Australian Government
Department of Defence
Defence Science and
Technology Organisation

Extending the Wireshark Network Protocol Analyser to Decode Link 16 Tactical Data Link Messages

William Robertson and Peter Ross

Aerospace Division
Defence Science and Technology Organisation

DSTO-TN-1257

ABSTRACT

This technical note describes the development of a tactical data link message dissector for the Wireshark network protocol analyser. Link 16 is a United States and North Atlantic Treaty Organization standard for secure real-time exchange of tactical information between warfighting units. Concurrent with military adoption of Link 16 equipment, training simulators are being fitted with simulated tactical data links. The extensions made to Wireshark provide simulation engineers with a tool to troubleshoot Link 16 simulations.

Published by

Aerospace Division
DSTO Defence Science and Technology Organisation
506 Lorimer St
Fishermans Bend, Victoria 3207 Australia

Telephone: 1300 333 362
Fax: (03) 9626 7999

AR-015-847
January 2014

APPROVED FOR PUBLIC RELEASE

Extending the Wireshark Network Protocol Analyser to Decode Link 16 Tactical Data Link Messages

Executive Summary

Tactical Data Links (TDLs) enable the military to exchange tactical information in a precise, efficient and timely manner. They complement, and in many instances substitute, traditional voice-based communication bearers such as VHF radio. Wireshark is an open-source network protocol analyser that provides a comprehensive filtering and query system, utilities for performing statistical analysis, and *dissector* modules to decode and analyse protocol content.

This technical note describes modifications to Wireshark enabling it to analyse Link 16 tactical data link messages. A dissector module was written to decode J-series messages communicated using the Simulation Interoperability Standards Organization's *Standard for Link 16 Simulations*. This work was completed as a summer vacation student project over a 10 week period, using sources from the open literature on Link 16. It has been used to develop and test distributed mission training simulators.

This page is intentionally blank

Contents

This page is intentionally blank

Acronyms

bps	Bits per second
DIS	Distributed Interactive Simulation
DTS	Data Terminal Set
GCCS	Global Command and Control System
HLA	High Level Architecture
JREAP	Joint Range Extension Application Protocol
JTIDS	Joint Tactical Information Distribution System
LSB	Least Significant Bit
MSB	Most Significant Bit
MTC	Multi-TADIL Capability
NATO	North Atlantic Treaty Organization
PDU	Protocol Data Unit
PPLI	Precise Participant Location and Identification
QPSK	Quadrature Phased-Shift Keying
RPR-FOM	Real-time Platform Reference Federation Object Model
SIMPLE	Standard Interface for Multiple Platform Link Evaluation
STANAG	Standardisation Agreement (NATO)
TADIL	Tactical Digital Information Link
TCP	Transmission Control Protocol
TDL	Tactical Data Link
TDMA	Time Division Multiple Access
UDP	User Datagram Protocol

This page is intentionally blank

1. Introduction

Tactical Data Links (TDLs) enable the military to exchange tactical information in a precise, efficient and timely manner. They complement, and in many instances substitute, traditional voice-based communication bearers such as VHF radio. Major military platforms operated by the Australian Defence Force are either equipped, or are being equipped with TDL systems. Concurrent with real-world adoption, training simulators are being fitted with simulated tactical data links.

This technical note describes the development of a Link 16 message dissector for the Wireshark network protocol analyser. Wireshark is a freely-available open-source protocol analyser that is an industry standard tool for network protocol troubleshooting. The dissector supports analysis of J-series messages adhering to the Simulation Interoperability Standards Organization's *Standard for Link 16 Simulations*. While existing Link 16 test tools are available, they are more suited to non-simulated data link operations, where quality assurance and safety are paramount. The cost and thus availability of these dedicated test tools make them less desirable for use with simulators.

Development of the dissector was completed as a summer vacation student project over a 10 week period during 2009–2010, using sources from the open literature on Link 16. No export controlled technical data was used in the production of this technical note.

2. Link 16

Link 16 is a United States and North Atlantic Treaty Organization (NATO) tactical data link standard that defines the message format, methods of radio transmission, and network management procedures[1]. It is the current data link system operated by the United States, NATO members and allied countries. Development commenced in 1975 [1], with systems fielded by the United States Air Force and Navy in the early 1990s [2].

Link 16 may also be referred to as the Joint Tactical Information Distribution System (JTIDS), which is the name assigned to the original project and first-generation terminal, or more simply Tactical Digital Information Link J (TADIL-J).

2.1 J-series Message Format

Tactical data link messages are the units of information exchanged between the participants of a tactical data link network. Link 16 supports the exchange of fixed formatted messages, more commonly referred to as J-series messages, to convey information. There are over 70 messages defined [3].

[1] Link 16 is defined by several standards including MIL-STD-6016 *Tactical Data Link 16 Message Standard* and STANAG-5516 *Tactical Data Exchange – Link 16;* STANAG 4175 *Technical Characteristics of the Multifunctional Information Distribution System;* and US CJCSM 6120.01 *Joint Multi-TADIL Operating Procedures* and NATO-ADATP-16 *Standard Operating Procedures for NATO Link 16*. These standards are not published in the open literature and were not used in the production of this technical note.

Each J-series message comprises an initial word, followed by extension word(s) and optional continuation words. Each word is 75 bits long, containing 70 bits of data and 5 bits parity. An example of the initial word layout is shown in Figure 1.

Messages are identified by a label and sub-label. For example, label 2, sub-label 2, identified in shorthand as J2.2, conveys the Precise Participant Location and Identification (PPLI) of an air platform. This message is published on the network to advise other interested participants of an air platform's location and identifying information. United States Department of Defense MIL-STD-6016 and NATO Standardisation Agreement (STANAG) 5516 define how the information is stored and retrieved from the 75-bit words, and rules governing the issuance and receipt of those words.

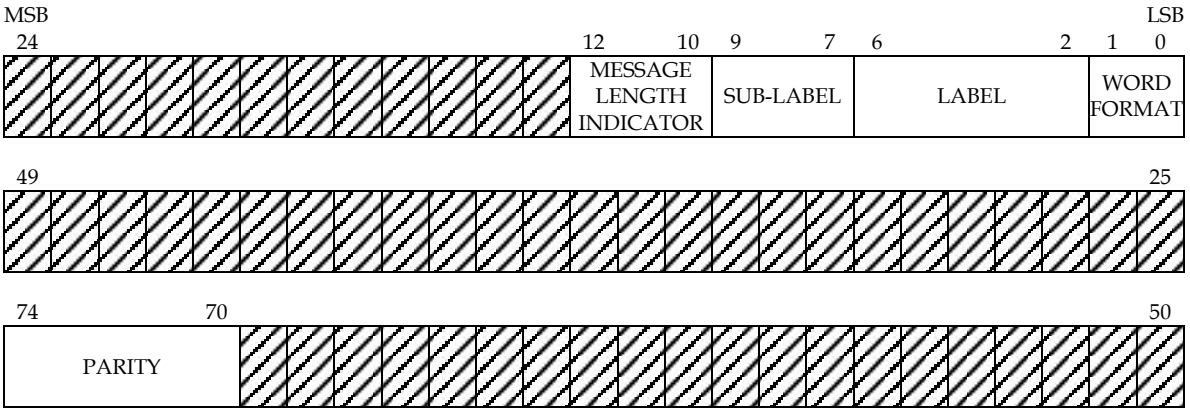

Figure 1: Example J-series Initial Word (reproduced from [4], §5.2.1). The dashed area concerns message specific information.

2.2 Distribution System

When operating within line of sight of one another, the participants of a Link 16 network exchange messages using frequency hopping Time Division Multiple Access (TDMA) transceiver terminals operating in the UHF Lx radar band (960 - 1215 MHz). The distribution system provides resistance against electronic countermeasures whilst maintaining security and integrity of the message content. Network throughput is 28800, 57600 or 115200 bps depending on the required level of electronic countermeasure protection.

Where there is a need to exchange messages beyond line of sight, as is the case with distributed mission training exercises, protocols exist that allow for the data link messages to be exchanged over satellite or fixed-line networks. These protocols are sometimes referred to as *wrappers*, as their primary task is to encapsulate J-series messages in a format suitable for transmission over alternate media [5]. A list of frequently used beyond line of sight distribution protocols is given below:

- *United States Department of Defense MIL-STD-3011 Joint Range Extension Application Protocol (JREAP)*. JREAP is the military standard protocol for exchanging messages beyond line of sight over satellite, serial line or Internet Protocol [6].

- *NATO STANAG 5602 Standard Interface for Multiple Platform Link Evaluation (SIMPLE)* [7]. SIMPLE was developed by the NATO Tactical Data Link Interoperability Test Syndicate to support interoperability testing between development laboratories. The protocol may be operated over serial line or Internet Protocol.

- *Simulation Interoperability Standards Organization SISO-STD-002 Standard for Link 16 Simulations* [8]. Also known as SISO-J, this protocol was developed by the simulation community to address the needs of training and research simulations. SISO-J is an extension of the Distributed Interactive Simulation (DIS) application protocol, and High Level Architecture (HLA) Real-time Platform Reference Federation Object Model (RPR-FOM). It encapsulates J-series messages inside the DIS Transmitter and Signal Protocol Data Units (PDUs) or RPR-FOM equivalents, allowing line of sight, signal propagation and interference to be modelled within the simulation. The standard defines optional rules for modelling the characteristics of the TDMA network.

- *Global Command and Control System (GCCS) Multi TADIL Capability (MTC).* The MTC protocol was developed initially for exchanging J-series messages with GCCS software [9], but has been since adopted by other systems. Whereas the other distribution protocols listed here employ detailed encapsulation headers and issuance and receipt rules, the MTC protocol uses minimalist headers. It may be operated over serial line or Internet Protocol, and is referred to informally as Serial-J or Socket-J.

Of the above protocols, only SIMPLE and SISO-J have been published in the open literature. JREAP protocol data structures are described in [10], but the issuance and receipt rules are omitted.

3. Link 11/11B

Link 16 complements an earlier generation TDL standard known as Link 11. Development of Link 11 commenced in 1954, with sea trials undertaken in the 1960s. The airborne and maritime variant of Link 11 is also known as TADIL-A, and the variant intended for point to point links is known as Link 11B, or TADIL-B [2]. Link 11 is governed by several standards that define the message format, method of signal modulation, and procedures for network management[2]. Although Link 11 is not the focus of our work, appreciating the similarities between it and Link 16 provides for meaningful discussion later in this technical note.

[2] Link 11 is defined by several standards including MIL-STD-188-203-1 *Interoperability and Performance Requirements for TADIL*-A and MIL-STD-188-203-2 *Subsystem Design and Engineering Standards for TADIL-B*; MIL-STD-6011 *Tactical Data Link 11/11B Message Standard* and STANAG-5511 *Tactical Data Exchange – Link 11/Link 11B*; and NATO-ADATP-11 *Standard Operating Procedures for NATO Link 11/Link 11B*. With the exception of MIL-STD-188-203-1 and MIL-STD-188-203-2, these standards are not published in the open literature.

3.1 M-series Message Format

Link 11 M-series messages are transmitted as two 30 bit frames, with each frame containing 24 bits of data, and 6 bits for forward error correction. Messages are identified by a 4-bit label. United States Department of Defense MIL-STD-6011 and NATO STANAG 5511 describe how information is stored and retrieved from the 30 bit frames. An example of the message layout is shown in Figure 2.

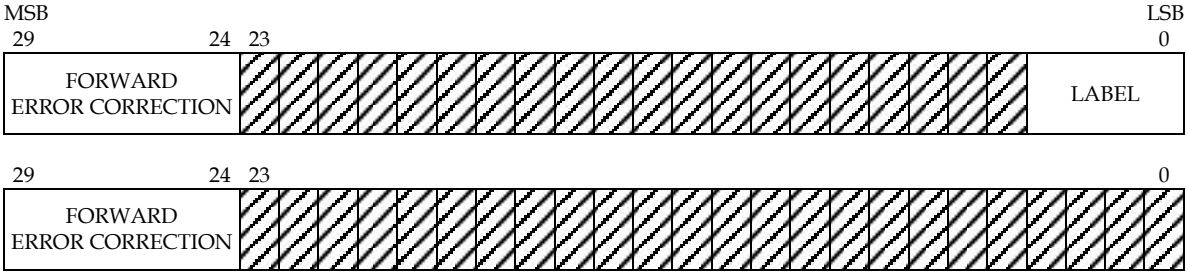

Figure 2: Example M-series message (reproduced from [11], §2.1). The dashed area concerns message specific information.

3.2 Distribution System

Link 11 messages are modulated using Quadrature Phased-Shift Keying (QPSK) and exchanged over half-duplex HF or UHF radio. The modulator/demodulator is known as the Data Terminal Set (DTS). Security is achieved by encrypting the digital message stream prior to modulation. The system supports two data rates: 2250 bps (fast) and 1364 bps (slow) [2].

