Avionics Interface Computer

DDC
Connectivity Power Control

Software User's Manual

Model: BU-69094SX

AL enc_shift_data : std_logic_vector(15 downto 0)
OCESS (clk, hsp_rst_l)
EGIN
if (hsp_rst_l = '0') then
enc_shift_data <= (others => '0');
elsif ((clk'event) and (clk = '1')) then
if (erst_l = '0') then
enc_shift_data <= (others => '0');
elsif ((edw = '0') or (stat_bit_ld = '1')) then
enc_shift_data <= enc;
elsif (encclk2 = '1') and (bit_count(0) = '1')) then

The Avionics Interface Computer (AIC) Software Development Kit (SDK) provides the framework for efficient development of applications utlilizing DDC's AIC for protocol conversion and remote access testing.

Need a Custom Solution?
DDC can customize designs for all boards, ranging from simple modifications of standard products to fully customized solutions for commercial, military, aerospace, and industrial applications.

For more information: www.ddc-web.com/BU-69094SX

DDC's Data Networking Solutions

MIL-STD-1553 | ARINC 429 | Fibre Channel | Ethernet

As the leading global supplier of data bus components, boards, modules, computers, and software solutions for the military and commercial aerospace markets, DDC's data bus networking solutions encompass the full range of data interface protocols to support the real-time processing demands of field-critical data networking between systems and subsystems on the platform. These products, along with our traditional MIL-STD-1553 solutions, represent a wide and flexible array of performance and cost solutions, enabling DDC to support multi-generational programs.

Whether employed in increased bandwidth, high-speed serial communications, or traditional avionics and ground support applications, DDC's data bus solutions fulfill the expanse of military, civil aerospace, and space requirements including reliability, determinism, low CPU utilization, real-time performance, and ruggedness within harsh environments. Our use of in-house intellectual property ensures superior multi-generational support, independent of the life cycles of commercial devices. Moreover, we maintain software compatibility between product generations to protect our customers' investments in software development, system testing, and end-product qualification.

MIL-STD-1553

DDC, the world leader in MIL-STD-1553 technology, provides the broadest selection of quality MIL-STD-1553 rugged embedded and lab grade computers, boards and components to meet your data conversion and data interface needs. Our 1553 data bus board solutions are integral elements of military, aerospace, and industrial applications. Our extensive line of military and space grade components provide MIL-STD-1553 interface solutions for microprocessors, PCI buses, and simple systems. Our 1553 data bus solutions are designed into almost every aircraft, helicopter, unmanned vehicle, missile programs, and space system that utilizes MIL-STD-1553.

ARINC 429

DDC has a wide assortment of quality ARINC 429 embedded and lab grade boards, LRUs, and components, to serve your data conversion and data interface needs. DDC's ARINC 429 components ensure the accurate and reliable transfer of flight-critical data. Our 429 interfaces support data bus development, validation, and the transfer of flight-critical data aboard commercial aerospace platforms.

Fibre Channel

DDC has developed its line of high-speed Fibre Channel network access controllers and switches to support the real-time processing demands of field-critical data networking between sensors, computer nodes, data storage, displays, and weapons, for air, sea, and ground military vehicles. Fibre Channel's architecture is optimized to meet the performance,reliability, and demanding environmental requirements of embedded, real time, military applications, and designed to endure the multi-decade life cycle demands of military/aerospace programs.

Ethernet

DDC offers convenient solutions to convert MIL-STD-1553, ARINC 429, and Ethernet protocol in any direction, in real-time, without a host computer, enabling seamless and cost saving multi-protocol connectivity for test and embedded applications.

Extensions to MIL-STD-1553

DDC offers a wide variety of solutions based on extensions of MIL-STD-1553 for emerging aerospace applications. Turbo 1553 increases the data rate of 1553 from 1 Mbps to 5 Mbps while maintaining the architectural features of MIL-STD-1553. Hyper 1553 provides high speed communication (50 to 100+ Mbps) over MIL-STD-1553 buses while operating concurrently with legacy 1 Mbps 1553 (similar to ADSL for telephone networks).

Form Factors, Software, & Drivers

DDC supplies MIL-STD-1553 and ARINC 429 board level products in a variety of form factors including USB, PCI-Express, PCMCIA, ExpressCard, AMC, PMC, XMC, PCI-104, PC/104-Plus, PC/104, PCI, cPCI, VME, and ISAbus boards. Our laboratory simulation and in-flight products include multi-function and single-function for system integration and production test environments. Our extensive line of military and space grade components provide MIL-STD-1553 interface solutions for microprocessors and simple systems. Our software is supplied in the form of menus, libraries, and drivers. We also offer additional software to expand our data networking range of options.

AVIONICS INTERFACE COMPUTER SDK
SOFTWARE USER’S MANUAL

MN-69094SX-002

105 Wilbur Place
Bohemia, New York 11716-2426
Tel: (631) 567-5600, Fax: (631) 567-7358
World Wide Web - http://www.ddc-web.com

For Technical Support - 1-800-DDC-5757 ext. 7771
United Kingdom - Tel: +44-(0)1635-811140, Fax: +44-(0)1635-32264
France - Tel: +33-(0)1-41-16-3424, Fax: +33-(0)1-41-16-3425
Germany - Tel: +49-(0)89-15 00 12-11, Fax: +49-(0)89-15 00 12-22
Japan - Tel: +81-(0)3-3814-7688, Fax: +81-(0)3-3814-7689
Asia - Tel: +65- 6489-4801
India - Tel: +91 080 301 10 200

RECORD OF CHANGE

Revision	Date	Pages	Description
Pre Rev A	1/2015	All	Preliminary Release
Rev A	4/2016	All	Initial Release
Rev B	8/2017	All	Minor text edits throughout document

1 PREFACE

This manual uses typographical conventions to assist the reader in understanding the content. This section will define the text formatting used in the rest of the manual.

1.1 Text Usage

- **BOLD** – text that is written in bold letters indicates important information.
- `Courier New –` is used to indicate code examples.
- <…> - Indicates user entered text or commands.

1.2 Standard Definitions

AIC	Avionics Interface Computer

1.3 Special Handling and Cautions

The BU-69094S is delivered on a Compact Disc. Proper care should be used to ensure that the discs are not damaged by heat.

The USB **OS Recovery Device** is used to flash a saved image from the USB Device onto the AIC. This is a fully automated procedure that happens automatically when the USB **OS Recovery Device** is connected to one of the AIC's USB ports at power up. The USB **OS Recovery Device** will erase all data on the AIC after a thirty second timeout unless the user presses a key at the boot menu.

1.4 Trademarks

All trademarks are the property of their respective owners.

1.5 Technical Support

In the event that problems arise beyond the scope of this manual, you can contact DDC by the following:

US Toll Free Technical Support:
1-800-DDC-5757, ext. 7771

Outside of the US Technical Support:
1-631-567-5600, ext. 7771

Fax:
1-631-567-5758 to the attention of DATA BUS Applications

DDC Website:
www.ddc-web.com/ContactUs/TechSupport.aspx

Please note that the latest revisions of Software and Documentation are available for download at DDC's Web Site, www.ddc-web.com.

2 OVERVIEW

This document will cover the Avionics Interface Computer (AIC) specific version of the BU-69094S1 software package, version 2.0.0 or later. Earlier versions of the BU-69094S1 (1.x.x) are for the AceXtreme Bridge Device® and are not compatible with the AIC.

2.1 Description

DDC's Avionics Interface Computer (AIC), (or as it's known by its part number BU-67121Wx) is a development system for lab and production test applications which provides a scalable, programmable, and portable platform to develop and test MIL-STD-1553 and ARINC 429 system applications via an Ethernet network. By utilizing the Ethernet network with the AIC's Remote Access or Protocol Conversion modes eliminates the need and cost of long cabling/wire runs from the test lab to the onboard 1553/429 interface under test. The AIC's use of a Linux OS provides the ability to customize the AIC as it allows for the integration of DDC line of cards as well as 3rd party PMC and PCIe cards.

The AIC comes with Fedora Core 20 installed on its Solid State Drive and has an Intel® Q7 Processor and 2 GB of DDR3L SDRAM, which enables the AIC to serve a broad range of data conversion and unique application needs. The AIC includes one 10/100/1000 Ethernet channel, two USB 2.0 ports, one RS-232 port, and one VGA port. The AIC operates over an ambient air temperature range of 0°C to +55° C, and includes an internal fan.

The AIC has two PMC and two mini PCIe sites providing hardware flexibility for additional hardware components such as MIL-STD-1553, ARINC 429, ARINC 717, Avionics and Discrete I/O, AFDX and many other configurations. For other configurations not mentioned, please see the Configuration Options or Ordering Information sections for additional details on customizing the AIC. Please also contact the factory with custom requirements that is not seen on the ordering information tables.

The AIC offers a high degree of flexibility, and is therefore suitable for a wide range of lab applications. The AIC has three modes of operation Remote Access Mode, Protocol Conversion Mode and Stand Alone Mode.

In Protocol Conversion Mode, the AIC is configured to provide autonomous communication from any input channel(s) to any other channel(s). To minimize setup time and provide turnkey operation, the AIC includes a high-level protocol conversion API. Alternatively, users may develop their own conversion applications by means of DDC's AceXtreme MIL-STD-1553 and/or ARINC 429 APIs, along with Linux TCP/IP and UDP/IP socket interfaces for the AIC's Ethernet channel.

In Remote Access Mode, users are able to develop applications running on a remote computer communicating over Ethernet to the AIC's MIL-STD-1553 and/or ARINC 429 channels. In the remote access configuration, the user is able to write applications running on a remote host invoking the AceXtreme MIL-STD-1553 and/or ARINC 429 APIs. As an alternative to developing application software, in Remote Access Mode, the user is able to operate the AIC using any of DDC's GUI software programs. These include:

- **BusTrACEr**, a simple menu program for generating and monitoring MIL-STD-1553 messages. Further, BusTrACEr includes an option for the automatic generation of ANSI 'C' source code.
- ***dataSIMS*****,** a software GUI tool for test and simulation applications. *dataSIMS* converts data to engineering units, allows the creation of graphical display formats, and may be used for either passive monitoring and/or simulation.
- **LabVIEW® and LabVIEW® Real-Time Support**. The **BU-69093S0-XX0** software operates in conjunction with National Instruments' LabVIEW® or LabVIEW® Real-Time system design software to provide a simple interface and easy programming of the AIC's MIL-STD-1553 and/or ARINC 429 interfaces. Users can either create their own custom interfaces "from scratch" or may modify the samples that are provided.
- **Commercial Avionics Utilities Software Package. The DD-42999S0-XX0 Data Bus Analyzer and Data Loader GUI software** is for ARINC 429 data bus analysis and simulation. This GUI provides advanced filtering, message scheduling, and triggering. In addition, it includes a graphical ARINC 615 data loader, providing a software interface to load data to and from airborne computers.
- Standalone mode allows a user to operate the AIC as a user programmable computer system. Software Development Kits (SDKs) are provided for MIL-STD-1553 and ARINC 429 to facilitate the development of applications requiring communication on these avionics I/Os. Onboard video, a Serial port and USB ports can also be utilized to further enhance Standalone mode. The user will not need to depend on a host PC.

2.2 Features

General

- Bridging Ethernet, MIL-STD-1553, and ARINC 429
- Development Computer
    - Intel® Atom™ E3845 Quad Core 1.91GHz Processor
    - 2GB DDR3L SDRAM
    - 30 GByte SSD
    - 10/100/1000 Base-T Ethernet, USB 2.0, RS-232

  - Linux Operating System
  - Lab Grade, Rack-Mountable Chassis
  - 2 PMC and 2 Mini-PCIe Expansion Slots
- Various PMC and Mini-PCIe Modules Support a Range of Avionics Interfaces
  - MIL-STD-1553
  - ARINC 429
  - ARINC 717
  - Avionics Discrete I/O
  - Custom I/O
- Three Modes of Operation, Using DDC's Hardware and Software
  1. **Remote Access Mode** Uses Ethernet as a Virtual Backplane Between Applications Running on a Host Computer and 1553/429 Interfaces Located Within the AIC, Eliminating the Need and Cost of Long Cabling to Onboard 1553/429 Connections from the Test Lab
  2. **Protocol Conversion Mode** Uses Bridging SDK, Which Allows Userts to Easily Create Embedded Software on the AIC that will Autonomously Forward Data Between MIL-STD-1553, ARINC 429, and Ethernet Interfaces
  3. **Standalone Mode** Allows the AIC to Operate as a User Programmable Computer System

Software

- Linux Operating System and BSP
  - Ethernet Stacks, with UDP/IP and TCP/IP Sockets, Telnet, FTP, TFTP, SSH, and HCTP.
- DDC Protocol Conversion API, Providing Turnkey Conversion From Any Ethernet, 1553, or ARINC 429 Port to Any Other Port(s)
- DDC AceXtreme MIL-STD-1553 API, Including Sample Programs
- DDC ARINC 429 API, Including Sample Programs
- Built-in Editor, Allowing Editing and Saving Files Over Telnet
- Built-in 'C' Compiler
- Can transfer internal files to a host computer, edit remotely, and transfer files back to the AIC before compiling.

2.3 System Requirements

2.3.1 System Requirements for Protocol Conversion Mode

- Remote computer with Ethernet interface and Telnet.

2.3.2 System Requirements for Remote Access Mode

- Remote computer with Ethernet interface.
- Windows XP, Windows Vista 32/64-bit, Windows 7 32/64-bit, Windows 8 32/64-bit, Linux, or VxWorks Operating System
  - Workbench software development environment for VxWorks platforms
- An appropriate compiler or development environment.
- Contact factory for additional operating systems

2.4 MIL-STD-1553 Capability

The Avionics Interface Computer can provide many MIL-STD-1553 channels utilizing the DDC AceXtreme architecture.. Features include a highly autonomous bus controller with expanded instruction set, a Multi-RT interface providing a wide variety of buffering options, a 1553 bus monitor that allows you to store data in an industry standard IRIG-106 Chapter 10 format, IRIG-B time code input and 48-bit 100 nanosecond resolution Time Tag for each MIL-STD-1553 channel.

The AIC configured with DDC’s single function boards support only operation of individual modes at one time per channel. These modes include BC/MTI, or Multi-RT, or Multi-RT/MTI, or MTI mode. The AIC configured with DDC’s multi-function boards support running all modes concurrently on one channel. Each 1553 channel is configured with up to 2MB of onboard RAM per installed channel.

Each MIL-STD-1553 interface implements a transformer-coupled dual-redundant bus connection (consult factory for direct-coupled).

2.4.1 Bus Controller Mode

The AceXtreme’s MIL-STD-1553 Bus Controller (BC) is based on the 32-bit architecture of DDC’s AceXtreme 1553 Bus Controller.

AceXtreme’s BC architecture retains much of the previous generation (Enhanced Mini-ACE, Mini-ACE Mark 3, Micro-ACE (TE), and Total-ACE) 1553 Bus Controller architecture. However, it expands upon it in specific areas to provide improved capabilities.

The AceXtreme® BC architecture is based on a built-in command interpreter with a set of 32 instructions. The command interpreter is a message sequence control engine that provides a high degree of flexibility for implementing 1553 message lists, including major and minor frame scheduling. It separates 1553 message data from control/status data for the purposes of implementing different data block handling schemes, performing bulk data transfers, and implementing automatic message retries. It also includes the capability for automatic bus switchover for failed messages and reporting of various error and status conditions to the host processor by means of five user-defined interrupts and a general-purpose queue.

Two asynchronous queues are also included, to improve the Bus Controller’s efficiency and flexibility. The High Priority Queue (HPQ) enables the user to easily insert asynchronous messages into a running message list, causing it to operate on the new message immediately. The Low Priority Queue (LPQ) enables the user to insert asynchronous messages which will only be processed when there’s sufficient “dead-time” available on the bus at the end of a minor frame.

The AceXtreme’s BC Engine implements all MIL-STD-1553B message formats. Message format is programmable on a message-by-message basis. Automatic retries and interrupt requests may be enabled or disabled for each individual messages. The BC performs all error checking required by MIL-STD-1553B. This includes validation of response time, sync type and sync encoding, Manchester II encoding, parity, bit count, word count, Status Word RT Address field, and various RT-to-RT transfer errors. The BC No-Response timeout value is also programmable to enable operation over long buses or through repeaters.

2.4.2 Remote Terminal Operation

The AceXtreme RT architecture builds upon the single-RT architecture of Enhanced Mini-ACE, Mini-ACE Mark 3, Micro-ACE(TE), and Total-ACE.

One of the major features of AceXtreme is its Multi-RT capability. That is, the AceXtreme provides the capability to implement up to 31 independent Remote Terminals (up to 32 RTs if Broadcast is disabled).

The AceXtreme RT engine can also be configured to operate in a Single-RT legacy mode of operation. Single-RT operation supports hardware control of the RT address and automatic boot, allowing the AceXtreme to respond to commands with Status with its Busy bit set immediately following power turn-on without requiring configuration by the host.

For RT (and/or Monitor) applications, where the possibility of BC operation must be absolutely prohibited, the AceXtreme Bridge Device includes a DISABLE_BC input signal. In addition to single-RT and Multi-RT operation, the AceXtreme includes the following capabilities:

- Meets MIL-STD-1553A, MIL-STD-1553B, MIL-STD-1760, and STANAG 3838 standards.
- Multiple Data Handling Modes:
    - Single Buffer Mode
    - Double Buffer Mode
    - Circular Buffer Mode
    - Global Circular Buffer
- Command Illegalization by Subaddress/Word Count, and Mode Codes
- Programmable Busy by Subaddress
- Flexible Interrupt Conditions, Including 50% and 100% Rollover Interrupts for Circular Buffers
- Interrupt Status Queue with Programmable Filtering
- Time Tagging Options for Synchronize Mode Codes
- Option for RT Auto-Boot with Busy bit Set

2.4.3 Monitor Mode Operation

The AceXtreme® Monitor engine provides the next generation DDC MIL-STD-1553 Monitor (MT) architecture. This new Monitor autonomously stores individual messages into a contiguous memory stack formatted as IRIG 106 Chapter 10 Data Packets.

The legacy Monitor modes of operation traditionally implemented in previous generations of DDC MIL-STD-1553 engines can be emulated easily through host software. All information and functionality supported on the legacy Monitor engines is supported on the AceXtreme Monitor engine, or may be extracted from the stored messages.

IRIG 106 Chapter 10 provides interoperability for such applications as test range telemetry, flight test instrumentation, mission recorders, video/data servers; surveillance and reconnaissance; health and usage monitoring; mission planning, debriefing, and training; and flight operations. IRIG 106 Chapter 10 defines a standardized file format, and within that, specific representations for several types of flight data, including MIL-STD-1553 buses, PCM, analog, computer-generated data, images, discretes, UARTs, IEEE 1394, parallel, IRIG time, video, and voice. In addition, Chapter 106 provides standardization of time bases.

For MIL-STD-1553, IRIG 106 Chapter 10 defines packets that can encapsulate one or more 1553 messages. Within these packets, all messages are tagged with either a 48-bit relative or 64-bit absolute time stamp. For each message, there is also a block status word, which includes indications of bus channel and message validity, and identifies specific errors. The 1553 format also defines indications of response time, plus storage of all 1553 Command, Status, and Data Words, in the order received. For supporting IRIG 106 Chapter 10, the AceXtreme Bridge Device includes DMA capability, which enables high-speed transfers of monitored data from the 1553 monitor to the AceXtreme Bridge's Atom processor host space.

2.5 ARINC 429 Capability

The AIC can be configured with options to include many ARINC 429 channels.. Each channel may be programmed to operate as a transmit or receive channel. All channels comply with the ARINC 429 electrical specification.

Please see each card's hardware manual for signal lists.

Each channel implements numerous software configurable interrupts, data handling, and error control options.

2.6 ARINC-717 Capabiltiy

Many of the AIC's ARINC 429 channels may be programmed by software for either transmit or receive operation, and also for low-speed or high-speed operation. In addition, the AIC's ARINC 429 controllers are each capable of operating in ARINC 429 or ARINC 575 mode. Each ARINC channel is independently programmable to operate in either mode of operation.

Depending on which model of the AIC you have, some channels can be programmed as 429 or 717 channels, as well as Receive or Transmit and Hi or Lo speeds. Other channels are dedicated 429 Receive channels only.

2.7 10/100/1000 Ethernet Capability

The AIC includes an Ethernet interface. The Ethernet interface is capable of operating over 10 BASE-T, 100 BASE-T, or 1000 BASE-T physical layers, with auto-negotiation capability. The Linux stack running on the AIC's Atom processor supports TCP/IP and UDP/IP protocols. The IP address on the AIC can be configured for either static or DHCP.

2.8 Digital Discrete I/O Capability

An AIC that is configured with a DDC card that includes discrete digital signals, can be individually programmable as inputs or outputs. Discrete digital I/O channel 1 is the LSB and channel 8 is the MSB for any software accesses to these signals.

The discrete digital I/O signals are 5V tolerant with 10K pull-up resistors to +3.3V. These I/O signals default to inputs after power up. The outputs are +3.3V totem-pole type with 12mA drive capability.

In the enhanced BC mode, during the execution of a Wait for External Trigger (WTG) instruction, the BC will wait for a low-to-high transition on DIO X (where X is the channel number) before proceeding to the next instruction.

This feature must first be enabled using the **aceBCExtTriggerEnable()** function.

Discrete Digital I/O channels 3-8 also function as RT address inputs for 1553 channel #2.

The discrete digital I/O signals can also be linked to BC/RT IMRs. BC IMRs can only be linked to DIO 0-3, while RT IMRs can only be linked to DIO 4-7.

The BU-67210 architecture also allows for the discrete digital I/O signals to be linked to triggers. When a trigger is linked to a discrete I/O, the number of the trigger is the same as the number of the discrete I/O, i.e. GPTn is linked to DIOn.

2.9 Avionics Discrete I/O Option

An AIC that is configured with a DDC card that includes Avionic Level Discrete I/O Channels that can be individually programmable as inputs or outputs. Avionics Discrete digital I/O channel 1 is the LSB and channel 8 is the MSB for any software accesses to these signals.

As outputs, they are Open-Drain type drivers. When used as inputs, these channels are configured to sense Ground / Open discrete inputs with +35V input voltage tolerance.

If an Avionics Discrete is used with an external Avionics Input, an external 2K ohm resistor should be connected between the AIO signal and system power to ensure full swing of the discrete signal, unless the other device has an internal pull-up of no greater than 2K ohms. Avionics I/O channels 3-8 also function as RT address inputs for 1553 channel #1.

Multi-Function architecture also allows for the avionics discrete I/O signals to be linked to triggers. When a trigger is linked to a discrete I/O, the number of the trigger is the same as the number of the discrete I/O, i.e. GPTn is linked to DIOn.

3 MODES OF OPERATION

The AIC is supports three distinct modes of operation:

1. Protocol conversion mode.
2. Remote access mode.
3. Standalone Mode.

3.1 Protocol Conversion Mode

Figure 1 shows an example of the Avionics Interface Computer operating in its Protocol Conversion Mode. In this mode, the AIC may be configured to provide autonomous communication bridging between any channel and any other channel(s).

To minimize setup time and provide turnkey operation, the AIC includes a high-level protocol bridging API. Alternatively, users may develop their own bridging applications invoking DDC's AceXtreme MIL-STD-1553 and/or ARINC 429 APIs, along with the Linux socket interfaces for the two Ethernet channels.

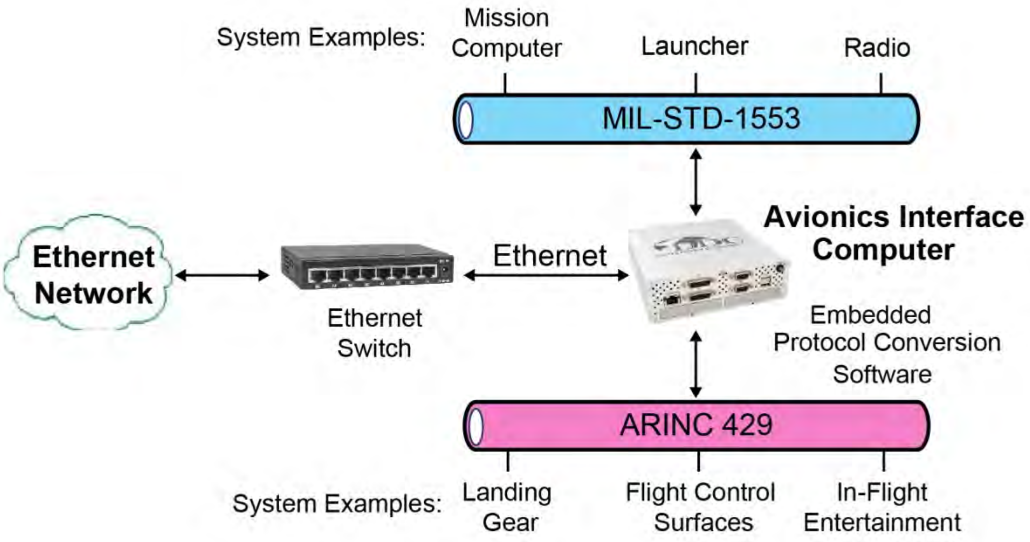


Figure 1. Example of the AIC in Protocol Conversion Mode

3.2 Remote Access Mode

Figure 2 shows an example of the Avionics Interface Computer operating in its remote access mode. In remote access mode, users can develop applications running on a remote computer that communicates over Ethernet to the AIC's MIL-STD-1553 and/or ARINC 429 channels. In the remote access configuration, the user will be able to write applications running on a remote host that invoke the AceXtreme MIL-STD-1553 and/or ARINC 429 APIs. For use in remote access mode, DDC offers AceXtreme software drivers Remote Host Drivers for Windows, XP, Vista, 7 and 8; Linux kernel version 2.6.x; and WindRiver VxWorks versions 6.x.

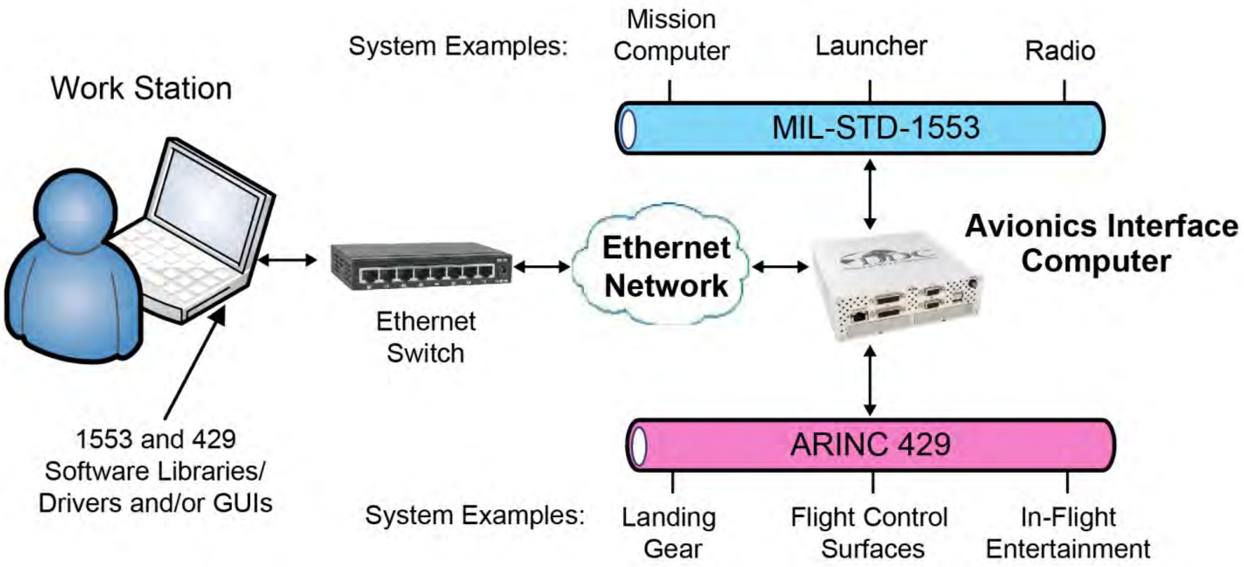


Figure 2. Example of the Avionics Interface Computer Used in Remote Access Mode

3.3 Standalone Mode

Standalone mode allows the AIC to operate as a user programmable computer system. Software Development Kits (SDKs) are provided for MIL-STD-1553 and ARINC 429 to facilitate the development of applications requiring communication on these avionics I/Os. Onboard video, a Serial port and USB ports can also be utilized to further enhance Standalone mode. The user will not need to depend on a host PC.

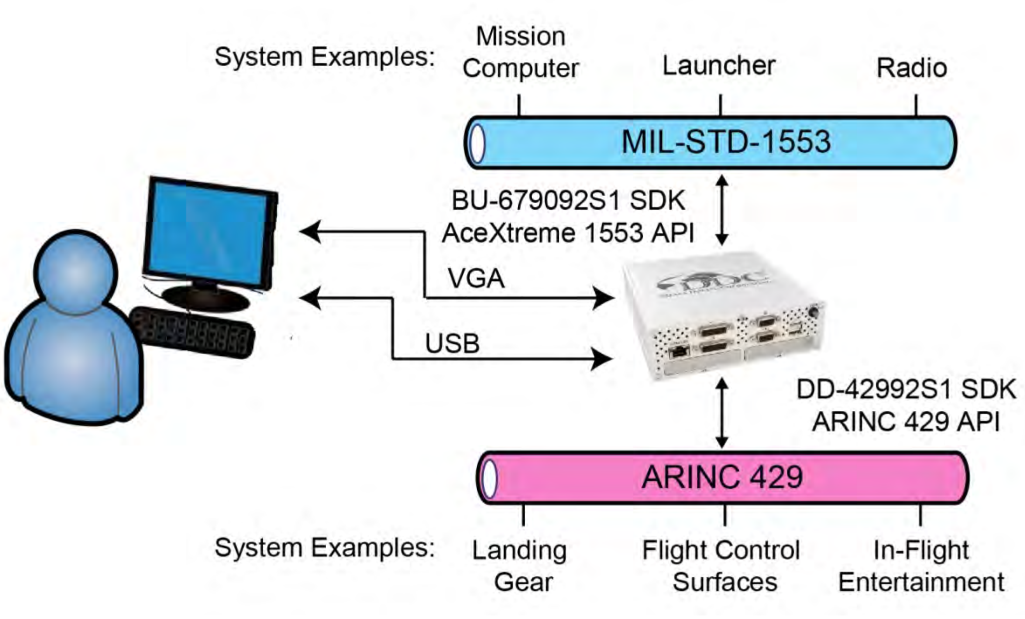


Figure 3. Example of Avionics Interface Computer Used in Standalone Mode

4 DEVICE STARTUP

The following section defines the steps to initially setup the Avionics Interface Computer and remotely communicate with it.

4.1 Minimal Cable Connections

The following is a list of equipment required to configure and communicate with the AIC. Please see the BU-67x21W Quick Start Guide for more information.

- DDC Avionics Interface Computer (BU-67121W)
- Standard RJ-45 Ethernet Cable
- Wall Outlet
- AIC Power Supply Adapter
- Desktop/Laptop with Telnet/SSH Client Installed
- Ability to assign static IP address/mask or use DDCNDF utility to find the AIC and retrieve its associated IP address

4.2 Powering up the device

- First make sure to unpack the Avionics Interface Computer and accessories
- Connect the Ethernet cable to the ‘**10/100/100’** connector port on the AIC
- Connect the AIC Power Supply Adapter to the PWR IN receptacle on the back of the AIC
    - See the Avionics Interface Computer Hardware Manual for more information
- Connect the AIC Power Adapter to the Wall Outlet, and press the PWR button on the front of the AIC to power it on

4.3 Entering the BIOS

The BIOS of the AIC is password protected.

It is recommended that the BIOS <u>**not**</u> be tampered with at all, so that your system is not corrupted beyond restore.

If the BIOS password is needed, please contact the factory.

4.4 Internal Date and Time

The Avionics Interface Computer does contain a battery backup to store system time. By default, the device will boot up with the current data and time. If you would like to configure a different date and/or time, please run the Linux command to set the date and time accordingly. A reboot of the system may be necessary for the time change to take effect.

4.5 IP Address Configuration

DDC's Avionics Interface Computer will ship from the factory with a static IP configuration (See Section 4.5.1).

It is highly likely that you will need to modify the IP Settings before connecting the device to the desired final network location.

Once the AIC IP address settings have been modified for your network, it is acceptable to connect the device to any switch within your LAN.

In order to connect to the AIC for the first time, please set your host laptop/desktop to an IP address on the 192.168.1.x network.

(Recommended: address: 192.168.1.90 net mask: 255.255.255.0)

4.5.1 Factory IP Address Settings

Default configuration for the AIC IP Address settings is as follows:

Device:	**Ethernet Port (p3p1)**
Address:	**192.168.1.80**
Net mask:	**255.255.255.0**
Gateway:	**0.0.0.0**

Note: *If the IP Address settings are accidentally lost, you use the DDCNDF utility to find the AIC and retrieve its associated IP address.*

4.5.2 Modifying AIC IP Configuration

The Ports List link from the Avionics Interface Computer's Web Server displays the installed ports on the AIC. The ports include the number of MIL-STD-1553 and ARINC 429 channels, which includes their Vendor and device IDs, and the driver the device is using. The Ethernet Ports on the AIC are also listed under the Ports List link, displaying the Ethernet port name and the current IP address of the AIC, along with the AIC's NETBIOS Name. For modifying the AIC IP Address see section 6.1.1. The **DDCNDF** Utility can then be used to find the new IP address of the AIC see section 6.1.2.

