United Electronic Industries

The High-Performance Alternative

DNA/DNR-ARINC-708/453 Communications Interface with 2TX and 2RX Channels

—

User Manual

Dual-channel, ARINC-708/453 Communications Interface
for the PowerDNA Cube and RACKtangle Chassis
ARINC-708 and ARINC-453 Compatible

January 2011 Edition

PN Man-DNx-ARINC-708/453-0111

Version 1.0

Contacting United Electronic Industries

Mailing Address:

27 Renmar Ave.
Walpole, MA 02081
U.S.A.

For a list of our distributors and partners in the US and around the world, please see http://www.ueidaq.com/partners/

Support:

Telephone:(508) 921-4600
Fax:(508) 668-2350

Also see the FAQs and online “Live Help” feature on our web site.

Internet Support:

Support:support@ueidaq.com
Web-Site:www.ueidaq.com
FTP Site:ftp://ftp.ueidaq.com

Product Disclaimer:

WARNING!

DO NOT USE PRODUCTS SOLD BY UNITED ELECTRONIC INDUSTRIES, INC. AS CRITICAL COMPONENTS IN LIFE SUPPORT DEVICES OR SYSTEMS.

Products sold by United Electronic Industries, Inc. are not authorized for use as critical components in life support devices or systems. A critical component is any component of a life support device or system whose failure to perform can be reasonably expected to cause the failure of the life support device or system, or to affect its safety or effectiveness. Any attempt to purchase any United Electronic Industries, Inc. product for that purpose is null and void and United Electronic Industries Inc. accepts no liability whatsoever in contract, tort, or otherwise whether or not resulting from our or our employees' negligence or failure to detect an improper purchase.

NOTE: Specifications in this document are subject to change without notice. Check with UEI for current status.

Table of Contents

List of Figures

Chapter 1 Introduction

This document outlines the feature-set and describes the operation of the DNx-ARINC-708/453 Communication Interface boards. The DNA- version is designed for use with a PowerDNA Cube data acquisition system. The DNR-version is designed for use with a DNR-12 RACKtangle or DNR-6 HalfRACK rack-mounted systems. Both versions handle ARINC 708 messaging over a dual redundant MIL-1553 bus. Please ensure that you have the PowerDNA Software Suite installed before attempting to run examples.

1.1 Organization of this manual

The DNx-ARINC-708/453 User Manual is organized as follows:

- **Introduction**
  This chapter provides an overview of the document content, the device architecture, connectivity, and logic of the layer. It also includes connector pinout, notes, and specifications.
- **Programming with the High-Level API**
  This chapter describes the use of the UeiDaq Framework High-Level API for programming the board. It includes information such as how to create a session, configure the session for ARINC 708 bus communication, and interpret results.
- **Programming with the Low-Level API**
  This chapter describes the use of low-level API commands for configuring and using the DNx-ARINC-708/453 series boards.
- **Appendix A: Accessories**
  This appendix provides a list of accessories available for DNx-ARINC-708/453 board(s).
- **Index**
  This is an alphabetical listing of the topics covered in this manual.

Document Conventions

To help you get the most out of this manual and our products, please note that we use the following conventions:

Tips are designed to highlight quick ways to get the job done, or to reveal good ideas you might not discover on your own.

NOTE: Notes alert you to important information.

CAUTION! *Caution advises you of precautions to take to avoid injury, data loss, and damage to your boards or a system crash.*

Text formatted in **bold** typeface generally represents text that should be entered verbatim. For instance, it can represent a command, as in the following example: “You can instruct users how to run setup using a command such as **setup.exe**.”

Frequently Asked Questions

For frequently answered questions, application notes, and support, visit us online at:

http://www.ueidaq.com/faq/

1.2 The 708/453 Interface Boards

The DNx-ARINC-708/453 interface boards have the following features:

- 2 independent channels/ports
- Dual redundant bus interfaces
- Each channel is independently software-configurable as a Bus Monitor.
- Transformer-coupled Bus Interface standard, (Direct coupling software selectable)
- Supports 1553A and 1553B protocols (Notice 1 and/or 2)
- Completely independent bit rate settings for each port
- 350 Vrms isolation between 1553 bus, other I/O ports, and chassis

[DNR-708-453 Shown, includes cable and loop-back self-test adaptor included]

DNA version is functionally identical but with different bus connector and different front panel

Figure 1-1. Photo of DNR-ARINC-708/453 Interface Module

1.3 What is ARINC-708?

The ARINC 708 is a derivative of MIL-STD 1553 technology, developed specifically for use with airborne weather radar systems. It is normally used to display the output from the radar on the radar weather display. The bus uses 2-wires, is simplex, Manchester encoded, and runs at a one-megabit data rate. It was originally based upon a simple derivative of MIL-STD-1553 technology and is compliant with the ARINC 708 standard. The data words are 1600 bits long and are composed of a single 64-bit status word plus 512 3-bit data words (1600 bits total).

Each 1600-bit data word is used to write 1600 pixels across the screen of a weather display. Each data word writes one row of pixels on the radar display screens on successive lines, thus displaying a complete picture of current weather conditions in the vicinity of the radar unit.. Duplicate screens are usually provided, one for the pilot, and a second for the copilot.

1.4 What is MIL-STD-1553?

MIL-STD-1553 is a military standard that defines mechanical, electrical, and operating characteristics of a serial data communication bus for the U.S. Department of Defense.It is now commonly used for both military and civilian applications in avionics, aircraft, and spacecraft data handling. A 1553 system typically uses a dual redundant, balanced-line, physical layer with a differential network interface with time division multiplexing, half-duplex, command/ response data communication protocol with up to 31 remote terminal devices. It was first used in the F-16 fighter aircraft and is now widely used by all branches of the U.S. military and NATO.

The current standard, MIL-STD-1553B was introduced in 1978, the goal of which was to define explicitly how each option should function, so that compatibility among manufacturers could be guaranteed.

