Timeout (CT) setting** (DS), connection timeout (*VSP* for *Windows*)

On-the-fly commands are used to change the serial port configuration of the DS as needed (i.e. "on the fly"). Serial port configuration made through the on-the-fly commands overrides the permanent one, defined by the serial port settings of the DS. The difference between the changes made using on-the-fly commands and changes made through altering DS settings is that, unlike serial settings, on-the-fly commands have immediate effect and do not require the DS to be rebooted in order for the new values to be recognized.

With on-the-fly commands enabled, the serial port of the DS is always setup as required by the PC application that communicates with this DS through the *VSP*. When the PC application opens the *VSP* (or some communications parameters are changed) the application informs the *VSP* about required changes and the *VSP* relates this information to the DS by sending on-the-fly commands.

Additionally, on-the-fly commands are used by the *VSP* to control the RTS and DTR

outputs of the DS serial port. The status of the CTS and DSR input of the DS serial port can be passed to the *VSP too*- this is done using so-called "notification messages".

On-the-fly commands drop-down box provides three choices:

"outofband"	On-the-fly commands are enabled and sent in the form of out-of-band (UDP) commands. **On-the-fly commands (RC) setting** of the DS must be programmed to 1 (enabled) for the out-of-band on-the-fly commands to be accepted.
"inband"	On-the-fly commands are enabled and sent in the form of inband (TCP) commands. **On-the-fly commands (RC) setting** of the DS must be programmed to 1 (enabled) for the on-the-fly commands to be accepted. Additionally, there are some other programming steps that must be performed before the DS will recognize inband commands- see preparing the DS for inband access.
"disabled"	On-the-fly commands are not sent at all, so the serial port of the DS will use "permanent" serial port configuration defined by the serial port settings. In this mode it doesn't matter what serial port parameters are set in the PC software application- the DS will not be aware of them!

In general, we recommend you to keep on-the-fly commands enabled (unless there are some special reasons preventing you from doing so). Enabling on-the-fly commands keeps the serial port setup of the DS "in sync" with the requirements of the software application using the *VSP*.

As for choosing between out-of-band and inband modes, follow these recommendations:

Out-of-band commands work most of the time, especially when the PC (running *VSP*) and the DS are located on the same network segment*. Out-of-band commands may not work very well or not work at all for the remote Device Servers located behind the routers, firewalls, etc**. This is because:

- The routers are known to "drop" UDP datagrams (on which out-of-band on-the-fly commands are based) under heavy network traffic.
- UDP traffic is banned by the firewalls of many networks (hence, out-of-band on-the-fly commands cannot be used at all). If you want out-of-band on-the-fly commands to work then your network must allow UDP traffic to port 65535 or 32767!

If you encounter one of the above situations then you should use inband on-the-fly commands or not use on-the-fly commands at all!

** The definition of the network segment implies that there are only network hubs (and no routers, bridges, firewalls, etc.) between the PC and all other devices on the segment.*

*** Here we touch on a very complicated subject. Modern routers offer a bewildering array of setup options. We will attempt to cover this in details in our upcoming white papers.*

Password for On-the-fly Commands (Yconnection clogin>)

Syntax:	<**connection** ... [**clogin**="*pwd*"]/>
If omitted:	Password will not be added to on-the-fly commands
Relevance conditions:	onthefly= "outofband" or "inband"
See also:	Out-of-band (UDP) programming, **Login (L) command**, **On-the-fly Password (OP) setting**, **Password (PW) setting**

Clogin (command login) parameter specifies the password that should be added to the on-the-fly commands. On-the-fly commands can be sent without prior login using **Login (L) command**. Instead, the password can be added to each command individually (see out-of-band (UDP) programming). **On-the-fly Password (OP) setting**on the DS must be programmed to 1 (enabled) to make the DS check the password. Password specified by the clogin parameter must match the one defined through the **Password (PW) setting** of the DS.

When clogin parameter is omitted all on-the-fly commands are sent without password. This parameter is only relevant when on-the-fly commands are not disabled.

Data Login (Yconnection dlogin>)

Syntax:	<**connection** ... [**dlogin**="*pwd*"]/>
If omitted:	Data login procedure will not be performed
Relevance conditions:	proto= "tcp"
See also:	Command-phase (TCP) programming, **Data Login (DL) setting**, **Password (PW) setting**

Dlogin (data login) parameter specifies whether a "data login" procedure will be performed when a TCP connection is established between the VSP and the DS. When the data login is required the VSP cannot start exchanging the data with the DS immediately, but needs to login with a valid password first. This provides a password protection for the data connection with the DS. on the DS side, data logins are enabled through the **Data Login (DL) setting**(additional info on data logins can be found in command-phase (TCP) programming).

Password specified by the clogin parameter must match the one defined through the **Password (PW) setting** of the DS.

Note, that omitting dlogin parameter is not the same as and specifying dlogin=""! In the first case the VSP will expect to be able to start the data exchange with the DS as soon as the connection is established. In the second case the VSP will still perform a data login procedure with NULL password.

Dlogin parameter is irrelevant when the transport protocol is "udp" because data logins are only supported for TCP connections.

Destination Device Parameters (Ydestination> section)

<Destination> section contains parameters that define the target DS to which the VSP will be establishing connections and also port numbers on the destination DS. The section has the following syntax (**purple** marks default values that will be used if corresponding parameter is omitted):

```
<connection ip="IP_address"|mac="MAC_address" [port="port(1001)"] [
cport="cport(65535)"]/>
```

Click on the links above to jump to individual parameter description topics.

Destination IP-address (Yip> parameter)

Syntax:	<**destination** ... **ip**="*IP-address*"/>
If omitted:	mac parameter will be used (at least one- ip or mac must be specified)
Relevance conditions:	rmode= "client" or "server/client"
See also:	**---**

Ip (IP-address) parameter specifies the IP-address of the target DS to which the VSP will be establishing outgoing connections.

This parameter is irrelevant when the routing mode is "server" because in this mode the VSP never establishes outgoing connections.

If both the ip and mac parameter are specified the ip parameter takes precedence and mac parameter is ignored. At least one of the two parameters has to be specified to define the target DS.

Destination MAC-address (Ymac> parameter)

Syntax:	<**destination** ... **mac**="*MAC-address*"/>
If omitted:	ip parameter will be used (at least one- ip or mac must be specified)
Relevance conditions:	rmode= "client" or "server/client"
See also:	**DHCP (DH) setting** (DS), single-destination mode (*VSP* for *Windows*)

Mac (MAC-address) parameter specifies the MAC-address of the target DS to which the VSP will be establishing outgoing connections. Before establishing a connection the VSP will perform a so-called "MAC->IP mapping" to find out which IP-address currently corresponds to a specified MAC-address. When the IP-address is "resolved" the VSP will proceed in a normal way i.e. attempt to establish a data connection with the DS at this IP-address.

Specifying MAC-address instead of the IP-address is useful in cases when the DS is running with DHCP enabled (see **DHCP (DH) setting**) and its IP-address can potentially change over time. If the target DS is specified by its MAC the VSP will still be able to find it. Note, that MAC->IP mapping only works for local* Device Servers, remote Device Servers have to be referenced by their IP-address.

If both the ip and mac parameter are specified the ip parameter takes precedence and mac parameter is ignored. At least one of the two parameters has to be specified to define the target DS.

This parameter is irrelevant when the routing mode is "server" because in this mode the VSP never establishes outgoing connections.

MAC->IP mapping works just like the one on the *VSP* for *Windows* (see single-destination mode).

** I.e. Device Servers located on the same network segment with the PC. The definition of the network segment implies that there are only network hubs (and no routers, bridges, firewalls, etc.) between the PC and all other devices on the segment.*

Destination Data Port (Yport> parameter)

Syntax:	<**destination** ... **port**="*port*"/>
Default value (if omitted):	"1001"
Relevance conditions:	rmode= "client" or "server/client"
See also:	**Port Number (PN) setting** (DS), single-destination mode (*VSP* for *Windows*)

Port (Destination data port) parameter specifies the port number on the DS to which the VSP will try to connect when establishing outgoing connections. If omitted, this parameter defaults to port 1001.

For the connection to work, the port number specified by this parameter must match the one defined by the **Port Number (PN) setting** of the DS.

This parameter is irrelevant when the routing mode is "server" because in this mode the VSP never establishes outgoing connections.

Port option works like the port parameter of the *VSP* for *Windows* (see single-destination mode).

Destination Command Port (Ycport> parameter)

Syntax:	<**destination** ... **cport**="*cport*"/>
Default value (if omitted):	"65535"
Relevance conditions:	onthefly= "outofband"
See also:	Out-of-band (UDP) programming

Cport (command port) parameter specifies the port number on the DS to which the VSP will send its on-the-fly commands (when commands are sent in the out-of-band mode). If omitted, this parameter defaults to port 65535.

Cport parameter allows you to set any command port number but the DS itself actually accepts out-of-band UDP commands on two fixed ports: 65535 and 32767 (either port can be used). For more information see out-of-band (UDP) programming.

Outbound Packet Generation Options (Ypackets> section)

The challenge of sending the data from the application to the DS is in deciding how to group this data into the network packets of reasonable size. Carrying too little data in each packet increases network load while sending packets with too much data slows down the delivery of this data to the DS. **<Packets>** section contains

parameters that define how the VSP will divide the "serial" data sent by the application into network packets.

To give you a better understanding of the options available in this section we will use the concept of data blocks. Many serial systems use some sort of communication packets (we will call them *data blocks* to avoid possible confusion with *network packets*). Since the data block is a basic unit of data transmission in such systems it is only logical to divide the data sent by the *VSP* into network packets basing on these data blocks.

Parameters in the <packets> section allow you to define conditions that "open" the data blocks (see starton and startchar), "close" the data block (stopchar), and "break" the data block into smaller chunks of data without closing the block (maxlen, maxdelay). The VSP ignores all the data received from the application past the end of the previous data block and before the beginning of the next data block (this means, the data is not even recorded into the TX buffer). The VSP sends out the data in the TX buffer as soon as the break or close condition is encountered.

Of course, not all systems rely on formatted data blocks, many just send unformatted "random" data. This data can be viewed as one endless data block that starts on the first character received from the application.

The section has the following syntax (**purple** marks default values that will be used if corresponding parameter is omitted):

```
<packets [maxlen="len(1024)"] [maxdelay="delay(0)"] [starton="any|char
"] [startchar="hex"] [stopchar="hex"]/>
```

Click on the links above to jump to individual parameter description topics.

Note, that default values of this section's parameters will work fine for most applications. Therefore, simply defining this section as `<packets/>` will (most probably) be OK.

Maximum Packet Length (Ymaxlen> parameter)

Syntax:	<packets ... **maxlen**="*len*"/>, where *len* has the range of 32...255 bytes
Default value (if omitted):	255 bytes
Relevance conditions:	**---**
See also:	**Maximum Packet Length (ML) setting**

Maxlen (maximum packet length) parameter breaks the large chunks of data sent by the application into smaller portions. Once the number of bytes in the TX buffer reaches the limit defined by the maxlen parameter the VSP sends all the data in the buffer to the DS. This does not close the data block and all subsequently received data is still considered to be a part of the same data block. If omitted this parameter defaults to 255 bytes.

For the UDP transport protocol this parameter directly defines the (maximum) packet size of each UDP packet sent by the VSP. In TCP there is no guarantee that individual packets won't occasionally carry larger chunks of data.

Maxlen option works like the **Maximum Packet Length (ML) setting** of the DS.

Maximum Intercharacter Delay (Ymaxdelay> parameter)

Syntax: <packets ... **maxdelay**="*delay*"/>, where *delay* is in milliseconds

Default value (if omitted): 0 milliseconds (i.e. "immediately")

Relevance conditions: **---**

See also: **Maximum Intercharacter Delay (MD) setting**

When the data block is already opened the *VSP* sends out all the data in its TX bufferwhen no new data is received from the application for a period of time defined by the **maxdelay** parameter (in milliseconds). This does not close the data block and all subsequently received data is still considered to be a part of the same data block. If omitted, this parameter defaults to 0 milliseconds, which means that the data is sent out as soon as it is received from the application.

By using this option you can combine the data that is sent by the application in "rapid succession" into a larger packets (but still not exceeding the maximum packet length).

Maxdelay option works like the **Maximum Intercharacter Delay (MD) setting** of the DS.

Start on Any Character (Ystarton> parameter)

Syntax: <packets ... [**starton**="*any*|*char*"]/>

Default value (if omitted): "any"

Relevance conditions: **---**

See also: **Start On Any Character (SA) setting**

Starton (start on character) parameter defines what will constitute the beginning of the data block:

"any" (default)	First character received from the application past the end of the previous data block will open a new data block. This option is very suitable both for unformatted and formatted data.
"char"	Only a specific character (defined by the start character parameter will open a new data block). All characters received from the application between the end of the previous data block and the start character will be ignored and not recorded into the VSP's TX buffer. This option is suitable for formatted data only.

This option works like the **Start On Any Character (SA) setting** of the DS.

Start Character (Ystartchar> parameter)

Syntax: <packets ... **startchar**="*hex*"/>, where *hex* is the ASCII code of the start character in the HEX form (i.e. "0D" for <CR>)

If omitted:	start character is not defined
Relevance conditions:	starton= "any"
See also:	**Start Character Code (S1) setting**

Startchar (start character code) parameter defines the ASCII code of the character that will open the data block in case the starton parameter is "char". Parameter is irrelevant if starton= "any".

This option works like the **Start Character Code (S1) setting**.

Stop Character (Ystopchar> parameter)

Syntax:	<**packets** ... **stopchar**="*hex*"/>, where *hex* is the ASCII code of the stop character in the HEX form (i.e. "0D" for <CR>)
If omitted:	stop character is not defined
Relevance conditions:	**---**
See also:	**Stop Character Code (E1) setting**

Stopchar (stop character code) parameter defines the ASCII code of the character that will close the data block. If the stop character is not defined the data block is never closed (although it is still subdivided using "breaking" parameters maxlen and maxdelay).

This option works like the **Stop Character Code (E1) setting**.

Event Logging Configuration for the VSP (Ylog> section)

<Log> section describes the logging options for events generated by the VSP. Notice, that the <log> section within the VSP section itself (i.e. inside <VSP> .... </VSP>) defines logging options for this particular VSP only. Logging options can also be defined for the VSPdaemon itself.

The section has the following syntax:

<**log** [**type**="*syslog*|*pipe*|*file*"] **level**="*EMR*|*ALR*|*ERR*|*WRN*|*NTC*|*INF*|*DBG*" [**path**="*path*"]/>

Type parameter defines the output destination for the event. If **type** is set to "syslog", all events are logged to a syslog facility "USER". If omitted, this parameter defaults to "syslog".

Level parameter defines the type of event for which this entry is created. Standard level names are used.

Path parameter is only needed when type="pipe" or "file" and defines the file to which events will be saved or a program that will accept events into its STDIN.

Because each entry in the format shown above specifies logging only for one type of events (**level**) the configuration file can have several entries, for example:

```
<log type="file" level="EMR" path="var/dev.0.log"/>
<log type="file" level="ALR" path="var/dev.0.log"/>
<log type="file" level="CRT" path="var/dev.0.log"/>
<log type="file" level="ERR" path="var/dev.0.log"/>
```

```
<log type="file" level="WRN" path="var/dev.0.log"/>
<log type="file" level="NTC" path="var/dev.0.log"/>
<log type="file" level="INF" path="var/dev.0.log"/>
```

In general, we recommend you to keep the list of logged events like the one shown above. Logging "DBG" events is not necessary.

Data Dump (Ydump> section)

<Dump> section offers an option of capturing the data that flows between the VSP and the DS. The section has the following syntax:

```
[<dump port="no|yes" path="path"/>]
```

Port parameter defines whether the data sent between the VSP and the DS will be logged.

Path parameter defines the file to which the data will be "dumped".

Dump section can be omitted completely. If it is present then both parameters must be present.

Application Notes

This part contains all our Application Notes in chronological order:

- **AN001. Customization options in our Products**
- **AN002. Practical advice on integrating EM Module into your device**
- **AN003. Time delays when the DS is opening TCP connection to the PC**
- **AN004. How to send the same data to several DS**
- **AN005. Remotely controlling I/O lines on the DS**
- **AN006. Using Device Server Toolkit with Windows Firewall (XP/SP2)**
- **AN007. Installing and Configuring LinkServer**
- **AN008. Using HyperTerminal**
- **AN009. WAN Basics**
- **AN010. Controlling the DS from the Serial Side**
- **AN011, Reading the Production Label**

AN001. Customization Options in Our Products

What's in this Application Note

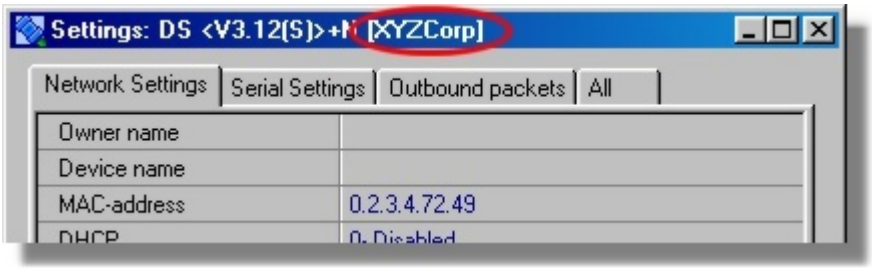
Our Customers that supply Tibbo Device Servers with their own equipment (systems) often wonder if it is possible to alter default setting values of the DS, hide certain settings so they could not be seen and edited in the DS Manager, hide some installation components (for example, VSP Manager), display their Company name instead of Tibbo, etc. This article describes an array of customization options that are available.

Contents:

- Define your own default (post-initialization) setting values
- Hide settings so they cannot be viewed (edited) through the DS Manager
- Select which DST components should be installed on your User's PC

- Replace default name and logo (bitmap) with your own name and logo in the installation program
- Prepare your own installation file (distribution CD)

Define your own default (post-initialization) setting values



Important note: Release3.5 branch of application firmware (i.e. firmware for second-generation devices) only supports custom profiles starting from **V3.54**. Unfortunately, there remains a limitation regarding how the firmware file with custom profile attached can be loaded into the DS. Custom profile will only work if upgrade was performed through the network. If firmware file (with custom profile attached) was loaded through the serial port then custom profile will not work correctly. We apologize for inconvenience but nothing can be done to remove this limitation. Release3.0 branch of application firmware does not have this limitation.

It is now possible to define your own, different, post-initialization setting values. Why would you want this? Well, if you supply a system in which the DS is always used in TCP/IP mode, then you might want to make the **Transport Protocol (TP) setting** default to 1(TCP) instead of 0(UDP), as it is "by default". And if you know that your serial device works at 19200bps, then wouldn't it be good to set this baudrate as a default value for the **Baudrate (BR) setting** (instead of the "factory default" of 38400bps).

The process of defining your own post-initialization defaults starts with creating a *profile file*. This file should have a *.txt* extension. You can use a *Notepad* or any other simple text editor to create this file. In our example we will create a profile for a Company named "XYZCORP". We will create a custom profile that will make the following changes to the original post-initialization values of the DS:

-**Transport Protocol (TP) setting**= 1(TCP)
-**Routing Mode (RM) setting**= 2(client)
-**Baudrate (BR) setting**= 4(19200bps)
-**Parity (PR) setting**= 2(odd)

Open the notepad and create the file named *profile.txt* that contains the following data:

```
_XYZCORP
TP1
RM2
BR4
PR2
```

Note: put <CR> even on the last line so this line is complete.

Each line of this file except the first one corresponds to a particular setting. Setting mnemonics and desired values are exactly the same as the ones used for DS programming via the serial port or network- you can find this data here.

The first line of this file is reserved for the profile name. This name is displayed by

the DS Manager in the caption of the *settings dialog*. This feature is provided so that it was possible to tell whether the DS is running a "profile-modified" firmware or default firmware. The profile name must start with the underscore sign ("_") or it will not be recognized.

Now that your profile is created you need to merge it with the DS firmware file. To do this, you need to use a small DOS program called *add_prof.exe*. The program can be emailed to you on request.

Put all three files (firmware file, profile file, and add_prof.exe) into the same directory. The firmware file should be of "W" or "S" type (i.e. *em_314w.bin* or *em_314s.bin*). As explained above, this means that only "older" devices (EM100-00/ -01/ -02, DS100-00/ -01/ -02) can use profile files at the moment.

The executable is a DOS program, so open a DOS session and run the program like this:

add_prof *input_firmware_file profile_file output_binary_file*

Note: you don't need to specify file extensions, the program will assume that *input_firmware_file* and *output_binary_file* have *.bin* extension, and the *profile_file* has the *.txt* extension. So, in our example you should type:

add_prof em_312s.bin profile em_xyz

The output file, *em_xyz.bin* will contain exactly the same firmware code plus the profile data you have defined. Now upgrade your DS using this new file.

After the upgrade launch the *DS Manager* and open the settings dialog of the DS that you've just upgraded. Notice that dialog caption now displays the name of your profile.

Now close the dialog and initialize the DS. After the initialization is finished open the *settings dialog* again and observe new default setting values that have been set.

Hide settings so they cannot be viewed (edited) through the DS Manager

Custom initialization profiles allow you to alter the initial post-initialization values of settings but they do not protect those settings from being edited by the User. Sometimes it makes a good sense to hide certain settings. For example, if you know that in your system the DS is always used in TCP/IP mode then it may be a good idea to hide the **Transport Protocol (TP) setting** so it is not displayed by the DS Manager at all. The principle here is: "less choices offered- less questions asked".

In our DST software the list of settings displayed for the DS by the *DS Manager* is defined by the *.sdf* files (SDF= Setting Definition File). During installation those files are copied into the *.../Program files/Tibbo/Device Server Toolkit/sdf folder*. Filenames correspond to the firmware versions of DS (except for the file *wizserial.sdf*- this one is used by the Connection Wizard and must be always kept as is). For example, your */sdf* folder can contain the files *V2-21.sdf* and *V3-00.sdf*. These files correspond to firmware V2.21 or higher and V3.00 or higher. That is, when the *DS Manager* encounters the DS with firmware V2.53 it will use the file *V2-21.sdf* because 2.53 is later that 2.21 but earlier than 3.00.

The data structure inside the *.sdf* file is very straightforward: each line represents one Setting or Setting group. Here is one (abstracted) example of such a file:

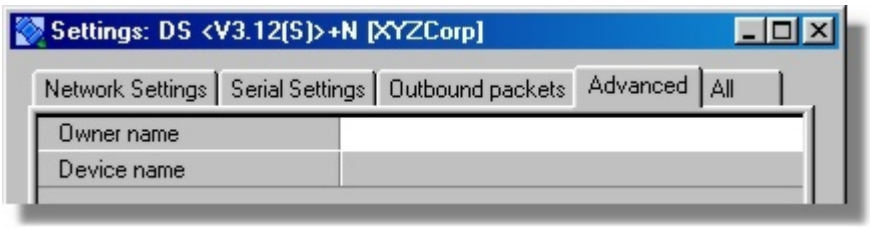
```
I=$NET;D=Network Settings;T=GROUP
I=ON;D=Owner name;T=STRING;C=EDIT;MAXLEN=8;F=R
I=DN;D=Device name;T=STRING;C=EDIT;MAXLEN=8;F=R
...
...
```

```
I=E1;D=Stop-character (ASCII
code);T=INT;C=STATIC;M=CHARDLG;V=E1>255? "ASCII code cannot
exceed 255":""
I=P1;D=Number of
post-characters;T=INT;C=EDIT/SPIN/0/1/1/1;V=P1>15? "Maximum
number of post-characters is 15":""
```

The first line in the above example creates a setting group "Network Settings". Each group gets its own tab in the settings dialog of the *DS Manager*. All other lines in the above example represent individual settings. You can change the way the *DS Manager* displays the settings by rearranging and/or editing the data in the *.sdf* file. For example, supposing you want to create another group of settings called "Advanced" and put two settings there: **Owner Name (ON)**, and **Device Name (DN)**. All you have to do is add three lines to the end of the *.sdf* file:

```
...
...
I=$ADVANCED;D=Advanced;T=GROUP
I=ON;D=Owner name;T=STRING;C=EDIT;MAXLEN=8;F=R
I=DN;D=Device name;T=STRING;C=EDIT;MAXLEN=8;F=R
```

Save the file and reopen the *settings dialog*: you will see the new tab:



To hide settings simply remove corresponding lines from the *.sdf* file.

All *.sdf* files are supplied to the User as a part of the Device Server Toolkit installation. To see those files download the *.zip* archive of our distribution to your PC and unzip this archive into a separate folder. You can change the *.sdf* files and then prepare your own installation that you will distribute with your system (see below for details).

Some settings in the file refer to other settings. For example, "PPPoE login name" may depend on the "PPPoE Mode" setting. So when removing "PPPoE Mode" you must also remove "PPPoE login name", or else you would get an error when loading the SDF file.

You can see this according to the "S=" or "V=" fields for each setting. If these refer to another setting, you cannot remove that "other" setting without changing or removing the current one. For example:

I=PP;D=PPPoE mode;T=INT;C=STATIC;O=0- Disabled/0/1- Enabled (on connection)/1/2- Enabled (on powerup)/2

I=PL;E=1;D=PPPoE login name;T=STRING;C=EDIT;MAXLEN=20;F=R*;S=PP!=0?"e":"i"

Note the "S=PP!=0" in the second line -- the S= parameter refers to the PP setting. So you cannot remove the PP setting as long as the PL setting refers to it. You would have to remove both, or at least remove the "PP" reference from the "PL" line (but this might cause functional issues -- these settings are linked for a reason).

Select which DST components should be installed on your User's PC

You can hide certain DST components from being installed on your User's PC. This may come handy sometimes. For example, if you distribute the DS in a system that does not utilize the VSPD then why install the *VSP Manager*? This will only create confusion and extra questions!

Components that can be selected/deselected are listed in the *tdst.ini* file found inside the DST distribution. To see the *tdst.ini* file download the *.zip* archive of our distribution to your PC and unzip this archive into a separate folder. Open the file using any simple text editor (i.e. *Notepad*). You will see that the file contains the following data:

```
...
showdlg = 1
conwizard = 1
vsp = 1
```

The first line specifies whether the User will be presented with the screen asking him to choose which components to install. When *showdlg = 1* the screen will be shown during the installation. When *showdlg = 0* the screen will not be shown. The second line defines the default choice for the Connection Wizard: 1- "install", 0- "do not install". When *showdlg = 1* the User will still have a chance to reverse this default selection, but when the *showdlg = 0* the User won't be able to choose and the *Connection Wizard* will be installed or not installed according to the value of *conwizard* entry in *tdst.ini*. This means that if you set *showdlg = 0* and *conwizard = 0* then the *Wizard* will never be installed (and the User won't even know that there *is* such an option).

The third line (*vsp=x*) defines whether the *VSP Manager* and the Port Monitor will be installed (*Port Monitor* is always installed together with the *VSP Manager*). Currently, there is no any way to disable the **DS Manager** so this component is always present.

It is not enough to just change the file *tdst.ini.* You need to "sign it up" in order for the installation program to recognize this changed file. See below for more information.

Replace default name and logo (bitmap) with your own name and logo in the installation program

When you distribute the DST as part of your own system you may wish to display your own Company name and logo in the installation screens.

You can achieve this by editing the data in the *tdst.ini* file. To see the *tdst.ini* file download the *.zip* archive of our distribution to your PC and unzip this archive into a separate folder. Open the file using any simple text editor (i.e. *Notepad*). You will see that the file contains the following data:

```
companyname = Tibbo Inc.
companyshortname = Tibbo
productname = Device Server Toolkit
setupimage = Tibbo.bmp
setupimageparams = 1;;;255,0,255
...
```

Companyshortname and *productname* define the default installation directory of the software. Directory is:

```
.../Program Files/companyshortname/productname
```

So, if your Company name is "XYZCorp" and you wish the software to be called "DST" then input the following data:

```
companyshortname = XYZCorp
productname = DST
```

In this case the default installation directory for the software will be:

```
.../Program Files/XYZCorp/DST
```

Productname is also shown during the software installation. You can define your own *setupimage* as well. This is a bitmap that is displayed in the installation screens. Unfortunately, there is no way to define the position of this bitmap in the window- *InstallShield* software used to generate installation files doesn't provide this flexibility. So, the bitmap position is fixed. The only thing you can do is define if this bitmap will contain a transparent color.

Setupimageparams line has several parameters. The first one specifies if there is a transparent color (1 for "yes" and 0 for "no"). Last three numbers specify which color is to be transparent. For example, if you don't want any transparency then set *setupimageparams=0;;;0,0,0*. And if you want the white color to be considered transparent then set *setupimageparams=1;;;255,255,255*.

One remaining parameter that we haven't mentioned yet is *companyname*. It is not displayed anywhere in the software itself but it is saved into the *Windows registry* upon installation.

Prepare your own installation file (distribution CD)

If you have made changes to the file *tdst.ini* or have modified the bitmap specified by the *setupimage* line then you need to *signup* your changes first. Signup process calculates the hash on the file contents and saves this hash in the *signature*= line of the *tdst.ini*. When the signature and the contents of the files do not match the changes you have made are ignored. You need to perform the signup process after you have made all the changes. Signup does not include the *.sdf* files so if you just changed these files you don't need to do it.

Here is what you need to do to signup your changes. First, request (from us) an archive called *signup.zip*. After you receive the file unzip it into the same directory where you unzipped all the installation files for the DST. *Signup.exe* is a DOS program and you run it like this:

```
signup tdst.ini
```

If everything is OK the program will generate the following output:

```
tdst.ini has been successfully signed up
```

You can distribute the modified version of the *DST* as a set of files or you can create a self-extracting ZIP archive (recommended). *Signup.exe* and *tdstsign.dll* files should not be included with your distribution. Please, make sure these files don't get included by mistake!

AN002. Practical Advice on Integrating EM Module into Your Device

What's in this Application Note

Over the cause of the past several years we have been in constant contact with Companies that are using our Ethernet-to-serial Modules (such as EM100) to

network-enable their products. Invaluable experience and feedback that we have received in the process has helped us realize that some practical aspects of the Module integration are not always understood early enough in the product design cycle. As a result some devices that utilize our Modules were designed without proper consideration for the practical issues that may arise during the manufacturing or operation of the Device. This Application Note summarizes our experience. Follow the suggestions below and you may well save yourself a lot of headache in the future.

This application note does *not only apply to EM100!* EM100 is used as an example device, but the advice below also applies to EM203A, EM200, and other Tibbo embedded products.

Contents:

- Position the Module as close as possible to the RJ45 connector, do proper layout
- Make sure you supply good power at required current
- Proper reset is a must!
- Provide status LEDs, prog/init button, and serial access connector
- Let your main CPU control the Module...
- ...Or let the EM Module control your main CPU

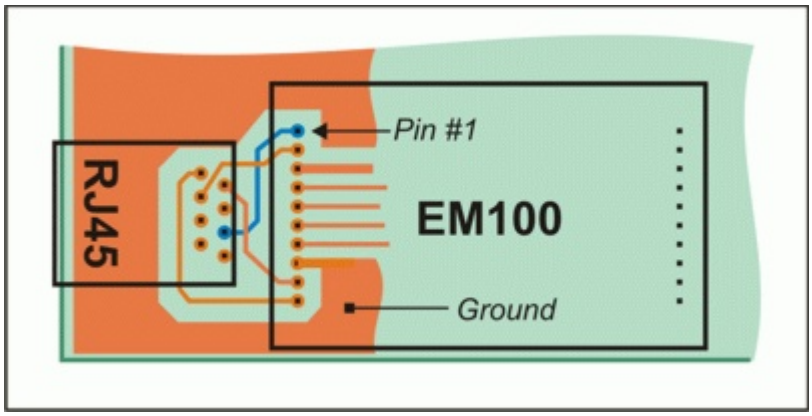
Position the Module as close as possible to the RJ45 connector, do proper layout

Note: this part applies to all Tibbo Modules.

Improperly positioned EM Module will generate a lot of noise (EMI or Electro-Magnetic Emission). Put the EM, its Ethernet magnetics (if not built into the Module), and RJ45 connector as close as possible to each other. Failing to do so will make passing CE (FCC) EMI test very difficult! Additional problems may include communications problems and susceptibility to electrostatic discharge (ESD) damage.

There is a common misunderstanding regarding the CE/FCC EMI certification of our Modules. Being Modules, they are not certified "as is" but are tested on a particular PCB, usually our Evaluation Board. These boards have proper PCB layout and a conclusion is drawn that if the EM Module can pass the test on such a board then it can pass a test on another board with a proper layout. So, even though the EM Module itself is pre-tested your whole device may require CE/FCC certification.

Shown below is a proper layout for the EM100 Ethernet Module. This Module has Ethernet magnetics built-in, so only RJ45 connector must be added externally. The main point here is the length of wires that link the pins of the RJ45 connector to the RX-, RX+, TX-, TX+ pins of the EM100. The shorter the length, the lower the noise.



