Contents

1 Overview

This document details the GoFree Tier 1 networking interface provided by Navico Multifunctional Displays. The GoFree Tier 1 networking interface is designed to allow mobile devices (tablets, smartphones and other wi-fi enabled devices) to display and process selected navigation, instrument and engine data (future) from the Navico system.

In order to access GoFree data, you need a Navico marine network equipped with at least one GoFree enabled MFD and a GoFree wireless access point.

2 Service Discovery

2.1 UDP Multicast

Mobile devices wishing to connect can listen to multicast address 239.2.1.1, port 2052 to discover the IP address and port of MFDs that support the data bridging function.

Any MFD will send the following JSON string to this multicast address:

{ “Name” : “*Name*”, “IP”: “*N.N.N.N*”, “Model” : “*Model*”, “Services” : [{ “Service” : “*Service*” , “Version”: *N* , “Port”: *N* }] }

Where:

<*Name*> The friendly name of the MFD. If this is changed on the MFD the multicast will change to match.

<*IP*> IP Address of the MFD. This will be obtained from a DHCP server if one is available, otherwise it will be a Zeroconfig address. Note that if a DHCP server is added to the network after power on the IP address announced will switch from Zeroconfig to DHCP.

<*Model*> Model name of the MFD (e.g. "NSS-12").

<*Service*> Describes the type of data services that the MFD supports. For MFDs that support the webserver, this will be “nmea-0183”.

<*Version*> Version of the data protocol.

<*Port*> Port number used to transmit the data (e.g. 80).

MFDs will send this message at a rate of 1Hz. If this message is not received for a period of 2 second or longer, clients should consider the MFD to be unavailable, and attempt to connect to a different MFD for data.

A code example is included in the appendices.

2.2 Bonjour

The server will be announced via the Bonjour protocol as “_nmea-0183._tcp”. This is automatically supported in OS X and iOS, and can be added to Windows by installing Bonjour for Windows:

http://support.apple.com/kb/DL999

3 Security

No security is required to receive the data.

4 Data Transmission

In order to receive Tier 1 data, clients can create a TCP connection to any of the MFDs transmitting to the multicast address.

If different MFDs send different version strings, then the client should connect to the MFD with the latest (highest version number) MFD.

4.1 Message Format

Messages are formatted using NMEA0183 version 4.00. Readers are referred to that specification for a detailed explanation of the protocol.

4.2 TAG Blocks

TAG blocks, as defined by NMEA0183 version 4.00, are not currently used by GoFree. However, devices that read GoFree data should be robust and tolerant of TAG blocks, since they may be used in future implementations.

4.3 Checksum

NMEA0183 checksums are transmitted by the gateway as part of each GoFree sentence.

4.4 Two-way Communication

Two way communication is not currently supported. Data transmitted by mobile devices to the MFDs will be ignored. Mobile devices must not transmit data to the GoFree multicast address.

4.5 Handling Multiple Sources Of The Same Data

Clearly it is possible for a network to be configured where multiple sources are available for the same data. For example, a network may have two or more GPS antennas. The GoFree gateway will only transmit one source of data for each data type.

For Simrad products, this will be the data source that has been selected by the system for the “Simrad group”. For the first release of GoFree, it will not be possible to view data from sources other than the “Simrad group” sources. However, it is possible to view and change which devices are selected in the “Simrad group” from the MFD. The same applies to the B&G Zeus, data selected in the “default” group will be transmitted.

For Lowrance products, the transmitted data will be from the “global” data source for each data type. As with Simrad, it will not be possible to view data from sources other than the “global” source for first release. However, it is possible to view and change the “global” source from the MFD.

4.6 Automatic Failover

Automatic failover is supported in systems where multiple MFDs are present. Clients should monitor the multicast transmissions to ensure the MFD they are in communication with is still active, and if that MFD stops transmitting for a period of 2 seconds or longer:

- Close the TCP socket to the unavailable MFD.
- Choose another MFD from the devices that are transmitting to the multicast address.
- Open a TCP socket to that MFD and continue receiving data.

4.7 Supported Sentences

Standard Sentences

Identifier	**Data**	Output rate
GGA	Global Positioning System Fix Data	5Hz
GLL	Geographic Position – Latitude/Longitude	5Hz
GSA & GSV	GNSS DOP and Active Satellites/Satellites in View	1Hz
VTG	Course Over Ground and Ground Speed	1Hz
ZDA	Time and Date	1Hz
AAM	Waypoint Arrival Alarm	1Hz
APB	Heading/Track Controller (Autopilot) Sentence “B”	1Hz
BOD	Bearing – Origin to Destination	1Hz
BWC	Bearing and Distance to Waypoint – Great Circle	1Hz
BWR	Bearing and Distance to Waypoint – Rhumb Line	1Hz
RMC	Recommended Minimum Navigation Information	1Hz
RMB	Recommended Minimum GNSS Data	1Hz
XTE	Cross Track Error	1Hz
DBT	Depth Below Transducer	1Hz
DPT	Depth	1Hz
MTW	Water Temperature	1Hz
VLW	Dual Ground/Water Distance	1Hz
VHW	Water Speed and Heading	1Hz
HDG	Heading, Deviation and Variation	10Hz
MWV	Wind Speed and Angle (Relative)	1Hz
MWV	Wind Speed and Angle (True)	1Hz
MWD	Wind Direction and Speed	1Hz

VPW	VMG to Wind	1Hz
TLL	Target Latitude and Longitude	1Hz
TTM	Tracked Target Message	1Hz
VDM	UAIS VHF Data-link Message	1Hz
VDO	UAIS VHF Data-link Own-vessel report	1Hz

Proprietary Sentences

XDR	Heel, Trim, Barometric Pressure	1Hz

$IIXDR,A,2.5,D,HEEL,A,-0.2,D,TRIM,P,1.016,B,BARO

$IIXDR,ANGULAR DISPLACEMENT,2.5,DEGREES,HEEL,ANGULAR DISPLACEMENT,-0.2,DEGREES,TRIM,PRESSURE,1.016,BAR,BARO

4.8 Frequently Asked Questions

4.8.1 What happens to NMEA0183 data that is not plugged in to the "Gateway" MFD?

Most types of NMEA0183 data are bridged on to the NMEA2000 network and will therefore be available to the gateway MFD via NMEA2000. If the NMEA0183 device in question is selected as the "Simrad group" or "Global" source, then its data will be bridged by the gateway and will be available to mobile devices.