4.6 Logging into the device

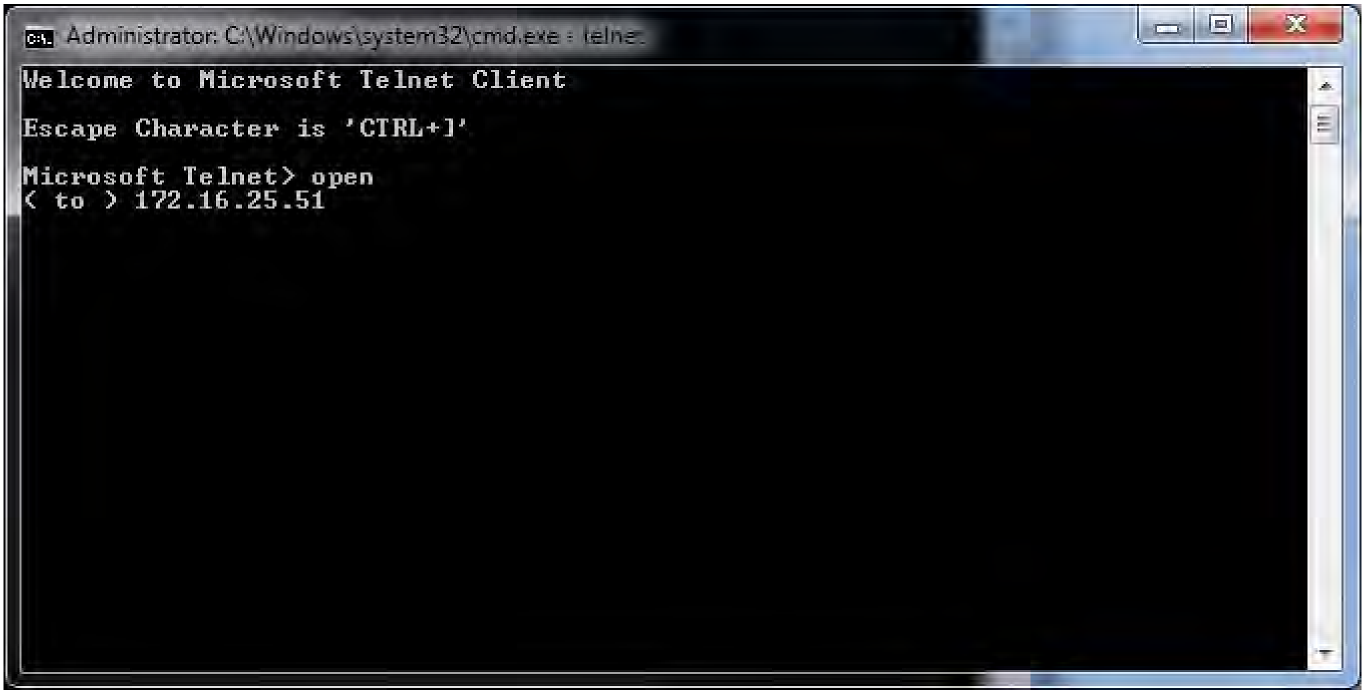
To connect to the Avionics Interface Computer through Telnet, open a Telnet shell on your host system and type **"telnet"** at the prompt. Once the telnet application has been started, type **"open"** and enter the IP address of your Avionics Interface Computer (see Section 4.5.2).

You can also log into the AIC via the serial COM port on the rear of the device.

Use the following parameters to login via a terminal emulator program, such as TerraTerm or PuTTy:

Baud Rate:	**115200**
Data bits:	**8**
Stop bits:	**1**
Parity bits:	**none**
Flow control:	**none**

You will see the same prompt as Figure 5., below.

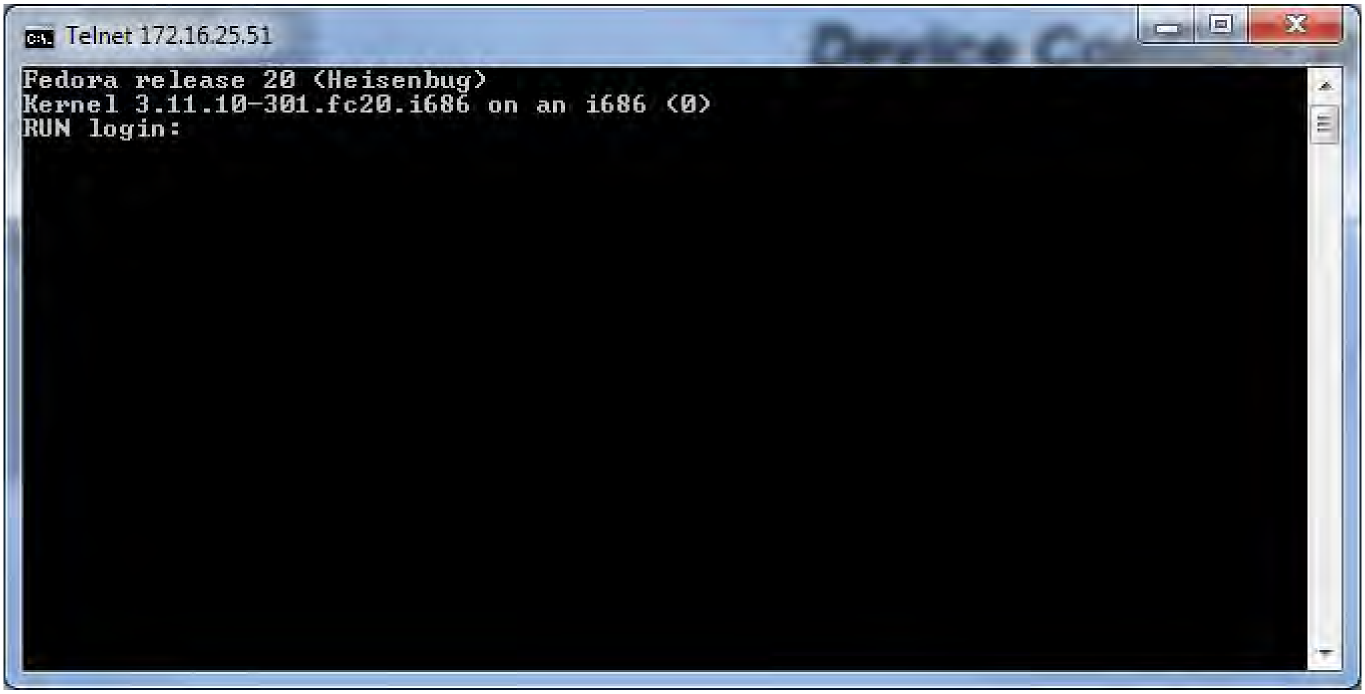


```
Welcome to Microsoft Telnet Client

Escape Character is 'CTRL+]'

Microsoft Telnet> open
( to ) 172.16.25.51
```

Figure 4. Connecting to AIC with Telnet

Note: *The Telnet service is not installed by default on a Windows 7/8 PC. To enable Telnet see section 6.2.1.*

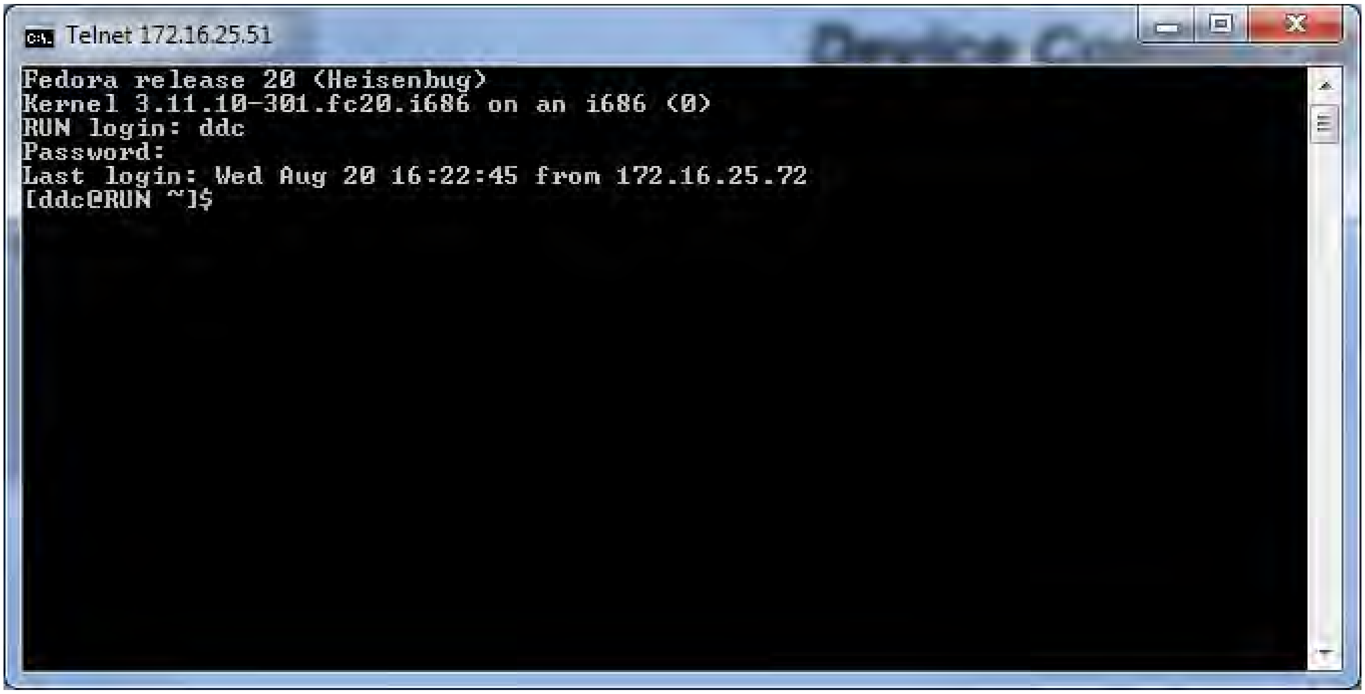
Once you have connected to the AIC with Telnet, you will be prompted to enter the username and password of your account on the AIC.



```
Fedora release 20 (Heisenbug)
Kernel 3.11.10-301.fc20.i686 on an i686 (0)
RUN login:
```

Figure 5. Login via Telnet

After entering the username and password you are ready to start using the Linux Operating system installed on the Avionics Interface Computer.



```
Fedora release 20 (Heisenbug)
Kernel 3.11.10-301.fc20.i686 on an i686 (0)
RUN login: ddc
Password:
Last login: Wed Aug 20 16:22:45 from 172.16.25.72
[ddc@RUN ~]$
```

Figure 6. Logged into AIC via Telnet

5 USING THE REMOTE SHELL INTERFACE

DDC's Avionics Interface Computer (AIC) uses the SSH built-in Shell interface, distributed by Fedora 20. The shell gives the user access to the internal file system, compilers, and sample applications.

The AIC will ship from the factory with all necessary software to begin developing custom bridging applications.

Users familiar with UNIX/LINUX shell interfaces should be comfortable using the built-in shell.

5.1 Usernames and Passwords

All DDC Avionics Interface Computers ship with 2 user accounts, **root** and **ddc**. It is recommended that the '**root'** account only be used for extraordinary circumstances that require "Administrator" access. The '**ddc'** account can be used for normal configuration and programming of the device.

Username: **ddc**	Password: **ddc**	Description: **Normal User**
Username: **root**	Password: **ddc**	Description: **Administrator**

5.2 Changing Passwords

To change the password for either the **root** or the **ddc** account, first login (See Section 4.6) to the device as the user whose password is to be changed.

Type the following command:

```
passwd
```

You will be instructed to enter the user's current (old) password, followed by the new desired password. You should observe a confirmation that the password was changed successfully.

```
# passwd
Changing password for ddc
Old password:  << TYPE OLD PASSWORD >>
New password:  << TYPE NEW PASSWORD >>
Password updated successfully!
```

5.3 Directory Structure

The following is the directory structure within the Avionics Interface Computer. It consists of numerous samples, libraries, and build files to show proper use of the DDC SDK Programming Interface.

All user files reside in the **"/home"** directory, which will be the default directory after logging in.

Table 1. Avionics Interface Computer SDK Directory Structure

Folder Name	Description of Contents
/home/ddc	Main User DDC Home Directory.
/home/ddc/run_mode_prog	Contains applications to execute on startup.
/home/ddc/1553_429/samples/bridging	Samples on using DDC's Protocol Conversion SDK directly.
/home/ddc/1553_429/samples/dd429	Samples on using DDC's ARINC 429 SDK directly.
/home/ddc/1553_429/samples/emacepl	Samples on using DDC's MIL-STD-1553 SDK directly.
/home/ddc/1553_429/libraries/bridging	Protocol Conversion Layer (PCL) Home Directory.
/home/ddc/1553_429/libraries/dd429	ARINC 429 library folder.
/home/ddc/1553_429/libraries/emacepl	MIL-STD-1553 AceXtreme library folder.
/home/ddc/1553_429/libraries/ethernet_socket	Remote Access library folder.
/home/ddc/1553_429/drivers	Contains the acex and legacy drivers. Legacy drivers consist of the acexusb, emapci and e2mausb drivers.
/home/ddc/1553_429/tools	Contains tools needed by BU-69094S package.
/home/ddc/1553_429/docs	Contains package documentation.

5.4 Editing Files

Once you have initiated a Telnet/SSH session (See Section 4.6), you can use the built-in editor to edit and save any text-based files (Such as Sample 'C' Source code).

The Built-in editor is *vi*, a popular text-based file editor.

Alternatively, you can transfer any file to a remote computer, edit them remotely, and transfer them back to the Avionics Interface Computer before compiling. See Section 5.5 for more information on transferring files.

For more information on using the *vi* Editor, please see www.vim.org

5.5 Transferring Files

The Avionics Interface Computer contains an FTP (File Transfer Protocol) Server that allows external clients to transfer files remotely to and from the device.

Note: *The FTP Sever Port is 21. See Section 5.1 for login information.*

FTP Clients are included with Windows, Linux, as well as numerous 3rd Party Tools.

Please see the links below for more information using FTP.

Using Windows: http://windows.microsoft.com/en-us/windows-vista/File-Transfer-Protocol-FTP-frequently-asked-questions

Using Linux: http://tldp.org/HOWTO/FTP-3.html

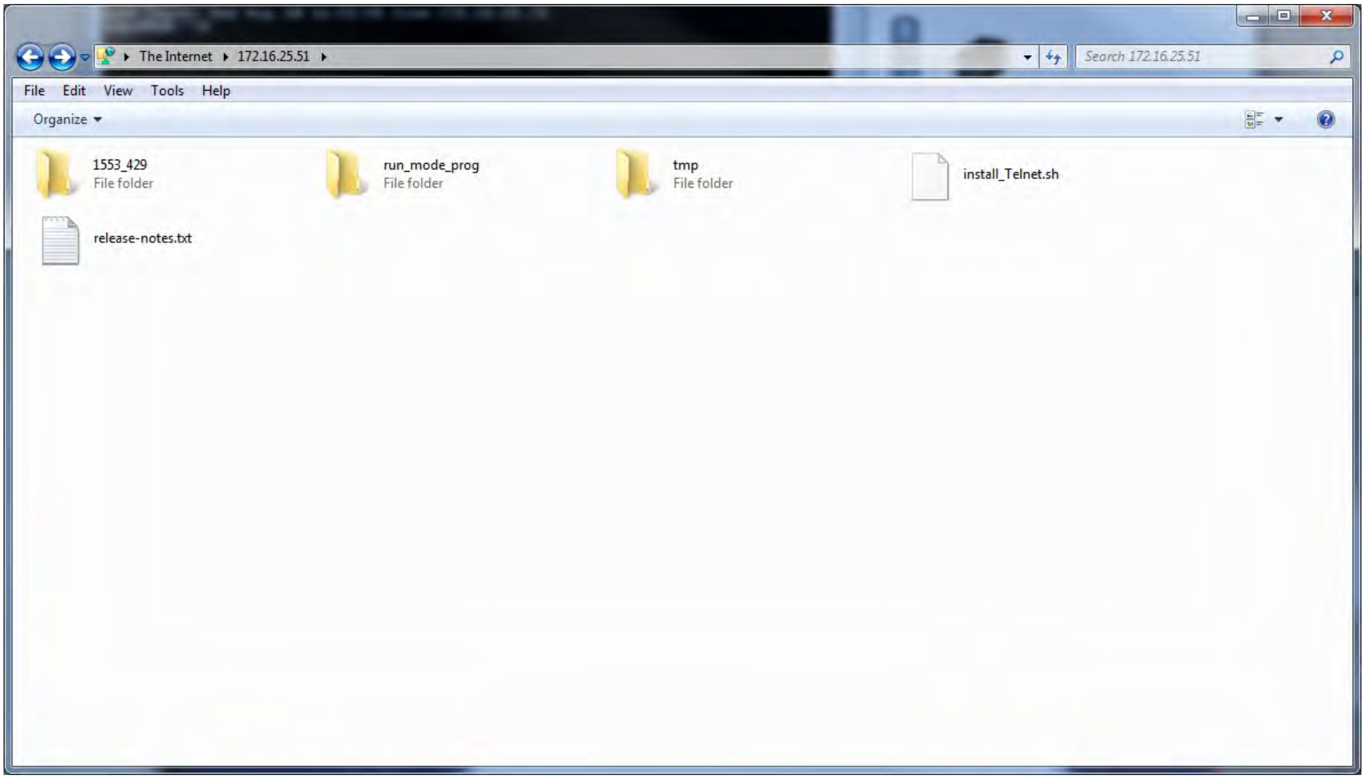


Figure 7. Accessing AIC FTP Server using Windows Client (IE)

6 CONFIGURING NETWORK SERVICES

The Avionics Interface Computer is configured with a Web, Telnet, and FTP server. The Web, Telnet and FTP servers will give the required access to configure, and use the Avionics Interface Computer.

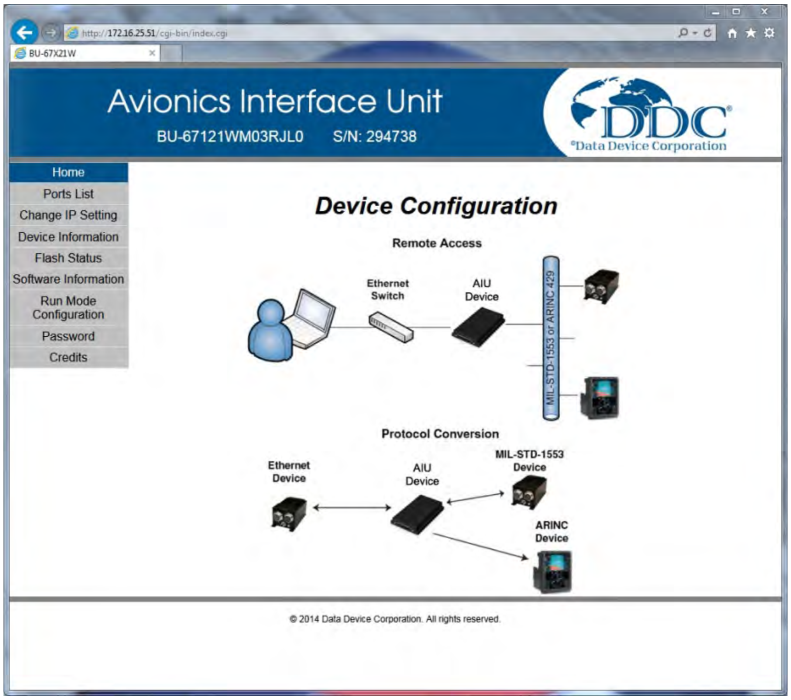
The AIC will ship from the factory with all supported services enabled. Users familiar with UNIX/LINUX shell interfaces should be comfortable using the built-in servers.

6.1 HTTP (Web) Server

The web server can be used to configure the Avionics Interface Computer. The web server will provide information on the number of MIL-STD-1553 channels, ARINC 429 Receiver/Transmitter channels, and the Ethernet Port on the AIC. The web server will also allow the user to modify settings on the AIC, such as the IP address of the Ethernet port, upgrade firmware, drivers or library of the MIL-STD-1553 and ARINC 429 channels.

6.1.1 Connecting to Web Server

To connect to the web server, open a web browser (such as Internet Explorer) and enter in the IP address of your device (see section 4.4) into the address bar of your web browser.



http://172.16.25.51/cgi-bin/index.cgi

Avionics Interface Unit

BU-67121WM03RJL0 S/N: 294738

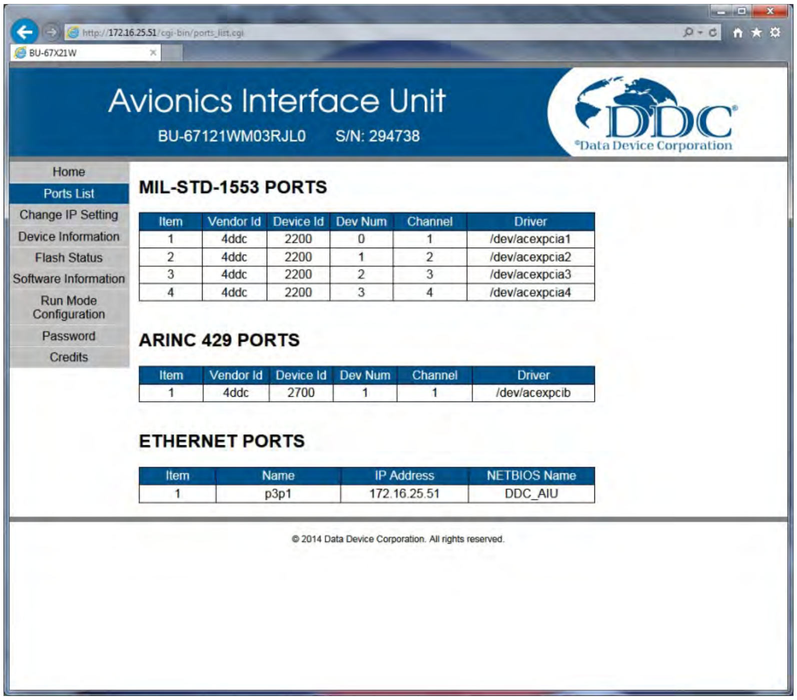
Device Configuration

© 2014 Data Device Corporation. All rights reserved.

Figure 8. Accessing AIC Web Server using Windows Client (IE)

6.1.1 Ports List

The **Ports List** button from the Avionics Interface Computer's Web Server displays the installed ports on the AIC. The ports include the number of MIL-STD-1553 and ARINC 429 channels, which includes their Vendor and Device IDs, and the driver the device is using. The Ethernet Ports on the AIC are also listed under the Ports List link, displaying the Ethernet port name and the current IP address of the AIC, along with the AIC's NETBIOS Name



Avionics Interface Unit

BU-67121WM03RJL0 S/N: 294738

MIL-STD-1553 PORTS

Item	Vendor Id	Device Id	Dev Num	Channel	Driver
1	4ddc	2200	0	1	/dev/acexpcia1
2	4ddc	2200	1	2	/dev/acexpcia2
3	4ddc	2200	2	3	/dev/acexpcia3
4	4ddc	2200	3	4	/dev/acexpcia4

ARINC 429 PORTS

Item	Vendor Id	Device Id	Dev Num	Channel	Driver
1	4ddc	2700	1	1	/dev/acexpcib

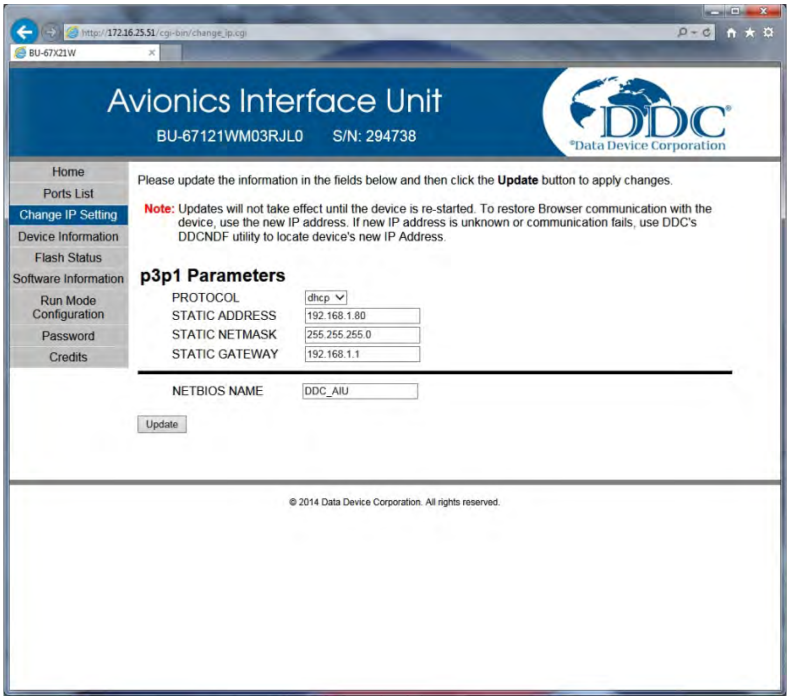
ETHERNET PORTS

Item	Name	IP Address	NETBIOS Name
1	p3p1	172.16.25.51	DDC_AIU

© 2014 Data Device Corporation. All rights reserved.

Figure 9. Ports List from Web Server

The **Change IP Address** button will open a new page that allows the user to configure the IP address of the Avionics Interface Computer.



Please update the information in the fields below and then click the **Update** button to apply changes.

Note: Updates will not take effect until the device is re-started. To restore Browser communication with the device, use the new IP address. If new IP address is unknown or communication fails, use DDC's DDCNDF utility to locate device's new IP Address.

p3p1 Parameters

PROTOCOL	dhcp
STATIC ADDRESS	192.168.1.80
STATIC NETMASK	255.255.255.0
STATIC GATEWAY	192.168.1.1
NETBIOS NAME	DDC_AIU

© 2014 Data Device Corporation. All rights reserved.

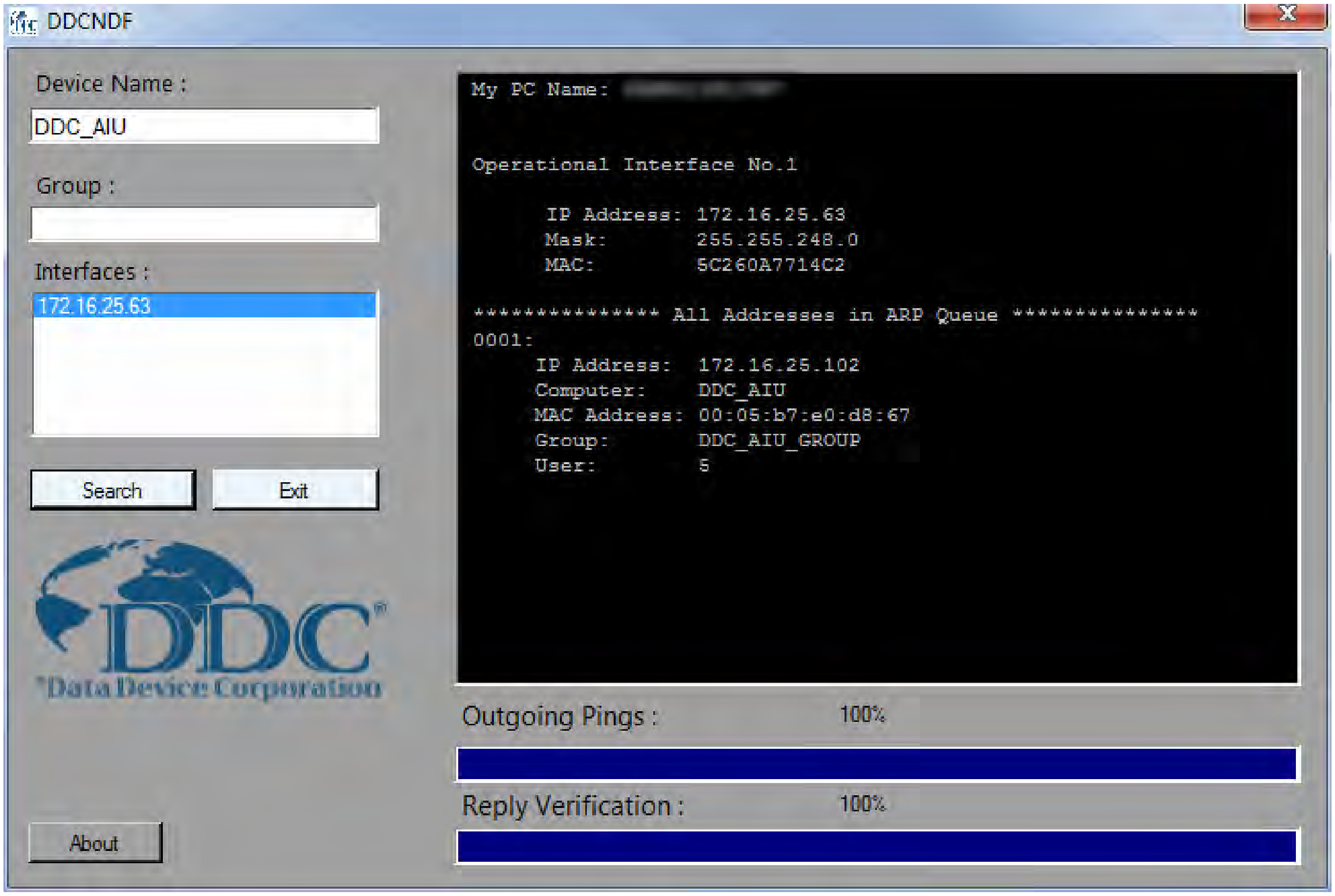
Figure 10. Changing IP Address through Web Server

There is one configurable port on the AIC, the IP Address; netmask and gateway are all configurable on the AIC, when using a Static IP Address. Make sure these settings are configured according to your network settings in order to access the device over your network. If you are using DHCP, the IP Address of the AIC will be generated automatically by the network. The AIC's NETBIOS Name can also be modified in this section. The NETBIOS name will default to DDC_AIC.

6.1.2 DHCP IP Address Locator

The AIC will incorporate static as well as dynamic IP address allocation utilizing DHCP. For dynamic IP address allocation, a utility, **DDCNDF** (DDC Network Device Finder) can be used to find the AIC and retrieve its associated IP address. To find the AIC, the utility uses the DHCP host name request.

The utility **DDCNDF** can be found on the USB **OS Recovery Device** that is shipped with the AIC. The utility is located in the **DDC** folder of the USB OS Recovery Device.



Device Name : DDC_AIU

Group :

Interfaces : 172.16.25.63

```
My PC Name: [redacted]


Operational Interface No.1

      IP Address: 172.16.25.63
      Mask:       255.255.248.0
      MAC:        5C260A7714C2

*************** All Addresses in ARP Queue ***************
0001:
     IP Address:  172.16.25.102
     Computer:    DDC_AIU
     MAC Address: 00:05:b7:e0:d8:67
     Group:       DDC_AIU_GROUP
     User:        5
```

Outgoing Pings : 100%

Reply Verification : 100%

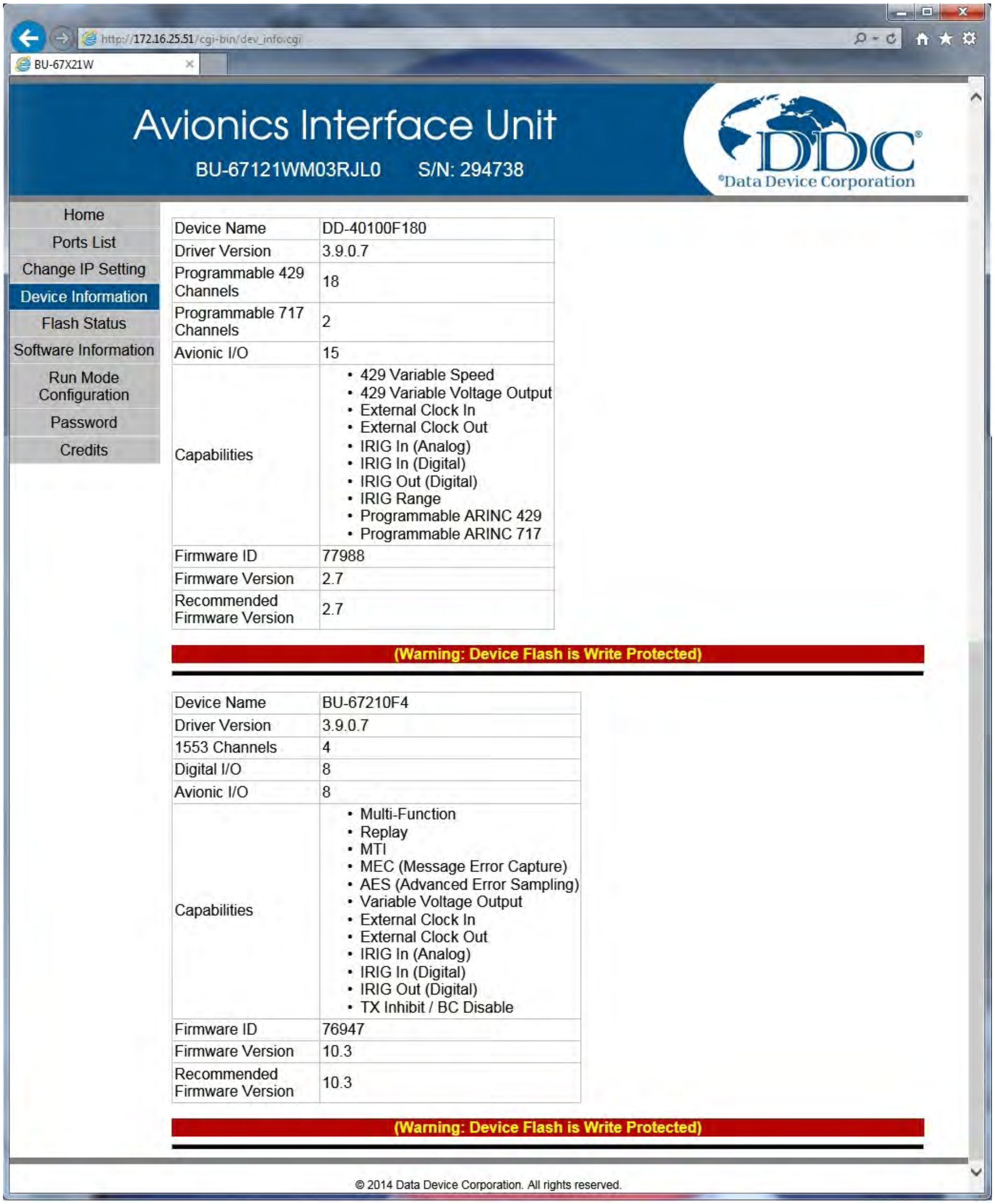
Figure 11. DDCNDF

To use the **DDCNDF** utility, enter in the AIC's Device Name in the **Device Name:** text box. The AIC's default name when shipped from DDC is DDC_AIC. Select your Ethernet interface from the list of **Interfaces:**, and press the Search button. The utility will then perform a search and display and device with the NETBIOS name of AIC and display the AIC's IP address.

***<u>Note:</u>* After finished running the DDCNDF utility, make sure to remove the USB OS Recovery Device from your computer. The OS Recovery Device is bootable, and configured to automatically start the recovery process after a thirty second timeout. Booting your system with the USB OS Recovery Device connected (and USB boot support enabled in your BIOS) will cause the recovery software to perform the recovery operation on your system cause a loss of all your data.**

6.1.3 Device Information

The Device Information page on the Avionics Interface Computer displays the installed MIL-STD-1553 and ARINC 429 devices. The capabilities and driver information for these devices will also be listed on this page.