Another important point. Do not create a ground plane in the vicinity of the RJ45 connector and RX-, RX+, TX-, and TX+ lines. The ground plane should "wrap around" the RJ45 area, but not cross any of the 4 interface wires. This is very important! Providing a ground plane beneath the connector area will make your device very susceptible to the ESD.

Make sure you supply good power at required current

Tibbo EM Modules are just like IC chips- they require a regulated DC power source, 5V nominal, +/- 5% deviation. Some Modules (such as EM200) consume relatively high current (~230mA)! We provide this data in our Documentation and yet we still have cases where our Customers use "bad" power supplies. So, this advise is simple: make sure you supply regulated 5V DC and that your power supply can offer necessary current.

Proper reset is a must!

Tibbo EM Modules do not have internal reset circuits so you must provide an external power-on reset. It is not enough to just tie the RST pin to VCC, this won't work! We also do not recommend using a simple RC-circuit. Such circuits are not reliable because they do not monitor the supply voltage and won't work properly under many circumstances.

For example, if you have a power supply that "starts slowly" (i.e. its voltage rises slowly after the powerup) then the reset pulse generated by the RC circuit may end before the VCC comes to within 5V-5%. Another problem condition is a "brownout" (this is when the VCC momentarily goes below 5V-5%). Brownouts can cause the EM Module to hang up. Use a proper reset IC instead- this will give you a very reliable reset both on startup and in case of a brownout. Some reset ICs (for example MAX810 from MAXIM) have a very attractive price now (about USD0.5). Reset IC you use must have the following spec: (1) active high reset, (2) reset pulse of at least 100ms, (3) reset threshold of around 4.7V. If your device already has a matching reset IC then you can use its output for the EM Module as well.

Another possible reset arrangement is to use a free I/O pin of your device's main CPU. I/O pins of most CPUs default to HIGH state upon hardware reset. For example, supposing that your device is based on the x51 microcontroller. You already have a reset circuit for this microcontroller so why would you add yet another one to reset the EM Module? You can connect an unused I/O pin of the x51 to the RST pin of the EM. On startup the x51 will receive a proper reset from the reset circuit and its I/O pins will be in the HIGH state. This will provide a reliable reset to the EM. You can modify the code of your main microcontroller to switch the "reset" I/O pin to LOW after a delay of, say, 100ms.

We recommend you to use this "CPU-controlled reset" approach whenever possible. Not only it lowers your cost (no need for an extra reset IC) but also gives your main CPU the ability to reset the EM Module at any time. This, as we will show later, can come very handy if you want to enable your CPU to program the settings of the EM and/or upgrade its internal firmware. Of course, it is not always possible to control the reset from the main CPU (no spare I/O lines may be available, it may be impossible to modify the firmware of the main CPU, etc.). In this case just go with a proper reset IC.

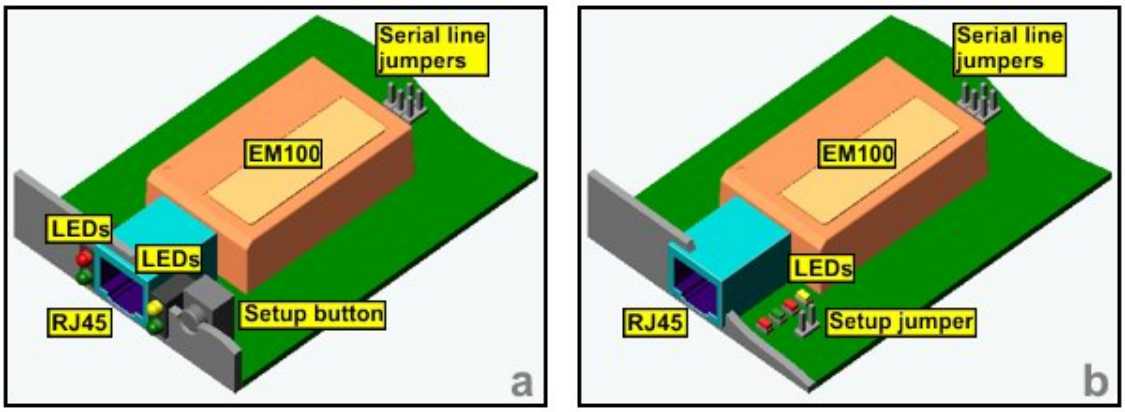
Provide status LEDs, prog/init button, and serial access connector

Note: part about LEDs applies to all Modules.

Tibbo EM Modules have four LEDs control lines (ER, EG, SR, SG). We strongly recommend you to provide these LEDs on your board. Not having them leaves you

"blind". Supposing, the EM in your device is not working as expected. If you have those LEDs then you can solve the problem much easier. SR and SG LEDs display many different signal patterns so one glance at them can tell you a lot about the current EM status. ER and EG LEDs tell you about the Ethernet connection, which is also important.

Ideally, you should make those LEDs visible from the outside (left figure). This way your Users can also see them. If this is impossible then at least provide these LEDs on your board so they can be observed when your device's cover is removed (right figure).



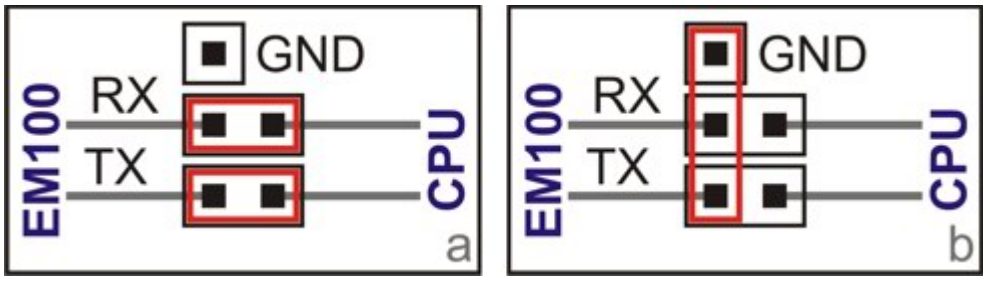
Besides LEDs, we strongly recommend you to provide a setup button that is connected to the MD pin of the EM. Again, it is best to make this button "pushable" from the outside (left figure above). If this is impossible you can provide an internal button or a simple jumper on your board (right figure above). Why you need this? Because in the life of your device you (or your Users) can encounter a situation when they need to do one of the following: (1) initialize the EM, (2) program the EM through the serial port, or (3) upgrade internal firmware of the EM.

Initialization may be required because the EM has somehow lost its settings (due to powerful ESD or for some other reason) or was incorrectly setup. To initialize the EM using the setup button you use the quick init procedure.

Programming through the serial port may be required if the EM Module cannot be programmed through the network for some reason. In this case you just press the setup button to put the EM into the serial programming mode and do all the programming through the serial port. Of course, you need to have an access to the EM's serial port for this. Read on, we will come to that.

Finally, the firmware upgrade through the serial port cannot be ruled out. Of course, you can mostly use network upgrades, but don't count on this. Upgrade through the serial port is the most reliable way that will always work. To do the serial upgrade, you need to have a button. You also need to have access to the EM's serial port, so we will now discuss the subject.

Our last suggestion is that you provide a simple way to break the serial connection between the EM and your main CPU. You can pass the TX and RX lines of the EM through two jumpers. These jumpers are to be closed during the normal operation so the EM is connected to your CPU (see figure below). When you need to access the serial port of the EM directly you remove the jumpers and plug in a simple serial cable. It only needs to have three lines: TX, RX, and Ground. Because the EM has a TTL serial port you cannot connect this cable to the COM port of your PC directly. Use a simple board with RS232 IC installed on it to provide TTL<-->RS232 signal translation.



By providing the jumpers you are leaving a back-door access to the EM AND to the serial port of your main CPU. This may prove very useful during repairs, field service, etc. One final note: if you thing that jumpers are too unreliable or you simply don't have enough space for them on the board then you can use solder jumpers instead.

Let your main CPU control the Module...

One other important suggestion: let your main CPU control the EM Module. Under "control" we understand the following: (1) ability to reset (restart) the EM at any time, (2) ability to program the EM, (3) ability to upgrade internal firmware of the EM. Implementing this is optional but so easy to do that there is no real reason why you shouldn't (if you can). True, the EM doesn't have to be programmed through the serial port, you can do all the programming through the network using the DS Manager. But this is not always consistent with the rest of your device's features.

Here is an example: your device has an LCD screen and a keypad and there is a setup mode that allows you to set all the functioning parameters (time, date, etc.). To be consistent, you should include EM-related settings into this onscreen setup as well (for example, IP-address of the device). For this, you must provide a way for your main CPU to put the EM into a serial programming mode.

Another possibility is to have the main CPU upgrade the firmware of the EM. Again, EM Module can be upgraded through the network using the DS Manager, and in those few cases when the network upgrade is not possible you can just use the "plug-in cable" method described in the previous section. Still, it is much more "neat" if the firmware can be upgraded by your main CPU itself. We have had a case in which the EM was used in a terminal device that also was field-upgradeable. Our Customer has implemented a system, under which the upgrade file for the main CPU contained the firmware of the EM as well (EM firmware file was merged into the firmware file of this "host" device). After the terminal was loaded with the new firmware it automatically initialized itself and also downloaded the new firmware into the EM! This gave the Users an overall feeling of "one-ness" of the device they use.

To achieve all this flexibility you only need to have your main CPU control two extra lines: RST and MD (because TX and RX lines are already there, right?). We have already discussed how to connect the RST line. MD line is connected in this same fashion. So, you only need two extra CPU I/O lines.

Note, that we still recommend you to provide all the bits and pieces that we have suggested in the previous section. This means that we suggest you to have the MD line of the EM connected to your main CPU and the setup button at the same time. Many CPUs and microcontrollers have the I/O pins that allow them to be driven low externally and high internally at the same time (for example, the x51 microcontroller is like that). In this case you can simply parallel the CPU output and the setup button (Fig. 4a). If this is not permissible you can use two (Shottky) diodes to separate the I/O pin from the button (Fig. 4b).

In some cases it is not possible, of course, to have the main CPU control the RST and MD lines. You may have no spare I/O pins or interconnection between PCBs in your device may have no spare wires to carry RST and MD signals. In this case you won't be able to put the EM into the firmware upgrade mode but you can still have your main CPU program the EM's settings whenever necessary. There is an alternative method (two methods, actually) of putting the EM into the serial programming mode- by sending a so-called escape sequence.

The only complication is that appropriate escape sequence must be enabled for this to work. This is defined by the **Soft Entry (SE) setting**, which defaults to the post-initialization value of 0 (escape sequence disabled). Yes, you can manually enable this by using the **DS Manager** but if the EM gets initialized the escape sequence will stop working and you will have to manually edit the SE setting again!

To avoid this situation you can define your own post-initialization setting values. Instead of going with the factory default of 0(disabled), you can choose to have the EM default to 1 or 2 (escape sequence type1 or type2). Our Application Note 1 ("Customization options in our Products") explains how to do this.

...Or let the EM Module control your main CPU

And here is a complete reversal of the idea: sometimes it may be better to have the EM Module control the main CPU in your device. We have had a case in which an EM100 was used in a very simple serial machine. This machine also had a firmware upgrade feature. The User had to press a special button, then power the device up and upload the firmware through the serial port. After we have added the EM100 the serial port of the device was connected to the serial port of the EM100. Of course, new firmware could be loaded into the device through the EM100 (hence, through the network) as well. This could have allowed for the remote firmware upgrades if it wouldn't have been for a small problem- the "upgrade" button still had to be pressed on the machine itself- and this killed the whole "remote upgradeability" idea.

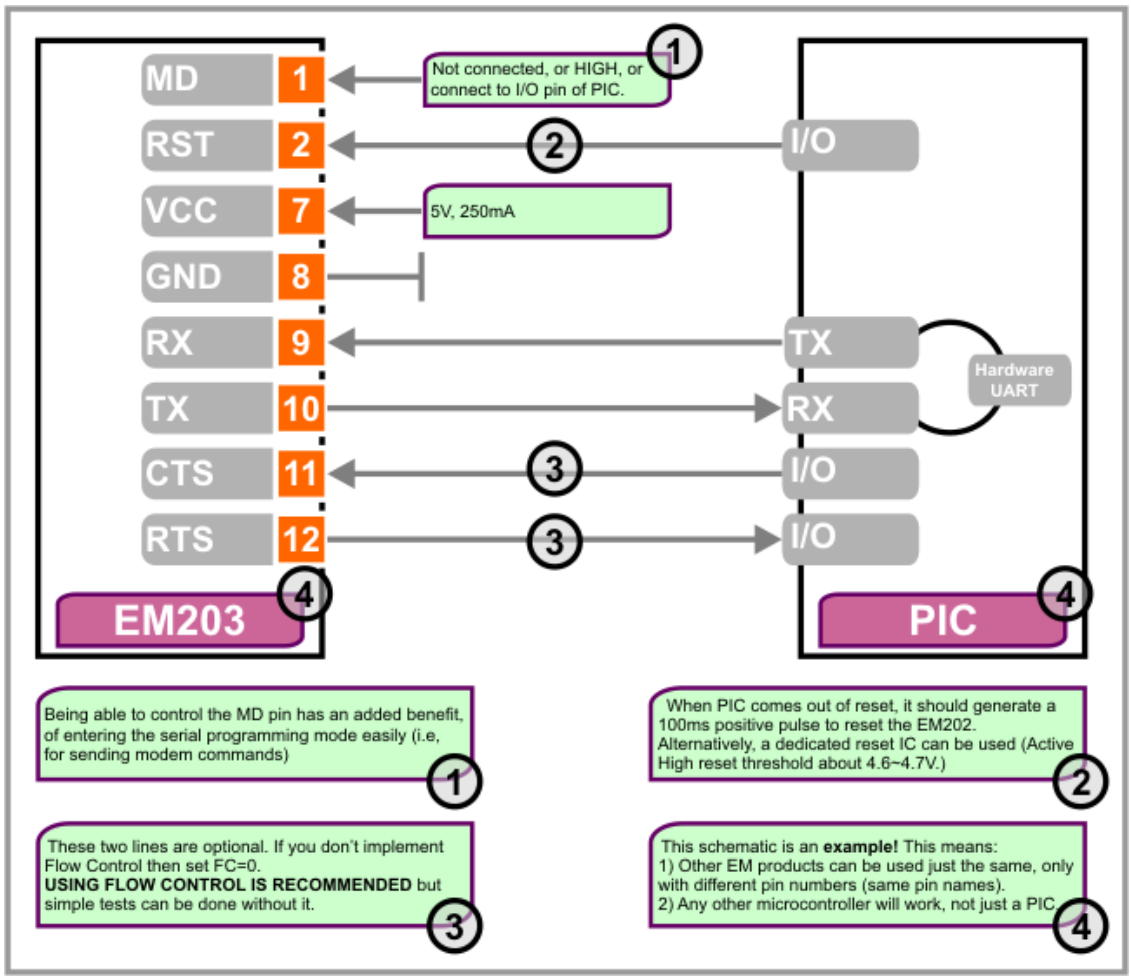
To solve this we have connected the download line of the device's main CPU to a general-purpose I/O pin of the EM100. By remotely controlling the state if this I/O line we can now put the main CPU into the firmware upgrade mode! Now the upgrades are truly remote.

Example: PIC with EM203

Below is an annotated schematic, showing how one would go about connecting an EM203A device to a PIC (Programmable Interrupt Controller -- a type of common controller chip). The PIC is used merely as an example -- the information below applies to any similar connection.

Note that the pins shown below are *not* all of the pins an EM203A has. There are other pins, which are not used in this example.

Schematic Diagram:



AN003. Time Delays When the DS is Opening TCP Connection to the PC

What's in this Application Note

Some of our Customers have noticed that sometimes, when the DS is trying to establish a TCP connection to the PC, it takes pretty long time to connect. It looks as if PC is ignoring a connection request. This problem is often encountered when the DS is trying to "re-connect" after an abrupt reboot (i.e. due to power loss). This Application Note explains why this happens and what can be done to counter the problem.

Contents:

- How connection delay happens
- Using "connect immediately" mode to let the DS handle reconnects
- Using modem commands and DTR line to monitor connection status and repair connections

How connection delay happens

Connection delay problem is related to how "TCP stack" on the PC handles existing TCP connections. We will explain this on the example of one frequent

communications scenario:

- The DS establishes a TCP connection to the PC (for example, because it has some serial data to send). Connection is accepted immediately. PC identifies this connection by a combination of three parameters: remote host's IP (i.e. IP-address of the DS), remote host's port (i.e. port number on the DS from which this connection was established- 10001 by default), and local port on the PC to which this connection was established (i.e. "listening" port of your application or VSP).
- The DS suddenly reboots, for instance due to power loss. Your PC doesn't know about this so it thinks that this TCP connection is still alive.
- The DS is powered on again and makes an attempt to connect to the PC. Once again, it tries to open a connection from port 10001. PC responds with "ACK", as if this was a regular data packet. Basically, PC just ignores this connection attempt. Long time needs to pass before PC realizes that this connection is dead already.
- The DS notices that the PC did not respond to the connection request and sends "RST" packet to reset the connection. At the same time, the DS discards the data it was going to send to the PC and goes to "IDLE" state.
- When new serial data is received by the DS the latter makes another attempt to make a connection. This time the source port on the DS is 10002, not 10001. This is because source port from which the DS is establishing its TCP connection is incrementing with each connection attempt (this is normal, PC does this too).
- PC gets this new connection attempt and treats it as a completely new connection. This is because in the unique combination of source IP + source port + destination port something is now different: the source port number has changed. Because of this, PC accepts this new connection immediately.

The following network log made by WinDUMP sniffer software illustrates the above scenario. The DS is at 192.168.100.97 and the PC is at 192.168.100.90.

```
==========
*** DS makes an attempt to connect to the PC ***

192.168.100.97.10001 > 192.168.100.90.1002: S 640000:640000(0) win 1024
<mss 255> (DF)
192.168.100.90.1002 > 192.168.100.97.10001: S 10461646:10461646(0) ack
640001 win 8415 <mss 1460> (DF)
192.168.100.97.10001 > 192.168.100.90.1002: . ack 1 win 1024 (DF)

*** Connection is now established ***

*** DS reboots abruptly ***

*** DS makes an attempt to establish connection again ***

192.168.100.97.10001 > 192.168.100.90.1002: S 256000:256000(0) win 1024
<mss 255> (DF)

*** PC replies with ACK which is not correct, so DS resets this connection attempt,
discards data ***

192.168.100.90.1002 > 192.168.100.97.10001: . ack 1 win 8415 (DF)
192.168.100.97.10001 > 192.168.100.90.1002: R 640001:640001(0) win 8415
```

(DF)

*** DS tried to connect again, this time from a different port, this time connection is accepted ***

192.168.100.97.10002 > 192.168.100.90.1002: S 640000:640000(0) win 1024 <mss 255> (DF)
192.168.100.90.1002 > 192.168.100.97.10002: S 10470652:10470652(0) ack 640001 win 8415 <mss 1460> (DF)
192.168.100.97.10002 > 192.168.100.90.1002: . ack 1 win 1024 (DF)
==========

Bottom line: the first connection attempt by the DS after an abrupt reboot can fail. If your serial device just sends several bytes of data and waits for TCP connection to be established then this may never happen. You serial device needs to resend the same data second time after a delay- only then TCP connection may be finally accepted by the PC.

There are two ways of speeding thins up and prevent the PC from delaying the connection.

Using "connect immediately" mode to let the DS handle reconnects

Program **Connection Mode (CM) setting** to 0 (connect immediately)- this way the DS will make infinite attempts to connect unstill succeeded. This way your serial device doesn't need to send data repeatedly to have connection established. Most of the time connection will be ready.

Using modem commands and DTR line to monitor connection status and "repair" connections

Modem commands (officially called serial-side parameters and instructions) give your serial device a way to monitor connection status and control connection establishment/termination. Instead of sending serial data and hoping it will pass through your serial device you can first make sure that connection is, indeed, established.

Here is a brief description of how this might work:

- Your serial device first makes the DS enter the serial programming mode.
- Your serial device then uses **Establish Connection (CE) instruction** to make the DS start connection establishment procedure.
- After this, your serial devices uses **Echo (X) command** to check the result (*c flag* in the reply). If connection is not found to be established rather quickly and *c flag* is unexpectedly returns to "*" then your serial device can conclude that the DS has returned to idle mode because the PC did not respond to the connection establishment request.
- Then, the serial device can quickly issue another **Establish Connection (CE) instruction** and, again, monitor connection status via **Echo (X) command**.
- After the **Echo command** shows that connection is established (*c flag*="C") the serial device uses **Logout (O) command** to make the DS exit the serial programming mode. Data exchange can start after that.

The above procedure, although seemingly lengthy, takes only a fraction of a second to complete.

One remaining question is: how to detect that connection has been broken while being in the "normal mode". Indeed, when the DS in the serial programming mode

connection status can be verified at any time through an **Echo command**. But how to do this when the DS is in the data mode and Echo command cannot be issued?

This is done by programming the **DTR Mode (DM) setting** to 1 (connection status). In this case the DTR line of the DS shows whether network connection is established or not. Once the line is "dropped" the serial device will know that something has happened to the connection. At this time the serial device should force the DS into the serial programming mode and use modem commands to "repair" the connection.

AN004. How to Send the Same Data to Several DS

What's in this Application Note

We are often asked if it is possible to send the same data to several DS at the same time. We understand why such need may arise. For example, in RS485 systems there is often a master "node" that sends commands to several slaves. Every slave receives the command but only a particular one will reply back. This is because the frame (packet) sent by the master contains an address of the slave being addressed.

Network-enabling such system requires that this "send to all" communications method is somehow adopted to network communications. This Application Note explains how this can be done.

Contents:

- Using UDP broadcasts to emulate multi-drop communications
- Limitations

Using UDP broadcasts to emulate multi-drop communications

The only way to arrange multi-drop communications system on the TCP/IP network is by using UDP/IP protocol and sending data as UDP broadcasts. TCP protocol is, by definition, a point-to-point protocol and cannot be used for data delivery to several nodes simultaneously. UDP, on the contrary, can and is often used to send the data to several nodes on the network.

To make one DS (we will call it "master") send data to several other DS ("slaves") through the network perform the following setup (only "important" setup changes are shown):

- On the master DS side make the following setup changes:
  - **Routing Mode (RM) setting**: 1 (server/client) or 2 (client);
  - **Transport Protocol (TP) setting**: 0 (UDP);
  - **Destination IP-address (DI) setting**: 255.255.255.255- this will make the master DS send data as link-level UDP broadcasts.
- On each slave DS make the following setup changes:
  - **Routing Mode (RM) setting**: 0 (server);
  - **Transport Protocol (TP) setting**: 0 (UDP);
  - **Broadcast UDP (BU) setting**: 1 (enabled)- this will make slave Device Servers accept the data sent in broadcast packets.

Same can be applied to using Virtual Serial Port as a "master". Use VSP Manager

and set the properties of the VSP in a way similar to the setup of the master DS.

Limitations

It is important to understand the limitations of such a system:

- **Unreliable data transmission.** UDP, as opposed to TCP, does not guarantee data delivery. If any UDP packet is lost the UDP protocol itself won't be able to detect this. UDP packets are considered to be "less important" (compared to TCP traffic) so many routers drop such packets first when becoming overloaded. The bottom line is that you should not base your system on the assumption that all UDP packets will always be received by all slaves. This is usually not a problem since most multi-drop systems include some sort of retry algorithm.
- **The system is limited to local network segments only.** This second limitiation is more serious. Broadcast UDP packets *are not routed* by network equipment such as routers, bridges, etc. This means that UDP broadcasting is only possible within a single network segment. Do not expect to be able to make such a system work throught the Internet, for instance.

AN005. Remotely Controlling I/O Lines on the DS

What's in this Application Note

Many of our Customers have asked us if it possible to make our Modules (such as EM100) or "finished" devices (DS100) work as a sort of remote I/O i.e. use the DS to control "loads" (switch something on/off) and monitor "sensors" (detect if the switch is closed or opened). The answer is "YES" and this Application Note examines how this can be done.

Contents:

- General-purpose I/O lines of Tibbo Device Servers
- Controlling I/O lines through a Virtual Serial Port
- Setting and sensing the status of I/O lines using on-the-fly commands
- Using notifications to get the status of I/O lines

General-purpose I/O lines of Tibbo Device Servers

You probably know that every DS model we manufacture contains some number of lines that can be used as remote I/O. For example, the serial port of our DS100R Serial Device Server contains RTS (output) and CTS (input) serial lines. From the serial port's standpoint, RTS and CTS have a specific serial port-related function but at the same time they can be used as generic output and input!

Other "external" Serial Device Servers, such as DS100B and DS203 also have DTR (output) and DSR (input) lines. Together with RTS/CTS, this brings the total number of available lines to two outputs and two inputs.

Most embedded Modules such as EM100 and EM200 have even larger number of I/O lines, some of which are not related to the standard serial port control lines in any way. For example, in addition to RTS, CTS, DTR, and DSR the EM200 has five additional I/O lines- P0, P1, P6, P7, P8. All 9 lines can be used as universal

inputs/outputs- on the Module level there are no dedicated "input only" or "output only" pins*.

Controlling I/O lines through a Virtual Serial Port

The simplest way to control I/O lines of the DS is through a Virtual Serial Port. Naturally, this will be limited to setting the status of RTS and DTR lines and sensing the status of CTS and DSR lines. Extra lines such as P0, P1, etc. cannot be controlled this way. Additionally, RTS/DTR will only work as outputs and CTS/DSR- as inputs, even on embedded Modules whose I/O pins are bi-directional (see above). The advantage of using VSP is in simplicity- you don't need to write a lot of additional code.

To test how VSP works with I/O lines of the DS you can create a simple application in Visual Basic. Use **MSComm** ActiveX control to work with the VSP. Remember, that same rules that apply to a regular COM port of the PC also apply to the remote control of the serial port on the DS:

- If you want to set RTS and sense CTS you need to set **MSComm.Handshaking="None"**. If you set handshaking to **"ComRTS"** your VB program won't be able to work with RTS/CTS directly.
- After you set **MSComm.Handshaking="None"** you can control RTS through **MSComm.RTSEnable** and sense CTS state through **MSComm.CTSHolding**.
- Likewise, DTR state can be controlled through **MSComm.DTREnable** while DSR state can be sensed through **MSComm.DSRHolding**.
- To be able to control DTR remotely, you need to program the **DTR Mode (DT) setting** of the DS to 0 (idle). Also, to have the DSR work as a pure input line program do not set **Connection Mode (CM)** to 3 (on command or DSR=HI). These two settings will be taken care of automatically by the Connection Wizard when you use it as described below.

Before you can make this test you need to have a proper setup for the VSP and DS. Run Connection Wizard and make the following choices. Important points:

- Job: choose to create a link between the Virtual Serial Port and the Device Server.
- Initiator of data exchange: Virtual Serial Port.
- On-the-fly commands: yes, enable on-the-fly commands, use out-of-band access method.

(all remaining choices are obvious or irrelevant).

After the Wizard has done its job and is at the last screen you need to do the following:

- Click "open COM settings"- the properties of the VSP will be opened (you can reach the same screen from the VSP Manager).
- Change Connection Mode of the VSP to "immediately".
- Click OK to close the dialog.

Here are some comments on the setup that we've just described. On-the-fly commands are enabled because control of the state of I/O lines is effected through these on-the-fly commands. Connection mode of the VSP was set to "immediately" to enable so called *status change notifications* to be sent from the DS to the VSP (PC) right from the moment the VSP is opened. More on this can be found in one of the sections below (see "using notifications to get the status of I/O lines").

After the setup is finished you can use your VB program to work with RTS, CTS, DTR, and DSR lines. While you are playing with the system take a look at what is recorded by the Port Monitor. Here is a sample output:

```
...
COM4 (INFO): "On-the-Fly" command for 192.168.100.97: enabling line change notification for DSR, CTS...success
 <--- notifications setup
...
...
COM4 (INFO): "On-the-Fly" command for 192.168.100.97: set DTR to low...success   <--- command from VSP to DS
COM4 (INFO): "On-the-Fly" command for 192.168.100.97: set DTR to high...success   <--- command from VSP to
DS
COM4 (INFO): "On-the-Fly" command for 192.168.100.97: set RTS to low...success   <--- command from VSP to DS
COM4 (INFO): "On-the-Fly" command for 192.168.100.97: set RTS to high...success   <--- command from VSP to
DS
COM4 (INFO): Line status change notification: DSR:low CTS:high   <--- notification from DS to VSP
COM4 (INFO): Line status change notification: DSR:low CTS:low   <--- notification from DS to VSP
```

As you can see from this log, an individual command is sent from the VSP to the DS when it is necessary to change the state of a particular output. A notification containing the status of *both* inputs are sent from the DS to the VSP when *at least one* of the inputs (DSR or DTR) is found to have changed the state. Notifications are explained below (see "using notifications to get the status of I/O lines").

Setting and sensing the status of I/O lines using on-the-fly commands

Virtual Serial Port works with RTS, CTS, DTR, and DSR lines using so-called on-the-fly commands. On-the-fly commands are officially known as "network-side parameters and instructions". As the name implies, these commands are sent through the network. The word "on-the-fly" refers to the fact the sending any command causes an immediate corresponding change in the serial port or I/O pin of the DS.

Related to the discussion of controlling I/O lines are two instructions:

- **Set I/O Pin Status (Sx) instruction**
- **Get I/O Pin Status (Gx) instriction**

For these instructions to work, the DS must be preset to accept them. This is done by setting **On-the-fly Commands (RC) setting** to 1 (enabled).
On-the-fly commands are send just like all other network programming commands which means that delivery method can be out-of-band, inband, etc. Out-of-band on-the-fly commands can be optionally password-protected by programming **On-the-fly Commands (RC) setting** to 1 (enabled) and defining a password in the **Password (PW) setting**.

If you are planning to work with P2(DSR), P3(DTR), P4(CTS), or P5(RTS) lines you need to disable their "special" functions to turn them into "pure" I/O lines. When you control those lines:

- Disable RTS/CTS flow control by programming **Flow Control (FC) setting** to 0 (idle or remote).
- Program **DTR Mode (DT) setting** of the DS to 0 (idle).
- Also, to have the DSR work as a pure input line program do not set **Connection Mode (CM)** to 3 (on command or DSR=HI).

Note: when you are using the VSP you don't need to make these changes by yourself. Running Connection Wizard as described above will take care of everything.

We are not going into "much" details regarding the use of **Set I/O Pin Status** and **Get I/O Pin Status** instructions here as this would be a repetition of

information already found in the topics highlighted above.

Using notifications to get the status of I/O lines

Get I/O Pin Status (Gx) instriction mentioned above offers a way to get the state of any I/O pin of the DS but in a "polled" way. If you want to monitor the status of a particular pin you need to send this command repeatedly (i.e. "poll" this pin).

Another, more convenient, possibility is make use of **Notification (J) message**. With proper preset, you can make the DS send a special message to the PC whenever one of "monitored" pins of the DS changes its state. Read **Notification (J) message** topic and you will get complete picture on how this works.

One important point that needs to be understood: notification messages are only sent when there is a data connection between the network host (that is supposed to get notifications) and the DS. This is why we have asked you to change the Connection Mode of the VSP to "connect immediately". If you didn't do this you would have to have the data connection established first (i.e. send some data). Only after that the DS would start sending notifications.

** The only exception is pin P2 of EM100 which can only work as an input.*

AN006. Using Device Server Toolkit with Windows Firewall (XP/SP2)

What's in this Application Note

This Application Note explains how to use TibboDevice Server Toolkit (DST) on PCs running *Windows XP* with *Service Pack2* (*XP/SP2*). One of the new components included in the *SP2* is a *Windows Firewall*. Unless setup correctly, the *Firewall* will prevent certain features of *DST* software from operating properly.

This Application Note assumes that you are running *DST* version 3.56 (or later one). This version had certain adjustments made to enable *DST* use under *Windows Firewall*.

