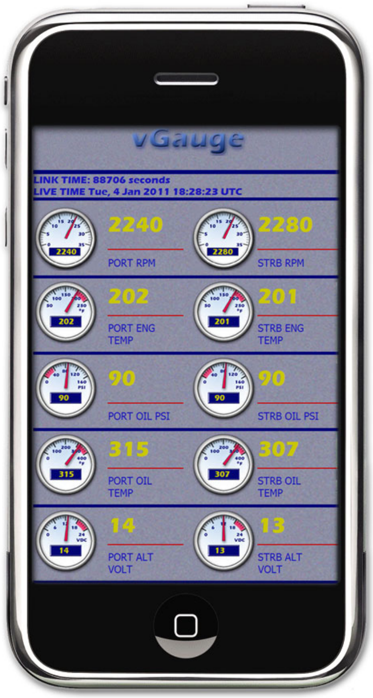
SeaSmart.Net™

Version 1.4.0 – Protocol Specification



Chetco Digital Instruments

Preliminary Specification 122711

WARNING!

USE THIS UNIT ONLY AS AN AID TO MONITORING ENGINE PERFORMANCE INFORMATION.

CAUTION

When showing sensor data, this unit will only show information based on the sender used and its installed position.

The operating and storage temperature for your unit is from -4 degrees to+167 degrees Fahrenheit (-20 to +75 degrees Celsius). Extended storage temperatures higher or lower than specified will cause the liquid crystal display to fail. Neither this type of failure nor its consequences are covered by the warranty. For more information, consult the factory customer service department.

All features and specifications subject to change without notice.

Chetco Digital Instruments may find it necessary to change or end our policies, regulations, and special offers at any time. We reserve the right to do so without notice.

All screens in this manual are simulated.

NOTICE!

Free software upgrades will be available on our website at http:// www.chetcodigital.com as they are released. Please check our website periodically for these and other information as they become available.

Thank you for choosing Chetco Digital Instruments

This device complies with Part 15 of the FCC Rules. Operation is subject to the following two conditions: (1) this device may not cause harmful interference, and (2) this device must accept any interference received, including interference that may cause undesired operation.

Note:

This equipment has been tested and found to comply with the limits for a Class B digital device, pursuant to Part 15 of the FCC Rules. These limits are designed to provide reasonable protection against harmful interference in a residential installation. This equipment generates, uses and can radiate radio frequency energy and, if not installed and used in accordance with the instructions, may cause harmful interference to radio communications. However, there is no guarantee that interference will not occur in a particular installation. If this equipment does cause harmful interference to radio or television reception, which can be determined by turning the equipment off and on, the user is encouraged to try to correct the interference by one or more of the following measures:

- Reorient or relocate the receiving antenna.

- Increase the separation between the equipment and receiver.

- Connect the equipment into an outlet on a circuit different from that to which the receiver is connected.

- Consult the factory customer service department for help.

SPECIFICATIONS

NMEA 2000 Instrumentation Sentences Supported

126992 – System Time
127250 – Vessel Heading
127257 – Vessel Attitude
127251 – Rate of Turn
127488 – Engine Data - Rapid Update
127489 – Engine Data - Dynamic Update
127493 – Transmission Data - Dynamic Update
127505 – Fluid Data - Dynamic Update
127508 – Battery Status - Dynamic Update
127501 – Binary Switch Status - Dynamic Update
130306 – Wind Data
130311 – Environmental Data
130312 – Temperature Data
129025 – Position Data - Rapid
130323 – Weather Station Location Data
129026 – SOG and COG Rapid Update

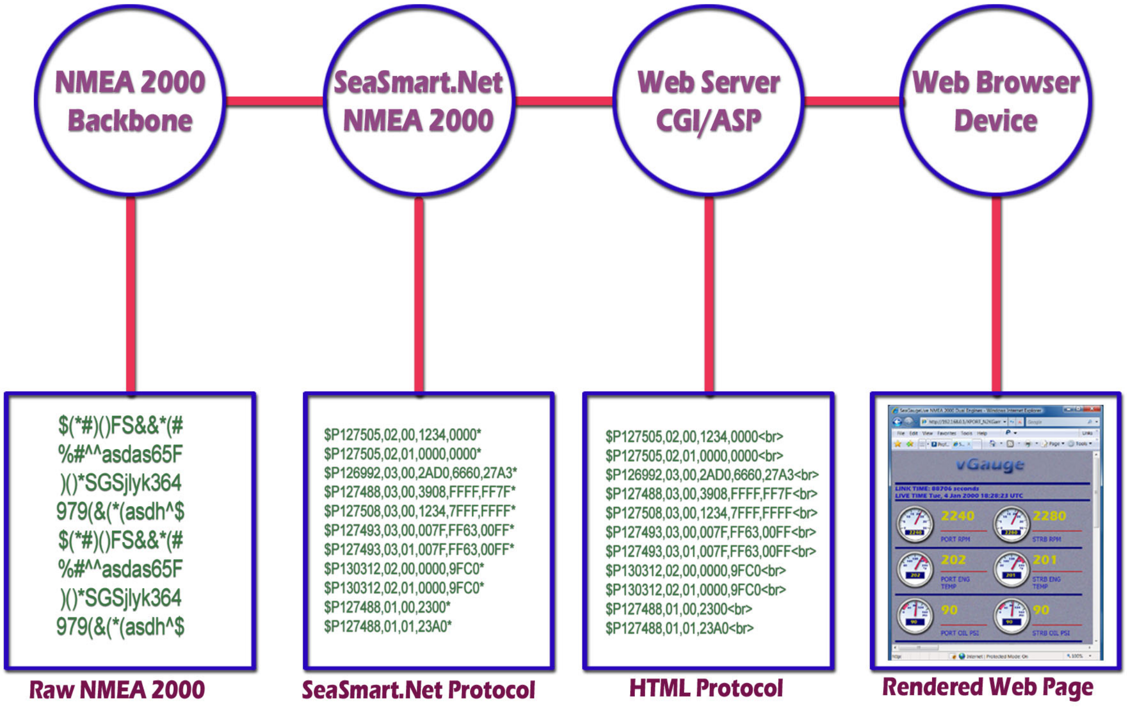
Table of Contents

Introduction

Welcome

Thank you for purchasing a Chetco Digital Instruments product.

SeaSmart.Net™ is a hardware and software system that converts raw NMEA 2000 PGNs into a protocol compatible with standard Web Browsers. SeaSmart.Net™ consists of a NMEA 2000 gateway, HTTP Protocol Translator, and embedded Web Server. The Web Server stores HTML documents which render NMEA 2000 PGNs into real-time graphical display by using the JavaScript programming language. The embedded CGI engine dynamically creates a simple HTML document which is processed by the client Browser.



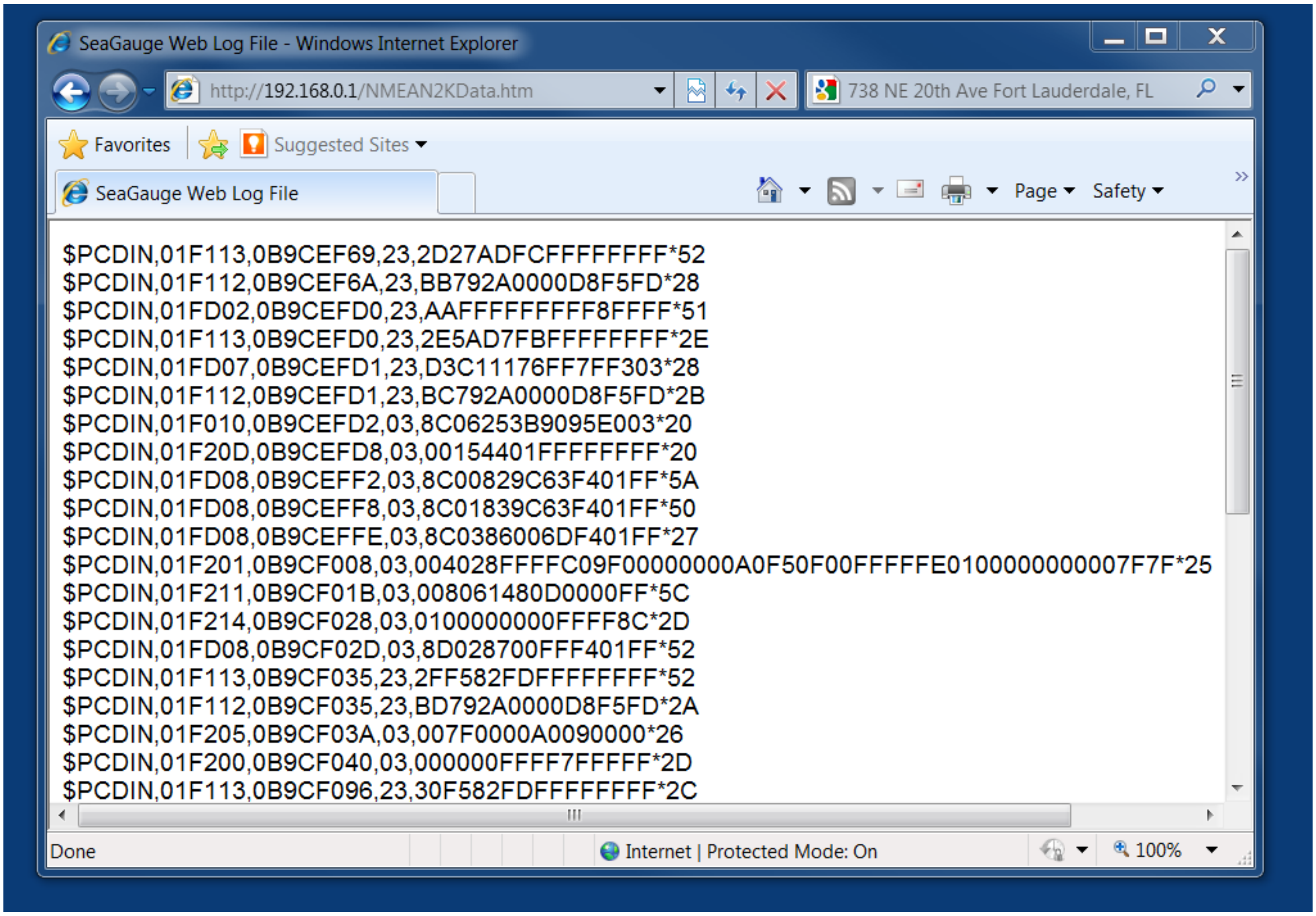
```
$(*#)()FS&&*(#
%#^^asdas65F
)()*SGSjlyk364
979(&(*(asdh^$
$(*#)()FS&&*(#
%#^^asdas65F
)()*SGSjlyk364
979(&(*(asdh^$
```

```
$P127505,02,00,1234,0000*
$P127505,02,01,0000,0000*
$P126992,03,00,2AD0,6660,27A3*
$P127488,03,00,3908,FFFF,FF7F*
$P127508,03,00,1234,7FFF,FFFF*
$P127493,03,00,007F,FF63,00FF*
$P127493,03,01,007F,FF63,00FF*
$P130312,02,00,0000,9FC0*
$P130312,02,01,0000,9FC0*
$P127488,01,00,2300*
$P127488,01,01,23A0*
```

```
$P127505,02,00,1234,0000<br>
$P127505,02,01,0000,0000<br>
$P126992,03,00,2AD0,6660,27A3<br>
$P127488,03,00,3908,FFFF,FF7F<br>
$P127508,03,00,1234,7FFF,FFFF<br>
$P127493,03,00,007F,FF63,00FF<br>
$P127493,03,01,007F,FF63,00FF<br>
$P130312,02,00,0000,9FC0<br>
$P130312,02,01,0000,9FC0<br>
$P127488,01,00,2300<br>
$P127488,01,01,23A0<br>
```