Device Name	DD-40100F180
Driver Version	3.9.0.7
Programmable 429 Channels	18
Programmable 717 Channels	2
Avionic I/O	15
Capabilities	• 429 Variable Speed • 429 Variable Voltage Output • External Clock In • External Clock Out • IRIG In (Analog) • IRIG In (Digital) • IRIG Out (Digital) • IRIG Range • Programmable ARINC 429 • Programmable ARINC 717
Firmware ID	77988
Firmware Version	2.7
Recommended Firmware Version	2.7

(Warning: Device Flash is Write Protected)

Device Name	BU-67210F4
Driver Version	3.9.0.7
1553 Channels	4
Digital I/O	8
Avionic I/O	8
Capabilities	• Multi-Function • Replay • MTI • MEC (Message Error Capture) • AES (Advanced Error Sampling) • Variable Voltage Output • External Clock In • External Clock Out • IRIG In (Analog) • IRIG In (Digital) • IRIG Out (Digital) • TX Inhibit / BC Disable
Firmware ID	76947
Firmware Version	10.3
Recommended Firmware Version	10.3

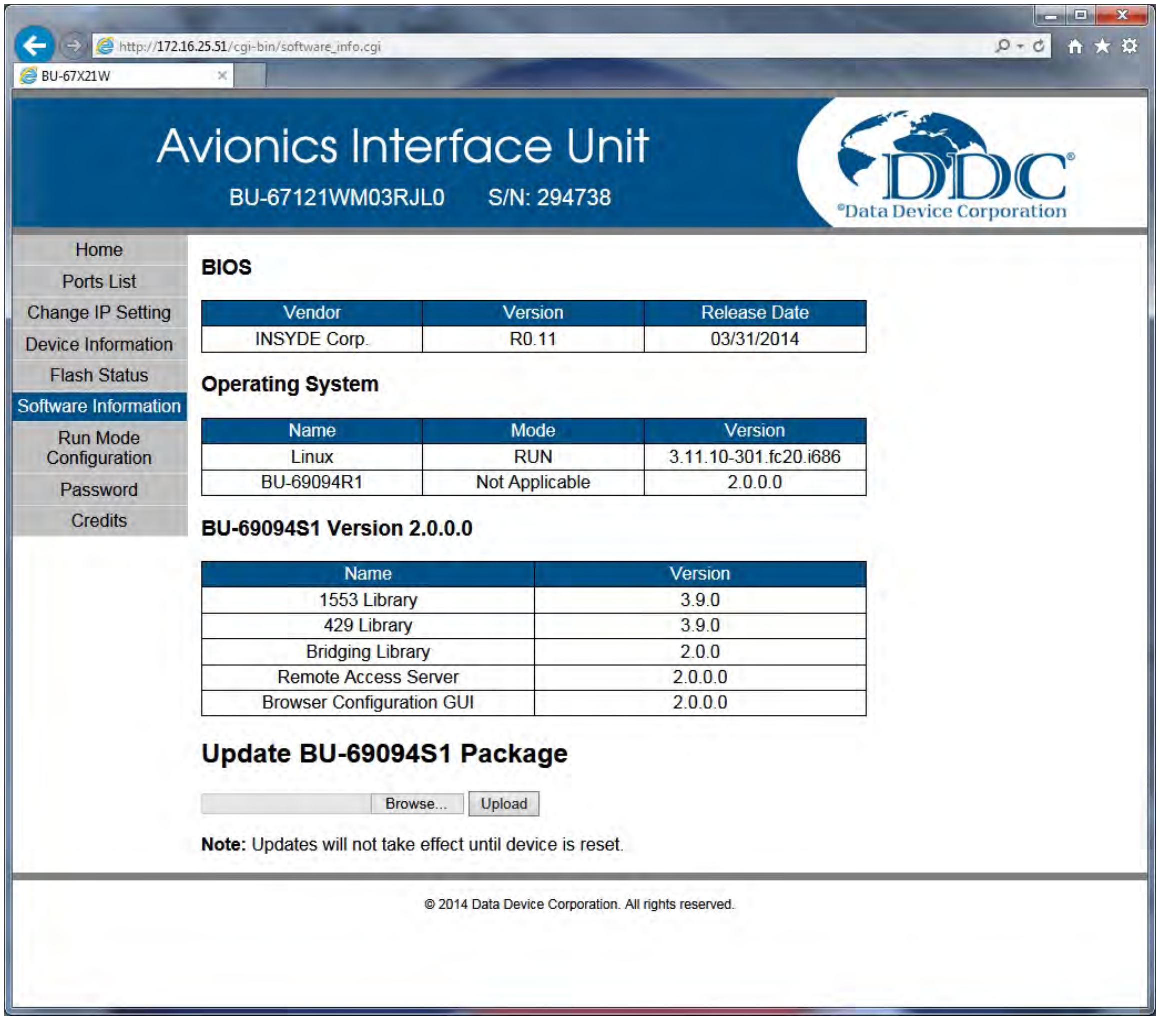
(Warning: Device Flash is Write Protected)

© 2014 Data Device Corporation. All rights reserved.

Figure 12. AIC Device Information

6.1.4 Software Information

The Software Information page on the Avionics Interface Computer displays the installed MIL-STD-1553 and ARINC 429 SDK library versions. This page will also list the Operating System version and the BU-69094R1 version installed on the AIC. The Browse and Upload button can be used to upload a new library and / or driver provided by DDC.



Avionics Interface Unit

BU-67121WM03RJL0 S/N: 294738

BIOS

Vendor	Version	Release Date
INSYDE Corp.	R0.11	03/31/2014

Operating System

Name	Mode	Version
Linux	RUN	3.11.10-301.fc20.i686
BU-69094R1	Not Applicable	2.0.0.0

BU-69094S1 Version 2.0.0.0

Name	Version
1553 Library	3.9.0
429 Library	3.9.0
Bridging Library	2.0.0
Remote Access Server	2.0.0.0
Browser Configuration GUI	2.0.0.0

Update BU-69094S1 Package

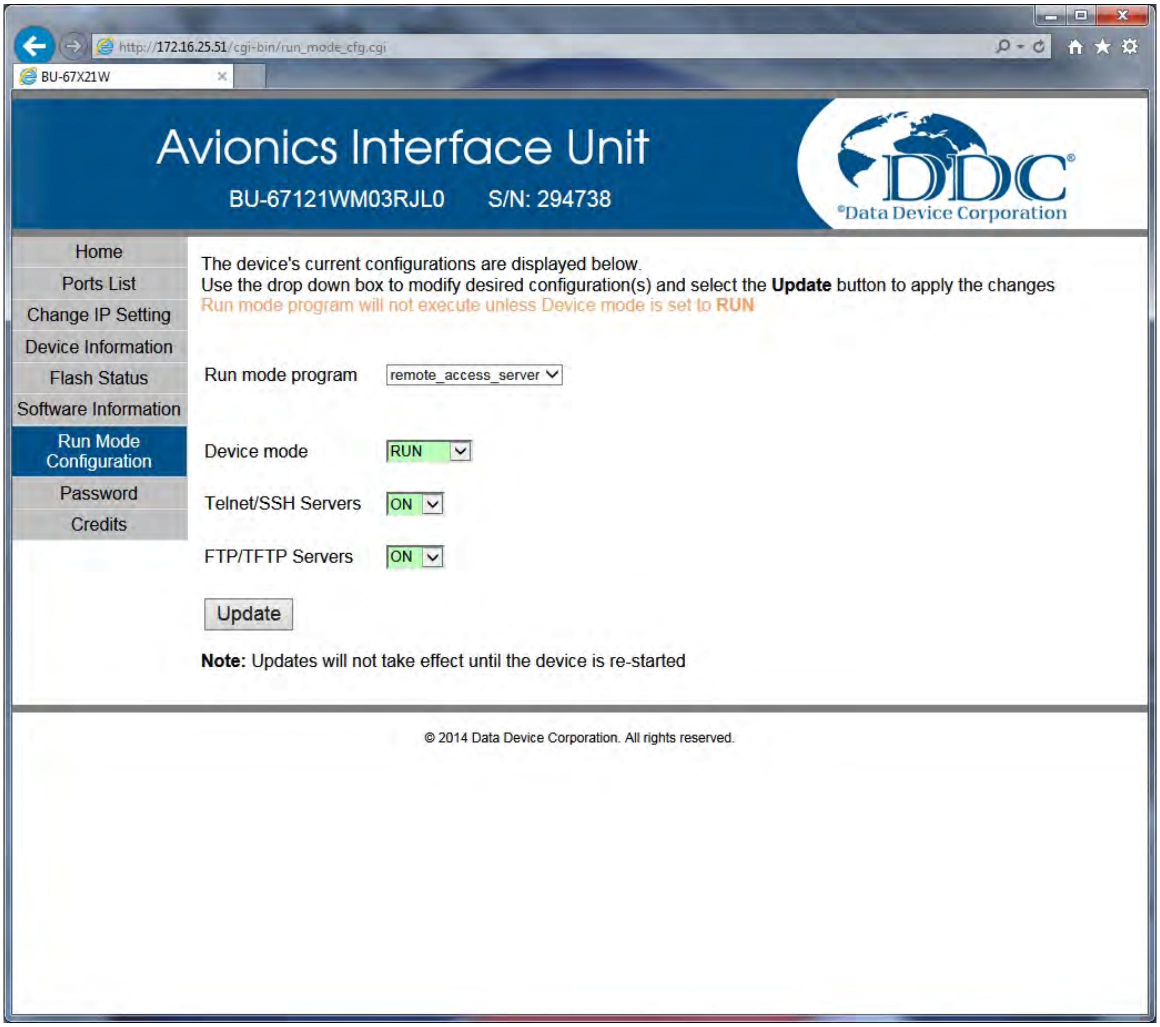
Note: Updates will not take effect until device is reset.

© 2014 Data Device Corporation. All rights reserved.

Figure 13. AIC Software Information

6.1.5 Run Mode Configuration

The **Run Mode Configuration** page on the Avionics Interface Computer displays the current status of Run Mode Program and which services are on/off while the device is in ‘Run Mode’.



Avionics Interface Unit

BU-67121WM03RJL0 S/N: 294738

The device's current configurations are displayed below.
Use the drop down box to modify desired configuration(s) and select the **Update** button to apply the changes
Run mode program will not execute unless Device mode is set to **RUN**

Run mode program: remote_access_server

Device mode: RUN

Telnet/SSH Servers: ON

FTP/TFTP Servers: ON

Note: Updates will not take effect until the device is re-started

© 2014 Data Device Corporation. All rights reserved.

Figure 14. AIC Run Mode Configuration

The user can select which program to start when the device is powered on while in run mode. The Web Server, Telnet Server, and FTP Server can also be turned On or Off.

6.2 Telnet/SSH Server

DDC's Avionics Interface Computer will ship from the factory with the Telnet/SSH Server enabled. The use of the Telnet server will allow the user to connect directly to the Avionics Interface Computer in order to interface with the Linux Operating System. In order to use the Telnet Server, the host system must have a Telnet client installed, and the service must be enabled on the host Operating System.

6.2.1 Enabling Telnet Windows 7 and Windows 8

The Telnet service is not enabled by default under Windows 7 and Windows 8. To use Telnet on a Windows 7 / 8, the Telnet Windows Feature will have to be enabled by opening the **Windows Control Panel** and selecting **Programs and Features**.

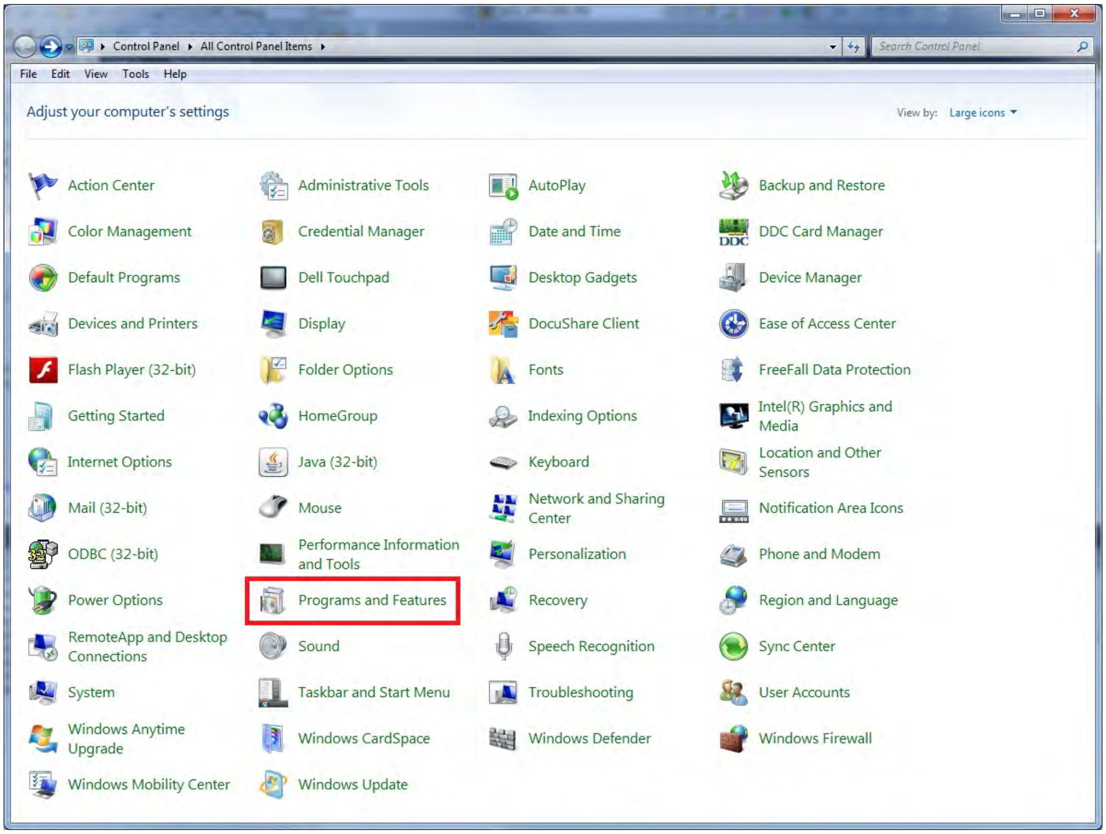
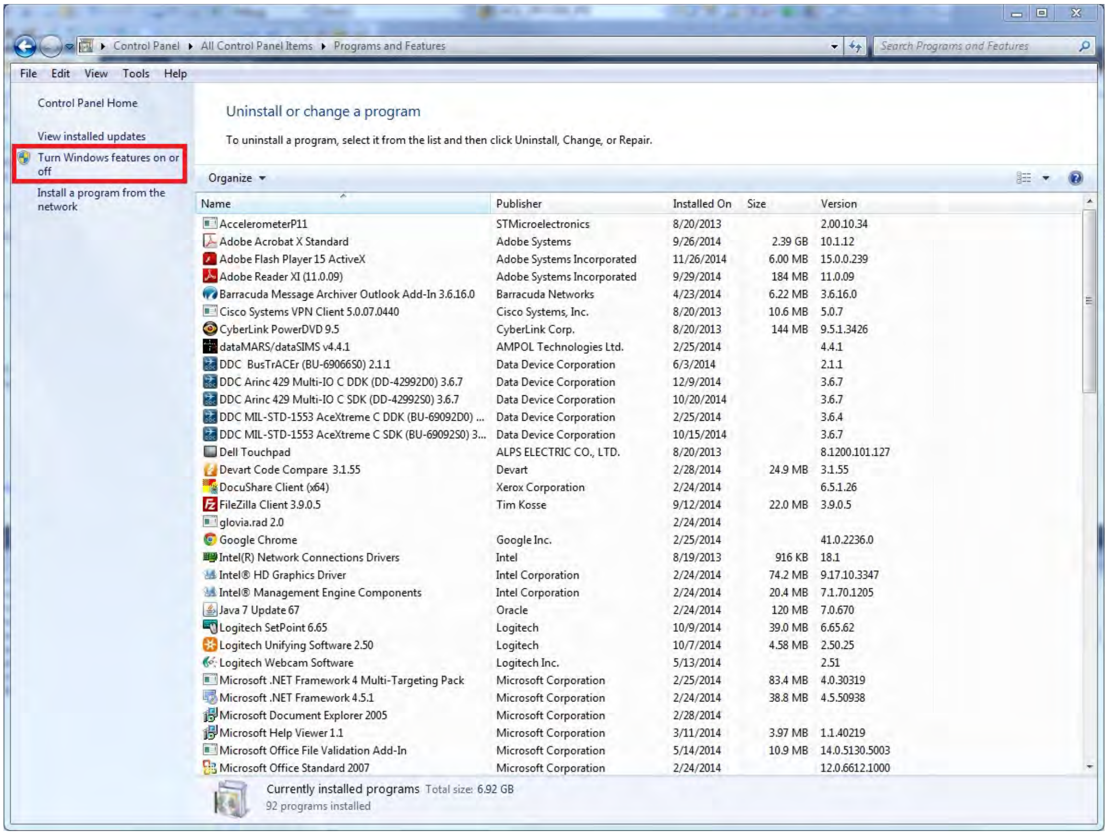


Figure 15. Programs and Features – Control Panel

Once the **Programs and Features** has been selected and opened, click on the link for **Turn Windows Features on or off**.



Control Panel ▸ All Control Panel Items ▸ Programs and Features
Search Programs and Features
File Edit View Tools Help
Control Panel Home
View installed updates
Turn Windows features on or off
Install a program from the network

Uninstall or change a program

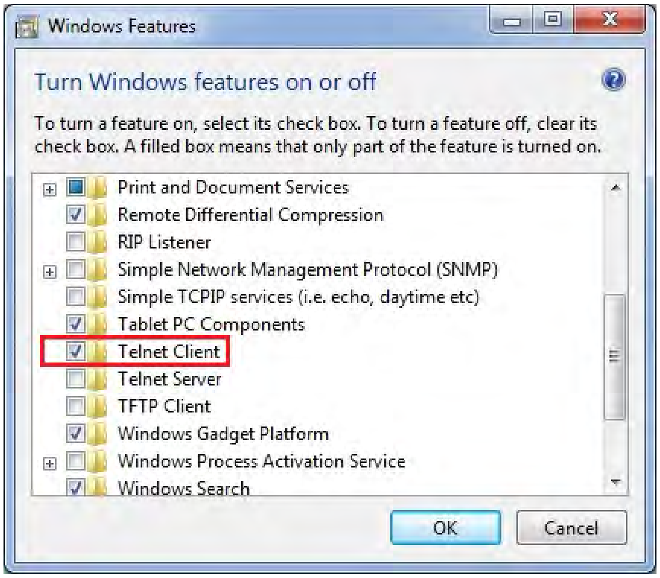
To uninstall a program, select it from the list and then click Uninstall, Change, or Repair.

Name	Publisher	Installed On	Size	Version
AccelerometerP11	STMicroelectronics	8/20/2013		2.00.10.34
Adobe Acrobat X Standard	Adobe Systems	9/26/2014	2.39 GB	10.1.12
Adobe Flash Player 15 ActiveX	Adobe Systems Incorporated	11/26/2014	6.00 MB	15.0.0.239
Adobe Reader XI (11.0.09)	Adobe Systems Incorporated	9/29/2014	184 MB	11.0.09
Barracuda Message Archiver Outlook Add-In 3.6.16.0	Barracuda Networks	4/23/2014	6.22 MB	3.6.16.0
Cisco Systems VPN Client 5.0.07.0440	Cisco Systems, Inc.	8/20/2013	10.6 MB	5.0.7
CyberLink PowerDVD 9.5	CyberLink Corp.	8/20/2013	144 MB	9.5.1.3426
dataMARS/dataSIMS v4.4.1	AMPOL Technologies Ltd.	2/25/2014		4.4.1
DDC BusTrACEr (BU-69066S0) 2.1.1	Data Device Corporation	6/3/2014		2.1.1
DDC Arinc 429 Multi-IO C DDK (DD-42992D0) 3.6.7	Data Device Corporation	12/9/2014		3.6.7
DDC Arinc 429 Multi-IO C SDK (DD-42992S0) 3.6.7	Data Device Corporation	10/20/2014		3.6.7
DDC MIL-STD-1553 AceXtreme C DDK (BU-69092D0) ...	Data Device Corporation	2/25/2014		3.6.4
DDC MIL-STD-1553 AceXtreme C SDK (BU-69092S0) 3...	Data Device Corporation	10/15/2014		3.6.7
Dell Touchpad	ALPS ELECTRIC CO., LTD.	8/20/2013		8.1200.101.127
Devart Code Compare 3.1.55	Devart	2/28/2014	24.9 MB	3.1.55
DocuShare Client (x64)	Xerox Corporation	2/24/2014		6.5.1.26
FileZilla Client 3.9.0.5	Tim Kosse	9/12/2014	22.0 MB	3.9.0.5
glovia.rad 2.0		2/24/2014		
Google Chrome	Google Inc.	2/25/2014		41.0.2236.0
Intel(R) Network Connections Drivers	Intel	8/19/2013	916 KB	18.1
Intel® HD Graphics Driver	Intel Corporation	2/24/2014	74.2 MB	9.17.10.3347
Intel® Management Engine Components	Intel Corporation	2/24/2014	20.4 MB	7.1.70.1205
Java 7 Update 67	Oracle	2/24/2014	120 MB	7.0.670
Logitech SetPoint 6.65	Logitech	10/9/2014	39.0 MB	6.65.62
Logitech Unifying Software 2.50	Logitech	10/7/2014	4.58 MB	2.50.25
Logitech Webcam Software	Logitech Inc.	5/13/2014		2.51
Microsoft .NET Framework 4 Multi-Targeting Pack	Microsoft Corporation	2/25/2014	83.4 MB	4.0.30319
Microsoft .NET Framework 4.5.1	Microsoft Corporation	2/24/2014	38.8 MB	4.5.50938
Microsoft Document Explorer 2005	Microsoft Corporation	2/28/2014		
Microsoft Help Viewer 1.1	Microsoft Corporation	3/11/2014	3.97 MB	1.1.40219
Microsoft Office File Validation Add-In	Microsoft Corporation	5/14/2014	10.9 MB	14.0.5130.5003
Microsoft Office Standard 2007	Microsoft Corporation	2/24/2014		12.0.6612.1000

Currently installed programs Total size: 6.92 GB
92 programs installed

Figure 16. Turn Windows Features on or off

From the Windows Features window, scroll down the list of features while looking for **Telnet Client**.



Turn Windows features on or off

To turn a feature on, select its check box. To turn a feature off, clear its check box. A filled box means that only part of the feature is turned on.

- ⊞ ■ Print and Document Services
- ☑ Remote Differential Compression
- □ RIP Listener
- ⊞ □ Simple Network Management Protocol (SNMP)
- □ Simple TCPIP services (i.e. echo, daytime etc)
- ☑ Tablet PC Components
- ☑ Telnet Client
- □ Telnet Server
- □ TFTP Client
- ☑ Windows Gadget Platform
- ⊞ □ Windows Process Activation Service
- ☑ Windows Search

Figure 17. Windows Features – Telnet

Once found, click the check box next to **Telnet Client** to enable the Windows feature, and click the **OK** button. Windows will then install the service, and the Windows Features Window will be closed. To confirm the feature has been installed open a command prompt and type telnet /?. The help feature for telnet will be displayed indicating it has been properly installed.

6.3 FTP/TFTP Server

DDC's Avionics Interface Computer will ship from the factory with the FTP/TFTP Server enabled. The use of the FTP/TFTP server will allow the user to connect directly to the Avionics Interface Computer in order to transfer file on/off the AIC. In order to use the FTP/TFTP Server, the host system must have a FTP/TFTP client installed.

6.3.1 Connecting to AIC via FTP

To connect to the Avionics Interface Computer over FTP, open your FTP client and enter the IP address of the AIC. Once you have connected to the AIC with your FTP client, you will be prompted to enter the username and password of your account on the AIC. Please refer to Section 5.1 for the default username and password on the Avionics Interface Computer.

7 USING THE PROTOCOL CONVERSION LAYER

The Protocol Conversion Layer (PCL) is a software middleware layer that allows the user to easily transfer data from one protocol to another.

It supports user-defined bridging of MIL-STD-1553, ARINC 429 and Ethernet traffic in any combination and direction.

Bridging is achieved by developing an application using the PCL to define what specific data should be transferred. The user can define the source protocol/message and the destination protocol/message; the PCL will autonomously copy the data as defined.

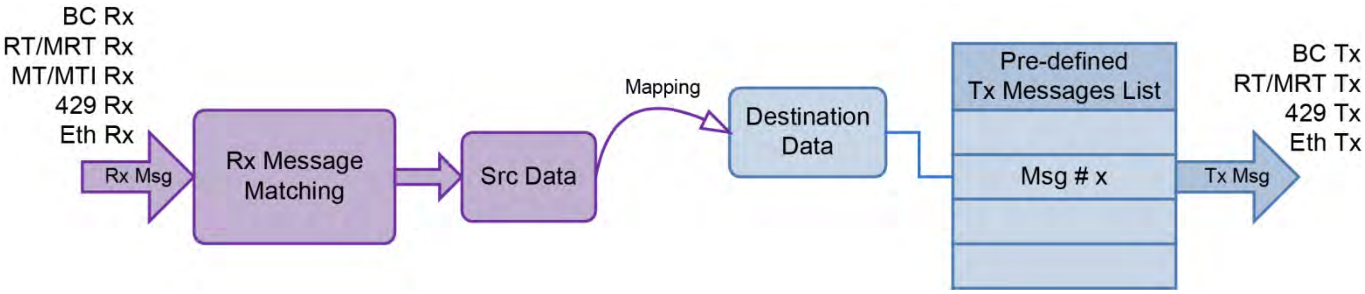


Figure 18. Protocol Conversion Layer Flow Diagram

The Protocol Conversion Layer is supplied as ANSI 'C' full source code and leverages DDC's MIL-STD-1553 and ARINC 429 Software Development Kits. This allows users unique customizations to complete their desired Application.

7.1 Forwarding a port through Windows Firewall

To enable the use of the bridging samples, you must first open the port "0x4ddc" in your firewall. The bridging samples on the AIC are configured to use port 0x4ddc by default. This port can be changed by the user by modifying the source of the sample and recompiling. To enable the port used by the AIC for Protocol Conversion, open the Windows Firewall settings located in the Windows Control Panel.

7.1.1 Enabling the port through the Windows Firewall Settings

First navigate to the **Control Panel** and find **Windows Firewall**.

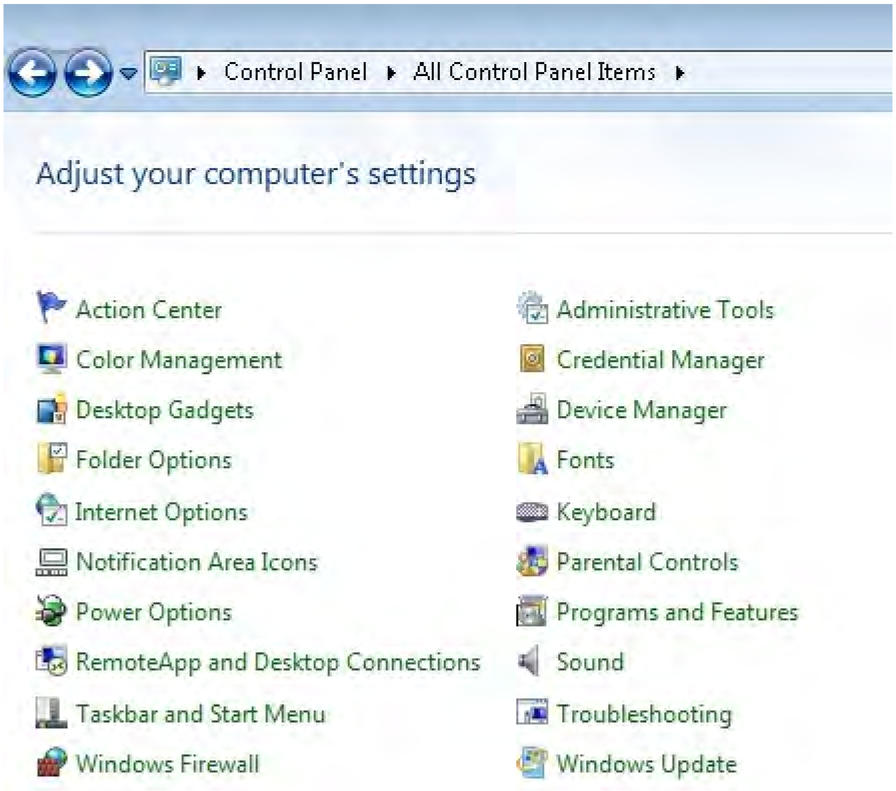
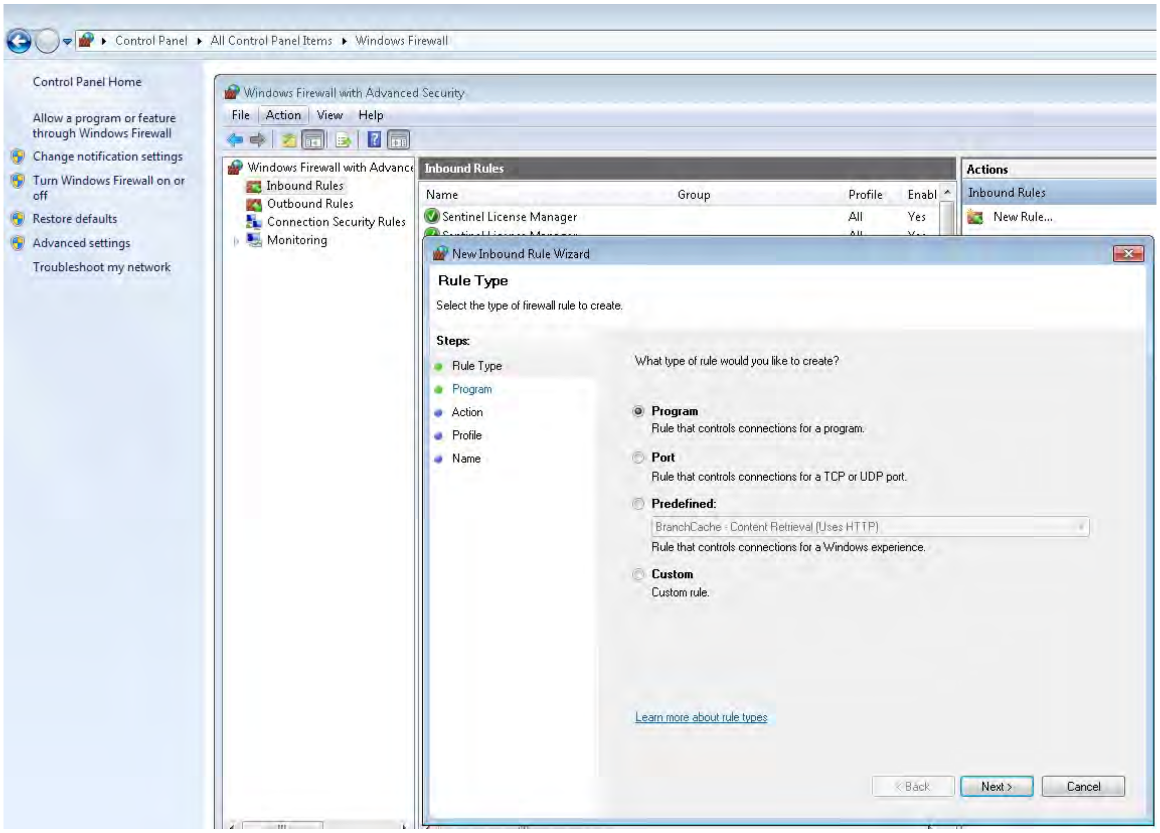


Figure 19. Windows Control Panel

1. Click **Advanced Settings** on the left side of the Windows Firewall dialog window. A new dialog window will open displaying the Windows Firewall with Advanced Security settings.

2. Click **Inbound Rules** on the left side of the Windows Firewall with Advanced Security window, and select **New Rule**, on the right side of the dialog window.

3. The **New Inbound Rule Wizard** will open.



Rule Type

Select the type of firewall rule to create.

What type of rule would you like to create?

● Program
Rule that controls connections for a program.

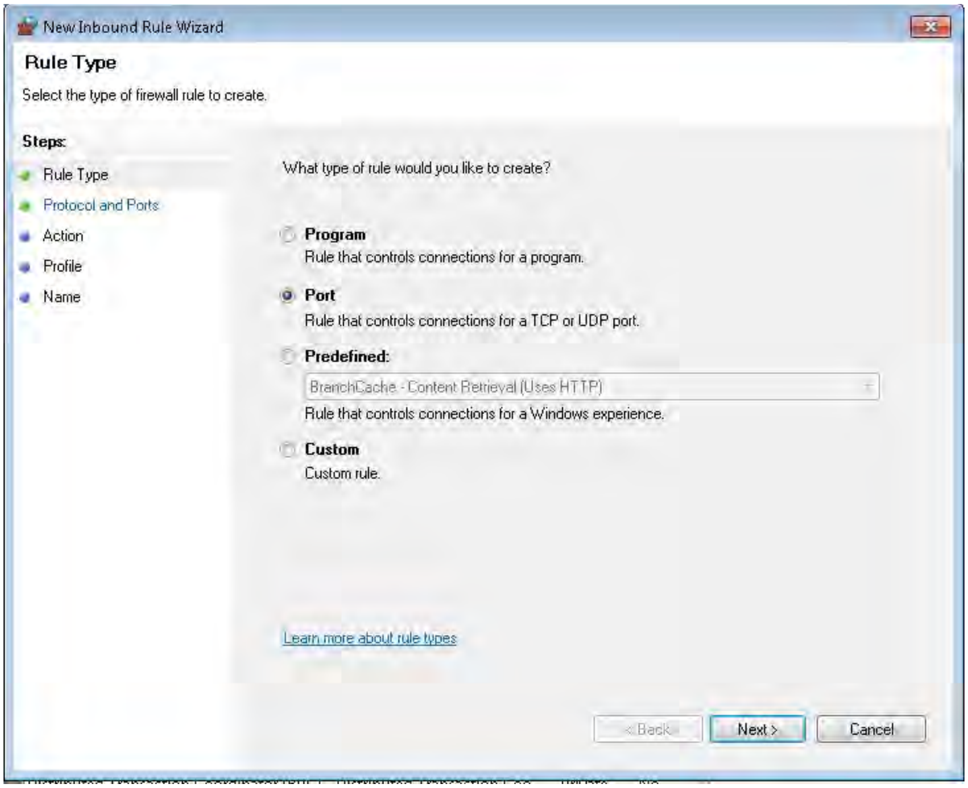
○ Port
Rule that controls connections for a TCP or UDP port.

○ Predefined:
BranchCache - Content Retrieval (Uses HTTP)
Rule that controls connections for a Windows experience.

○ Custom
Custom rule.

Figure 20. New Inbound Rule Wizard

4. In the Wizard, under the **Rule Type** step, select **Port**, and click **Next**.



New Inbound Rule Wizard

Rule Type

Select the type of firewall rule to create.

Steps:

- Rule Type
- Protocol and Ports
- Action
- Profile
- Name

What type of rule would you like to create?

○ **Program**
Rule that controls connections for a program.

● **Port**
Rule that controls connections for a TCP or UDP port.