1.4.1 Physical Layer

A single 1553 bus consists of a shielded twisted-wire pair with 70 - 85 ohm impedance at 1 MHz. If a coaxial connector is used, the center pin is used for the high Manchester bi-phase signal. All transmitter and receiver devices connect to the bus either through coupling transformers or directly through stub connectors and isolation transformers, as shown in **Figure 1-2**. Stubs can be a maximum of 1 foot in length for direct coupling or a maximum of 20 feet for transformer coupling. (The type of coupling, direct or transformer, is determined by the wiring connections to the transformer.) To limit reflections, the data bus must be terminated by resistors equal to the cable characteristic impedance (within ±2%). **Figure 1-2** shows one of the two buses. Each transceiver is also connected in the same way to the second (redundant) bus. Note that although a 1553 system may also have additional redundant buses, the DNx-ARINC-708/ 453 interface module is designed for systems with two buses only.

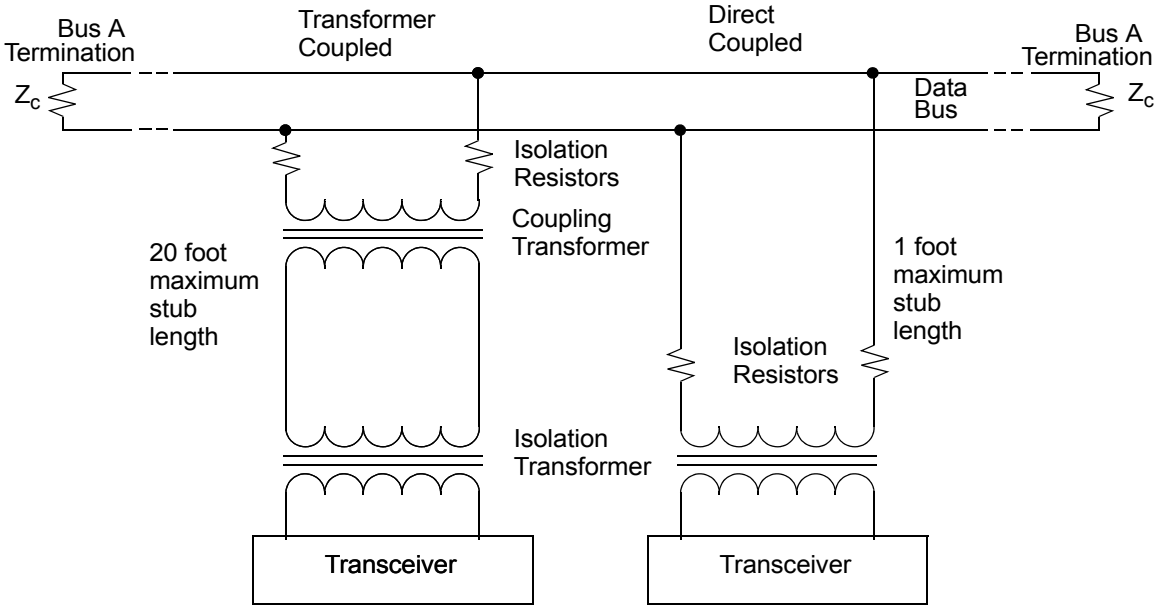

Figure 1-2. Terminal Connection Types (One Bus Shown)

Although multi-stub couplers may be used in place of individual couplers according to the standard, this does not apply when the DNx-MIL-1553 board is used because the DNx-ARINC-708/453 interface module is designed with software-selectable direct or transformer coupling and has termination resistors built in. The data bus, therefore can be connected directly to the I/O connector of the interface module.

1.5 Bus Protocol

All ARINC 708 messages contain one 1600-bit word, composed of one 64-bit status word followed by 512 3-bit data words indicating weather intensity in each range bin along an azimuth line at each scan angle. The words in a message are transmitted with no gap between words, but a 4 μs gap is inserted between successive messages. All devices must start transmitting a response to a command within 4 to 12 μs. If they do not start transmitting within 14 μs, they are considered to have not received the command message.

1.6 ARINC 708 Word Formats

ARINC 708 is designed for use with airborne pulse Doppler weather radar display systems. ARINC 708 data is transmitted over the display data bus from the radar transmitter/receiver to the radar weather display units for the pilot and copilot.

ARINC 708 uses a transformer-coupled Manchester encoded signal with a 1MHz bit-rate (same as the MIL-STD-1553 protocol).

In an ARINC 708 weather display system, all data frames (words) are 1600 bits long (100 16-bit data words). Each word contains both header information and data information. Each frame starts with a unique 3us sync pattern (1.5us high, 1.5us low) and ends with another 3us sync pattern (1.5us low, 1.5us high). Therefore, a complete frame takes 1606us on the bus. Each 1600-bit frame writes one line of pixels on the display screen on successive lines. Each group of frames, therefore, displays a color picture of the current weather pattern in the vicinity of the radar sensor.

Header

The header portion consists of a 64-bit status word. The table below shows what this word contains:

Table 1-1. Header Portion (Bits 1-64)

Bits	Function	Status	Description
1-8	Label	1	Always octal 055 (binary 10110100)
9-10	Control Accept		See **Table 1-2** below.
11	Slave		0=Master (normal), 1=Slave
12-13	Spare		
14-18	Mode Annunciation		See **Table 1-3** below. 0 indicates normal, 1 indicates condition listed.
19-25	Faults		See **Table 1-4** below. 0 indicates normal, 1 indicates condition listed below
26	Stabilization		0=OFF, 1=ON
27-29	Operating Mode		See **Table 1-5** below.

Table 1-1. Header Portion (Bits 1-64)

Bits	Function	Status	Description
30-36	Tilt		See **Table 1-6** below.(indicates a 2s complement value for tilt in degrees.
37-42	Gain		See **Table 1-7** below. 0=OFF, 1=ON
43-48	Range		See **Table 1-8** below. Value indicate range from zero in nautical miles.
49	Spare		
50-51	Data Accept		See **Table 1-9** below.
52-63	Scan Angle		See **Table 1-10** below.
64	Spare		

The following tables define the content of each bit group:

.