Raw NMEA 2000 data is read directly from the Backbone and translated to SeaSmart.Net protocol so it can be passed on and processed by Web Servers. The Web Servers then in turn format the data into HTML friendly documents which can be retrieved by common Browser based devices. The Browsers render the live data into graphical displays that are easily customizable via standard Web design tools.

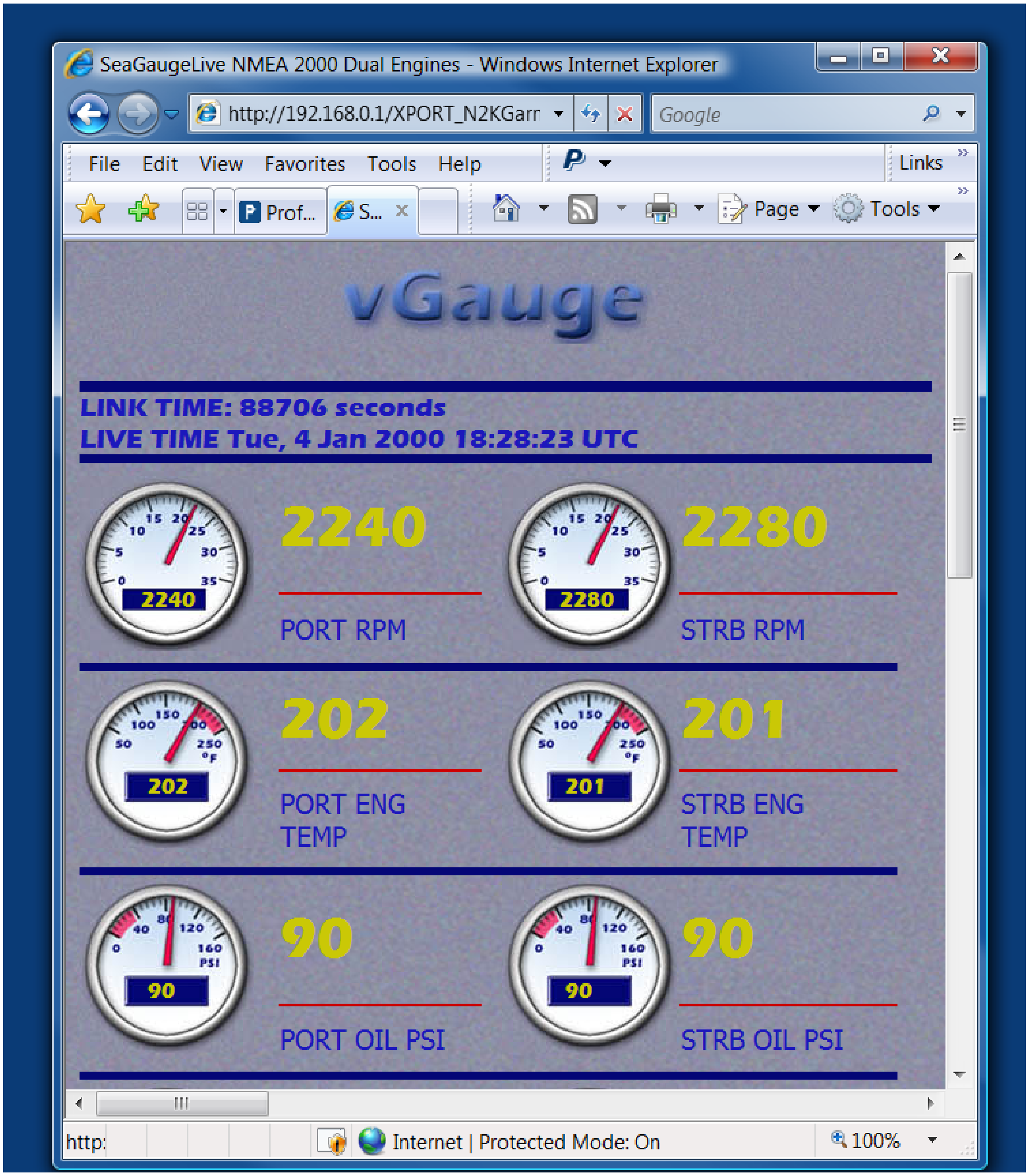
The native NMEA 2000 PGNs are processed in real time to create a new HTML document each second. The following is a example of typical update page.


SeaGauge Web Log File - Windows Internet Explorer
http://192.168.0.1/NMEAN2KData.htm
738 NE 20th Ave Fort Lauderdale, FL
Favorites
Suggested Sites
SeaGauge Web Log File
Page
Safety
Done
Internet | Protected Mode: On
100%

```
$PCDIN,01F113,0B9CEF69,23,2D27ADFCFFFFFFFF*52
$PCDIN,01F112,0B9CEF6A,23,BB792A0000D8F5FD*28
$PCDIN,01FD02,0B9CEFD0,23,AAFFFFFFFFF8FFFF*51
$PCDIN,01F113,0B9CEFD0,23,2E5AD7FBFFFFFFFF*2E
$PCDIN,01FD07,0B9CEFD1,23,D3C11176FF7FF303*28
$PCDIN,01F112,0B9CEFD1,23,BC792A0000D8F5FD*2B
$PCDIN,01F010,0B9CEFD2,03,8C06253B9095E003*20
$PCDIN,01F20D,0B9CEFD8,03,00154401FFFFFFFF*20
$PCDIN,01FD08,0B9CEFF2,03,8C00829C63F401FF*5A
$PCDIN,01FD08,0B9CEFF8,03,8C01839C63F401FF*50
$PCDIN,01FD08,0B9CEFFE,03,8C0386006DF401FF*27
$PCDIN,01F201,0B9CF008,03,004028FFFFC09F00000000A0F50F00FFFFFE0100000000007F7F*25
$PCDIN,01F211,0B9CF01B,03,008061480D0000FF*5C
$PCDIN,01F214,0B9CF028,03,0100000000FFFF8C*2D
$PCDIN,01FD08,0B9CF02D,03,8D028700FFF401FF*52
$PCDIN,01F113,0B9CF035,23,2FF582FDFFFFFFFF*52
$PCDIN,01F112,0B9CF035,23,BD792A0000D8F5FD*2A
$PCDIN,01F205,0B9CF03A,03,007F0000A0090000*26
$PCDIN,01F200,0B9CF040,03,000000FFFF7FFFFF*2D
$PCDIN,01F113,0B9CF096,23,30F582FDFFFFFFFF*2C
```

This HTML document is then available to client Browsers to process and render in any fashion.

A typical JavaScript enabled Browser may use a combination of XMLHttpRequest and images to render as in the following example


SeaGaugeLive NMEA 2000 Dual Engines - Windows Internet Explorer
http://192.168.0.1/XPORT_N2KGarr
Google
File Edit View Favorites Tools Help
Links
Prof...
S...
Page
Tools
vGauge
LINK TIME: 88706 seconds
LIVE TIME Tue, 4 Jan 2000 18:28:23 UTC
2240
PORT RPM
2280
STRB RPM
202
PORT ENG TEMP
201
STRB ENG TEMP
90
PORT OIL PSI
90
STRB OIL PSI
http:
Internet | Protected Mode: On
100%

The purpose of this document is to describe the HTTP compatible translated NMEA 2000 Protocol so that customized Web Pages can be created.

SeaSmart.Net Protocol

Protocol Format

The SeaSmart.Net Protocol is an 7-bit ASCII based format to retain compatibility with all types of Web Browsers. Each received PGN instance is converted in to a sequence of comma separated fields and terminated with the standard NMEA 0183 "*" character and two character (1 byte) checksum.

Each field is a fixed length with a variable number of fields depending on the type of data.

The Protocol header starts with a "$" symbol (0x24 HEX) followed by the letters "PCDIN" then the specified six digit NMEA 2000 PGN number. This PGN number corresponds to the type of data to follow as well as the number of parameters.

The following is an example for the PGN 127505 (Fluid Level)

Start	PGN	Time Stamp	Source ID	PGN DATA	Termination	Check Sum
$PCDIN	01F211	0B9CF01B	03	008061480D0000FF	*	5C

The first four fields in the protocol is fixed length separated by a comma. The PGN data field is variable length depending on the PGN type. The only variable on the overall length is the number of data fields included in each PGN.

The following table summarizes the size for each field.

Field	Size	Range	Type
Start	2 Bytes	-	ASCII Character
PGN Number	6 Bytes	-	ASCII Characters
Time Stamp	8 Bytes	0 - 256	ASCII Hexadecimal
Source ID	2 Bytes	0 - 256	ASCII Hexadecimal
Data Field	1-80 Bytes	0 - 65535	ASCII Hexadecimal
End	1 Bytes	-	ASCII Characters
Check Sum	2 Bytes	-	ASCII Characters

A single update interval may contain one or more PGN sentences depending on the number of newly received NMEA 2000 PGNs. Each unique PGN Instance will have an individual sentence. Only PGNs recognized by the NMEA 2000 Gateway will be processed with all others being ignored to reduce network load.