Contents:

- Using auto-discovery access mode of DS Manager with *Windows Firewall*
- Manual way of making the auto-discovery mode work
- Why *Windows Firewall* needs configuration for auto-discovery to work
- Opening the *Firewall* for DS data connections

Using auto-discovery access mode of DS Manager with Windows Firewall

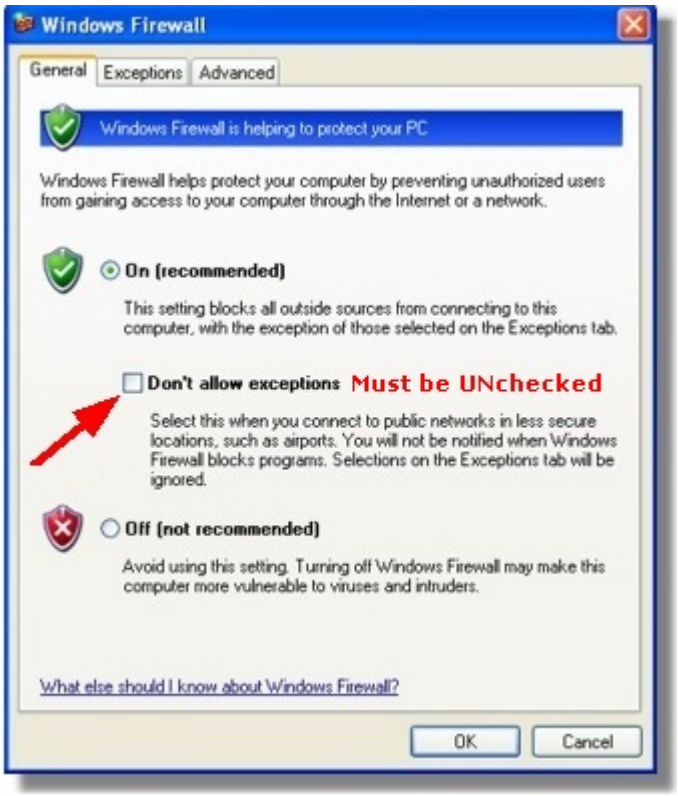
One difference you will immediately notice after *XP2* is installed and the *Firewall* is enabled is that the DS Manager can no longer find local Device Servers in the auto-discovery access mode (address book mode continues to operate properly). You can find an explanation on why this is happening later in this Note.

Once you run the *DS Manager* (or click *Refresh* in the auto-discovery mode), and provided that your *Firewall* is using default configuration, you will get this warning message:

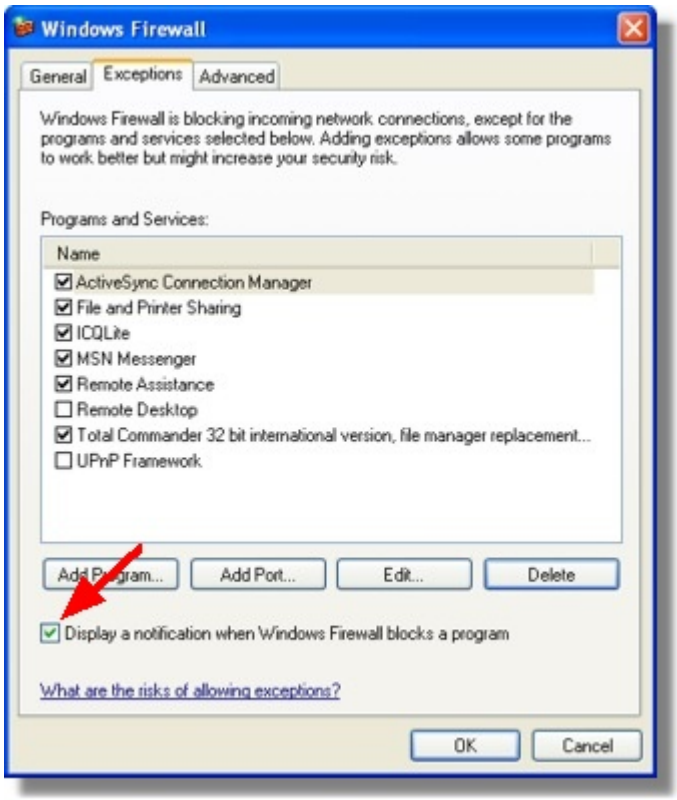
This message means that *Windows Firewall* has detected certain network activity that is currently not allowed. Application name is "Run DLL as an App" and, believe it or not, this is how *Windows* sees the *DS Manager* (yes, DS Manager is a "DLL" but *Windows* should be smart enough to understand its "real name".... alas, it isn't that smart). Click *Unblock* and the *DS Manager* will be able to auto-discover local Device Servers again.

If, when you run the *DS Manager* (click *Refresh*), the warning is not displayed (and the *DS Manager* is still unable to find Device Servers) then this may be because the *Firewall* is not allowing "exceptions" and/or firewall notifications are not enabled.

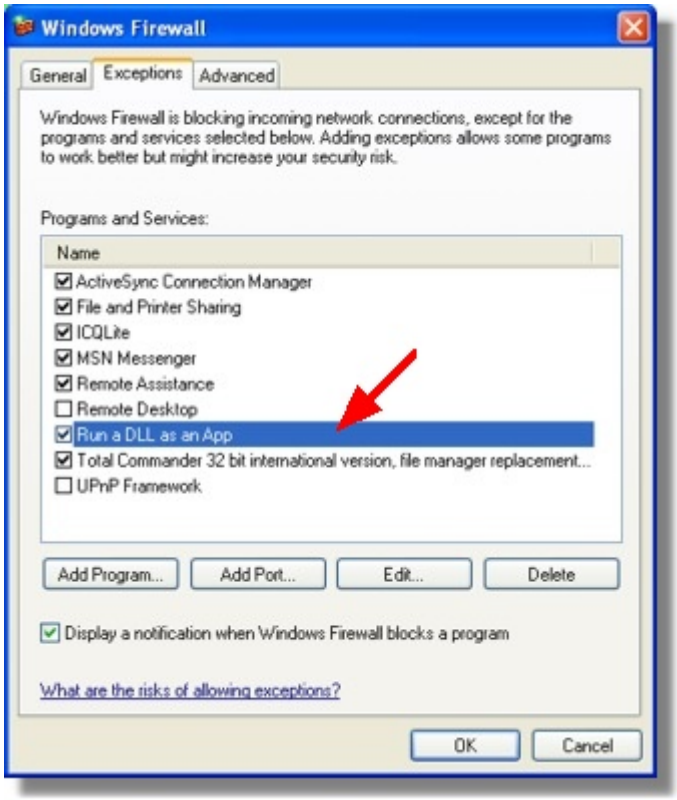
Run the *Firewall* (*Start--> Control Panel--> Windows Firewall*) and make sure that *Don't allow exceptions checkbox* is *un*checked (*clear*):



Next, click on the *Exceptions tab* and make sure that *Display a notification when Windows Firewall blocks a program checkbox* is checked:



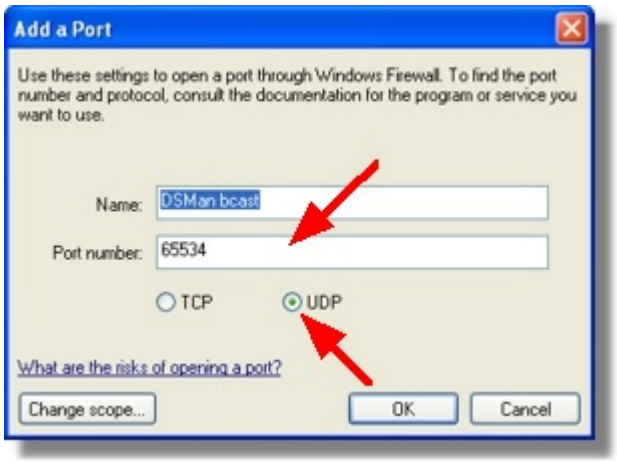
After you "unblock" the *DS Manager* the *Firewall* puts it into the list of "exceptions" i.e. programs whose traffic is allowed to pass through the *Firewall*:



Manual way of making the auto-discovery mode work

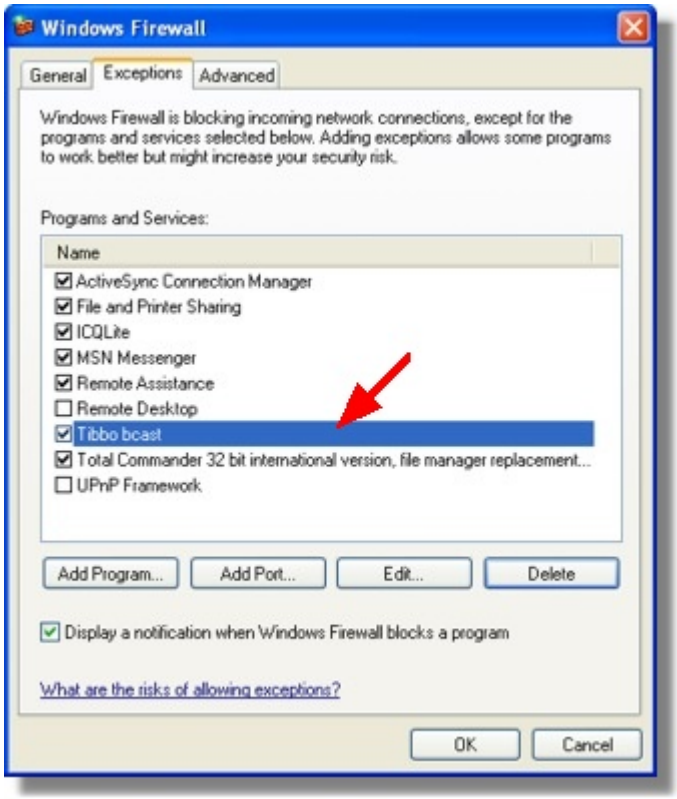
Same result can be achieved by telling the *Firewall* which port on the PC should be

opened. To do this click on the *Exceptions tab* of the *Windows Firewall dialog*, then press *Add Port... button- Edit a port dialog* will open:



Input any meaningful name into the *Name textbox* (i.e. "DSMan bcast"- this is because what we are opening here is a port for *DS Manager's* auto-discovery broadcasts to work). In the *Port number textbox* input 65534- this is the port number that must be opened on your PC. Finally, select *UDP*- this is a protocol the *DS Manager* is using to find Device Servers on the network. Click OK when finished.

New entry will appear in the list of exceptions and the *DS Manager* will start working properly:



Why Windows Firewall needs configuration for auto-discovery to work

This section explains why auto-discovery access mode will work only after you configure the *Windows Firewall* properly while address book access mode will

require no adjustment to the *Firewall* setup. You can happily skip this section if you are not that interested to know.

In the auto-discovery access mode the *DS Manager* finds Device Servers on the network by sending **Echo (X) command** as UDP broadcast. Every Tibbo Device that receives this broadcast will respond to it and the *DS Manager* will build a list of available Device Servers basing on received responses.

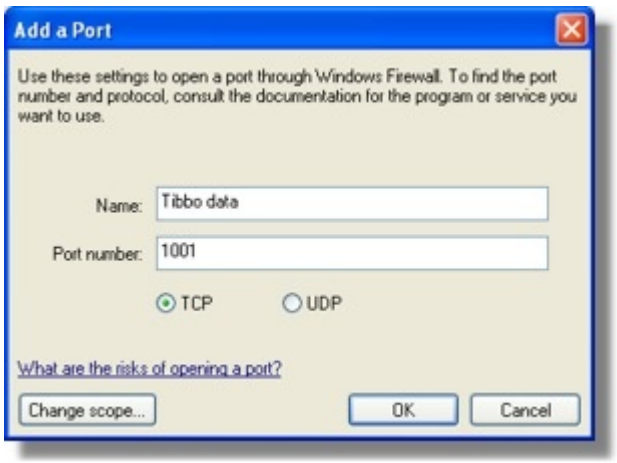
Trouble with *Windows Firewall* arises because the *Firewall* cannot link received responses to the broadcast sent from the PC. So, from the *Firewall*'s point of view these responses are individual incoming UDP "connections" and by defaults the firewall blocks any incoming connection unless it is explicitly allowed. By unblocking "Run DLL as an App" or manually opening UDP port 65534 you let the responses from Device Servers to pass through. Port 65534 is the port from which the *DS Manager* is sending its broadcasts and to which all local Device Servers send their replies.

Contrary to what you'd probably expected, *Windows Firewall* does not interfere with *DS Manager*'s operation in the address book access mode. This is because in this mode the *DS Manager* sends **Echo (X) command** individually to each DS in the *device list* (no broadcasts are used at all). When replies are received from Device Servers the *Firewall* is able to link those replies to the requests that were sent by the *DS Manager* earlier. The *Firewall* then assumes that these were outgoing connections initiated by a program on the PC and doesn't block incoming replies.

Opening the Firewall for DS data connections

Windows Firewall monitors all incoming connections and that means that if your DS is supposed to connect to the *VSP* on the PC (or directly to your application) then a specific port (to which the DS will be connecting) must be opened on the *Firewall*.

For example, if you know that the DS will be opening a TCP connection to the *VSP "COM3"* with listening port number 1001 then you need to "open" this port in the *Firewall*. Use *Add port...* feature to do this:



Notice, that you only need to open ports if your DS is going to connect to your PC. If it is the *VSP* (or application) on your PC that is going to connect to the DS then you don't need to setup the *Firewall*. This is because the *Firewall* doesn't block any connections that originate from within the PC.

AN007. Installing and Configuring LinkServer

This application note discusses the actual processes involved in installing and using LinkServer. If you work along with it, it will walk you through, step by step, in doing the following procedures:

- Preparing your Network (Router Configuration)
- Downloading and Installing Software and Firmware
- Creating Trial AuthKey
- Initial LinkServer Configuration
- Creating a User Account
- Adding DS as User
- Configuring Device Server
- Configuring Virtual Serial Port
- Testing with HyperTerminal

Each of these sub-topics can also stand on its own -- you can click on a sub-topic to see the step-by-step explanation on how perform that specific procedure.

The application note does not replace the LinkServer manual, but adds to it, and discusses the practical aspects of deploying LinkServer.

Preparing your Network (Router Configuration)

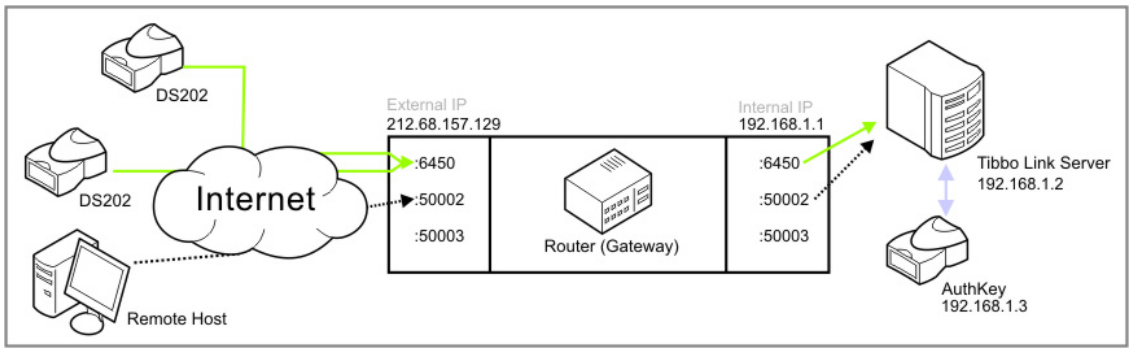
Quite possibly, you are going to use Tibbo LinkServer to communicate with Device Servers which are outside the subnet in which the server (host) running LinkServerresides. For instance, you might have an office LAN in which you're going to install LinkServer. However, you have dozens of Device Servers in the field you'd like to communicate with.

Your office LAN is connected to the outside world through a router. Only this router has a real IP address -- the rest of the LAN has internal, non-routable, IP addresses (such as the familiar 192.168.1.x). So, to get to the LinkServer which is inside your LAN, the Device Servers in the field would have to go through the router leading to (and from) your LAN. For this, you would need to use *port forwarding* on the router.

Port forwarding is a mechanism by which a router receives packets on a certain port, and *forwards* (passes) them to a specific IP address and port within its local network. For instance, a router can be configured that whenever it gets a packet to TCP port *585* (for instance), it should forward this packet to *192.168.1.100*, to port *6500*. Port forwarding can also apply to whole ranges of ports -- you can have your router forward everything that comes to ports 50000-60000 to ports 50000-60000 of a specific host within your network.

Of course, the packets do not merely *arrive* at the destination host within your network -- this host can also reply, and the router takes care of sending the reply back to the originating network host (or DS, in our case).

Exactly *how* to configure your router for port forwarding is something specific to the router -- you would have to consult the user's manual for your router. But the diagram below shows an example for such a configuration scenario. Here, we have DS203 devices spread throughout the Internet. We also have a remote host somewhere on the Internet. They all connect to the router's external IP, which is real (in this case, *212.68.157.129*), and the router handles things from there on.



The Device Servers connect to port *6450*, which is the default LinkServer login port for Device Servers. The remote host connects to port 50002, which is one of the ports assigned to Device Servers within the LinkServer, so it actually reaches a Device Server through this connection (via the LinkServer).

Notice that no port forwarding has to be set up for the AuthKey --the AuthKey communicates directly with the host running the LinkServer, and does not need direct communication with the outside world.

Downloading and Installing Software and Firmware

Naturally, before you can start working with the system, you need to obtain it and install its components:

Downloading the Needed Software and Firmware

- Go to http://www.tibbo.com/betazone.php.
- Navigate to the heading *Download LinkServer software*
- You will see two versions for your platform (*Linux* or *Windows*): One with "bundled JVM" and one without. *JVM* stands for *Java Virtual Machine*. If you're not sure whether you have JVM installed already, download the "bundled JVM" version.
- Next, you have to download the new firmware for all of your Device Servers. It is listed lower down the page. Make sure you download the appropriate firmware for your Device Server, according to the description.
- Now download the firmware for the AuthKey. It comes as an EXE file, titled something like AK_*xxx*D.exe (where *xxx* is the current version).
- Now download the latest *Device Server Toolkit*. You must use this version to test the LinkServer.

Installing the Software and Firmware

- Install the most recent *Device Server Toolkit* by clicking on the file you've downloaded and working through the wizard.
- Install the LinkServer in the same way (by clicking on the file and working through the wizard).
- Configure any firewalls you may have so they would allow access to TCP port 6450 and to TCP ports 50000 to 60000 (access should be allowed from the subnet in which the Device Servers are connected, and from the subnet in which the client hosts are connected).
- Note down the IP address of the server, and keep it from changing (buy a "real"

static IP for the server, if needed). You will use this information for Configuring a Device Server.

- Next, upgrade the Device Servers you would like to work with using the firmware you've downloaded, according to these instructions.

That's it! Now proceed to the next step (Creating Trial AuthKey).

Creating Trial AuthKey

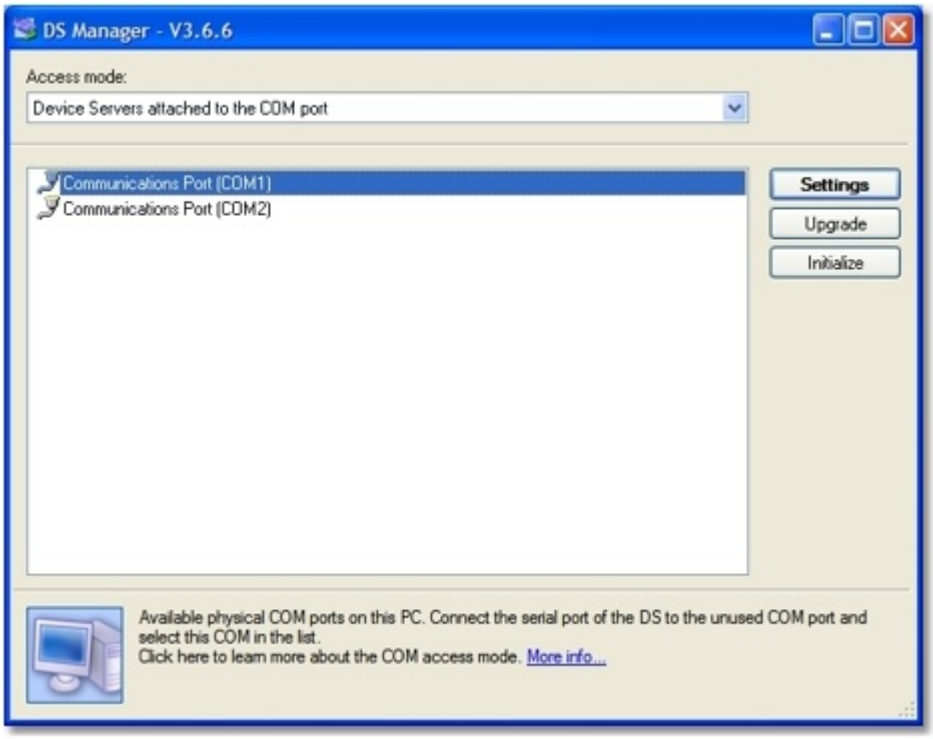
To start using the LinkServer, you first need to create a trial AuthKey. This is done using a DS203.

After upgrading, don't forget to perform Quick Initialization -- press the setup button, release it, and then press it again for 4 seconds.

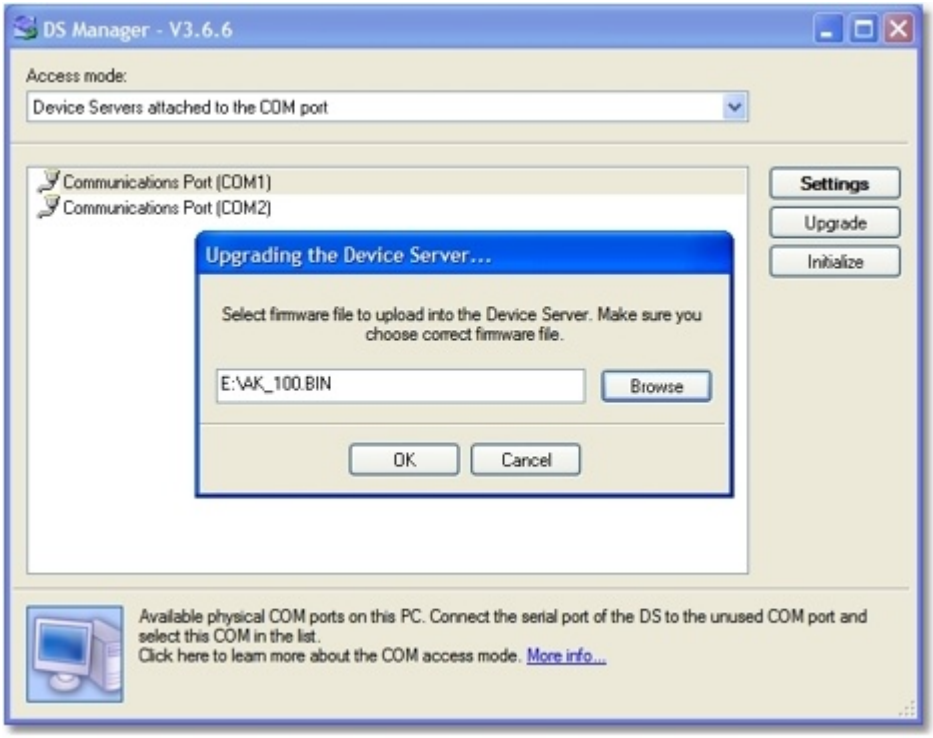
Perform the following:

- Connect the DS203 you want to convert to a Trial AuthKey to the computer, using a cross serial cable. Don't connect the power cable yet.
- Double-click the EXE file containing the AuthKeyfirmware -- you should have downloaded this, as described under Downloading and Installing Software and Firmware.

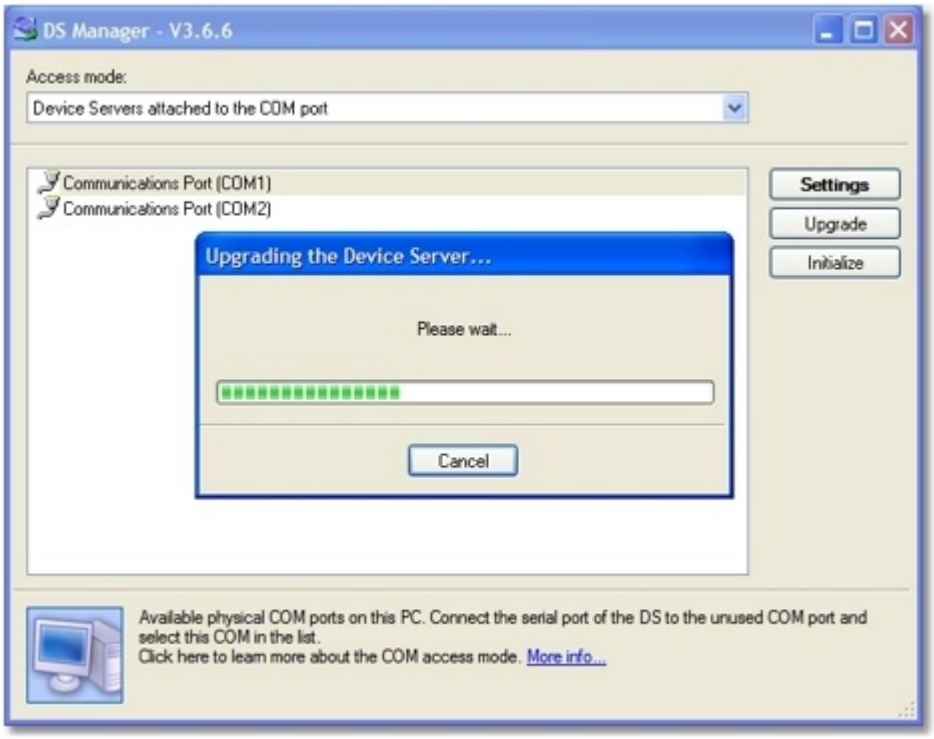
- Extract the file -- it contains a BIN file (firmware) and a README. Note the location of the BIN file.
- Run DS Manager.
- Under *Access Mode*, select *Device Servers attached to the COM port.*



- Click *Upgrade*.
- Select the BIN file.



- Now press the SETUP button on the DS203. After pressing the button, connect the power cable.
- The firmware will start transferring over to the DS203, and you will see a progress bar.



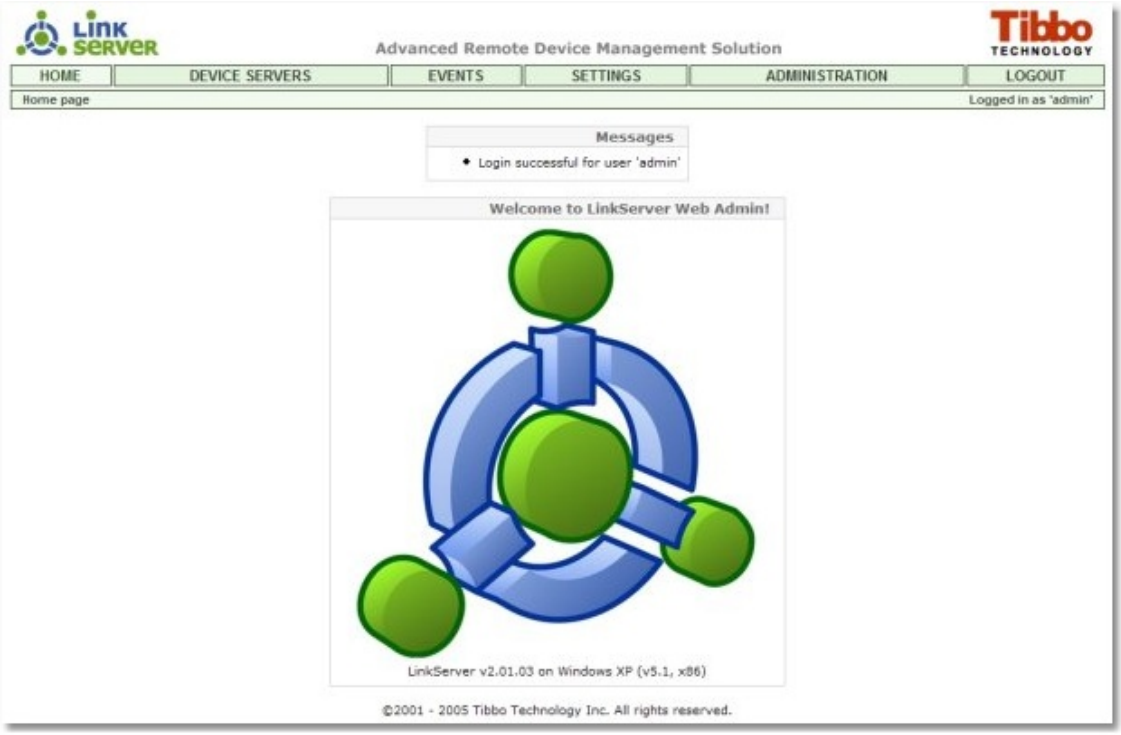
- Once you get a message that the process is done, disconnect the DS203 from the power.
- Power the DS203 again. Wait for a few seconds. Now you have to initialize it by Quick Initialization -- press the setup button, release it, and then press it again for 4 seconds.
- That's it! Don't forget to set an IP address for the AuthKey, and you're ready to move on.

Initial LinkServer Configuration

To begin configuring *LinkServer,* you must run it first. This is described under Startup and Shutdown in the LinkServer manual.

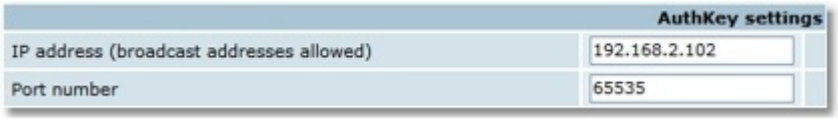
Once *LinkServer* is running, you should see the System Tray Icon.

Now, open the Web Admin interface as the default administrator, as described under Accessing Administrator Account. After logging in, you should see the following screen:



The first thing you should do is configure the AuthKey. Without a properly configured AuthKey, your Device Servers will not be able to connect to the server. Perform the following:

- In the top bar, click *Administration*.
- Under *Administration*, click *Global configuration*.
- Go to the bottom of the screen, and find the section titled *AuthKey settings*:



- Under *IP address* enter the address you've assigned to the AuthKey in th previous step (Creating Trial AuthKey).
- Under *Port number* enter the port which is listening on the AuthKey. The default value is 65535 -- make sure it corresponds with the port number in the AuthKey settings.
- Click *Save configuration and restart server*. After restarting, your server could connect to the AuthKey and use it to authenticate incoming connections. Further details can be found under AuthKey, in the LinkServer manual.

Creating a User Account

The next step is creating a user account (which is not an administrator account). You can read more about Account Types in the LinkServer manual. Perform the following, while logged on as administrator:

- Go to *Administration* > *User management*.
- At the bottom line of the top bar, click *Create new user account*.
- Enter a username and a password (repeat the password for verification). Click

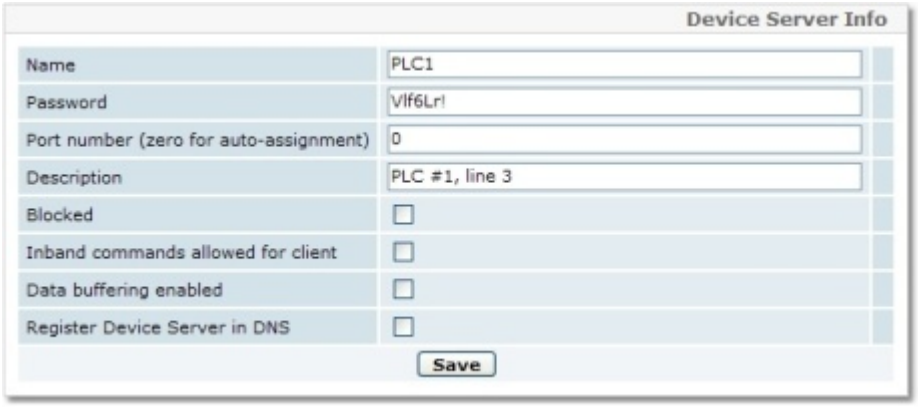
Save.

- Now log off the administrator account, and log on as the user you've just created. Do so by clicking *Logout* on the top bar.

Adding a DS as User

Before a Device Server can communicate with the LinkServer, it must be registered with the LinkServer. There are two ways to register a DS: automatically and manually. For our purposes, we will use the manual method:

- Before you begin, make sure you're logged on as the user you've created in the previous step. In our examples, this user will be called *johndoe*.
- Once logged on as *johndoe*, click *Device Servers* in the top bar.
- Next, click "Add Device Server" (Also on the top bar, one line below the main commands).
- The *Device Server Info* screen will appear:

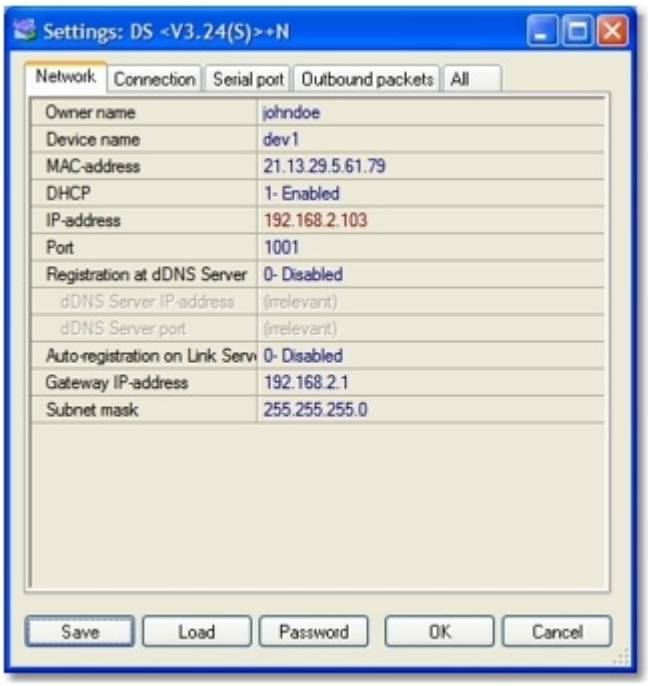


- Select a name for the Device Server which you will be able to easily recognize in the list. Note down this name for later.
- Select a password you will remember (for our example, this is *Vlf6Lr!*). It has to be at least 4 characters long. Remember this password, as you will need it in the next step (Configuring a Device Server).
- Leave the other entries at their default values.
- Click *Save*.
- You will now find yourself back at the *Device Servers* page, and your device will show up on the list. **Note down the number under the *port* column.** You will need it for later (for Configuring a Virtual Serial Port).