Table 1-2. Control Accept Bits (Bits 9-10)

Bit 10	Bit 9	Control Accept
0	0	Do not accept control
0	1	IND1 accept control
1	0	IND2 accept control
1	1	All INDs accept control

.

Table 1-3. Mode Annunciation Bits (Bits 14-18)

Bit	Mode Annunciation
14	Automatic sensing of a **turbulence alert** has occurred.
15	Automatic sensing of a **reflectivity weather alert** has occurred.
16	**Clutter elimination** circuitry is in operation.
17	Reduced **sector scan** is in operation.
28	Aircraft attitude and/or tilt exceeds system design limits.

Bits 14-18 are treated as discrete bits, where 0 indicates normal and 1 indicates the condition listed above.

.

Table 1-4. Fault Bits (Bits 19-25)

Bit	Fault
19	Cooling Fault
20	Display Fault
21	Calibration Fault
22	Altitude input Fault
23	Control Fault
24	Antenna Fault
25	Transmitter/receiver Fault

Bits 19-25 are treated as discrete bits, where 0 indicates normal and 1 indicates the fault listed above.

.

Table 1-5. Operating Mode (Bits 27-29)

Bit 29	Bit 28	Bit 27	Operating Mode
0	0	0	Standby
0	0	1	Weather (only)
0	1	0	Map
0	1	1	Contour
1	0	0	Test
1	0	1	Turbulence (only)
1	1	0	Weather and Turbulence
1	1	1	Reserved (calibration annunciation)

.

Table 1-6. Tilt (Bits 30-36)

Bit	Tilt (in Degrees)
36	-16
35	+8
34	+4
33	+2
32	+1
32	+0.5
30	+0.25

.

Table 1-7. Gain (Bits 37-42)

Bit 42	Bit 41	Bit 40	Bit 39	Bit 38	Bit 37	Gain)
1	1	1	1	1	1	Cal
0	0	0	0	0	0	Max
0	0	0	1	1	1	-5
0	0	1	0	1	1	-11
1	1	1	1	1	0	-62

.

Table 1-8. Range (Bits 43-48)

Bit 48	Bit 47	Bit 46	Bit 45	Bit 44	Bit 43	Range in Nautical Miles
0	0	0	0	0	0	5
0	0	0	0	1	0	10
0	0	0	1	0	0	20
0	0	1	0	0	0	40

Table 1-8. Range (Bits 43-48) (Cont.)

Bit 48	Bit 47	Bit 46	Bit 45	Bit 44	Bit 43	Range in Nautical Miles
0	1	0	0	0	0	80
1	0	0	0	0	0	160
1	1	1	1	1	1	315
0	0	0	0	0	0	320

.

Table 1-9. Data Accept (Biits 50-51)

Bit 51	Bit 50	Data Accept
0	0	Do not accept data
0	1	Accept data 1
1	0	Accept data 2
1	1	Accept any data

Scan Angle (Bits 52-63)

Bits 52-63 represent the Scan Angle in degrees as a numerical value. Bit 63 is the MSB with a value of 180 degrees. Bit 52 is the LSB with a value of 0.087890625 degrees.

.

Table 1-10. Range (Bits 43-48)

Bit 48	Bit 47	Bit 46	Weather Condition/Reflectivity	Color Example
0	0	0	No precipitation (<Z2)	Black
0	0	0	Light precipitation (Z2 to Z3)	Green
0	0	0	Moderate precipitation (Z3 to Z4)	Yellow
0	0	1	Heavy precipitation (Z4 to Z5)	Red
0	1	0	Very heavy precipitation (>Z5)	Magenta
1	0	0	Reserved (out of calibration indication)	
1	1	1	Medium turbulence	
0	0	0	Heavy turbulence	

Each data frame represents a single radius line emanating from the center of a circle. The color values in the range bins are displayed on this radius line to show the weather conditions in the surrounding area..

Each word type has a specific format within a common structure.The first three bits are a synchronization field, which enables the decoding clock to re-sync at the beginning of each new word.

All bit encoding is based on bi-phase Manchester II format, which provides a self-clocking waveform. The signal is symmetrical about zero and is therefore compatible with transformer coupling.

In Manchester coding, signal transitions occur only at the center of a bit time. A logic “0” is defined as a transition from negative to positive level; a logic “1” is the reverse. Note that the voltage levels on the bus are ***not*** the information signal; all information is contained in the timing and direction of the ***zero crossings*** of the signal on the bus.

The terminal hardware provides the encoding and decoding of the various word types. The encoder also calculates parity. For received messages, the decoder signals the logic what sync type a word is and whether or not parity is valid. For transmitted messages, input to the encoder defines what sync type to place at the beginning of a word. The encoder calculates parity automatically for each word.

1.6.2 Bus Monitor (MT)

When used in a 1553 system, a Bus Monitor is not able to transmit messages on the bus; its function is to monitor and record messages being transmitted on the bus without disrupting other devices. When used in an ARINC 708 system, however, a Bus Monitor can transmit or receive data on either or both buses.

1.7 DNx-ARINC-708/453 Architecture

Figure 1-3 is a block diagram of the architecture of the DNx-ARINC-708/453 interface module.