$PCDIN,01F010 – System Time (126992)

Field	Name	Length	Resolution	Offset	Value	Comment
1	Start	6 Bytes	ASCII CHAR	0	$PCDIN	
2	PGN ID	6 Bytes	ASCII CHAR	0	01F010	
3	Time Stamp	6 Bytes	ASCII HEX	7	3	
4	Source ID	2 Bytes	ASCII HEX	16	0-255	
5	Sequence ID	2 Bytes	ASCII HEX	19	0-255	
6	Reserved	2 Byte	-	21	0XFF	
7	Source	2 Bytes	1	22	1	
					2	
					3	
					4	
					5	Crystal Clock
					6	SeaGauge
8	Days	4 Bytes	1	23	Day	Days since 1970
9	Seconds LB	4 Bytes	1	27	Seconds	
10	Seconds HB	4 Bytes	1	31	Seconds	Seconds * 65536
11	END	1 Bytes	ASCII CHAR	35	*	
12	Check Sum	2 Bytes	ASCII CHAR	36	HEX	XOR from $ to *
Sample	$PCDIN,01F010,000C72E0,09,35F05D3B20462501*5A					
Notes	Total seconds = Seconds HB * 65536 + Seconds LB					

$PCDIN,01F112 – Vessel Heading (127250)

Field	Name	Length	Resolution	Offset	Value	Comment
1	Start	6 Bytes	ASCII CHAR	0	$PCDIN	
2	PGN ID	6 Bytes	ASCII CHAR	0	01F112	
3	Time Stamp	6 Bytes	ASCII HEX	7	3	
4	Source ID	2 Bytes	ASCII HEX	16	0-255	
5	Sequence ID	2 Bytes	ASCII HEX	19	0-255	-
6	Heading	4 Bytes	.0001	21	radians	Degrees = X * 57.29 * .0001
7	Deviation	4 Bytes	.0001	25	radians	Degrees = X * 57.29 * .0001
8	Variation	4 Bytes	.0001	29	radians	Degrees = X * 57.29 * .0001
9	Reference	1 Bytes		33	1	
					2	Apparent
					3	-
					4	Ref to Water
10	Reserved	1 Bytes		34	0XFF	
8	END	2 Bytes	ASCII CHAR	35	*	
9	Check Sum	1 Bytes	ASCII CHAR	36	HEX	XOR from $ to *
Sample	$PCDIN,01F112,000C72EA,09,28C36A0000B40AFD*56					
Notes						

$PCDIN,01F119 – Vessel Attitude (127257)

Field	Name	Length	Resolution	Offset	Value	Comment
1	Start	6 Bytes	ASCII CHAR	0	$PCDIN	
2	PGN ID	6 Bytes	ASCII CHAR	0	01F119	
3	Time Stamp	6 Bytes	ASCII HEX	7	3	
4	Source ID	2 Bytes	ASCII HEX	16	0-255	
5	Sequence ID	2 Bytes	ASCII HEX	19	0-255	-
6	Yaw	4 Bytes	.0001	21	radians	Degrees = X * 57.29 * .0001
7	Pitch	4 Bytes	.0001	25	radians	Degrees = X * 57.29 * .0001
8	Roll	4 Bytes	.0001	29	radians	Degrees = X * 57.29 * .0001
9	Reserved	2 Bytes	ASCII CHAR	33	0XFF	
10	END	1 Bytes	ASCII CHAR	35	*	
11	Check Sum	1 Bytes	ASCII CHAR	36	HEX	XOR from $ to *
Sample	$PCDIN,01F119,000C76CA,09,3DFF7F86FFBF00FF*5B					
Notes						

$PCDIN,01F113 – Rate of Turn (127251)

Field	Name	Length	Resolution	Offset	Value	Comment
1	Start	6 Bytes	ASCII CHAR	0	$PCDIN	
2	PGN ID	6 Bytes	ASCII CHAR	0	01F113	
3	Time Stamp	6 Bytes	ASCII HEX	7	3	
4	Source ID	2 Bytes	ASCII HEX	16	0-255	
5	Sequence ID	2 Bytes	ASCII HEX	19	0-255	-
6	Rate LB	4 Bytes	.0001	21	radians	Degrees = X * 57.29 * .0001
7	Rate HB	4 Bytes	.0001 * 65536	25	radians	Degrees = X * 57.29 * .0001
8	Reserved	6 Bytes	ASCII CHAR	29	0XFFFFFF	
9	END	1 Bytes	ASCII CHAR	35	*	
10	Check Sum	2 Bytes	ASCII CHAR	36	HEX	XOR from $ to *
Sample	$PCDIN,01F113,000C76CA,09,626CA90100FFFFFF*55					
Notes	Total ROT = ROT HB * 65536 + ROT LB					

$PCDIN,01F200 – Engine Data - Rapid Update (127488)

Field	Name	Length	Resolution	Offset	Value	Comment
1	Start	6 Bytes	ASCII CHAR	0	$PCDIN	
2	PGN ID	6 Bytes	ASCII CHAR	0	01F200	
3	Time Stamp	6 Bytes	ASCII HEX	7	3	
4	Source ID	2 Bytes	ASCII HEX	16	0-255	
5	Instance ID	2 Bytes	ASCII HEX	19	0-255	-
6	RPM	4 Bytes	.25	21	Rev/sec	
7	BOOST	4 Bytes	.01	25	Pascal’s	
8	TRIM	2 Bytes	.01	29	Percent	
9	Reserved	4 Bytes	ASCII CHAR	31	0XFFFF	FF
10	END	1 Bytes	ASCII CHAR	35	*	
11	Check Sum	2Bytes	ASCII CHAR	36	HEX	XOR from $ to *
Sample	$PCDIN,01F200,000C7A4F,02,000000FFFF7FFFFF*21					
Notes						

$PCDIN,01F201 – Engine Data - Dynamic Update (127489)

Field	Name	Length	Resolution	Offset	Value	Comment
1	Start	6 Bytes	ASCII CHAR	0	$PCDIN	
2	PGN ID	6 Bytes	ASCII CHAR	0	127489	
3	Time Stamp	6 Bytes	ASCII HEX	7	3	
4	Source ID	2 Bytes	ASCII HEX	16	0-255	
5	Instance ID	2 Bytes	ASCII HEX	19	0-255	-
6	OIL Pressure	4 Bytes	.01	21	Pascal’s	
7	OIL Temp	4 Bytes	.10	25	Kelvin	
8	Engine Temp	4 Bytes	.01	29	Kelvin	
9	Alternator Volts	4 Bytes	.01	33	Volts	
10	Fuel Rate	4 Bytes	.10	37	L/Hr	
11	Engine Hours LB	4 Bytes	.01	41	Seconds	
12	Engine Hours HB	4 Bytes	.01	45	Seconds	
13	Coolant Pressure	4 Bytes	.10	49	Pascal’s	
14	Fuel Pressure	4 Bytes	.01	53	Pascal’s	
15	Reserved	2 Bytes		57	0XFF	
16	Status 1	2 Bytes		59		
17	Status 2	2 Bytes		63		
18	Load	2 Bytes	1%	67	Percent	
19	Torque	2 Bytes	1%	69	Percent	
20	END	1 Bytes	ASCII CHAR	71	*	
21	Check Sum	1 Bytes	ASCII CHAR	72	HEX	XOR from $ to *
Sample	$PCDIN,01F201,000C7E1B,02,000000FFFF407F0005000000000000FFFF000000000000007F7F*24					
Notes	Total Engine Hours = Engine Hours HB * 65536 + Engine Hours LB					

$PCDIN,01F205 – Transmission Data - Dynamic Update (127493)

Field	Name	Length	Resolution	Offset	Value	Comment
1	Start	6 Bytes	ASCII CHAR	0	$PCDIN	
2	PGN ID	6 Bytes	ASCII CHAR	0	01F205	
3	Time Stamp	6 Bytes	ASCII HEX	7	3	
4	Source ID	2 Bytes	ASCII HEX	16	0-255	
5	Instance ID	2 Bytes	ASCII HEX	19	0-255	
6	Gear	1 Byte		21		
7	Reserved	1 Byte		22		
8	Tran Pressure	4 Bytes	.01	23	Pascal’s	
9	Tran Temp	4 Bytes	.01	27	Kelvin	
10	Tran Status	4 Bytes		31		
11	Reserved	2 Bytes		33	0XFF	
12	END	1 Bytes	ASCII CHAR	35	*	
13	Check Sum	1 Bytes	ASCII CHAR	36	HEX	XOR from $ to *
Sample	$PCDIN,01F205,000C7E33,02,007F0000000C0000*21					
Notes						

$PCDIN,01F211 – Fluid Data - Dynamic Update(127505)

Field	Name	Length	Resolution	Offset	Value	Comment
1	Start	6 Bytes	ASCII CHAR	0	$PCDIN	
2	PGN ID	6 Bytes	ASCII CHAR	0	01F211	
3	Time Stamp	6 Bytes	ASCII HEX	7	3	
4	Source ID	2 Bytes	ASCII HEX	16	0-255	
5	Instance ID	1 Bytes	ASCII HEX	19	0-255	
6	Fluid Type	1 Bytes	-	20	0 - 15	Type of Fluid Tank
7	Level	4 Bytes	.01	21	Percent	
8	Capacity LB	4 Bytes	.10	25	Liters	
9	Capacity HB	4 Bytes	.10 * 65536	29	Liters	
10	Reserved	2 Bytes		33	0XFF	
11	END	1 Bytes	ASCII CHAR	35	*	
12	Check Sum	1 Bytes	ASCII CHAR	36	HEX	XOR from $ to *
Sample	$PCDIN,127505,04,00,0001,9500,0000,0000*7E					
Notes	Total Capacity = Capacity HB * 65536 + Capacity LB					

$PCDIN,01F209 – Trip Parameters - Dynamic Update(127497)