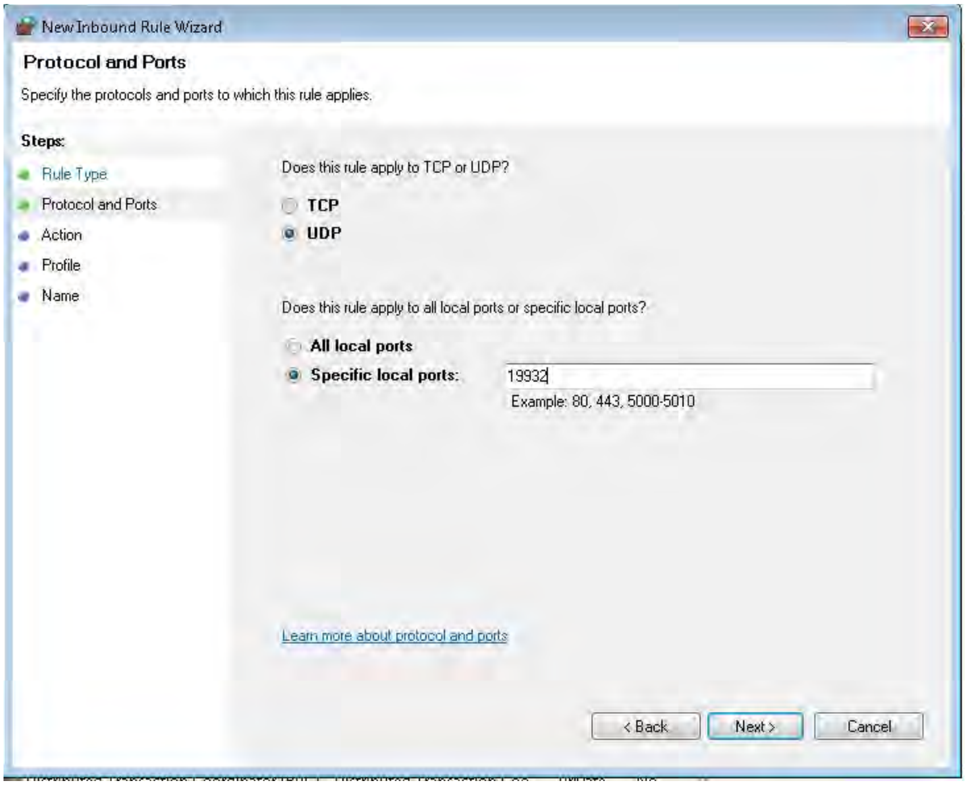
○ **Predefined:**
BranchCache - Content Retrieval (Uses HTTP)
Rule that controls connections for a Windows experience.

○ **Custom**
Custom rule.

Learn more about rule types

Figure 21. Windows Firewall Rule Type

5. Under the **Protocol and Ports** step, select **UDP**. Under **Specific local ports**, type 19932. This is the hex value 0x4DDC port number converted to decimal. Once the value has been entered click **Next** to move onto the next step.



Protocol and Ports

Specify the protocols and ports to which this rule applies.

Does this rule apply to TCP or UDP?

○ TCP

● UDP

Does this rule apply to all local ports or specific local ports?

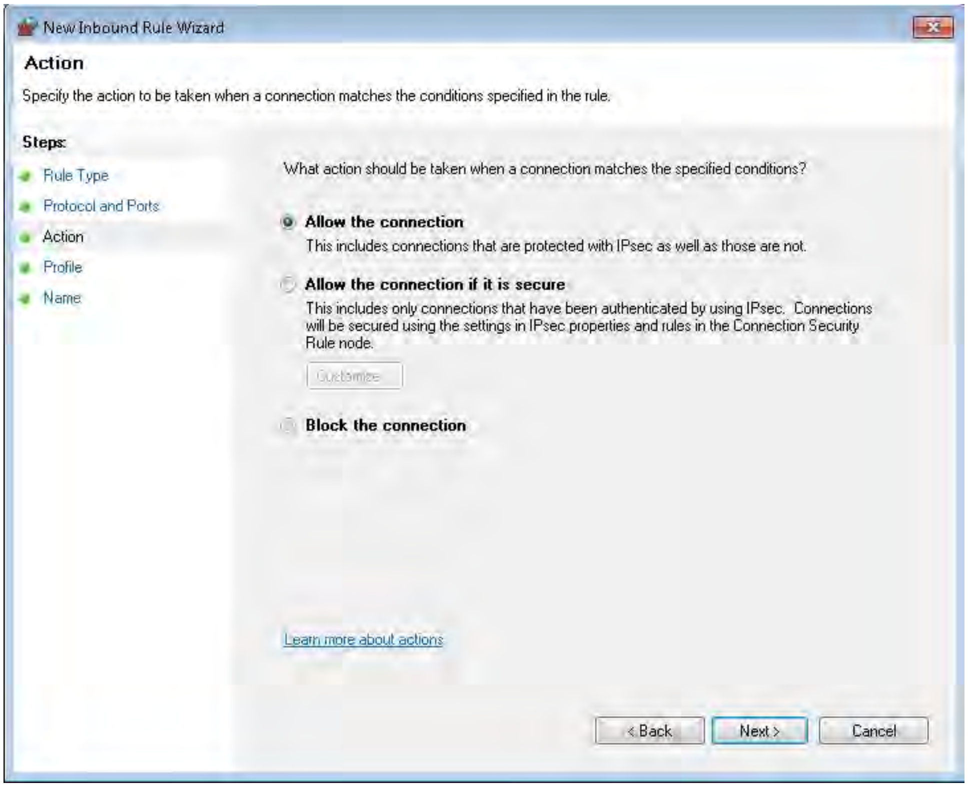
○ All local ports

● Specific local ports: 19932

Example: 80, 443, 5000-5010

Figure 22. Windows Firewall Protocol and Ports

6. The next step is to select the **Action** type. The **Allow the connection** action should be selected. Once selected, click **Next** to move onto the next step.



Action

Specify the action to be taken when a connection matches the conditions specified in the rule.

What action should be taken when a connection matches the specified conditions?

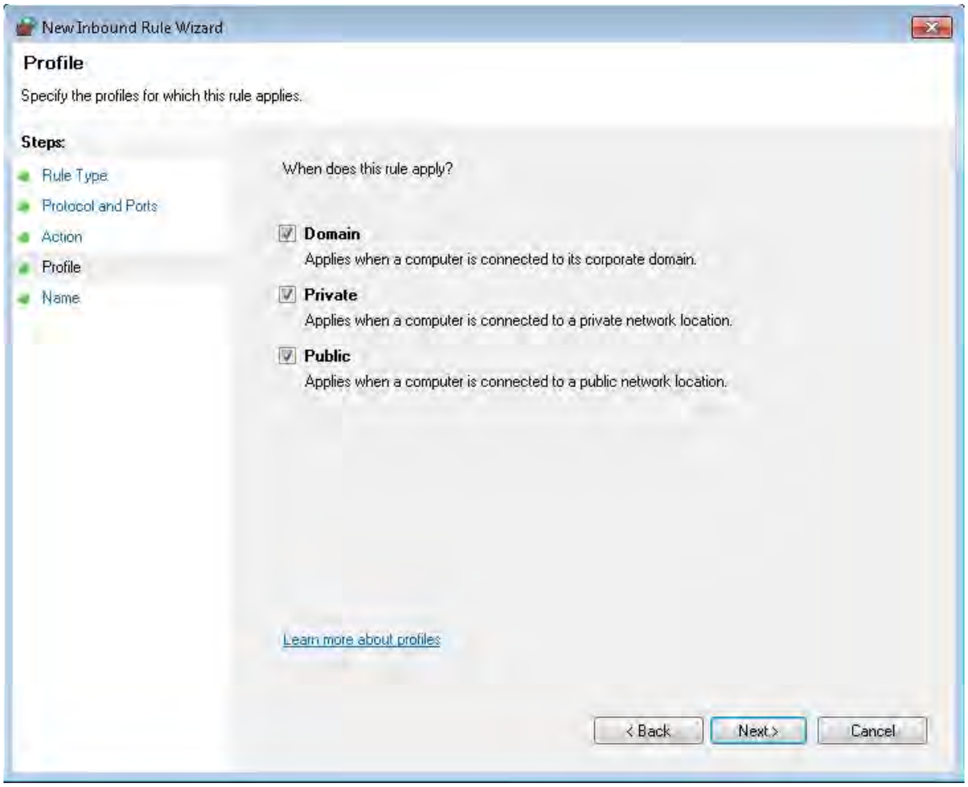
● **Allow the connection**
This includes connections that are protected with IPsec as well as those are not.

○ **Allow the connection if it is secure**
This includes only connections that have been authenticated by using IPsec. Connections will be secured using the settings in IPsec properties and rules in the Connection Security Rule node.

○ **Block the connection**

Figure 23. Windows Firewall Action selection

7. Under the **Profile** step, all of the options should be enabled. Click **Next**.



New Inbound Rule Wizard

Profile

Specify the profiles for which this rule applies.

Steps:

- Rule Type
- Protocol and Ports
- Action
- Profile
- Name

When does this rule apply?

☑ **Domain**
Applies when a computer is connected to its corporate domain.

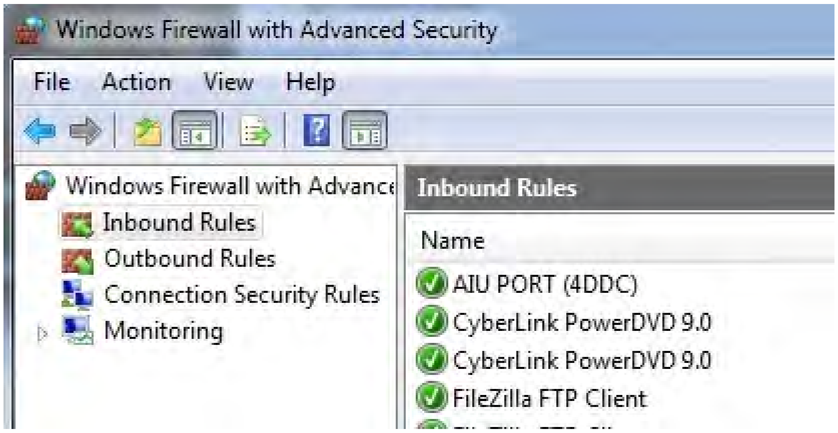
☑ **Private**
Applies when a computer is connected to a private network location.

☑ **Public**
Applies when a computer is connected to a public network location.

Learn more about profiles

Figure 24. Windows Firewall Profile

8. In the **Name** field, you can name the new rule anyway you see fit. This example uses AIC PORT (4DDC). There is also a field for an optional description. Click **Finish** to complete the new rule and have it added to your Windows Firewall settings.



Windows Firewall with Advanced Security

Inbound Rules

Name
AIU PORT (4DDC)
CyberLink PowerDVD 9.0
CyberLink PowerDVD 9.0
FileZilla FTP Client

Figure 25. Windows Firewall Inbound Rules

7.2 Compiling Applications

All DDC Avionics Interface Computers contain a complete target complier to compile and build any supplied samples or to compile any custom user applications.

Compiler Version: gcc version 4.8.2, (Fedora 20).

To build the samples, first navigate to the PCL Samples directory:

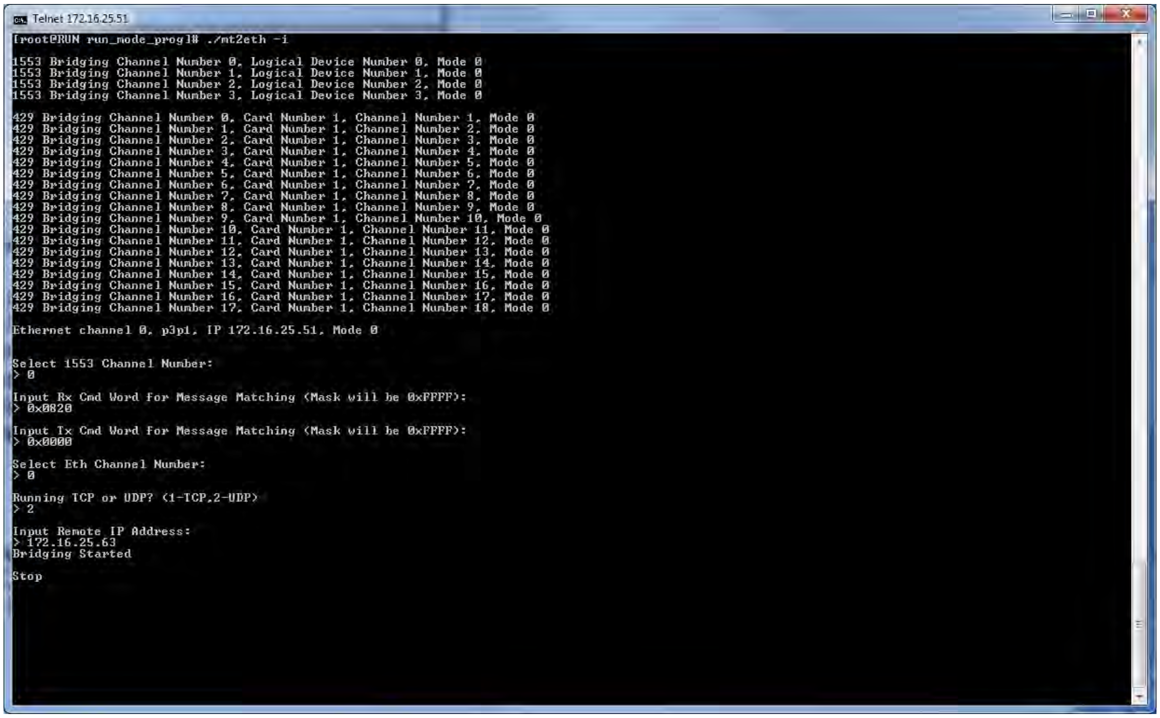
cd /home/ddc/1553_492/samples/bridging/prj/

Then type **make** to build the supplied samples. The binaries of the samples will be located in the /home/<USER>/run_mode_prog directory.

In order to use the sample applications with the settings of your network, make sure the same is executed with the **'-i'** option. This option allows a user to input the necessary information from the way their system is setup. This avoids having to manually edit the sample and recompile.

Figure 26 shows the execution of the mt2eth Protocol Conversion sample (with the **'-i'** option). The sample will first query the user for the logical device number (Select 1553 Channel number) to be used with the sample. The next input parameter is the receive command a user would want to forward (Input Rx Cmd Word for Message Matching). The third input parameter is the specific transmit command that is forward (Input Tx Cmd Word for Message Matching).

The sample will also request the Ethernet Port of the AIC you are going to use (Select Eth Channel Number), if the sample uses TCP or UDP, and the IP address of the remote system.



```
[root@RUN run_mode_prog]# ./mt2eth -i

1553 Bridging Channel Number 0, Logical Device Number 0, Mode 0
1553 Bridging Channel Number 1, Logical Device Number 1, Mode 0
1553 Bridging Channel Number 2, Logical Device Number 2, Mode 0
1553 Bridging Channel Number 3, Logical Device Number 3, Mode 0

429 Bridging Channel Number 0, Card Number 1, Channel Number 1, Mode 0
429 Bridging Channel Number 1, Card Number 1, Channel Number 2, Mode 0
429 Bridging Channel Number 2, Card Number 1, Channel Number 3, Mode 0
429 Bridging Channel Number 3, Card Number 1, Channel Number 4, Mode 0
429 Bridging Channel Number 4, Card Number 1, Channel Number 5, Mode 0
429 Bridging Channel Number 5, Card Number 1, Channel Number 6, Mode 0
429 Bridging Channel Number 6, Card Number 1, Channel Number 7, Mode 0
429 Bridging Channel Number 7, Card Number 1, Channel Number 8, Mode 0
429 Bridging Channel Number 8, Card Number 1, Channel Number 9, Mode 0
429 Bridging Channel Number 9, Card Number 1, Channel Number 10, Mode 0
429 Bridging Channel Number 10, Card Number 1, Channel Number 11, Mode 0
429 Bridging Channel Number 11, Card Number 1, Channel Number 12, Mode 0
429 Bridging Channel Number 12, Card Number 1, Channel Number 13, Mode 0
429 Bridging Channel Number 13, Card Number 1, Channel Number 14, Mode 0
429 Bridging Channel Number 14, Card Number 1, Channel Number 15, Mode 0
429 Bridging Channel Number 15, Card Number 1, Channel Number 16, Mode 0
429 Bridging Channel Number 16, Card Number 1, Channel Number 17, Mode 0
429 Bridging Channel Number 17, Card Number 1, Channel Number 18, Mode 0

Ethernet channel 0, p3p1, IP 172.16.25.51, Mode 0


Select 1553 Channel Number:
> 0

Input Rx Cmd Word for Message Matching (Mask will be 0xFFFF):
> 0x0820

Input Tx Cmd Word for Message Matching (Mask will be 0xFFFF):
> 0x0000

Select Eth Channel Number:
> 0

Running TCP or UDP? (1-TCP,2-UDP)
> 2

Input Remote IP Address:
> 172.16.25.63
Bridging Started

Stop
```

Figure 26. Running Protocol Conversion Sample

7.3 Procotol Conversion API Dictionary

The following API functions and structures encompass the Protocol Conversion Layer. Fully-functional User Samples are also supplied to show expected usage (See Section 7.5). The valid values for each the MIL-STD-1553 and ARINC 429 channel will depend on the individual AIC unit being used, and its respective channel count.

Table 2. Protocol Bridging Layer (PBL) API Functions

Function Name	Category	Page
abd1553AddMappingRule	MIL-STD-1553	77
abd1553AddMatchingEntry	MIL-STD-1553	75
abd1553BcAddMsgToList	MIL-STD-1553	69
abd1553BcCreateMsg	MIL-STD-1553	67
abd1553EnableRT	MIL-STD-1553	63
abd1553EnableSA	MIL-STD-1553	65
abd1553Free	MIL-STD-1553	46
abd1553Initialize	MIL-STD-1553	45
abd1553Setup	MIL-STD-1553	51
abd1553Start	MIL-STD-1553	57
abd1553Stop	MIL-STD-1553	58
abd429AddMappingRule	ARINC 429	82
abd429AddMatchingEntry	ARINC 429	80
abd429AddTxMsg	ARINC 429	71
abd429Free	ARINC 429	48
abd429Initialize	ARINC 429	47
abd429Setup	ARINC 429	53
abd429Start	ARINC 429	59
abd429Stop	ARINC 429	60
abdEthAddMappingRule	Ethernet	87
abdEthAddMatchingEntry	Ethernet	85
abdEthAddTxMsg	Ethernet	73
abdEthFree	Ethernet	50
abdEthInitialize	Ethernet	49
abdEthSetup	Ethernet	55
abdEthStart	Ethernet	61
abdEthStop	Ethernet	62
abdFree	All Modes	44
abdInitialize	All Modes	43

abdInitialize

This function will initialize all Protocols and Channels for use with Protocol Bridging.

PROTOTYPE

#include “abdDev.h”
S16BIT abdInitialize(void);

HARDWARE

BU-67X21WX

STATE

RESET

MODE

Any

PARAMETERS

None

DESCRIPTION

This function will initialize all populated protocol (1553, 429, Ethernet) channels and place them in the SETUP state.

RETURN VALUE

Negative Value = Error
0 = Success

EXAMPLE

```
S16BIT wResult = 0;

/* Initialize All Channels */
if((wResult = abdInitialize())
{
    printf(“abdInitialize Failed.  Error Code: %d”, wResult);
}
```

SEE ALSO

abd1553Initialize() **abd429Initialize()**
abdEthInitialize()

abdFree

This function will free (reset) all Protocols and Channels to an uninitialized state.

PROTOTYPE

#include “abdDev.h”
S16BIT abdFree(void);

HARDWARE

BU-67X21WX

STATE

RESET, SETUP, RUN

MODE

Any

PARAMETERS

None

DESCRIPTION

This function will stop and reset all populated protocol (1553, 429, Ethernet) channels and return them in the RESET state.

RETURN VALUE

Negative Value = Error
0 = Success

EXAMPLE

```
S16BIT wResult = 0;

/* Free All Channels */
if((wResult = abdFree())
{
    printf(“abdFree Failed.  Error Code: %d”,wResult);
}
```

SEE ALSO

abd1553Free()	**abd429Free()**
abdEthFree()	

abd1553Initialize

This function will initialize a MIL-STD-1553 channel to an uninitialized state.

PROTOTYPE

#include “setup1553.h”
S16BIT abd1553Free(U16BIT u16Abd1553Chnl);

HARDWARE

BU-67X21WX

STATE

RESET, SETUP, RUN

MODE

Any

PARAMETERS

u16Abd1553Chnl	(input parameter) MIL-STD-1553 Channel Number Valid Values: 0 – 31

DESCRIPTION

This function will initialize a specific MIL-STD-1553 channel for protocol bridging and place it in the SETUP state.

RETURN VALUE

Negative Value = Error
0 = Success

EXAMPLE

```
S16BIT wResult = 0;
U16BIT w1553Chnl = 0;

/* Initialize 1553 Channel */
if((wResult = abd1553Initialize(w1553Chnl))
{
    printf(“abd1553Initialize    %d    Failed.    Error    Code:    %d”,
w1553Chnl,wResult);
}
```

SEE ALSO

abdInitialize()	**abd429Initialize()**
abdEthInitialize()	

abd1553Free

This function will reset a MIL-STD-1553 channel to an uninitialized state.

PROTOTYPE

#include “setup1553.h”
S16BIT abd1553Free(U16BIT u16Abd1553Chnl);

HARDWARE

BU-67X21WX

STATE

RESET

MODE

Any

PARAMETERS

u16Abd1553Chnl (input parameter)
MIL-STD-1553 Channel Number
Valid Values: 0 – 31

DESCRIPTION

This function will stop and reset a specific MIL-STD-1553 channel and return it to the RESET state.

RETURN VALUE

Negative Value = Error
0 = Success

EXAMPLE

```
S16BIT wResult = 0;
U16BIT w1553Chnl = 0;

/* Free 1553 Channel */
if((wResult = abd1553Free(w1553Chnl))
{
    printf(“abd1553Free %d Failed. Error Code: %d”,w1553Chnl,wResult);
}
```

SEE ALSO

abdFree()
abdEthFree()
abd429Free()

abd429Initialize

This function will initialize an ARINC 429 channel for use with Protocol Bridging.

PROTOTYPE

#include “setup429.h”
S16BIT abd429Initialize(U16BIT u16Abd429Chnl);

HARDWARE

BU-67X21WX

STATE

RESET

MODE

Any

PARAMETERS

u16Abd429Chnl	(input parameter) ARINC 429 Channel Number Valid Values: 0 – 31

DESCRIPTION

This function will initialize a specific ARINC 429 channel for protocol bridging and place it in the SETUP state.

RETURN VALUE

Negative Value = Error
0 = Success

EXAMPLE

```
S16BIT wResult = 0;
U16BIT w429Chnl = 0;

/* Initialize 429 Channel */
if((wResult = abd429Initialize(w429Chnl))
{
    printf(“abd429Initialize %d Failed. Error Code: %d”, w429Chnl,
wResult);
}
```

SEE ALSO

abdInitialize()	**abd1553Initialize()**
abdEthInitialize()	

abd429Free

This function will reset an ARINC 429 channel to an uninitialized state.

PROTOTYPE

#include “setup429.h”
S16BIT abd429Free(U16BIT u16Abd429Chnl);

HARDWARE

BU-67X21WX

STATE

RESET, SETUP, RUN

MODE

Any

PARAMETERS

u16Abd429Chnl	(input parameter) ARINC 429 Channel Number Valid Values: 0 – 31

DESCRIPTION

This function will stop and reset a specific ARINC 429 channel and return it to the RESET state.

RETURN VALUE

Negative Value = Error
0 = Success

EXAMPLE

```
S16BIT wResult = 0;
U16BIT w429Chnl = 0;

/* Free 429 Channel */
if((wResult = abd429Free(w429Chnl))
{
    printf(“abd429Free %d Failed. Error Code: %d”,w429Chnl,wResult);
}
```

SEE ALSO

abdFree() **abd1553Free()**
abdEthFree()

abdEthInitialize

This function will initialize an Ethernet channel for use with Protocol Bridging.

PROTOTYPE

#include “setupEth.h”
S16BIT abdEthInitialize(U16BIT u16Abd429Chnl);

HARDWARE

BU-67X21WX

STATE

RESET

MODE

Any

PARAMETERS

u16AbdEthChnl	(input parameter) Ethernet Channel Number Valid Values: 0

DESCRIPTION

This function will initialize a specific Ethernet channel for protocol bridging and place it in the SETUP state.

RETURN VALUE

Negative Value = Error
0 = Success

EXAMPLE

```
S16BIT wResult = 0;
U16BIT wEthChnl = 0;

/* Initialize Ethernet Channel */
if((wResult = abdEthInitialize(wEthChnl))
{
     printf(“abdEthInitialize     %d     Failed.     Error     Code:
     %d,wEthChnl,wResult);
}
```

SEE ALSO

abdInitialize() **abd1553Initialize()**
abd429Initialize()

abdEthFree

This function will reset an Ethernet channel to an uninitialized state.

PROTOTYPE

#include “setupEth.h”
S16BIT abdEthFree(U16BIT u16Abd429Chnl);

HARDWARE

BU-67X21WX

STATE

RESET, SETUP, RUN

MODE

Any

PARAMETERS

u16AbdEthChnl	(input parameter) Ethernet Channel Number Valid Values: 0,1

DESCRIPTION

This function will stop and reset a specific Ethernet channel and return it to the RESET state.

RETURN VALUE

Negative Value = Error
0 = Success

EXAMPLE

```
S16BIT wResult = 0;
U16BIT wEthChnl = 0;

/* Free Ethernet Channel */
if((wResult = abdEthFree(wEthChnl))
{
    printf(“abdEthFree %d Failed. Error Code: %d”,wEthChnl,wResult);
}
```

SEE ALSO

abdFree() **abd1553Free()**
abd429Free()

abd1553Setup

This function will setup a specific 1553 channel in a specific mode of operation.

PROTOTYPE

```
#include “setup1553.h”
S16BIT abd1553Setup(U16BIT u16Abd1553Chnl, U16BIT abd1553Mode);
```

HARDWARE

BU-67X21WX

STATE

SETUP

MODE

Any

PARAMETERS

u16Abd1553Chnl	(input parameter) 1553 Channel Number Valid Values: 0,1
u16Abd1553Mode	(input parameter) 1553 Mode of Operation Valid Values:

ABD_1553_BC	Bus Controller Mode
ABD_1553_MRT	Multi-Remote Terminal Mode
ABD_1553_MT	Bus Monitor Mode

DESCRIPTION

This function will setup an initialized 1553 channel into one of three modes (Bus Controller, Multi-Remote Terminal, or Bus Monitor)

RETURN VALUE

Negative Value = Error
0 = Success

abd1553Setup (continued)

EXAMPLE

```
S16BIT wResult   = 0;
U16BIT w1553Chnl = 0;

/* Initialize 1553 channel in Bus Monitor Mode */
if((wResult = abd1553Setup(w1553Chnl, ABD_1553_MT))
{
    printf("abd1553Setup %d Failed. Error Code: %d",w1553Chnl,wResult);
}
```

SEE ALSO

abd429Setup()   **abdEthSetup()**

abd429Setup

This function will setup a specific ARINC 429 channel in a specific mode of operation.

PROTOTYPE

```
#include "setup429.h"
S16BIT abd429Setup(U16BIT u16Abd429Chnl, U16BIT abd429Mode,
                        S16BIT s16Abd429Speed, S16BIT s16Abd429Parity);
```

HARDWARE

BU-67X21WX

STATE

SETUP

MODE

Any

PARAMETERS

u16Abd429Chnl
(input parameter)
1553 Channel Number
Valid Values: 0 – 31

u16Abd429Mode
(input parameter)
429 Data Direction
Valid Values:

ABD_429_RX	ARINC 429 Receiver
ABD_429_TX	ARINC 429 Transmitter

s16Abd429Speed
(input parameter)
429 Data Speed
Valid Values:

DD429_LOW_SPEED	Low-Speed 429
DD429_HIGH_SPEED	High-Speed 429

u16Abd429Parity
(input parameter)
429 Parity Selection
Valid Values:

DD429_NO_PARITY	No Parity
DD429_ODD_PARITY	Odd Parity
DD429_EVEN_PARITY	Even Parity

DESCRIPTION

This function will setup the ARINC 429 direction, speed and parity for an initialized 429 channel.

RETURN VALUE

Negative Value = Error
0 = Success

abd429Setup (continued)

EXAMPLE

```
S16BIT wResult   = 0;
U16BIT w429Chnl = 0;

/* Initialize 429 channel as a receiver */
if((wResult = abd429Setup(w429Chnl, ABD_429_RX,
    DD429_HIGH_SPEED, DD429_ODD_PARITY))
{
    printf(“abd429Setup %d Failed. Error Code: %d”,w429Chnl,wResult);
}
```

SEE ALSO

abd1553Setup () **abdEthSetup()**

abdEthSetup

This function will setup a specific Ethernet channel in a specific mode of operation.

PROTOTYPE

```
#include “setupEth.h”
S16BIT abdEthSetup(U16BIT u16AbdEthChnl, U16BIT abdEthMode, U16BIT u16PortNum);
```

HARDWARE

BU-67X21WX

STATE

SETUP

MODE

Any

PARAMETERS

u16Abd429Chnl
(input parameter)
1553 Channel Number
Valid Values: 0,1

u16AbdEthMode
(input parameter)
IP Protocol Selection
Valid Values:

ABD_ETH_TCP	Use TCP/IP Protocol
ABD_ETH_UDP	Use UDP/IP Protocol

u16PortNum
(input parameter)
TCP/UDP Port (Socket) Number
Valid Values:
1-65535

DESCRIPTION

This function will setup the Ethernet Upper IP Protocol and the Socket Port Number. Note: please make sure to select a port number that does not conflict with existing network services.

RETURN VALUE

Negative Value = Error
0 = Success

abdEthSetup (continued)

EXAMPLE

```
S16BIT wResult   = 0;
U16BIT wEthChnl = 0;
U16BIT wEthPort = 5000;


/* Initialize Ethernet channel for UDP/IP on Port 5000 */
if((wResult = abdEthSetup(w429Chnl, ABD_ETH_UDP, wEthPort));
{
    printf(“abdEthSetup %d Failed. Error Code: %d”,wEthChnl,wResult);
}
```

SEE ALSO

abd1553Setup () **abd429Setup()**

abd1553Start

This function will start operation of a setup MIL-STD-1553 channel.

PROTOTYPE

#include “setup1553.h”
S16BIT abd155Start(U16BIT u16Abd1553Chnl);

HARDWARE

BU-67X21WX

STATE

SETUP

MODE

Any

PARAMETERS

u16Abd1553Chnl	(input parameter) MIL-STD-1553 Channel Number Valid Values: 0 – 31

DESCRIPTION

This function will start a specific MIL-STD-1553 channel and make it active (transmit and receive) on the 1553 bus.

RETURN VALUE

Negative Value = Error
0 = Success

EXAMPLE

```
S16BIT wResult = 0;
U16BIT w1553Chnl = 0;

/* Start 1553 Channel */
if((wResult = abd1553Start(w1553Chnl))
{
    printf(“abd1553Start %d Failed. Error Code: %d”,w1553Chnl,wResult);
}
```

SEE ALSO

abd1553Stop() **abd1553Setup ()**
abd1553Initialize()

abd1553Stop

This function will stop operation of a running MIL-STD-1553 channel.

PROTOTYPE

#include “setup1553.h”
S16BIT abd155Stop(U16BIT u16Abd1553Chnl);

HARDWARE

BU-67X21WX

STATE

RUN

MODE

Any

PARAMETERS

u16Abd1553Chnl	(input parameter) MIL-STD-1553 Channel Number Valid Values: 0 – 31

DESCRIPTION

This function will stop a specific MIL-STD-1553 channel from being active (transmit and receive) on the 1553 bus.

RETURN VALUE

Negative Value = Error
0 = Success

EXAMPLE

```
S16BIT wResult = 0;
U16BIT w1553Chnl = 0;

/* Stop 1553 Channel */
if((wResult = abd1553Stop(w1553Chnl))
{
    printf(“abd1553Stop %d Failed. Error Code: %d”,w1553Chnl,wResult);
}
```

SEE ALSO

abd1553Start() **abd1553Setup ()**
abd1553Initialize()

abd429Start

This function will start a setup ARINC 429 channel.

PROTOTYPE

```
#include “setup429.h”
S16BIT abd429Start(U16BIT u16Abd429Chnl);
```

HARDWARE

BU-67X21WX

STATE

SETUP

MODE

Any

PARAMETERS

u16Abd429Chnl	(input parameter) ARINC 429 Channel Number Valid Values: 0 – 31

DESCRIPTION

This function will start a specific ARINC 429 channel and allow it to transmit or receive.

RETURN VALUE

Negative Value = Error
0 = Success

EXAMPLE

```
S16BIT wResult = 0;
U16BIT w429Chnl = 0;

/* Start 429 Channel */
if((wResult = abd429Start(w429Chnl))
{
    printf(“abd429Start %d Failed. Error Code: %d”,w429Chnl,wResult);
}
```

SEE ALSO

abd429Initialize() **abd429Stop()**
abd429Setup()

abd429Stop

This function will stop a running ARINC 429 channel.

PROTOTYPE

#include “setup429.h”
S16BIT abd429Stopt(U16BIT u16Abd429Chnl);

HARDWARE

BU-67X21WX

STATE

RUN

MODE

Any

PARAMETERS

u16Abd429Chnl	(input parameter) ARINC 429 Channel Number Valid Values: 0 – 31

DESCRIPTION

This function will stop a specific ARINC 429 channel from running (transmit and received).

RETURN VALUE

Negative Value = Error
0 = Success

EXAMPLE

```
S16BIT wResult = 0;
U16BIT w429Chnl = 0;

/* Stop 429 Channel */
if((wResult = abd429Stop(w429Chnl))
{
    printf(“abd429Stop %d Failed. Error Code: %d”,w429Chnl,wResult);
}
```

SEE ALSO

abd429Initialize() **abd429Start()**
abd429Setup()

abdEthStart

This function will start a setup Ethernet channel.

PROTOTYPE

#include “setupEth.h”
S16BIT abdEthStart(U16BIT u16AbdEthChnl);

HARDWARE

BU-67X21WX

STATE

SETUP

MODE

Any

PARAMETERS

u16AbdEthChnl	(input parameter) Ethernet Channel Number Valid Values: 0,1

DESCRIPTION

This function will start a specific Ethernet channel and allow it to transmit or receive.

RETURN VALUE

Negative Value = Error
0 = Success

EXAMPLE

```
S16BIT wResult = 0;
U16BIT wEthChnl = 0;

/* Start Ethernet Channel */
if((wResult = abdEthStart(wEthChnl))
{
    printf(“abdEthStart %d Failed. Error Code: %d”,wEthChnl,wResult);
}
```

SEE ALSO

abdEthInitialize() **abdEthStop()**
abdEthSetup()

abdEthStop

This function will stop a running Ethernet channel.