Link 11B messages are transmitted over point to point radio links, such as microwave or satellite.

Protocols to distribute Link 11 messages beyond radio coverage include:

- *The plain old telephone system*. To achieve point-to-point connectivity, the radio transceiver is replaced with a standard telephone line.

- *TADIL-B over Internet Protocol.* The TADIL-B transmission frame format [12] can be exchanged over serial line or Internet Protocol.

- *NATO STANAG 5602 Standard Interface for Multiple Platform Link Evaluation (SIMPLE).* Refer to the previous description of SIMPLE in section 2.2.

- *SISO-STD-005 Standard for Link 11/11B Simulation* [13]. Currently in draft form, this standard encapsulates M-series messages within the DIS Transmitter and Signal PDUs (or RPR-FOM equivalents). The performance characteristics of the network may be optionally modelled.

4. Wireshark

Wireshark, previously named Ethereal, is an open-source network protocol analyser. It provides a comprehensive filtering and query system, utilities for performing statistical analysis, and *dissector* modules to decode and inspect protocol content. A graphical and text-mode (known as *tshark*) user interface is provided [14]. Wireshark uses the libpcap library [15] (or WinPcap libraries for Microsoft Windows) to capture Ethernet frames directly from a computer's network interface adaptor.

Dissectors are modules that exist within Wireshark to decode specific network protocols and present the information in a human-readable format. A diverse range of dissectors is included with Wireshark, such as Ethernet, Internet Protocol, Transmission Control Protocol (TCP) and User Datagram Protocol (UDP) and over 1000 other protocols. Wireshark includes routines for dissecting DIS PDUs. However at the time of the student project only the most common of PDU types were supported: Entity State, Fire and Detonation PDUs. Since then other developers have expanded the DIS dissector to support many more PDUs from the Distributed Emission Regeneration, Radio Communication and Simulation Management families.

Wireshark was chosen as a foundation for this student project for the following reasons:

- *Industry standard.* Wireshark is an industry standard tool for network protocol troubleshooting. Any effort spent improving Wireshark is likely to benefit its other users.

- *Familiarity.* For distributed simulation exercises, staff at DSTO already use Wireshark to diagnose faults and capture packets in simulation exercises. Where practical the libpcap file format is used to archive exercise recordings.

- *Heritage.* Wireshark has been under development since 1998. Infrastructure to decode byte sequences, display information, and manage configuration files has been previously built and thoroughly tested.

- *Portability.* Wireshark is written in the C89 programming language and runs on Linux, Mac OS X, BSD, Solaris and Microsoft Windows operating systems.

- *Active open-source project.* The software is distributed using the GNU General Public License, and is supported by an active developer and user community. The project features a high volume email reflector (averaging 11, 11 and 7 messages per day over 2010, 2011 and 2012 respectively [16]), library of reference packet capture files, regression testing system and issue tracker.

5. Requirements

Requirements for the student project were defined as follows.

1. Modify the existing DIS dissector to decode Signal PDUs and SISO-J headers and display content of fields, performing enumeration lookups where necessary.
2. Support dissection of J-series messages; display the label and sub-label of each message.
3. Ensure the Link 16 routines are sufficiently modular, allowing them to be reused by other protocol dissectors (such as a hypothetical JREAP or MTC dissector). An experimental SIMPLE protocol dissector, developed some years earlier by DSTO, provides a use case for this requirement.
4. Support filtering of messages such that search queries can be answered. For example '*display all packets containing the J2.2 message*.'
5. Document all material used in the development of the dissector to allow auditing of the source code.
6. Reuse or adapt existing Wireshark routines where possible.
7. Produce a working software patch that can be applied to a known version of Wireshark.

6. Development

The modifications to Wireshark were developed over several weeks in 2010. They were revised in 2013 in order to be compatible with Wireshark v1.11.0 (the current version at time of publication).

6.1 Environment

An appropriate build environment was needed to begin programming of the dissector. As the primary development computer was a Linux system (Ubuntu Linux 9.10 and 12.10), this was achieved by running the included *configure* shell script and the required dependencies via the Linux package manager.

6.2 Implementation

It was necessary to become familiar with the design and internal workings of Wireshark and its existing DIS dissector. There were plenty of development resources available on the Internet; a particularly useful resource was an article in *The Code Project* on writing a new dissector [17]. A review of the existing Wireshark DIS dissector found it to use a custom system for parsing DIS PDUs. Given a description of a DIS PDU or Record, the parsing system would automatically extract the appropriate bytes from the PDU, perform endian conversion, and register fields with Wireshark's filtering and query system. This approach was found to be unique compared to other existing protocol dissectors.

The task of extending Wireshark to support Link 16 was broken into three main activities:

1. Write a new Link 16 dissector module that decodes J-series message words.
2. Modify the existing DIS dissector to process SISO-J Signal PDUs.
3. Modify an experimental SIMPLE dissector to use the new Link 16 dissector.

Logic to detect SISO-J headers was added to the Signal PDU processor of the DIS dissector. When detected, the relevant headers are decoded, and the Link 16 J-series messages are handed off individually to the Link 16 dissector. By separating the Link 16 J-series message dissector from the DIS dissector, the requirement for a modular system was fulfilled.

Distribution protocols were each found to format J-series message words differently for transmission. The SIMPLE protocol stores each word as a 80-bit little endian word, whereas the SISO-J Signal PDU stores them as a series of three interleaved big endian 32-bit words [18]. JREAP and MTC may use different formatting again. It was therefore necessary to make the Link 16 dissector agnostic to the underlying transmission format. This was achieved by *normalising* the format accepted by the Link 16 dissector module, and making the SISO-J and SIMPLE dissectors responsible for converting between the transmission and normalised format (80-bit little endian was chosen as the normalised format).

Because only the initial word of a J-series message contains the label and sublabel identifiers, and subsequent extension and continuation messages rely on that information, it was also necessary to store these values within the context of the dissector. This enabled subsequent words within the message to be identified correctly.

6.3 Testing

The modified Wireshark has been evaluated against the example J2.2 Air PPLI message defined in the SISO-J standard (refer to Appendix A), and data generated by third-party SISO-J capable equipment. Significant observations from the test phase are given below.

Using the built in Wireshark tool *fuzz-test.sh*, more than 2000 iterations of fuzzed capture files were fed into the dissectors to test for its ability to handle corrupted packets. The test was completed with no errors encountered. This was done to gain confidence that the dissector would not crash when subjected to invalid data. As a troubleshooting tool, it must be able to cope with malformed DIS PDUs and J-series messages.

The software was tested on Ubuntu Linux 12.10 (on both i386 and 64 bit PowerPC hardware), Debian 6.06 (SPARC64) and Windows Server 2003 R2 (i386) operating systems – providing coverage of 32 bit, 64 bit, little endian and big endian environments. Two faults were found during this testing; these have since been corrected.

1. Initially Link 16 words were not being decoded correctly. This was due to misinterpretation of the method SISO-J used to store J-series messages.

2. Output from the text-mode *tshark* program did not match the output from the Wireshark graphical user interface – the data in the *information* column was not being updated. Moving the code that handled this to outside the *if(tree){ ... }* section allowed tshark to function properly.

6.4 Development Observations

Through the development of the dissector it became apparent that it is very important to be working with up to date materials. Although the latest published version of the SISO-J standard document was being used, a draft revision of the standard gave useful examples of the byte swapping operations required to decode the Link 16 messages.

Software development began using an older version of the Wireshark source code. Changes were being made to the DIS dissector upstream[3], and halfway through the development process it was necessary to merge these changes (some of which were overlapping with updates that had been made in the local repository).

6.5 Software Patch

In accordance with the Wireshark development guidelines [19], a source code patch against the current version of Wireshark was made. Three patch files were created, one to introduce the Link-16 dissector, and another to add SISO-J header parsing to the existing DIS dissector. A third patch addresses typographic errors found in the existing DIS dissector. Refer to Appendix B for the source code patch.

7. Conclusion and Further Work

The Wireshark network protocol analyser has been extended to support analysis of tactical data link messages. This work was done as a summer vacation student project. The extended version of Wireshark has been evaluated against several sources of J-series messages and demonstrated publically [20]. It has been used to support the development and testing of distributed mission training simulators.

There are several ways in which this work could be furthered. The authors' suggestions are given below.

Develop a more comprehensive J-series message dissector. The dissector presented in this work only decodes the label and sub-label fields. While this is sufficient for superficial analysis of Link 16 networks, there are many fields within the J-series messages (such as track and coordinate numbers) that would make the dissector more useful for troubleshooting faults.

Support dissection of other distribution protocols. Wireshark could be further extended to dissect the JREAP and MTC protocols.

Develop a Link 11 dissector. Using the approach described in this technical note, one could develop a SISO-STD-005 protocol dissector and complementary M-series message dissector.

[3] In this context, upstream refers to changes made to the official Wireshark project.

Visualisation and statistical utilities In addition to analysing packets, Wireshark includes utilities to visualise and gather statistics on particular protocols. There are potentially use cases where visualisation or presentation of the statistics of the link network would assist with fault diagnosis.

8. References

1. Hura, M., et al. (2000), *Interoperability – A Continuing Challenge In Coalition Air Operations*, RAND Document No. MR-1235-AF, ISBN 0-8330-2912-6.

2. Friedman, N., (2006), *The Naval Institute Guide to World Naval Weapon Systems*, Naval Institute Press, ISBN 1557502625.

3. Viasat, Inc., (2012), *Link 16 Network Participant Group and Message Card*, accessed from <*http://www.viasat.com/files/assets/assets/Link16_NPG_Message_Card_100112a.pdf*> on 15 April 2013.

4. Elmasry, G., (2012), *Tactical Wireless Communications and Networks: Design Concepts and Challenges*, Wiley, ISBN 9781119951766.

5. Hill, F., (2003), *Systemic Problems With Data Link Simulation,* Fall Simulation Interoperability Workshop 2003, Paper No. 03F-SIW-002.

6. Boardman, B., (2008), *Introduction to Tactical Data Links in the ADF*, accessed from <*http://www.milcis.com.au/milcis2008pdf/Tue/Brett%20Boardman.pdf*> on 27 January 2010.

7. STANAG-5602 (2010), *Standard Interface for Multiple Platform Link Evaluation (SIMPLE)*, Edition 3, NATO Standardization Agency, accessed from <*https://assist.dla.mil/*> on 30 October 2013.

8. Simulation Interoperability Standards Organization, (2006), *SISO-STD-002 Standard for Link 16 Simulations*, June 2006.

9. Andersen, D.P. and Thomas, K.D., (2001), *Systems Integration Facility: Past, Present, and Future*, Space and Naval Warfare Systems Center Biennial Review 2001, accessed from <*http://www.spawar.navy.mil/sti/publications/pubs/td/3117/index.html*> on 15 April 2013.

10. Teege, G., Eggendorfer, T., Eiseler, V., and Göhner, M., (2007), *Militärische mobile Kommunikationsnetze*, Universität der Bundeswehr München, Institute of Information Systems Report 2007-02, accessed from <*https://dokumente.unibw.de/pub/bscw.cgi/1799259*> on 15 April 2013.

11. Sorroche, J. (2006), *Tactical Digital Information Link-Technical Advice and Lexicon for Enabling Simulation (TADIL-TALES) II: Link 11/11B*, 11th International Command and Control Research and Technology Symposium.

12. United State Department of Defense, (1984), *MIL-STD-188-203-2 Subsystem Design and Engineering Standards for Tactical Digital Information Link (TADIL) B*, 23 March 1984.

13. Simulation Interoperability Standards Organization, (2008), *SISO-STD-005 Standard for Link 11/11B Simulation* (Draft), 8 September 2008.

14. Orebaugh, A., (2007), *Wireshark & Ethereal: Network Protocol Analyzer Toolkit*, Syngress, ISBN 1597490733.

15. Jacobson, V., Leres, C., and McCanne, S., (1994), *libpcap*. Lawrence Berkeley Laboratory, Berkeley, CA, initial public release June 1994, accessed from <*http://www.tcpdump.org/*> on 18 February 2010.

16. GMANE, (2013), *Email archive statistics for the wireshark-dev mailing list*, accessed from <*http://gmane.org/output-rate.php?group=gmane.network.wireshark.devel*> on 15 April 2013.

17. Thompson, K., (2007), *Creating Your Own Custom Wireshark Dissector*, Code Project article, accessed from <*http://www.codeproject.com/KB/IP/custom_dissector.aspx*> on 27 January 2010.

18. Sorroche, J., (2008), *SISO J to SIMPLE Translation Advice and Lexicon for Enabling Simulations (SIMPLE TALES)*, European Simulation Interoperability Workshop 2008, Paper No. 08E-SIW-046.

19. Lamping, U., (2013), *Wireshark Developer's Guide for Wireshark 1.11*, accessed from <*http://www.wireshark.org/docs/wsdg_html_chunked/*> on 16 October 2013.