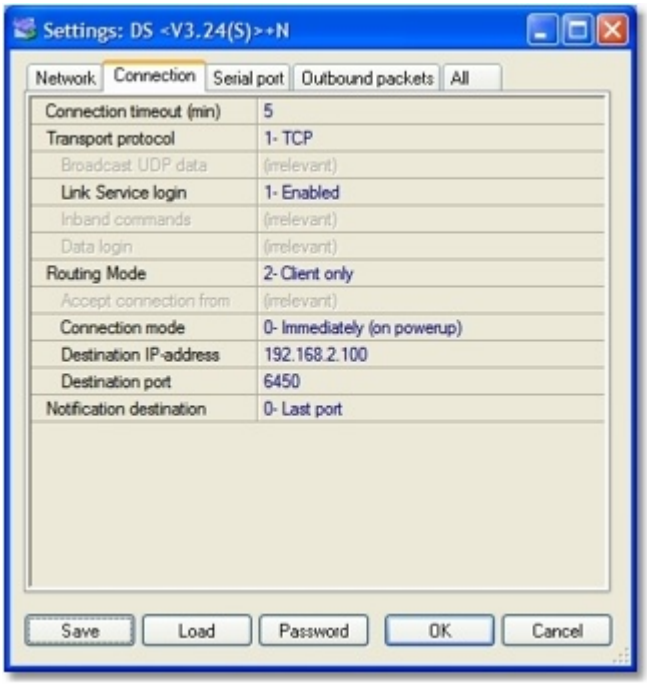
Configuring a Device Server

In this step you will configure a Device Server to connect to the LinkServer. Perform the following:

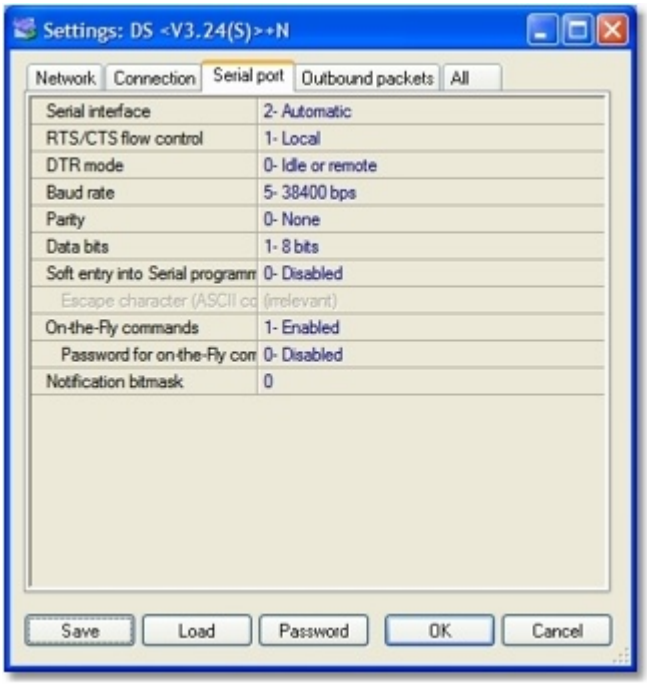
- Connect a Device Server to the network and power it.
- Run *DS Manager* (*Start* > *Programs* > *Tibbo* > *DS Manager*).
- You should see the Device Server in the list. If you can't see it, use the Address Book Access Mode.
- Select the Device Server and click *Settings*. This is the first tab you will see:



- Set *Owner name* to the same name as your account in the previous step (Adding a DS as User). In our example, this is *johndoe*.
- Set *Device name* to the same name as your Device Server in the previous step. In our example, this is *dev1*.
- Don't forget to make sure your Device Server has a valid IP -- by enabling DHCP, or setting a valid IP address manually.
- Set *Gateway IP-address* to the correct gateway for your subnet, so the Device Server could get reach the LinkServer (assuming the LinkServer is not on the local subnet).
- Click the *Password* button and set a password for the DS. This has to be the same password you configured for it in the previous step. In our example, this is *Vlf6Lr!.*
- Now, click the *Connection* tab:



- Set *Transport Protocol* to *TCP*.
- Set *Link Service login* to *Enabled*.
- Set *Routing Mode* to *Client only*.
- Set *Connection mode* to *Immediately (on powerup).*
- Set *Destination IP-address* to the IP address of the server (which you've noted down in step 1, Downloading and Installing Software and Firmware).
- Set *Destination Port* to *6450*.
- Now, click the *Serial Port* tab:

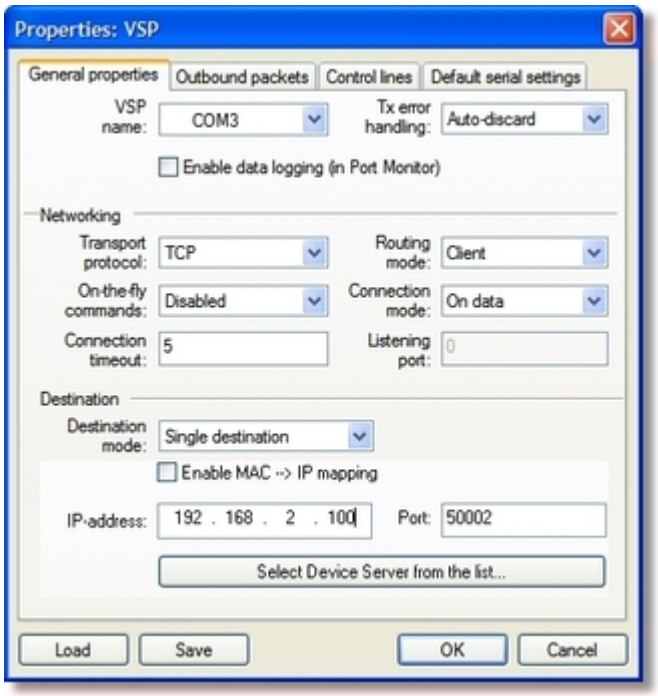


- Set the serial settings according to your device (*RTS/CTS flow control, DTR mode, Baud rate, Parity, Data bits*)
- Set *On-the-Fly commands* to *Disabled*.
- Click *OK* to apply the settings and close the dialog.
- *DS Manager* will reset the Device Server.
- Assuming you did everything correctly up to here (including all previous steps), the DS should now blink the green LED a few times and then light it steadily. This means that a connection has been established. If you get another LED pattern, find out what it is under Status LED Signals and try to troubleshoot it from there.
- If you now log-on to your account in the LinkServer, you will see the Device Server as Online under Device Servers.

Configuring a Virtual Serial Port

The next step, which is actually the very last configuration step, is configuring the *Virtual Serial Port*. This is done as follows:

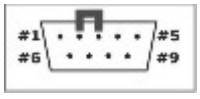
- Run *VSP Manager* (*Start > Programs > Tibbo > VSP Manager*).
- Click *Add*.
- If you get a *Windows XP* warning messagebox, click *Continue Anyway*.
- Set *Transport protocol* to *TCP*.
- Set *On-the-fly commands* to *Disabled*.
- Set *Routing mode* to *Client.*
- Set *Port* to the port which was assigned to your Device Server in the LinkServer. If you're not sure what this is, check it under the Device Servers page. This must be correct for the process to work.
- Set *IP-address* to the address of the host running LinkServer. Make sure the computer running the *Virtual Serial Port* can reach this host.
- The screenshot below shows the correct settings:



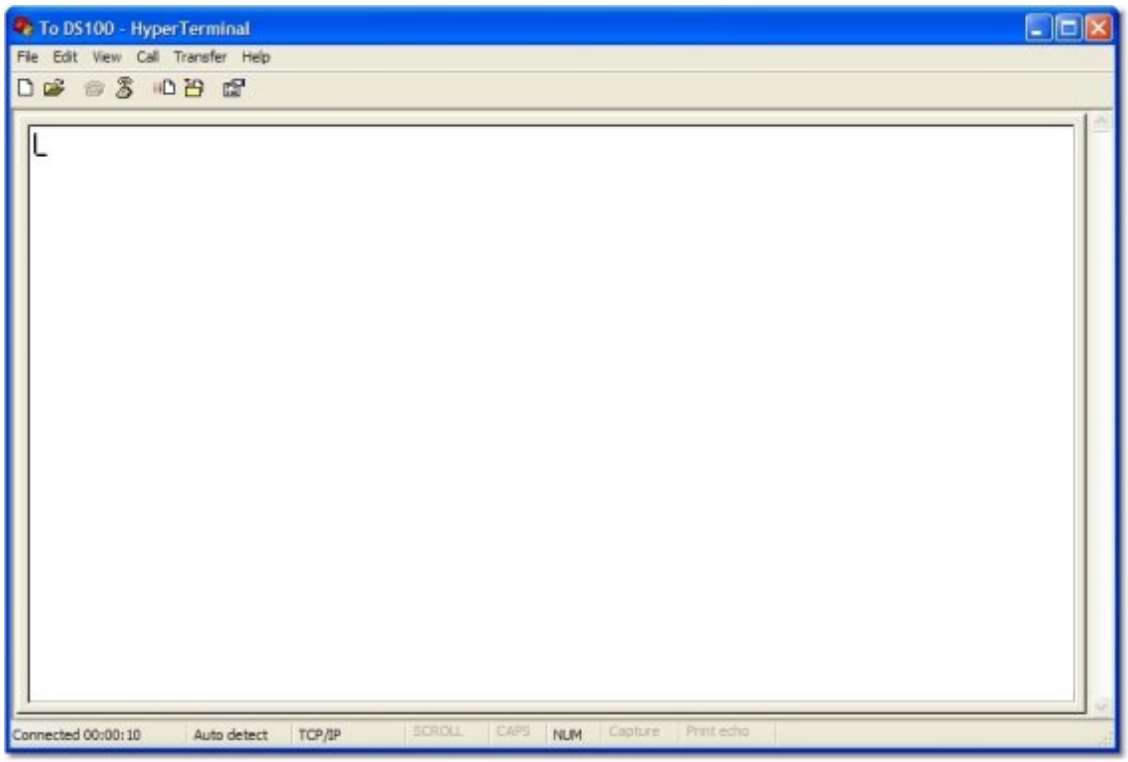
Testing with HyperTerminal

You will now test the connection you've created, using HyperTerminal. Perform the following steps:

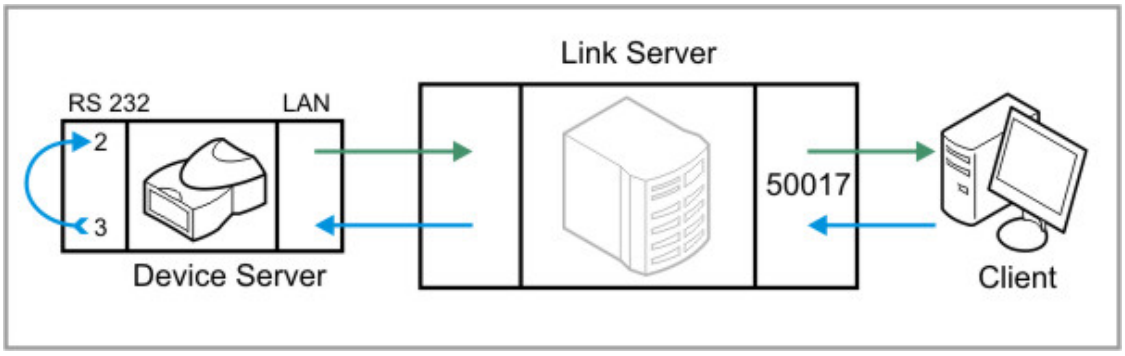
- Connect the DS you have configured previously (under Configuring a Device Server) to the network and power. Make sure the green LED lights steady -- to show you have a connection.
- Take this DS, and create a connection between serial pin #2 and serial pin #3 on its connector. This is just a temporary connection. Pin #3 is the output pin and pin #2 is the input pin -- so whatever gets out, immediately goes back in again. This is called a *loopback* and we will use it for the test. After the test the connection won't be necessary anymore, so don't solder the connector itself or anything of the sort. The connection should be like this:



- On the PC where you have configured the Virtual Serial Port, run *HyperTerminal*. Do this by clicking *Start* > *Programs* > *Accessories* > *Communication* > *HyperTerminal.*
- Choose a name for the connection.
- in the next screen, under "connect using", select the Virtual Serial Port you have created (such as COM6).
- On the next screen select any serial settings you would like, but make sure *Flow Control* is set to *None*.
- Click OK. You should now get a white screen where you can type:



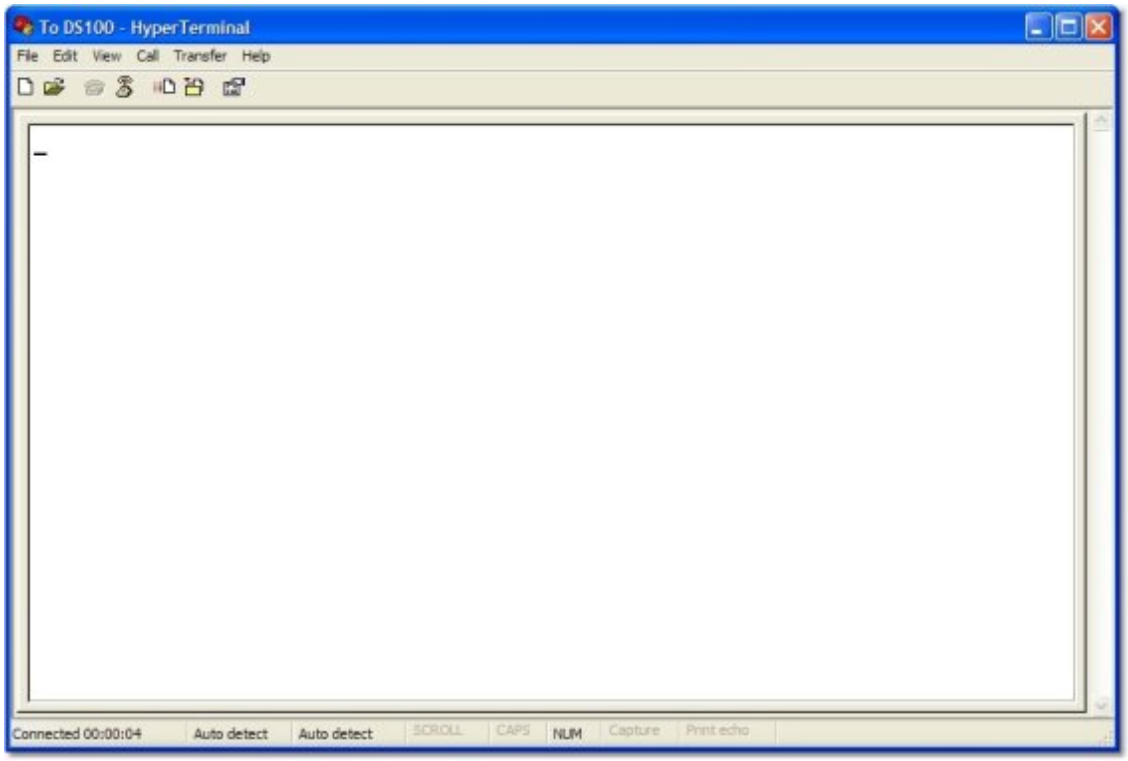
- Type any text on the keyboard. You should see it on-screen now. If you see what you type, it means that the following sequence happens:
  - The HyperTerminal session sends the data to the Virtual Serial Port.
  - The Virtual Serial Port sends the data to the LinkServer.
  - The LinkServer sends the data to the Device Server.
  - The DS gets the data and immediately sends it back (using the loopback we created in the beginning of this step).
  - The LinkServer gets the data from the DS and sends it back to the host running HyperTerminal.
  - The Virtual Serial Port gets the data back, and passes it to HyperTerminal.
  - You see the data you type on-screen.



The simplest way to make sure that this is really what happens, is to remove the loopback from the DS. You will continue sending data but you will not get anything back -- this means it worked!

AN008. Using HyperTerminal

HyperTerminal is a communications program, which comes installed by default on almost every type of Windows Operating System, from Windows 95 and up to Windows 2003 Server (and probably for future version as well). The main screen for the program looks like this:



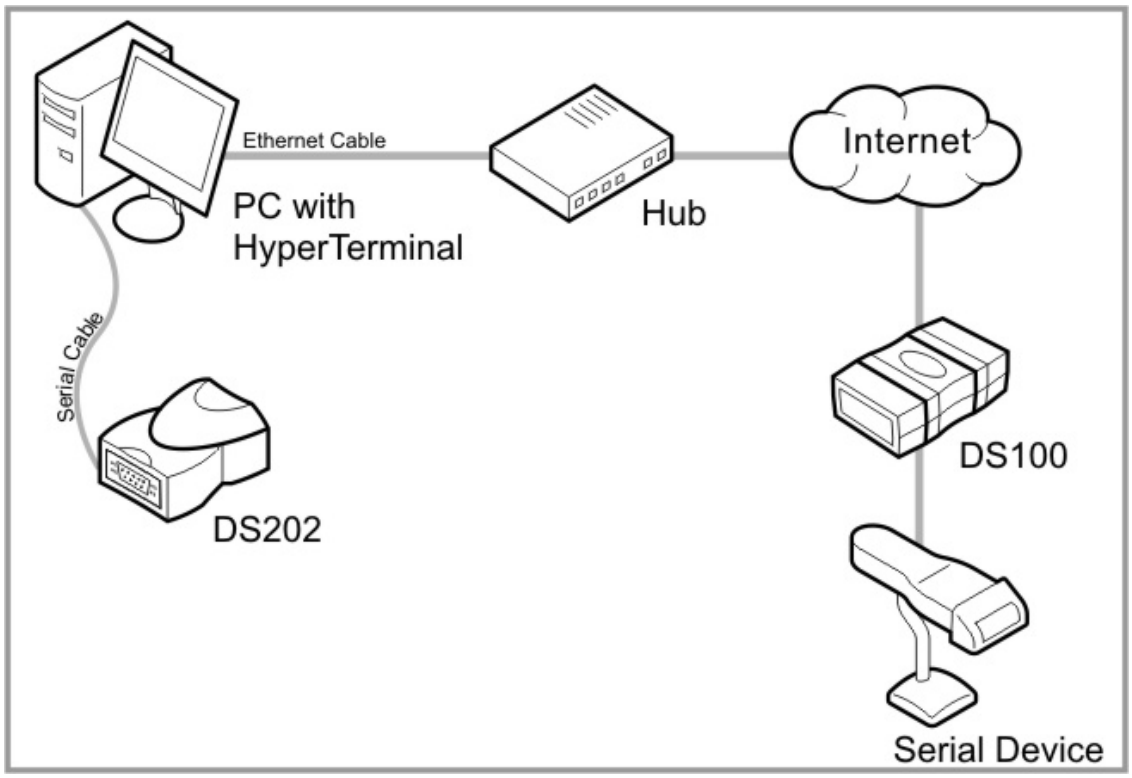
Basically, HyperTerminal lets you type characters (or send files in specific protocols) from your computer, which is acting as a TERMINAL (computer originating or receiving communications), to another computer or device (such as a DS100) which is connected to it.

Communication Methods

The 'classic' use of a terminal program is for serial communications. And indeed, the 'natural' use of *HyperTerminal* is for serial communications, and it is best suited for this purpose. However, with time, *HyperTerminal* was greatly extended, and today allows also for TCP/IP communication (telnet) in addition to serial and modem communication.

Think of the TCP/IP mode as 'extended functionality'. Any instructions in this Application Note which relate to this mode of communication, can also be performed using any regular telnet program. They do not utilize any 'special' capabilities of HyperTerminal -- it just serves as a generic telnet program.

The following diagram illustrates the typical two uses of HyperTerminal in relation to Device Servers. We have a workstation, running HyperTerminal. HyperTerminal is used to communicate with a nearby DS203 over a serial cable, and it can also be used to communicate, through a firewall, with a remote DS100 which is connected to the Internet and has a serial Barcode reader attached to it (the user can even access the output of the Barcode reader using HyperTerminal with this configuration).



One very important thing to note about HyperTerminal is that you do *not* see what you type on the screen. What you type is sent to the other end of the line, and the screen is used to show *incoming* data from the other end of the line -- i.e, you only see the replies you get (if you're getting any replies).

However, this default behaviour can be changed. The procedure for changing it is described under Setting Optional Parameters.

What is it Good For?

As you probably already know, almost every serial device comes with its own proprietary communication software, optimized to its needs. The same is true also for the Device Server -- it comes with the Device Server Toolkit, ordinarily used to communicate with it. This, then, begs the question: What do we need HyperTerminal for?

Well, the answer is quite simple: *Testing.* HyperTerminal's strength lies in its simple interface. You just type your commands in, and watch the raw output on the screen. There aren't any buttons to click, or actions which are done without the user knowing it. This is very close to 'raw' communication -- just your input and the device's output, with no software to interpret it in the middle.

This lets you answer very quickly questions such as "what happens when I send..." -- and this, in turn, helps in the development of applications which will communicate with the Device Server directly. Before writing a whole routine in Visual Basic or another language just to send a specific command to the device, you can first send the command yourself, manually, and see what happens in real-time. Then you'll be able to write your code in full confidence that you're doing the correct thing.

Another common use for HyperTerminal is *troubleshooting*. HyperTerminal accesses the serial port in a very standard way. This means that it can be used when the proprietary application software for the serial device cannot open the Virtual Serial Port, or when communication fails in some other way. You can just run HyperTerminal and play with it, to see if the COM port is indeed opened, if communication reaches the other side of the line, if you get a reply, etc.

Such testing would help you decide if the problems you're having are related to software, hardware, network connectivity, etc.

Running HyperTerminal

There are two common ways to launch HyperTerminal:

Using the Run Dialog:

- Click *Start*.
- Select *Run*.
- Type e óéÉáéã KÉñÉ
- Press båíÉê.

Using the Start Menu:

- Click *Start*.
- Go to *Programs* > *Accessories* > *Communications* > *HyperTerminal*
- In *Windows 2000/XP*: *HyperTerminal* would run.
- In *Windows 9x* (*95*, *98*, etc): A program group called *HyperTerminal* would open. Click *HyperTrm* to run HyperTerminal.

Setting Correct Parameters on Startup

If this is the first time you're running *HyperTerminal*, *Windows* will ask several questions regarding your location. Your answers are later used for modem connections -- these settings aren't necessary for direct serial or TCP/IP communication (the types of communication used for the Device Server).

After answering the questions regarding location (or if the location is already configured), you might get the following dialog:

As covered above, *HyperTerminal* can be used for telnet communications, and can also be the system default for telnet communication (i.e, the program which the system runs by default whenever telnet is needed). This setting, too, isn't vital to our purposes. Select whichever setting seems appropriate to you.

The next dialog is often the first dialog, in systems which are already configured. It is titled *Connection Description*, and looks like this:

Here you can select a descriptive name for your connection (such as "To DS100"), and also an icon for the connection. Select these settings and click OK.

The next dialog, *Connect To*, is used to configure the communication method and destination for this connection:

By default, HyperTerminal tries to configure a connection using a modem (if one is installed). However, we don't need a modem for our configuration. As covered above, we usually need one of two connection types: TCP/IP connection, or a direct serial connection.

Naturally, you would need a device server for this, and you have to connect it to a power source (Power supply) which is appropriate for it.

You may skip to the section dealing with the connection type you'd like to establish:

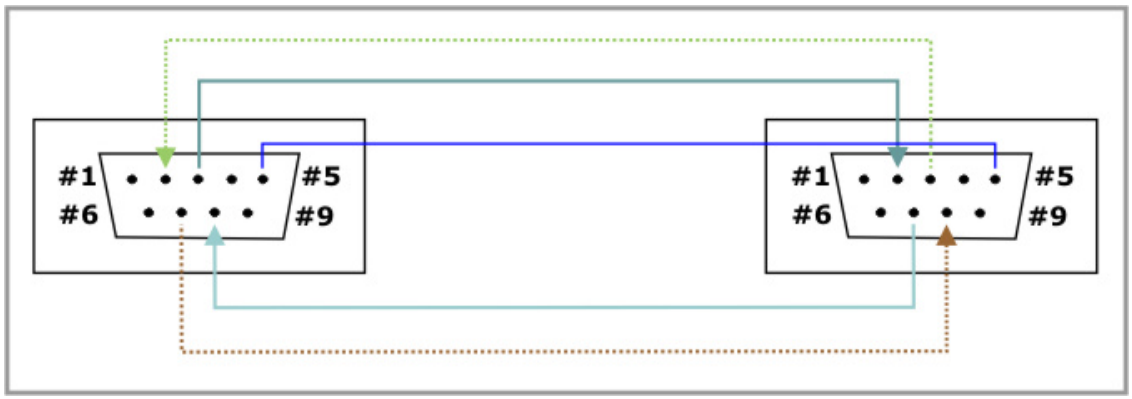
- Establishing a TCP/IP Connection with a Device Server
- Establishing a Serial Connection with a Device Server

Establishing a Serial Connection with a Device Server

This section covers connecting the Device Server serially to the computer, using a cable, and configuring *HyperTerminal* to connect to it.

Selecting The Correct Cable

The cable used for direct serial communication with the Device Server should be a *cross cable*. This means pin #2 goes to pin #3 on the other end. And #3 goes to #2, and #7 goes to #8, and #8 goes to #7:



Like so. Both of the connectors for this cable should be "female". Tibbo makes exactly such a cable, called WAS-P0005(B) DS-to-PC Serial Cable.

Configuring HyperTerminal

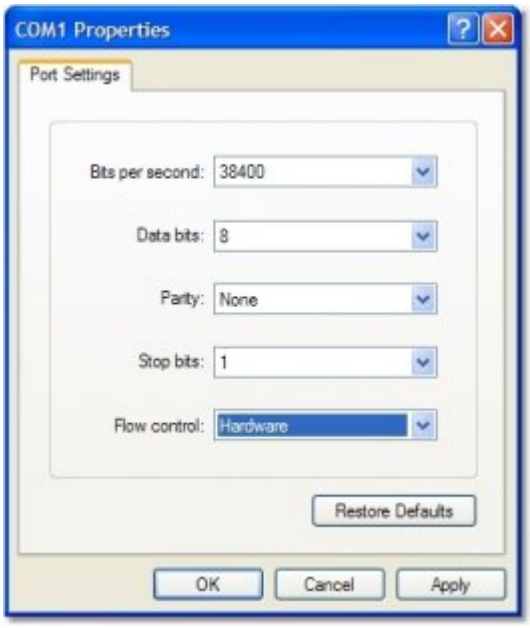
After connecting the proper cable to the Device Server and to your computer, it's time to continue configuring *HyperTerminal*. We continue from where we left off on Setting Correct Parameters on Startup. In the next window, open the *Connect using* drop-down, and select *COM1* (assuming you connected the DS to COM1):

The next screen deals with serial settings. Here you have to select the correct serial parameters.

What are the *correct* serial parameters?

When talking about *correct* serial parameters, it is important to understand two things:

1) **If you would like to program the DS serially,** the correct settings are 38400, 8, N, 1 (like in the screenshot below). These are the correct settings for the Serial Programming method.



2) **If you would like to simply communicate using the DS,** the correct settings are those which were configured using the *DS Manager* or *Connection Wizard* for this DS. They can be 38400, 8, N, 1 but they can also be different.

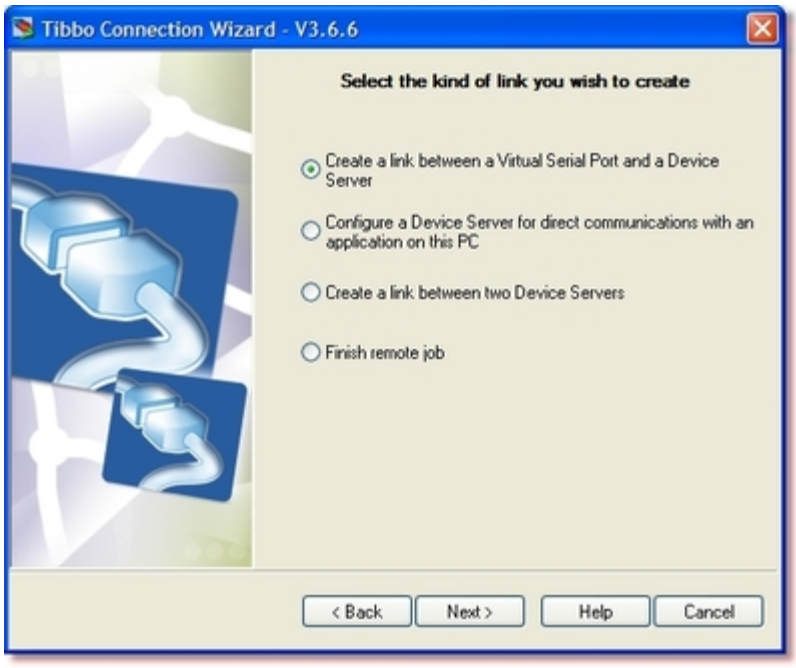
After setting the correct parameters, click OK. That's it! You've now established a serial connection with the DS100. Continue on to Sending Commands To the Device Server or Using HyperTerminal to Test a Connection.

Establishing a Connection Through a Virtual Serial Port

In this section, you will see how to create a *Virtual Serial Port* using the *Connection Wizard*, associate it with a DS, and then connect to it using *HyperTerminal*.

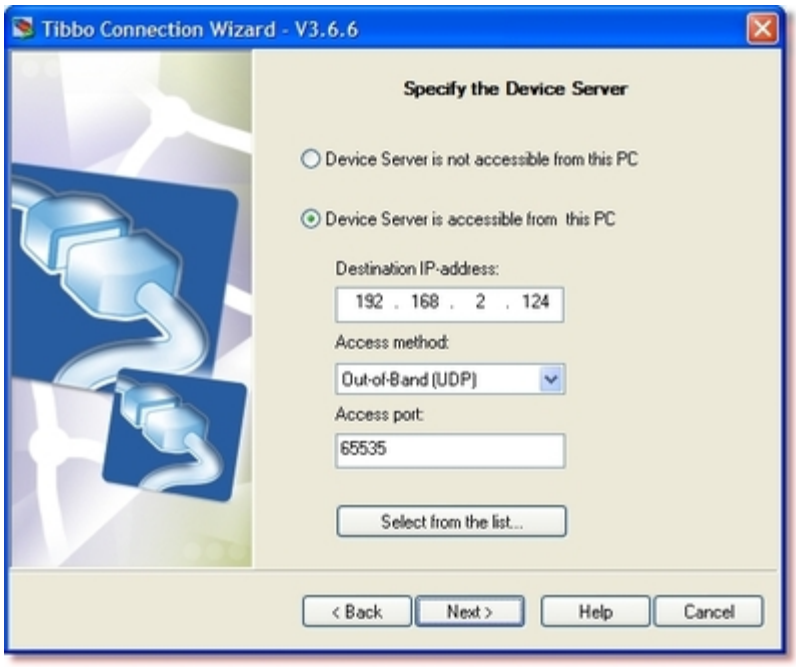
- Click *Start* > *Programs* > *Tibbo* > *Connection Wizard:*

- Click *Next*.