Field	Name	Length	Resolution	Offset	Value	Comment
1	Start	6 Bytes	ASCII CHAR	0	$PCDIN	
2	PGN ID	6 Bytes	ASCII CHAR	0	01F211	
3	Time Stamp	6 Bytes	ASCII HEX	7	3	
4	Source ID	2 Bytes	ASCII HEX	16	0-255	
5	Instance ID	2 Bytes	ASCII HEX	19	0-255	
6	Fuel Used	4 Bytes	.001	21	Cubic Meters	
7	Fuel Rate Avg	4 Bytes	.0001	25	Cubic Meters	
8	Fuel Rate Ecno	4 Bytes	.0001	29	Cubic Meters	
9	Instantaneous	4 Bytes	.0001	31	Cubic Meters	
11	END	1 Bytes	ASCII CHAR	35	*	
12	Check Sum	1 Bytes	ASCII CHAR	36	HEX	XOR from $ to *
Sample	$PCDIN,01F209,0B9CF166,10,03C19800003B053B05*22<BR>					
Notes	Total Capacity = Capacity HB * 65536 + Capacity LB					

$PCDIN,01F214 – Battery Status - Dynamic Update (127508)

Field	Name	Length	Resolution	Offset	Value	Comment
1	Start	6 Bytes	ASCII CHAR	0	$PCDIN	
2	PGN ID	6 Bytes	ASCII CHAR	0	01F214	
3	Time Stamp	6 Bytes	ASCII HEX	7	3	
4	Source ID	2 Bytes	ASCII HEX	16	0-255	
5	Instance ID	2 Bytes	ASCII HEX	19	0-255	
6	Battery Volts	4 Bytes	.01	21	Volts	
7	Battery Current	4 Bytes	.01	25	AMPS	
8	Battery Temp	4 Bytes	.01	29	Kelvin	
9	Sequence ID	2 Bytes		33	0-255	
10	END	1 Bytes	ASCII CHAR	35	*	
11	Check Sum	1 Bytes	ASCII CHAR	36	HEX	XOR from $ to *
Sample	$PCDIN,01F214,000C7E2C,02,01B0040000FFFF36*20					
Notes						

$PCDIN,01F212 – DC Status - Detail (127506)

Field	Name	Length	Resolution	Offset	Value	Comment
1	Start	6 Bytes	ASCII CHAR	0	$PCDIN	
2	PGN ID	6 Bytes	ASCII CHAR	0	01F214	
3	Time Stamp	6 Bytes	ASCII HEX	7	3	
4	Source ID	2 Bytes	ASCII HEX	16	0-255	
5	Sequence ID	2 Bytes	ASCII HEX	19	0-255	
5	Instance ID	2 Bytes	ASCII HEX	21	0-255	
	Battery Type	2 Bytes	1	23	0-255	0 = Battery
						1 = Alternator
						2 = Converter
						3 = Solar
						4 = Wind
6	State Of Charge	2 Bytes	1	25	Percent	
7	State of Health	2 Bytes	1	27	Percent	
8	Time Remain	4 Bytes	1	29	Minutes	
9	Ripple Volts	2 Bytes	1 mV	33	0-255	
10	END	1 Bytes	ASCII CHAR	35	*	
11	Check Sum	2 Bytes	ASCII CHAR	36	HEX	XOR from $ to *
Sample	$PCDIN,01F212,0DBBEA42,00,30F00F00004088FF*57					
Notes						

$PCDIN,01F219 – DC Config - Detail (127513)

Field	Name	Length	Resolution	Offset	Value	Comment
1	Start	6 Bytes	ASCII CHAR	0	$PCDIN	
2	PGN ID	6 Bytes	ASCII CHAR	0	01F214	
3	Time Stamp	6 Bytes	ASCII HEX	7	3	
4	Source ID	2 Bytes	ASCII HEX	16	0-255	
5	Instance ID	1 Bytes	ASCII HEX	19	0-255	
	Battery Type	1 Bytes	1	20	0-255	0 = Flooded
						1 = Gel
						2 = AGM
6	Equalization	1 Bytes	1	21	Yes/No	0
7	Reserved	1 Bytes	1	22	0XF	
8	Nom Volts	1 Bytes	1	23	Index	0 = 6 Volts
						1 = 12 Volts
						2 = 24 Volts
						3 = 32 Volts
						4 = 36 Volts
						5 = 42 Volts
						6 = 48 Volts
9	Chemistry	1 Bytes	1	24	Index	0 = Lead
						1 = Lion
						2 = NiCad
						3 = ZnO
						4 = NiMH
	Capacity	4 Bytes	1	25	1 Amp Hour	
	Temp Coeff	2 Bytes	1	29	1%/C	
	Peukert	2 Bytes	0.002	31		
	Charge Factor	2 Bytes	1	33	Percent	
10	END	1 Bytes	ASCII CHAR	35	*	
11	Check Sum	2 Bytes	ASCII CHAR	36	HEX	XOR from $ to *
Sample	$PCDIN,01F219,0D98AF03,00,00F00F00004088FF*58					
Notes						

$PCDIN,01F20D – Binary Switch Status - Dynamic Update (127501)

Field	Name	Length	Resolution	Offset	Value	Comment
1	Start	6 Bytes	ASCII CHAR	0	$PCDIN	
2	PGN ID	6 Bytes	ASCII CHAR	0	01F20D	
3	Time Stamp	6 Bytes	ASCII HEX	7	3	
4	Source ID	2 Bytes	ASCII HEX	16	0-255	
5	Instance ID	2 Bytes	ASCII HEX	19	0-255	
6	Switch Status LB 0	4 Bytes	ON, OFF	21	Switch 0-7	00 = OFF, 01 = ON
7	Switch Status LB 1	4 Bytes	ON, OFF	25	Switch 8-15	10 = UNDEF, 11 = ERR
8	Switch Status LB 2	4 Bytes	ON, OFF	29	Switch 16-24	
9	Switch Status LB 3	4 Bytes	ON, OFF	33	Switch 24-29	
10	END	1 Bytes	ASCII CHAR	35	*	
11	Check Sum	2 Bytes	ASCII CHAR	36	HEX	XOR from $ to *
Sample	$PCDIN,01FD02,000C8377,09,03C3007F0AFAFFFF*54					
Notes						

$PCDIN,01FD02 – Wind Data (130306)

Field	Name	Length	Resolution	Offset	Value	Comment
1	Start	6 Bytes	ASCII CHAR	0	$PCDIN	
2	PGN ID	6 Bytes	ASCII CHAR	0	01FD02	
3	Time Stamp	6 Bytes	ASCII HEX	7	3	
4	Source ID	2 Bytes	ASCII HEX	16	0-255	
5	Sequence ID	2 Bytes		19	0-255	
6	Speed	4 Bytes	.01	21	m/sec	Knots = X * 1.9438 * .01
7	Direction	4 Bytes	.0001	25	radians	Degrees = X * 57.29 * .0001
8	Reference	2 Bytes		29	1	-
					2	Apparent
					3	-
					4	Ref to Water
9	Reserved	4 Bytes		31	0XFFFF	
10	END	1 Bytes	ASCII CHAR	35	*	
11	Check Sum	2 Bytes	ASCII CHAR	36	HEX	XOR from $ to *
Sample	$PCDIN,01FD02,000C8472,09,04C300487FF8FFFF*55					
Notes						

$PCDIN,01FD07 – Environmental Data (130311)

Field	Name	Length	Resolution	Offset	Value	Comment
1	Start	6 Bytes	ASCII CHAR	0	$PCDIN	
2	PGN ID	6 Bytes	ASCII CHAR	0	01FD07	
3	Time Stamp	6 Bytes	ASCII HEX	7	3	
4	Source ID	2 Bytes	ASCII HEX	16	0-255	
5	Sequence ID	2 Bytes		19	0-255	
6	Instance ID	2 Bytes		21	0-255	Lower Byte
7	Air Temp	4 Bytes	.01	23	Kelvin	
8	Humidity	4 Bytes	.01	27	%	
9	Barometric	4 Bytes	.01	31	inHg	
10	END	1 Bytes	ASCII CHAR	35	*	
11	Check Sum	2 Bytes	ASCII CHAR	36	HEX	XOR from $ to *
Sample	$PCDIN,01FD07,000C8473,09,80C17D73FF7FF703*28					
Notes						

$PCDIN,01FD08 – Temperature Data (130312)

Field	Name	Length	Resolution	Offset	Value	Comment
1	Start	6 Bytes	ASCII CHAR	0	$PCDIN	
2	PGN ID	6 Bytes	ASCII CHAR	0	01FD08	
3	Time Stamp	6 Bytes	ASCII HEX	7	3	
4	Source ID	2 Bytes	ASCII HEX	16	0-255	
5	Sequence ID	2 Bytes		19	0-255	
6	Instance ID	2 Bytes		21	0-255	
7	Temp Type	2 Bytes		23	0-255	0x80-8x8F – CDI type
8	Temp	4 Bytes	.01	25	Kelvin	
9	Set Temp	4 Bytes	.01	29	Kelvin	
10	Reserved	2 Bytes		33	0XFF	
11	END	1 Bytes	ASCII CHAR	35	*	
12	Check Sum	2 Bytes	ASCII CHAR	36	HEX	XOR from $ to *
Sample	$PCDIN,01FD08,000C85DD,02,3800821C7DF401FF*54					
Notes						

$PCDIN,01FD13 – Weather Station Location Data (130323)