20. Robertson, W., Ross, P., and Robbie, A., (2010), *Open Source Analyzer for SISO-J Tactical Data Link Simulation*, SimTecT 2010 Conference Proceedings, 31 May—3 June 2010.

Appendix A: Reference Capture File

An exemplar DIS Signal PDU containing J-series messages is described by SISO-STD-002 Annex B. Ethernet, IP and UDP headers were added to this PDU to create a reference capture file compatible with Wireshark. The reference file is shown at Figure B1 and dissected output is shown at Figures B2, B3 and B4.

```
begin-base64 600 siso_std_002_annex_b_example.pcap
1MOyoQIABAAAAAAAAAAAAP//AAABAAAAAAAAAAAAAACCAAAAggAAAAEAXgEB
AQAAAAAAAAgARQAAdAjBAACAEY8MwKgBAeABAQELuAu4AGDavgYBGgQAAAAA
AFgAAAAwAAEAAQABQAMAZAAAAAABwAAAAYA//8AAAAAAAAA/////////8A
AACQCQgAANgL8LoAAP/kLGtLKs10tzQ1BQAFkgByBwAAAAE=
====
```

Figure B1: Input file (uuencoded)

```
% tshark -r siso_std_002_annex_b_example.pcap

  1   0.000000  192.168.1.1 -> 224.1.1.1    Link 16 130 PDUType: Signal, RadioID=1, STN=011,
Link 16 Words: J2.2I J2.2E0 J2.2C1
```

Figure B2: Dissector output using tshark

```
% tshark -r siso_std_002_annex_b_example.pcap –V

Frame 1: 130 bytes on wire (1040 bits), 130 bytes captured (1040 bits)
    Encapsulation type: Ethernet (1)
    Arrival Time: Jan  1, 1970 10:00:00.000000000 EST
    [Time shift for this packet: 0.000000000 seconds]
    Epoch Time: 0.000000000 seconds
    [Time delta from previous captured frame: 0.000000000 seconds]
    [Time delta from previous displayed frame: 0.000000000 seconds]
    [Time since reference or first frame: 0.000000000 seconds]
    Frame Number: 1
    Frame Length: 130 bytes (1040 bits)
    Capture Length: 130 bytes (1040 bits)
    [Frame is marked: False]
    [Frame is ignored: False]
    [Protocols in frame: eth:ip:udp:dis:link16:link16:link16]
Ethernet   II,   Src:   00:00:00_00:00:00   (00:00:00:00:00:00),   Dst:   IPv4mcast_01:01:01
(01:00:5e:01:01:01)
    Destination: IPv4mcast_01:01:01 (01:00:5e:01:01:01)
        Address: IPv4mcast_01:01:01 (01:00:5e:01:01:01)
        .... ..0. .... .... .... .... = LG bit: Globally unique address (factory default)
        .... ...1 .... .... .... .... = IG bit: Group address (multicast/broadcast)
    Source: 00:00:00_00:00:00 (00:00:00:00:00:00)
        Address: 00:00:00_00:00:00 (00:00:00:00:00:00)
        .... ..0. .... .... .... .... = LG bit: Globally unique address (factory default)
        .... ...0 .... .... .... .... = IG bit: Individual address (unicast)
    Type: IP (0x0800)
Internet Protocol Version 4, Src: 192.168.1.1 (192.168.1.1), Dst: 224.1.1.1 (224.1.1.1)
    Version: 4
    Header Length: 20 bytes
    Differentiated Services Field: 0x00 (DSCP 0x00: Default; ECN: 0x00: Not-ECT (Not ECN-
Capable Transport))
        0000 00.. = Differentiated Services Codepoint: Default (0x00)
        .... ..00 = Explicit Congestion Notification: Not-ECT (Not ECN-Capable Transport)
(0x00)
    Total Length: 116
    Identification: 0x08c1 (2241)
    Flags: 0x00
```

```
        0... .... = Reserved bit: Not set
        .0.. .... = Don't fragment: Not set
        ..0. .... = More fragments: Not set
    Fragment offset: 0
    Time to live: 128
    Protocol: UDP (17)
    Header checksum: 0x8f0c [correct]
        [Good: True]
        [Bad: False]
    Source: 192.168.1.1 (192.168.1.1)
    Destination: 224.1.1.1 (224.1.1.1)
User Datagram Protocol, Src Port: remoteware-cl (3000), Dst Port: remoteware-cl (3000)
    Source Port: remoteware-cl (3000)
    Destination Port: remoteware-cl (3000)
    Length: 96
    Checksum: 0xdabe [validation disabled]
        [Good Checksum: False]
        [Bad Checksum: False]
Distributed Interactive Simulation
    Header
        Proto version: IEEE 1278.1A-1998 (6)
        Excercise ID: 1
        PDU type: Signal (26)
        Proto Family: Radio communications (4)
        Timestamp = 00:00 000 relative
        PDU Length: 88
        Explicit Padding (2 bytes)
    Signal PDU
        Entity ID
            Entity ID Site: 48
            Entity ID Application: 1
            Entity ID Entity: 1
        Radio ID: 1
        Encoding Scheme: 0x4003
            01.. .... .... .... = Encoding Class: Raw Binary Data (1)
            ..00 0000 0000 0011 = Encoding Type: 3
        TDL Type: Link 16 Standardized Format (JTIDS/MIDS/TADIL J) (100)
        Sample Rate: 0
        Data Length: 448
        Number of Samples: 0
        Link 16 Network Header
            NPG Number: PPLI and Status (6)
            Network Number = 0
            TSEC CVLL: NO STATEMENT (255)
            MSEC CVLL: NO STATEMENT (255)
            Message Type: JTIDS Header/Messages (0)
            Padding = 0
            Time Slot ID = 0
            Perceived Transmit Time: NO STATEMENT
        Link 16 Message Data: JTIDS Header/Messages
            Time Slot Type: 0
            Relay Transmission Indicator: 0
            Source Track Number: 011
            Secure Data Unit Serial Number: 0
            Link 16 J2.2I Air PPLI
                Word Format: Initial Word (0)
                Label: Precise Participant Location and Identificaton (2)
                Sublabel: 2
                Message Length Indicator: 2
            Link 16 J2.2E0 Air PPLI
                Word Format: Extension Word (2)
            Link 16 J2.2C1 Air PPLI
                Word Format: Continuation Word (1)
                Continuation Word Label: 1
```

Figure B3: Dissector output using tshark (with packet details)