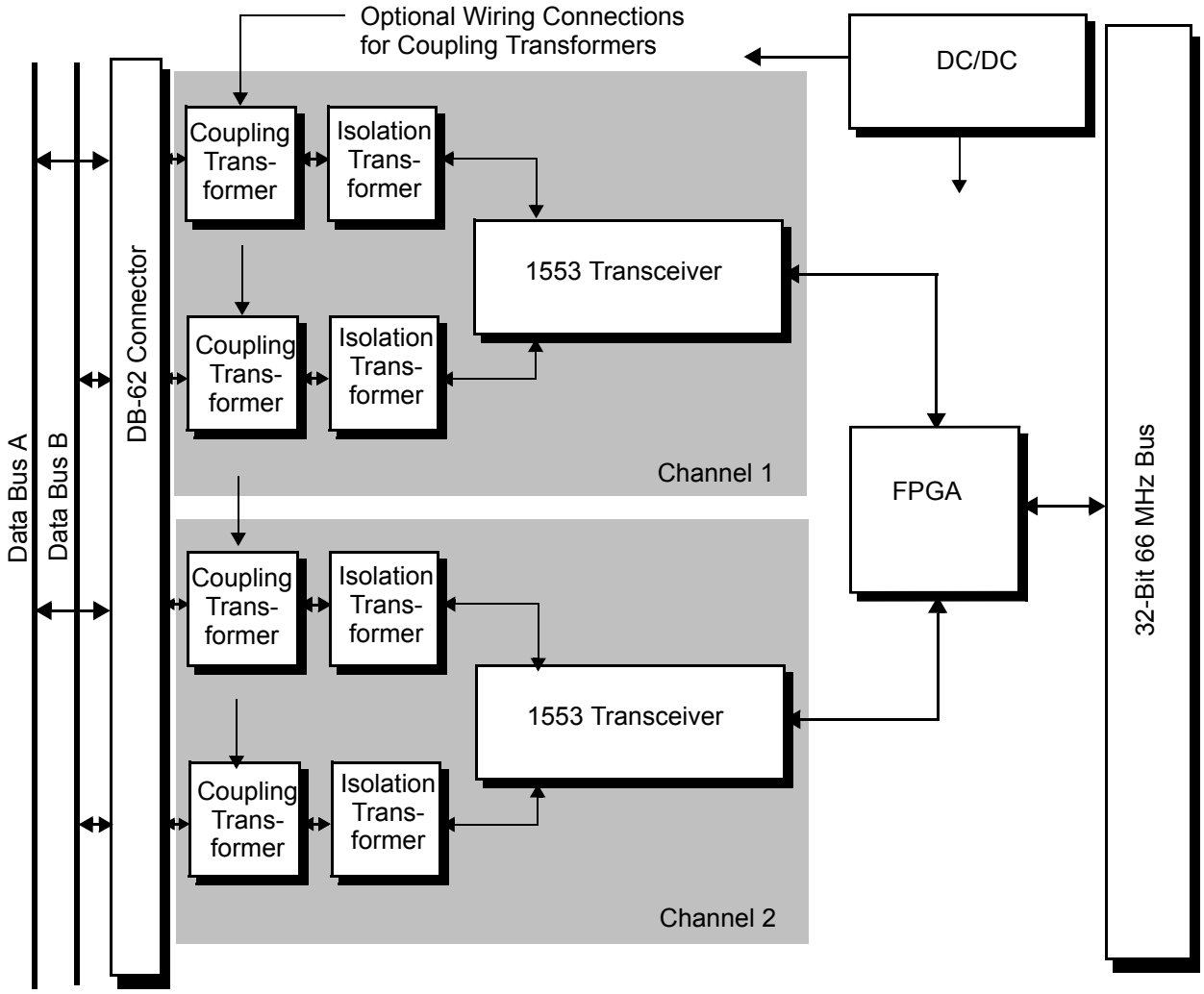

Figure 1-3. Block Diagram of the DNx-ARINC-708/453 Interface Module

1.7.3 Functional Description

As shown in **Figure 1-3**, the module has two independent channels, each connected to a dual-redundant data bus via coupling/isolation transformers or isolation transformers only, and each with an independent dedicated transceiver. The coupling transformers and the transformer taps can be selected by opto-isolated relays under program control. (Note that different transformer ratios are used for direct or transformer coupling.)

When the channel is operating as a transmitter, messages are sent from the host through the DNx-ARINC-708/453 module to the selected Data Bus. The address contained in the message itself determines the destination of the message. The transceiver transmitter accepts Manchester-encoded biphase data and converts it to differential voltages that are passed to the bus through the isolation and coupling transformers (or through an isolation transformer circuit only, if direct coupling is selected).

When the channel is operating as a receiver, the process described above is reversed.

1.8 Specifications

Technical Specifications:

General Specifications		
Number channels/ports	2 channels, each with two ports	
Channel configuration	Port A and Port B. Channels may receive data from either Port A or Port B on a channel, but not both simultaneously. Channels may transmit on Port A, or Port B, or both. However, if transmission is set for both, then both ports transmit identical data.	
Specification compliance	ARINC 708/453	
Configuration	Standard MIL-1553 based signal levels, fully compliant with ARINC-708 and 453	
Interface (software selectable) [measured at connector]	Transformer: 18-27 Vpp into 70 Ω load	Direct Coupling: 6-9 Vpp into 35 Ω load
Isolation	350 Vrms	
Power Consumption	5 W (not including load)	
FIFO and Data TX/RX Specs		
Configuration	Fully buffered FIFO connections	
FIFO size	20 packets (each packet contains the standard 1600 bit/100 word frame.)	
Environmental		
Operating Temp. (tested)	-40°C to +85°C	
Operating Humidity	0 - 95%, non-condensing	
MTBF	275,000 hours	
Vibration *IEC 60068-2-6* *IEC 60068-2-64*	5 g, 10-500 Hz, sinusoidal 5 g (rms), 10-500 Hz, broad-band random	
Shock *IEC 60068-2-27*	50 g, 3 ms half sine, 18 shocks @ 6 orientations 30 g, 11 ms half sine, 18 shocks @ 6 orientations	
Altitude	0 - 70,000 feet (0 - 21,336 m)	

Figure 1-4. DNx-ARINC-708/453 Specifications

1.9 Software

Refer to Chapters 2 and 3 for information on how to program the DNx-ARINC-708/453 Interface Module.