4.8.2 Why not just implement IEC61162-450?

IEC61162-450 is the IEC's recently released Ethernet networking standard. Like the GoFree data networking standard, it is fundamentally based on transmitting NMEA0183 formatted sentences over UDP multicast. However, it has some key differences:

- IEC61162-450 defines 16 IP address/port pairs. The address/port pair that is used depends on the talker ID of the sentence. This additional complexity is not present in GoFree, only one address/port pair is used. We made this decision for simplicity of implementation. The chosen address/port for GoFree does not conflict with IEC61162-450.
- IEC61162-450 specifies the use of "TAG blocks" to carry additional data for each sentence. This functionality is not required in the initial release of GoFree. However, GoFree listeners must robustly parse and ignore sentences with TAG blocks so we can implement this functionality in the future.
- IEC61162-450 specifies manual allocation of IP addresses for devices. This is not compatible with Navico marine networks.
- IEC61162-450 is based on multicast UDP for message transmission. For transmission of data to a small number of clients over wireless networks, unicast is preferred. TCP is convenient since there are a number of existing applications for mobile devices that can process TCP streams of NMEA0183 data.

- More generally speaking, IEC61162-450 is a standard for Ethernet networking, whereas GoFree is a wireless networking standard.

Appendix A – Service Discovery

Sample C# code for service discovery. This uses the open source plugin Json.NET (http://www.codeplex.com/json/) for simplicity.

```
using System;
using System.Linq;
using System.Net;
using System.Net.Sockets;
using System.Text;
using System.Threading;
using Newtonsoft.Json;
using System.Diagnostics;

namespace NavicoDiscovery
{
    class Program
    {
        public class MFDService
        {
            public MFDService(string service, uint version, uint port)
            {
                Service = service;
                Version = version;
                Port = port;
            }

            public string Service;
            public uint Version;
            public uint Port;
        }

        public class MFD
        {
            public string Name;
            public string IP;
            public string Model;
            public MFDService[] Services;
        }

        static readonly object _locker = new object();

        private static void ReceiveOldMessage()
        {
            // Navico NSS 2.0: Multicast 239.2.1.1, port 2050
            UdpClient client = new UdpClient();

            IPEndPoint localEp = new IPEndPoint(IPAddress.Any, 2050);
            client.Client.Bind(localEp);

            IPAddress multicastaddress = IPAddress.Parse("239.2.1.1");
            client.JoinMulticastGroup(multicastaddress);

            string output;
            while (true)
            {
```

```
                Byte[] data = client.Receive(ref localEp);
                if (data.Length > 0)
                {
                    string strData = Encoding.ASCII.GetString(data);
                    char[] lineEnd = { '\r', '\n' };
                    strData = strData.TrimEnd(lineEnd);
                    string[] strParams = strData.Split(',');
                    output = string.Format("Service: {0}\tVersion: {1}\tIP: {2}\tPort: {3}",
strParams[0], strParams[1], strParams[2], strParams[3]);
                    lock (_locker)
                    {
                        Console.WriteLine(output);
                        Debug.Print(output + string.Format("\r\n"));
                    }
                }
            }
        }

        private static void ReceiveNewMessage()
        {
            // Navico NSS 2.5: Multicast 239.2.1.1, port 2052
            UdpClient client = new UdpClient();

            IPEndPoint localEp = new IPEndPoint(IPAddress.Any, 2052);
            client.Client.Bind(localEp);

            IPAddress multicastaddress = IPAddress.Parse("239.2.1.1");
            client.JoinMulticastGroup(multicastaddress);

            string output;
            while (true)
            {
                Byte[] data = client.Receive(ref localEp);
                if (data.Length > 0)
                {
                    string strData = Encoding.ASCII.GetString(data);
                    MFD deserializedMFD = JsonConvert.DeserializeObject<MFD>(strData);

                    output = string.Format("MFD: {0}\tModel: {1}\tIP: {2}", deserializedMFD.Name,
deserializedMFD.Model, deserializedMFD.IP);
                    lock (_locker)
                    {
                        Console.WriteLine(output);
                        Debug.Print(output);
                        for (uint service = 0; service < deserializedMFD.Services.Count(); service++)
                        {
                            output = string.Format("\tService {0}: {1}\tVersion: {2}\tPort: {3}",
service, deserializedMFD.Services[service].Service, deserializedMFD.Services[service].Version,
deserializedMFD.Services[service].Port);
                            Console.WriteLine(output);
                            Debug.Print(output);
                        }
                    }
                }
            }
        }
```

```
        static void Main(string[] args)
        {
            Thread oldThread;
            oldThread = new Thread(new ThreadStart(ReceiveOldMessage));
            oldThread.IsBackground = true;
            oldThread.Start();
            Thread newThread;
            newThread = new Thread(new ThreadStart(ReceiveNewMessage));
            newThread.IsBackground = true;
            newThread.Start();

            while (true) { }
        }
    }
}
```