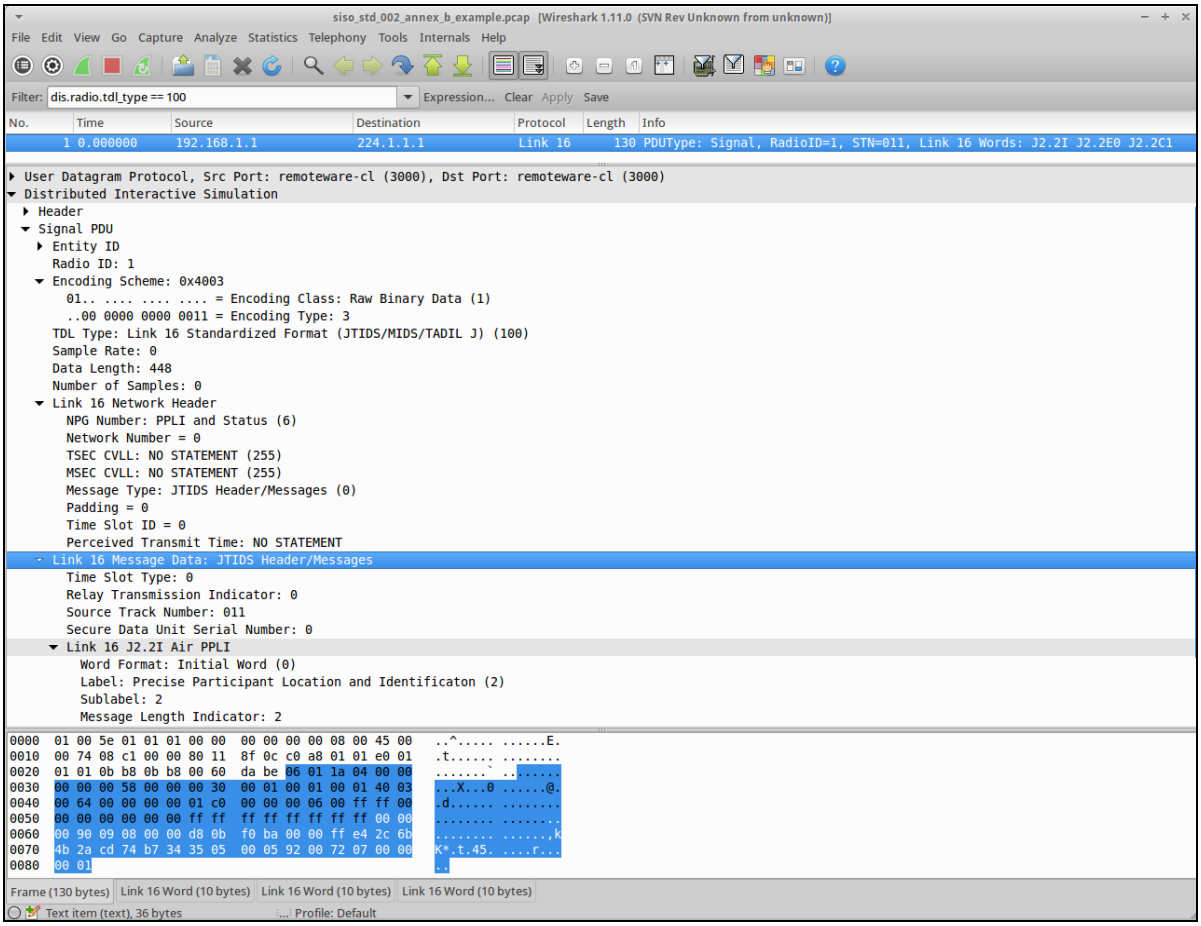

Figure B4: Dissector output using Wireshark graphical user interface

Appendix B: Software Patch

The modifications made against Wireshark v1.11.0 are shown at Figures A1, A2 and A3.

```
From 0f9b0c80580700cc4cf06f91b61cfe8b3b31fa6d Mon Sep 17 00:00:00 2001
From: Peter Ross <peter.ross@dsto.defence.gov.au>
Date: Wed, 16 Oct 2013 10:00:01 +1100
Subject: [PATCH 1/3] packet-link16: Link 16 message dissector (MIL-STD-6016)

---
 epan/CMakeLists.txt             |   1 +
 epan/dissectors/Makefile.common |   1 +
 epan/dissectors/packet-link16.c | 276 ++++++++++++++++++++++++++++++++++++++++
 epan/dissectors/packet-link16.h |  34 +++++
 4 files changed, 312 insertions(+)
 create mode 100644 epan/dissectors/packet-link16.c
 create mode 100644 epan/dissectors/packet-link16.h

diff --git a/epan/CMakeLists.txt b/epan/CMakeLists.txt
index 9baeaa4..c8f54fa 100644
--- a/epan/CMakeLists.txt
+++ b/epan/CMakeLists.txt
@@ -843,6 +843,7 @@ set(DISSECTOR_SRC
        dissectors/packet-ldp.c
        dissectors/packet-ldss.c
        dissectors/packet-lge_monitor.c
+       dissectors/packet-link16.c
        dissectors/packet-linx.c
        dissectors/packet-lisp-data.c
        dissectors/packet-lisp.c
diff --git a/epan/dissectors/Makefile.common b/epan/dissectors/Makefile.common
index ae9831b..bad899e 100644
--- a/epan/dissectors/Makefile.common
+++ b/epan/dissectors/Makefile.common
@@ -772,6 +772,7 @@ DISSECTOR_SRC = \
        packet-ldp.c            \
        packet-ldss.c           \
        packet-lge_monitor.c    \
+       packet-link16.c                 \
        packet-linx.c           \
        packet-lisp-data.c      \
        packet-lisp.c           \
diff --git a/epan/dissectors/packet-link16.c b/epan/dissectors/packet-link16.c
new file mode 100644
index 0000000..db131db
--- /dev/null
+++ b/epan/dissectors/packet-link16.c
@@ -0,0 +1,276 @@
+/* packet-link16.c
+ * Routines for Link 16 message dissection (MIL-STD-6016)
+ * William Robertson <aliask@gmail.com>
+ * Peter Ross <peter.ross@dsto.defence.gov.au>
+ *
+ * $Id$
+ *
+ * This program is free software; you can redistribute it and/or
+ * modify it under the terms of the GNU General Public License
+ * as published by the Free Software Foundation; either version 2
+ * of the License, or (at your option) any later version.
+ *
+ * This program is distributed in the hope that it will be useful,
+ * but WITHOUT ANY WARRANTY; without even the implied warranty of
+ * MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE.  See the
+ * GNU General Public License for more details.
+ *
+ * You should have received a copy of the GNU General Public License
+ * along with this program; if not, write to the Free Software
```

```
+ * Foundation, Inc., 51 Franklin Street, Fifth Floor, Boston, MA 02110-1301 USA
+ */
+
+#include "config.h"
+
+#include <glib.h>
+#include <epan/packet.h>
+#include <epan/value_string.h>
+#include <string.h>
+#include "packet-link16.h"
+
+/* Elmasry, G., (2012), Tactical Wireless Communications and Networks: Design Concepts and
Challenges, Wiley, ISBN 9781119951766. */
+enum {
+    WORDFORMAT_INITIAL = 0,
+    WORDFORMAT_CONTINUATION,
+    WORDFORMAT_EXTENSION,
+};
+
+static const value_string WordFormat_Strings[] = {
+    { WORDFORMAT_INITIAL, "Initial Word" },
+    { WORDFORMAT_CONTINUATION, "Continuation Word" },
+    { WORDFORMAT_EXTENSION, "Extension Word" },
+    { 0, NULL },
+};
+
+/* Viasat, Inc., (2012), Link 16 Network Participant Group and Message Card, accessed from
<http://www.viasat.com/files/assets/assets/Link16_NPG_Message_Card_100112a.pdf> on 15 April
2013. */
+static const value_string Link16_Label_Strings[] = {
+    { 0, "Network Management" },
+    { 1, "Network Management" },
+    { 2, "Precise Participant Location and Identificaton" },
+    { 3, "Surveillance" },
+    { 5, "Anti-submarine Warfare" },
+    { 6, "Intelligence" },
+    { 7, "Information Management" },
+    { 8, "Information Management" },
+    { 9, "Weapons Coordination and Management" },
+    { 10, "Weapons Coordination and Management" },
+    { 11, "Weapons Coordination and Management" },
+    { 12, "Control" },
+    { 13, "Platform and System Status" },
+    { 14, "Electronic Warfare" },
+    { 15, "Threat Warning" },
+    { 16, "Imagery" },
+    { 17, "Weather" },
+    { 28, "National Use" },
+    { 29, "National Use" },
+    { 30, "National Use" },
+    { 31, "Miscellaneous" },
+    { 0, NULL },
+};
+
+/* Viasat, Inc., (2012), Link 16 Network Participant Group and Message Card, accessed from
<http://www.viasat.com/files/assets/assets/Link16_NPG_Message_Card_100112a.pdf> on 15 April
2013. */
+#define MKPAIR(a, b) (((b) << 5) | (a))
+static const value_string Link16_Message_Strings[] = {
+    { MKPAIR(0, 0), "Initial Entry" },
+    { MKPAIR(0, 1), "Test" },
+    { MKPAIR(0, 2), "Network Time Update" },
+    { MKPAIR(0, 3), "Time Slot Assignment" },
+    { MKPAIR(0, 4), "Radio Relay Control" },
+    { MKPAIR(0, 5), "Repromulgation Relay" },
+    { MKPAIR(0, 6), "Communication Control" },
+    { MKPAIR(0, 7), "Time Slot Reallocation" },
+    { MKPAIR(1, 0), "Connectivity Interrogation" },
+    { MKPAIR(1, 1), "Connectivity Status" },
+    { MKPAIR(1, 2), "Route Establishment" },
+    { MKPAIR(1, 3), "Acknowledgment" },
```

```
+    { MKPAIR(1, 4), "Communication Status" },
+    { MKPAIR(1, 5), "Net Control Initialization" },
+    { MKPAIR(1, 6), "Needline Participation Group Assignment" },
+    { MKPAIR(2, 0), "Indirect Interface Unit PPLI" },
+    { MKPAIR(2, 2), "Air PPLI" },
+    { MKPAIR(2, 3), "Surface PPLI" },
+    { MKPAIR(2, 4), "Subsurface PPLI" },
+    { MKPAIR(2, 5), "Land Point PPLI" },
+    { MKPAIR(2, 6), "Land Track PPLI" },
+    { MKPAIR(3, 0), "Reference Point" },
+    { MKPAIR(3, 1), "Emergency Point" },
+    { MKPAIR(3, 2), "Air Track" },
+    { MKPAIR(3, 3), "Surface Track" },
+    { MKPAIR(3, 4), "Subsurface Track" },
+    { MKPAIR(3, 5), "Land Point or Track" },
+    { MKPAIR(3, 6), "Space Track" },
+    { MKPAIR(3, 7), "Electronic Warfare Product Information" },
+    { MKPAIR(5, 4), "Acoustic Bearing and Range" },
+    { MKPAIR(6, 0), "Amplification" },
+    { MKPAIR(7, 0), "Track Management" },
+    { MKPAIR(7, 1), "Data Update Request" },
+    { MKPAIR(7, 2), "Correlation" },
+    { MKPAIR(7, 3), "Pointer" },
+    { MKPAIR(7, 4), "Track Identifier" },
+    { MKPAIR(7, 5), "IFF/SIF Management" },
+    { MKPAIR(7, 6), "Filter Management" },
+    { MKPAIR(7, 7), "Association" },
+    { MKPAIR(8, 0), "Unit Designator" },
+    { MKPAIR(8, 1), "Mission Correlator Change" },
+    { MKPAIR(9, 0), "Command" },
+    { MKPAIR(10, 2), "Engagement Status" },
+    { MKPAIR(10, 3), "Handover" },
+    { MKPAIR(10, 5), "Controlling Unit Report" },
+    { MKPAIR(10, 6), "Pairing" },
+    { MKPAIR(11, 0), "From the Weapon" },
+    { MKPAIR(11, 1), "To the Weapon" },
+    { MKPAIR(11, 2), "Weapon Coordination" },
+    { MKPAIR(12, 0), "Mission Assignment" },
+    { MKPAIR(12, 1), "Vector" },
+    { MKPAIR(12, 2), "Precision Aircraft Direction" },
+    { MKPAIR(12, 3), "Flight Path" },
+    { MKPAIR(12, 4), "Controlling Unit Change" },
+    { MKPAIR(12, 5), "Target/Track Correlation" },
+    { MKPAIR(12, 6), "Target Sorting" },
+    { MKPAIR(12, 7), "Target Bearing" },
+    { MKPAIR(13, 0), "Airfield Status" },
+    { MKPAIR(13, 2), "Air Platform and System Status" },
+    { MKPAIR(13, 3), "Surface Platform and System Status" },
+    { MKPAIR(13, 4), "Subsurface Platform and System Status" },
+    { MKPAIR(13, 5), "Land Platform and System Status" },
+    { MKPAIR(14, 0), "Parametric Information" },
+    { MKPAIR(14, 2), "Electronic Warfare Control / Coordination" },
+    { MKPAIR(15, 0), "Threat Warning" },
+    { MKPAIR(16, 0), "Imagery" },
+    { MKPAIR(17, 0), "Weather Over target" },
+    { MKPAIR(28, 0), "U.S. National 1 (Army)" },
+    { MKPAIR(28, 1), "U.S. National 2 (Navy)" },
+    { MKPAIR(28, 2), "U.S. National 3 (Air Force)" },
+    { MKPAIR(28, 3), "U.S. National 4 (Marine Corps)" },
+    { MKPAIR(28, 4), "French National 1" },
+    { MKPAIR(28, 5), "French National 2" },
+    { MKPAIR(28, 6), "U.S. National 5 (NSA)" },
+    { MKPAIR(28, 7), "UK National" },
+    { MKPAIR(31, 0), "Over-the-Air Rekeying Management" },
+    { MKPAIR(31, 1), "Over-the-Air Rekeying" },
+    { MKPAIR(31, 7), "No Statement" },
+    { 0, NULL },
+};
+
+/* Viasat, Inc., (2012), Link 16 Network Participant Group and Message Card, accessed from
<http://www.viasat.com/files/assets/assets/Link16_NPG_Message_Card_100112a.pdf> on 15 April
```

```
2013. */
+const value_string Link16_NPG_Strings[] = {
+    { 1, "Initial Entry" },
+    { 2, "RTT-A" },
+    { 3, "RTT-B" },
+    { 4, "Network Management" },
+    { 5, "PPLI and Status" },
+    { 6, "PPLI and Status" },
+    { 7, "Surveillance" },
+    { 8, "Mission Management/Weapons Coordination" },
+    { 9, "Control" },
+    { 11, "Image Transfer" },
+    { 12, "Voice A" },
+    { 13, "Voice B" },
+    { 18, "Network Enabled Weapons" },
+    { 19, "Figher-to-Fighter A" },
+    { 20, "Figher-to-Fighter B" },
+    { 21, "Engagement Coordination" },
+    { 27, "Joint Net PPLI" },
+    { 28, "Distributed Network Management" },
+    { 0, NULL },
+};
+
+static int proto_link16 = -1;
+
+static dissector_handle_t link16_handle;
+
+static gint hf_link16_wordformat = -1;
+static gint hf_link16_label = -1;
+static gint hf_link16_sublabel = -1;
+static gint hf_link16_mli = -1;
+static gint hf_link16_contlabel = -1;
+
+static gint ett_link16 = -1;
+
+static void dissect_link16(tvbuff_t *tvb, packet_info *pinfo, proto_tree *tree)
+{
+    proto_item *link16_item = NULL;
+    proto_tree *link16_tree = NULL;
+    guint32 cache;
+    guint8 wordformat, contlabel, mli;
+
+    Link16State *state = (Link16State*)pinfo->private_data;
+    if (!state) {
+        REPORT_DISSECTOR_BUG("Link 16 dissector state missing");
+    }
+
+    cache = tvb_get_letohl(tvb, 0);
+    wordformat = cache & 0x3;
+
+    col_set_str(pinfo->cinfo, COL_PROTOCOL, "Link 16");
+
+    if (tree) {
+        link16_item = proto_tree_add_item(tree, proto_link16, tvb, 0, -1, TRUE);
+        link16_tree = proto_item_add_subtree(link16_item, ett_link16);
+        proto_tree_add_uint(link16_tree, hf_link16_wordformat, tvb, 0, 0, wordformat);
+    }
+
+    /* Elmasry, G., (2012), Tactical Wireless Communications and Networks: Design Concepts
and Challenges, Wiley, ISBN 9781119951766. */
+    if (wordformat == WORDFORMAT_INITIAL) {
+        state->label     = (cache >> 2) & 0x1F;
+        state->sublabel  = (cache >> 7) & 0x7;
+        state->extension = 0;
+        mli = (cache >> 10) & 0x7;
+        col_append_fstr(pinfo->cinfo, COL_INFO, " J%d.%dI", state->label, state->sublabel);
+        if (tree) {
+            proto_item_append_text(link16_item, " J%d.%dI", state->label, state->sublabel);
+            proto_tree_add_uint(link16_tree, hf_link16_label, tvb, 0, 0, state->label);
+            proto_tree_add_uint(link16_tree, hf_link16_sublabel, tvb, 0, 0, state-
>sublabel);
```

```
+            proto_tree_add_uint(link16_tree, hf_link16_mli, tvb, 0, 0, mli);
+        }
+    } else if (wordformat == WORDFORMAT_EXTENSION) {
+        col_append_fstr(pinfo->cinfo, COL_INFO, " J%d.%dE%d", state->label, state-
>sublabel, state->extension);
+        if (tree)
+           proto_item_append_text(link16_item, " J%d.%dE%d", state->label, state->sublabel,
state->extension);
+        state->extension++;
+    } else if (wordformat == WORDFORMAT_CONTINUATION) {
+        contlabel = (cache >> 2) & 0x1F;
+        proto_tree_add_uint(link16_tree, hf_link16_contlabel, tvb, 0, 0, contlabel);
+        col_append_fstr(pinfo->cinfo, COL_INFO, " J%d.%dC%d", state->label, state-
>sublabel, contlabel);
+        if (tree)
+            proto_item_append_text(link16_item, " J%d.%dC%d", state->label, state-
>sublabel, contlabel);
+    } else {
+        return;
+    }
+
+    proto_item_append_text(link16_item, " %s", val_to_str_const(MKPAIR(state->label, state-
>sublabel), Link16_Message_Strings, "Unknown"));
+}
+
+void proto_register_link16(void)
+{
+    static hf_register_info hf[] = {
+        { &hf_link16_wordformat,
+          { "Word Format", "link16.wordformat", FT_UINT8, BASE_DEC,
VALS(WordFormat_Strings), 0x0,
+            NULL, HFILL }},
+        { &hf_link16_label,
+          { "Label", "link16.label", FT_UINT8, BASE_DEC, VALS(Link16_Label_Strings), 0x0,
+            NULL, HFILL }},
+        { &hf_link16_sublabel,
+          { "Sublabel", "link16.sublabel", FT_UINT8, BASE_DEC, NULL, 0x0,
+            NULL, HFILL }},
+        { &hf_link16_mli,
+          { "Message Length Indicator", "link16.mli", FT_UINT8, BASE_DEC, NULL, 0x0,
+            NULL, HFILL }},
+        { &hf_link16_contlabel,
+          { "Continuation Word Label", "link16.contlabel", FT_UINT8, BASE_DEC, NULL, 0x0,
+            NULL, HFILL }}
+    };
+    static gint *ett[] = {
+        &ett_link16,
+    };
+
+    proto_link16 = proto_register_protocol("Link 16", "LINK16", "link16");
+    proto_register_field_array(proto_link16, hf, array_length (hf));
+    proto_register_subtree_array(ett, array_length(ett));
+    register_dissector("link16", dissect_link16, proto_link16);
+}
+
+void proto_reg_handoff_link16(void)
+{
+    link16_handle = create_dissector_handle(dissect_link16, proto_link16);
+}
diff --git a/epan/dissectors/packet-link16.h b/epan/dissectors/packet-link16.h
new file mode 100644
index 0000000..9df0d4f
--- /dev/null
+++ b/epan/dissectors/packet-link16.h
@@ -0,0 +1,34 @@
+/* packet-link16.h
+ * Routines for Link 16 message dissection (MIL-STD-6016)
+ * William Robertson <aliask@gmail.com>
+ * Peter Ross <peter.ross@dsto.defence.gov.au>
+ *
+ * $Id$
```

```
+ *
+ * This program is free software; you can redistribute it and/or
+ * modify it under the terms of the GNU General Public License
+ * as published by the Free Software Foundation; either version 2
+ * of the License, or (at your option) any later version.
+ *
+ * This program is distributed in the hope that it will be useful,
+ * but WITHOUT ANY WARRANTY; without even the implied warranty of
+ * MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE.  See the
+ * GNU General Public License for more details.
+ *
+ * You should have received a copy of the GNU General Public License
+ * along with this program; if not, write to the Free Software
+ * Foundation, Inc., 51 Franklin Street, Fifth Floor, Boston, MA 02110-1301 USA
+ */
+
+#ifndef __PACKET_LINK16_H__
+#define __PACKET_LINK16_H__
+
+#include <epan/value_string.h>
+
+extern const value_string Link16_NPG_Strings[];
+
+typedef struct {
+    gint label, sublabel, extension;
+} Link16State;
+
+#endif /* __PACKET_LINK16_H__ */
--
1.8.1.2
```