UEIDAQ Framework is a programming facility that allows you to connect transparently to a cube or RACKtangle and then configure the DNx-ARINC-708/453 module to communicate with otherARINC 708 devices. Attach the IOM to the ARINC 708 bus(es) and then send and/or receive ARINC 708 packets.

1.10 Wiring & Connectors

Figure 1-5 illustrates the pinout of the DB-62 connector on the DNx-ARINC-708/453 Interface Module.

Pinout Diagram:

The DNx-708-453 provides connections via a 62-pin "D" connector. A one foot, 62-pin to (quad connector) cable is also included which provides connection to standard MIL-STD-1553 style connectors.

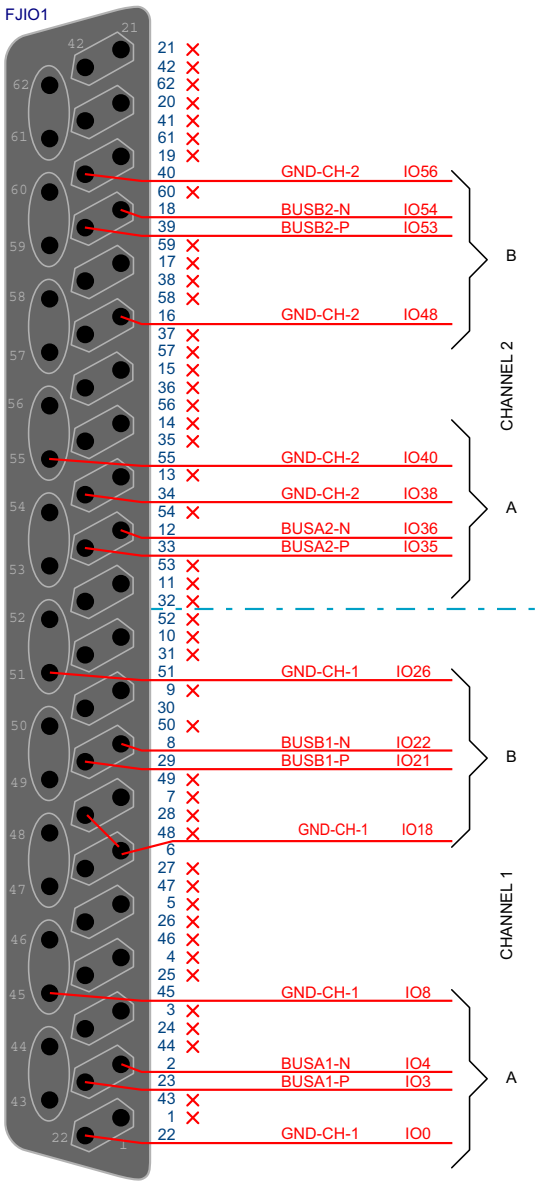

Figure 1-5. Pinout Diagram for DNx-ARINC-708/453Layer

Chapter 2 Programming with the High Level API

This section describes how to program a DNx-ARINC-708/453 layer using the UeiDaq Framework High-Level API. UeiDaq Framework is object-oriented and its objects can be manipulated in the same manner in various development environments, such as Visual C++. Visual Basic, LABView, or DASYLab.

UeiDaq Framework is bundled with examples for supported programming languages. They are located under the UEI Programs group in

Start >> Programs >> UEI >> Framework >> Examples

The following subsections focus on the C++ API, but the concept is the same regardless of the programming language used.

Please refer to the ”UeiDaq Framework User Manual” for more information on using other programming languages.

2.1 Creating a Session

The Session object controls all operations on your PowerDNA device. Therefore, the first task is to create a session object:

```
CUeiSession session;
```

2.2 Create MIL-1553 Ports

MIL-1553 ports are configured using the session object’s method ***“CreateMIL1553Port”,*** as follows:

```
// port 0 - bus monitor
CUeiMIL1553Port* pPort0 =
session.CreateMIL1553Port(
“pdna://192.168.100.2/dev0/milb0“,
UeiMIL1553CouplingTransformer,
UeiMIL1553OpModeBusMonitor
                );

// port 1 - remote terminal
CUeiMIL1553Port* pPort1 =
session.CreateMIL1553Port(
“pdna://192.168.100.2/dev0/milb1“,
UeiMIL1553CouplingTransformer,
UeiMIL1553OpModeBusMonitor
);
```

Each created port can be used in one of three modes):

- ***UeiMIL1553OpModeBusMonitor*** – a Bus Monitor port allows you to receive ongoing activity on the bus using a ***CUeiMIL1553Reader*** object. In this mode of operation, the ***CUeiMIL1553Writer*** object also allows you to send unscheduled continuous data on the bus.
- In ***UeiMIL1553OpModeBusMonitor***, a user can monitor a bus and send messages using ***BusWriter***. Bus monitor is enabled in all configurations.
- Note that Bus Monitor works in all modes.

Each port created can be used in one of four coupling modes:

- ***UeiMIL1553CouplingDisconnected*** – port is completely disconnected from the bus.
- ***UeiMIL1553CouplingTransformer*** – normal mode of operation.
- ***UeiMIL1553CouplingLocalStub*** – isolation coupler of the layer, which requires a special version of the hardware.
- ***UeiMIL1553CouplingDirect*** – direct connection without isolation transformer. Sometimes used in laboratories when a coupled network is not available. This mode is normally not recommended.

The normal coupling is ***UeiMIL1553CouplingTransformer***.

The ***UeiMIL1553CouplingLocalStub*** coupling option requires a local stub to be populated on the 1553 board (this is an extra cost option). Direct coupling is used sometimes a in laboratory environment but is extremely rare, probably in cases in which there is no network.

Note that you will need to create one reader and one writer per port to access port data in any mode of operation.

A user can select which bus on which to transmit data using the ***CUeiMIL1553Port::SetTxBus()*** method.