PROTOTYPE

#include “setupEth.h”
S16BIT abdEthStopt(U16BIT u16AbdEthChnl);

HARDWARE

BU-67X21WX

STATE

RUN

MODE

Any

PARAMETERS

u16AbdEthChnl	(input parameter) Ethernet Channel Number Valid Values: 0,1

DESCRIPTION

This function will stop a specific Ethernet channel from running (transmit and received).

RETURN VALUE

Negative Value = Error
0 = Success

EXAMPLE

```
S16BIT wResult = 0;
U16BIT wEthChnl = 0;

/* Stop Ethernet Channel */
if((wResult = abdEthStop(wEthChnl))
{
    printf(“abdEthStop %d Failed. Error Code: %d”,wEthChnl,wResult);
}
```

SEE ALSO

abdEthInitialize() **abdEthStart()**
abdEthSetup()

abd1553EnableRT

This function will enable a specific RT Address to transmit and/or receive.

PROTOTYPE

```
#include “setup1553.h”
S16BIT abd1553EnableRT(U16BIT u16Abd1553Chnl, S8BIT s8RtAddr);
```

HARDWARE

BU-67X21WX

STATE

SETUP

MODE

1553 RT

PARAMETERS

u16Abd1553Chnl	(input parameter) MIL-STD-1553 Channel Number Valid Values: 0 – 31
s8RtAddr	(input parameter) RT Address to Enable Valid Values: 0-31

DESCRIPTION

This function will enable a specific MIL-STD-1553 RT Address to be active on the specified 1553 channel. Note: this function does not enable RT Subaddresses, only the RT Address.

RETURN VALUE

Negative Value = Error
0 = Success

EXAMPLE

```
S16BIT wResult = 0;
U16BIT w1553Chnl = 0;

/* Enable RT #01 */
if((wResult = abd1553EnableRT(w1553Chnl, 0x01))
{
    printf(“abd1553EnableRT      %d      Failed.      Error      Code:
            %d,w1553Chnl,wResult);
}
```

abd1553EnableRT (continued)

SEE ALSO

abd1553EnableSA()
abdEthInitialize()
abd1553Start()

abd1553EnableSA

This function will enable an RT Subaddress within a setup and enabled RT Address.

PROTOTYPE

```
#include “setup1553.h”
S16BIT abd1553EnableSA(U16BIT u16Abd1553Chnl, S8BIT s8RtAddr, U16BIT u16SA,
                                    U16BIT* pTxDataBuf, U16BIT u16TxDataBufLen);
```

HARDWARE

BU-67X21WX

STATE

SETUP

MODE

1553 RT

PARAMETERS

u16Abd1553Chnl	(input parameter) MIL-STD-1553 Channel Number Valid Values: 0 – 31
s8RtAddr	(input parameter) RT Address Reference Valid Values:0-31
u16SA	(input parameter) RT Subaddress to Enable Valid Values:0-31
pTxDataBuf	(input parameter) Pointer to initial Data for RT Transmit Commands
u16TxDataBufLen	(input parameter) Size (in Words) of pTxDataBuf Valid Values:0-32

DESCRIPTION

This function will enable a specific MIL-STD-1553 RT Subaddress (within the specified RT Address) to be active on the assigned 1553 channel. Note that this the RT Address must first be enabled by calling **abd1553EnableRT()**.

RETURN VALUE

Negative Value = Error
0 = Success

abd1553EnableSA (continued)

EXAMPLE

```
S16BIT wResult = 0;
U16BIT w1553Chnl = 0;
U16BIT u16Data[64] =
{
    0x1111, 0x2222, 0x3333, 0x4444, 0x1111, 0x2222, 0x3333, 0x4444,
    0x1111, 0x2222, 0x3333, 0x4444, 0x1111, 0x2222, 0x3333, 0x4444,
    0x1111, 0x2222, 0x3333, 0x4444, 0x1111, 0x2222, 0x3333, 0x4444,
    0x1111, 0x2222, 0x3333, 0x4444, 0x1111, 0x2222, 0x3333, 0x4444
};

/* Enable RT #01 SA #02*/
if((wResult = abd1553EnableSA(w1553Chnl, 0x01, 0x02, u16Data, 32))
{
    printf(“abd1553EnableSA      %d      Failed.      Error      Code:
            %d,w1553Chnl,wResult);
}
```

SEE ALSO

abd1553EnableRT()

abd1553Start()

abd1553Stop()

abd1553BcCreateMsg

This function will create a 1553 Message to be sent by a Bus Controller.

PROTOTYPE

```
#include "setup1553.h"
S16BIT abd1553BcCreateMsg(U16BIT u16Abd1553Chnl, S16BIT s16MsgID,
                          ABD_1553_BC_MSG_CREATE_INFO sMsgCreateInfo,
                          U16BIT *pDataBuf, U16BIT u16DataBufSize);
```

HARDWARE

BU-67X21WX

STATE

SETUP

MODE

1553 BC

PARAMETERS

u16Abd1553Chnl	(input parameter) MIL-STD-1553 Channel Number Valid Values: 0 – 31
s16MsgID	(input parameter) User-Assigned Message ID 1-65535
sMsgCreateInfo	(input parameter) 1553 Message Information
pDataBuf	(input parameter) Pointer to initial Data for BC-RT Commands
u16TxDataBufLen	(input parameter) Size (in Words) of pTxDataBuf Valid Values:0-32

DESCRIPTION

This function will create a 1553 Bus Controller Message to be transmitted by the Bus Controller. The user has the option of addling this message to the synchronous 1553 BC Bus List or asynchronously sending it whenever new bridged data is received.

RETURN VALUE

Negative Value = Error
0 = Success

abd1553BcCreateMsg (continued)

EXAMPLE

```
S16BIT wResult = 0;
U16BIT w1553Chnl = 0;
ABD_1553_BC_MSG_CREATE_INFO sMsgCreateInfo;
U16BIT u16Data[32] =
{
    0x1111, 0x2222, 0x3333, 0x4444, 0x1111, 0x2222, 0x3333, 0x4444,
    0x1111, 0x2222, 0x3333, 0x4444, 0x1111, 0x2222, 0x3333, 0x4444,
};

/* BC->RT1/SA1 msg with 16 data words */
sMsgCreateInfo.msgType = BC_TO_RT; /* BC->RT Message */
sMsgCreateInfo.bSched = TRUE;      /* Will be added to the Bus List. */
sMsgCreateInfo.u16RTRx = 1;        /* RT 01 */
sMsgCreateInfo.u16SARx = 1;        /* SA 01 */
sMsgCreateInfo.u16WC   = 16;       /* 16 Data Words */
sMsgCreateInfo.u16MsgGapTime = 0;  /* No additional Gap Time */
sMsgCreateInfo.u32MsgOptions  =  ACE_BCCTRL_CHL_A;  /*  Send  on  Channel  A
*/

/* Create New 1553 BC Message */
if((wResult = abd1553BcCreateMsg(w1553Chnl,sMsgCreateInfo,u16Data,16)))
{
    printf(“abd1553BcCreateMsg     %d     Failed.     Error     Code:
            %d,w1553Chnl,wResult);
}
```

SEE ALSO

abd1553BcAddMsgToList() **abd1553Start()**

ABD_1553_BC_MSG_CREATE_INFO

abd1553BcAddMsgToList

This function will add an existing 1553 Message to the synchronous BC Bus List.

PROTOTYPE

#include “setup1553.h”
S16BIT abd1553BcAddMsgToList (U16BIT u16Abd1553Chnl, S16BIT s16MsgID);

HARDWARE

BU-67X21WX

STATE

SETUP

MODE

1553 BC

PARAMETERS

u16Abd1553Chnl	(input parameter) MIL-STD-1553 Channel Number Valid Values: 0 – 31
s16MsgID	(input parameter) User-Assigned Message ID 1-65535

DESCRIPTION

This function will create a 1553 Bus Controller Message to be transmitted by the Bus Controller. The user has the option of addling this message to the synchronous 1553 BC Bus List or asynchronously sending it whenever new bridged data is received.

RETURN VALUE

Negative Value = Error
0 = Success

EXAMPLE

```
S16BIT wResult = 0;
U16BIT w1553Chnl = 0;

/* Add Message ID 01 to the BC Bust List */
if((wResult = abd1553BcAddMsgToList(w1553Chnl,0x01))
{
    printf(“abd1553BcAddMsgToList            %d            Failed.Error
Code:%d,w1553Chnl,wResult);
}
```

abd1553BcAddMsgToList (continued)

SEE ALSO

abd1553BcCreateMsg()

abd1553Start()

abd1553Stop()

abd429AddTxMsg

This function will add a 429 Message to the Schedule or FIFO Pending Queue

PROTOTYPE

```
#include “setup429.h”
S16BIT abd429AddTxMsg(U16BIT u16Abd429Chnl, U16BIT u16MsgId, U32BIT u32InitMsgData,
                              BOOLEAN bSched, S16BIT s16Frequency, S16BIT s16Offset);
```

HARDWARE

BU-67X21WX

STATE

SETUP, RUN

MODE

ARINC 429

PARAMETERS

u16Abd429Chnl

(input parameter)
ARINC 429 Channel Number
Valid Values: 0 – 31

s16MsgID

(input parameter)
User-Assigned Message ID
1-65535

u32InitMsgData

(input parameter)
Initial ARINC 429 Message (32-bit value format)

bSched

(input parameter)
TRUE = Schedule the Message at a defined periodic Rate
FALSE = Send the message when new data is available (FIFO)

s16Frequency

(input parameter)
Only applies to Periodic Messages. (bSched = TRUE)
Defines the frequency of the transmission in milliseconds
Valid Values: 1-32767

s16Offset

(input parameter)
Only applies to Periodic Messages. (bSched = TRUE)
The offset in milliseconds relative to the other scheduled words.
Must be less than the “s16Frequency’ input parameter.
Valid Values: 1-32767

abd429AddTxMsg (continued)

DESCRIPTION

This function schedules a transmission of an ARINC 429 Message. If ‘bSched” is set to TRUE, the Message will transmit at the specified periodic rate. If ‘bSched” is FALSE, the message will only be sent when new bridge data linked to this message is received (FIFO). Note: the 429 Channel Must be started (using **abd429Start()**) for transmissions to begin.

RETURN VALUE

Negative Value = Error
0 = Success

EXAMPLE

```
S16BIT wResult = 0;
U16BIT w429Chnl = 0;
U32BIT u32Data = 0x12345678; /* ARINC SDI/Label/Data Message */

/* Setup Tx Msg.(ID=0x01) Scheduled, freq 1000ms, Offset 0ms */
if((wResult = abd429AddTxMsg(w429Chnl, 0x01, u32Data, TRUE, 1000, 0))
{
    printf(“abd429AddTxMsg      %d      Failed.      Error      Code:
%d”,w429Chnl,wResult);
}
```

SEE ALSO

abd429Setup() **abd429Start()**
abd429Stop()

abdEthAddTxMsg

This function will add an Ethernet Message to the FIFO Pending Queue.

PROTOTYPE

```
#include “setupEth.h”
S16BIT abdEthAddTxMsg(U16BIT u16AbdEthChnl, U16BIT u16MsgId, char * remoteIP,
                                    S16BIT s16DestPort, U32BIT u32MsgLength);
```

HARDWARE

BU-67X21WX

STATE

SETUP, RUN

MODE

ETHERNET

PARAMETERS

u16AbdEtheChnl

(input parameter)
Ethernet Channel Number
Valid Values: 0, 1

s16MsgID

(input parameter)
User-Assigned Message ID
1-65535

remoteIP

(input parameter)
Remote IP Address to send to (String Format)
Example: “192.168.1.1”

s16DestPort

(input parameter)
Only applies to UDP/IP configured Channels.
Enter -1 for TCP/IP configured Channels.
Remote Port to send to.
Valid Values: 1-65535

u32MsgLength

(input parameter)
Size of the message (in bytes)
Valid Values: 1-65535

abd4EthAddTxMsg (continued)

DESCRIPTION

This function schedules transmission of an Ethernet Message. The Message will only be sent when new bridge data linked to this message becomes available. Note, the Ethernet Channel must be started (using **abdEthStart()**) for transmissions to begin.

RETURN VALUE

Negative Value = Error
0 = Success

EXAMPLE

```
S16BIT wResult = 0;
U16BIT wEthChnl = 0;

/* Setup UDP Msg. (ID=0x01) to “192.168.1.1”, Port 5000 */
if((wResult = abdEthAddTxMsg(wEthChnl,0x01,”192.168.1.1”,5000,1024))
{
    printf(“abdEthAddTxMsg %d Failed. Error Code:
%d”,w429Chnl,wResult);
}
```

SEE ALSO

abdEthSetup()
abdEthStop()
abdEthStart()

abd1553AddMatchingEntry

This function will create a Matching Entry for received/monitored MIL-STD-1553 traffic.

PROTOTYPE

```
#include "setup1553.h"
S16BIT abd1553AddMatchingEntry(U16BIT u16Abd1553Chnl, U16BIT u16MatchingId,
                               U16BIT u16RxCmdWrd, U16BIT u16RxCmdWrdMask,
                               U16BIT u16TxCmdWrd, U16BIT u16TxCmdWrdMask);
```

HARDWARE

BU-67X21WX

STATE

SETUP

MODE

1553 ANY

PARAMETERS

u16Abd1553Chnl	(input parameter) MIL-STD-1553 Channel Number Valid Values: 0 – 31
u16MatchingId	(input parameter) User Assigned Matching ID. Valid Values: 1 - 65535
u16RxCmdWrd	(input parameter) RT Receive Command Word to Match. Valid Values: 0x0000 - 0xFFFF
u16RxCmdWrdMask	(input parameter) RT Receive Command Word Mask Value. Valid Values: 0x0000 - 0xFFFF
u16TxCmdWrd	(input parameter) RT Transmit Command Word to Match. Valid Values: 0x0000 - 0xFFFF
u16TxCmdWrdMask	(input parameter) RT Transmit Command Word Mask Value. Valid Values: 0x0000 - 0xFFFF

abd1553AddMatchingEntry (continued)

DESCRIPTION

This function will create a Matching Entry for received/monitored 1553 traffic. If any monitored 1553 message matches the Logical AND of the either the Receive Command Word/Mask or the Transmit Command Word/Mask, then the data will be made available to bridge to a destination channel using the Protocol “Mapping Rule” functions.

Note: 1553 data cannot be bridged to any other channel before defining a Matching Entry.

RETURN VALUE

Negative Value = Error
0 = Success

EXAMPLE

```
S16BIT wResult = 0;
U16BIT w1553Chnl = 0;

/* Create a Matching Entry for BC->RT01/SA01/WC32 (CmdWord=0x0820) */
if((wResult                                                            =
abd1553AddMatchingEntry(w1553Chnl,0x01,0x0820,0xFFFF,0,0xFFFF))
{
    printf(“abd1553AddMatchingEntry Failed.  Error Code: wResult);
}
```

SEE ALSO

abd1553EnableRT()
abd429AddMappingRule()
abd1553Start()

abd1553AddMappingRule

This function will create a bridge from a 1553 Matching Entry to the desired destination location.

PROTOTYPE

```
#include “setup1553.h”
S16BIT abd1553AddMappingRule(U16BIT u16Abd1553Chnl, U16BIT u16MatchingId,
                             U8BIT u8SrcStartingWrdIndex, U8BIT u8SrcStartingBit,
                             U16BIT u16SrcLength,
                             PABD_MAPPING_DEST_INFO pMappingDestInfo);
```

HARDWARE

BU-67X21WX

STATE

SETUP

MODE

1553 ANY

PARAMETERS

u16Abd1553Chnl	(input parameter) MIL-STD-1553 Channel Number Valid Values: 0 – 31
u16MatchingId	(input parameter) Matching ID of the desired Matching Entry Valid Values: 1 - 65535
u8SrcStartingWrdIndex	(input parameter) Starting 1553 Word Location to bridge (from Matching Entry) Valid Values: 0 - 31
u8SrcStartingBit	(input parameter) Starting Bit location (from Starting Word) to bridge Valid Values: 0 - 15
u16SrcLength	(input parameter) Length of data (in bits) to bridge (Starting from Word/Bit above) Valid Values: 1 - 704
pMappingDestInfo	(input parameter) Desired Destination Message to bridge data into.

abd1553AddMappingRule (continued)

DESCRIPTION

This function will bridge (copy) data from a defined 1553 Matching Entry to a desired destination message. If the referenced 1553 matching entry message has been updated, the protocol bridging layer will autonomously copy it into the destination message.

Note: this function does not effect the scheduling of the target destination message. It is up to the user to either schedule the destination message in a bus list or to assign it to be sent asynchronously.

The MIL-STD-1553 Message format is as follows. Users shall reference this format when assigning which words/bits to bridge 1553 Source or Destination data.

Table 3. ABD_1553_UNIFIED MSG_STRUCTURE

Struct Member	Word Location	Description
u16IntraPktTimeStamp	0-3	64-bit Timestamp of last reception.
u16BlkStats	4	1553 Block Status Word
u16GapTime	5	1553 Message Gap Time
u16Length	6	Length of Message (in bytes)
u16RxCmdWrd	7	1553 RT Receive Command Word
u16RxStatusWrd	8	1553 RT Receive Status Word
u16TxCmdWrd	9	1553 RT Transmit Command Word
u16TxStatusWrd	10	1553 RT Transmit Status Word
u16WrdCnt	11	1553 Word Count (Data Payload)
u16TRforModeCode	12-44	0-31 Data Words (Payload)

RETURN VALUE

Negative Value = Error
0 = Success

abd1553AddMappingRule (continued)

EXAMPLE

```
S16BIT wResult = 0;
U16BIT w1553Chnl = 0;
ABD_MAPPING_DEST_INFO sDestInfo;

/* Map 1553 Data to Ethernet */
sDestInfo.u8MappingDestType = 2;                  /*   Destination   =
Ethernet */
sDestInfo.u.sEthMappingDest.u16ChnlNum = 0;       /* Ethernet Channel 0
*/
sDestInfo.u.sEthMappingDest.u16MsgId = 1;         /* Ethernet Message ID
1 */
sDestInfo.u.sEthMappingDest.u16StartingByte = 0; /* Ethernet Start Byte
0 */
sDestInfo.u.sEthMappingDest.u16StartingBit = 0;   /* Ethernet Start Bit
0 */

/* Bridge 32 (16 bits*32) words from 1553 Msg ID #1 to an Ethernet
Message */
if((wResult=abd1553AddMappingRule(w1553Chnl,1,12,0,16*32,&sDestInfo))
{
    printf(“abd1553AddMappingRule Failed.  Error Code: wResult);
}
```

SEE ALSO

abd429AddMappingRule()
abd1553Start()
ABD_MAPPING_DEST_INFO

abd429AddMatchingEntry

This function will create a Matching Entry for received ARINC 429 traffic.

PROTOTYPE

```
#include "setup429.h"
S16BIT abd429AddMatchingEntry(U16BIT u16Abd429Chnl, U16BIT u16MatchingId,
                                         U16BIT u16LabelSDI, U16BIT u16LabelSDIMatchingMask)
```

HARDWARE

BU-67X21WX

STATE

SETUP

MODE

ARINC 429 RX

PARAMETERS

u16Abd429Chnl	(input parameter) ARINC 429 Channel Number Valid Values: 0 – 31
u16MatchingId	(input parameter) User Assigned Matching ID. Valid Values: 1 - 65535
u16LabelSDI	(input parameter) ARINC 429 Label/SDI Word to Match (A429 Word Bits 0-9) Valid Values: 0x0000 - 0xFFFF
u16LabelSDIMatchingMask	(input parameter) ARINC 429 Label/SDI Word Mask Value. Valid Values: 0x0000 - 0x03FF

DESCRIPTION

This function will create a Matching Entry for received ARINC 429 traffic. If any received ARINC 429 message matches the Logical AND of the Label/SDI Word and Mask, then the data will be made available to bridge to a destination channel using the Protocol "Mapping Rule" functions.

Note: ARINC 429 data cannot be bridged to any other channel before defining a Matching Entry.

RETURN VALUE

Negative Value = Error
0 = Success

abd429AddMatchingEntry (continued)

EXAMPLE

```
S16BIT wResult = 0;
U16BIT w429Chnl = 0;
U16BIT wLabelSDI = 0x0102; /* SDI = 0x01B, Label = 0x02 */

/* Create a Matching Entry (ID=0x01) for SDI=0x01B, Label=0x02 */
if((wResult = abd429AddMatchingEntry(w429Chnl,0x01,wLabelSDI,0x03FF))
{
    printf(“abd429AddMatchingEntry Failed.  Error Code: wResult);
}
```

SEE ALSO

abd429Setup()
abd429AddMappingRule()
abd429Start()

abd429AddMappingRule

This function will create a bridge from an ARINC 429 Matching Entry to the desired destination.

PROTOTYPE

```
#include “setup429.h”
S16BIT abd429AddMappingRule(U16BIT u16Abd429Chnl, U16BIT u16MatchingId,
                            U8BIT u8SrcStartingBit, U8BIT u8StartingDWord
                            U16BIT u16SrcLength,
                            PABD_MAPPING_DEST_INFO pMappingDestInfo);
```

HARDWARE

BU-67X21WX

STATE

SETUP

MODE

ARINC 429 RX

PARAMETERS

u16Abd429Chnl	(input parameter) ARINC 429 Channel Number Valid Values: 0 – 31
u16MatchingId	(input parameter) Matching ID of the desired Matching Entry Valid Values: 1 - 65535
u8SrcStartingDWord	(input parameter) Starting DWord (32-bits) within ARINC Message to bridge Valid Values: 0 - 2
u8SrcStartingBit	(input parameter) Starting Bit location (from above DWord) to bridge Valid Values: 0 - 31
u16SrcLength	(input parameter) Length of data (in bits) to bridge (Starting from Bit above) Valid Values: 1 - 96
pMappingDestInfo	(input parameter) Desired Destination Message to bridge data into.

abd429AddMappingRule (continued)

DESCRIPTION

This function will bridge (copy) data from a defined ARINC 429 Matching Entry to a desired destination message. If the referenced ARINC 429 message has been updated, the protocol bridging layer will autonomously copy it into the destination message.

Note: this function does not effect the scheduling of the target destination message. It is up to the user to either schedule the destination message in a bus list or to assign it to be sent asynchronously.

The ARINC 429 Message format is as follows. Users shall reference this format when assigning which DWords/bits to bridge ARIINC 429 Source or Destination data.

Table 4. ARINC_429_MSG_STRUCTURE

Variable Name	DWord Location	Bit Location	Description
Label	0	0-7	Message Label
SDI	0	8-9	Source/Destination Identifier (SDI)
Data	0	10-28	Data Payload
SSM	0	29-30	Sign/Status Matrix
P	0	31	Parity Bit
Time Stamp High	1	0-31	Upper 32-bits of 64-bit Time Stamp
Time Stamp Low	2	0-31	Lower 32-bits of 64-bit Time Stamp

RETURN VALUE

Negative Value = Error
0 = Success

abd429AddMappingRule (continued)

EXAMPLE

```
S16BIT wResult = 0;
U16BIT w429Chnl = 0;
ABD_MAPPING_DEST_INFO sDestInfo;

/* Map 1553 Data to Ethernet */
sDestInfo.u8MappingDestType = 2;                   /*    Destination    =
Ethernet */
sDestInfo.u.sEthMappingDest.u16ChnlNum = 0;        /* Ethernet Channel 0
*/
sDestInfo.u.sEthMappingDest.u16MsgId = 1;         /* Ethernet Message ID
#1 */
sDestInfo.u.sEthMappingDest.u16StartingByte = 0; /* Ethernet Start Byte
0 */
sDestInfo.u.sEthMappingDest.u16StartingBit = 0;   /* Ethernet Start Bit
0 */

/* Bridge 32 bits (all) from 429 Msg ID #1 to an Ethernet Message */
if((wResult=abd429AddMappingRule(w429Chnl,1,0,0,32,&sDestInfo))
{
    printf(“abd429AddMappingRule Failed.  Error Code: wResult);
}
```

SEE ALSO

abd429AddMappingRule()	**ABD_MAPPING_DEST_INFO**
abd429Start()	

abdEthAddMatchingEntry

This function will create a Matching Entry for received Ethernet traffic.

PROTOTYPE

```
#include “setupEth.h”
S16BIT abdEthAddMatchingEntry(U16BIT u16AbdEthChnl, U16BIT u16MatchingId,
                              char * remoteIpaddr, char * addrMask);
```

HARDWARE

BU-67X21WX

STATE

SETUP

MODE

ETHERNET

PARAMETERS

u16AbdEthChnl	(input parameter) Ethernet Channel Number Valid Values: 0, 1
u16MatchingId	(input parameter) User Assigned Matching ID. Valid Values: 1 - 65535
RemoteIpaddr	(input parameter) IPv4 IP Address to Match (String Format)
addrMask	(input parameter) IPv4 IP Address Word Mask Value.

DESCRIPTION

This function will create a Matching Entry for received Ethernet traffic. If any received Ethernet message matches the Logical AND of the Remote IP Address Value and Mask, then the data will be made available to bridge to a destination channel using the Protocol “Mapping Rule” functions.

Note: Ethernet data cannot be bridged to any other channel before defining a Matching Entry.

RETURN VALUE

Negative Value = Error
0 = Success

abdEthAddMatchingEntry (continued)

EXAMPLE

```
S16BIT wResult = 0;
U16BIT wEthChnl = 0;

/* Create a Matching Entry for IP 192.168.1.2 */
if((wResult =

abdEthAddMatchingEntry(wEthChnl,0x01,”192.168.1.2”,”255.255.255.255”))
{
    printf(“abdEthAddMatchingEntry Failed.  Error Code: wResult);
}
```

SEE ALSO

abdEthSetup() **abdEthStart()**
abdEthAddMatchingEntry()

abdEthAddMappingRule

This function will create a bridge from an Ethernet Matching Entry to the desired destination.

PROTOTYPE

```
#include “setupEth.h”
S16BIT abdEthAddMappingRule(U16BIT u16AbdEthChnl, U16BIT u16MatchingId,
                             U32BIT u32SrcStartingByte, U32BIT u32SrcStartingBit,
                             U32BIT u32SrcLength,
                             PABD_MAPPING_DEST_INFO pMappingDestInfo);
```

HARDWARE

BU-67X21WX

STATE

SETUP

MODE

ETHERNET

PARAMETERS

u16AbdEthChnl
(input parameter)
Ethernet Channel Number
Valid Values: 0, 1

u16MatchingId
(input parameter)
Matching ID of the desired Matching Entry
Valid Values: 1 - 65535

u32SrcStartingByte
(input parameter)
Starting Byte location to bridge
Valid Values: 0 - 65535

u32SrcStartingBit
(input parameter)
Starting Bit location (from above Byte) to bridge
Valid Values: 0 - 7

u32SrcLength
(input parameter)
Length of data (in bits) to bridge (Starting from Byte/Bit above)
Valid Values: 1 - 524288

pMappingDestInfo
(input parameter)
Desired Destination Message to bridge data into.

abdEthAddMappingRule (continued)

DESCRIPTION

This function will bridge (copy) data from a defined Ethernet Matching Entry to a desired destination message. If the referenced Ethernet packet has been updated, the protocol bridging layer will autonomously copy it into the destination message.

Note: this function does not effect the scheduling of the target destination message. It is up to the user to either schedule the destination message in a bus list or to assign it to be sent asynchronously.

The Ethernet Message format is as follows. Users shall reference this format when assigning which bits to bridge Ethernet Source or Destination data.

Table 5. ETHERNET_MSG_STRUCTURE

Variable Name	Byte Location	Description
Length	0-3	Length of TCP Message (in bytes)
Data	4-65535	User-Defined Data Payload

RETURN VALUE

Negative Value = Error
0 = Success

EXAMPLE

```
S16BIT wResult = 0;
U16BIT wEthChnl = 0;
ABD_MAPPING_DEST_INFO sDestInfo;

/* Map Ethernet Data to 1553 Remote Terminal(RT 01, SA01 Word/Bit 0/0*/
sDestInfo.u8MappingDestType = 0;                /* Destination = 1553 */
sDestInfo.u.s1553MappingDest.u16ChnlNum = 0;   /* 1553 Channel 0 */
sDestInfo.u.s1553MappingDest.s16MsgId = -1;    /* N/A for 1553 RT */
sDestInfo.u.s1553MappingDest.s8RtAddr = 1;     /* 1553 RT Address 01 */
sDestInfo.u.s1553MappingDest.s8SA     = 1;     /* 1553 SA 01 */
sDestInfo.u.s1553MappingDest.u16StartingDataWord = 0;/* 1553 DataWord 0
*/
sDestInfo.u.s1553MappingDest.u16StartingBit = 0;  /* 1553 Bit 0 */

/* Bridge 16 Words from Ethernet to 1553 RT 01 SA 01, Words 0-15 */
if((wResult=abdEthAddMappingRule(wEthChnl,1,4,0,16*16,&sDestInfo))
{
    printf(“abdEthAddMappingRule Failed.  Error Code: wResult);
}
```

SEE ALSO

abdEthAddMatchingEntry()	**ABD_MAPPING_DEST_INFO**
abdEthStart()	

7.4 Structures

ABD_1553_BC_MSG_CREATE_INFO

Structure used in creating MIL-STD-1553 Bus Controller Messages.

PROTOTYPE

```
#include “setup1553.h”
typedef struct _ABD_1553_BC_MSG_CREATE_INFO
{
        ABD_1553_BC_MSG_TYPE msgType;
        BOOLEAN bSched;
        U16BIT u16RTRx;
        U16BIT u16SARx;
        U16BIT u16RTTx;
        U16BIT u16SATx;
        U16BIT u16WC;
        U16BIT u16MsgGapTime;
        U32BIT u32MsgOptions;
        U16BIT u16TRforModeCode;
        U16BIT u16ModeCmd;
        U16BIT u16MajorFrmTime;
} ABD_1553_BC_MSG_CREATE_INFO, *PABD_1553_BC_MSG_CREATE_INFO;
```

Table 6. ABD_1553_BC_MSG_CREATE_INFO Structure

Struct Member	Description
msgType	Type of Message. Valid Choice are: BC_TO_RT, RT_TO_BC, RT_TO_RT, MODE, BROADCAST, BROADCAST_RT_TO_RT, BROADCAST_MODE, END_OF_MINOR, END_OF_MAJOR.
bSched	TRUE = Message will be added to the sched. Bus List via abd1553BcAddMsgToList FALSE = Message will be send asynchronously when new data is available.
u16RTRx	RT Address for Receive RT.
u16SARx	RT Subaddress for Receive RT.
u16RTTx	RT Address for Transmit RT.
u16SATx	RT Subaddress for Transmit RT.
u16WC	Word Count or Mode Code
u16MsgGapTime	Gap Time (in us) to next message (Only applied to sched. Messages).
u32MsgOptions	1553 Message Options.
u16TRforModeCode	Transmit (1) / Receive (0) for Mode Code commands
u16ModeCmd	1553 Mode Command.
u16MajorFrmTime	Major Frame Time (in 100us increments). Only applies to END_OF_MAJOR messages.

ABD_1553_BC_MSG_CREATE_INFO (continued)

SEE ALSO

abd1553BcCreateMsg()	**abd1553Start()**
abd1553BcAddMsgToList()	

ABD_MAPPING_DEST_INFO (*PABD_MAPPING_DEST_INFO)

Structure (union) to define protocol bridging destination location.

PROTOTYPE

```
#include “abdDev.h”
typedef struct _ABD_MAPPING_DEST_INFO
{
         U8BIT u8MappingDestType;

         union ABD_MAPPING_DEST_INFO_UNION
         {
             ABD_1553_MAPPING_DEST_INFO s1553MappingDest;
             ABD_429_MAPPING_DEST_INFO  s429MappingDest;
             ABD_ETH_MAPPING_DEST_INFO  sEthMappingDest;
         } u;
}ABD_MAPPING_DEST_INFO, *PABD_MAPPING_DEST_INFO;
```

Table 7. ABD_MAPPING_DEST_INFO Structure

Struct Member	Description
U8MappingDestType	Destination Protocol: 1 = MIL-STD-1553 2 = ARINC 429 3 = ETHERNET
u.s1553MappingDest	MIL-STD-1553 Bridging Destination Information
u.s429MappingDest	ARINC 429 Bridging Destination Information
u.sEthMappingDest	Ethernet Bridging Destination Information

SEE ALSO

abd1553AddMappingRule() **abd429AddMappingRule()**
abdEthAddMappingRule()

ABD_1553_MAPPING_DEST_INFO

Structure to define MIL-STD-1553 protocol bridging destination location.

PROTOTYPE

```
#include “abdDev.h”
typedef struct _ABD_1553_MAPPING_DEST_INFO
{
        U16BIT u16ChnlNum;
        S16BIT s16MsgId;
        S8BIT  s8RtAddr;
        S8BIT  s8SA;
        U16BIT u16StartingDataWord;
        U16BIT u16StartingBit;
}ABD_1553_MAPPING_DEST_INFO, *PABD_1553_MAPPING_DEST_INFO;
```

Table 8. ABD_1553_MAPPING_DEST_INFO Structure

Struct Member	Description
u16ChnlNum	Channel Number of the destination MIL-STD-1553 channel.
s16MsgId	Message ID of destination BC Message. Note, Only applicable if 1553 Target is configured as a Bus Controller. Set to ‘-1’ for 1553 channels configured as Remote Terminals. Valid Values: 1-65535
s8RtAddr	RT Address of destination. Note, Only applicable if 1553 Target is configured as Remote Terminal(s). Set to ‘-1’ for 1553 channels configured as Bus Controller.
s8SA	RT Subaddress of destination. Note, Only applicable if 1553 Target is configured as Remote Terminal(s). Set to ‘-1’ for 1553 channels configured as Bus Controller.
u16StartingDataWord	Starting 1553 Word within destination to bridge (copy) data to. Valid Values : 0-31
u16StartingBit	Starting Bit (from above 1553 Word) within destination to bridge (copy) data to. Valid Values : 0-15

SEE ALSO

abd1553AddMatchingEntry() **abd1553AddMappingRule()**
ABD_MAPPING_DEST_INFO

ABD_429_MAPPING_DEST_INFO

Structure to define ARINC 429 protocol bridging destination location.

PROTOTYPE

```
#include “abdDev.h”
typedef struct _ABD_429_MAPPING_DEST_INFO
{
        U16BIT u16ChnlNum;
        S16BIT s16MsgId;
        U16BIT u16StartingBit;
}ABD_429_MAPPING_DEST_INFO, *PABD_429_MAPPING_DEST_INFO;
```

Table 9. ABD_429_MAPPING_DEST_INFO Structure

Struct Member	Description
u16ChnlNum	Channel Number of the destination ARINC 429 channel.
s16MsgId	Message ID of destination ARINC 429 Message. Note, Only applicable if destination 429 channel is configured as a transmitter.
u16StartingBit	Starting Bit within ‘Data’ portion (bits 10-30) of destination ARINC 429 32-bit Message. Valid Values: 10-30

SEE ALSO

abd429AddMatchingEntry() **abd429AddMappingRule()**
ABD_MAPPING_DEST_INFO

ABD_ETH_MAPPING_DEST_INFO

Structure to define Ethernet protocol bridging destination location.