Figure A1: Software patch (part 1 of 3)

```
From 93afe993b7f801c6a8b761c978bbdc26cf7bca06 Mon Sep 17 00:00:00 2001
From: Peter Ross <peter.ross@dsto.defence.gov.au>
Date: Wed, 16 Oct 2013 10:00:02 +1100
Subject: [PATCH 2/3] packet-dis: SISO-J Link 16 PDU dissector (SISO-STD-002)

---
 epan/dissectors/packet-dis-enums.c  |  20 +++++
 epan/dissectors/packet-dis-enums.h  |  16 ++++
 epan/dissectors/packet-dis-fields.c |  33 +++++--
 epan/dissectors/packet-dis-fields.h |  24 ++++-
 epan/dissectors/packet-dis-pdus.c   | 170 ++++++++++++++++++++++++++++++------
 epan/dissectors/packet-dis-pdus.h   |   2 +-
 epan/dissectors/packet-dis.c        |  74 +++++++++++++++-
 7 files changed, 293 insertions(+), 46 deletions(-)

diff --git a/epan/dissectors/packet-dis-enums.c b/epan/dissectors/packet-dis-enums.c
index 73d8264..d53b3a7 100644
--- a/epan/dissectors/packet-dis-enums.c
+++ b/epan/dissectors/packet-dis-enums.c
@@ -453,6 +453,25 @@ const value_string DIS_PDU_MajorModulation_Strings[] =
     {0,                                         NULL }
 };

+const range_string DIS_PDU_Link16_CVLL_Strings[] = {
+    { 0,   127, "Crypto Variable" },
+    { 255, 255, "NO STATEMENT" },
+    { 0,   0,   NULL }
+};
+
+const value_string DIS_PDU_Link16_MessageType_Strings[] =
+{
+    { DIS_MESSAGE_TYPE_JTIDS_HEADER_MESSAGES, "JTIDS Header/Messages" },
+    { DIS_MESSAGE_TYPE_RTT_A_B,               "RTT A/B" },
+    { DIS_MESSAGE_TYPE_RTT_REPLY,             "RTT Reply" },
+    { DIS_MESSAGE_TYPE_JTIDS_VOICE_CVSD,      "JTIDS Voice CVSD" },
+    { DIS_MESSAGE_TYPE_JTIDS_VOICE_LPC10,     "JTIDS Voice LPC10" },
```

```
+    { DIS_MESSAGE_TYPE_JTIDS_VOICE_LPC12,     "JTIDS Voice LPC12" },
+    { DIS_MESSAGE_TYPE_JTIDS_LET,             "JTIDS LET" },
+    { DIS_MESSAGE_TYPE_VMF,                   "VMF" },
+    { 0,                                      NULL }
+};
+
 const value_string DIS_PDU_EmissionFunction_Strings[] =
 {
     {DIS_EMISSION_FUNCTION_OTHER,                    "Other" },
@@ -663,6 +682,7 @@ const value_string DIS_PDU_TerminalSecondaryMode_Strings[] =
     {0,    NULL }
 };

+/* http://discussions.sisostds.org/threadview.aspx?fid=18&threadid=53172 */
 const value_string DIS_PDU_ModParamSyncState_Strings[] =
 {
     {0,    "Undefined" },
diff --git a/epan/dissectors/packet-dis-enums.h b/epan/dissectors/packet-dis-enums.h
index 913eb1f..0ab3b96 100644
--- a/epan/dissectors/packet-dis-enums.h
+++ b/epan/dissectors/packet-dis-enums.h
@@ -48,6 +48,8 @@ extern const value_string DIS_PDU_TSAllocationFidelity_Strings[];
 extern const value_string DIS_PDU_TerminalPrimaryMode_Strings[];
 extern const value_string DIS_PDU_TerminalSecondaryMode_Strings[];
 extern const value_string DIS_PDU_ModParamSyncState_Strings[];
+extern const range_string DIS_PDU_Link16_CVLL_Strings[];
+extern const value_string DIS_PDU_Link16_MessageType_Strings[];


 typedef enum
@@ -438,6 +440,20 @@ extern const value_string DIS_PDU_MajorModulation_Strings[];

 typedef enum
 {
+    DIS_MESSAGE_TYPE_JTIDS_HEADER_MESSAGES = 0,
+    DIS_MESSAGE_TYPE_RTT_A_B,
+    DIS_MESSAGE_TYPE_RTT_REPLY,
+    DIS_MESSAGE_TYPE_JTIDS_VOICE_CVSD,
+    DIS_MESSAGE_TYPE_JTIDS_VOICE_LPC10,
+    DIS_MESSAGE_TYPE_JTIDS_VOICE_LPC12,
+    DIS_MESSAGE_TYPE_JTIDS_LET,
+    DIS_MESSAGE_TYPE_VMF
+} DIS_PDU_MessageType;
+
+extern const value_string DIS_PDU_JTIDS_MessageType_Strings[];
+
+typedef enum
+{
     DIS_EMISSION_FUNCTION_OTHER                         = 0,
     DIS_EMISSION_FUNCTION_MULTI_FUNCTION                = 1,
     DIS_EMISSION_FUNCTION_EARLY_WARNING_SURVEILLANCE    = 2,
diff --git a/epan/dissectors/packet-dis-fields.c b/epan/dissectors/packet-dis-fields.c
index 2b2007f..c6d7440c 100644
--- a/epan/dissectors/packet-dis-fields.c
+++ b/epan/dissectors/packet-dis-fields.c
@@ -43,7 +43,9 @@ guint32 category;
 guint32 radioID;
 guint32 disRadioTransmitState;
 guint32 encodingScheme;
+guint32 tdlType;
 guint32 numSamples;
+guint32 messageType;
 guint32 numFixed;
 guint32 numVariable;
 guint32 numBeams;
@@ -209,6 +211,20 @@ DIS_ParserNode DIS_FIELDS_MOD_PARAMS_JTIDS_MIDS[] =
     { DIS_FIELDTYPE_END,                          NULL,0,0,0,0 }
 };

+DIS_ParserNode DIS_FIELDS_SIGNAL_LINK16_NETWORK_HEADER[] =
+{
```

```
+    { DIS_FIELDTYPE_LINK16_NPG,          "Network Participant Group",0,0,0,0 },
+    { DIS_FIELDTYPE_UINT8,               "Network Number",0,0,0,0 },
+    { DIS_FIELDTYPE_LINK16_TSEC_CVLL,    "TSEC CVLL",0,0,0,0 },
+    { DIS_FIELDTYPE_LINK16_MSEC_CVLL,    "MSEC CVLL",0,0,0,0 },
+    { DIS_FIELDTYPE_LINK16_MESSAGE_TYPE, "Message Type",0,0,0,&messageType },
+    { DIS_FIELDTYPE_UINT16,              "Padding",0,0,0,0 },
+    { DIS_FIELDTYPE_UINT32,              "Time Slot ID",0,0,0,0 },
+    { DIS_FIELDTYPE_LINK16_PTT,          "Perceived Transmit Time",0,0,0,0 },
+    { DIS_FIELDTYPE_LINK16_MESSAGE_DATA, "Message Data",0,0,0,0 },
+    { DIS_FIELDTYPE_END,                 NULL,0,0,0,0 }
+};
+
 /* Array records
  */
 DIS_ParserNode DIS_FIELDS_FIXED_DATUM[] =
@@ -513,6 +529,7 @@ void initializeFieldParsers(void)
     initializeParser(DIS_FIELDS_VR_UA_BEAM);
     initializeParser(DIS_FIELDS_MOD_PARAMS_CCTT_SINCGARS);
     initializeParser(DIS_FIELDS_MOD_PARAMS_JTIDS_MIDS);
+    initializeParser(DIS_FIELDS_SIGNAL_LINK16_NETWORK_HEADER);

 }

@@ -868,6 +885,10 @@ gint parseField_Enum(tvbuff_t *tvb, proto_tree *tree, gint offset,
DIS_ParserNod
             break;
         }
         break;
+    case DIS_FIELDTYPE_LINK16_MESSAGE_TYPE:
+        enumStrings = DIS_PDU_Link16_MessageType_Strings;
+        dis_hf_id = hf_dis_signal_link16_message_type;
+        break;
     default:
         enumStrings = 0;
         break;
@@ -1019,7 +1040,7 @@ gint parseField_Timestamp(tvbuff_t *tvb, proto_tree *tree, gint
offset, DIS_Pars

 /* Parse a variable parameter field.
  */
-gint parseField_VariableParameter(tvbuff_t *tvb, proto_tree *tree, gint offset)
+gint parseField_VariableParameter(tvbuff_t *tvb, proto_tree *tree, gint offset, packet_info
*pinfo)
 {
     DIS_ParserNode *paramParser = 0;

@@ -1045,7 +1066,7 @@ gint parseField_VariableParameter(tvbuff_t *tvb, proto_tree *tree,
gint offset)
     /* Parse the variable parameter fields */
     if (paramParser)
     {
-        offset = parseFields(tvb, tree, offset, paramParser);
+        offset = parseFields(tvb, tree, offset, paramParser, pinfo);
     }

     return offset;
@@ -1053,7 +1074,7 @@ gint parseField_VariableParameter(tvbuff_t *tvb, proto_tree *tree,
gint offset)

 /* Parse a variable record field.
  */
-gint parseField_VariableRecord(tvbuff_t *tvb, proto_tree *tree, gint offset)
+gint parseField_VariableRecord(tvbuff_t *tvb, proto_tree *tree, gint offset, packet_info
*pinfo)
 {
     DIS_ParserNode *paramParser = 0;

@@ -1086,7 +1107,7 @@ gint parseField_VariableRecord(tvbuff_t *tvb, proto_tree *tree, gint
offset)
     /* Parse the variable record fields */
     if (paramParser)
```

```
     {
-        offset = parseFields(tvb, tree, offset, paramParser);
+        offset = parseFields(tvb, tree, offset, paramParser, pinfo);
     }

     /* Should alignment padding be added */
@@ -1105,7 +1126,7 @@ gint parseField_VariableRecord(tvbuff_t *tvb, proto_tree *tree, gint
offset)
 /* Parse a variable electromagnetic emission system beam.
  */
 gint parseField_ElectromagneticEmissionSystemBeam(
-    tvbuff_t *tvb, proto_tree *tree, gint offset)
+    tvbuff_t *tvb, proto_tree *tree, gint offset, packet_info *pinfo)
 {
     DIS_ParserNode *paramParser = 0;

@@ -1116,7 +1137,7 @@ gint parseField_ElectromagneticEmissionSystemBeam(
     /* Parse the variable parameter fields */
     if (paramParser)
     {
-        offset = parseFields(tvb, tree, offset, paramParser);
+        offset = parseFields(tvb, tree, offset, paramParser, pinfo);
     }

     return offset;
diff --git a/epan/dissectors/packet-dis-fields.h b/epan/dissectors/packet-dis-fields.h
index d306a4d..477bcdb 100644
--- a/epan/dissectors/packet-dis-fields.h
+++ b/epan/dissectors/packet-dis-fields.h
@@ -54,6 +54,7 @@ extern int hf_dis_radio_id;
 extern int hf_dis_ens;
 extern int hf_dis_ens_class;
 extern int hf_dis_ens_type;
+extern int hf_dis_ens_type_audio;
 extern int hf_dis_tdl_type;
 extern int hf_dis_sample_rate;
 extern int hf_dis_data_length;
@@ -96,9 +97,17 @@ extern int hf_dis_antenna_pattern_parameter_dump;
 extern int hf_dis_num_shafts;
 extern int hf_dis_num_apas;
 extern int hf_dis_num_ua_emitter_systems;
+extern int hf_dis_signal_link16_npg;
+extern int hf_dis_signal_link16_tsec_cvll;
+extern int hf_dis_signal_link16_msec_cvll;
+extern int hf_dis_signal_link16_message_type;
+extern int hf_dis_signal_link16_ptt;
+extern int hf_dis_signal_link16_stn;

 extern int ett_dis_ens;
 extern int ett_dis_crypto_key;
+extern int ett_dis_signal_link16_network_header;
+extern int ett_dis_signal_link16_message_data;



@@ -214,6 +223,10 @@ typedef enum
     DIS_FIELDTYPE_TRANSMITTER_SECONDARY_MODE,
     DIS_FIELDTYPE_JTIDS_SYNC_STATE,
     DIS_FIELDTYPE_NETWORK_SYNC_ID,
+    DIS_FIELDTYPE_LINK16_NPG,
+    DIS_FIELDTYPE_LINK16_TSEC_CVLL,
+    DIS_FIELDTYPE_LINK16_MSEC_CVLL,
+    DIS_FIELDTYPE_LINK16_MESSAGE_TYPE,
     DIS_FIELDTYPE_NUM_ELECTROMAGNETIC_EMISSION_SYSTEMS,
     DIS_FIELDTYPE_NUM_OF_SHAFTS,
     DIS_FIELDTYPE_NUM_OF_APAS,
@@ -244,6 +257,8 @@ typedef enum
     DIS_FIELDTYPE_ANTENNA_PATTERN_PARAMETERS,
     DIS_FIELDTYPE_MOD_PARAMS_CCTT_SINCGARS,
     DIS_FIELDTYPE_MOD_PARAMS_JTIDS_MIDS,
+    DIS_FIELDTYPE_LINK16_MESSAGE_DATA,
```

```
+    DIS_FIELDTYPE_LINK16_PTT,
     DIS_FIELDTYPE_ELECTROMAGNETIC_EMISSION_SYSTEM_BEAM,
     DIS_FIELDTYPE_ELECTROMAGNETIC_EMISSION_SYSTEM,
     DIS_FIELDTYPE_EMITTER_SYSTEM,
@@ -316,6 +331,7 @@ extern DIS_ParserNode DIS_FIELDS_VECTOR_FLOAT_32[];
 extern DIS_ParserNode DIS_FIELDS_VECTOR_FLOAT_64[];
 extern DIS_ParserNode DIS_FIELDS_MOD_PARAMS_CCTT_SINCGARS[];
 extern DIS_ParserNode DIS_FIELDS_MOD_PARAMS_JTIDS_MIDS[];
+extern DIS_ParserNode DIS_FIELDS_SIGNAL_LINK16_NETWORK_HEADER[];
 extern DIS_ParserNode DIS_FIELDS_EMITTER_SYSTEM[];
 extern DIS_ParserNode DIS_FIELDS_FUNDAMENTAL_PARAMETER_DATA[];
 extern DIS_ParserNode DIS_FIELDS_TRACK_JAM[];
@@ -360,11 +376,11 @@ extern gint parseField_Double(tvbuff_t *tvb, proto_tree *tree, gint
offset, DIS_

 extern gint parseField_Timestamp(tvbuff_t *tvb, proto_tree *tree, gint offset,
DIS_ParserNode parserNode);

-extern gint parseField_VariableParameter(tvbuff_t *tvb, proto_tree *tree, gint offset);
+extern gint parseField_VariableParameter(tvbuff_t *tvb, proto_tree *tree, gint offset,
packet_info *pinfo);

-extern gint parseField_VariableRecord(tvbuff_t *tvb, proto_tree *tree, gint offset);
+extern gint parseField_VariableRecord(tvbuff_t *tvb, proto_tree *tree, gint offset,
packet_info *pinfo);

-extern gint parseField_ElectromagneticEmissionSystemBeam(tvbuff_t *tvb, proto_tree *tree,
gint offset);
+extern gint parseField_ElectromagneticEmissionSystemBeam(tvbuff_t *tvb, proto_tree *tree,
gint offset, packet_info *pinfo);

 extern guint32 disProtocolVersion;
 extern guint32 pduType;
@@ -375,7 +391,9 @@ extern guint32 entityDomain;
 extern guint32 radioID;
 extern guint32 disRadioTransmitState;
 extern guint32 encodingScheme;
+extern guint32 tdlType;
 extern guint32 numSamples;
+extern guint32 messageType;
 extern guint32 numFixed;
 extern guint32 numVariable;
 extern guint32 numBeams;
diff --git a/epan/dissectors/packet-dis-pdus.c b/epan/dissectors/packet-dis-pdus.c
index 939c759..f1e1f30 100644
--- a/epan/dissectors/packet-dis-pdus.c
+++ b/epan/dissectors/packet-dis-pdus.c
@@ -29,6 +29,8 @@
 #include "packet-dis-pdus.h"
 #include "packet-dis-fields.h"
 #include "packet-dis-enums.h"
+#include "packet-link16.h"
+#include "packet-ntp.h"

 #define DIS_PDU_MAX_VARIABLE_PARAMETERS             16
 #define DIS_PDU_MAX_VARIABLE_RECORDS                16
@@ -131,7 +133,7 @@ DIS_ParserNode DIS_PARSER_SIGNAL_PDU[] =
     { DIS_FIELDTYPE_ENTITY_ID,               "Entity ID",0,0,0,0 },
     { DIS_FIELDTYPE_RADIO_ID,                "Radio ID",0,0,0,&radioID },
     { DIS_FIELDTYPE_ENCODING_SCHEME,         "Encoding Scheme",0,0,0,&encodingScheme },
-    { DIS_FIELDTYPE_TDL_TYPE,                "TDL Type",0,0,0,0 },
+    { DIS_FIELDTYPE_TDL_TYPE,                "TDL Type",0,0,0,&tdlType },
     { DIS_FIELDTYPE_SAMPLE_RATE,             "Sample Rate",0,0,0,0 },
     { DIS_FIELDTYPE_DATA_LENGTH,             "Data Length",0,0,0,0 },
     { DIS_FIELDTYPE_NUMBER_OF_SAMPLES,       "Number of Samples",0,0,0,&numSamples },
@@ -732,9 +734,92 @@ void initializeParser(DIS_ParserNode parserNodes[])
     }
 }

+/* Parse Link 16 Message Data record (SISO-STD-002, Tables 5.2.5 through 5.2.12)
+ */
```

```
+static gint parse_Link16_Message_Data(proto_tree *tree, tvbuff_t *tvb, gint offset,
packet_info *pinfo)
+{
+    guint32 cache, value, i;
+    Link16State state;
+    tvbuff_t *newtvb;
+
+    switch (messageType) {
+    case DIS_MESSAGE_TYPE_JTIDS_HEADER_MESSAGES:
+        cache = tvb_get_ntohl(tvb, offset);
+        value = cache & 0x7;
+        proto_tree_add_text(tree, tvb, offset, 4,
+            "Time Slot Type: %d", value);
+
+        value = (cache >> 3) & 0x1;
+        proto_tree_add_text(tree, tvb, offset, 4,
+            "Relay Transmission Indicator: %d", value);
+
+        value = (cache >> 4) & 0x7FFF;
+        proto_tree_add_uint(tree, hf_dis_signal_link16_stn, tvb, offset, 4, value);
+
+        col_append_fstr(pinfo->cinfo, COL_INFO, ", STN=0%o, Link 16 Words:", value);
+
+        value = (cache >> 19);
+        offset += 4;
+
+        cache = tvb_get_ntohl(tvb, offset);
+        value |= (cache & 0x7) << 13;
+        proto_tree_add_text(tree, tvb, offset - 4, 8,
+            "Secure Data Unit Serial Number: %d", value);
+
+        offset += 4;
+
+        memset(&state, 0, sizeof(state));
+        pinfo->private_data = &state;
+
+        for (i = 0; i < (encodingScheme & 0x3FFF); i++) {
+            gint8 *word = (gint8 *)g_malloc(10);
+            if (!(i & 1)) {
+                word[0] = (cache >> 16) & 0xFF;
+                word[1] = (cache >> 24) & 0xFF;
+                cache = tvb_get_ntohl(tvb, offset);
+                offset += 4;
+                word[2] = cache & 0xFF;
+                word[3] = (cache >> 8) & 0xFF;
+                word[4] = (cache >> 16) & 0xFF;
+                word[5] = (cache >> 24) & 0xFF;
+                cache = tvb_get_ntohl(tvb, offset);
+                offset += 4;
+                word[6] = cache & 0xFF;
+                word[7] = (cache >> 8) & 0xFF;
+                word[8] = (cache >> 16) & 0xFF;
+                word[9] = (cache >> 24) & 0xFF;
+            } else {
+                cache = tvb_get_ntohl(tvb, offset);
+                offset += 4;
+                word[0] = cache & 0xFF;
+                word[1] = (cache >> 8) & 0xFF;
+                word[2] = (cache >> 16) & 0xFF;
+                word[3] = (cache >> 24) & 0xFF;
+                cache = tvb_get_ntohl(tvb, offset);
+                offset += 4;
+                word[4] = cache & 0xFF;
+                word[5] = (cache >> 8) & 0xFF;
+                word[6] = (cache >> 16) & 0xFF;
+                word[7] = (cache >> 24) & 0xFF;
+                cache = tvb_get_ntohl(tvb, offset);
+                offset += 4;
+                word[8] = cache & 0xFF;
+                word[9] = (cache >> 8) & 0xFF;
+            }
```

```
+
+            newtvb = tvb_new_child_real_data(tvb, word, 10, 10);
+            tvb_set_free_cb(newtvb, g_free);
+            add_new_data_source(pinfo, newtvb, "Link 16 Word");
+            call_dissector(find_dissector("link16"), newtvb, pinfo, tree);
+        }
+        break;
+    }
+    return offset;
+}
+
 /* Parse packet data based on a specified array of DIS_ParserNodes.
  */
-gint parseFields(tvbuff_t *tvb, proto_tree *tree, gint offset, DIS_ParserNode
parserNodes[])
+gint parseFields(tvbuff_t *tvb, proto_tree *tree, gint offset, DIS_ParserNode
parserNodes[], packet_info *pinfo)
 {
     guint        fieldIndex     = 0;
     guint        fieldRepeatLen = 0;
@@ -839,13 +924,17 @@ gint parseFields(tvbuff_t *tvb, proto_tree *tree, gint offset,
DIS_ParserNode pa
             pi = proto_tree_add_item(tree, hf_dis_ens, tvb, offset, 2, ENC_BIG_ENDIAN);
             sub_tree = proto_item_add_subtree(pi, ett_dis_ens);
             proto_tree_add_item(sub_tree, hf_dis_ens_class, tvb, offset, 2,
ENC_BIG_ENDIAN);
-            proto_tree_add_item(sub_tree, hf_dis_ens_type, tvb, offset, 2, ENC_BIG_ENDIAN);
+            proto_tree_add_item(sub_tree,
+                (uintVal >> 14) == DIS_ENCODING_CLASS_ENCODED_AUDIO ? hf_dis_ens_type_audio
: hf_dis_ens_type,
+                tvb, offset, 2, ENC_BIG_ENDIAN);
             proto_item_set_end(pi, tvb, offset);
             *(parserNodes[fieldIndex].outputVar) = (guint32)uintVal;
             offset += 2;
             break;
         case DIS_FIELDTYPE_TDL_TYPE:
+            uintVal = tvb_get_ntohs(tvb, offset);
             proto_tree_add_item(tree, hf_dis_tdl_type, tvb, offset, 2, ENC_BIG_ENDIAN);
+            *(parserNodes[fieldIndex].outputVar) = (guint32)uintVal;
             offset += 2;
             break;
         case DIS_FIELDTYPE_SAMPLE_RATE:
@@ -862,12 +951,31 @@ gint parseFields(tvbuff_t *tvb, proto_tree *tree, gint offset,
DIS_ParserNode pa
             *(parserNodes[fieldIndex].outputVar) = (guint32)uintVal;
             offset += 2;
             break;
-
         case DIS_FIELDTYPE_RADIO_DATA:
-            newtvb = tvb_new_subset_remaining(tvb, offset);
-            proto_tree_add_item(tree, hf_dis_signal_data, newtvb, 0, -1, ENC_NA );
+            if (tdlType == DIS_TDL_TYPE_LINK16_STD) {
+                pi = proto_tree_add_text(tree, tvb, offset, 16, "Link 16 Network Header");
+                sub_tree = proto_item_add_subtree(pi,
ett_dis_signal_link16_network_header);
+                offset = parseFields(tvb, sub_tree, offset,
DIS_FIELDS_SIGNAL_LINK16_NETWORK_HEADER, pinfo);
+                proto_item_set_end(pi, tvb, offset);
+
+                pi = proto_tree_add_text(tree, tvb, offset, -1, "Link 16 Message Data: %s",
+                    val_to_str(messageType, DIS_PDU_Link16_MessageType_Strings, ""));
+                sub_tree = proto_item_add_subtree(pi, ett_dis_signal_link16_message_data);
+                offset = parse_Link16_Message_Data(sub_tree, tvb, offset, pinfo);
+                proto_item_set_end(pi, tvb, offset);
+            } else {