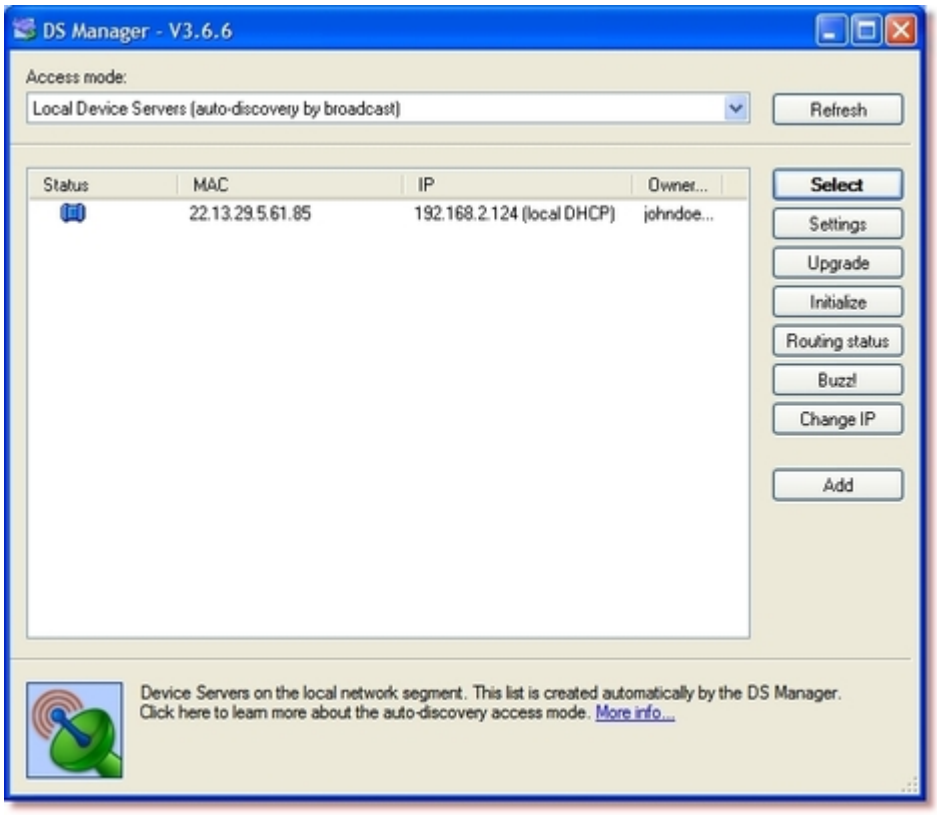


- Select *Create a link between a Virtual Serial Port and a Device Server.* Click *Next.*

- Select *Create new VSP* and select the *Port name*. Remember what port you created, you will need it for later. Click *next.*

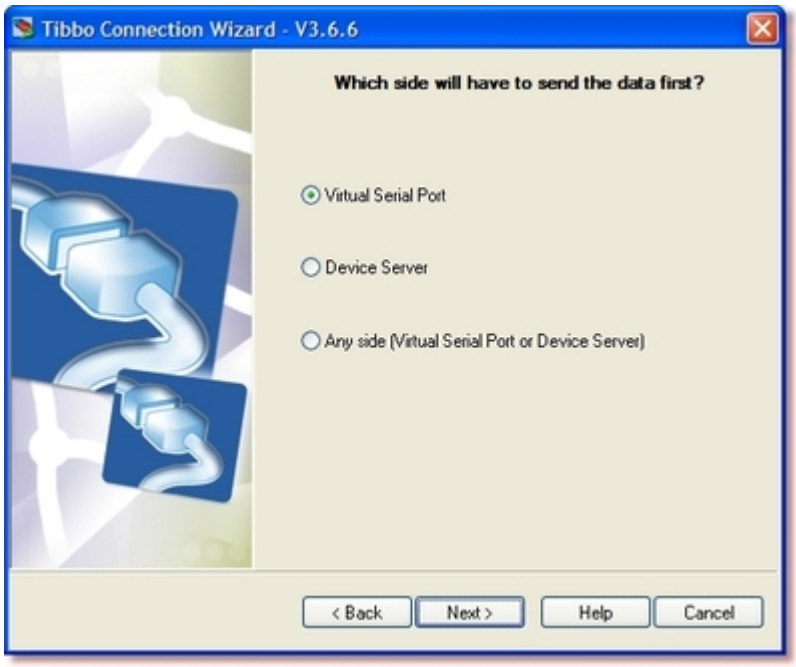


- Click *Select from the list*.

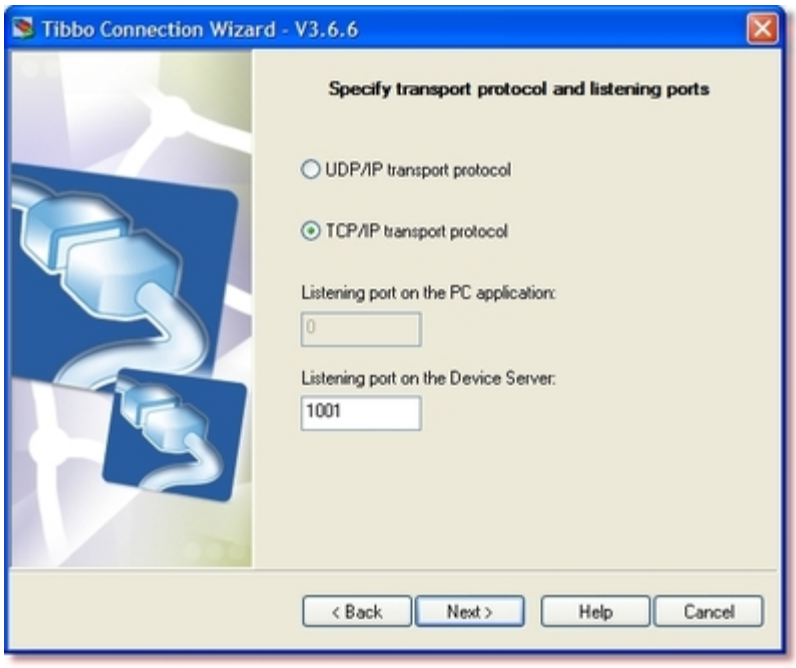


- Select the DS you would like to work with. If its IP address is invalid (it says so on the status bar), click *Change IP*.

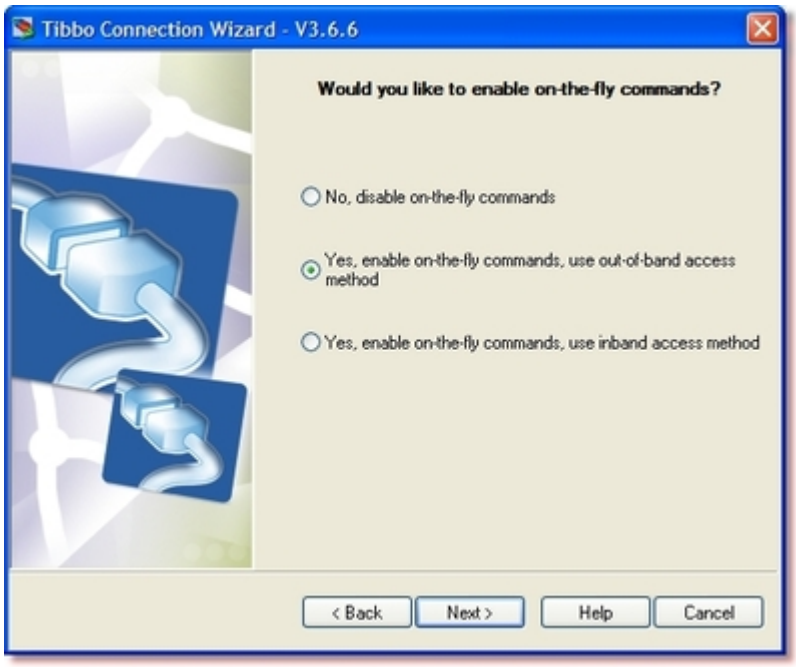
- Having made sure the IP of the device is correct, select the DS in the list, click *Select* to go back to the Wizard, and click *Next*.



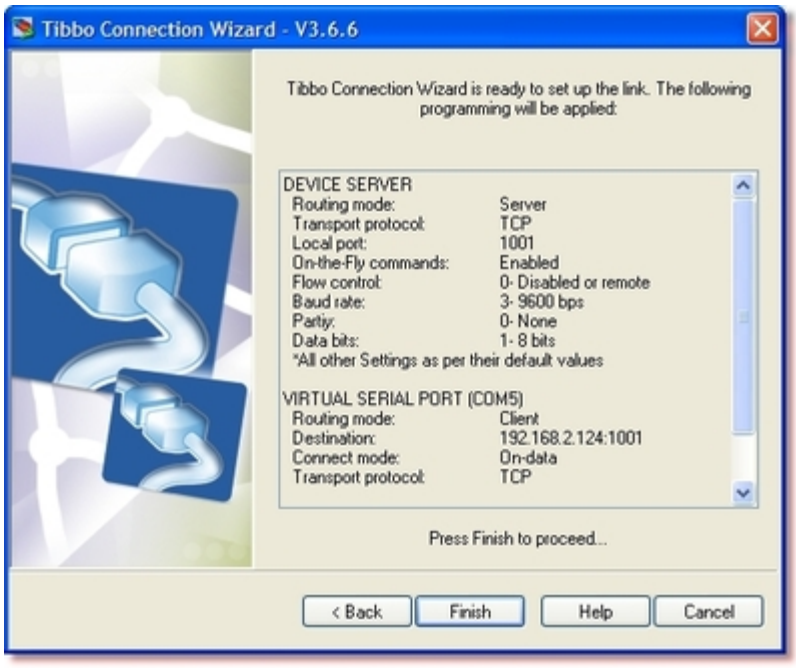
- Select *Virtual Serial Port.* Click *Next.*



- Select *TCP/IP transport protocol*. Don't change the port, but remember it's 1001. You'll need this for later.



- Select *Yes, enable on-the-fly commands, use out-of-band access method.*



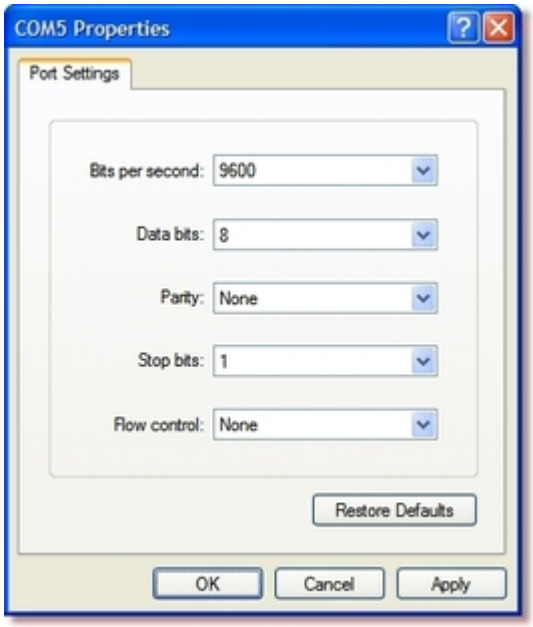
- This is just a summary screen. Click *Finish.*

Configuring HyperTerminal

Now that you have a Virtual Serial Port, it's time to continue configuring *HyperTerminal*. We continue from where we left off on Setting Correct Parameters on Startup. In the next window, open the *Connect using* drop-down, and select *COM5* (assuming created a VSP under COM5):

The next screen deals with serial settings. Here you have to select the correct serial parameters. In this case it's not so important what you choose, but keep in mind the following:

- If you're going to connect a serial device for testing with *HyperTerminal*, of course the settings you choose here much be compatible with this device.
- If you are going to perform a loopback test, set *Flow control* to *None*:



After setting the desired parameters, click OK. That's it! You've now established a connection with the DS100 via the *Virtual Serial Port*. Continue on to Using HyperTerminal to Test a Connection.

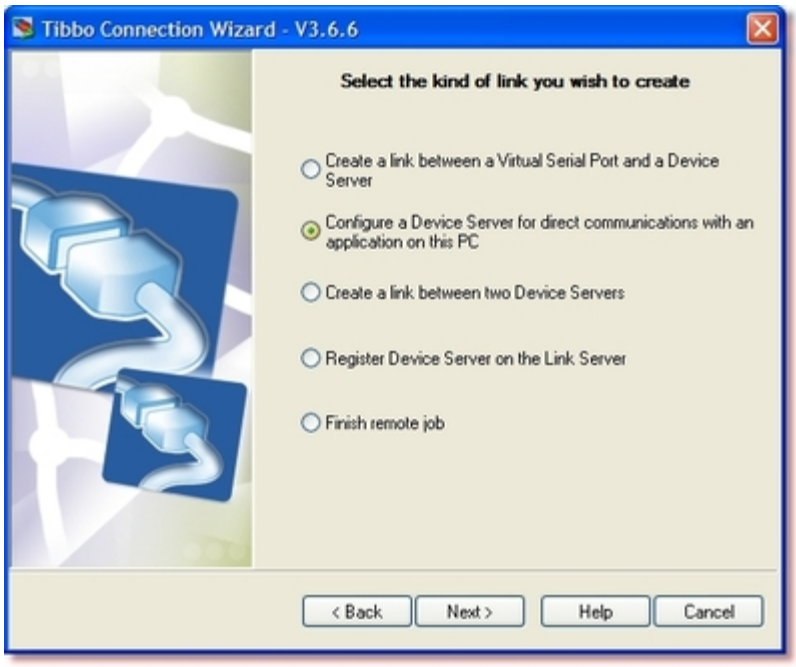
Establishing a TCP/IP Connection with a Device Server

Setting up the Device Server for TCP/IP Communication

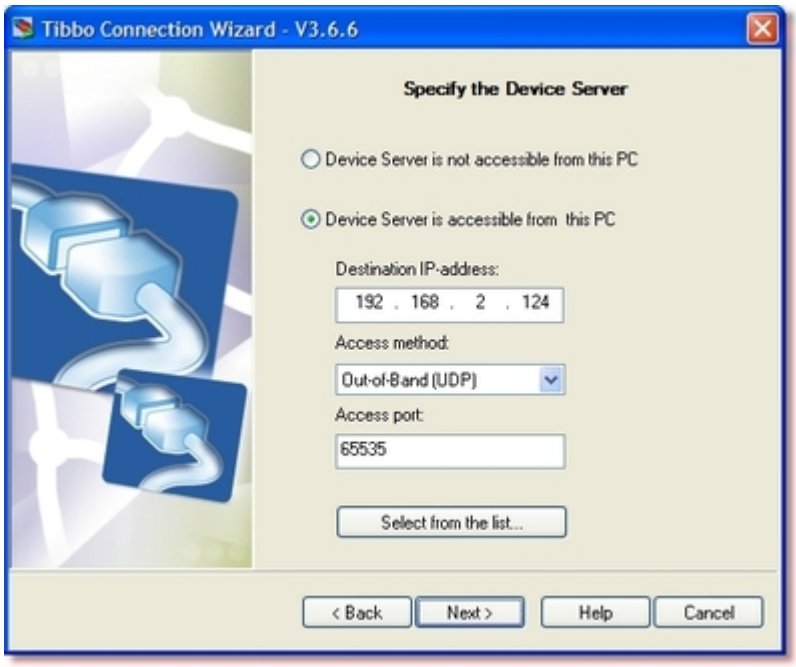
The simplest and most reliable way to set the DS up for TCP/IP communication is using the *Connection Wizard*.

- Click *Start* > *Programs* > *Tibbo* > *Connection Wizard:*

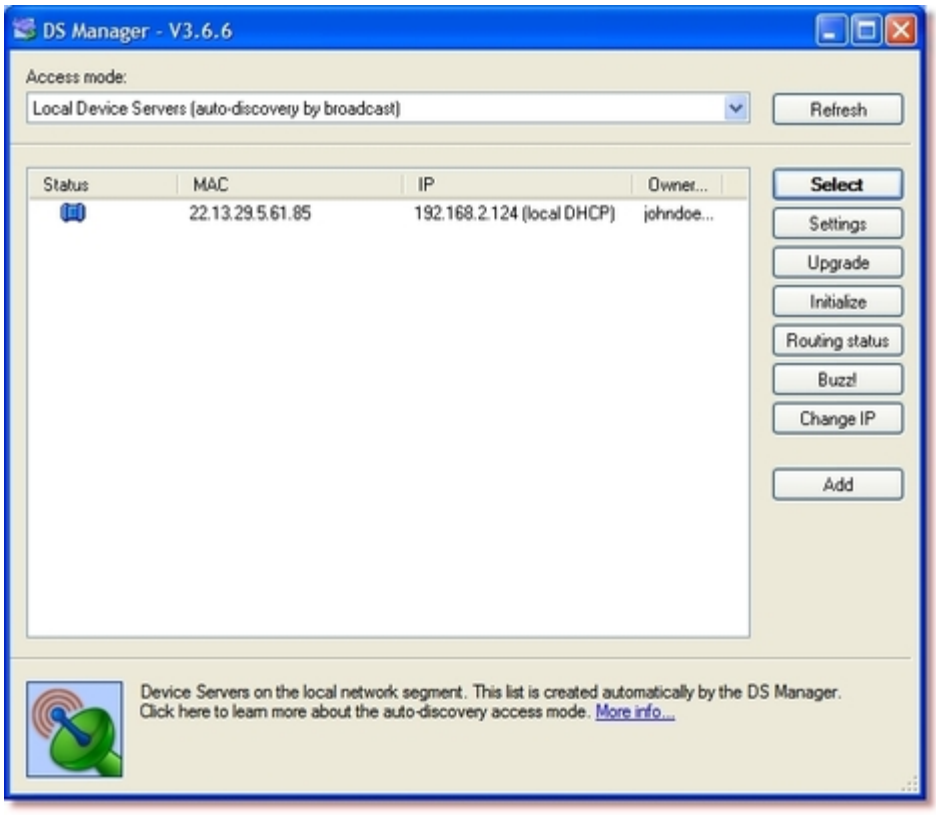
- Click *Next*.



- Select *Configure a Device Server for direct communications with an application on this PC*. Click *Next.*

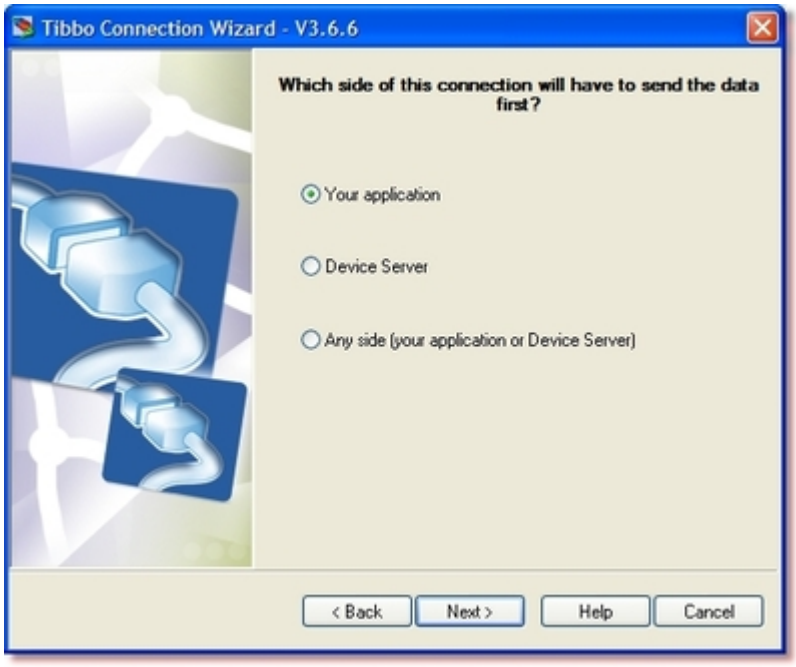


- Click *Select from the list*.

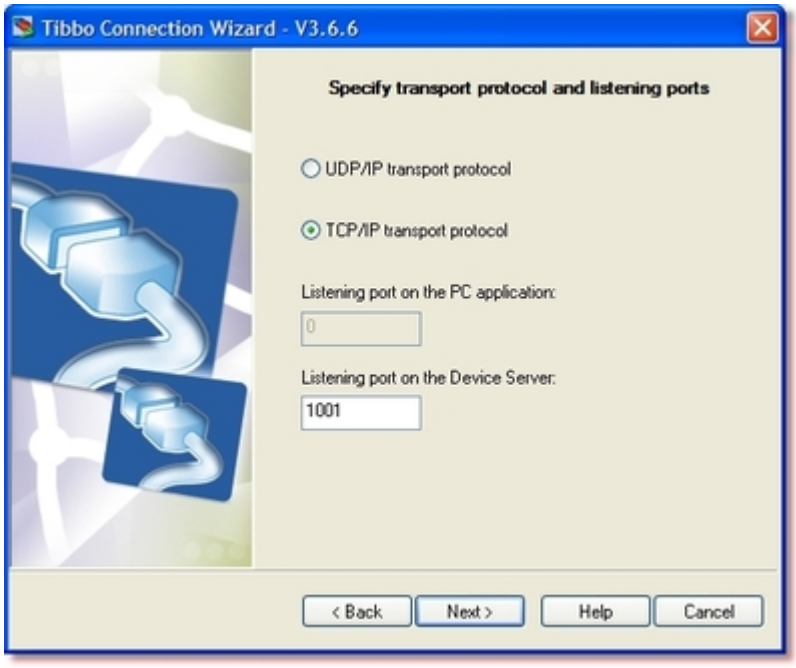


- Select the DS you would like to work with. If its IP address is invalid (it says so on the status bar), click *Change IP*.

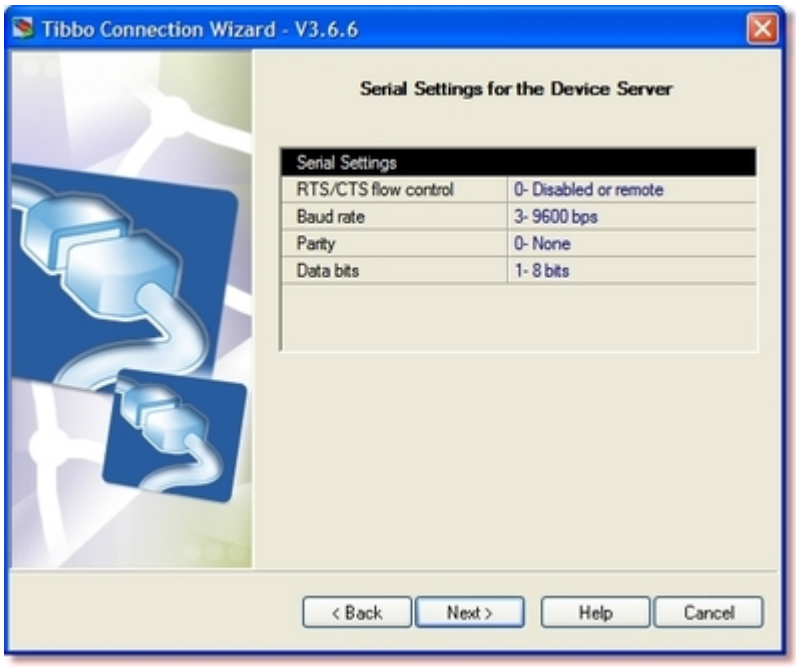
- Having made sure the IP of the device is correct, select the DS in the list, click *Select* to go back to the Wizard, and click *Next*.



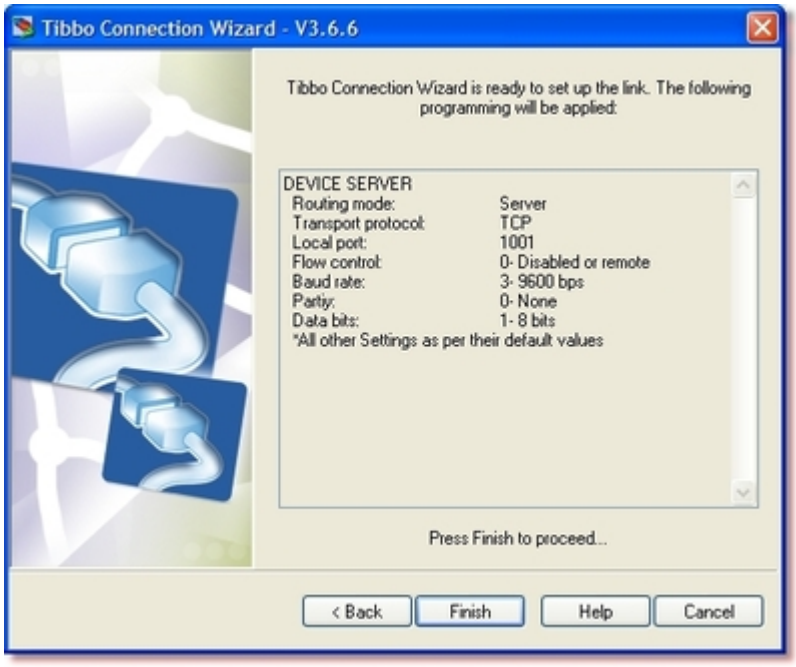
- Select *Your application.* Click *Next*.



- Select *TCP/IP transport protocol*. Don't change the port, but remember it's 1001. You'll need this for later.



- Set whatever serial parameters you need. Just remember what you set here, as you might need it for later (or for Establishing a Serial Connection with a Device Server). If you are going to perform a loopback test, set *RTS/CTS flow control* to *Disabled or remote*.



- This is just a summary screen. Click *Finish.*

Now we know the IP address and port for this device server, and have set it so it would listen for incoming TCP connections. On to the next phase:

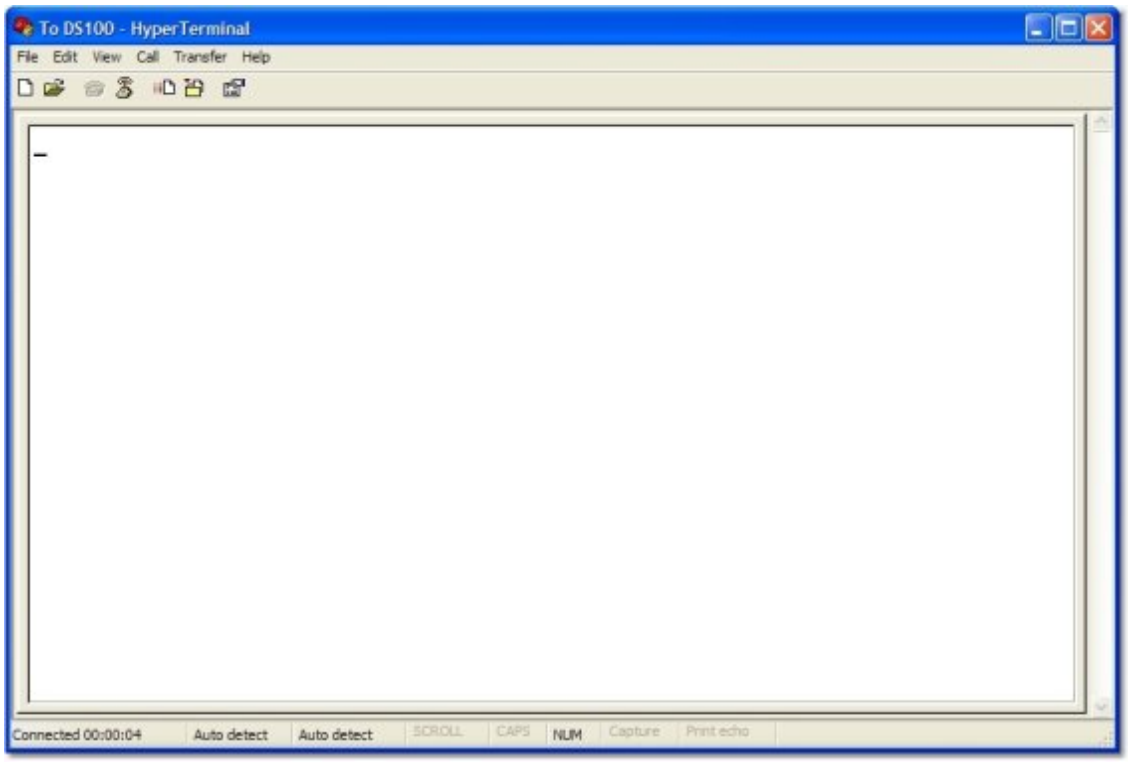
Setting up HyperTerminal for TCP/IP Communications

We resume from where we left off on Setting Correct Parameters on Startup. To establish a TCP/IP connection with the Device Server, open the drop-down titled *Connect using*, and select *TCP/IP (Winsock)*:

Next, enter the IP address and the port number we got earlier from the DS Manager:

Now click *OK*.

The following screen should appear:



Congratulations! You're connected. Continue to Sending Commands To the Device Server or Using HyperTerminal to Test a Connection.

Setting Optional Parameters

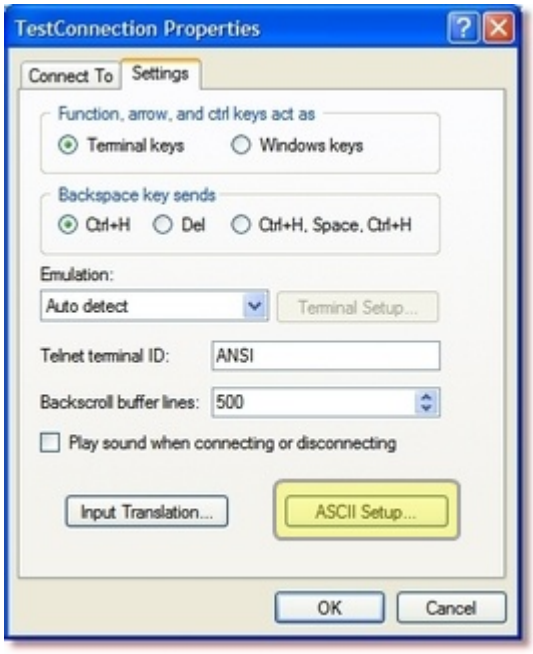
Having established the connection, you may now configure *HyperTerminal* so that it would display the characters you are typing, and also add automatic line feed characters (i.e, move one line down) whenever necessary, to make the communication easier to read.

This is also useful for making sure you're sending the correct commands. Otherwise, when the DS returns an error code, you may be left wondering what went wrong. Enabling a local echo of sent commands will show you exactly what the DS received just before it returned an error code.

To enable local echo, perform the following steps:

- After establishing a connection (as described in the previous sections), go to *File* > *Properties*:

- Switch to the *Settings* tab, and press the *ASCII Setup* button:



- In the *ASCII Setup* dialog, mark the options *Send line ends with line feeds, Echo typed characters locally* and *Append line feeds to incoming line ends:*

- Press *OK* to confirm all of the different dialogs, until you reach the main screen again.
- You can also save your current configuration, so that next time you wouldn't have to set it up again. Do this by clicking *File* > *Save*. Next time, run HyperTerminal using the file created by the saving process (it is named after your current session name -- *TestConnection* in our example).

Using HyperTerminal to Test a Connection

One of the common uses for *HyperTerminal* is just to see if a connection is alive. To test your equipment, or to test specific settings. Following are two procedures:

- Using *HyperTerminal* with Two HyperTerminal Windows, one DS (connecting both

the serial cable and the network cable to the computer).

- Using *HyperTerminal* with a loopback connection, as described under Creating a Loopback for Testing.

Two HyperTerminal Windows, one DS

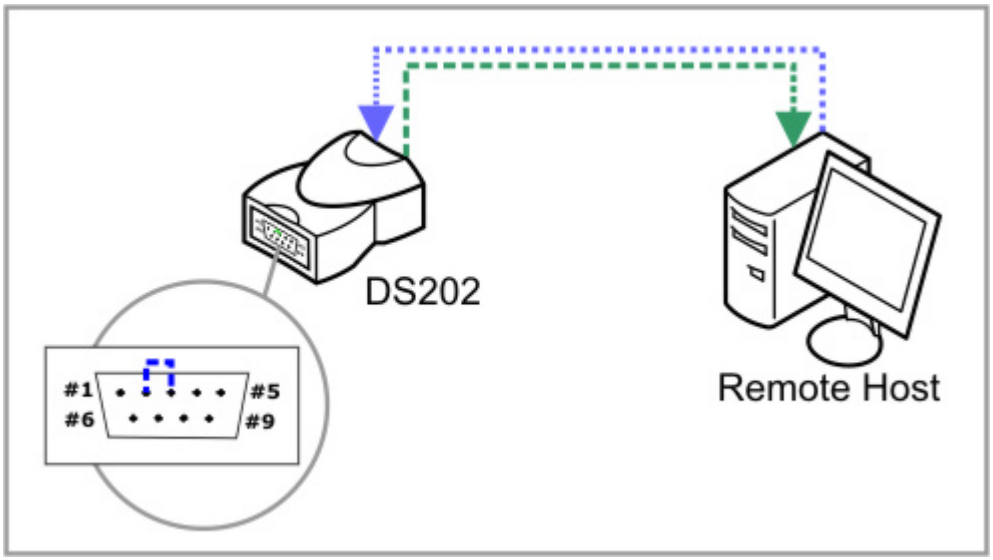
Here, we will connect one DS to a computer, using two separate cables; Thus, both ports of the DS (LAN and serial) will be connected to their respective ports on the computer. Follow these steps:

- Take a Device Server. If it's a model capable of several serial protocols (such as the DS100-B, who is also RS422/RS485 capable), make sure it's set to RS232.
- Power it on.
- Take a serial cable and follow the steps described in Establishing a Serial Connection with a Device Server. Return to this section once done.
- Now, connect the same Device Server to the same computer using a cross LAN cable, or to the network segment the computer currently belongs to (using a straight cable to a hub, etc).
- Follow the steps described in Establishing a TCP/IP Connection with a Device Server or in Establishing a Connection Through a Virtual Serial Port. Return to this section once done.
- That's it! Now, whatever you type in the window connected to the serial end, will show up in the window connected to the network end. And vice versa. If you can see this, you've performed all of the above correctly!

Creating a Loopback for Testing

A *loopback* connection is one where the signal being sent also comes back to the sender -- similar to an echo. This allows you to test a line (including the devices on it) to verify its correct operation.