PROTOTYPE

```
#include “abdDev.h”
typedef struct _ABD_ETH_MAPPING_DEST_INFO
{
        U16BIT u16ChnlNum;
        U16BIT u16MsgId;
        U16BIT u16StartingByte;
        U16BIT u16StartingBit;
}ABD_ETH_MAPPING_DEST_INFO, *PABD_ETH_MAPPING_DEST_INFO;
```

Table 10. ABD_ETH_MAPPING_DEST_INFO Structure

Struct Member	Description
u16ChnlNum	Channel Number of the destination Ethernet channel.
s16MsgId	Message ID of destination Ethernet packet
u16StartingByte	Starting Byte within destination Ethernet Packet. Valid Values : 0-65535
u16StartingBit	Starting Bit (from above starting byte) within destination Ethernet Packet. Valid Values : 0-7

SEE ALSO

abdEthAddMatchingEntry() **abdEthAddMappingRule()**
ABD_MAPPING_DEST_INFO

7.5 Available Samples

This Protocol Conversion Layer API is supplied with many examples of the proper use of the SDK and the capabilities of the hardware. The examples provided demonstrate numerous bridging techniques between the 3 available protocols (MIL-STD-1553, ARINC 429, and Ethernet).

In all cases, the examples have been provided as source codes with an appropriate “Makefiles” that may be used to build the executable.

The samples reside in the following directory on the ABD Target:

/home/ddc/1553_429/samples/bridging

7.5.1 Received ARINC 429 to MIL-STD-1553 Bus Controller (arinc2bc)

This sample application shows how the user can bridge a received ARINC Message to a MIL-STD-1553 Receive (BC→RT) Message.

7.5.2 Received ARINC 429 to Transmit Ethernet (arinc2eth)

This sample application shows how the user can bridge a received ARINC Message to an Ethernet transmit Packet.

7.5.3 MIL-STD-1553 Bus Controller to Transmit Ethernet (bc2eth)

This sample application shows how the user can bridge data from a MIL-STD-1553 Transmit (RT→BC) to an Ethernet transmit Packet.

7.5.4 Received Ethernet to ARINC 429 Transmit (eth2arinc)

This sample application shows how the user can bridge a received Ethernet Packet to an ARINC Transmit Message.

7.5.5 Received Ethernet to MIL-STD-1553 Bus Controller (eth2bc)

This sample application shows how the user can bridge a received Ethernet Packet to a MIL-STD-1553 Receive (BC→RT) Message.

7.5.6 Received Ethernet to MIL-STD-1553 Remote Terminal (eth2rt)

This sample application shows how the user can bridge a received Ethernet Packet to a MIL-STD-1553 Remote Terminal.

7.5.7 MIL-STD-1553 Remote Terminal to Transmit Ethernet (mrt2eth)

This sample application shows how the user can bridge data received on a MIL-STD-1553 Remote Terminal to an Ethernet transmit Packet.

7.5.8 MIL-STD-1553 Bus Monitor to ARINC 429 Transmit (mt2arinc)

This sample application shows how the user can bridge monitored MIL-STD-1553 Data to an ARINC Transmit Message.

7.5.9 MIL-STD-1553 Bus Monitor to Transmit Ethernet (mt2eth)

This sample application shows how the user can bridge monitored MIL-STD-1553 Data to an Ethernet Transmit Packet.

8 USING REMOTE ACCESS MODE

When the Avionics Interface Computer is put into Remote Access Mode, a virtual backplane is created between the applications running on a host computer and the MIL-STD-1553 / ARINC 429 interfaces on the AIC. Remote Access mode allows for the AIC to be plugged into the network while the user uses the 1553 and/or 429 channels on the AIC from a remote location. Remote Access mode is supported under Windows, Linux, and VxWorks.

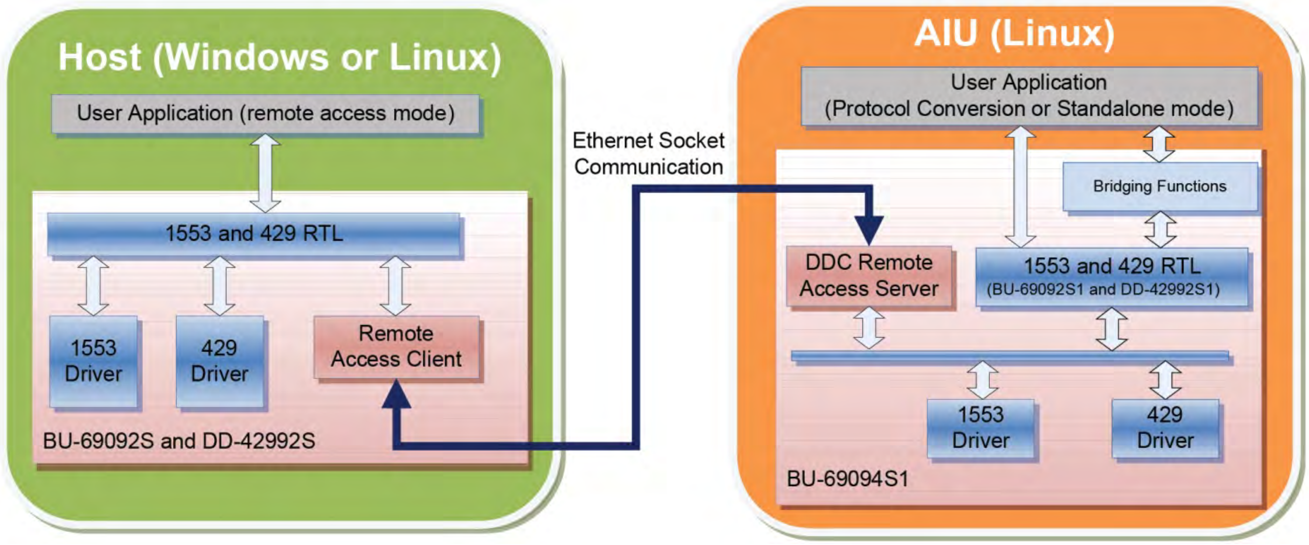


Figure 27. AIC Remote Access interaction

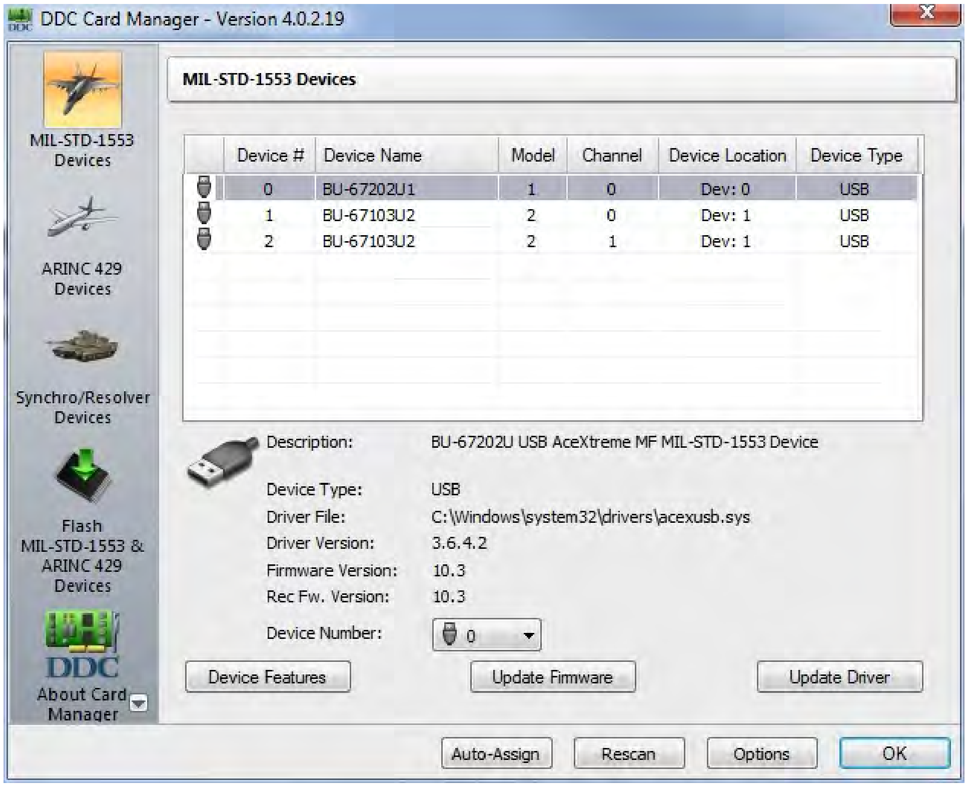
When using a Windows Operating system with the Avionics Interface Computer in Remote Access mode, all of DDC's software tools can be used. These tools include the AceXtreme® SDK, ARINC 429 SDK, BusTrACEr®, *data*SIMS®, Commercial Avionics Utilities, and DDC's LabVIEW® Support Package.

8.1 Remote Access Mode With Windows

To use the AIC with a Windows host Operating system, the DDC Card Manager must be used to find the AIC and assign Logical Device Numbers (LDNs) to the 1553 and 429 channels on the AIC. The DDC Card Manager is bundled with the DDC SDK software in the BU-69092S0 AceXtreme SDK, or the DD-42992S0 ARINC 429 SDK. These SDKs are needed to use the AIC under Windows and should be installed per the SDK installation instructions. Once the SDKs have been installed and the LDNs have been assigned, the AIC is ready for use with DDC's Windows software support packages.

8.1.1 Assigning Logical Devices Numbers

Assigning a LDN to the 1553 and 429 channels on the AIC is done by first opening the DDC Card Manager, located in the Windows Control Panel or as a shortcut in the Start Menu. Once the DDC Card Manager is opened, there will be several buttons on the left side of the card manager. These buttons allow you to select which interface type you will be assigning an LDN for your device. The button options include **MIL-STD-1553, ARINC 429** and **Synchro/ Resolver Devices** (the AIC does not contain any Synchro / Resolver channels).



MIL-STD-1553 Devices

Device #	Device Name	Model	Channel	Device Location	Device Type
0	BU-67202U1	1	0	Dev: 0	USB
1	BU-67103U2	2	0	Dev: 1	USB
2	BU-67103U2	2	1	Dev: 1	USB

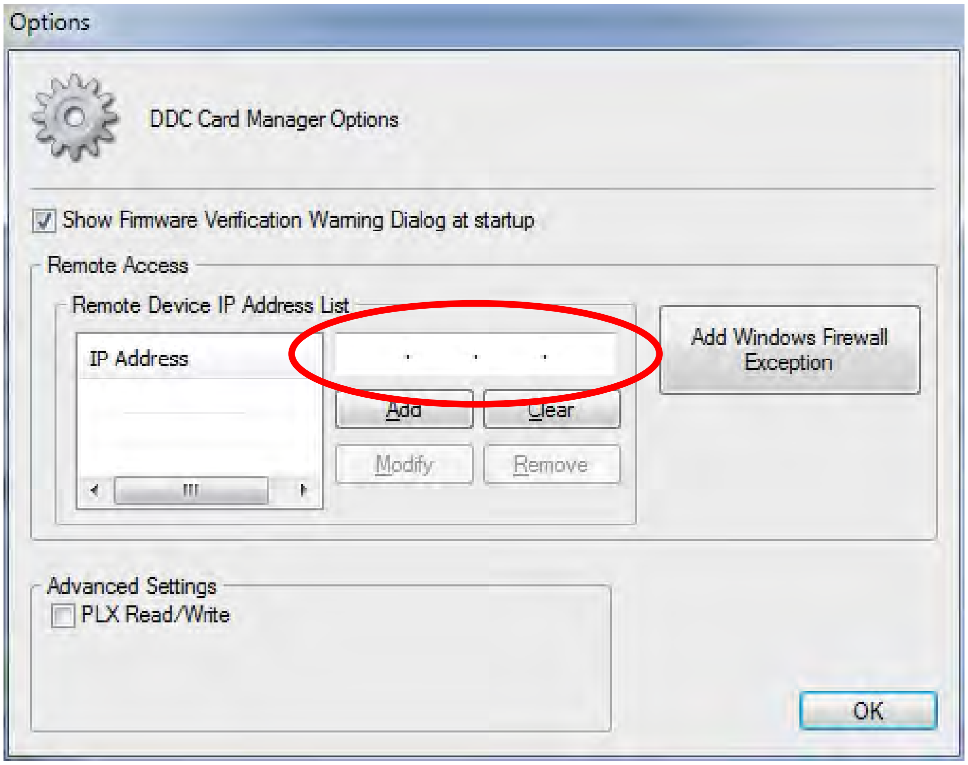
Description: BU-67202U USB AceXtreme MF MIL-STD-1553 Device

Device Type: USB
Driver File: C:\Windows\system32\drivers\acexusb.sys
Driver Version: 3.6.4.2
Firmware Version: 10.3
Rec Fw. Version: 10.3

Device Number: 0

Figure 28. DDC Card Manager (Windows)

The DDC Card Manager will need to know the IP address of your AIC. To enter in the IP Address click **Options** on the lower right side of the DDC Card Manager.



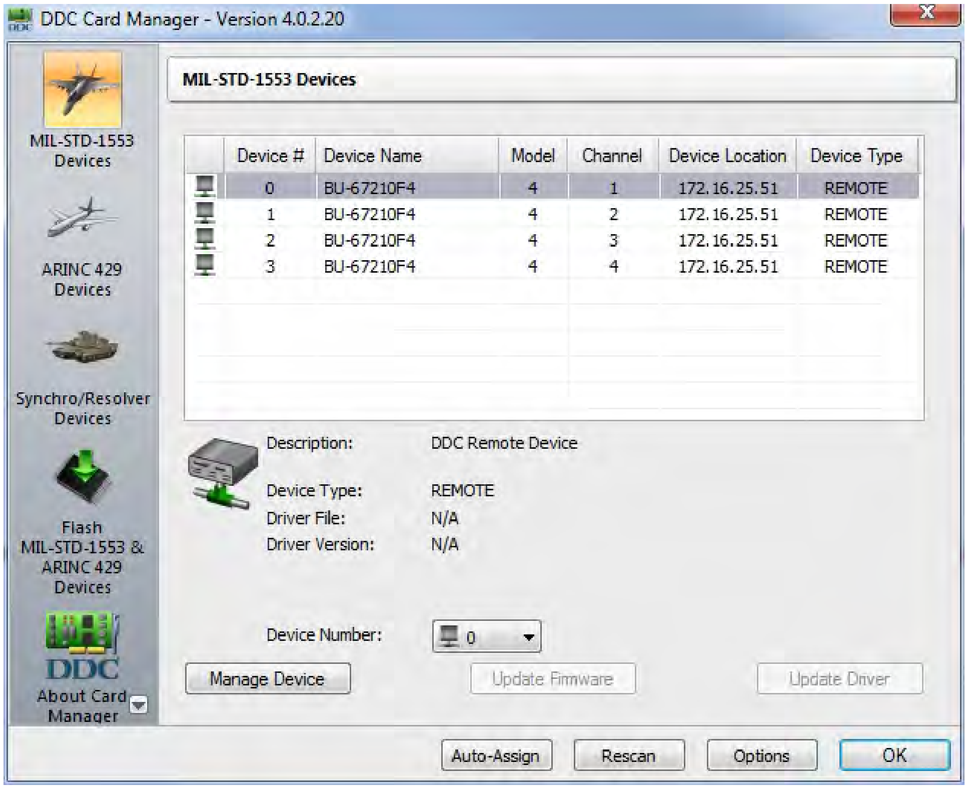
☑ Show Firmware Verification Warning Dialog at startup

☐ PLX Read/Write

Figure 29. DDC Card Manager Options Dialog

The options dialog window will be displayed, allowing the user to enter the IP Address of the AIC in the circled area above. Clicking **Add** will insert the IP Address in the list of addresses the DDC Card Manager will search for upon loading, or when rescan is selected. **Modify** will allow you to change one of the IP addresses in the list. **Remove** will delete the IP Address from the list of IP Addresses. **Clear** will remove the IP Address from the circled box.

Once the IP Address of the AIC has been added, click **OK** to return to the main dialog window of the DDC Card Manager. Clicking **Rescan** will cause the DDC Card Manager to search for any AICs based on the provided IP address and add the AIC's channels to the appropriate sections. At this point, a LDN may be assigned to the 1553 and or 429 channels, and the LDNs may be used with the Windows Host operation system and DDC software.



MIL-STD-1553 Devices

Device #	Device Name	Model	Channel	Device Location	Device Type
0	BU-67210F4	4	1	172.16.25.51	REMOTE
1	BU-67210F4	4	2	172.16.25.51	REMOTE
2	BU-67210F4	4	3	172.16.25.51	REMOTE
3	BU-67210F4	4	4	172.16.25.51	REMOTE

Description: DDC Remote Device
Device Type: REMOTE
Driver File: N/A
Driver Version: N/A

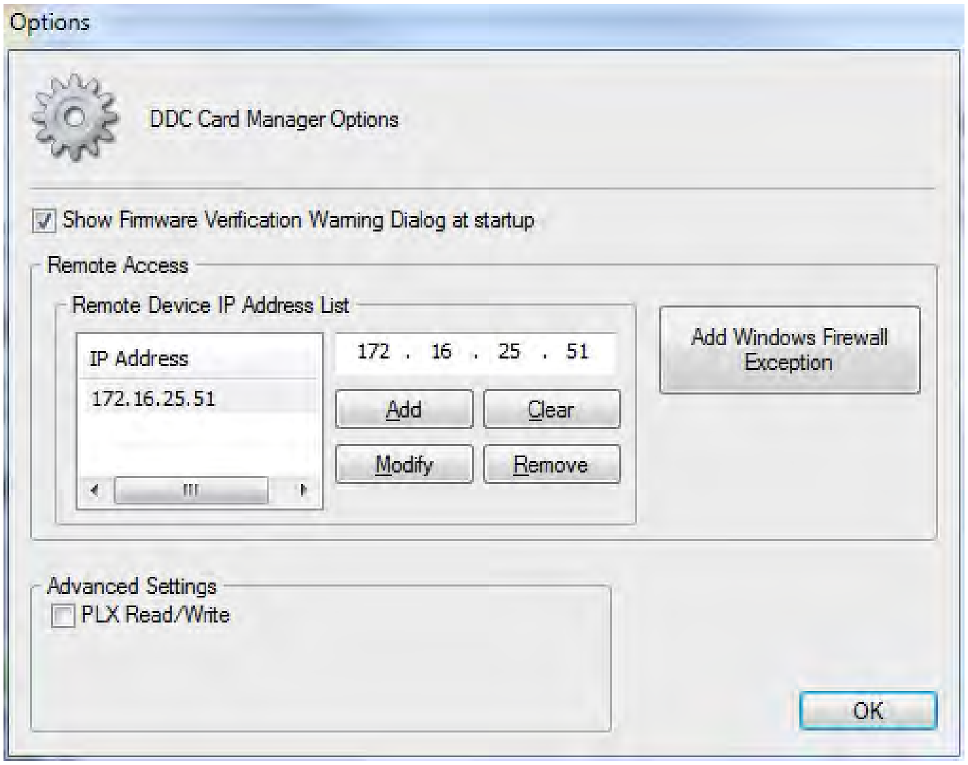
Device Number: 0

Figure 30. DDC Card Manager with Remote MIL-STD-1553 Devices.

Note: The 429 LDNs are assigned by clicking **ARINC 429 Devices**.

8.1.2 Removing AIC from Card Manager

When removing the AIC from the Windows host, the LDNs and IP address must be removed. It is recommended to first set all LDNs to **NONE** for any channels on the AIC. Then click **Options** to open the DDC Card Manager Options dialog. The AIC’s IP address will be in the IP Address list. Click on it to highlight the IP address and then click the remove button to remove the IP Address from the list.



Options

DDC Card Manager Options

☑ Show Firmware Verification Warning Dialog at startup

Remote Access

Remote Device IP Address List

IP Address
172.16.25.51

172 . 16 . 25 . 51

Advanced Settings

□ PLX Read/Write

Figure 31. DDC Card Manager – Remove IP

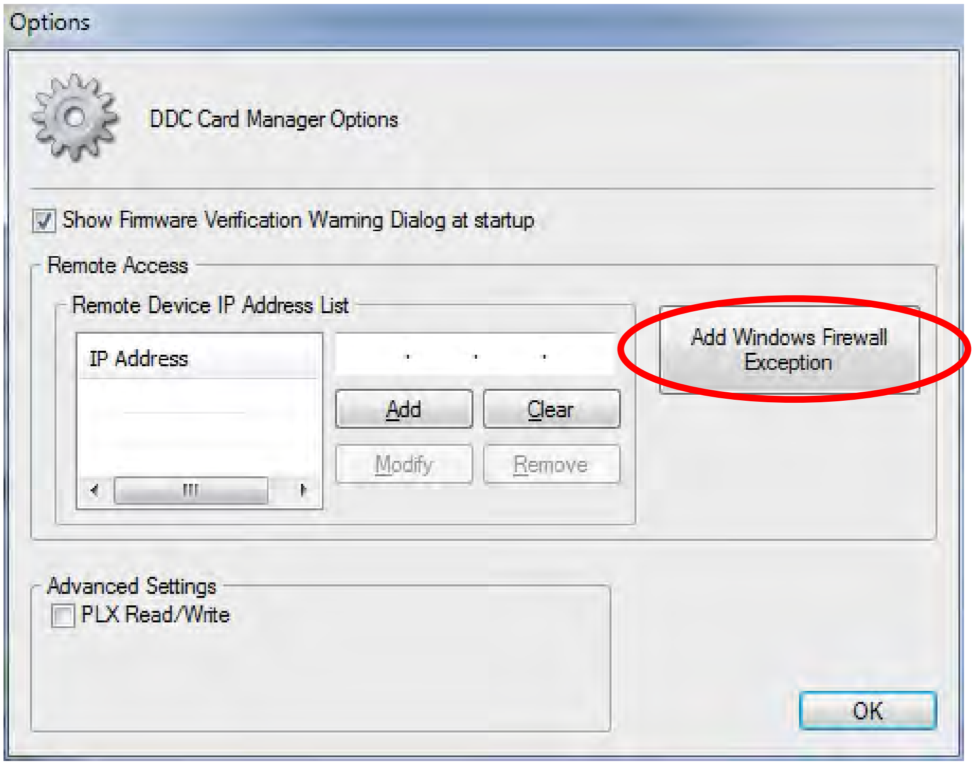
Once the IP Address has been removed, the DDC Card Manager will no longer see the 1553/429 channels on the AIC.

8.1.3 Forwarding Port through Windows Firewall for Remote Access

Windows 7 and Windows 8 may require Ports to be opened within the Windows Firewall for Remote Access Mode. These ports may be opened through the DDC Card Manager (version 4.0.2.19 or later).

The DDC Card Manager (version 4.0.2.19 or later) has the option to open the ports needed by the AIC when using Remote Access Mode. In order to open these ports the user must first open the DDC Card Manager and click **Options** (see Figure 28).

Once the options dialog has been opened, click **Add Windows Firewall Exception** to add the required ports to your firewall.



Options

DDC Card Manager Options

☑ Show Firmware Verification Warning Dialog at startup

Remote Access

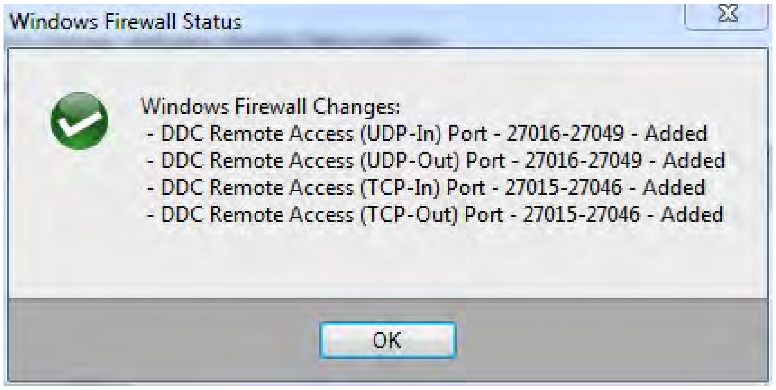
Remote Device IP Address List

Advanced Settings

□ PLX Read/Write

Figure 32. DDC Card Manager Options Dialog

Clicking **Add Windows Firewall Exception** will add the necessary ports starting at port 27016. The number of ports opened will depend on the channel count of your device.



Windows Firewall Status

Windows Firewall Changes:
- DDC Remote Access (UDP-In) Port - 27016-27049 - Added
- DDC Remote Access (UDP-Out) Port - 27016-27049 - Added
- DDC Remote Access (TCP-In) Port - 27015-27046 - Added
- DDC Remote Access (TCP-Out) Port - 27015-27046 - Added

Figure 33. DDC Card Manager Firewall Changes.

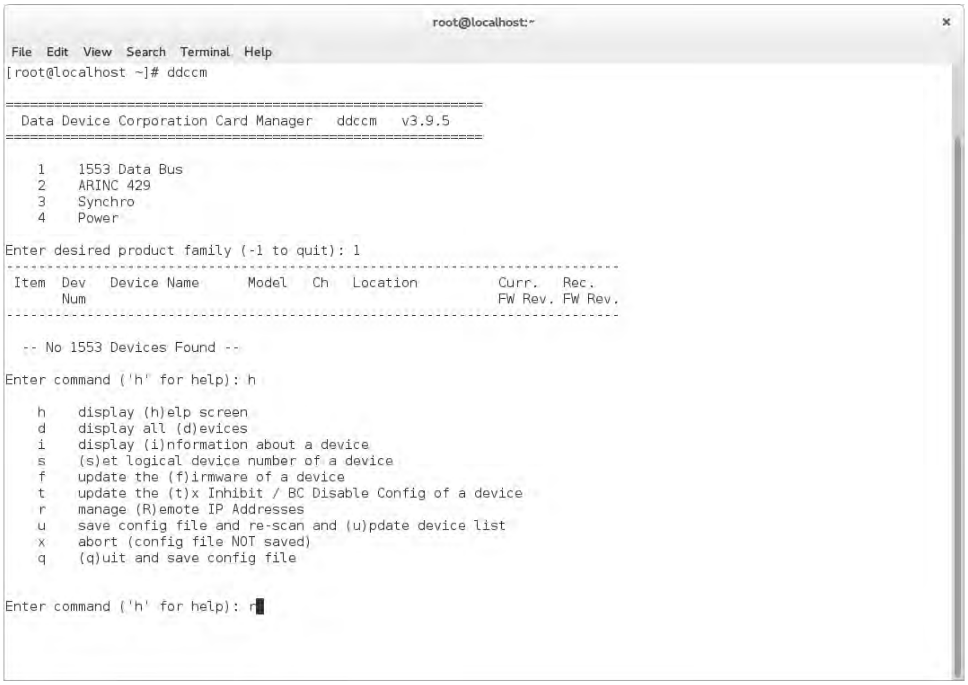
8.2 Remote Access Mode With Linux

To use the AIC with a Linux host Operating system, the DDC Card Manager must be used to find the AIC and assign Logical Device Numbers (LDNs) to the 1553 and 429 channels on the AIC. The DDC Card Manager is bundled with the DDC SDK software in the BU-69092S1 AceXtreme SDK, or the DD-42992S1 ARINC 429 SDK. These SDKs are needed to use the AIC under Linux and should be installed per the SDK installation instructions. Once the SDKs have been installed and the LDNs have been assigned, the AIC is ready for use with DDC's Linux software support packages.

8.2.1 Assigning Logical Devices Numbers

Assigning a LDN to the 1553 and 429 channels on the AIC is done by first opening the DDC Card Manager. This is done from the prompt, by typing **ddccm**. To run **ddccm**, the user must have root privileges; otherwise the Card Manager will report an error.

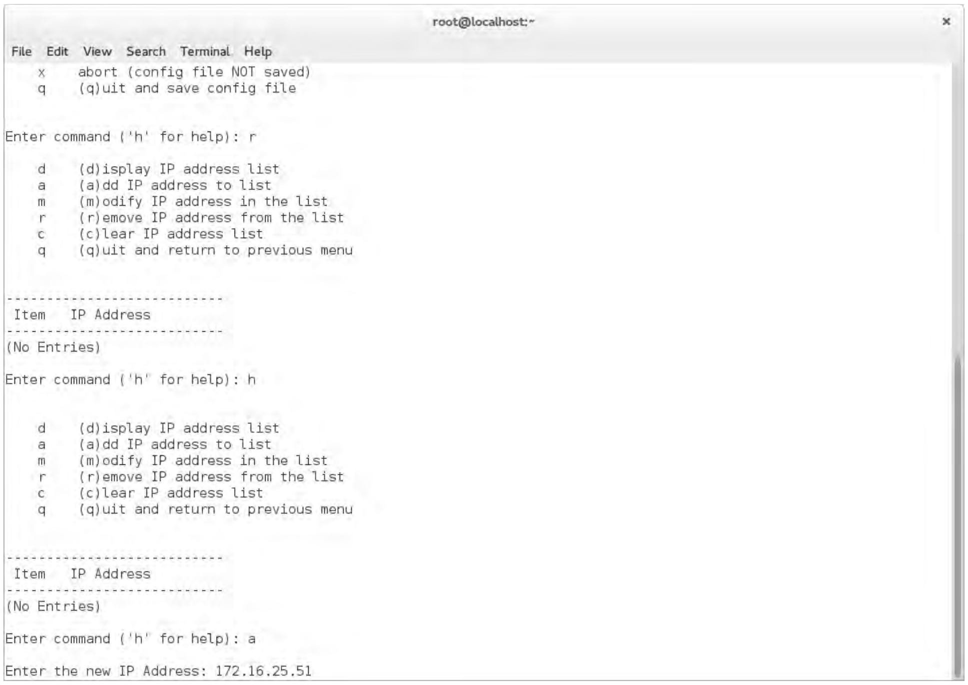
After starting **ddccm**, enter '**1'** for **1553 Data Bus channels** or '**2'** for **ARINC 429**. Figure 34 shows the MIL-STD-1553 channels that are available when selecting 1 from the prompt. The lack of channels is due to the IP Address of the AIC not being added to the IP Address list. At the prompt enter in '**h**' to have **ddccm** display the help context menu which displays a list of commands.



```
[root@localhost ~]# ddccm

============================================================
  Data Device Corporation Card Manager   ddccm   v3.9.5
============================================================

    1    1553 Data Bus
    2    ARINC 429
    3    Synchro
    4    Power

Enter desired product family (-1 to quit): 1
-----------------------------------------------------------------------------
 Item  Dev    Device Name      Model   Ch   Location          Curr.   Rec.
       Num                                                    FW Rev. FW Rev.
-----------------------------------------------------------------------------

  -- No 1553 Devices Found --

Enter command ('h' for help): h

    h    display (h)elp screen
    d    display all (d)evices
    i    display (i)nformation about a device
    s    (s)et logical device number of a device
    f    update the (f)irmware of a device
    t    update the (t)x Inhibit / BC Disable Config of a device
    r    manage (R)emote IP Addresses
    u    save config file and re-scan and (u)pdate device list
    x    abort (config file NOT saved)
    q    (q)uit and save config file


Enter command ('h' for help): r
```

Figure 34. Running ddccm (Linux)

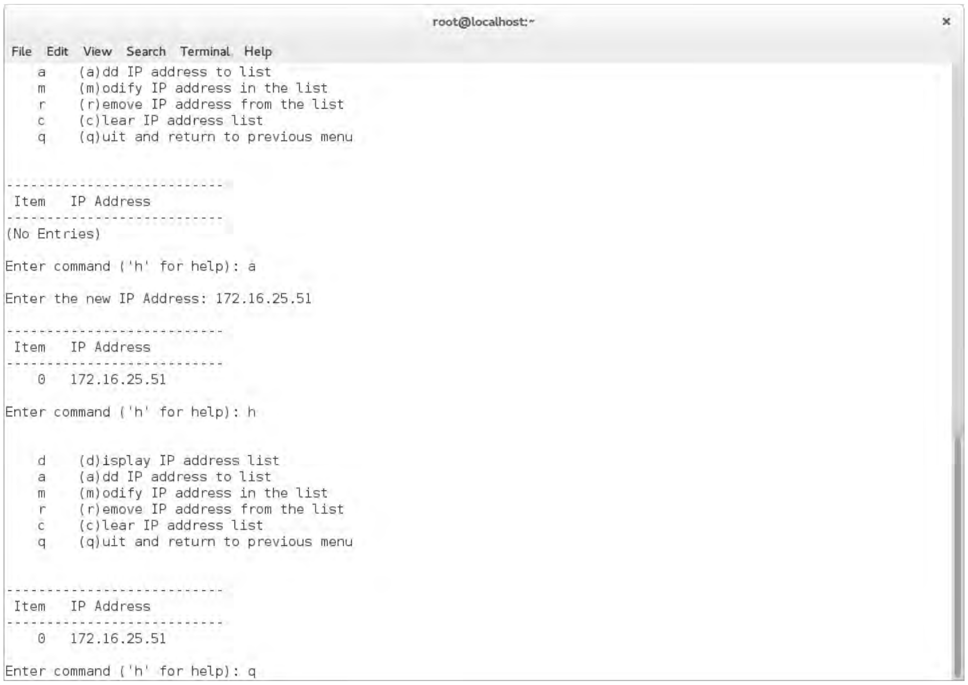
From the prompt, enter '**a**' to **(a)dd an IP address to the list** of searchable IP addresses for the card manager. The card manager will now prompt the user for the IP address to add to the list of IP addresses. Enter in the IP address of the AIC and press enter.



```
    x    abort (config file NOT saved)
    q    (q)uit and save config file


Enter command ('h' for help): r

    d    (d)isplay IP address list
    a    (a)dd IP address to list
    m    (m)odify IP address in the list
    r    (r)emove IP address from the list
    c    (c)lear IP address list
    q    (q)uit and return to previous menu


---------------------------
 Item   IP Address
---------------------------
(No Entries)

Enter command ('h' for help): h


    d    (d)isplay IP address list
    a    (a)dd IP address to list
    m    (m)odify IP address in the list
    r    (r)emove IP address from the list
    c    (c)lear IP address list
    q    (q)uit and return to previous menu


---------------------------
 Item   IP Address
---------------------------
(No Entries)

Enter command ('h' for help): a

Enter the new IP Address: 172.16.25.51
```