+                newtvb = tvb_new_subset_remaining(tvb, offset);
+                proto_tree_add_item(tree, hf_dis_signal_data, newtvb, 0, -1, ENC_NA );
+            }
             /* ****ck******* need to look for padding bytes */
             break;
+        case DIS_FIELDTYPE_LINK16_PTT:
```

```
+            if (tvb_get_ntohl(tvb, offset) == 0xFFFFFFFF)
+                proto_tree_add_text(tree, tvb, offset, 8, "%s: NO STATEMENT",
parserNodes[fieldIndex].fieldLabel);
+            else
+                proto_tree_add_item(tree, hf_dis_signal_link16_ptt, tvb, offset, 8,
ENC_TIME_NTP|ENC_BIG_ENDIAN);
+            offset += 8;
+            break;
         case DIS_FIELDTYPE_RADIO_CATEGORY:
             proto_tree_add_item(tree, hf_dis_radio_category, tvb, offset, 1,
ENC_BIG_ENDIAN);
             offset += 1;
@@ -1003,7 +1111,7 @@ gint parseFields(tvbuff_t *tvb, proto_tree *tree, gint offset,
DIS_ParserNode pa
                     pi = proto_tree_add_text(tree, tvb, offset, -1, "%s",
                                              parserNodes[fieldIndex].fieldLabel);
                     sub_tree = proto_item_add_subtree(pi, parserNodes[fieldIndex].ettVar);
-                    offset = parseFields(tvb, sub_tree, offset,
DIS_FIELDS_MOD_PARAMS_CCTT_SINCGARS);
+                    offset = parseFields(tvb, sub_tree, offset,
DIS_FIELDS_MOD_PARAMS_CCTT_SINCGARS, pinfo);
                     proto_item_set_end(pi, tvb, offset);
                     break;
                 }
@@ -1011,7 +1119,7 @@ gint parseFields(tvbuff_t *tvb, proto_tree *tree, gint offset,
DIS_ParserNode pa
                     pi = proto_tree_add_text(tree, tvb, offset, -1, "%s",
                                              parserNodes[fieldIndex].fieldLabel);
                     sub_tree = proto_item_add_subtree(pi, parserNodes[fieldIndex].ettVar);
-                    offset = parseFields(tvb, sub_tree, offset,
DIS_FIELDS_MOD_PARAMS_JTIDS_MIDS);
+                    offset = parseFields(tvb, sub_tree, offset,
DIS_FIELDS_MOD_PARAMS_JTIDS_MIDS, pinfo);
                     proto_item_set_end(pi, tvb, offset);
                     break;
                 }
@@ -1028,7 +1136,18 @@ gint parseFields(tvbuff_t *tvb, proto_tree *tree, gint offset,
DIS_ParserNode pa
             newtvb = tvb_new_subset_remaining(tvb, offset);
             proto_tree_add_item(tree, hf_dis_antenna_pattern_parameter_dump, newtvb, 0, -1,
ENC_NA );
             break;
-
+        case DIS_FIELDTYPE_LINK16_NPG:
+            proto_tree_add_item(tree, hf_dis_signal_link16_npg, tvb, offset, 2,
ENC_BIG_ENDIAN);
+            offset += 2;
+            break;
+        case DIS_FIELDTYPE_LINK16_TSEC_CVLL:
+            proto_tree_add_item(tree, hf_dis_signal_link16_tsec_cvll, tvb, offset, 1,
ENC_NA);
+            offset++;
+            break;
+        case DIS_FIELDTYPE_LINK16_MSEC_CVLL:
+            proto_tree_add_item(tree, hf_dis_signal_link16_msec_cvll, tvb, offset, 1,
ENC_NA);
+            offset++;
+            break;

         /* padding */
         case DIS_FIELDTYPE_PAD8:
@@ -1066,6 +1185,7 @@ gint parseFields(tvbuff_t *tvb, proto_tree *tree, gint offset,
DIS_ParserNode pa
         case DIS_FIELDTYPE_PERSISTENT_OBJECT_TYPE:
         case DIS_FIELDTYPE_EMISSION_FUNCTION:
         case DIS_FIELDTYPE_BEAM_FUNCTION:
+        case DIS_FIELDTYPE_LINK16_MESSAGE_TYPE:
             offset = parseField_Enum(tvb, tree, offset,
                 parserNodes[fieldIndex], 1);
             break;
@@ -1210,7 +1330,7 @@ gint parseFields(tvbuff_t *tvb, proto_tree *tree, gint offset,
```

```
DIS_ParserNode pa
                  proto_item *newSubtree = proto_item_add_subtree(newField,
                      parserNodes[fieldIndex].ettVar);
                  offset = parseFields(tvb, newSubtree, offset,
-                     parserNodes[fieldIndex].children);
+                     parserNodes[fieldIndex].children, pinfo);
              }
              proto_item_set_end(newField, tvb, offset);
              break;
@@ -1248,7 +1368,7 @@ gint parseFields(tvbuff_t *tvb, proto_tree *tree, gint offset,
DIS_ParserNode pa
                                                      parserNodes[fieldIndex].fieldLabel);
                      newSubtree = proto_item_add_subtree(newField, ettFixedData);
                      offset = parseFields (tvb, newSubtree, offset,
-                          parserNodes[fieldIndex].children);
+                          parserNodes[fieldIndex].children, pinfo);
                      proto_item_set_end(newField, tvb, offset);
                  }
              }
@@ -1274,7 +1394,7 @@ gint parseFields(tvbuff_t *tvb, proto_tree *tree, gint offset,
DIS_ParserNode pa
                      /* XXX is this really necessary? */
                      tvb_ensure_length_remaining(tvb, offset+4);
                      offset = parseFields (tvb, newSubtree, offset,
-                         parserNodes[fieldIndex].children);
+                         parserNodes[fieldIndex].children, pinfo);
                  }
                  proto_item_set_end(newField, tvb, offset);
              }
@@ -1296,7 +1416,7 @@ gint parseFields(tvbuff_t *tvb, proto_tree *tree, gint offset,
DIS_ParserNode pa
                          (newField, ettVariableData);
                      offset = parseFields
                          (tvb, newSubtree, offset,
-                          parserNodes[fieldIndex].children);
+                          parserNodes[fieldIndex].children, pinfo);
                      proto_item_set_end(newField, tvb, offset);
                  }

@@ -1321,7 +1441,7 @@ gint parseFields(tvbuff_t *tvb, proto_tree *tree, gint offset,
DIS_ParserNode pa
                  {
                      offset = parseFields
                          (tvb, newSubtree, offset,
-                          parserNodes[fieldIndex].children);
+                          parserNodes[fieldIndex].children, pinfo);
                  }
                  proto_item_set_end(newField, tvb, offset);
              }
@@ -1344,9 +1464,9 @@ gint parseFields(tvbuff_t *tvb, proto_tree *tree, gint offset,
DIS_ParserNode pa
                          ettVariableParameters[i]);
                      offset = parseFields
                          (tvb, newSubtree, offset,
-                          parserNodes[fieldIndex].children);
+                          parserNodes[fieldIndex].children, pinfo);
                      offset = parseField_VariableParameter
-                          (tvb, newSubtree, offset);
+                          (tvb, newSubtree, offset, pinfo);
                      proto_item_set_end(newField, tvb, offset);
                  }
              }
@@ -1369,9 +1489,9 @@ gint parseFields(tvbuff_t *tvb, proto_tree *tree, gint offset,
DIS_ParserNode pa
                                                          ettVariableRecords[i]);
                      offset = parseFields
                          (tvb, newSubtree, offset,
-                          parserNodes[fieldIndex].children);
+                          parserNodes[fieldIndex].children, pinfo);
                      offset = parseField_VariableRecord
-                          (tvb, newSubtree, offset);
```

```
+                       (tvb, newSubtree, offset, pinfo);
                    proto_item_set_end(newField, tvb, offset);
                }
            }
@@ -1390,7 +1510,7 @@ gint parseFields(tvbuff_t *tvb, proto_tree *tree, gint offset,
DIS_ParserNode pa
                            proto_item_add_subtree(newField,
                            parserNodes[fieldIndex].ettVar);
                        offset = parseFields(tvb, newSubtree, offset,
-                            parserNodes[fieldIndex].children);
+                            parserNodes[fieldIndex].children, pinfo);
                    }
                    proto_item_set_end(newField, tvb, offset);
                }
@@ -1410,7 +1530,7 @@ gint parseFields(tvbuff_t *tvb, proto_tree *tree, gint offset,
DIS_ParserNode pa
                            proto_item_add_subtree(newField,
                            parserNodes[fieldIndex].ettVar);
                        offset = parseFields(tvb, newSubtree, offset,
-                            parserNodes[fieldIndex].children);
+                            parserNodes[fieldIndex].children, pinfo);
                    }
                    proto_item_set_end(newField, tvb, offset);
                }
@@ -1441,7 +1561,7 @@ gint parseFields(tvbuff_t *tvb, proto_tree *tree, gint offset,
DIS_ParserNode pa
                            proto_item_add_subtree(newField,
                            parserNodes[fieldIndex].ettVar);
                        offset = parseFields(tvb, newSubtree, offset,
-                            parserNodes[fieldIndex].children);
+                            parserNodes[fieldIndex].children, pinfo);
                    }
                    proto_item_set_end(newField, tvb, offset);
                }
@@ -1474,7 +1594,7 @@ gint parseFields(tvbuff_t *tvb, proto_tree *tree, gint offset,
DIS_ParserNode pa
                            proto_item_add_subtree(newField,
                            parserNodes[fieldIndex].ettVar);
                        offset = parseFields(tvb, newSubtree, offset,
-                            parserNodes[fieldIndex].children);
+                            parserNodes[fieldIndex].children, pinfo);
                    }
                    proto_item_set_end(newField, tvb, offset);
                }
@@ -1507,7 +1627,7 @@ gint parseFields(tvbuff_t *tvb, proto_tree *tree, gint offset,
DIS_ParserNode pa
                            proto_item_add_subtree(newField,
                            parserNodes[fieldIndex].ettVar);
                        offset = parseFields(tvb, newSubtree, offset,
-                            parserNodes[fieldIndex].children);
+                            parserNodes[fieldIndex].children, pinfo);
                    }
                    proto_item_set_end(newField, tvb, offset);
                }
@@ -1547,7 +1667,7 @@ gint parseFields(tvbuff_t *tvb, proto_tree *tree, gint offset,
DIS_ParserNode pa
                            proto_item_add_subtree(newField,
                            parserNodes[fieldIndex].ettVar);
                        offset = parseFields(tvb, newSubtree, offset,
-                            parserNodes[fieldIndex].children);
+                            parserNodes[fieldIndex].children, pinfo);
                    }
                    proto_item_set_end(newField, tvb, offset);
                }
@@ -1572,7 +1692,7 @@ gint parseFields(tvbuff_t *tvb, proto_tree *tree, gint offset,
DIS_ParserNode pa
                            proto_item_add_subtree(newField,
                            parserNodes[fieldIndex].ettVar);
                        offset = parseFields(tvb, newSubtree, offset,
-                            parserNodes[fieldIndex].children);
+                            parserNodes[fieldIndex].children, pinfo);
```

```
                    }
                    proto_item_set_end(newField, tvb, offset);
                }
diff --git a/epan/dissectors/packet-dis-pdus.h b/epan/dissectors/packet-dis-pdus.h
index e964371..b9d2448 100644
--- a/epan/dissectors/packet-dis-pdus.h
+++ b/epan/dissectors/packet-dis-pdus.h
@@ -111,6 +111,6 @@ void initializeParser(DIS_ParserNode parserNodes[]);

 void initializeParsers(void);

-gint parseFields(tvbuff_t *tvb, proto_tree *tree, gint offset, DIS_ParserNode
parserNodes[]);
+gint parseFields(tvbuff_t *tvb, proto_tree *tree, gint offset, DIS_ParserNode
parserNodes[], packet_info *pinfo);

 #endif /* packet-dis-pduparsers.h */
diff --git a/epan/dissectors/packet-dis.c b/epan/dissectors/packet-dis.c
index da012a9..0a00009 100644
--- a/epan/dissectors/packet-dis.c
+++ b/epan/dissectors/packet-dis.c
@@ -46,6 +46,7 @@
 #include "packet-dis-enums.h"
 #include "packet-dis-pdus.h"
 #include "packet-dis-fields.h"
+#include "packet-link16.h"

 #define DEFAULT_DIS_UDP_PORT 3000

@@ -80,6 +81,7 @@ int hf_dis_radio_id = -1;
 int hf_dis_ens = -1;
 int hf_dis_ens_class = -1;
 int hf_dis_ens_type = -1;
+int hf_dis_ens_type_audio = -1;
 int hf_dis_tdl_type = -1;
 int hf_dis_sample_rate = -1;
 int hf_dis_data_length = -1;
@@ -122,6 +124,12 @@ int hf_dis_antenna_pattern_parameter_dump = -1;
 int hf_dis_num_shafts = -1;
 int hf_dis_num_apas = -1;
 int hf_dis_num_ua_emitter_systems = -1;
+int hf_dis_signal_link16_npg = -1;
+int hf_dis_signal_link16_tsec_cvll = -1;
+int hf_dis_signal_link16_msec_cvll = -1;
+int hf_dis_signal_link16_message_type = -1;
+int hf_dis_signal_link16_ptt = -1;
+int hf_dis_signal_link16_stn = -1;

 /* Initialize the subtree pointers */
 static gint ett_dis = -1;
@@ -130,6 +138,8 @@ static gint ett_dis_po_header = -1;
 static gint ett_dis_payload = -1;
 int ett_dis_ens = -1;
 int ett_dis_crypto_key = -1;
+int ett_dis_signal_link16_network_header = -1;
+int ett_dis_signal_link16_message_data = -1;

 static const true_false_string dis_modulation_spread_spectrum = {
     "Spread Spectrum modulation in use",
@@ -206,7 +216,7 @@ static gint dissect_dis(tvbuff_t *tvb, packet_info *pinfo, proto_tree
*tree, voi
      */
     dis_header_node = proto_tree_add_text(dis_tree, tvb, offset, -1, "Header");
     dis_header_tree = proto_item_add_subtree(dis_header_node, ett_dis_header);
-    offset = parseFields(tvb, dis_header_tree, offset, DIS_FIELDS_PDU_HEADER);
+    offset = parseFields(tvb, dis_header_tree, offset, DIS_FIELDS_PDU_HEADER, pinfo);

     proto_item_set_end(dis_header_node, tvb, offset);

@@ -230,7 +240,7 @@ static gint dissect_dis(tvbuff_t *tvb, packet_info *pinfo, proto_tree
*tree, voi
```

```
                (dis_po_header_node, ett_dis_po_header);
            offset = parseFields
                (tvb, dis_po_header_tree, offset,
-                DIS_FIELDS_PERSISTENT_OBJECT_HEADER);
+                DIS_FIELDS_PERSISTENT_OBJECT_HEADER, pinfo);
            proto_item_set_end(dis_po_header_node, tvb, offset);

            /* Locate the appropriate PO PDU parser, if type is known.
@@ -400,13 +410,15 @@ static gint dissect_dis(tvbuff_t *tvb, packet_info *pinfo, proto_tree
*tree, voi
         break;
     }

+    col_clear(pinfo->cinfo, COL_INFO);
+
     /* If a parser was located, invoke it on the data packet.
      */
     if (pduParser != 0)
     {
         dis_payload_tree = proto_item_add_subtree(dis_payload_node,
             ett_dis_payload);
-        offset = parseFields(tvb, dis_payload_tree, offset, pduParser);
+        offset = parseFields(tvb, dis_payload_tree, offset, pduParser, pinfo);

         proto_item_set_end(dis_payload_node, tvb, offset);
     }
@@ -434,13 +446,17 @@ static gint dissect_dis(tvbuff_t *tvb, packet_info *pinfo, proto_tree
*tree, voi
                     );
         break;
     case DIS_PDUTYPE_SIGNAL:
-        col_add_fstr( pinfo->cinfo, COL_INFO,
-                      "PDUType: %s, RadioID=%u, Encoding Type=%s, Number of Samples=%u",
-                      pduString,
-                      radioID,
-                      val_to_str_const(DIS_ENCODING_TYPE(encodingScheme),
DIS_PDU_Encoding_Type_Strings, "Unknown Encoding Type"),
-                      numSamples
-                      );
+        if (numSamples)
+            col_prepend_fstr(pinfo->cinfo, COL_INFO, ", Number of Samples=%u",
+                numSamples);
+
+        if ((encodingScheme & 0xC000) >> 14 == DIS_ENCODING_CLASS_ENCODED_AUDIO)
+            col_prepend_fstr(pinfo->cinfo, COL_INFO,", Encoding Type=%s",
+                val_to_str_const(DIS_ENCODING_TYPE(encodingScheme),
+                DIS_PDU_Encoding_Type_Strings, "Unknown"));
+
+        col_prepend_fstr( pinfo->cinfo, COL_INFO,
+                      "PDUType: %s, RadioID=%u", pduString, radioID);
         break;
     case DIS_PDUTYPE_TRANSMITTER:
         col_add_fstr( pinfo->cinfo, COL_INFO,
@@ -593,6 +609,11 @@ void proto_register_dis(void)
             },
             { &hf_dis_ens_type,
               { "Encoding Type", "dis.radio.encoding_type",
+                FT_UINT16, BASE_DEC, NULL, 0x3fff,
+                NULL, HFILL }
+            },
+            { &hf_dis_ens_type_audio,
+              { "Encoding Type",  "dis.radio.encoding_type.audio",
                 FT_UINT16, BASE_DEC, VALS(DIS_PDU_Encoding_Type_Strings), 0x3fff,
                 NULL, HFILL }
             },
@@ -791,6 +812,35 @@ void proto_register_dis(void)
                FT_BYTES, BASE_NONE, NULL, 0x0,
                NULL, HFILL}
             },
+            { &hf_dis_signal_link16_npg,
+              { "NPG Number", "dis.signal.link16.npg",
```

```
+                 FT_UINT16, BASE_DEC, VALS(Link16_NPG_Strings), 0x0,
+                 NULL, HFILL }
+            },
+            { &hf_dis_signal_link16_tsec_cvll,
+              { "TSEC CVLL", "dis.signal.link16.tsec_cvll",
+                 FT_UINT8, BASE_RANGE_STRING | BASE_DEC,
RVALS(DIS_PDU_Link16_CVLL_Strings), 0x0,
+                 NULL, HFILL }
+            },
+            { &hf_dis_signal_link16_msec_cvll,
+              { "MSEC CVLL", "dis.signal.link16.msec_cvll",
+                 FT_UINT8, BASE_RANGE_STRING | BASE_DEC,
RVALS(DIS_PDU_Link16_CVLL_Strings), 0x0,
+                 NULL, HFILL }
+            },
+            { &hf_dis_signal_link16_message_type,
+              { "Message Type", "dis.signal.link16.message_type",
+                 FT_UINT8, BASE_DEC, VALS(DIS_PDU_Link16_MessageType_Strings), 0x0,
+                 NULL, HFILL }
+            },
+            { &hf_dis_signal_link16_ptt,
+              { "Perceived Transmit Time", "dis.signal.link16.ptt",
+                FT_ABSOLUTE_TIME, ABSOLUTE_TIME_UTC, NULL, 0x0,
+                NULL, HFILL }
+            },
+            { &hf_dis_signal_link16_stn,
+              { "Source Track Number", "dis.signal.link16.stn", FT_UINT16, BASE_OCT, NULL,
0x0,
+                 NULL, HFILL }
+            },
             { &hf_dis_num_shafts,
               { "Number of Shafts",  "dis.ua.number_of_shafts",
                FT_UINT8, BASE_DEC, NULL, 0x0,
@@ -816,7 +866,9 @@ void proto_register_dis(void)
         &ett_dis_po_header,
         &ett_dis_ens,
         &ett_dis_crypto_key,
-        &ett_dis_payload
+        &ett_dis_payload,
+        &ett_dis_signal_link16_network_header,
+        &ett_dis_signal_link16_message_data,
     };

     module_t *dis_module;
--
1.8.1.2
```