Creating a loopback with a Device Server is rather simple. You just have to make sure you're working on RS232 (if your Device Server is also capable of other serial protocols) and connect serial pin #3 (*TX*, output) to serial pin #2 (*RX*, input), like so:



For this setup to work:

- Connect the DS to the network, and using *DS Manager*, set *RTS/CTS Flow Control* to *Disabled or remote*.
- Follow the steps laid out on Establishing a TCP/IP Connection with a Device Server and create a connection with your DS from *HyperTerminal*.

Now, anything you will send will get back to you. Just type, and see what you type on the screen. To confirm that this is indeed a loopback, just disconnect pin #2 from pin #3 on the DS, and you'll no longer see what you type.

Sending Commands Using HyperTerminal or Telnet

There are many methods of sending commands and getting information from the Device Server. In the following sections we will try those which can be performed with HyperTerminal (i.e, all except for UDP programming and in-band TCP programming). They are all documented here. The following text simply shows you how to really use them in a testing environment using *HyperTerminal*, so you could play around with them.

We assume that by now, you've established a network or serial connection with your Device Server using *HyperTerminal*. This is our starting point for each of the examples.

Each of the following sections discusses entering programming mode in a different manner. At the end of each section, once you've put the DS in programming mode, you will be directed to the programming exercise itself.

In the exercise, you will login, find out the IP of the Device Server, change its flow control mode, and log out. The purpose of this Application Note is just to show how to send the commands -- once you understand this, you could send any of the available commands which are all documented here.

Using telnet: For the Telnet Programming and Command-Phase TCP methods, any telnet client can be used. This does not necessarily have to be HyperTerminal.

Sending a Command (Command Format)

Now is the time to really get down to business. How do you actually send a command? Well, there are two major things you have to know:

Basic Command Format

The basic format for a command is:

STX	Command/reply	CR

- **<STX>** means "Start of Text". This character sometimes looks like a small smiley. To create it, press ` qoi H_ on your keyboard.
- **<CR>** means "Carriage Return" -- or, simply put, this is what the båíÉê key on your keyboard usually does. The create it, just hit båíÉê after typing the command.

Logging In

Sometimes you have to be *logged in* to the Device Server in order to send the

command. This is done by first making sure the Device Server is listening (i.e, is in a programming mode, like the ones described below) and then sending the **Login (L) command** command as the first one.

To know when exactly you have to be logged on (for what commands), see the *L* column of this table.

Logging Out

Programming methods which require you to log *in* will also require you to log *out*. This is done using the **Logout (O) command** as the last command in your programming session.

Serial Programming

There are two major method to get to serial programming mode. The main difference is that in one of them you need to press the RESET button (that's the whole method, basically) and in the other one, you need to send a string of characters.

The first method is used for troubleshooting and testing (similar to what we are doing here), and the second method is used by 'smart' RS232 devices, to control the Device Server. Both methods are documented in detail here.

This is the procedure for using the "regular" method of pressing the button:

- Perform the instructions under Establishing a Serial Connection with a Device Server. Select 38400, 8 bit, no parity.
- Press the SETUP button.
- The LEDs should start alternately blinking (green and red).
- Go to the Programming Exercise and start programming the device.

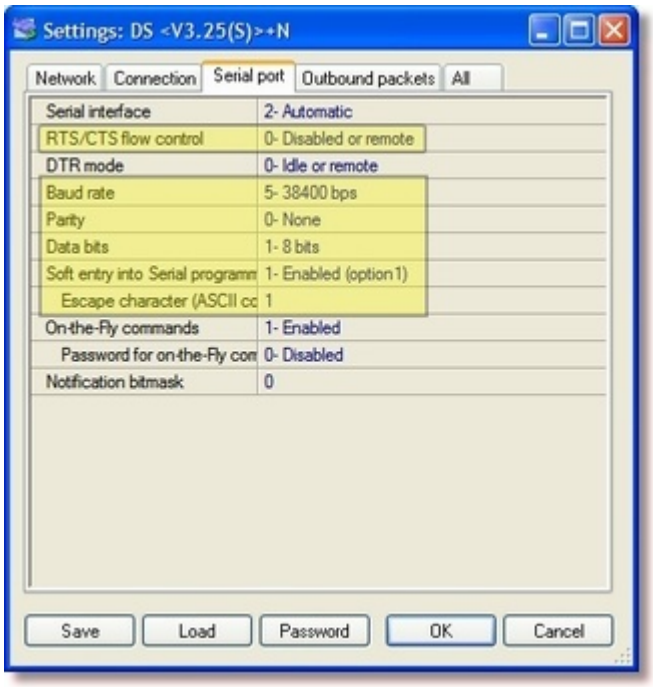
To know all there is to know about serial programming, please read Serial Programming in the full manual.

Serial Parameters (Modem Commands)

Modem commands are used to control the DS from the serial side. If your serial device is 'smart', it could change the destination address or port, or establish a connection, etc, from the serial side.

To Test Modem Commands Using HyperTerminal

- Run DS Manager and open the settings dialog for the DS to which you wish to connect, and switch to the 'Serial Port' tab:

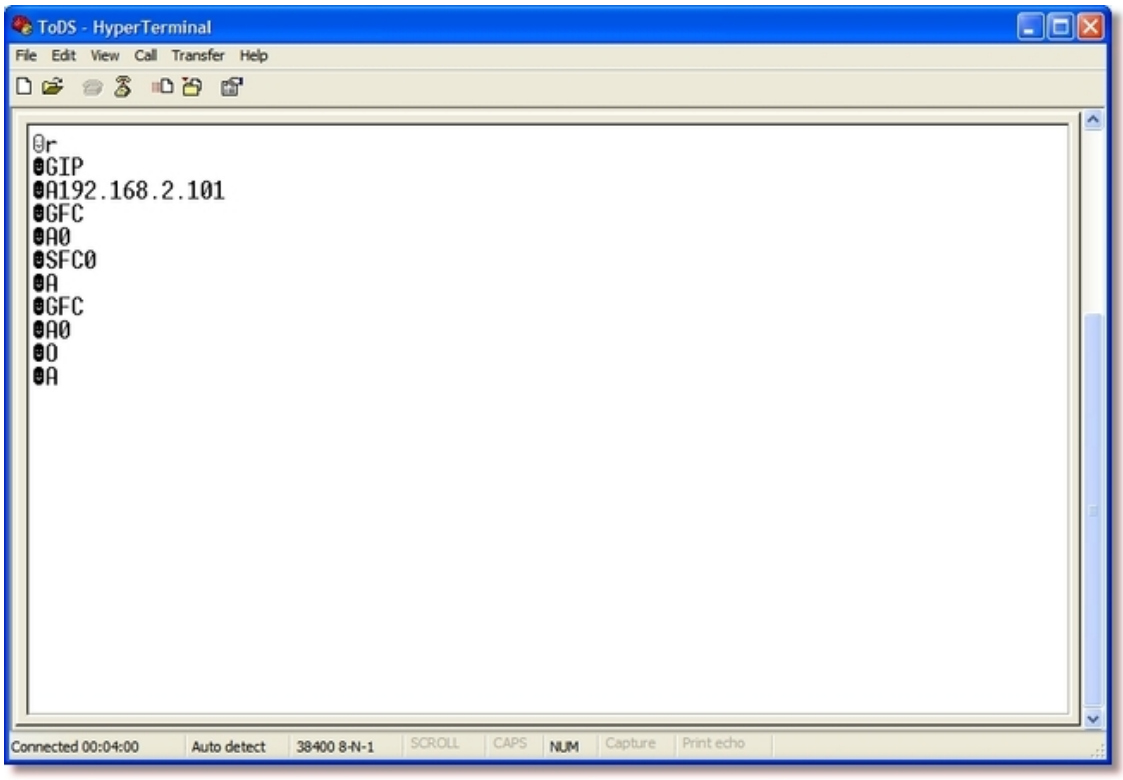


- Take note of the following settings:
  - *RTS/CTS flow control:* Set it to *Disabled or Remote*. (A complete explanation can be found here.)
  - *Baud rate, Parity, Data bits:* Can be anything, but note down what they are.
  - *Soft entry into Serial programming:* That's the whole point of this section. Should not be *0 (Disabled)*. Can be set to Option 1 or Option 2 -- we later have a seperate step for each of these options.
  - *Escape character (ASCII code):* Should be set to '1'.
- Click OK to close the DS settings dialog.
- Perform the instructions under Establishing a Serial Connection with a Device Server. Select the serial settings you've previously set for the DS.
- Perform the instructions under Setting Optional Parameters so you could see what you type (important in this case).
- If you selected ***Option 1*** for *Soft entry into serial programming:* Hit Ctrl-A 3 times in a row, with spaces larger than 100ms (i.e, not very very fast).
- If you selected ***Option 2*** for *Soft entry into serial programming:* Hit Ctrl-A just once and then type a random character (e.g, hit the *r* key).
- The LEDs should start alternately blinking (green and red). The DS is now in programming mode!

You can now do one of two things:

Program the DS Using Regular Commands

For this, switch to the Programming Exercise and perform the steps described. Here is the complete programming session of the exercise, performed using *Soft Entry mode 2*:



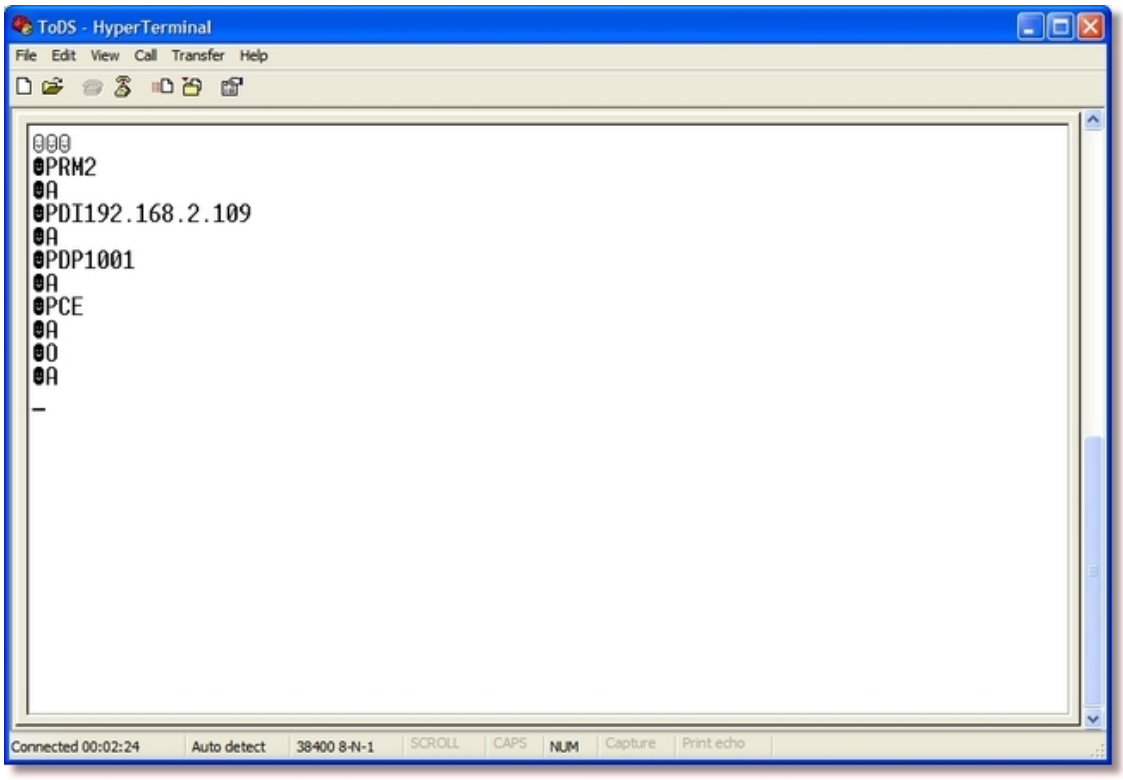
Program the DS Using Modem Commands

Modem commands are different than the commands in the example session in several respects:

- They *override* the default values for the settings.
- They can be sent only from the serial side.
- They allow you to perform important changes without rebooting the device -- thus, they are fast.
- They are temporary. On the next reboot, the previous settings are restored.

Read more about Modem Commands under Serial Programming and Modem (Serial-Side) Parameters & Instructions.

Below is a complete programming session using modem commands, performed using *Soft Entry mode 1*:



In short, what this session does:

- Changes the **Routing Mode (RM) parameter** to 2 (Client).
- Changes the **Destination IP-address (DI) parameter** to 192.168.2.109
- Changes the **Destination Port Number (DP) parameter** to 1001.
- Establishes a connection using the **Establish Connection (CE) instruction**.
- Logs out of the session (*not* rebooting the DS) using the **Logout (O) command**.

To try the programming session yourself, remember that the white smileys represent ASCII 001 chars and are produced using Ctrl+A, and the black smileys represent ASCII 002 (STX) chars and are produced using Ctrl+B.

Telnet Programming

This method of programming is enabled only for firmware version 3.51 and up, which runs on Tibbo second-generation devices, such as EM120, EM200, and DS203.

To enter telnet programming, establish a connection using HyperTerminal to the IP address of the device, but to port 23. Once the connection is established, you are in Telnet Programming mode. Perform the following steps:

- Send the first command, to login. As with all commands, you will **not** see what you send, but only what you receive:
  - Hit ` íêäH_.
  - Type i (capital i -- type äwhile pressing pefcq)
  - Hit båíÉê.
  - You should get an STX (Smiley) and ^. This means you **are now logged on.** It

looks like this: .

- Go to the Programming Exercise and start programming the device.

To know all there is to know about Telnet programming, please read Telnet TCP Programming [V3.50 and Above] in the full manual.

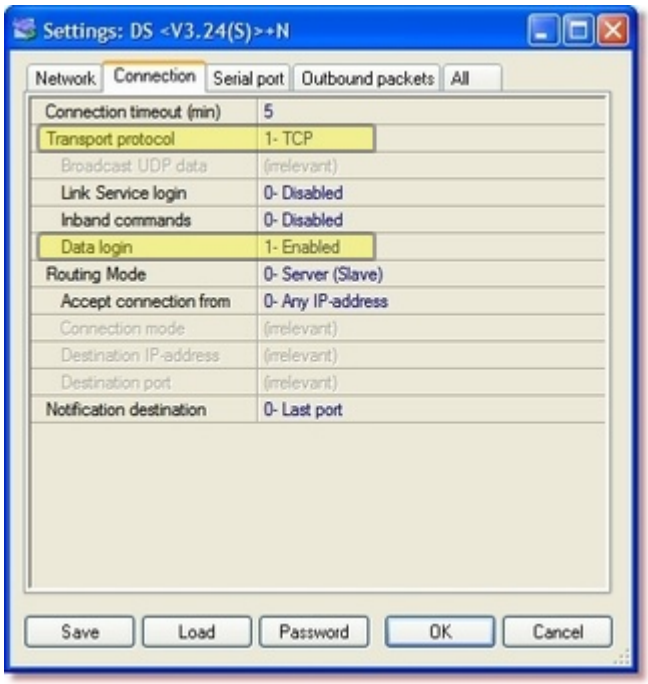
Command-Phase TCP

This is considered to be an *advanced* method, and is slightly more complex than the others. Usually there is no need for command-phase TCP programming. This example merely illustrates the capabilities of this mode -- do not feel compelled to try it, if you have no use for this feature in your actual environment.

In command-phase programming, every network communication session *starts* with a programming session. So, to actually get to your device and start communicating with it, you would have to send the **Logout (O) command** first. And to be able to send this command (and log out), you would first have to be logged *in.* This is accomplished using the **Login (L) command**.

Now, to actually test this method of programming, do the following:

- Run *DS Manager*.
- Open the settings for the DS you want to configure.
- Make sure its IP address is valid for your network, and note it down for later.
- Switch to the *Connection* tab.
- Make sure *Transport protocol* is *TCP* and that *Routing Mode* is *Server (Slave).*
- Set *Data login* to *Enabled*. Both of these settings can be seen below:



- Now, follow the instructions under Establishing a TCP/IP Connection with a Device Server and return here once you're done.

- Send the first command, to login. As with all commands, you will **not** see what you send, but only what you receive:
  - Hit ` íëäH_.
  - Type i (capital i -- type äwhile pressing pefcq)
  - Hit båíÉê.
  - You should get an STX (Smiley) and ^. This means you **are now logged on.** It looks like this: ☻A.
- Go to the Programming Exercise and start programming the device.

To know all there is to know about Command-Phase programming, please read Command-Phase (TCP) Programming in the full manual.

Programming Exercise

If at any time during this procedure you get a D reply (looks like ☻D) you need to login again. Just send the **Login (L) command** again, as described here.

We are assuming you are already connected to the device server (and also logged on, if your connection method requires logging on). If this isn't so, please read the previous sections and create a working connection first.

Getting the IP Address

Send the second command, to get the current IP address of the Device Server:

- Hit ` íëäH_.
- Type d fm (In caps. This is the **Get (G) command** with the **IP-address (IP) setting** as an argument).
- Hit båíÉê.
- You should get an STX character, ^, and the current IP address, like this:
  ☻A192.168.2.102

Getting the Current Flow Control Status

Now we will get the flow control status of the Device Server.

- Hit ` íëäH_.
- Type d c` (**Get (G) command** with the **Flow Control (FC) setting** as an argument).
- Hit båíÉê.
- You should get an STX char, an ^ (ack) and N (i.e, flow control is *Enabled*) or M (flow control is *Disabled*). Looks like this:
  ☻A1

In this case, this means flow control is *Enabled*.

Changing the Flow Control Status

Now we will change it to disable flow control.

- Hit ` íëäH_.

- Type pc` M (**Set (S) command** with the **Flow Control (FC) setting** as an argument, M = disabled).
- Hit båíÉê.
- You should get an STX character and ^.

Now let's check again the **Flow Control (FC) setting** to see if it is indeed disabled:

- Hit ` íêäH_.
- Type dc` .
- Hit båíÉê.
- You should get an STX char, an ^ (ack) and M (i.e, flow control is disabled).

Logging Out

The last thing we will do is log out of the programming session:

- Hit ` íêäH_.
- Type l (**Logout (O) command**).
- Hit båíÉê.
- You will not see anything sent back to you, because the session has just ended.

That's it. You've now successfully performed a demo programming session. Well done!

AN009. WAN Basics

This is a general overview of basic concepts in Wide Area Networks and Local Area Networks. If things like subnets, gateways and IP addresses confuse you -- you certainly **should** be reading this.

What Is This Good For?

Basically, if you want to reach your DS across the Internet and you're not so sure how this should be done, this application note should answer your questions.

Amongst the subject we'll take up:

- What Is a WAN.
- What Is a LAN.
- What are Subnets, and how many hosts can one subnet contain.
- Internal and External Addresses.
- Dynamic and Static Addresses.
- What Is a Gateway.
- Network Address Translation.
- How NAT Applies To Device Servers.
- Port Forwarding and how to connect to a DS which is inside of a LAN, from the outside.

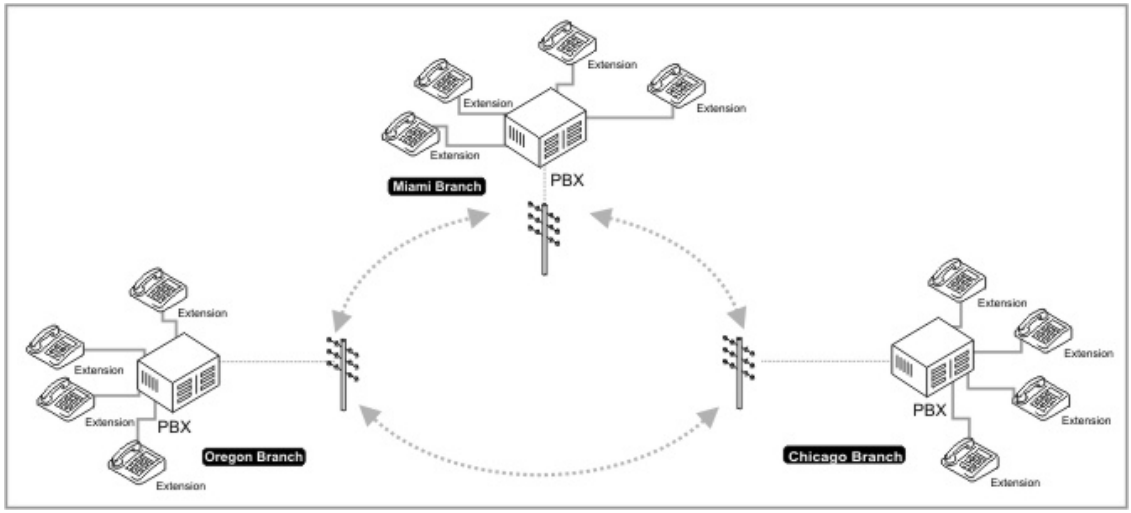
What Is a WAN

A WAN is, quite simply, a network which covers a large geographical area. It connects several physically remote locations one to the other.

The simplest way to think of a WAN is by thinking of a wide area network we already know: The public telephone system. When you think about it, the telephone system we all know and use in our day-to-day life is simply one huge network, connecting remote geographical locations. I.e, it is a Wide Area Network!

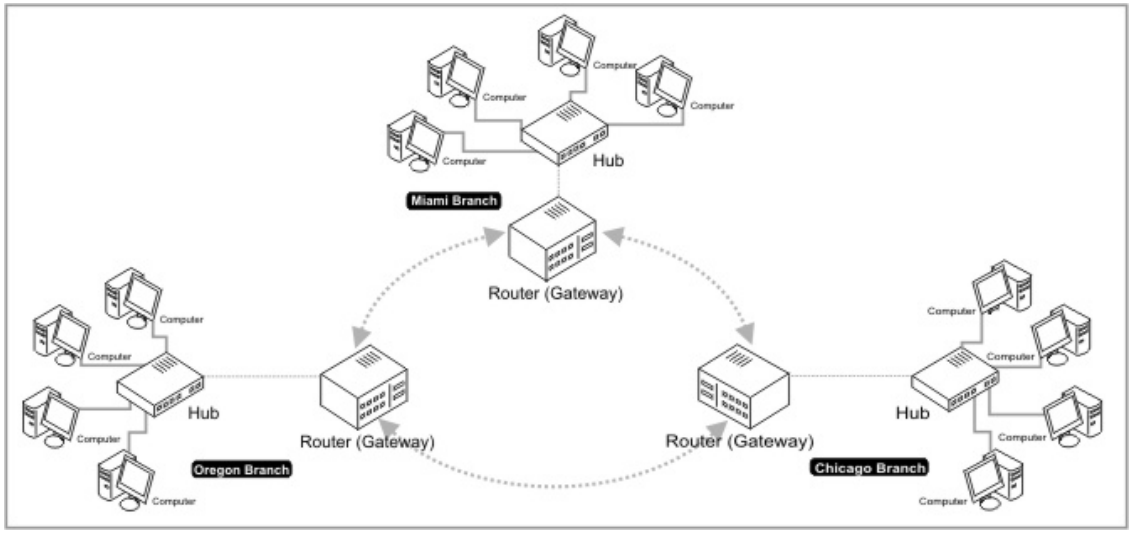
Since this analogy is so simple yet relatively accurate, we will use the phone system as an analogy throughout this text. Let us first begin with two diagrams, showing the telephone network next to a computer WAN (notice the similarities).

The Phone System:



The term PBX stands for **P**rivate **B**ranch e**X**change. It is the internal phone system of a business. In our diagrams, we refer to the "brain" of the system (the main box from which all lines come out) as the PBX, but in fact, the whole system (including the extensions) can also be called a PBX.

A Wide Area Network:



There is actually another common type of WAN that we all know to some degree -- the *Internet*. However, the inner workings of the Internet may be unfamiliar to readers who are not already proficient in networking, and so, we will not use the Internet to explain WAN concepts here.

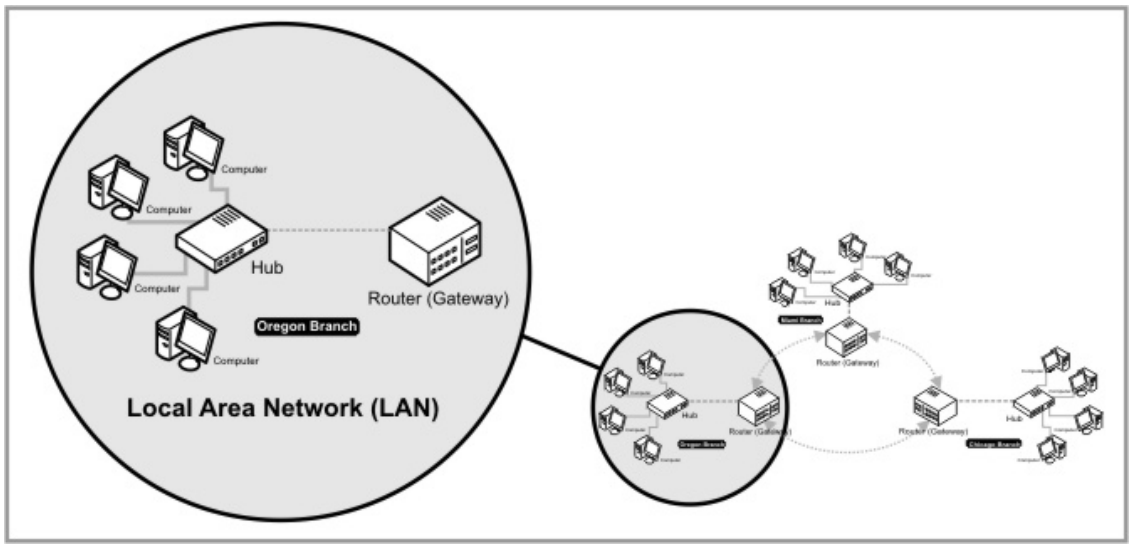
However, rest assured that once you are done reading this text, you will have a much better grasp of the inner workings of the Internet, as well.

What Is a LAN

A LAN is a Local Area Network. This is a network which exists in one specific location, and is relatively small. You can think of a LAN like you think of the internal phone system of an office. Every employee has an extension, but it is only an internal extension -- it is not a "real" phone line which is connected only to his own phone. To call from one extension within the office to another extension, you only have to dial two or three digits -- it is an "internal" call. It does not go through the public telephone system, and it will actually work even if the office has *no* external phone lines at all.

This is exactly the same also for a LAN. To get from one computer on a LAN to another computer on the same LAN, you do not have to go through any other network. It is a *private* network.

Below, you can see what is a LAN, in relation to a WAN (as previously explained):



There are certain types of communication which are unique to a LAN. One such type is called 'broadcast' communication. This is where one node (station, host, computer) on the LAN 'shouts out' so that all other nodes hear the message. This is similar to pressing the 'announcement' button on an office telephone, and using all the phones connected to the system to broadcast a message to all employees (through the speaker of the phone).

Naturally, such a feature would not be practical with the national telephone system. The system would simply collapse. The same is true for broadcast requests in a computer network -- they work only on a LAN, and will not go out to the whole WAN.

Subnets

In the phone system, we have one large network, spanning a whole country. We could also say that this network is subdivided into smaller, sub-networks -- one for each city, approximately.

So, when we have several phone numbers, how can we tell if they are from the same geographical area, or sub-network? We get this information from the *area code*. When we have two numbers, such as (323) 337-5578 and (323) 823-8461, we can immediately tell they're both from the same town, or from the same part of the country.

The same goes for IP addresses, but with a slight twist.

Every IP address is composed of two parts -- The *network identifier* and the *host identifier*. The network identifier is the left side of the address. For instance, in the address *192.168.0.1*, the first three octets (parts) are the network identifier. So, the addresses for all hosts (computers) on the same network will start with *192.168.0.x*.

This can be referred to as a *subnet.* A subnet is a portion of a network which shares a common *network identifier.* So, all computers which are (A) on the same network (physically interconnected) and (B) whose addresses start with the same *host identifier*, belong to the same *subnet*. Just like (323) 337-5578 and (323) 823-8461 belong to the same area code, so do 192.168.0.1 and 192.168.0.72 belong to the same *subnet*.

How Many Hosts Can a Subnet Contain?

With a local telephone number, we can quite easily tell the theoretical limit. If we have 7 unique digits per phone number (excluding the area code, which is shared by all numbers), we can theoretically have up to 9,999,999 phone numbers in the same area code.

Of course this isn't accurate -- if we have emergency numbers such as *911*, we lose significant portions of this range. You cannot have any number which starts with the digits *911* -- so you lose 9,999 potential numbers, which now cannot be assigned.

With IP addresses, the limitation is slightly different. Each octet in an IP address can range from 0 to 255. This is because an *octet* is composed of 8 bits. In binary notation, 8 bits can be any combination between 00000000 and 11111111. When you convert these values to decimal notation, 00000000 remains 0 (naturally), and 11111111 equals *255*. Hence, our range -- 0 to 255 per octet.

So, if we selected a network identifier composed of 3 octets -- such as 192.168.0.*x* -- we have just one octet left for the *host identifier*. The host identifier must be unique for each host on the network. This means we can have up to 256 hosts per each such subnet.

What if we want to have more than 256 hosts in our subnet? In this case, we must use a network identifier which is composed of less octets. If we use only the first two octets as the network identifier, and use the last two octets as the host identifier, we can have up to 65,536 hosts on the same *subnet*.

Subnet Masks

Subnet masks are used to denote which part of the IP address is designated as the *network identifier* and which part is designated as the *host identifier*.

A typical subnet mask is 255.255.255.0. This means the first three octets contain the *network identifier* and the last octet contains the *host identifier*.

For the purposes of this application note, what you need to know is that in a subnet mask, when an octet contains the number *255*, that octet is supposed to contain a portion of the *network identifier*. When an octet contains a *0* in a subnet mask, then that octet is supposed to contain a portion of the *host identifier.*

It may happen that an octet in a subnet mask will contain a number which is somewhere between 0 and 255 (such as 224). This is relatively rare, and further limits the number of available addresses in the subnet. However, this is beyond the scope of this application note.

The following table lists the most common subnet masks, along with their names and number of hosts available for each subnet mask.

Subnet Mask	Number of Hosts	Network Designation
255.0.0.0	16,777,216	Class A Network
255.255.0.0	65,536	Class B Network
255.255.255.0	256	Class C Network

Internal and External Addresses

So, we have internal networks (LAN) and a large "external" network which combines them together (the WAN). And like in the phone system, every node (host) on the network has a "number", or an address.

However, as covered here, we do not have an infinity of numbers (addresses) to give out. Each subnet has a finite size.

If we have an office with 150 phone extensions, do we also need to lease 150 phone lines so that each extension would have a number? Of course not -- the extensions get internal numbers, from the internal phone system. We can have just 10 "real" phone lines for this office, and all of the internal phones would use them. You would just hit "9" on a phone and get an outside line.

This also means that the internal extension numbers are under local control (the local administrator assigns them), and they don't cost anything. This is because they can be repeated over and over in different offices -- every office can have an internal extension numbered "101". But only one office can have a "real" phone number like (343) 553-7592.

This is exactly the same for a LAN. Inside a LAN subnet, the computers use "internal" IP addresses. These are specific address ranges which can repeat over and over across many different networks. For instance, all addresses which begin with 192.168.x.x are *internal* addresses -- they are reserved for use inside of LANs. If you search all across the Internet (the huge WAN we all share) for such an address (like, a website running on 192.168.0.100), you will never find it, as this address is not allowed for "public" use.

Reserved Address Ranges

The following table shows the address ranges specifically reserved for use on a LAN. These are also sometimes called 'non-routable addresses', as they cannot be used to reach other computers in other subnets (i.e, they cannot get across a router, which is a device for connecting two subnets).

Start Address	End Address	Number of Individual IP Addresses
192.168.0.0	192.168.255.255	65,536
172.166.0.0	172.31.255.255	1,048,576
10.0.0.0	10.255.255.255	16,777,216

External Addresses

The term *external address* refers to an IP address which is not restricted to the local LAN. This is an address which can be reached from all parts of the WAN. It is sometimes also called a "real" IP address -- as it is not an internal address, but is leased and officially assigned to one specific host on the WAN (this assignment may not be permanent, but more on that in the next section).

You can think of the *external address* like an actual phone number -- this is an address which is used to identify a specific host in the context of the whole WAN, and not just internally in a LAN. Just like phone numbers, these addresses are managed and assigned by licensed companies -- you don't just "take" whichever address you want. They are guaranteed to be unique for every host on the whole network -- no two hosts will have the same "real" IP address at the same time, across the whole width and breadth of the Internet (or any other WAN for that matter).

Dynamic and Static Addresses

As noted above, there is only a finite number of available IP addresses, even on the largest of WANs, the Internet. Simply put, there aren't enough addresses for everyone.

Dynamic Addresses on The Internet

Let's say an Internet Service Provider has 100,000 users subscribed. Does that mean the ISP must also lease 100,000 "real" IP addresses -- one per user account?