Field	Name	Length	Resolution	Offset	Value	Comment
1	Start	6 Bytes	ASCII CHAR	0	$PCDIN	
2	PGN ID	6 Bytes	ASCII CHAR	0	01FD13	
3	Time Stamp	6 Bytes	ASCII HEX	7	3	
4	Source ID	2 Bytes	ASCII HEX	16	0-255	
5	Mode	1 Byte		19		
6	Reserved	1 Byte		20		
7	Days	4 Bytes	1	25	Day	Days since 1970
8	Seconds LB	4 Bytes	1	27	Seconds	
9	Seconds HB	4 Bytes	1	31	Seconds	Seconds * 65536
10	Latitude LB	4 Bytes	.0000001	33	degrees	
11	Latitude HB	4 Bytes	.0000001	37	degrees	
12	Longitude LB	4 Bytes	.0000001	41	degrees	
13	Longitude HB	4 Bytes	.0000001	45	degrees	
14	Speed	4 Bytes	.01	49	m/sec	Knots = X * 1.9438 * .01
15	Direction	4 Bytes	.0001	53	radians	Degrees = X * 57.29 * .0001
16	Reference	2 Bytes		57	1	-
					2	Apparent
					3	-
					4	Ref to Water
17	Reserved	2 Bytes		59	0XFF	
18	Wind Gusts	4 Bytes		61		
19	Barometric	4 Bytes	.01	65	inHg	
20	Air Temp	4 Bytes	.01	69	Kelvin	
21	Station ID	4 Bytes		73		
22	Station Name	4 Bytes		75		
23	END	1 Bytes	ASCII CHAR	79	*	
24	Check Sum	2 Bytes	ASCII CHAR	80	HEX	XOR from $ to *
Sample	$PCDIN,01FD13,000C858B,09,F05D3B700926013A611019CB29EEB5C300F90AFAFFFFF7037D7302010201*5C					
Notes	Degrees = (X HB * 65536 + X LB) * .0000001					

$PCDIN,01F802 – SOG and COG Rapid Update (129026)

Field	Name	Length	Resolution	Offset	Value	Comment
1	Start	6 Bytes	ASCII CHAR	0	$PCDIN	
2	PGN ID	6 Bytes	ASCII CHAR	0	01F802	
3	Time Stamp	6 Bytes	ASCII HEX	7	3	
4	Source ID	2 Bytes	ASCII HEX	16	0-255	
5	Sequence ID	2 Bytes		19	0-255	
6	Reference	1 Byte		21	1	-
					2	Apparent
					3	-
					4	Ref to Water
7	Reserved	1 Byte		22	F	
8	Speed	4 Bytes	.01	27	m/sec	Knots = X * 1.9438 * .01
9	Direction	4 Bytes	.0001	23	radians	Degrees = X * 57.29 * .0001
10	Speed	4 Bytes	.01	27	m/sec	Knots = X * 1.9438 * .01
11	Reserved	4 Bytes		31	0XFFFF	
12	END	1 Bytes	ASCII CHAR	35	*	
13	Check Sum	2 Bytes	ASCII CHAR	36	HEX	XOR from $ to *
Sample	$PCDIN,01F802,000C8286,09,3AFC8CCA0500FFFF*58					
Notes						

$PCDIN,01F801 –Position Data – Rapid (129025)

Field	Name	Length	Resolution	Offset	Value	Comment
1	Start	6 Bytes	ASCII CHAR	0	$PCDIN	
2	PGN ID	6 Bytes	ASCII CHAR	0	01F801	
3	Time Stamp	6 Bytes	ASCII HEX	7	3	
4	Source ID	2 Bytes	ASCII HEX	16	0-255	
5	Latitude LB	4 Bytes	.0000001	19	degrees	
6	Latitude HB	4 Bytes	.0000001	23	degrees	
7	Longitude LB	4 Bytes	.0000001	27	degrees	
8	Longitude HB	4 Bytes	.0000001	31	degrees	
9	END	1 Bytes	ASCII CHAR	35	*	
10	Check Sum	2 Bytes	ASCII CHAR	36	HEX	XOR from $ to *
Sample	$PCDIN,129025,04,00,A2C9,190A,2C81,B603*7A					
Notes	Degrees = (X HB * 65536 + X LB) * .0000001					

Command Protocol Format

The SeaSmart.Net Command Protocol is an 7-bit ASCII based format and is used to configure the SeaSmart.net adapter. Each Command contains a sequence of comma separated fields and terminated with the standard NMEA 0183 "*" character and two character (1 byte) checksum.

Commands can issued over TCP/UDP, Web GET, and serial port interfaces

The Protocol header starts with a "$" symbol (0x24 HEX) followed by the letters "PCDIC" then the specified six digit NMEA 2000 PGN number equal to 0xFFFFFF, Time Stamp, Source ID (equal to 0xFF) and COMMAND Hexadecimal String.

The following is an example for command "ENABLE ALL RX"

Start	PGN	Time Stamp	Source ID	COMMAND DATA	Termination	Check Sum
$PCDIC	FFFFFF	0B9CF01B	FF	110200	*	5C

The first four fields in the protocol is fixed length separated by a comma. The Command data field is variable length depending on the command type. The only variable on the overall length is the number of data fields included in each command.

The following table summarizes the size for each field.

Field	Size	Range	Type
Start	6 Bytes	$PCDIC	ASCII Character
PGN Number	6 Bytes	0xFFFFFF	ASCII Characters
Time Stamp	8 Bytes	0 - 256	ASCII Hexadecimal
Source ID	2 Bytes	0xFF	ASCII Hexadecimal
Command Data Field	1-80 Bytes	0 - 65535	ASCII Hexadecimal
End	1 Bytes	-	ASCII Characters
Check Sum	2 Bytes	-	ASCII Characters

Some Command Actions may require more than one command string. For example, updating a RX PGN List element requires sending the specified PGN Enable followed by the Enable List Command to become active.

$PCDIC – ENABLE ALL RX

Field	Name	Length	Resolution	Offset	Value	Comment
1	Start	6 Bytes	ASCII CHAR	0	$PCDIC	
2	PGN ID	6 Bytes	ASCII CHAR	7	0xFFFFFF	
3	Time Stamp	6 Bytes	ASCII HEX	14	3	
4	Source ID	2 Bytes	ASCII HEX	16	0xFF	
5	Command ID	2 Bytes	ASCII HEX	19	0x11	-
6	Enable	2 Bytes	ASCII HEX	21	0x02	02=enable, 01=disable
7	Reserved	2 Bytes	ASCII CHAR	23	0X00	
8	END	1 Bytes	ASCII CHAR	25	*	
9	Check Sum	1 Bytes	ASCII CHAR	26	HEX	XOR from $ to *
Sample	$PCDIC,FFFFFF,000C76CA,FF,110200*5B					
Notes	Single Command					

$PCDIC – USED STORED RX LIST

Field	Name	Length	Resolution	Offset	Value	Comment
1	Start	6 Bytes	ASCII CHAR	0	$PCDIC	
2	PGN ID	6 Bytes	ASCII CHAR	7	0xFFFFFF	
3	Time Stamp	6 Bytes	ASCII HEX	14	3	
4	Source ID	2 Bytes	ASCII HEX	16	0xFF	
5	Command ID	2 Bytes	ASCII HEX	19	0x11	-
6	Enable	2 Bytes	ASCII HEX	21	0x01	02=enable, 01=disable
7	Reserved	2 Bytes	ASCII CHAR	23	0X00	
8	END	1 Bytes	ASCII CHAR	25	*	
9	Check Sum	1 Bytes	ASCII CHAR	26	HEX	XOR from $ to *
Sample	$PCDIC,FFFFFF,000C76CA,FF,110100*5B					
Notes	Single Command					

$PCDIC – SAVE CURRENT RX LIST (commit to EEPROM)

Field	Name	Length	Resolution	Offset	Value	Comment
1	Start	6 Bytes	ASCII CHAR	0	$PCDIC	
2	PGN ID	6 Bytes	ASCII CHAR	7	0xFFFFFF	
3	Time Stamp	6 Bytes	ASCII HEX	14	3	
4	Source ID	2 Bytes	ASCII HEX	16	0xFF	
5	Command ID	2 Bytes	ASCII HEX	19	0x01	-
9	END	1 Bytes	ASCII CHAR	21	*	
7	Check Sum	1 Bytes	ASCII CHAR	23	HEX	XOR from $ to *
Sample	$PCDIC,FFFFFF,000C76CA,FF,01*5B					
Notes	Single Command – paired with Enable RX or TX PGN					

$PCDIC – USE CURRENT RX List

Field	Name	Length	Resolution	Offset	Value	Comment
1	Start	6 Bytes	ASCII CHAR	0	$PCDIC	
2	PGN ID	6 Bytes	ASCII CHAR	7	0xFFFFFF	
3	Time Stamp	6 Bytes	ASCII HEX	14	3	
4	Source ID	2 Bytes	ASCII HEX	16	0xFF	
5	Command ID	2 Bytes	ASCII HEX	19	0x4B	-
9	END	1 Bytes	ASCII CHAR	21	*	
10	Check Sum	1 Bytes	ASCII CHAR	23	HEX	XOR from $ to *
Sample	$PCDIC,FFFFFF,000C76CA,FF,4B*5B					
Notes	Single Command – Used with Enable RX or TX PGN					

$PCDIC – ADD RX PGN TO LIST

Field	Name	Length	Resolution	Offset	Value	Comment
1	Start	6 Bytes	ASCII CHAR	0	$PCDIC	
2	PGN ID	6 Bytes	ASCII CHAR	7	0xFFFFFF	
3	Time Stamp	6 Bytes	ASCII HEX	14	3	
4	Source ID	2 Bytes	ASCII HEX	16	0xFF	
5	Command ID	2 Bytes	ASCII HEX	19	0x46	-
6	PGN	6 Bytes	ASCII HEX	21	0x000000	PGN Number in Hex
7	Enable	4 Bytes	ASCII HEX	27	0x0001	0x0001 = enable
8	Mask	8 Bytes	ASCII HEX	31	0x00FFFF03	Default = match all
9	END	1 Bytes	ASCII CHAR	39	*	
10	Check Sum	1 Bytes	ASCII CHAR	40	HEX	XOR from $ to *
Sample	$PCDIC,FFFFFF,00A55979,00,4612F101000100FFFF03*22					
Notes	Dual Command – must follow with USE CURRENT RX LIST command to activate					

$PCDIC – ADD TX PGN TO LIST

Field	Name	Length	Resolution	Offset	Value	Comment
1	Start	6 Bytes	ASCII CHAR	0	$PCDIC	
2	PGN ID	6 Bytes	ASCII CHAR	7	0xFFFFFF	
3	Time Stamp	6 Bytes	ASCII HEX	14	3	
4	Source ID	2 Bytes	ASCII HEX	16	0xFF	
5	Command ID	2 Bytes	ASCII HEX	19	0x47	-
6	PGN	6 Bytes	ASCII HEX	21	0x000000	PGN Number in Hex
7	Enable	4 Bytes	ASCII HEX	27	0x0001	0x0001 = enable
8	Rate	8 Bytes	ASCII HEX	31	0x32000000	Rate (X 100 mSec)
8	Interval	8 Bytes	ASCII HEX	31	0x32000000	Interval (X 100 mSec)
9	END	1 Bytes	ASCII CHAR	39	*	
10	Check Sum	1 Bytes	ASCII CHAR	40	HEX	XOR from $ to *
Sample	$PCDIC,FFFFFF,00A55979,00,4712F1010001320000003200000*22					
Notes	Dual Command – must follow with USE CURRENT RX LIST command to activate					

Protocol Polling

The SeaSmart.Net Protocol is embedded in a simple HTML document created dynamically from a Browser GET request. One simple method is to use the XMLHttpRequest Object to issue a GET of the target HTML file from the server.

```
var datafile = window.location.href.substring(0,
     window.location.href.lastIndexOf("/") + 1) +
     "NMEAN2KData.htm?";

objXml = new XMLHttpRequest();


objXml.open("GET",datafile , true);
```

This will create a new Xml Object and copy the contents of the NMEAN2KData.htm file located in the same directory as the calling Web Page.