Figure A2: Software patch (part 2 of 3)

```
From 6ef9c448729ba0270a4f2552b817d1ceb4fc4de7 Mon Sep 17 00:00:00 2001
From: Peter Ross <peter.ross@dsto.defence.gov.au>
Date: Wed, 16 Oct 2013 10:00:03 +1100
Subject: [PATCH 3/3] packet-dis: correct 'Terminal Primary Mode' and 'Sync
 State' enumeration values

SISO-STD-002 Standard for Link 16 Simulations, June 2006:
http://www.sisostds.org/DigitalLibrary.aspx?Command=Core_Download&EntryId=30265
---
 epan/dissectors/packet-dis-enums.c | 9 ++++-----
 1 file changed, 4 insertions(+), 5 deletions(-)

diff --git a/epan/dissectors/packet-dis-enums.c b/epan/dissectors/packet-dis-enums.c
index d53b3a7..835bb17 100644
--- a/epan/dissectors/packet-dis-enums.c
+++ b/epan/dissectors/packet-dis-enums.c
@@ -668,8 +668,8 @@ const value_string DIS_PDU_TSAllocationFidelity_Strings[] =

 const value_string DIS_PDU_TerminalPrimaryMode_Strings[] =
 {
```

```
-    {0,    "NTR" },
-    {1,    "JTIDS Unit Participant" },
+    {1,    "NTR" },
+    {2,    "JTIDS Unit Participant" },
     {0,    NULL }
 };

@@ -685,9 +685,8 @@ const value_string DIS_PDU_TerminalSecondaryMode_Strings[] =
 /* http://discussions.sisostds.org/threadview.aspx?fid=18&threadid=53172 */
 const value_string DIS_PDU_ModParamSyncState_Strings[] =
 {
-    {0,    "Undefined" },
-    {1,    "Coarse Synchronization" },
-    {1,    "Fine Synchronization" },
+    {2,    "Coarse Synchronization" },
+    {3,    "Fine Synchronization" },
     {0,    NULL }
 };

--
1.8.1.2
```