Note that for transmission of messages, either bus A or bus B should be selected (default is bus A).

CUeiMIL1553Port::SetRxBus() selects which bus to listen to. Table 2-1 on page 17 describes the bus settings that are allowed for various modes of operation:

Table 2-1. Selection of Transmit Bus

Settings	BM (BW)	
	Listen (Rx)	Transmit (Tx)
Rx A	A only	
Rx B	B only	
Rx Both	A and B (normal)	
Tx A		A only
Tx B		B only
Tx Both		prohibited

2.3 Configure Timing

On MIL-1553 ports, messages are represented in a ***tUeiMIL1553*Frame*** structures. Note that the same approach is used for all modes of operation. Bus monitor timing depends on the frame type specified.

```
session.ConfigureTimingForMessagingIO(1, 0);
session.GetTiming()->SetTimeout(1000);
```

Asynchronous operations with MIL-708/453 layers are currently not implemented. If a user needs to operate BM asynchronously, the best way is to use it in a separate thread.

2.4 Creating a Reader Object and Writer Object for each Port

Before a user can communicate with the layer reader and (for everything except a BM), writer objects need to be created for each port, as shown below:

```
CUeiMIL1553Reader* readers = new
CUeiMIL1553Reader(session.
GetDataStream(), session.GetChannel(ch)-
>GetIndex());
CUeiMIL1553Writer* writers[ch] = new
CUeiMIL1553Writer(session.
GetDataStream(), session.GetChannel(ch)-
>GetIndex());
```

By doing this, you are creating MIL-708/453 specific Reader and Writer classes and connecting them to the appropriate data stream and channel.

The default behavior of reader and writer objects is to block until the specified number of frames is ready to be transferred. You can also configure those objects to work asynchronously. The method used to program readers and writers asynchronously is very dependent on the programming language. You can find more information on how to do this in the Reference manual for each development environment.

CUeiMIL1553Reader and ***CUeiMIL1553Writer*** are polymorphic. There are multiple overloaded implementations of these functions that can accept different types of frames. The type of frame dictates the data passed and the operation to be performed.

The following types of frames are defined:

CUeiMIL1553BMFrame class – read BM messages separated by idle state of the bus.

This class is used to construct and manipulate a ***CUeiMIL1553BMFrame*** to simplify its use. It can be used only with ***CUeiMIL1553Reader::read()***.

```
class CUeiMIL1553BMFrame : public
tUeiMIL1553BMFrame
```

- ***CUeiMIL1553BMCmdFrame*** class – read BM messages separated by 1553 protocol commands.

This class is used to construct and manipulate ***CUeiMIL1553BMCmdFrame*** to simplify its use. It can be used only with ***CUeiMIL1553Reader::read().***

```
class CUeiMIL1553BMCmdFrame : public
tUeiMIL1553BMCmdFrame
```

- ***CUeiMIL1553TxFifoFrame*** class – write data to BusWriter.

This class is used to construct and manipulate ***CUeiMIL1553TxFifoFrame*** to simplify its use. It can be used only with ***CUeiMIL1553Writer::write()***.

```
class CUeiMIL1553TxFifoFrame : public
tUeiMIL1553TxFifoFrame
```

Frame type ***UeiMIL1553FrameTypeBusMon*** is used to receive data from the bus monitor. Each command and status word on the bus is stored in a separate frame.

Call the session object's method **"ConfigureTimingForMessagingIO"** to perform message communication, provided that the device allows it.

2.5 Starting the Session

You can start the session by calling the session object's method "Start":

```
// Start the session
mySession.Start();
```

Note that if you don't explicitly start the session, it will be automatically started the first time you try to transfer data using a reader or writer object.

2.6 Reading/ Writing Data from/to a Device

To write or read data to/from a MIL-708/453 board, do the following:

```
CUeiMIL1553RTDataFrame* pFrame = new
CUeiMIL1553RTDataFrame;
writer->Write(numFramestoWrite, pFrame,
&numFramesWritten);
reader->Read(numFramestoRead, pFrame,
&numFramesRead);
```

Note that the first read or write to/from a 1553 channel configures all operations and starts the layer.

2.6.1 Reading Bus Monitor

Bus Monitor is the simplest function to use.

To do so, first create a new bus monitor frame:

```
CUeiMIL1553BMFrame* bmFrm = new CUeiMIL1553BMFrame;
```

Then read bus monitor data from that accumulated in the 1553 bus monitor buffer (it can accumulate up to 1024 32-bit data words).

```
readers[1]->Read(1, bmFrm, &numFramesRead);
```

You can either work with the frame members directly or use helper methods to display data as strings:

```
if (numFramesRead) std::cout << bmFrm-
>GetBmDataStr() << std::endl;
```

Raw bus monitor data is represented as follows:

First 32-bit word: —

bit 31: parity error on the bus, if any
bit 30: set to 1 for command or status
bits 29 thru 16: time in 15.15ns interval since previous command or status.
bits 15 thru 0: command or status as received from the bus.

If there is data following the command, it is represented in the following format:

bit 31: parity error on the bus, if any
bit 30: set to 0 for data word
bits 29 thru 16: time in 15.15ns interval since previous command or status.
bits 15 thru 0: data as received from the bus.

If timestamps are enabled using the ***`CUeiMIL1553Port::EnableTimestamping()`*** method, (timestamps are enabled by default), the last two words contain a 32 bit “absolute” timestamp of the message in 10us resolution (timestamps are reset when the session starts) and the various flags defining the current bus status and which bus (A or B) the command was received on. See “PowerDNA API Reference Manual” for further detail.