Use the objXml.onreadystatechange event to determine when the new data is ready.

```
objXml .onreadystatechange = function()
{
      if(objXml.readyState == 4)
      {
            if(objXml.status == 200)
            {
                  mydata= objXml .responseText;
            }
      }
}
```

The **mydata** object will now contain a copy of the NMEAN2KData.htm file

Protocol Parsing

The SeaSmart.Net Protocol is easily parsed in JavaScript by using the Java .split function

```
            mydata= objXml .responseText;

            mySubStrings = mydata.split("$PCDIN");
```

This will create a variable number of array elements based on the starting “$PCDIN” character that correspond to each of the received PGN sentences.

From there, each PGN can be decoded by fixed reference to character position in the string.

Start by extracting the PGN number to determine the number and types of data fields.

```
for( myIndex = 0; myIndex < myArrayLength; myIndex++)
{

// First extract the six character PGN number
      myHexStr = mySubStrings[myIndex];
      myHexStr = myHexStr.substr(0,6) ;
      myPGN = parseInt(myHexStr);

      // Then Parse based on PGN Number
      if(myPGN == 130311) // Environ Data
      {
            myDataLabels[myIndex]="BARO";
            myHexStr = mySubStrings[myIndex];
            myHexStr = myHexStr.substr(18,4) ;
            myHexStr = "0x" + myHexStr;
            myPGNValue = parseInt(myHexStr);
            myHexStr =((myPGNValue * 0.0295229) );

            myDataValues[myIndex]=myHexStr.toFixed(2) ;
            myDialIndexes[myIndex]=Math.floor(((myPGNValue*0.0295229) -28)*64 );

            myDataLabels[myIndex]="AIR TEMP";
            myHexStr = mySubStrings[myIndex];
            myHexStr = myHexStr.substr(13,4) ;
            myHexStr = "0x" + myHexStr;
            myPGNValue = parseInt(myHexStr);
            myDataValues[myIndex]=Math.floor(((myPGNValue * 0.018) - 459));

            myDialIndexes[myIndex] =Math.floor(((myPGNValue * 0.018) - 459)*2);

      }

}
```

Refer to the SeaSmart.Net Protocol descriptions for definitions on each of the received PGN data structures

The resulting data variables can be easily written to the target Browser Window using a variety of methods. The document Object is one option

```
document.getElementById("dataLabel7").innerHTML = myDataLabels[myIndex];
document.getElementById("dialValue7").innerHTML = myDataValues[myIndex];
```

HTTP POST Protocol

SeaSmart.Net can support forwarding data to external Web Servers on the Local or Global network by using the HTTP POST Protocol. When this option is enabled, a TCP connection is made to the target IP address (or Host Name if DNS is available) and incoming NMEA 2000 data transferred in blocks using the HTTP POST once a second. The TCP connection is maintained as long as new data is available within the interval.

The SeaSmart.Net module is configured with the HOST IP Address or HOST NAME and the target file to handle the POST which is usually an Active Server Script (ASP) or CGI Script depending on the Server Platform type.

A typical POST message may look like:

```
POST /XPORTN2KWrite.asp HTTP/1.1
Host: 192.168.0.1:80
Content-Length: 361
Content-type: application/x-www-form-urlencoded

Name=
$PCDIN,01F211,0B9CF01B,03,008061480D0000FF*5C
$PCDIN,01F214,0B9CF028,03,0100000000FFFF8C*2D
$PCDIN,01FD08,0B9CF02D,03,8D028700FFF401FF*52
$PCDIN,01F113,0B9CF035,23,2FF582FDFFFFFFFF*52
$PCDIN,01F112,0B9CF035,23,BD792A0000D8F5FD*2A
$PCDIN,01F205,0B9CF03A,03,007F0000A0090000*26
$PCDIN,01F200,0B9CF040,03,000000FFFF7FFFFF*2D
$PCDIN,01F113,0B9CF096,23,30F582FDFFFFFFFF*2C
$PCDIN,01F112,0B9CF096,23,BE792A0000D8F5FD*22
```

Where XPORTN2KWrite.asp is the handler and 192.168.0.1:80 is the host IP address/Port.

The number of PGNs transferred in each POST is dependent on the number received within the one second interval. The hosting server is then responsible for processing the data and passing it along. In most cases, the .ASP script will just write it to a local .htm file so that Browser Apps can access it.

The resulting HTML file would be.

```
<html>
<body>
$PCDIN,01F211,0B9CF01B,03,008061480D0000FF*5C<BR>
$PCDIN,01F214,0B9CF028,03,0100000000FFFF8C*2D<BR>
$PCDIN,01FD08,0B9CF02D,03,8D028700FFF401FF*52<BR>
$PCDIN,01F113,0B9CF035,23,2FF582FDFFFFFFFF*52<BR>
$PCDIN,01F112,0B9CF035,23,BD792A0000D8F5FD*2A<BR>
$PCDIN,01F205,0B9CF03A,03,007F0000A0090000*26<BR>
$PCDIN,01F200,0B9CF040,03,000000FFFF7FFFFF*2D<BR>
$PCDIN,01F113,0B9CF096,23,30F582FDFFFFFFFF*2C<BR>
$PCDIN,01F112,0B9CF096,23,BE792A0000D8F5FD*22<BR>
 </body>
</html>
```

Sample .ASP script

The following sample .ASP file simply takes the incoming post data and writes out to existing file while replacing the "*" with "<BR>" to be compatible with HTML syntax.

```
<html>

<head>
<meta name="GENERATOR" content="Microsoft FrontPage 5.0">
<title>Main</title>

<meta name="Microsoft Border" content="none">
</head>

<body>

<%
Dim name, oldN2KData
' Declare our vaiables
Dim objFSO, objCountFile  ' object vars for FSO and File
Dim strCountFileName      ' filename of count text file
Dim iCount                ' count variable

Dim objOldFSO, objOldDataFile  ' object vars for FSO and File
Dim strOldDataFileName      ' filename of count text file
Dim iOldDataCount           ' count variable

Dim myHTTPHeader
Dim myHTTPFooter

myHTTPHeader = "<HTML><head><Title> SeaGauge Web Log File </Title></head><Body>"
myHTTPFooter = "</Body></HTML>"


name = Request.Form("Name")


' Old N2K data File Name
strOldDataFileName = Server.MapPath("OldNMEAN2KData.txt")

' Create FileSystemObject to deal with file access
Set objOldFSO = Server.CreateObject("Scripting.FileSystemObject")

' Open Old file and get a text stream to new one
Set objOldDataFile = objOldFSO.OpenTextFile(strOldDataFileName, 1 )

  ' Read from the file.
   If objOldDataFile.AtEndOfStream Then
       oldN2KData = ""
   Else
       oldN2KData = objOldDataFile.ReadAll
   End If

objOldDataFile.Close
```

```
'  file's filename
strCountFileName = Server.MapPath("NMEAN2KData.htm")

' Create FileSystemObject to deal with file access
Set objFSO = Server.CreateObject("Scripting.FileSystemObject")

' Overwrite existing file and get a text stream to new one
Set objCountFile = objFSO.CreateTextFile(strCountFileName, True)

' Write updated count
objCountFile.Write myHTTPHeader

oldN2KData = Replace(oldN2KData,"*","<BR>")

' Write Old Data
objCountFile.WriteLine oldN2KData

oldN2KData = Replace(name,"*","<BR>")
' Write new data
objCountFile.WriteLine oldN2KData

' Write updated count
objCountFile.Write myHTTPFooter

' Close the file and destroy the object
objCountFile.Close
Set objCountFile = Nothing

' Open Old file and get a text stream to new one
Set objOldDataFile   = objOldFSO.OpenTextFile(strOldDataFileName, 2, True)

' Write new data
objOldDataFile.WriteLine name

objOldDataFile.Close
Set objOldDataFile = Nothing

' Destroy the FSO object
Set objFSO = Nothing
Set objOldFSO = Nothing

%>

</body>
</html>
```

Embedded Server vs External Server

There is a small difference between Web Pages designed for the Embedded Web Server and those that Run on an External Web Server.

The Embedded Server uses a special form of CGI Script to capture the translated data from the NMEA 2000 backbone and renders it for further on processing by Browser Apps that may call for it. Since the Embedded Server has no mass storage device, it keeps the incoming data temporally in memory until the receive buffers are filled, after which it will dump the oldest data as newer data arrives. With a buffer size of over 32 Kbytes, that is over 10 minutes of traffic on a heavily loaded bus.

Therefore, the Embedded Server CGI Script will service Browser Requests from Memory and not file Storage.

The External Web Server on the other hand usually has plenty of mass storage so that data can be written to files and files overwritten as new data arrives.

With this in mind, the Get Request for the Embedded Server is slightly different then the External Server.

Embedded Server GET Request file Name

```
var datafile = window.location.href.substring(0,
    window.location.href.lastIndexOf("/") + 1) +
    "GetNMEANData?";
```

Calls CGI script to grab data directly from NMEA 2000 Bus

External Server GET Request file Name

```
var datafile = window.location.href.substring(0,
    window.location.href.lastIndexOf("/") + 1) +
    "GetNMEANData.htm?";
```

Gets contents of stored HTML file created by a HTTP POST

Other then how the GET REQUEST is called and processed, all other elements of the .HTML document remain the same for both environments with support for Dynamic HTML and JavaScript.