Figure A3: Software patch (part 3 of 3)

DEFENCE SCIENCE AND TECHNOLOGY ORGANISATION
DOCUMENT CONTROL DATA

1. PRIVACY MARKING/CAVEAT (OF DOCUMENT)

2. TITLE

Extending the Wireshark Network Protocol Analyser to Decode Link 16 Tactical Data Link Messages

3. SECURITY CLASSIFICATION (FOR UNCLASSIFIED REPORTS THAT ARE LIMITED RELEASE USE (L) NEXT TO DOCUMENT CLASSIFICATION)

Document	(U)
Title	(U)
Abstract	(U)

4. AUTHOR(S)

William Robertson and Peter Ross

5. CORPORATE AUTHOR

DSTO Defence Science and Technology Organisation
506 Lorimer St
Fishermans Bend Victoria 3207 Australia

6a. DSTO NUMBER	6b. AR NUMBER	6c. TYPE OF REPORT	7. DOCUMENT DATE
DSTO-TN-1257	AR-015-847	Technical Note	January 2014

8. FILE NUMBER	9. TASK NUMBER	10. TASK SPONSOR	11. NO. OF PAGES	12. NO. OF REFERENCES
2013/1123978/1	N/A	CAD	32	20

13. URL on the World Wide Web

http://dspace.dsto.defence.gov.au/dspace/

14. RELEASE AUTHORITY

Chief, Aerospace Division

15. SECONDARY RELEASE STATEMENT OF THIS DOCUMENT

Approved for public release

OVERSEAS ENQUIRIES OUTSIDE STATED LIMITATIONS SHOULD BE REFERRED THROUGH DOCUMENT EXCHANGE, PO BOX 1500, EDINBURGH, SA 5111

16. DELIBERATE ANNOUNCEMENT

No Limitations

17. CITATION IN OTHER DOCUMENTS Yes

18. DSTO RESEARCH LIBRARY THESAURUS http://web-vic.dsto.defence.gov.au/workareas/library/resources/dsto_thesaurus.shtml

Network analysis, Network protocols, Simulation, Software tools, Tactical data links, Testing.

19. ABSTRACT

This technical note describes the development of a tactical data link message dissector for the Wireshark network protocol analyser. Link 16 is a United States and North Atlantic Treaty Organization standard for secure real-time exchange of tactical information between warfighting units. Concurrent with military adoption of Link 16 equipment, training simulators are being fitted with simulated tactical data links. The extensions made to Wireshark provide simulation engineers with a tool to troubleshoot Link 16 simulations.