Figure 35. Selecting Add/Entering IP Address of AIC

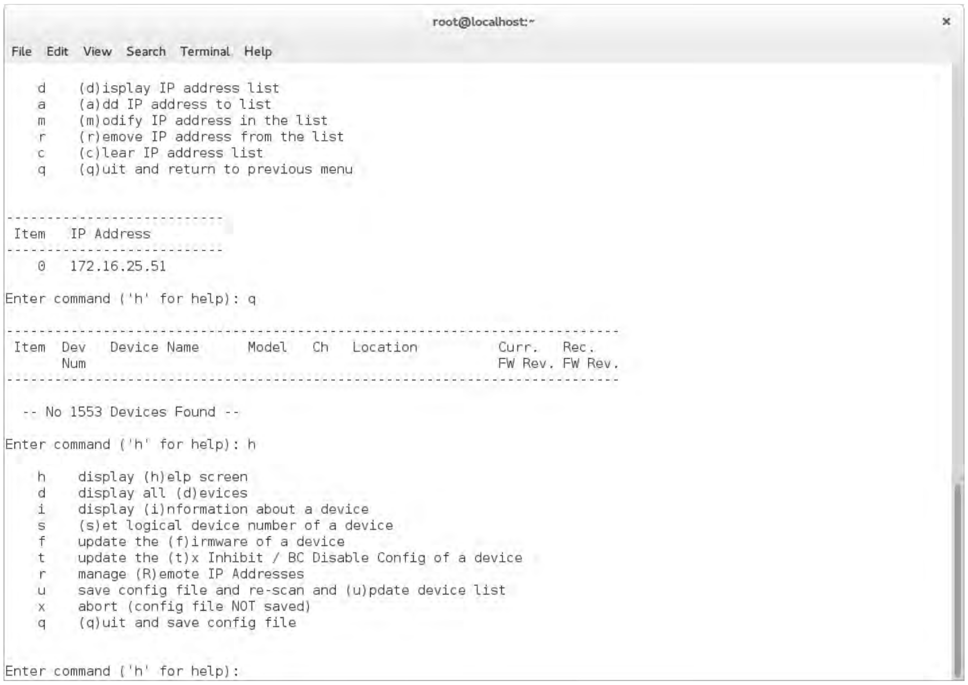
After entering in the IP address, enter in ‘**h**’ at the prompt again to display a new list of selectable commands to perform within **ddccm**. Enter in ‘**q**’ to exit the current level and return back to the Device listing portion of the **ddccm**.



```
    a    (a)dd IP address to list
    m    (m)odify IP address in the list
    r    (r)emove IP address from the list
    c    (c)lear IP address list
    q    (q)uit and return to previous menu


---------------------------
 Item   IP Address
---------------------------
(No Entries)

Enter command ('h' for help): a

Enter the new IP Address: 172.16.25.51

---------------------------
 Item   IP Address
---------------------------
    0   172.16.25.51

Enter command ('h' for help): h


    d    (d)isplay IP address list
    a    (a)dd IP address to list
    m    (m)odify IP address in the list
    r    (r)emove IP address from the list
    c    (c)lear IP address list
    q    (q)uit and return to previous menu


---------------------------
 Item   IP Address
---------------------------
    0   172.16.25.51

Enter command ('h' for help): q
```

Figure 36. Entering IP Address

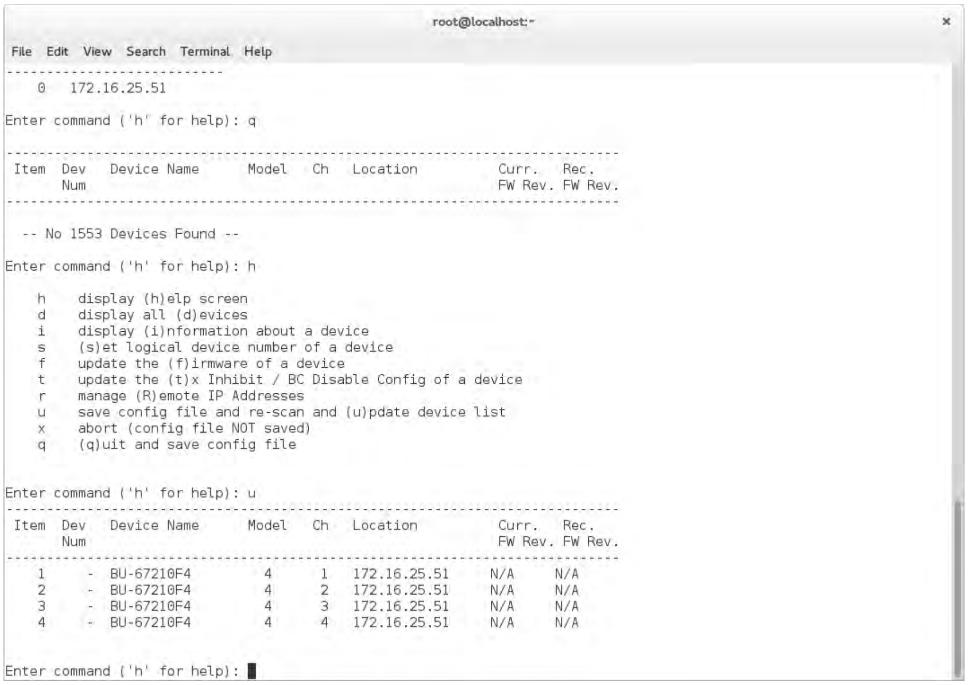
From the prompt, select ‘**u**’ to **save the configuration file and to perform a re-scan and (u)pdate device list** for MIL-STD-1553 and ARINC 429 devices. Any found devices on the AIC will be displayed as shown by Figure 37. To see the ARINC 429 devices, ddccm must be exited and run again.



```
    d    (d)isplay IP address list
    a    (a)dd IP address to list
    m    (m)odify IP address in the list
    r    (r)emove IP address from the list
    c    (c)lear IP address list
    q    (q)uit and return to previous menu


---------------------------
 Item   IP Address
---------------------------
    0   172.16.25.51

Enter command ('h' for help): q

----------------------------------------------------------------------------
 Item  Dev   Device Name      Model   Ch   Location          Curr.   Rec.
       Num                                                   FW Rev. FW Rev.
----------------------------------------------------------------------------

  -- No 1553 Devices Found --

Enter command ('h' for help): h

    h    display (h)elp screen
    d    display all (d)evices
    i    display (i)nformation about a device
    s    (s)et logical device number of a device
    f    update the (f)irmware of a device
    t    update the (t)x Inhibit / BC Disable Config of a device
    r    manage (R)emote IP Addresses
    u    save config file and re-scan and (u)pdate device list
    x    abort (config file NOT saved)
    q    (q)uit and save config file


Enter command ('h' for help):
```

Figure 37. Saving Configuration file

The final step is to assign an LDN to each channel on the AIC. This is done by entering '**s**' from the prompt for **(s)et logical device number of a device**. This causes the card manager to ask for which item number, and what LDN number to use for the channel. To switch between ARINC 429 and MIL-STD-1553 devices, the user must close and then reopen **ddccm**.

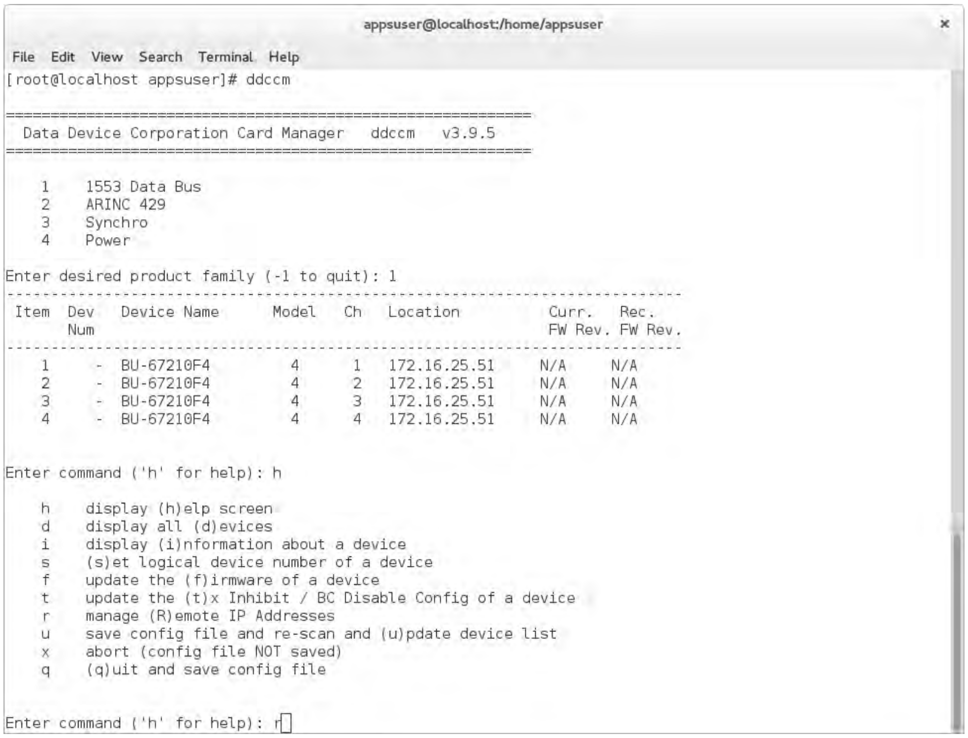


```
---------------------------
    0   172.16.25.51

Enter command ('h' for help): q

----------------------------------------------------------------------------
 Item  Dev   Device Name      Model   Ch   Location          Curr.   Rec.
       Num                                                   FW Rev. FW Rev.
----------------------------------------------------------------------------

  -- No 1553 Devices Found --

Enter command ('h' for help): h

    h    display (h)elp screen
    d    display all (d)evices
    i    display (i)nformation about a device
    s    (s)et logical device number of a device
    f    update the (f)irmware of a device
    t    update the (t)x Inhibit / BC Disable Config of a device
    r    manage (R)emote IP Addresses
    u    save config file and re-scan and (u)pdate device list
    x    abort (config file NOT saved)
    q    (q)uit and save config file


Enter command ('h' for help): u
----------------------------------------------------------------------------
 Item  Dev   Device Name      Model   Ch   Location          Curr.   Rec.
       Num                                                   FW Rev. FW Rev.
----------------------------------------------------------------------------
    1     -  BU-67210F4         4      1   172.16.25.51     N/A     N/A
    2     -  BU-67210F4         4      2   172.16.25.51     N/A     N/A
    3     -  BU-67210F4         4      3   172.16.25.51     N/A     N/A
    4     -  BU-67210F4         4      4   172.16.25.51     N/A     N/A


Enter command ('h' for help): █
```

Figure 38. Listing Found Devices

8.2.2 Removing AIC from Card Manager

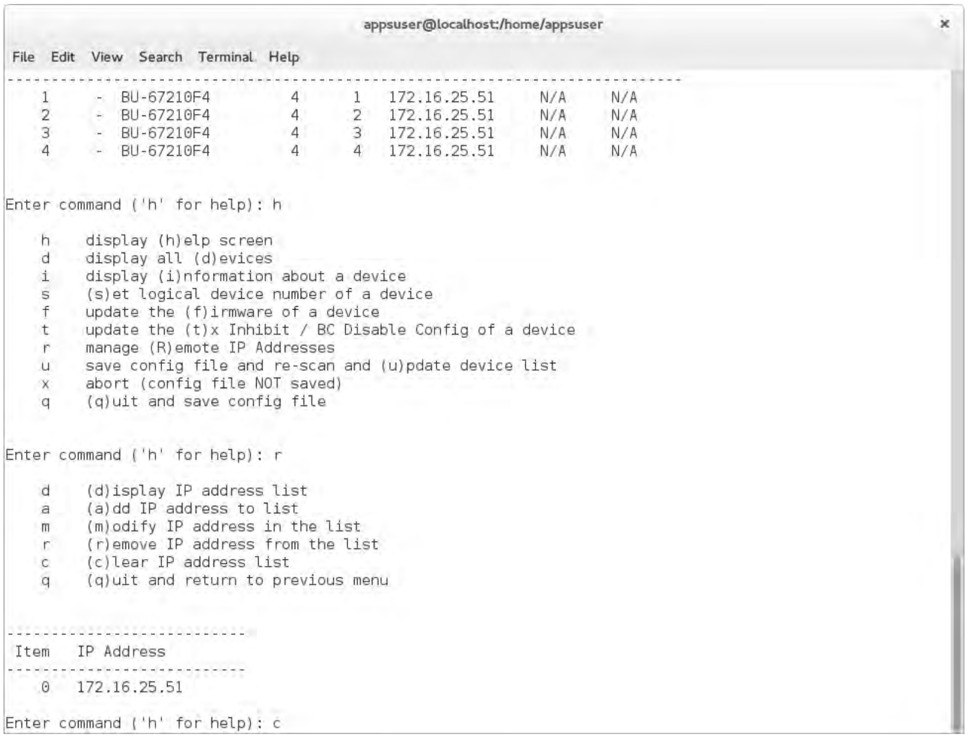
The DDC Card Manager (**ddccm**) can be used to remove the AIC’s IP Address from the searchable IP Address list as well. To remove the IP Address, open the DDC Card Manager by entering in **ddccm** at the prompt with superuser privileges. Select either MIL-STD-1553 or ARINC 429 from the first menu selection. Enter ‘**r**’ to select **(r)emove IP addres from the list**.


appsuser@localhost:/home/appsuser
File Edit View Search Terminal Help

```
[root@localhost appsuser]# ddccm

============================================================
  Data Device Corporation Card Manager   ddccm   v3.9.5
============================================================

    1    1553 Data Bus
    2    ARINC 429
    3    Synchro
    4    Power

Enter desired product family (-1 to quit): 1
-----------------------------------------------------------------------------
 Item  Dev   Device Name      Model   Ch   Location          Curr.   Rec.
       Num                                                   FW Rev. FW Rev.
-----------------------------------------------------------------------------
    1     -  BU-67210F4         4      1   172.16.25.51     N/A     N/A
    2     -  BU-67210F4         4      2   172.16.25.51     N/A     N/A
    3     -  BU-67210F4         4      3   172.16.25.51     N/A     N/A
    4     -  BU-67210F4         4      4   172.16.25.51     N/A     N/A


Enter command ('h' for help): h

    h    display (h)elp screen
    d    display all (d)evices
    i    display (i)nformation about a device
    s    (s)et logical device number of a device
    f    update the (f)irmware of a device
    t    update the (t)x Inhibit / BC Disable Config of a device
    r    manage (R)emote IP Addresses
    u    save config file and re-scan and (u)pdate device list
    x    abort (config file NOT saved)
    q    (q)uit and save config file


Enter command ('h' for help): r
```

Figure 39. Removing IP Address

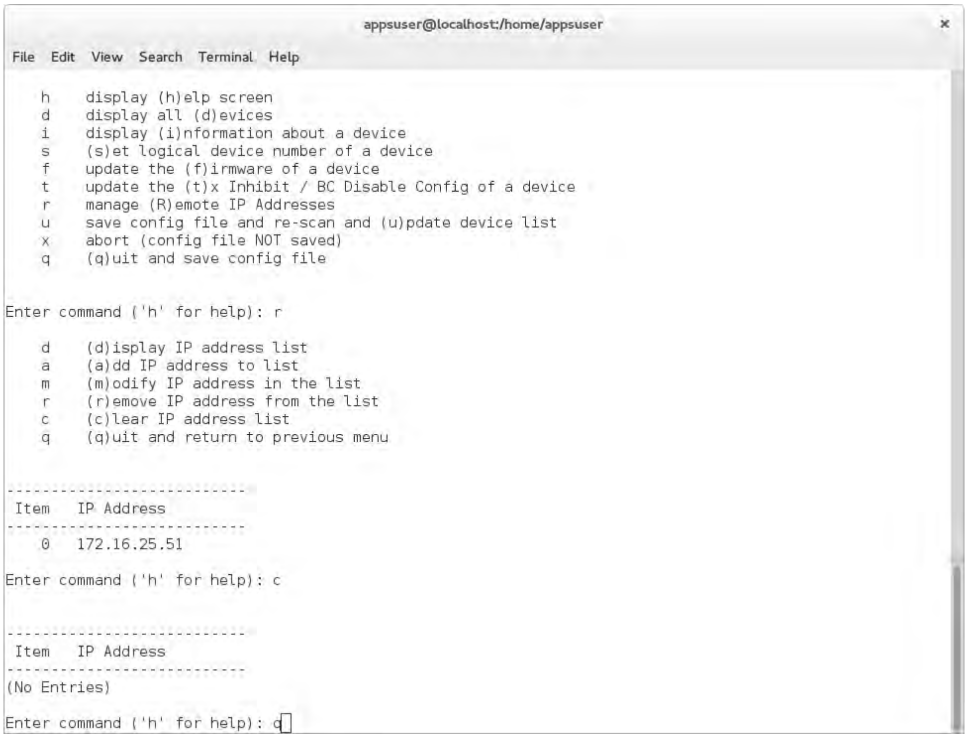
After selecting ‘**r**’ to enter the mange **(R)emote IP Addresses** section of the card manager, select ‘**c**’ to **(c)lear IP address list**. This will remove all IP Addresses from the DDC Card Managers list. If the user wishes to remove a particular IP Address the ‘**r**’ selection can be used.


appsuser@localhost:/home/appsuser
File Edit View Search Terminal Help

```
-----------------------------------------------------------------------------
    1     -  BU-67210F4        4      1   172.16.25.51     N/A      N/A
    2     -  BU-67210F4        4      2   172.16.25.51     N/A      N/A
    3     -  BU-67210F4        4      3   172.16.25.51     N/A      N/A
    4     -  BU-67210F4        4      4   172.16.25.51     N/A      N/A


Enter command ('h' for help): h

    h    display (h)elp screen
    d    display all (d)evices
    i    display (i)nformation about a device
    s    (s)et logical device number of a device
    f    update the (f)irmware of a device
    t    update the (t)x Inhibit / BC Disable Config of a device
    r    manage (R)emote IP Addresses
    u    save config file and re-scan and (u)pdate device list
    x    abort (config file NOT saved)
    q    (q)uit and save config file


Enter command ('h' for help): r

    d    (d)isplay IP address list
    a    (a)dd IP address to list
    m    (m)odify IP address in the list
    r    (r)emove IP address from the list
    c    (c)lear IP address list
    q    (q)uit and return to previous menu


---------------------------
 Item   IP Address
---------------------------
    0   172.16.25.51

Enter command ('h' for help): c
```

Figure 40. Clearing IP Address.

After removing the IP Address from the list, select ‘**q**’ to exit back to Device List portion of the DDC Card Manager.

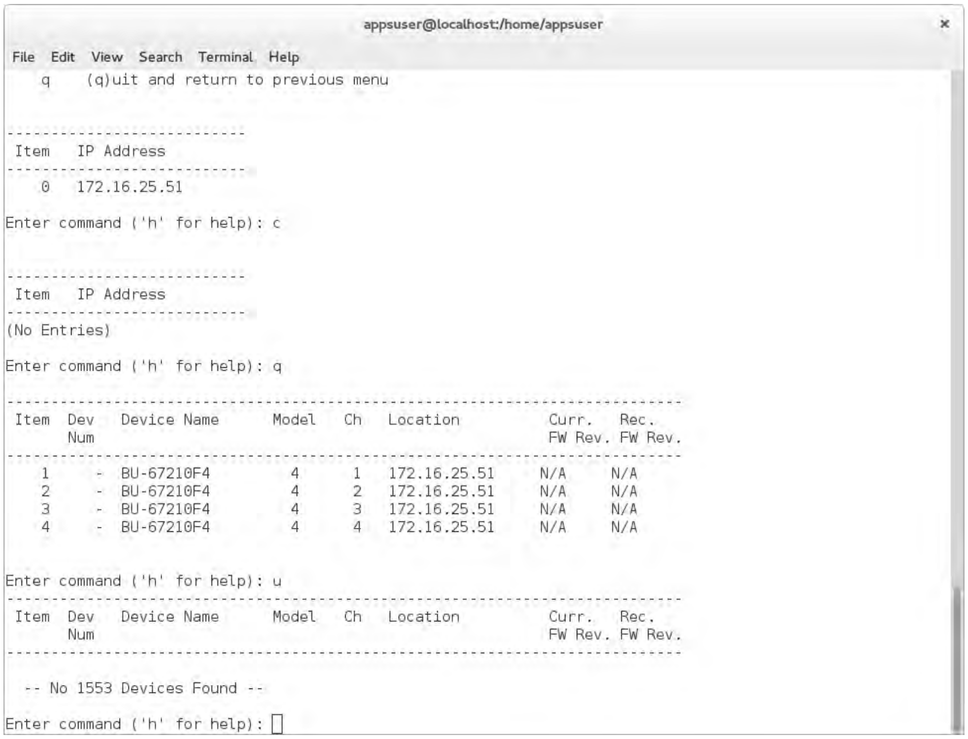


appsuser@localhost:/home/appsuser
File Edit View Search Terminal Help

```
    h    display (h)elp screen
    d    display all (d)evices
    i    display (i)nformation about a device
    s    (s)et logical device number of a device
    f    update the (f)irmware of a device
    t    update the (t)x Inhibit / BC Disable Config of a device
    r    manage (R)emote IP Addresses
    u    save config file and re-scan and (u)pdate device list
    x    abort (config file NOT saved)
    q    (q)uit and save config file


Enter command ('h' for help): r

    d    (d)isplay IP address list
    a    (a)dd IP address to list
    m    (m)odify IP address in the list
    r    (r)emove IP address from the list
    c    (c)lear IP address list
    q    (q)uit and return to previous menu


---------------------------
 Item   IP Address
---------------------------
    0   172.16.25.51

Enter command ('h' for help): c


---------------------------
 Item   IP Address
---------------------------
(No Entries)

Enter command ('h' for help): q
```

Figure 41. Returning to previous menu.

To update the device list, enter ‘**u**’ at the prompt, and the DDC Card Manager will no longer see any AIC devices.


appsuser@localhost:/home/appsuser
File Edit View Search Terminal Help

```
   q    (q)uit and return to previous menu


----------------------------
 Item   IP Address
----------------------------
   0   172.16.25.51

Enter command ('h' for help): c

----------------------------
 Item   IP Address
----------------------------
(No Entries)

Enter command ('h' for help): q

----------------------------------------------------------------------------
 Item  Dev   Device Name      Model   Ch   Location          Curr.   Rec.
       Num                                                   FW Rev. FW Rev.
----------------------------------------------------------------------------
   1     -   BU-67210F4         4      1   172.16.25.51     N/A     N/A
   2     -   BU-67210F4         4      2   172.16.25.51     N/A     N/A
   3     -   BU-67210F4         4      3   172.16.25.51     N/A     N/A
   4     -   BU-67210F4         4      4   172.16.25.51     N/A     N/A


Enter command ('h' for help): u
----------------------------------------------------------------------------
 Item  Dev   Device Name      Model   Ch   Location          Curr.   Rec.
       Num                                                   FW Rev. FW Rev.
----------------------------------------------------------------------------

  -- No 1553 Devices Found --

Enter command ('h' for help): 
```

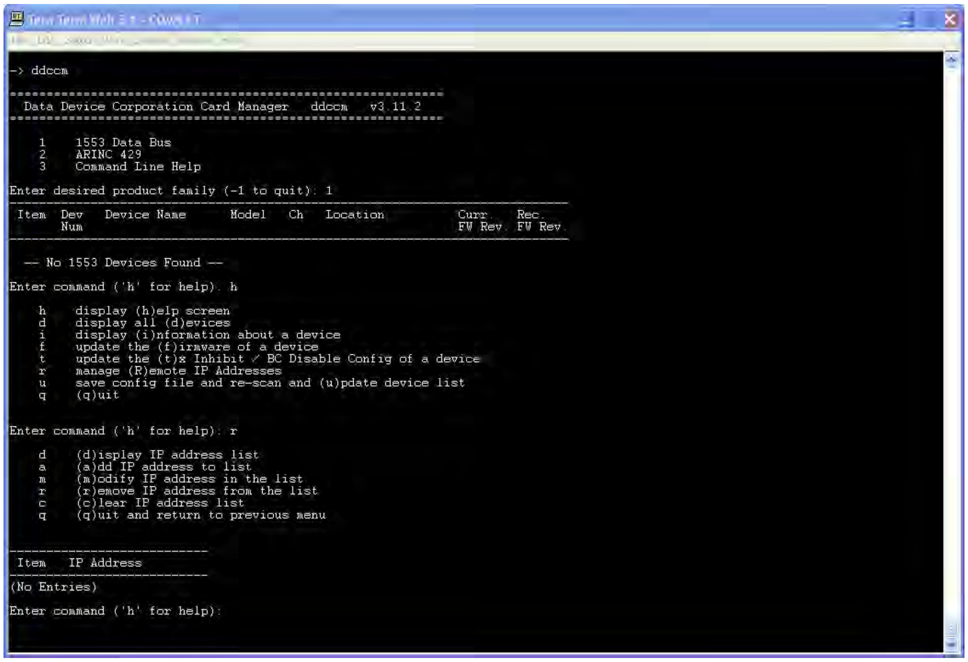
Figure 42. Updating Device List.

8.3 Remote Access Mode With VxWorks

To use the AIC with a VxWorks host Operating system, the DDC Card Manager must be used to find the AIC and assign Logical Device Numbers (LDNs) to the 1553 and 429 channels on the AIC. The DDC Card Manager is bundled with the DDC SDK software in the BU-69092S2 AceXtreme SDK, or the DD-42992S2 ARINC 429 SDK. These SDKs are needed to use the AIC under VxWorks and should be installed per the SDK installation instructions. Once the SDKs have been installed and the LDNs have been assigned, the AIC is ready for use with DDC’s VxWorks software support packages.

8.3.1 Assigning Logical Devices Numbers

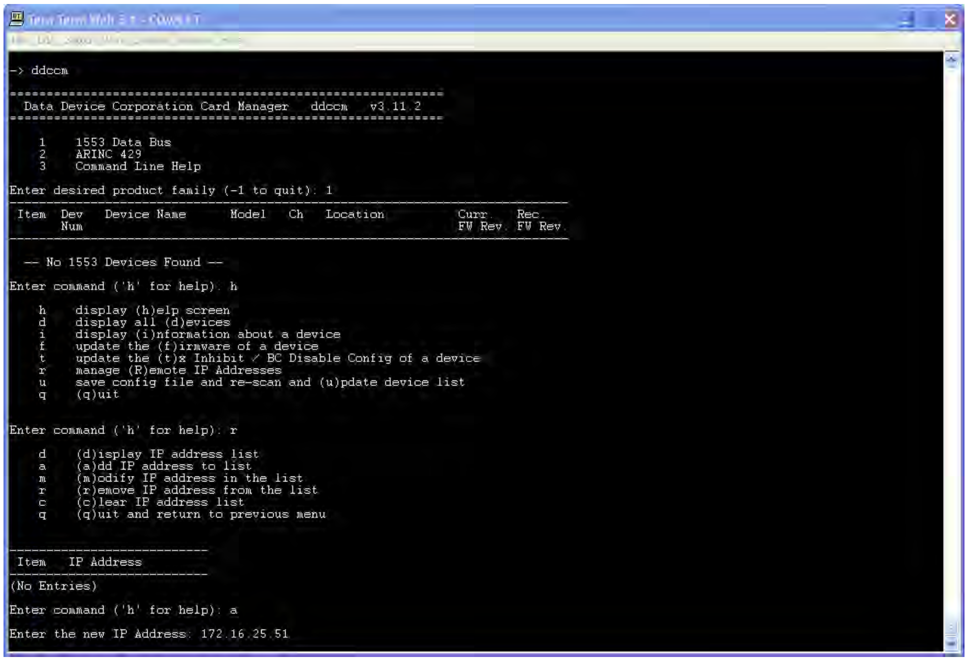
Assigning a LDN to the 1553 and 429 channels on the AIC is done by first opening the DDC Card Manager. This is done from the prompt, by typing **ddccm**. After starting **ddccm**, enter ‘**1’** for **1553 Data Bus channels** or ‘**2’** for **ARINC 429**. Figure 43 shows the MIL-STD-1553 channels that are available when selecting 1 from the prompt. The lack of channels is due to the IP Address of the AIC not being added to the IP Address list.   At the prompt enter in ‘**h**’ to have **ddccm** display the help context menu which displays a list of commands.



```
-> ddccm

============================================================
  Data Device Corporation Card Manager   ddccm   v3.11.2
============================================================

    1    1553 Data Bus
    2    ARINC 429
    3    Command Line Help

Enter desired product family (-1 to quit): 1
---------------------------------------------------------------------------
 Item  Dev   Device Name      Model   Ch   Location          Curr.   Rec.
       Num                                                   FW Rev. FW Rev.
---------------------------------------------------------------------------

  -- No 1553 Devices Found --

Enter command ('h' for help): h

    h    display (h)elp screen
    d    display all (d)evices
    i    display (i)nformation about a device
    f    update the (f)irmware of a device
    t    update the (t)x Inhibit / BC Disable Config of a device
    r    manage (R)emote IP Addresses
    u    save config file and re-scan and (u)pdate device list
    q    (q)uit

Enter command ('h' for help): r

    d    (d)isplay IP address list
    a    (a)dd IP address to list
    m    (m)odify IP address in the list
    r    (r)emove IP address from the list
    c    (c)lear IP address list
    q    (q)uit and return to previous menu

---------------------------
 Item   IP Address
---------------------------
(No Entries)

Enter command ('h' for help):
```

Figure 43. Running ddccm (VxWorks)

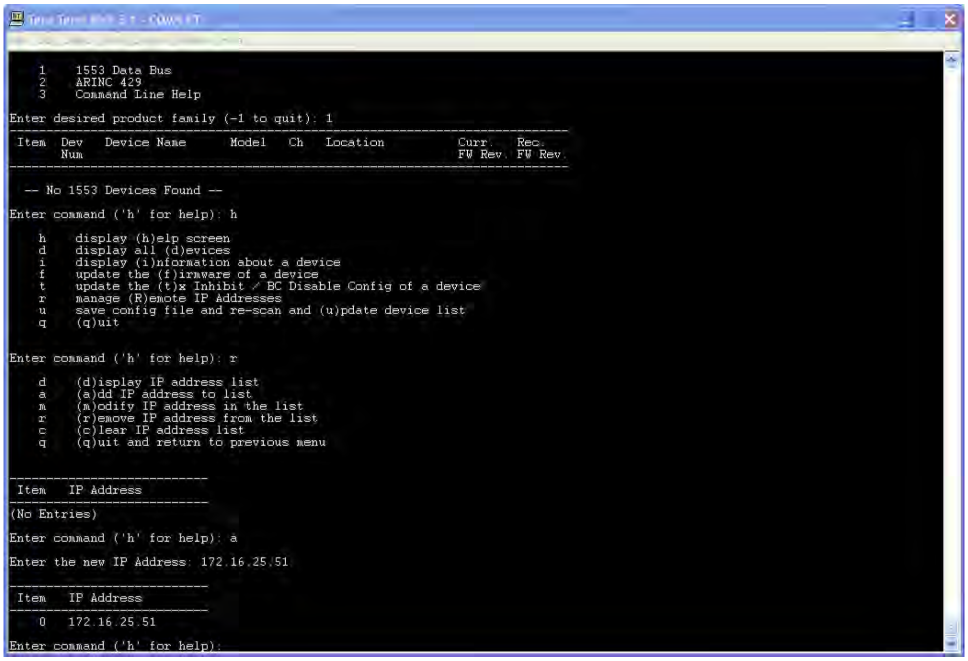
From the prompt, type ‘**a**’ to **(a)dd an IP address to the list** of searchable IP addresses for the card manager. The card manager will now prompt the user for the IP address to add to the list of IP addresses. Enter in the IP address of the AIC and press **enter**.



```
-> ddccm

============================================================
  Data Device Corporation Card Manager   ddccm   v3.11.2
============================================================

    1    1553 Data Bus
    2    ARINC 429
    3    Command Line Help

Enter desired product family (-1 to quit): 1
---------------------------------------------------------------------------
 Item  Dev   Device Name      Model   Ch   Location          Curr.   Rec.
       Num                                                   FW Rev. FW Rev.
---------------------------------------------------------------------------

  -- No 1553 Devices Found --

Enter command ('h' for help): h

    h    display (h)elp screen
    d    display all (d)evices
    i    display (i)nformation about a device
    f    update the (f)irmware of a device
    t    update the (t)x Inhibit / BC Disable Config of a device
    r    manage (R)emote IP Addresses
    u    save config file and re-scan and (u)pdate device list
    q    (q)uit

Enter command ('h' for help): r

    d    (d)isplay IP address list
    a    (a)dd IP address to list
    m    (m)odify IP address in the list
    r    (r)emove IP address from the list
    c    (c)lear IP address list
    q    (q)uit and return to previous menu

---------------------------
 Item   IP Address
---------------------------
(No Entries)

Enter command ('h' for help): a

Enter the new IP Address: 172.16.25.51
```

Figure 44. Selecting Add / Entering IP Address of AIC

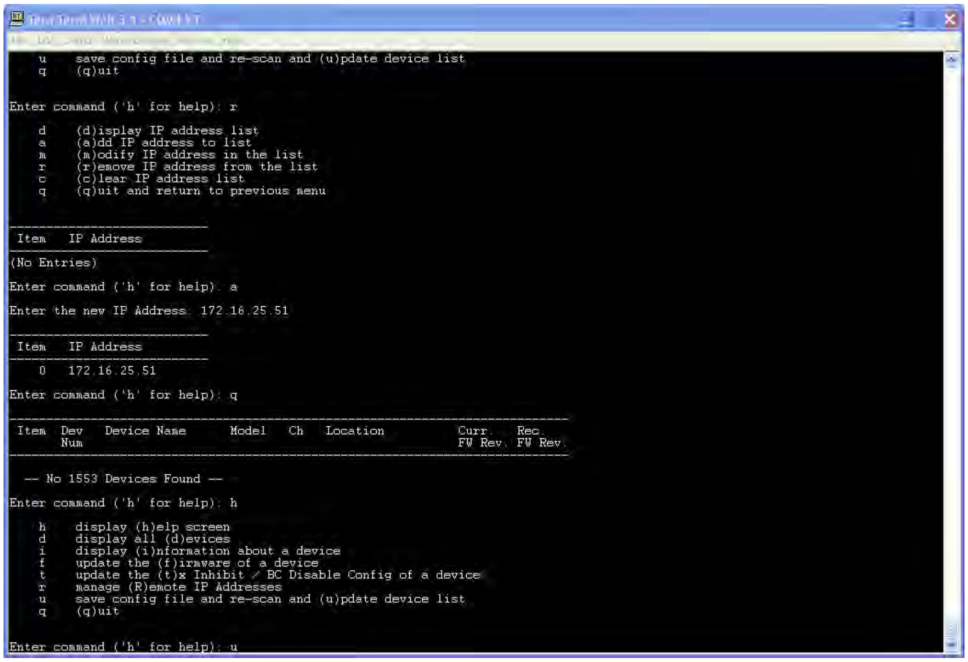
After entering in the IP address, type ‘**h**’ at the prompt again to display a new list of selectable commands to perform within **ddccm**. Type ‘**q**’ to exit the current level and return back to the Device listing portion of the **ddccm**.



```
    1    1553 Data Bus
    2    ARINC 429
    3    Command Line Help

Enter desired product family (-1 to quit): 1
-------------------------------------------------------------------------------
 Item  Dev   Device Name      Model   Ch   Location            Curr.   Rec.
       Num                                                     FW Rev. FW Rev.
-------------------------------------------------------------------------------

  -- No 1553 Devices Found --

Enter command ('h' for help): h

    h    display (h)elp screen
    d    display all (d)evices
    i    display (i)nformation about a device
    f    update the (f)irmware of a device
    t    update the (t)x Inhibit / BC Disable Config of a device
    r    manage (R)emote IP Addresses
    u    save config file and re-scan and (u)pdate device list
    q    (q)uit


Enter command ('h' for help): r

    d    (d)isplay IP address list
    a    (a)dd IP address to list
    m    (m)odify IP address in the list
    r    (r)emove IP address from the list
    c    (c)lear IP address list
    q    (q)uit and return to previous menu


---------------------------
 Item   IP Address
---------------------------
(No Entries)

Enter command ('h' for help): a

Enter the new IP Address: 172.16.25.51

---------------------------
 Item   IP Address
---------------------------
    0   172.16.25.51

Enter command ('h' for help):
```

Figure 45. Entering IP Address

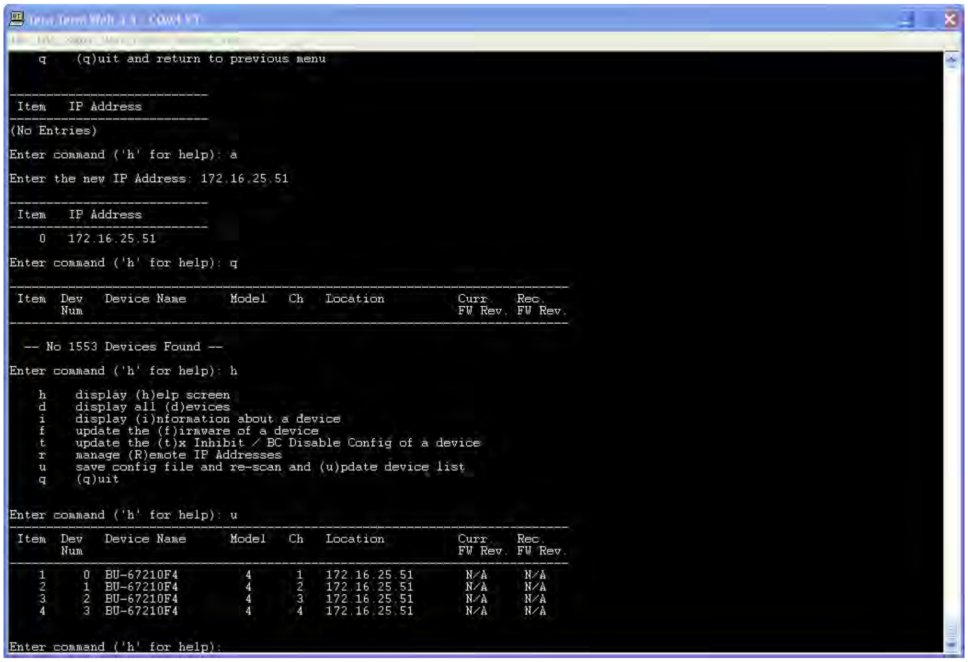
From the prompt, select ‘**u**’ to **save the configuration file and to perform a re-scan and (u)pdate device list** for MIL-STD-1553 and ARINC 429 devices. Any found devices on the AIC will be displayed as shown by Figure 46. To see the ARINC 429 devices, **ddccm** must be exited and run again.



```
    u    save config file and re-scan and (u)pdate device list
    q    (q)uit


Enter command ('h' for help): r

    d    (d)isplay IP address list
    a    (a)dd IP address to list
    m    (m)odify IP address in the list
    r    (r)emove IP address from the list
    c    (c)lear IP address list
    q    (q)uit and return to previous menu


---------------------------
 Item   IP Address
---------------------------
(No Entries)

Enter command ('h' for help): a

Enter the new IP Address: 172.16.25.51

---------------------------
 Item   IP Address
---------------------------
    0   172.16.25.51

Enter command ('h' for help): q

----------------------------------------------------------------------------
 Item  Dev   Device Name      Model   Ch   Location          Curr.   Rec.
       Num                                                   FW Rev. FW Rev.
----------------------------------------------------------------------------

  -- No 1553 Devices Found --

Enter command ('h' for help): h

    h    display (h)elp screen
    d    display all (d)evices
    i    display (i)nformation about a device
    f    update the (f)irmware of a device
    t    update the (t)x Inhibit / BC Disable Config of a device
    r    manage (R)emote IP Addresses
    u    save config file and re-scan and (u)pdate device list
    q    (q)uit


Enter command ('h' for help): u
```

Figure 46. Save Configuration file

The final step is to assign an LDN to each channel on the AIC. This is done by entering ‘**s**’ from the prompt for **(s)et logical device number of a device**. This causes the card manager to ask for which item number, and what is the LDN number to use for the channel. To switch between ARINC 429 and MIL-STD-1553 devices, the user must close and then reopen **ddccm**.