Discover IP Address

If your router doesn't disclose the IP table, there are at least four ways you can discover the SeaSmart.net IP; two PC-dependent, one Mac-dependent, and one platform independent.

Mac-dependent: download IPNetMonitorX and use it to ping all devices on your subnet. It won't find SeaSmart.net specifically, it'll just find all active devices. Do it once with the SeaSmart.net unplugged, and note all addresses active. Do it again with the SeaSmart.net plugged in, and note the new address in your list. That's the SeaSmart.net. (note: IPNetMonitorX is not free, but it is very handy software if you administer a system and work on a Mac)

PC-dependent: download the Lantronix DeviceInstaller software from SeaSmart Web Site on your PC. It's free and it's designed to sniff out and configure SeaSmart.net on a network. It'll find your SeaSmart.net and tell you its IP and let you configure it.

Independent: Use the IP query app below. It will send out broadcast UDP packets, querying every device on the subnet. Any that are SeaSmart.net will reply, and you'll have their IP addresses. To use it, you'll need Processing and the UDP library from Hypermedia.

IP Discovery Program Example

```
import processing.net.*;

/* SeaSmart.net UDP Device Query
 Sends out a UDP broadcast packet to query a subnet for SeaSmart.net
 serial-to-ethernet devices.
 SeaSmart.net devices are programmed to respond to UDP messages received on
 port 30718.  If a SeaSmart.net device receives the string 0x00 0x00 0x00 0xF6,
 it respond with a UDP packet containing the status message on port 30718.
 This program uses the Hypermedia UDP library available at
http://hypermedia.loeil.org/processing/ */

// import UDP library
import hypermedia.net.*;
UDP udp;                  // define the UDP object
int queryPort = 30718;    // the port number for the device query
String broadcastIpAddress = "128.122.151.255";  // fill in IP address here

void setup() {
  // create a new connection to listen for
  // UDP datagrams on query port
  udp = new UDP(this, queryPort);
// listen for incoming packets:
    udp.listen( true );
}
```

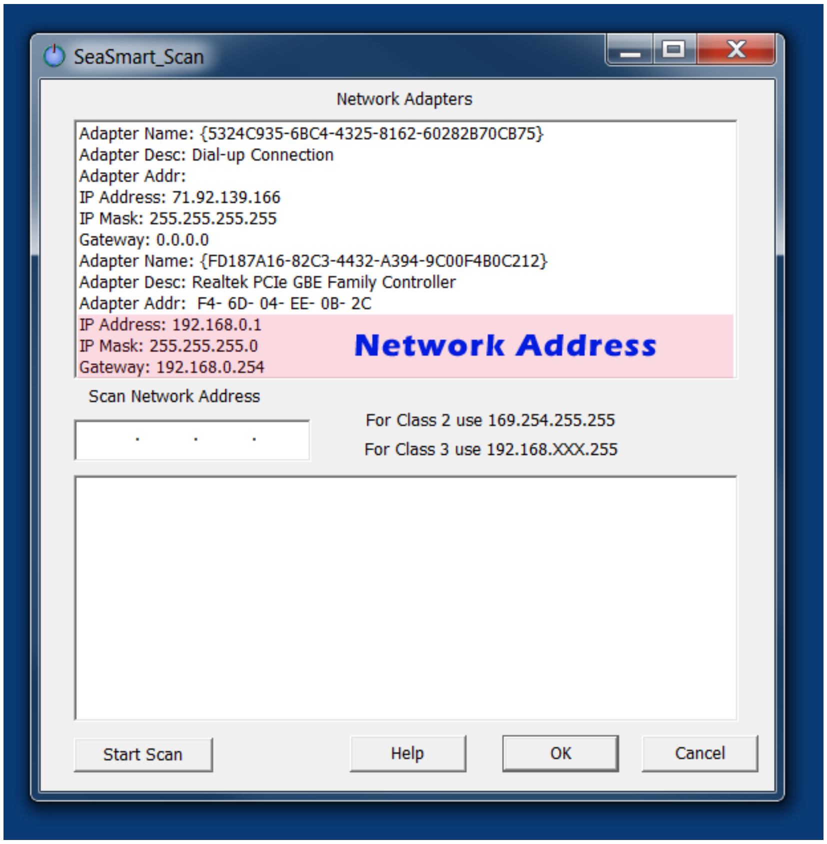
```
//process events
void draw() {
  // twiddle your thumbs.  Everything is event generated.
}
/*
 send the query message when any key is pressed:
 */
void keyPressed() {
  byte[] queryMsg = new byte[4];
  queryMsg[0] = 0x00;
  queryMsg[1] = 0x00;
  queryMsg[2] = 0x00;
  queryMsg[3] = (byte)0xF6;

  // send the message
  udp.send( queryMsg, broadcastIpAddress, queryPort );
}

/*  listen for responses */
void receive( byte[] data, String ip, int port ) {

  String inString = new String(data);    // data converted to a string
  int[] intData = int(data);             // data converted to ints
  int i = 0;                             // counter
  // print the result:
  println( "received "+inString+"  from "+ip+" on port "+port );

  // parse the payload for the appropriate data:
  print("Opcode: ");
  println(intData[3]);

  // if the fourth byte is <F7>, we got a status reply:
  if (intData[3] == 0xF7) {
    // firmware data is bytes 4 to 20:
    print("Firmware data: ");
    for (i=4; i < 20; i++) {
      print(" " + Integer.toHexString(intData[i]));
    }
    // MAC address is bytes 24 to 30 (the end):
    print("\nMAC Addr: ");
    for (i=24; i < intData.length; i++) {
      print(" " + Integer.toHexString(intData[i]));
    }
    print("\n\n");
  }
}
```

SeaSmart Scan Utility

The SeaSmart Scan Utility can be used to discover devices on the local network and report the IP address. The Windows compatible application uses the UDP Broadcast on selected local networks to listen for SeaSmart devices that respond with network configuration information.

The Visual Studio 2010 Source Code is available for download at www.seasmart.net.

To discover SeaSmart devices, first choose the network to scan. PC/Laptops often have multiple network adapters (Ethernet, WiFi, Dial-up Modems) each with their own unique Broadcast Network ID. The SeaSmart Scan Utility will initialize with list of all active networks.



```
Adapter Name: {5324C935-6BC4-4325-8162-60282B70CB75}
Adapter Desc: Dial-up Connection
Adapter Addr:
IP Address: 71.92.139.166
IP Mask: 255.255.255.255
Gateway: 0.0.0.0
Adapter Name: {FD187A16-82C3-4432-A394-9C00F4B0C212}
Adapter Desc: Realtek PCIe GBE Family Controller
Adapter Addr:  F4- 6D- 04- EE- 0B- 2C
IP Address: 192.168.0.1
IP Mask: 255.255.255.0
Gateway: 192.168.0.254
```

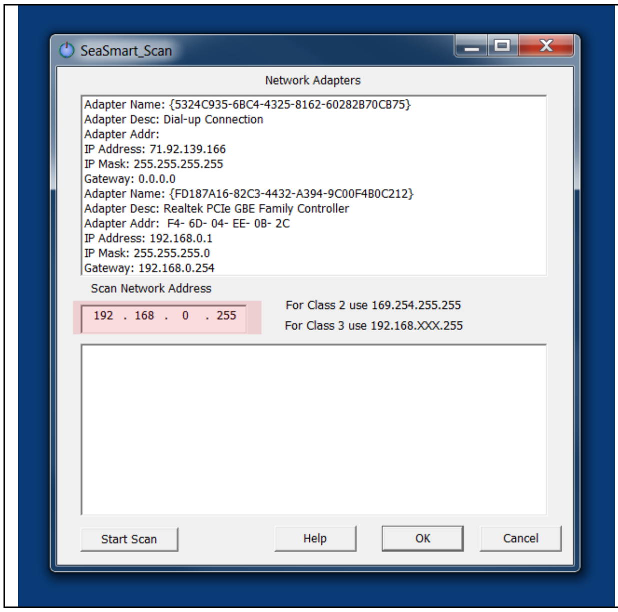
For Class 2 use 169.254.255.255

For Class 3 use 192.168.XXX.255

Once a network adapter is selected, enter the broadcast address to start a scan. The Broadcast Address is always the last address in the subnet. Most local Private networks use a Class C address which means the first 3 groups (octets) of the Network IP need to match the adapter with 255 for the fourth group. For example, if the network adapter is using 192.168.0.1 then the Broadcast would be 192.168.0.255.

WiFi adapters often use a Class B address in which the first two groups (octets) make up the Network ID and the last two must be set to 255.255. For example, if the WiFi adapter is using 169.254.22.35, the Broadcast Address would be 169.254.255.255.