A Bus Monitor works as follows:

Each time a command/status word is decoded on the 1553 A/B bus, it is validated against an RT filter. If the RT address is not included in the monitoring, all command/status and consequential data are ignored. If the RT address is included into the monitoring, the timestamp is stored into the internal register and the command/status with gap timeout counter is stored into the FIFO. After that, each received data word is stored into the FIFO until the next command/status word is received and the process repeats itself. Once a data gap interval is detected OR another control/status word is received, the following optional information is stored into the BM FIFO:

- Timestamp word (30 LSBs of the timestamp) – optional
- Flags/status word (status information and extra bits of the timestamp if enabled)

- Bus IDLE tag, 0xC0000000, which indicates that the 1553 bus that was driving the BM went into the idle state. The idle tag is used internally in the firmware to separate messages and is not exposed in the datastream into ***CUeiMIL1553BMFrame***.
- Note that when BMALL bit is cleared (normal operation), the BM follows messages from bus A or B, but only one bus at the time. If for any reason, in violation of the 1553 protocol, any device sends data on both buses, only the transmission from one bus will be logged, data from the other bus will be processed upon the “first” bus going into the idle state; in the case in which more than one word was received prior to switching to the “second” bus, the data from the “second” bus will be corrupted.

BM data structure:

Command/Status, all 32-bits used, bit 30=1
Data word 0
Data word N
Optional: Timestamp[29:0] , bit 30=0
Optional: Flags[29:0] , bit 30=0

Command/Status/Data format in BM data

Bit	Name	Description
31	PARITY	One in this bit indicates parity error
30	WORD_TYPE	Word type (1=command/status, 0=data)
29-16	GAP1553	Gap interval on 1553 bus measured in 66MHz clocks, 248.2uS max; 0x3FFF indicates that gap counter has expired
15-0	DATA1553	1553 data from/to the decoders

Timestamp “LSB part” word format in BM data

Bit	Name	Description
31-30	ZERO	Upper two bits are always zeroes for the flags
29-0	TIME_LSB	30 LSBs of the timestamp tag

Flag word format in BM data

Bit	Name	Description
31-30	ZERO	Upper two bits are always zeroes for the flags
29	RES29	Reserved for future use
24	RES24	Reserved for future use
23	OVRE	=1 if decoder data overrun was detected
22	PE	=1 if parity error was detected
21	DBE	=1 if error detected during data bits reception
20	SBE	=1 if error detected during SYNC bit reception
19	ZCE	=1 if invalid combination is detected (positive line <> !negative line) ~ 150nS after zero crossing
18	SET	=1 if edge-edge timing is invalid for the SYNC bit

17	DET	=1 if edge-edge timing is invalid for the data bit
16	TOUT	=1 if timeout is detected while waiting for the edge on positive and negative input lines
15-1	TIME_MSB	15 MSBs of the 45-bit timestamp tag (currently bits15-3 are reserved and bits 2-1 contain upper two bits of the timestamp)
0	BUSID	1553 bus ID 1=A, 0=B

SetTxBus(tUeiMIL1553PortActiveBus portBus) can be used to select which bus to listen on – A or B or both. By default, a bus controller listens to communication on both buses.

2.6.2 Programming BusWriter Mode

In Bus Monitor mode, a user can send arbitrary data packets on the bus using the BusWriter mechanism.

To use BusWriter, an appropriate frame needs to be created first:

```
CUeiMIL1553TxFifoFrame* outFrm = new
CUeiMIL1553TxFifoFrame;
```

Then, the frame needs to be filled with appropriate information:

```
outFrm->CopyData(messageSize, data);
outFrm->SetCommand(startRt, startSa, messageSize,
UeiMIL1553CmdBCRT);
writer->Write(1, outFrm, &numFramesWritten);
```

The following command codes are defined:

```
UeiMIL1553CmdModeTxNoData, // Tx Status word
UeiMIL1553CmdModeTxWithData, //
UeiMIL1553CmdModeRxWithDataBroadcast, // Mode
command with data, remote
                              // terminals should
receive data
```

There are two overloaded methods of ***SetCommand():***

```
SetCommand(int Rt_, int Sa_, int WordCount_,
tUeiMIL1553CommandType Command_)
and
SetCommand(int Rt_, int Sa_, int Rt2_, int Sa2_,
int WordCount_, tUeiMIL1553CommandType Command_)
```

SetTxBus(tUeiMIL1553PortActiveBus portBus) can be used to select which bus to use – A or B.

2.7 Stopping the Session

You can stop the session by calling the session object's method *"Stop"*:

```
// Stop the session
mySession.Stop();
```

Note that if you don't explicitly stop the session, it will be automatically stopped when the session object is destroyed or when it goes out of scope.

2.8 Destroying the Session

In C++, if you created the session object on the stack, it will automatically free its resources when it goes out of scope. As an alternative, you can force it to free its resources by calling the method *"CleanUp"*, as shown below:

```
// Clean-up session
mySession.CleanUp();
```

If you dynamically created the session object, you need to destroy it to free all resources:

```
// Destroy session
delete(pMySession);
```

In C, you need to call *"UeiDaqCloseSession"* to free all resources:

```
UeiDaqCloseSession(mySession);
```

With .NET managed languages, the garbage collector will take care of freeing resources once the session object is not referenced anymore. You can also force the session to release its resources by calling the *"Dispose"* method.

2.8.3 Programming BusWriter Mode

In Bus Monitor mode, a user can send arbitrary data packets on the bus using the BusWriter mechanism.

To use BusWriter, an appropriate frame needs to be created first:

```
CUeiMIL1553TxFifoFrame* outFrm = new
CUeiMIL1553TxFifoFrame;
```

Then, the frame needs to be filled with appropriate information:

```
outFrm->CopyData(messageSize, data);
outFrm->SetCommand(startRt, startSa, messageSize,
UeiMIL1553CmdBCRT);
writer->Write(1, outFrm, &numFramesWritten);
```

The following command codes are defined:

```
UeiMIL1553CmdModeTxNoData, // Tx Status word
UeiMIL1553CmdModeTxWithData, //
UeiMIL1553CmdModeRxWithDataBroadcast, // Mode
command with data, remote
                              // terminals should
receive data
```

There are two overloaded methods of ***SetCommand():***

```
SetCommand(int Rt_, int Sa_, int WordCount_,
tUeiMIL1553CommandType Command_)
and
SetCommand(int Rt_, int Sa_, int Rt2_, int Sa2_,
int WordCount_, tUeiMIL1553CommandType Command_)
```

SetTxBus(tUeiMIL1553PortActiveBus portBus) can be used to select which bus to use – A or B.