Luckily, no. When you think about it -- it's highly unlikely that all 100,000 users will be connected at any given moment. Most likely, only a certain percentage of the users will be concurrently connected -- perhaps only 40% of all users registered with this ISP. So, this ISP theoretically only needs 40,000 "real" IP addresses to satisfy all the needs of its clients at any given moment -- which is 60,000 less IP addresses to lease.

So, the obvious solution for this ISP is to lease 40,000 "real" IP addresses, and to *dynamically assign* those IP addresses to the hosts who wish to connect. A host logs on, authenticates with the ISP (provides username and password), and is then assigned an IP address for use while online.

When that host closes the connection, the address is 'reserved' for him for a certain amount of time -- so if he re-connects within 5 minutes it is likely that he will get the same address. However, once that set amount of time elapses, the IP address is no longer reserved, and returns to the *address pool*. Some time later, a different host connects and gets that same address.

That is how dynamic addresses are usually used in the context of the internet.

Dynamic Addresses on A LAN

The internet isn't the only place where dynamic IP addresses can be of use. Some corporations run very large internal networks (whether LANs, in one large installation, or WANs, connecting several remote places). When you have thousands of hosts connected to the same network, manually assigning an IP address for each and every one of them can be quite a hassle. It is also error prone -- what if you happen to assign the same IP address to two hosts? Or what if you made a typo?

Thus, dynamic IP addresses are often used also for LANs. Once you perform the initial setup, the system 'runs itself', and every host gets a correct address automatically.

The protocol by which a host asks to be assigned an IP address, and gets that dynamic IP address, is called *DHCP*, which stands for *D*ynamic *H*ost *C*onfiguration *P*rotocol.

Disadvantages of Dynamic Addresses

So far we've come to see that dynamic IP addresses seem to save quite a lot of money and manual labor. So -- what could possibly wrong with them?

Let's say you wish to talk to a friend. You both know each other's number, so you call each other. One day, you get a new mobile phone, and your old number is no longer valid. Until you let your friend know your new number -- only you can call him. He won't be able to call you, as he doesn't have your number.

Now, what happens if before you've had a chance to call your friend and let him know you have a new number, *he* got a new number as well? Now, neither one of you can call the other!

That is the main drawback of using dynamic addresses. If the computer you're trying to reach has an IP address which sometimes changes, you can never be sure it will be there next time you'll want to access it.

For reliable communication, each connection must have at least one end which is fixed. The dynamic end of the connection will be able to reach the fixed end. If both ends are fixed, they will of course be able to freely originate communications to each other (just like when calling your friend from above).

Static Addresses

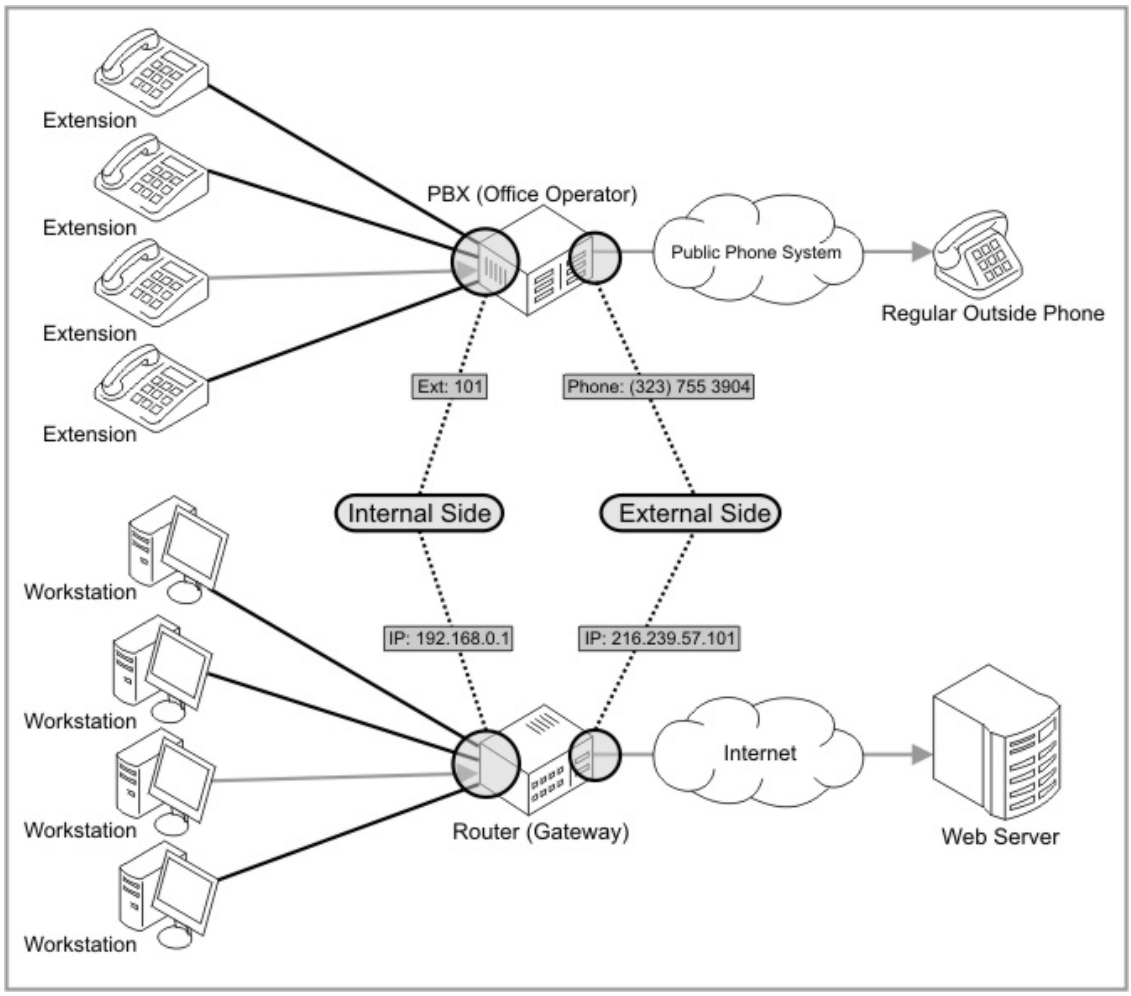
As covered above, static addresses are mostly needed for internet servers. When you need to be able to *reach* a certain host on the network, that host needs a *static IP address!* You need to know where it is, to reach it. Static IP addresses can be leased from any ISP.

What Is a Gateway

What a Gateway Is and What It Does

"Internal" IP addresses in a LAN are exactly like internal phone extensions in an office phone system. You can use them to call out, but nobody can reach you directly from the outside. If someone calls the office and wants to talk to you, one of two things may happen: (1) the receptionist manually transfers the call to your extension, or (2) you have a special outside number which is mapped to your internal extension, so whoever dials the external number is automatically forwarded to your extension.

In a LAN, when you want to "call out" (communicate from your computer to an external computer, like an Internet host running a website) you go through a *gateway*. A gateway is a unique device on the network, because it is connected to *two* networks at the same time: Both internally, to your LAN, and externally, to the WAN. Just like the office phone system (the PBX itself) is connected both to the internal extensions in the office, and to the "real" outside phone lines of the office.



So, as can be seen above, a gateway has two addresses -- one on each network. And it can be very easily used for making an outbound connection. *But* -- what happens when you want to make an *inbound* connection?

Network Address Translation

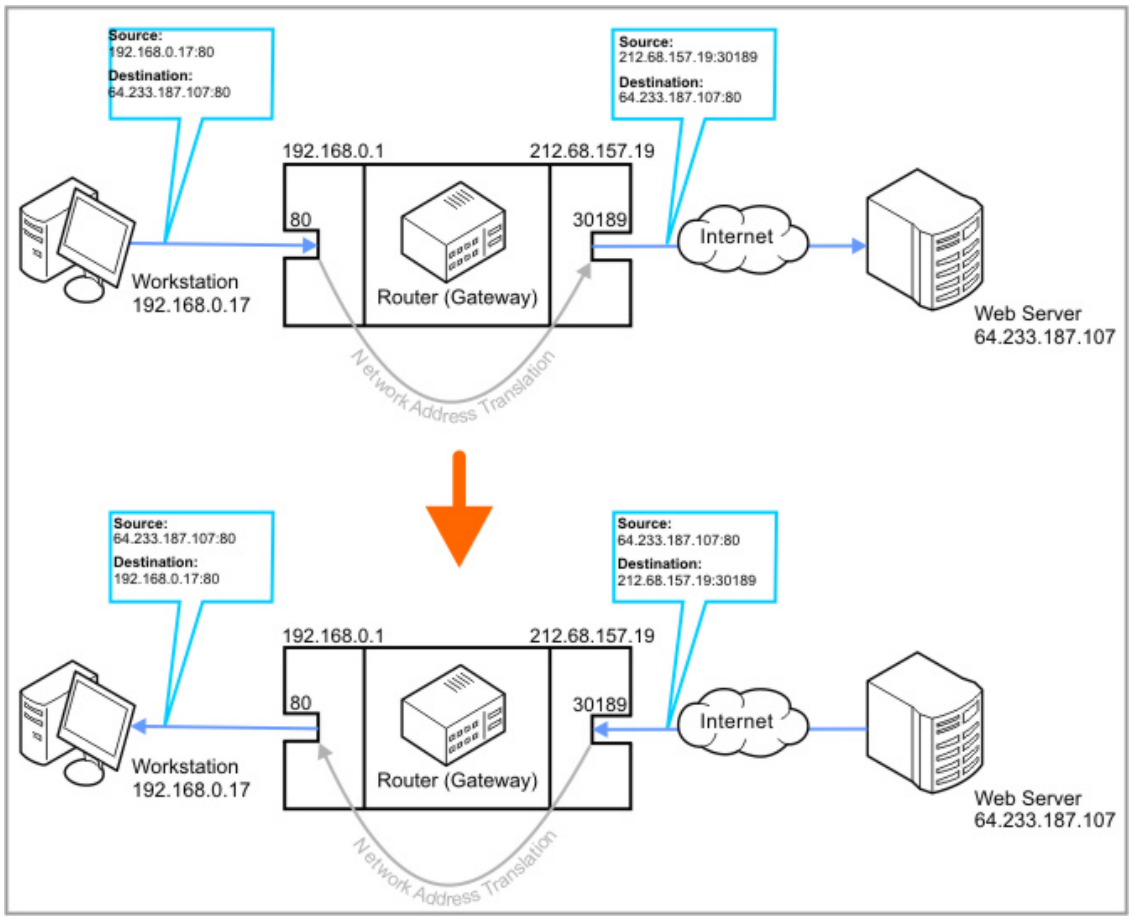
As covered above, a gateway is a device with two addresses -- one on each network it's connected to.

So, you have just one address coming out of your LAN into the WAN. What happens if you have 5, 10 or even 100 computer on your LAN, all trying to use the gateway at the same time for connecting to the internet?

This is where Network Address Translation (NAT) comes into play. With NAT, all computers in the LAN "hide" behind the gateway:

- A host on the LAN makes an outbound connection to somewhere on the WAN.
- The packets first arrive at the gateway (on their way out).

- The gateway then modifies the packets, so as to make them appear as if:
  - It (the gateway) has originated them itself.
  - The packets come from one specific port in the gateway (and not necessarily the port from which they originally came). This port is actually mapped to the LAN host which originally sent the packet. The gateway now knows that packets arriving to port 30189 (for example) should be forwarded internally to host 192.168.0.17 and to port 80 (for example).
- The gateway sends the packets on their way.
- The packets arrive at their remote destination host.
- The remote host replies, and directs the reply to the IP address and port of the router which previously sent the packets.
- The router gets the reply (to port 30189 in our example), modifies the packet and forwards it internally to the host which originally made the outbound c/connection.



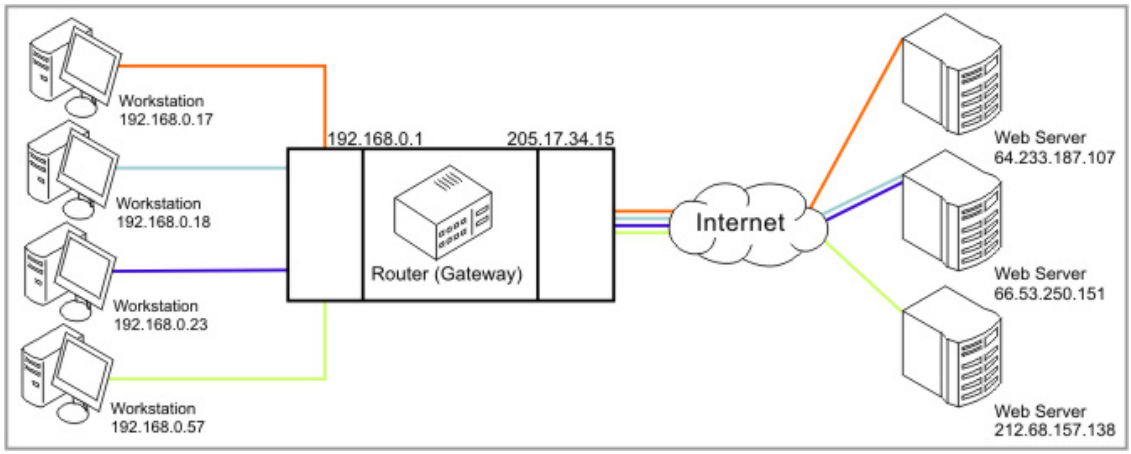
A colon (:) mark at the end of an IP address refers to the *port* for that address. So, the designation *64.233.187.107:80* refers to port *80* of the IP address 64.233.187.107.

Many Hosts Can Originate Outbound Connections

The biggest advantage of using NAT is in limiting the amount of "real" IP addresses you need. You can have hundreds of computers communicate with various hosts on the internet, using just one "real" IP address. This translates into significant

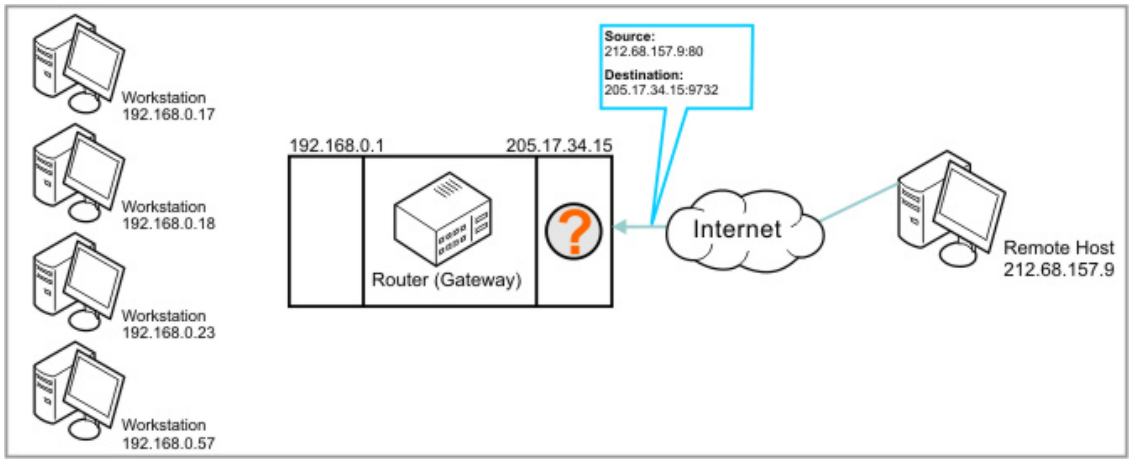
savings in cost.

Below you can see four different workstations on the same LAN communicating at the same time with three different web servers on the WAN through just one "real" IP address (that of the router):



No Host Can Receive an Inbound Connection

The biggest disadvantage of using NAT is that it's impossible to originate inbound connections. Supposing you have a host internally, within the network, and you wish to originate a connection to this host from outside (from some host on the WAN); The router will not know *where* to direct the incoming connection. No internal host tried to originate an outbound connection to this remote host, and so no internal host is currently mapped to that port on the router and expects a connection from that external host. So, when the packet comes to the router, it goes nowhere (is 'dropped'):



For more information regarding NAT, including other implementations of NAT, please see the Wikipedia article titled "Network address translation".

How NAT Applies To Device Servers

This topic adds no new information; It merely provides an explicit demonstration of what happens when you put a Device Server inside of a LAN, behind a NAT gateway.

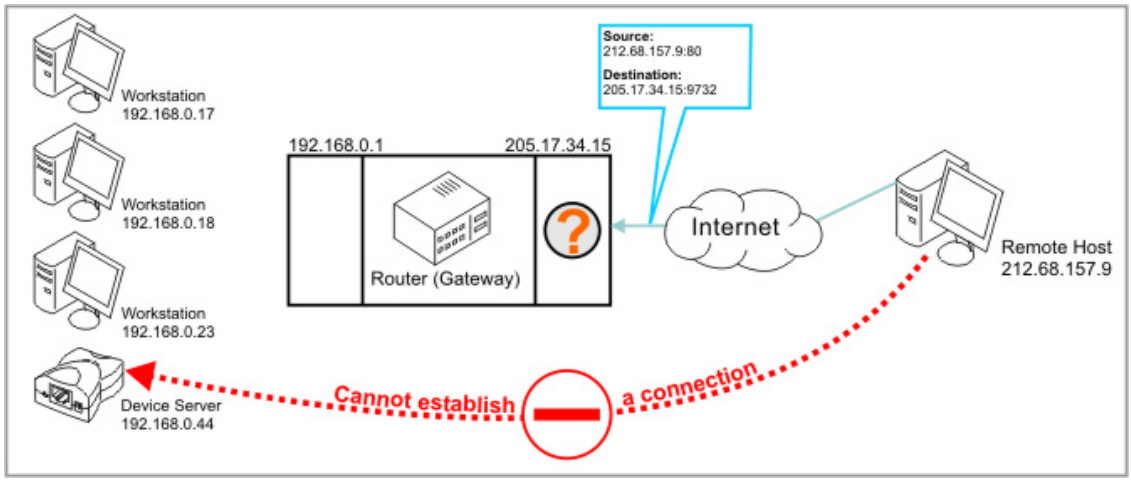
IMPORTANT NOTE: *establishing* the connection isn't the same as *using* the connection!

When one side establishes the connection, *both* sides can then use it. It does not mean that only the side who *established* the connection can now talk, and that the other side must listen.

Think of it like a phone call -- when you call your friend, *both of you can talk*, even though it's you who made the call. This is *important.*

Establishing an Inbound Connection

An image is worth a thousand words:

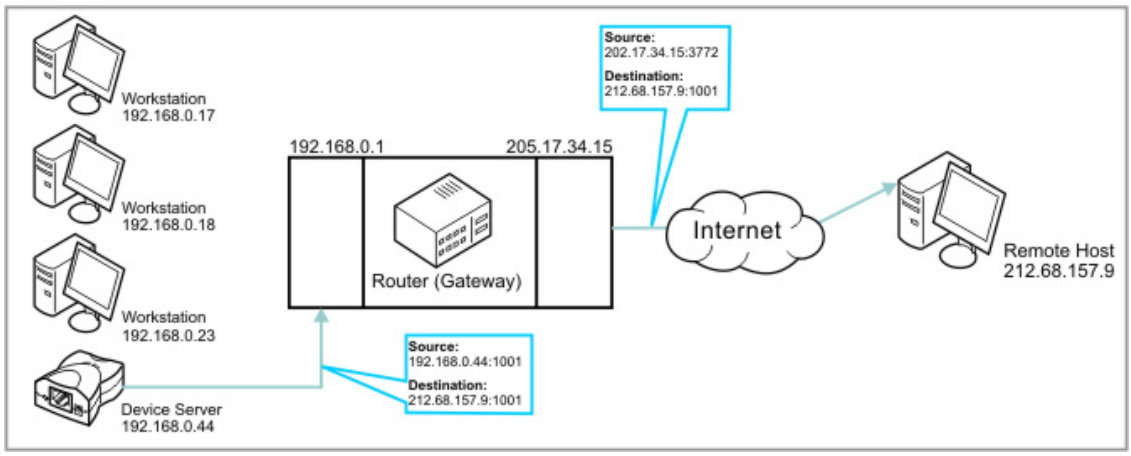


As can be seen above, when you have a DS in a LAN, behind a NAT router, you cannot simply establish a connection from outside. The port isn't mapped anywhere, so the router drops the packet.

There are three solutions for establishing a connection with a DS which is behind a NAT router:

Solution 1: The DS Establishes The Connection

Here, the DS initiates the connection. As covered above, there's no problem in setting up an outbound connection from behind a NAT router. In effect, it looks like this:



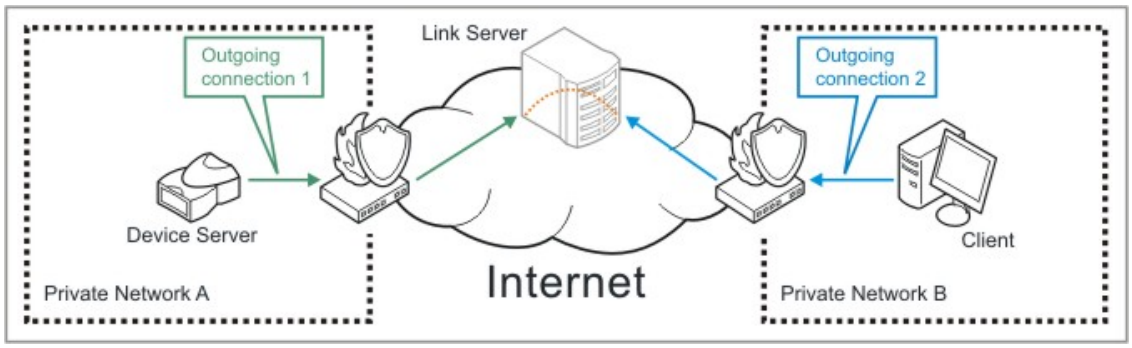
Points of attention:

- The routing mode for the DS, as configured in **Routing Mode (RM) setting**, must be *Client Only* or *Client Or Server*.
- The destination of the DS, as configured in **Destination IP-address (DI) setting**, must be actually reachable from within the NAT. Meaning, it cannot be behind *another* NAT router. You need to be able to ping it. See the diagram above -- the Remote Host can be reached directly and has a "real" IP address.
- It is advisable to set the **Connection Mode (CM) setting** as *Immediate (on Power-up)*, so that the DS would establish the outgoing connection immediately when it's turned on.

Solution 2: Use Tibbo LinkServer

The Tibbo LinkServer is a product developed to answer this exact need. What if both the remote host *and* the DS (or multiple Device Servers) are behind a NAT router, and you cannot allow inbound access for either one of them?

In this case, you use a *middle man*. You need a server in the middle, to which both the remote host and the DS could reach, and 'meet' there. Such a scenario would look like this:



This solution is discussed in detail in the LinkServer user manual. It does require one static IP address, and the purchasing and configuration of a separate product (The LinkServer).

Solution 3: Configure The Router for Inbound Access

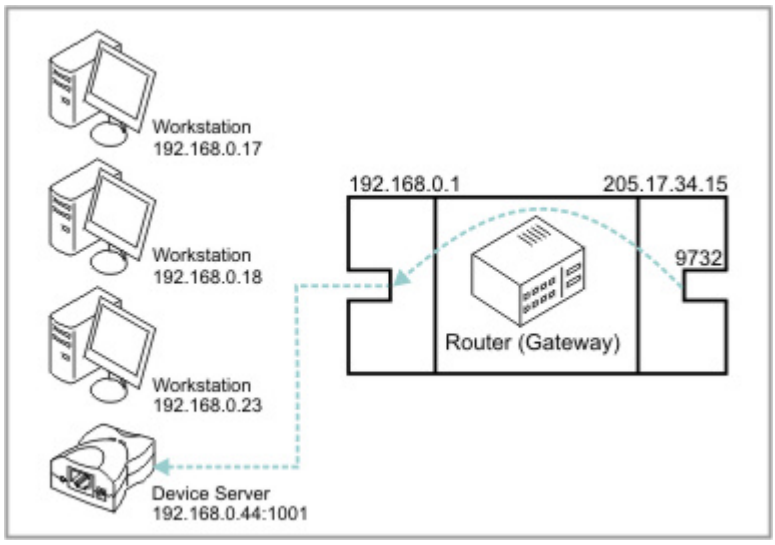
It is possible to configure a NAT router so it would allow certain inbound traffic,

and would correctly route it to a host within its LAN. This is done using Port Forwarding.

Port Forwarding

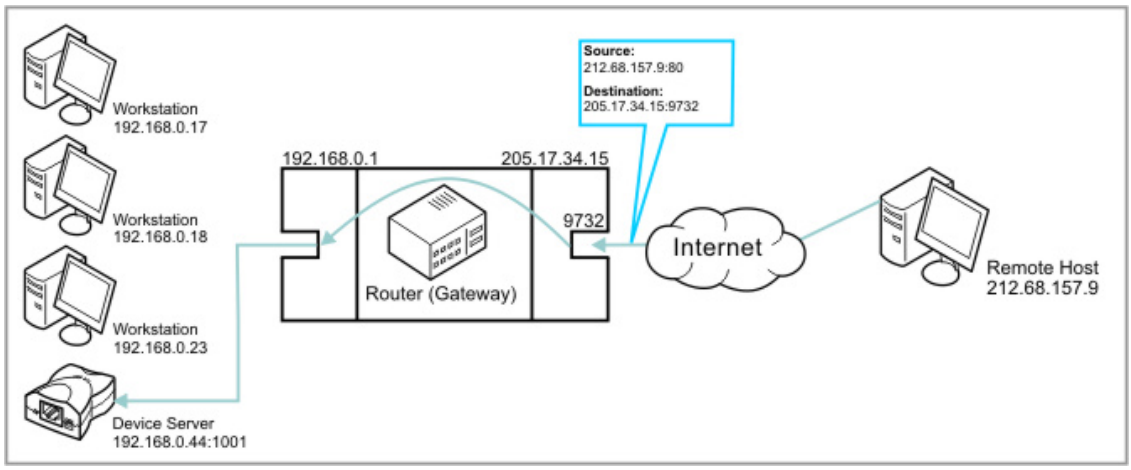
Port Forwarding is a feature present on many modern routers. In essence, it allows you to map a certain outside port of the router to a specific address and port in the LAN.

This means, for example, that every packet sent to port *9732* on the router from the WAN side is forwarded by the router to *192.168.0.44:1001* on the network:



As you can see, this is a *static* configuration. **This is configured on the router itself -- it is not related to any Tibbo equipment.** The diagram above shows the setup itself -- no connection is in progress, but the router knows what to do when it gets data to port 9732.

After properly configuring port forwarding on the router, every time a connection is established to a specific port on the "real" IP address of the router, the data is forwarded to a specific host and port on the LAN. Hence:



This shows the same set-up as above, but with a connection in progress. As you can see, the remote host can actually initiate a connection to the Device Server inside the network, because the router knows what to do with the data. Contrast the above diagram with the diagram on How NAT Applies To Device Servers.

Notes on Setting Up Port Forwarding

In order for port forwarding to work, several conditions must be true:

- Internally, in the LAN, the DS must use a static IP address. The router knows it should forward the packets to 192.168.0.44 -- so if the DS suddenly becomes 192.168.0.*53*, it doesn't get the packets. So its address must be static.
- The external ("real") IP address of the router must also be static. Otherwise, the Remote Host (212.68.157.9 in the diagram above) will not know where to connect.
- The router must be properly configured for port forwarding, using its internal configuration interface. The way to do this varies from router to router, and is documented in the manual for your specific router.
- The *internal* port and the *external* port aren't necessarily the same -- just because you set the router to forward incoming connections *from* port 1001, doesn't mean it would forward them inside the network *to* port 1001. You have to set this correctly.
- Remember -- *establishing* the connection isn't the same as *using* the connection. This is covered in detail under Important Note above.
- The DS must accept incoming connections. That means the **Routing Mode (RM) setting** must be set to "Server" or "Server/Client".

In some cases (depending on the application), it may be possible to obviate the requirement for a static IP on the external side of the router by using *Dynamic DNS*. This, however, is beyond the scope of this application note. To learn more, read the Wikipedia article titled Dynamic DNS.

AN010. Controlling the DS from the Serial Side

This Application Note is mainly aimed at manufacturers incorporating Tibbo modules into their products. It can also be used by anyone who can modify the behaviour of his serial device, and wishes to integrate it more tightly with a Device Server.

Device Servers can be made to act as 'modems'; Connections can be established and closed at will, by the serial device. You can also select the destination for the connection, and set other relevant parameters -- all from the serial side, and without writing to the EEPROM of the device.

This application note assumes prior knowledge in HyperTerminal. If you're not sure what HyperTerminal is, or how to use it to access a Device Server, please read AN008. Using HyperTerminal first and perform the exercises described in it, before moving on to study this present AN.

What You Need for Modem Commands to Work

To use modem commands, you need a serial device which can *send* them to the DS. This sounds elementary, but it is actually the first thing you should note. In

effect, this means that your serial device would have to be 'tailored' for working with the DS. It would have to speak in a language the DS understands -- i.e, send the serial commands the DS could understand.

Since these commands are used specifically to communicate with Device Servers from Tibbo, standard serial devices don't come with the option to send them already built in. So, you have to be able to modify the firmware for your serial device to have it send these commands. Usually manufacturers can do this, but some serial devices are very flexible, and allow even an end user to perform such modifications (especially Linux-based devices, etc).

Topics In This Application Note

Specifically, we will take up the following topics:

- Benefits of Modem Commands
- Issuing Commands
- Establishing a Connection
- Terminating a Connection
- Finding Out Connection Status (X)
- Finding Out Connection Details (U)
- Exiting Serial Programming Mode
- DSR/DTR
- Real-World Example

Benefits of Modem Commands

There are several good reasons for using modem commands, if you can do so:

Establishing a Connection

Using modem commands, you could allow the user to initiate a connection directly from the device, at his will. You could also dynamically change the IP address to which the connection is established. This is good when working with a primary and secondary server (if the primary server does not reply, you fall back to the secondary server). It's also good for WAN scenarios -- the serial device 'dials in' and doesn't have to be connected at all times.

Terminating a Connection

Modem commands can also be used to terminate the data connection from the serial side. This can lead to all sorts of interesting options: For instance, a serial device can send data to five (or more) different destinations, in rotation, again and again. It will establish the connection (using the **Establish Connection (CE) instruction**), send the data, disconnect (using the **Close Connection (CC) instruction**), and will establish a new connection with a new host, again.

Not Writing to EEPROM

The EEPROM memory the DS uses to store settings has a finite (although large) number of write cycles. Thus, every time you write the EEPROM, you somewhat shorten the operational life of the DS. While this is not felt under normal use (like

writing to the EEPROM once or twice per day), if you perform several EEPROM write actions every *hour*, this may shorten the lifespan of the DS more noticeably.

When you issue a modem command, it is not written to the EEPROM of the DS. It takes effect immediately after the programming session, and stays in effect until the DS is turned off, or until it receives another command contradicting the first one. Thus, using modem commands translates into a longer lifespan for the DS.

Also, since modem commands take effect after a programming session without the DS needing to be reset (powered off and back on), this means they can be applied very quickly. For instance, it takes less than 2 seconds to enter a programming session, change the routing mode of the DS, give it a new destination address and port, order it to connect, and end the programming session. Fast.

In Summary

Modem commands allow for much tighter integration of the serial device with the DS. Above are just several examples. The rest of the text contains technical details -- once you read them, you might come up with several ideas of your own.

Issuing Commands

Entering the Serial Programming Mode

Before issuing commands, the DS must listen (i.e, be in programming mode). There are two ways to enter Serial Programming mode, and they are both described under Serial Parameters (Modem Commands) in AN008. Using HyperTerminal.

Once you enter Serial Programming mode, you can start issuing commands. Basic command format is described under Sending a Command (Command Format).

Commands are Asynchronous

One of the most important things to remember about commands is that they are *asynchronous*. In simple terms, this means that when you issue a command and get back an **A** (Ack) response, *the command has not necessarily been executed.* The **A** means that the DS has received your command and has started executing it.

This is most prominent when issuing instructions to establish or close connections. When sending such instructions, the **A** reply comes immediately -- the DS takes very little time to receive and 'understand' the instruction. But having understood it, it now has to execute it -- and that sometimes takes time.

The time it takes to establish a TCP connection varies according to the topology of the network the DS is in. Thus, after issuing an instruction such as the **Establish Connection (CE) instruction**, you must check the DS status to see when the connection has actually been established. For this, use the **Echo (X) command**. This is described in detail under Finding Out Connection Status (X).

A complete listing of modem commands which can be issued appears under Modem (Serial-Side) Parameters & Instructions. The text below only highlights several commands with specific implementation details.

Establishing a Connection

A connection is established using the **Establish Connection (CE) instruction**. This instructions can be sent with no parameters, or with parameters specifying the destination IP and port.

Option One: With Parameters

With parameters, a CE instructions looks like this:

```
Ypqu[m` bNVOKNSUKMRLNMMNY` o[
```

The parameters are the IP address and the port to which the DS has to connect.