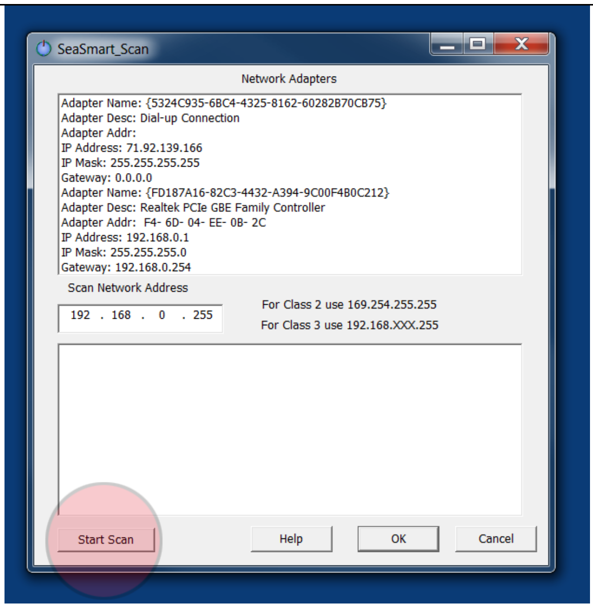
Enter the appropriate Network Broadcast address in the Scan Network Field and select the Start Scan Button



```
Adapter Name: {5324C935-6BC4-4325-8162-60282B70CB75}
Adapter Desc: Dial-up Connection
Adapter Addr:
IP Address: 71.92.139.166
IP Mask: 255.255.255.255
Gateway: 0.0.0.0
Adapter Name: {FD187A16-82C3-4432-A394-9C00F4B0C212}
Adapter Desc: Realtek PCIe GBE Family Controller
Adapter Addr:  F4- 6D- 04- EE- 0B- 2C
IP Address: 192.168.0.1
IP Mask: 255.255.255.0
Gateway: 192.168.0.254
```

192 . 168 . 0 . 255

For Class 2 use 169.254.255.255
For Class 3 use 192.168.XXX.255

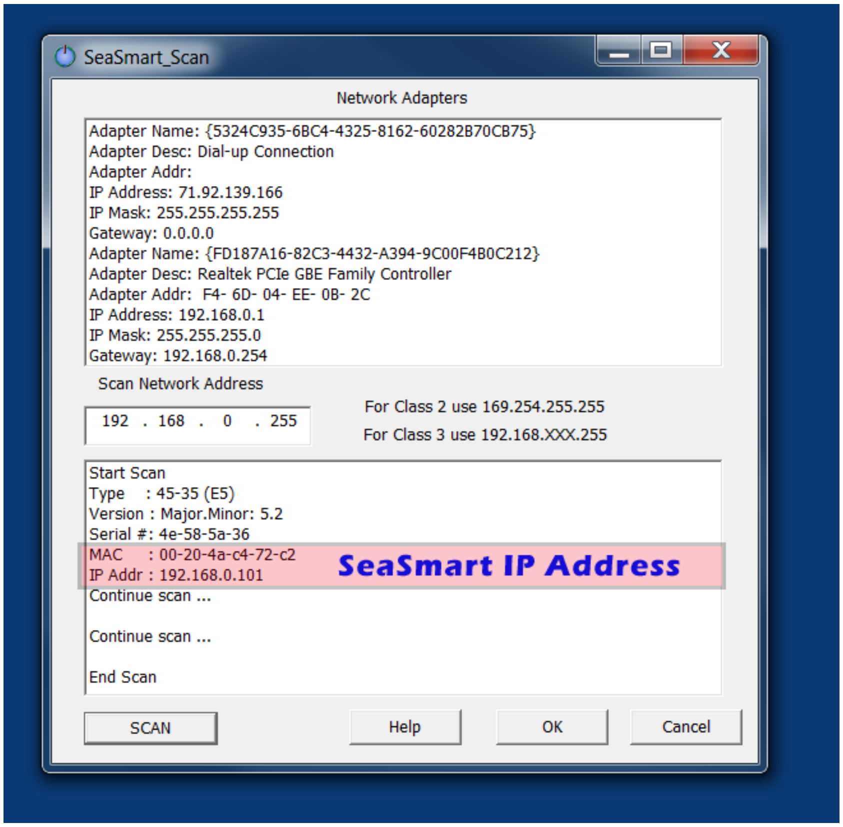


```
Adapter Name: {5324C935-6BC4-4325-8162-60282B70CB75}
Adapter Desc: Dial-up Connection
Adapter Addr:
IP Address: 71.92.139.166
IP Mask: 255.255.255.255
Gateway: 0.0.0.0
Adapter Name: {FD187A16-82C3-4432-A394-9C00F4B0C212}
Adapter Desc: Realtek PCIe GBE Family Controller
Adapter Addr:  F4- 6D- 04- EE- 0B- 2C
IP Address: 192.168.0.1
IP Mask: 255.255.255.0
Gateway: 192.168.0.254
```

192 . 168 . 0 . 255

For Class 2 use 169.254.255.255
For Class 3 use 192.168.XXX.255

As the scan discovers SeaSmart devices, it will display Network Configuration information including IP address and MAC address.



```
Adapter Name: {5324C935-6BC4-4325-8162-60282B70CB75}
Adapter Desc: Dial-up Connection
Adapter Addr:
IP Address: 71.92.139.166
IP Mask: 255.255.255.255
Gateway: 0.0.0.0
Adapter Name: {FD187A16-82C3-4432-A394-9C00F4B0C212}
Adapter Desc: Realtek PCIe GBE Family Controller
Adapter Addr:  F4- 6D- 04- EE- 0B- 2C
IP Address: 192.168.0.1
IP Mask: 255.255.255.0
Gateway: 192.168.0.254
```

192 . 168 . 0 . 255

For Class 2 use 169.254.255.255
For Class 3 use 192.168.XXX.255

```
Start Scan
Type    : 45-35 (E5)
Version : Major.Minor: 5.2
Serial #: 4e-58-5a-36
MAC     : 00-20-4a-c4-72-c2
IP Addr : 192.168.0.101
Continue scan ...

Continue scan ...

End Scan
```

One Year Warranty

"We", "our", or "us" refers to Chetco Digital Instruments, the manufacturer of this product. "You" or "your" refers to the first person who purchases this product as a consumer item for personal, family, or household use.

We warrant this product against defects or malfunctions in materials and workmanship, and against failure to conform to this product's written specifications, all for one year (1) from the date of original purchase by you. WE MAKE NO OTHER EXPRESS WARRANTYOR REPRESENTATION OF ANY KIND WHATSOEVER CONCERNING THIS PRODUCT. Your remedies under this warranty will be available so long as you can show in a reasonable manner that any defect or malfunction in materials or workmanship, or any nonconformity with the product's written specifications, occurred within one year from the date of your original purchase, which must be substantiated by a dated sales receipt or sales slip. Any such defect, malfunction, or non-conformity which occurs within one year from your original purchase date will either be repaired without charge or be replaced with a new product identical or reasonably equivalent to this product, at our option, within a reasonable time after our receipt of the product. If such defect, malfunction, or non-conformity remains after a reasonable number of attempts to repair by us, you may elect to obtain without charge a replacement of the product or a refund for the product. THIS REPAIR, REPLACEMENT, OR REFUND (AS JUST DESCRIBED) IS THE EXCLUSIVE REMEDY AVAILABLE TO YOU AGAINST US FOR ANY DEFECT, MALFUNCTION, OR NON-CONFORMITY CONCERNING THE PRODUCT OR FOR ANY LOSS OR DAMAGE RESULTING FROM ANY OTHER CAUSE WHATSOEVER. WE WILL NOT UNDER ANY CIRCUMSTANCES BE LIABLE TO ANYONE FOR ANY SPECIAL, CONSEQUENTIAL, INCIDENTAL, OR OTHER INDIRECT DAMAGE OF ANY KIND.

Some states do not allow the exclusion or limitation of incidental or consequential damages, so the above limitations or exclusions may not apply to you.

This warranty does NOT apply in the following circumstances: (1) when the product has been serviced or repaired by anyone other than us, (2) when the product has been connected, installed, combined, altered, adjusted, or handled in a manner other than according to the instructions furnished with the product, (3) when any serial number has been effaced, altered, or removed, or (4) when any defect, problem, loss, or damage has resulted from any accident, misuse, negligence, or carelessness, or from any failure to provide reasonable and necessary maintenance in accordance with the instructions of the owner's manual for the product.

We reserve the right to make changes or improvements in our products from time to time without incurring the obligation to install such improvements or changes on equipment or items previously manufactured.

This warranty gives you specific legal rights and you may also have other rights which may vary from state to state.

REMINDER: You must retain the sales slip or sales receipt proving the date of your original purchase in case warranty service is ever required.

Chetco Digital Instruments, INC.
14377 Highway 101 South Unit C
Harbor, OREGON 97415
541-661-2051

VDASH SOFTWARE LICENSE AGREEMENT

THIS IS A LEGAL AGREEMENT BETWEEN THE END-USER WHOFIRST PURCHASES THIS PRODUCT AS A CONSUMER ITEM FORPERSONAL, FAMILY, OR HOUSEHOLD USE (“YOU”) AND CHETCO DIGITAL INSTRUMENTS, INC., THE MANUFACTURER OF THIS PRODUCT. (“WE”, “OUR”, OR “US”). USING THE PRODUCT ACCOMPANIED BY THIS LICENSE AGREEMENT CONSTITUTES ACCEPTANCE OF THESE TERMS AND CONDITIONS.

1. This License Agreement applies to the microcode and one or more lookup tables that your product may contain. We refer to these singly as a “SOFTWARE”.

2. The SOFTWARE that your product may contain are licensed, not sold. We grant to you the nonexclusive, non-assignable right to use these SOFTWARE for monitoring sensor/sender data, but only as long as you comply with the terms and conditions of this License Agreement. We reserve the right to terminate this license if you violate any aspect of this License Agreement.

3. The SOFTWARE housed in your product are protected by the copyright notices appearing on the product or its screen(s). You may NOT modify, adapt, translate, reverse engineer, decompile, disassemble, rent, lease, or resell any SOFTWARE, and you may NOT create derivative works based upon any SOFTWARE or its contents.. Any unauthorized reproduction, use, or transfer of a SOFTWARE may be a crime and may subject you to damages and attorney fees.

4. This License Agreement will terminate immediately without prior notice from us if you fail to comply with or violate any of the provisions of this Agreement. Upon termination, you will promptly return all products containing one or more SOFTWARE to us.

5. Prices and programs are subject to change without notice.

6. This License Agreement shall be governed by the laws of the State of Oregon and comprises the complete and exclusive understanding between you and us concerning the above subject matter.

How to Obtain Service

We back your investment in quality products with quick, expert service and genuine replacement parts. If you're in the United States and you have questions, please contact the Factory Customer Service Department using our number listed below. You must send the unit to the factory for warranty service or repair. Please call the factory before sending the unit. You will be asked for your unit's serial number (shown above). Use the following number:

541-469-4783

U.S.A.only. Monday through Friday, except holidays.

Your unit is covered by a full one-year warranty. (See inside for complete warranty details.) If your unit fails and the failure is not covered by the original warranty, Chetco Digital Instruments has a flat-rate repair policy that covers your unit and accessories packed with the unit at the factory. There is a 180-day warranty on all non-warranty repairs from the factory, which is similar to the original warranty, but is for 180 days rather than one year. For further details, please call us at the above number.

Remember, non-warranty repairs are subject to Chetco Digital Instruments published flat rate charges and 180-day warranty.

CHETCO DIGITAL INSTRUMENTS, INC

BOX 5359

Brookings, OR 97415

541-469-4783

http://www.chetcodigital.com