Chapter 3 Programming with the Low-Level API

The DqAdv functions of the low-level API, which are included in this chapter, offer direct access to PowerDNA DaqBIOS protocol and allow you to access device registers directly. For additional information, please refer to the `API Reference Manual` document under:

Start » Programs » UEI » PowerDNA » Documentation

for all pre-defined types, error codes, and functions you can use with this layer.

NOTE: High-level UEI Framework support for this layer is not available in the current release of this product.

3.1 Low-Level DqAdv Functions

The DNx-ARINC-708/453 Interface Module (Layer) is designed to support MIL-STD-1553A and MIL-STD-1553B interfaces. Two dual redundant independent channels are available. The layer can support one Bus Monitor (BM).

The layer has two independent channels; each channel incorporates all that is needed to communicate with a dual redundant 1553 bus. Bus coupling (transformer/direct) is software-selectable.

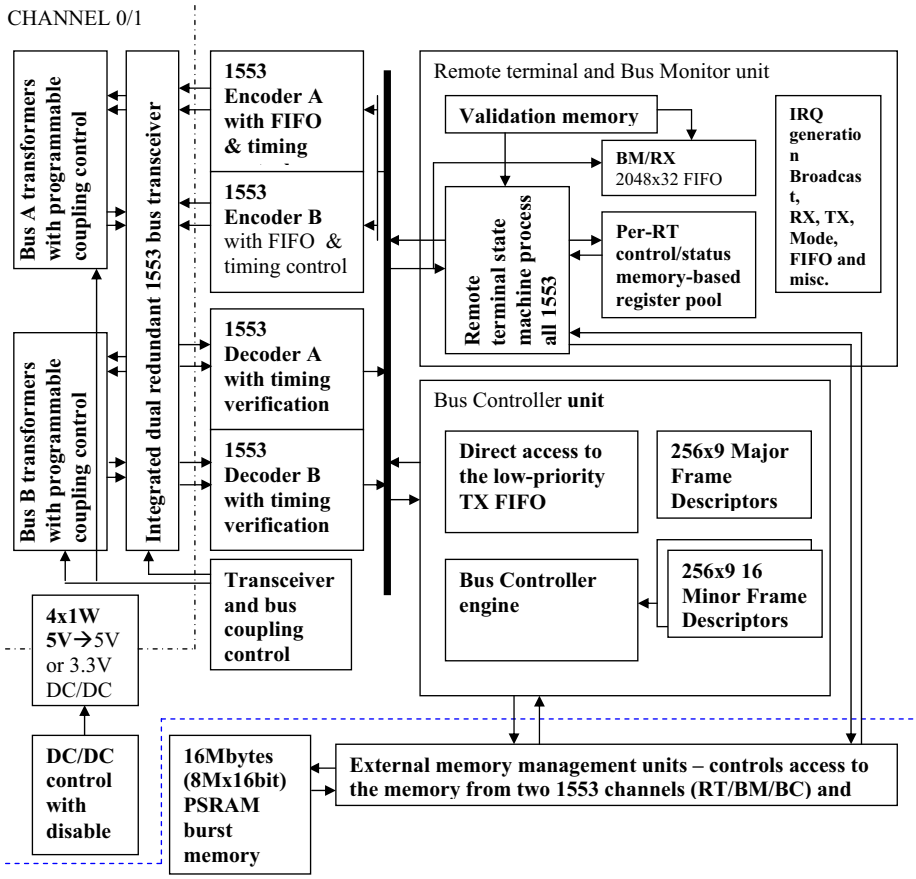

Figure 3-1. DNx-ARINC-708/453 Logic Block Diagram

As shown in **Figure 3-1**, each channel has dual 1553 decoders that are capable of decoding independent streams of 1553 Manchester words and passing them to upper-level subsystems. Decoders can detect various timing errors on the bus and also keep track of the gap interval between messages as well as data parity errors and type of received data words (command-status or data).

When data is stored into the 1024 32-bit word FIFO (BM mode), it is stored from both A and B buses and each control/status word may be optionally time-stamped. Also, each sub-address and each mode command have a flag that controls interrupt generation upon receiving a corresponding RX /TX or mode command. Switching between A and B redundant buses takes place automatically upon receiving a command on the bus, and the host may receive an interrupt once it happens.

The Manchester encoder allows data output on A and/or B buses. However, it always accepts data from the same source; i.e., it is impossible to send different data to A and B buses at the same time. The encoder has a two-256x32 word FIFO that accepts 1553 data in a format that includes bus inactivity gap time delay, word type, and parity information.

A dedicated memory controller interfaces with 16Mbytes of fast burst PSRAM. It keeps up with data requests from both channels for the TX data and accepts RX messages and status information as well. Once a message is received or transmitted for the particular subsystem, special flags are set and once new data is placed in the TX buffer or data is read from the RX buffer, those flags are cleared. DNA may write or read data in blocks as large as 1024 16-bit data words at a time. A read or write is executed as an atomic transaction, i.e., no data change allowed during a read or write to/from the DNA bus.

This document describes the initial function set for BM and RT modes of operation.

For detailed descriptions of the low-level functions you can use with DNx-ARINC-708/453 boards, refer to the PowerDNA API Reference Manual, which is available for download at **www.ueidaq.com**

Appendix

A. Accessories

DNA-CBL-COM
1-ft long, round shielded cable with 37-pin male and four 9-pin male D-sub connectors

Index