Option Two: With No Parameters

The CE instruction can be issued simply as `Ypqu[m` bY` o[`. In this case, the DS infers the parameters according to the most recent **Destination IP-address (DI) parameter** and **Destination Port Number (DP) parameter** it has received. **However**, if since the last PDI/PDP instructions, another network host has opened a connection *to* the DS, the DS then uses the **Destination IP-address (DI) setting** and **Destination Port Number (DP) setting** and attempts to connect to the destination host defined by these settings.

How to Prevent Another Host From Opening a Connection to The DS

To prevent another host from connection to the DS while you are sending it parameters and instructions to configure an outbound connection, use the **Routing Mode (RM) parameter** to set it to 2 (Client Only). Thus, a typical connection establishment sequence could go like this (bold lines are commands sent to DS).

Set the DS to work as Client Only, thus denying incoming connections:

`Ypqu[moj OY` o[`

DS acknowledges command:

`Ypqu[^Y` o[`

Set Destination IP address:

`Ypqu[mafNVOKNSUKMKOMY` o[`

DS acknowledges command:

`Ypqu[^Y` o[`

Set Destination Port:

`Ypqu[mamNMMNY` o[`

DS acknowledges command:

`Ypqu[^Y` o[`

Establish the connection:

`Ypqu[m` bY` o[`

DS acknowledges command:

`Ypqu[^Y` o[`

Log out of Serial Programming Mode:

`Ypqu[l Y` o[`

DS acknowledges command:

Ypqu[^Y` o[

You are now supposedly connected to 192.168.0.120:1001. To verify this, see Finding Out Connection Status (X).

Terminating a Connection

There are two ways to terminate a connection using modem commands: You can *close* the connection, or *abort* it. The two are not the same, when it comes to TCP connections.

Closing a Connection

The **Close Connection (CC) instruction** uses the proper termination sequence for TCP connections (FIN-ACK-FIN-ACK), and thus closes the connection in an orderly fashion. It is recommended to use this instruction.

Aborting a Connection

The **Abort Connection (CA) instruction** uses an RST packet to terminate a TCP connection. This is one-sided termination, and is usually less recommended.

Terminating UDP "Connections"

UDP is a connectionless protocol. The data "connection" the DS uses is merely a stream of packets, as described under UDP Data "Connections". Thus, you can either abort or close such a "connection" -- the action performed is identical in both cases.

Remember! Commands are asynchronous -- just because you told the DS to terminate the connection, and it answered back "^" doesn't mean the connection has actually been terminated. See Finding Out Connection Status (X).

Finding Out Connection Status (X)

The **Echo (X) command** causes the DS to return a string full of status information. The contents of this string is fully documented in the manual page for the command, so we will only take up the part which is pertinent to this AN: *Data Connection Status*.

Following is a quote from the manual page for the command, restructured for the purposes of this AN.

When issued using the serial programming mode, the reply for this command looks like this:

^ã ëÉáÅvézlbp

For now, we only care about the bold **c** which appears near the middle of the command (it is the 6th character). You can read about the meaning of the other letters in the command manual page. The ***c*** stands for "data **c**onnection status". This is just a position in the reply string of this command -- it does not actually say **c** in the command (otherwise, how could it be used to detect a status? Naturally, it must change.) Thus, this position in the reply string for the command can contain one of the following characters:

G Data connection is currently closed.

^ Sending ARP, or (V3.54+) establishing PPP link (using PPPoE).

l A data connection is being established

` A TCP connection is currently established, or is being closed (but not yet closed).

r A UDP connection is currently established.

o The connection was reset by the remote host -- no connection at the moment.

c LinkServer login failed.

i LinkServer login is currently in progress.

A Practical Example

Let's say you have a serial device and wish to make an outbound connection to a server. To do so, you need to first check if it is safe to try and connection (i.e, that there isn't an active data connection at the moment). Then, you have to issue the connection command. Then, you have to check that the connection has indeed been established, and continue checking periodically until the connection is established.

Such a programming session could look like this (bold lines are commands sent to DS):

Checking -- is there an established data connection?

Ypqu[uY` o[

The 6th char *(c)* is an asterisk - Data connection currently closed. Proceeding:

Ypqu[^kpG GGGY` o[

Establish a connection with 192.168.0.20:1001:

Ypqu[m` bNVCKNSUHWRLNMNY` o[

DS acknowledges command:

Ypqu[^Y` o[

Checking -- is there an established data connection?

Ypqu[uY` o[

The 6th char *(c)* is l - Data connection being established. We are still not connected:

Ypqu[^kpGl GGGY` o[

(After waiting for some time):

Checking -- is there an established data connection?

Ypqu[uY` o[

The 6th char *(c)* is ` - We have a TCP connection, verified:

Ypqu[^kpG` GGGY` o[

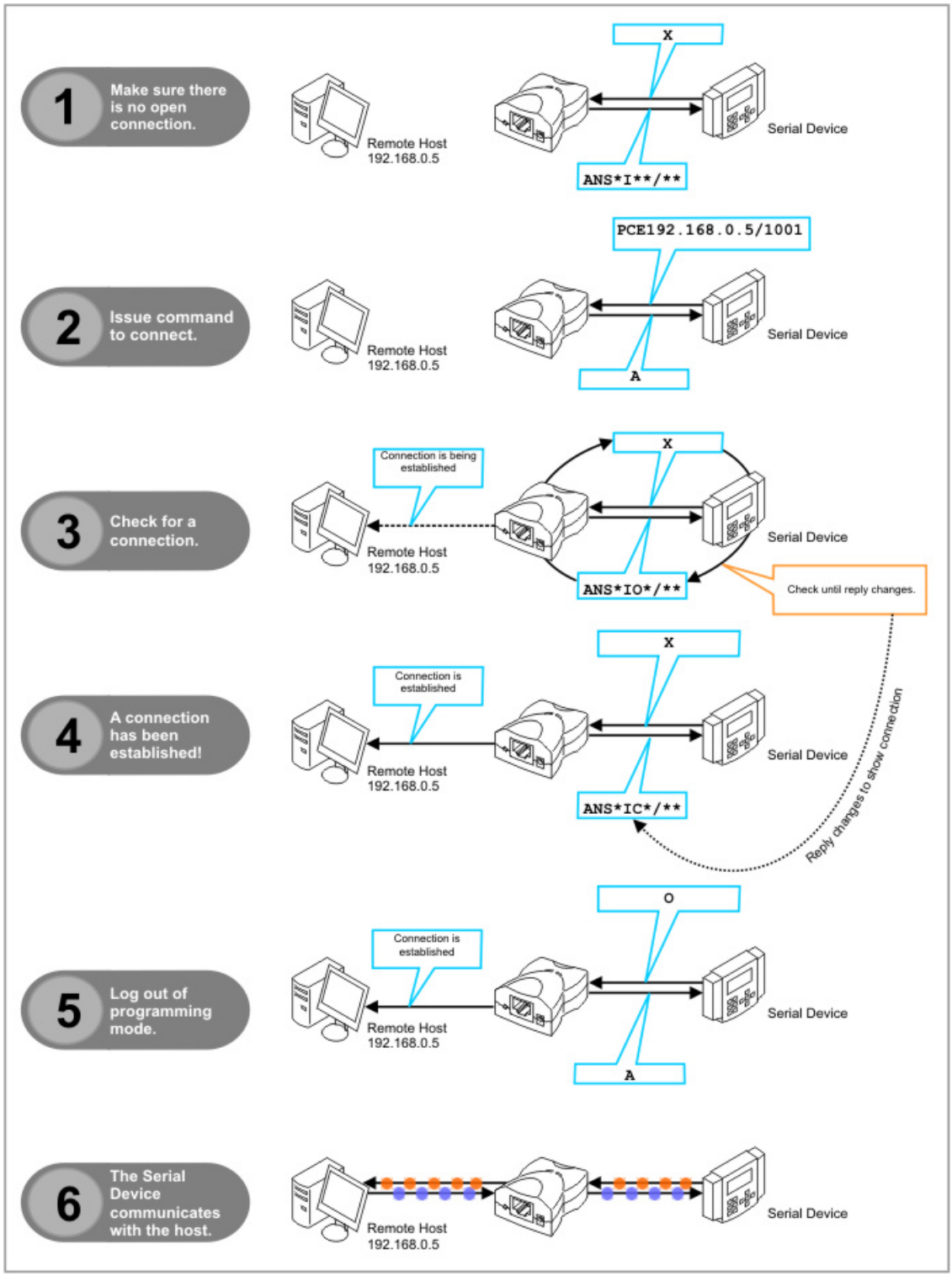
Logging out of programming mode to start communicating:

Ypqu[l Y` o[

DS acknowledges command:

Ypqu[^Y` o[

The Example, Illustrated:



Finding Out Connection Details (U)

The **Status (U) command** provides further information with regards to the DS. Specifically, when issued through the serial port, it returns two (or, optionally, three) parameters:

^ ÇÇÇÇÇÇÇÇÇÇÇÇéééééÀÁÂÃÄÅ

ÇÇÇÇÇÇÇÇÇÇÇÇStands for the IP address the DS is currently communicating with (or has last communicated with, or will next communicate with)

ééééé=Stands for the with which the DS is (has, will) communicating with.

ÀÁÂÃÄÅÆÇIf r is sent with the optional parameter N=(r N), then the reply includes this argument, which stands for the IP address of the DS in the PPPoE link (if such a link is active).

In essence, this command allows you to see *who* the DS is communicating with, once you have used the **Echo (X) command** and have found out that a connection is established.

Exiting Serial Programming Mode

To exit the serial programming mode, you should send the **Logout (O) command** . It is important to send this command, and *not* the **Reboot (E) command**. When rebooting the device, all current parameters (modem commands) are lost. Thus, the parameters you've entered in the programming session just ended will be lost and will have no effect on the DS.

This is why it's important to remember -- **To exit Serial Programming Mode, use the Logout (O) command**.

DSR/DTR

Until now, we have touched topics which involve sending and receiving ASCII data to and from the device. However, sometimes a simpler method is needed. This is true especially in the case of establishing and dropping connections, which is one of the simplest functions of any communication device, and is often used.

Thus, the advanced usage options for the DSR and DTR lines. Correct use of these lines will allow your serial device to find out the current connection status, and control it, without even entering programming mode or sending a single command. See below:

DTR - Detecting Data Connection Status

The **DTR Mode (DT) setting** controls the behaviour of the DTR line.

Assuming this setting is set to 1 (connection status), the DTR line will be LOW when no data connection is established, and HIGH when a data connection is established. Thus, you can have your serial device directly sense the status of the data connection, without using the **Echo (X) command**.

DSR - Controlling Data Connection Status

If the **Connection Mode (CM) setting** is set to 3 (on command or DSR=HI), the DS will establish an outbound data connection whenever the serial device sets the DSR line to HIGH for at least 20ms. This is, of course, only relevant if the **Routing Mode [setting/parameter]** is not 0 (when it is 0, the DS will not make an

outbound connection).

Real-World Example

Fallback to a Secondary Server

Let us say we have a fire detection system in a large building. This system is connected to a remote server (say, at a security company) via a DS. It sends out a heart-beat signal every minute, to report its status.

If the remote server at the security company fails, communication with the fire detection system will be lost, and the security company will not know what's going on in the building (and probably, in many other buildings). This cannot happen.

Thus, modem commands can be used to detect that there is no connection to the remote server, and change the destination address of the DS to a secondary server. Thus, if the primary server fails, the DS falls back to a secondary server, and all works well. Such a communication session will go like this:

Checking -- is there an established data connection?

Ypqu[uY` o[

The 6th char *(c)* is an asterisk - Data connection currently closed. Proceeding:

Ypqu[^kpGGGGGY` o[

Establish a connection with the primary server:

Ypqu[m` bONOKSUKNRTHPRLNMMNY` o[

DS acknowledges command:

Ypqu[^Y` o[

Checking -- is there an established data connection?

Ypqu[uY` o[

The 6th char *(c)* is | - Data connection being established. We are still not connected:

Ypqu[^kpG| GGGY` o[

(After waiting for some time):

Checking -- is there an established data connection?

Ypqu[uY` o[

The 6th char *(c)* is an asterisk - Data connection is closed. Could not get to server:

Ypqu[^kpGGGGGY` o[

Falling back to the secondary server -- establish connection with .37:

Ypqu[m` bONOKSUKNRTHPTLNMMNY` o[

DS acknowledges command:

Ypqu[^Y` o[

Checking -- is there an established data connection?

Ypqu[uY` o[

The 6th char *(c)* is | - Data connection being established. We are still not connected:

Ypqu[^kpG| GGGY` o[

(After waiting for some time):

Checking -- is there an established data connection?

Ypqu[uY` o[

The 6th char *(c)* is ` - Successfully connected to secondary server:

Ypqu[^kpG GGY` o[

Logging out of programming mode to start communicating:

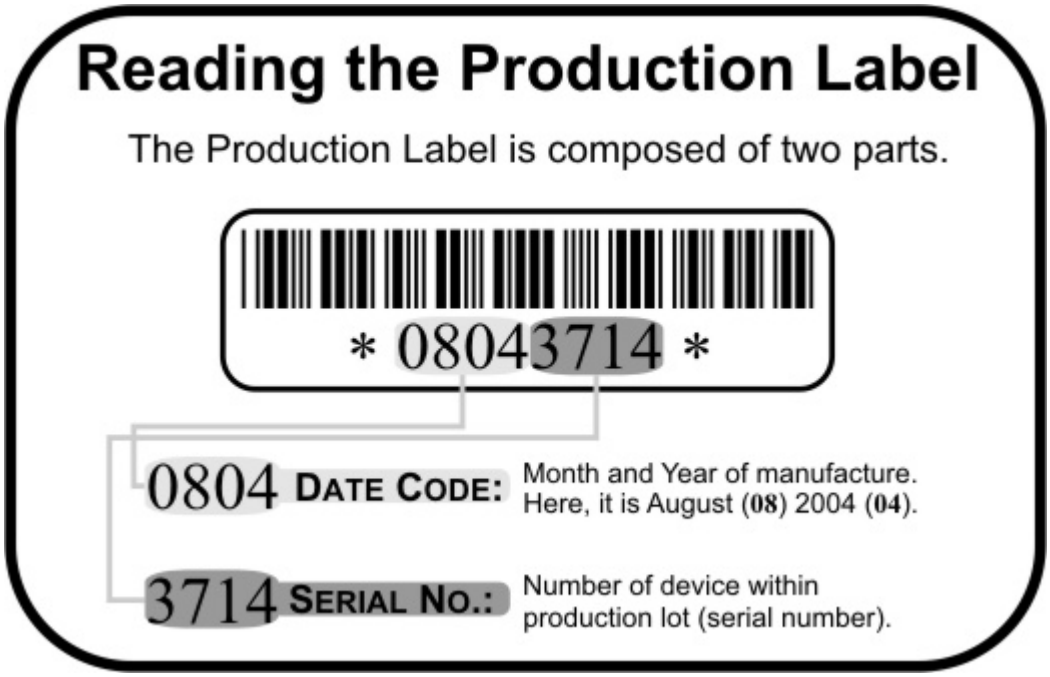
Ypqu[l Y` o[

DS acknowledges command:

Ypqu[^Y` o[

AN011, Reading the Production Label

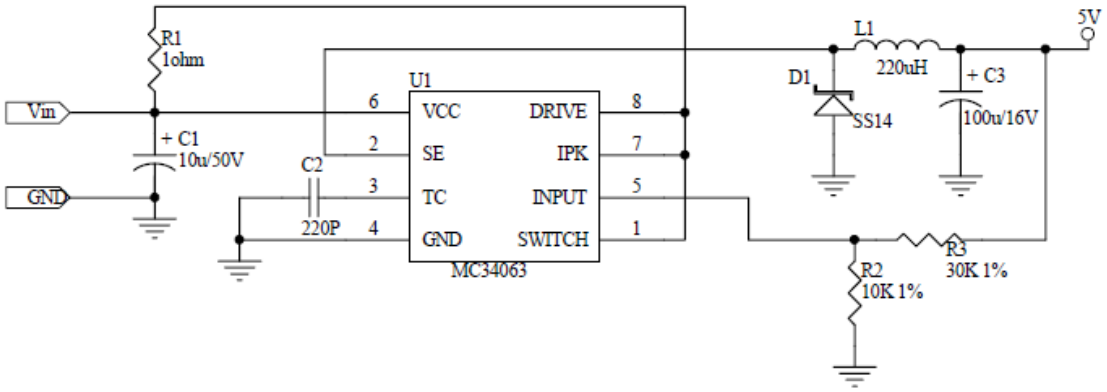
Every Tibbo product comes with a barcode production label. You can use this label to see if your product is still under warranty:



AN012, Creating an Integrated Power Supply

This AN is aimed at OEMs and customers integrating Tibbo modules within their existing products. It covers how to create a reliable power supply which can stably power an EM200 module (for example) with input voltage as low as 9V or as high as 35V.

Following is the schematic:



Notes:

- **U1** (MC35063) is a very popular power IC manufactured by ON Semiconductor.
- **R1** is very important. It is just 1 (one!) Ohm, but we really do not recommend the user to omit it.
- **R2** and **R3** are "1% tolerance" (high-precision) because they define the output voltage of the power supply.
- **C1** and **C3** capacitors: Do not use SMD capacitors -- use regular through-hole aluminum capacitors. This really helps reduce noise produced by the power supply.

This is an analog circuit, so layout matters. Apply reasonable "good layout" effort.

Note that it is not necessary to try and obtain the exact capacitors, diodes, etc that Tibbo currently employs. The part numbers are for reference only.

Ideally, one should use an oscilloscope to see what sort of "square wave" the PSU generates, both at low and high input voltages. **R1** can be adjusted to achieve a better (cleaner) square wave signal on a particular PCB layout. There are no recipes here -- just try and see what works for your circuit.

Update history

This topic details update history for this document system.

14OCT2008 release

- Documented EM203, RJ203, and DS203 devices.
- Updated Device Server Application Firmware topic.
- Removed comparison charts for modules and external devices servers, also Firmware Revision History topic. This information can be found on our website.

16SEP2008 release

- Added Legal Information topic.
- Documented EM203A and RJ203A devices.

11JUN2008 release

- Converted this manual into the "Serial-over-IP Solutions Manual". All BASIC-programmable products are now in a separate manual called "Programmable Hardware manual".

12MAY2008 release

- Small changes throughout the manual, mostly related to the WA1000 module.

18APR2008 release

- Posted a notice about current non-availability of the EM202 and EM202-EV products.

10MAR2008 release

- Documented new WA1000 Wi-Fi module.
- Updated EM1000 documentation -- practically every topic has been edited. Notable changes: I/O Pin Assignment and Pin Functions -- new I/O lines documented; SPI Port Connector -- new topic; Power, Reset, PLL Control, and Mode Selection Lines -- added schematic diagram of a power supply; Mechanical Dimensions and Specifications and Ordering Info -- added info on new EM1000 versions.
- Removed LinkServer documentation. The LinkServer is now called "AggreGate" and is documented in a separate manual.
- Removed information about the real-time counter from EM1202 documentation -- it was there by mistake.

14SEP2007 release

- Created the DS10xx section.
- Corrected I/O Pin Assignment and Pin Functions (EM1000) topic: pins 40-47 are NOT combined into P5, these are independent lines.
- Made slight corrections to the LEDs (EM1000) topic.
- New Inter-board Bus Connector topic.
- Updated (and renamed) 12VDC Power Adaptors topic. Also updated: Power Jack topic for EM202-EV, and Power Jack topic for DS202.

08AUG2007 release

- Created the EM1202 section.
- Created the RJ1202 section.
- Corrected minor misprints in the pin assignment table of the I/O Pin Assignment and Pin Functions topic (EM1000).
- Updated the list of the EM1000 features in the EM1000 BASIC-programmable Ethernet Module topic.
- Slightly corrected Specifications and EM1000 Modifications topic.
- Throughout the manual, changed all occurrences of "88MIPS" to "88MHz", which is more precise. 88MHz roughly translates into 50MIPS.
- Updated General-purpose I/O Lines (EM1000): info about 8-bit ports has been added.
- Updated and renamed the Comparison Chart for Modules topic.
- Renamed certain topics. For example, Modules instead of "Ethernet-to-serial Modules".
- Updated "Specifications" topic for all Modules: packaging has changed, many modules are now shipped in tubes.

29MAY2007 release

- Created new topic Retransmission Period (RP) setting and updated TCP Data Connections.
- Updated topic IP Address (IP) setting (default IP address is now 1.0.0.1).

02MAR2007 release

- Updated EM1000 picture in the EM1000 BASIC-programmable Ethernet Module topic.
- New EM1000-00 and -01 topic explains the difference between these two EM1000 versions.
- Updated EM1000 diagram in the I/O Pin Assignment and Pin Functions topic.
- Updated Ethernet Port Lines (of the EM1000) topic: the information on how to change the host PCB to accommodate new EM1000-...-01 parts was corrected.
- Updated EM1000-EV Evaluation Board topic.
- New EM1000-EV-00 and -01, Converting EM1000-EV-00 into -01 topics.

15FEB2007 release

- Updated "Specifications" topic for the following devices: EM100, EM120, EM200, EM202, DS100, and DS202.
- Updated "Ethernet port lines" topic (add schematic diagrams for connecting magnetics and RJ45 socket) for the following devices: EM120, EM200.
- Corrected an error on the diagram for the Ethernet port of the EM202 -- Built in RJ45 Ethernet Connector topic.
- Updated the following topics related to the EM1000 module: EM1000 BASIC-programmable Ethernet Module, I/O Pin Assignment and Pin Functions, Ethernet Port Lines, Real-time Counter, Mechanical Dimensions, Specifications and EM1000 Modifications.

07NOV2006 release

- Documented new Cable Status (C) command.
- Documented new Get My IP (T) command.
- Updated Firmware Revision History.

08AUG2006 release

- Small changes corrections related to the release of new 3.31 firmware.

07AUG2006 release

- Small corrections related to the release of new 3.30 and 3.62 firmware.

24JULY2006 release

- Corrected buffer size for non-programmable ("device server") firmware of EM120/EM200/EM202 modules (Comparison Chart for Ethernet-to-Serial Modules). Correct buffer size is 8KBytes in each direction.
- "Setting up AuthKey Using DS Manager" -- corrected a small typo.

06JULY2006 release

- A whole new section has been added- EM1000 BASIC-programmable Ethernet Module.

- These topics have been updated to reflect the fact that the product can work with "device server: firmware OR TiOS firmware (that executes BASIC application): EM120 Ethernet-to-Serial Module, EM200 Ethernet-to-Serial Module, EM202 Ethernet-to-Serial Module.
- Corrected DTR Startup Mode (DS) setting description. It erroneously stated that this setting was not available on **V3.5x** branch of firmware, which is not true.
- In the new firmware, line status changes are recognized much faster- it only takes 20ms instead of 0.5s. The following topics have been updated to reflect this change: Notification Bitmask (NB) setting, Notification (J) message, Handling of RTS, CTS, DTR, and DSR Signals.

26JUNE2006 release

- Corrected documentation error in Telnet TCP Programming topic. When Transport Protocol is TCP and Port Number is 23 telnet programming is impossible, as any TCP connection to port 23 will be interpreted as data connection. Previously, we have erroneously stated that in this case all connections to port 23 will be interpreted as programming, not data connections.

20June2006 release

- Updated Installation procedure for VSPDL

15JUNE2006 release

- Added Use WinSock for transport
- Added Managing Address Book Groups (Groups button)
- Added Specify Destination By
- Updated Unsupported Features and Limitations to remove note about not working through proxies
- Updated Access Modes to show new design (tabs instead of listbox)
- Updated DS Manager with new screenshot
- Updated Preferences Dialog (General Tab)
- Updated Transport Protocol & Listening Ports, Transport Protocol & Listening Ports, Transport Protocol & Listening Ports on all 3 Connection Wizard scenarios
- Updated DST Revision History

18MAY2006 release

- Added DTR Startup Mode (DS) setting
- Updated Serial Settings with **DTR Startup Mode (DS) setting**

16MAY2006 release

- Updated Jump To Netloader (N) command [Release3.0], Set Programming Request Flag (N) command [Release 3.5], Reset Upload Process (Q) command [Release 3.5], Upload Data Block (D) command [Release 3.5] to clarify that Telnet and TCP are not supported.
- Updated Telnet TCP Programming [V3.50 and Above] to reflect LF received from DS in reply.

24MAR2006 release

- Added AN012, Creating an Integrated Power Supply

16MAR2006 release

- Minor corrections

- Added warning note to AN001. Customization Options in Our Products

---
19FEB2006 release

- Updated Firmware Revision History and Device Server Application Firmware topics with the most recent firmware version number (V3.28/V3.56).

14FEB2006 release

- Corrected command format for Select In Broadcast Mode (W) command.

22DEC2005 release

- Updated Firmware Revision History and Device Server Application Firmware topics with the most recent firmware version number (V3.26/V3.56).

05DEC2005 release

- Added Cable Status (C) command

27SEP2005 release

- Added AN011, Reading the Production Label.

12SEP2005 release

- Added a new section to AN002. Practical Advice on Integrating EM Module into Your Device, Example: PIC with EM202

09AUG2005 release

- Added AN010. Controlling the DS from the Serial Side.

02AUG2005 release

- Added new section to AN008. Using HyperTerminal, Serial Parameters (Modem Commands).

01AUG2005 release

- Added information regarding RTS behaviour to Soft Entry (SE) setting.

29JUL2005 release

- Added AN009. WAN Basics.

21JUL2005 release

- Updated Firmware Revision History and Device Server Application Firmware topics with the most recent firmware version number (V3.25/V3.55).

19JUN2005 release

- Added AN008. Using HyperTerminal.

10JUN2005 release

- Added section to AN007, Preparing your Network (Router Configuration).

31MAY2005 release

- Added AN007. Installing and Configuring LinkServer.

15MAY2005 release

- Firmware release V3.14/3.51 is now used as a "baseline" release and any feature or change that was made in a later firmware version is marked with the sign that looks like this: **[V3.24/3.54+]** (this one means: this feature is available in firmware release V3.24/3.54 or higher). This brings to an end tracking of changes from much older firmware V2.5x which was done previously. All references to V2.5x has been removed.
- In connection with the above individual command, setting, and parameter (instruction) description topics (under reference section) now state clearly if a particular setting has existed in a baseline V3.14/3.51 or was added later
- Updated topics (this excludes some minor corrections):
  - Status LED signals
  - Ethernet port and network communications
  - UDP data connections
  - DHCP
  - Inband (TCP) programming
  - Commands, messages and replies
  - Echo (X) command
  - Status (U) command
  - Jump to Netloader (N) command [Release 3.0]
  - Settings
  - Network settings
  - Owner Name (ON) setting
  - Device Name (DN) setting
  - Gateway IP-address (GI) setting
  - Password (PW) setting
  - Inband Commands (IB) setting
  - Destination IP-address (DI) setting
  - Serial settings
  - Soft Entry (SE) setting
  - Serial Interface (SI) setting

  - Modem (serial-side) parameters and instructions
  - Destination IP-address (DI) parameter
  - Firmware revision history
  - VSP Manager
  - AN001. Customization options in our Products
- New topics:
  - PPPoE [V3.54+]
  - Link Server support
  - DS powerup procedure
  - Data connection establishment procedure
  - Set Programming Request Flag (N) command [Release 3.5]
  - Reset Upload Process (Q) command [Release 3.5]
  - Upload Data Block (D) command [Release 3.5]
  - dDNS Service Registration (DD) setting
  - dDNS Service IP-address (LI) setting
  - dDNS Service Port (LP) setting
  - LS Auto-registration (AR) setting
  - PPPoE Mode (PP) setting [V3.54+]
  - PPPoE Login Name (PL) setting [V3.54+]
  - PPPoE Login Password (PD) setting [V3.54+]
  - Connection settings
  - Link Server Login (TL) setting
  - Source IP Filtering (SF) setting
  - Escape Character (EC) setting
  - Link Server Login (TL) parameter
  - Source IP Filtering (SF) parameter
  - Control lines tab
  - Default serial settings tab
  - DST revision history
  - Entire "LinkServer" branch has been added.

21JAN2005 release

- Updated topics:
  - Corrected EM200 pin assignment drawing. Pin assignment itself was correct but the Module on the drawing was erroneously marked as "EM120"
  - EM202-EV->Power Jack and DS202->Power Jack topics referred to incorrect power adaptor numbers. We have corrected this
  - Updated Firmware Revision History

03JAN2005 release

- Updated topics:
  - Corrected mistake for Power Adaptor models. Power adaptors with "small" connector are: ARP-P0005 ("US"), ARP-P0006 ("Europe"), and ARP-P0007 ("OK")
  - Corrected mistake in **Password (PW) setting** description: maximum password length is 6, not 8 characters
  - Corrected the diagram depicting the layout of LEDs on the front of EM202
  - Installation document for VSPDL incorrectly stated that kernels up to 2.5 were supported. Actually, we also support kernel 2.6.x

---

14DEC2004 release

- New topics:
  - Added AN006
  - Added Unable to send a broadcast error message description for DS Manager
  - Added Unable to send a broadcast error message description for Connection Wizard
- Updated topics:
  - DTR and DSR lines were erroneously not shown here: RS232 port pin assignment
  - By mistake, watchdog reset function of the ER line of EM100 was not described

---

30OCT2004 release

- New topics:
  - Created new part- Application Notes and added AN001, AN002, AN003, AN004, and AN005
- Updated topics:
  - Corrected pin number errors in the following topics: LED lines (of EM120), LED lines (of EM200)

21SEP2004 release

Minor error corrections

20SEP2004 release

- New topics:
  - DK100 DIN Rail/Wall Mounting Kit
  - Comparison chart for Ethernet-to-serial Servers
  - DS202 Serial Device Server (and all child topics)
  - Additional features in firmware V3.5x (later deleted)
  - Telnet programming
- Updated topics (this excludes some minor corrections):
  - Device Server Application Firmware (V3.14/V3.51)
  - Network programming
  - Authentication
  - Programming priorities
  - Port Number (PN) setting
  - Access parameters for the address book mode

- Troubleshooting (address book mode)
- Editing the address book (Add, Remove, Edit buttons)

28JUL2004 release

- Power Jack (in EM100-EV Evaluation Board)
- Power Jack (in EM200-EV Evaluation Board)
- Power Jack (in EM202-EV Evaluation Board)
- Power Jack (in DS100 Serial Device Server)
- DHCP
- Firmware version history

- Updated topics (this excludes some minor corrections):
  - Owner Name (ON) setting
  - Device Name (DN) setting
  - Destination IP-address (DI) setting
  - Gateway IP-address (GI) setting
  - Netmask (NM) setting
  - Update history (this very topic)

01JUL2004 ("base") release

Updates are tracked from this point

Index

- A -

access modes 200
  Address Book 205
  Auto-Discovery 201
  COM port 212
Address Book 205
Application firmware 83
Application-to-DS Link 287
Auto-Discovery Access Mode 201

- B -

barcode 428
Broadcast Access Mode 201
buzz 219

- C -

cable
  crossover Ethernet 75
  serial 74
  straight Ethernet 75
Change IP 219
COM Access Mode 212
commands 110
connection delay 355
connection settings 145
Connection Wizard 271
control loads 359
crossover Ethernet cable 75
customization options 343

- D -

Device Server Toolkit 196
DIN Rail 77
DS Manager 197
DS100 63
DS100B 65
DS100B-00 67
DS100R 65
DS100R-03 67
DS202 68, 71
DST 196
DS-to-DS Link 298
Dynamic Addresses 411

- E -

EM100 3
EM100-EV 58
EM120 10
EM120/200-EV 60
EM200 16
em203 23, 34
encapsulation settings 168
error messages 226

- F -

firmware
  upgrading 187
firmware components 80
  Application firmware 83
  Monitor 80
  NetLoader 187

- G -

gateway 406, 412

- H -

HyperTerminal 379

- I -

initialize 109, 216
integration 349

- L -

label 428
Linux 321
loads
  control 359

- M -

messages 110
modem commands 181, 420
  example 427
Module integration 349
Monitor 80, 267

monitor sensors 359
mount
DIN Rail 77
wall 77

- N -

NAT 413
NetLoader 187
Network Address Translation 413
Network settings 132

- O -

On-the-fly 175

- P -

PIC 354
Port Forwarding 418
Port Monitor 267
power adaptor 77
Programmable Interrupt Controller 354
programming 100
example 405

- R -

rail mounting kit 77
remote I/O 359
replies 110
rj203 52
Routing Status Button 217
RS422 76
RS485 76

- S -

same data to several DS 358
sensors
monitor 359
serial cable 74
Serial settings 156
Settings Button 213
SP2 362
SP-to-DS Link 272
status icon 198
Status Messages 222
subnet 406
explained 409

- T -

Terminal Block Adaptor 76
test a connection 397
time to connect 355
Toolkit 196
troubleshooting 204, 209, 212

- V -

Virtual Serial Port 241
VSP Manager 241, 243
VSPD 241
VSPDL 321

- W -

wall mounting kit 77
warning messages 226
wide area network 406
Windows Firewall 362
Windows XP 362