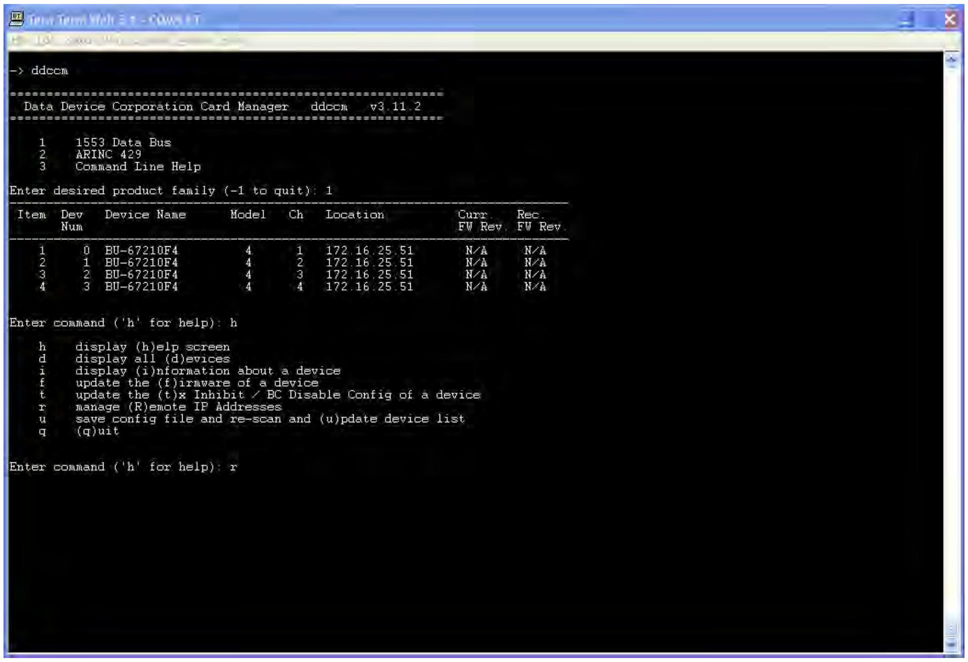


```
    q    (q)uit and return to previous menu


---------------------------
 Item   IP Address
---------------------------
(No Entries)

Enter command ('h' for help): a

Enter the new IP Address: 172.16.25.51

---------------------------
 Item   IP Address
---------------------------
    0   172.16.25.51

Enter command ('h' for help): q

------------------------------------------------------------------------------
 Item  Dev   Device Name      Model   Ch   Location           Curr     Rec.
       Num                                                    FW Rev.  FW Rev.
------------------------------------------------------------------------------

  -- No 1553 Devices Found --

Enter command ('h' for help): h

    h    display (h)elp screen
    d    display all (d)evices
    i    display (i)nformation about a device
    f    update the (f)irmware of a device
    t    update the (t)x Inhibit / BC Disable Config of a device
    r    manage (R)emote IP Addresses
    u    save config file and re-scan and (u)pdate device list
    q    (q)uit


Enter command ('h' for help): u
------------------------------------------------------------------------------
 Item  Dev   Device Name      Model   Ch   Location           Curr     Rec.
       Num                                                    FW Rev.  FW Rev.
------------------------------------------------------------------------------
    1     0  BU-67210F4         4      1   172.16.25.51        N/A      N/A
    2     1  BU-67210F4         4      2   172.16.25.51        N/A      N/A
    3     2  BU-67210F4         4      3   172.16.25.51        N/A      N/A
    4     3  BU-67210F4         4      4   172.16.25.51        N/A      N/A


Enter command ('h' for help):
```

Figure 47. Listing Found Devices

8.3.2 Removing AIC from Card Manager

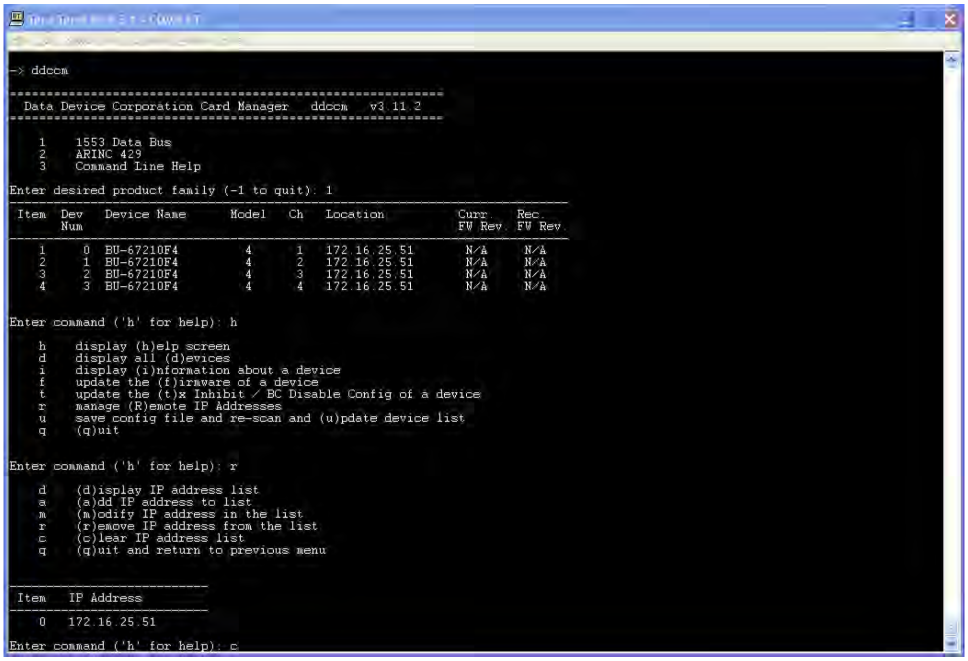
The DDC Card Manager (**ddccm**) can be used to remove the AIC's IP Address from the searchable IP Address list as well. To remove the IP Address, open the DDC Card Manager by typing **ddccm** at the prompt with superuser privileges. Select either MIL-STD-1553 or ARINC 429 from the first menu selection. Enter '**r**' to select **(r)emove IP addres from the list**.



```
-> ddccm

==========================================================
  Data Device Corporation Card Manager   ddccm   v3.11.2
==========================================================

    1    1553 Data Bus
    2    ARINC 429
    3    Command Line Help

Enter desired product family (-1 to quit): 1
------------------------------------------------------------------------------
 Item  Dev   Device Name      Model   Ch   Location           Curr.    Rec.
       Num                                                    FW Rev.  FW Rev.
------------------------------------------------------------------------------
    1     0  BU-67210F4         4      1   172.16.25.51        N/A      N/A
    2     1  BU-67210F4         4      2   172.16.25.51        N/A      N/A
    3     2  BU-67210F4         4      3   172.16.25.51        N/A      N/A
    4     3  BU-67210F4         4      4   172.16.25.51        N/A      N/A


Enter command ('h' for help): h

    h    display (h)elp screen
    d    display all (d)evices
    i    display (i)nformation about a device
    f    update the (f)irmware of a device
    t    update the (t)x Inhibit / BC Disable Config of a device
    r    manage (R)emote IP Addresses
    u    save config file and re-scan and (u)pdate device list
    q    (q)uit


Enter command ('h' for help): r
```

Figure 48. Removing IP Address

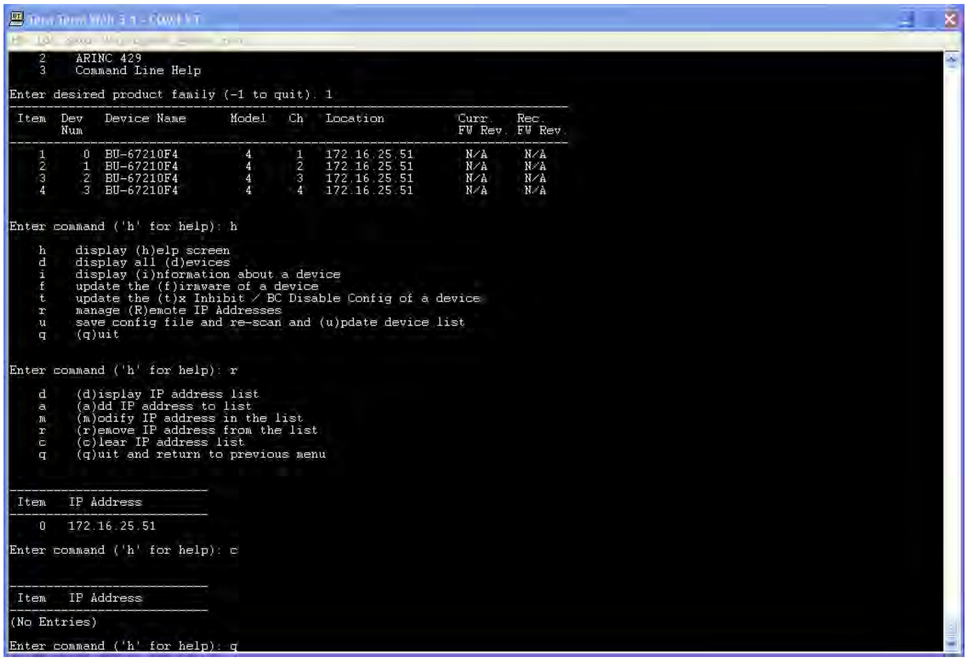
After selecting ‘**r**’ to enter the mange **(R)emote IP Addresses** section of the card manager, select ‘**c**’ to **(c)lear IP address list**. This will remove all IP Addresses from the DDC Card Managers list. If the user wishes to remove a particular IP Address the ‘**r**’ selection can be used.



```
-> ddccm

==========================================================
  Data Device Corporation Card Manager   ddccm   v3.11.2
==========================================================

    1    1553 Data Bus
    2    ARINC 429
    3    Command Line Help

Enter desired product family (-1 to quit): 1
------------------------------------------------------------------------------
 Item  Dev   Device Name      Model   Ch   Location           Curr.    Rec.
       Num                                                    FW Rev.  FW Rev.
------------------------------------------------------------------------------
    1     0  BU-67210F4         4      1   172.16.25.51        N/A      N/A
    2     1  BU-67210F4         4      2   172.16.25.51        N/A      N/A
    3     2  BU-67210F4         4      3   172.16.25.51        N/A      N/A
    4     3  BU-67210F4         4      4   172.16.25.51        N/A      N/A


Enter command ('h' for help): h

    h    display (h)elp screen
    d    display all (d)evices
    i    display (i)nformation about a device
    f    update the (f)irmware of a device
    t    update the (t)x Inhibit / BC Disable Config of a device
    r    manage (R)emote IP Addresses
    u    save config file and re-scan and (u)pdate device list
    q    (q)uit


Enter command ('h' for help): r

    d    (d)isplay IP address list
    a    (a)dd IP address to list
    m    (m)odify IP address in the list
    r    (r)emove IP address from the list
    c    (c)lear IP address list
    q    (q)uit and return to previous menu


---------------------------
 Item   IP Address
---------------------------
    0   172.16.25.51

Enter command ('h' for help): c
```

Figure 49. Clearing IP Address

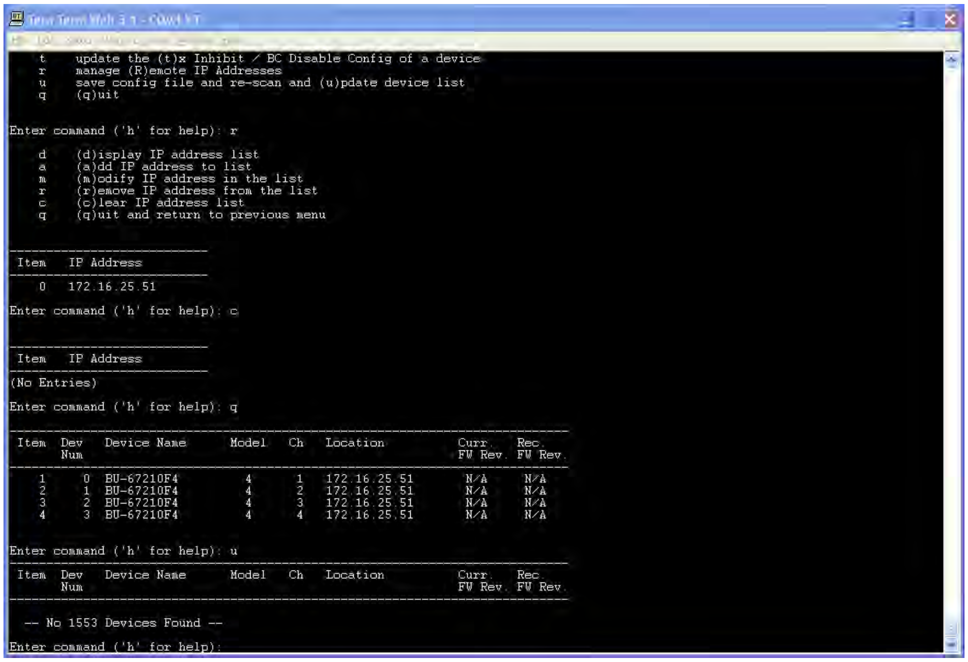
After removing the IP Address from the list, select ‘**q**’ to exit back to Device List portion of the DDC Card Manager.



```
    2    ARINC 429
    3    Command Line Help

Enter desired product family (-1 to quit): 1
---------------------------------------------------------------------------------
 Item  Dev   Device Name      Model   Ch   Location          Curr.   Rec.
       Num                                                   FW Rev. FW Rev.
---------------------------------------------------------------------------------
    1     0  BU-67210F4         4      1   172.16.25.51       N/A     N/A
    2     1  BU-67210F4         4      2   172.16.25.51       N/A     N/A
    3     2  BU-67210F4         4      3   172.16.25.51       N/A     N/A
    4     3  BU-67210F4         4      4   172.16.25.51       N/A     N/A


Enter command ('h' for help): h

    h    display (h)elp screen
    d    display all (d)evices
    i    display (i)nformation about a device
    f    update the (f)irmware of a device
    t    update the (t)x Inhibit / BC Disable Config of a device
    r    manage (R)emote IP Addresses
    u    save config file and re-scan and (u)pdate device list
    q    (q)uit


Enter command ('h' for help): r

    d    (d)isplay IP address list
    a    (a)dd IP address to list
    m    (m)odify IP address in the list
    r    (r)emove IP address from the list
    c    (c)lear IP address list
    q    (q)uit and return to previous menu


---------------------------
 Item   IP Address
---------------------------
    0   172.16.25.51

Enter command ('h' for help): c


---------------------------
 Item   IP Address
---------------------------
(No Entries)

Enter command ('h' for help): q
```

Figure 50. Returning to Previous Menu

To update the device list, enter ‘**u**’ at the prompt, and the DDC Card Manager will no longer see any AIC devices.



```
    t    update the (t)x Inhibit / BC Disable Config of a device
    r    manage (R)emote IP Addresses
    u    save config file and re-scan and (u)pdate device list
    q    (q)uit


Enter command ('h' for help): r

    d    (d)isplay IP address list
    a    (a)dd IP address to list
    m    (m)odify IP address in the list
    r    (r)emove IP address from the list
    c    (c)lear IP address list
    q    (q)uit and return to previous menu



---------------------------
 Item   IP Address
---------------------------
    0   172.16.25.51

Enter command ('h' for help): c


---------------------------
 Item   IP Address
---------------------------
(No Entries)

Enter command ('h' for help): q

-----------------------------------------------------------------------------
 Item  Dev   Device Name      Model   Ch   Location          Curr.   Rec.
       Num                                                   FW Rev. FW Rev.
-----------------------------------------------------------------------------
    1     0  BU-67210F4         4      1   172.16.25.51       N/A     N/A
    2     1  BU-67210F4         4      2   172.16.25.51       N/A     N/A
    3     2  BU-67210F4         4      3   172.16.25.51       N/A     N/A
    4     3  BU-67210F4         4      4   172.16.25.51       N/A     N/A


Enter command ('h' for help): u
-----------------------------------------------------------------------------
 Item  Dev   Device Name      Model   Ch   Location          Curr.   Rec.
       Num                                                   FW Rev. FW Rev.
-----------------------------------------------------------------------------

  -- No 1553 Devices Found --

Enter command ('h' for help):
```

Figure 51. Updating Device List

9 UPGRADING AND RECOVERING THE DEVICE

The Avionics Interface Computer is capable of In-Field upgrading and recovery. In-Field upgrading allows the AIC to be updated to DDC latest Drivers and Library without having to ship the unit back to DDC. A bootable USB storage device is shipped along with the AIC, and can be use to perform the Operating System restore procedure. The restore procedure is used to flash a backup image of the original configuration, or user created imaged onto the Avionics Interface Computer. Care should be taken when using the Operating System Restore as all settings and data will be erased after performing the recovery operation.

9.1 Operating System Restore

When necessary, the supplied USB **OS Recovery Device** can be used to perform a Operating System restore on the Avionics Interface Computer to put the AIC back into a know default state (factory settings). To perform the recovery, plug the USB **OS Recovery Device** into one of the USB ports on the AIC.

Figure 52. USB OS Recovery Device.

Power on the AIC and the OS recovery process will automatically begin without any user interaction. The recovery process should take about five minutes, and the AIC should turn off when the recovery has completed.

Once the recovery process has completed, and the AIC has powered down, remove the USB **OS Recovery Device** and power the AIC back on. The AIC will now be back to its factory defaults. Any files that were on the AIC will have been erased as this process installs a new image onto the AIC's hard drive.

Note: *The* ***OS Recovery Device*** *is* ***bootable****, and configured to automatically start the recovery process after a thirty second timeout. Booting your system with the* ***USB OS Recovery Device*** *connected (and USB boot support enabled in your BIOS) will cause the recovery software to perform the recovery operation on your system causing a loss of all your data.*

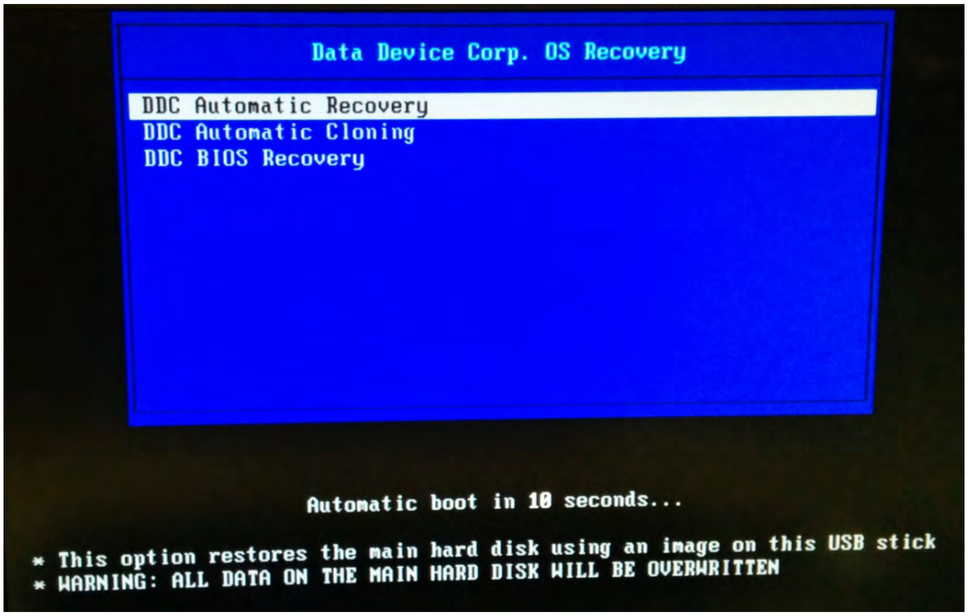
9.1.1 Creating USB OS Recovery Device

If necessary, a new USB OS Recovery device may be created with the BU-69094R1 Recovery package. To create a new recovery device, an 8 GB USB flash drive or larger is needed, and a computer running Fedora with a network connection to install RPMs is required. The BU-69094R1 comes with a readme.txt file which documents the steps needed to create the recovery device.

9.2 Creating a Backup Image

The USB **OS Recovery Device** also has the option of creating a backup image of the AIC. This backup image will replace the current image on the USB **OS Recovery Device**, so that the next time the OS Recovery Device is used to reimage the AIC, the newly created image will be deployed onto the AIC.

To create a backup image, with the AIC powered down, plug in the USB **OS Recovery Device** into one of the AIC's available USB ports. Before powering on the AIC, make sure you have a monitor and keyboard connected to the AIC. Power on the AIC and select **DDC Automatic Cloning** from the boot menu.



```
Automatic boot in 10 seconds...

* This option restores the main hard disk using an image on this USB stick
* WARNING: ALL DATA ON THE MAIN HARD DISK WILL BE OVERWRITTEN
```

Figure 53. OS Recovery Device – Boot Menu.

Note: *The USB **OS Recovery Device** is configured to automatically boot into Recovery mode after a thirty second timeout. Recovery mode will flash the saved image from the USB **OS Recovery Device** onto the AIC, erasing all data currently on the AIC.*

After selection the **DDC Automatic Cloning** option, the cloning of the AIC will automatically start. The process if fully automated and will copy the new image onto the USB **OS Recovery Device**. Once the cloning procedure has completed, the AIC will be powered off. At this point the **OS Recovery Device** must be removed before powering on the AIC.

9.3 Bios Update

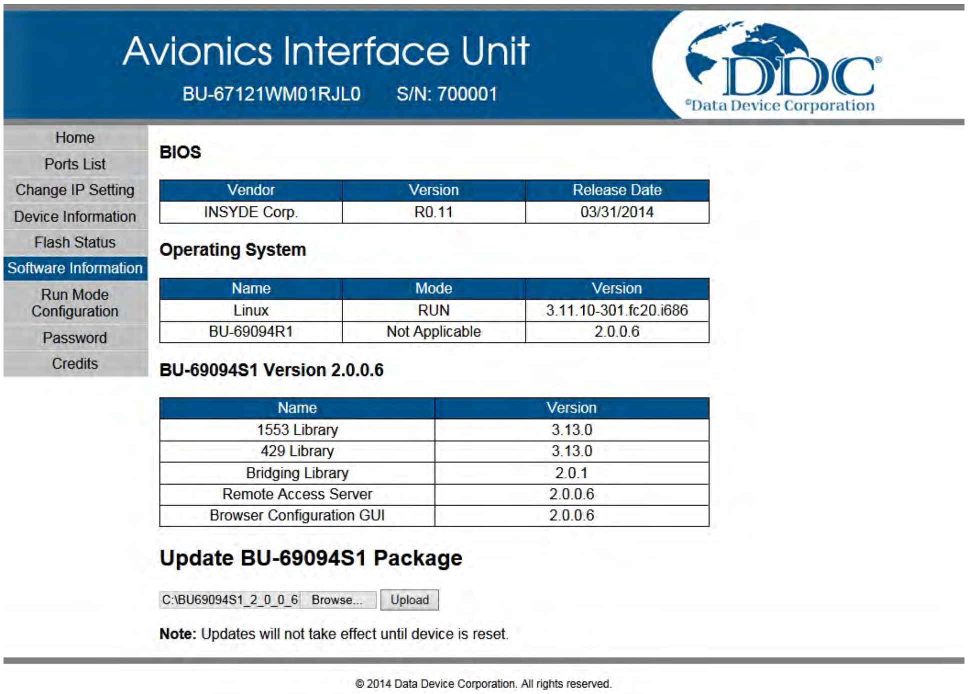
The USB **OS Recovery Device** also comes with the AIC's default BIOS image. The default BIOS image can be flashed onto the AIC, by booting with USB **OS Recovery Device** connected to the AIC's USB port.

> ***Note:*** *The USB **OS Recovery Device** is configured to automatically boot into Recovery mode after a thirty second timeout. Recovery mode will flash the saved image from the USB **OS Recovery Device** onto the AIC, erasing all data currently on the AIC.*

At the USB **OS Recovery Device's** boot menu, select **DDC BIOS Recovery** to start the BIOS flash procedure. Once the flash procedure has been completed, the AIC will power down. The USB **OS Recovery Device** should be removed before the AIC is restarted.

9.4 HTTP (Web) Update Method

A Web Server may be used to update the DDC SDKs and drivers. The **Software Information** page on the Avionics Interface Computer displays the installed MIL-STD-1553, ARINC 429 SDK library versions, as well as the versions of the Bridging (Protocol Conversion) API, Remote Access Server, and the Browser Configuration GUI. The **Browse** and **Upload** buttons can be used to upload a new **BU-69094S1** package, which contains the following software: MIL-STD-1553, ARINC 429, Bridging, Remote Access Server and Web Browser.



BIOS

Vendor	Version	Release Date
INSYDE Corp.	R0.11	03/31/2014

Operating System

Name	Mode	Version
Linux	RUN	3.11.10-301.fc20.i686
BU-69094R1	Not Applicable	2.0.0.6

BU-69094S1 Version 2.0.0.6

Name	Version
1553 Library	3.13.0
429 Library	3.13.0
Bridging Library	2.0.1
Remote Access Server	2.0.0.6
Browser Configuration GUI	2.0.0.6

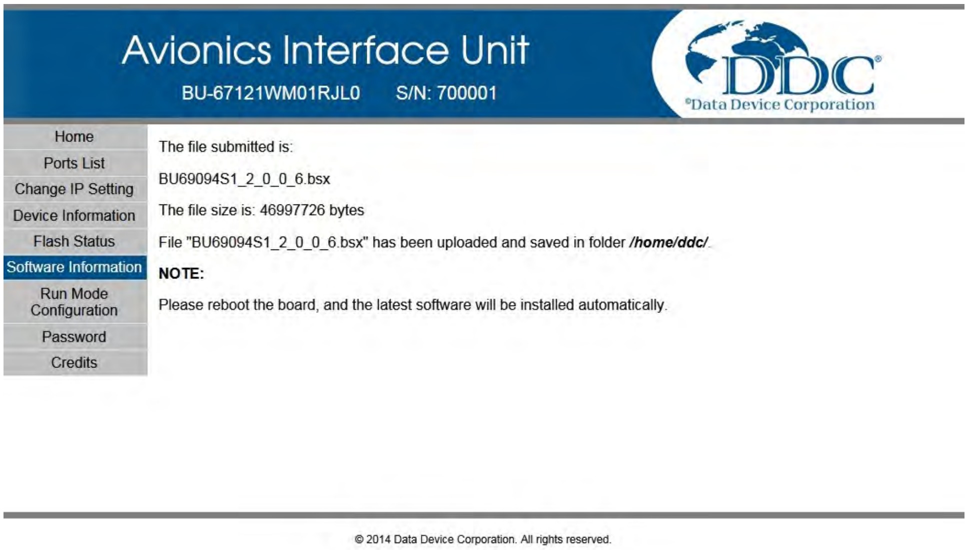
Update BU-69094S1 Package

C:\BU69094S1_2_0_0_6

Note: Updates will not take effect until device is reset.

Figure 54. Avionics Interface Computer Software Update

To update the software on the AIC, click **Browse** and select the **BU69094S1_X_Y_Z.bsx** file (where XYZ is the version of the BU-69094S1). Once the file has been selected, click **Upload** to install the new library software onto the AIC.



The file submitted is:

BU69094S1_2_0_0_6.bsx

The file size is: 46997726 bytes

File "BU69094S1_2_0_0_6.bsx" has been uploaded and saved in folder ***/home/ddc/***.

NOTE:

Please reboot the board, and the latest software will be installed automatically.

Figure 55. Avionics Interface Computer Software Update Complete

Once the software has been uploaded, the Web Server will display the message shown in Figure 55. The AIC needs to be rebooted to complete the installation of the new software. Once the AIC is rebooted, the new software is ready for use.

Device Firmware can also be updated in a similar fashion by the **Update Firmware** button on the **Device Information Tab**. This task is documented in the AIC's hardware manual.

Data Device Corporation

Leadership Built on Over 50 Years of Innovation

Military | Civil Aerospace | Space | Industrial

Data Device Corporation (DDC) is a world leader in the design and manufacture of high-reliability Connectivity, Power and Control solutions (Data Networking; Power Distribution, Control and Conversion; Motor Control and Motion Feedback) for aerospace, defense, space, and industrial applications. With awards for quality, delivery and support, DDC has served these industries as a trusted resource for more than 50 years... providing proven solutions optimized for efficiency, reliability, and performance. Data Device Corporation brands include DDC, Beta Transformer Technology Corporation, National Hybrid Inc., North Hills Signal Processing Corporation, Pascall Electronics Ltd., and XCEL Power Systems Ltd. DDC is headquartered in Bohemia, NY and has manufacturing operations in New York, California, Mexico, and the United Kingdom.

Beta Transformer Technology Corporation, a subsidiary of DDC and leader in high reliability transformer, magnetic and cable assembly solutions for the aerospace, defense, and space industries, offers field proven transformer solutions for the most demanding industrial environments… extreme temperature, shock, vibration, dust, fluid, and radiation. Beta Transformer developed many of the world's smallest transformers and inductors, and is recognized for superior quality and performance. Beta Transformer headquarters along with their main design and manufacturing operations are located in Bohemia, NY. Beta has expanded production capabilities through their manufacturing operations at Beta Transformer Mexico, S. DE R L. DE C.V., located in Ensenada, Mexico, and North Hills Signal Processing Corporation in H. Matamoros Tamaulipas, Mexico, both subsidiaries of Beta Transformer Technology Corporation.

XCEL Power Systems and Pascall Electronics are divisions of DDC Electronics, Ltd., a subsidiary of Data Device Corporation. DDC Electronics, Ltd. specializes in the design and manufacture of power supply solutions for extreme environments. With over 30 years of experience in the defense, aerospace and industrial sectors, DDC Electronics is a trusted source for complete solutions in the design, development and manufacture of electronic power conversion products – from single converters to complex multi- function conversion systems. DDC Electronics products are the first choice for power with In-Flight Entertainment & Connectivity (IFEC) and defense systems. There are more than 170,000 Pascall power supply units installed on commercial aircraft. XCEL and Pascall power supply units are in service with Ground, Air and Naval forces across the world, powering state of the art electronic systems, and trusted by industry leaders to deliver reliable proven performance in some of the most challenging environments to be found anywhere. DDC Electronics, Ltd. headquarters, along with the XCEL Power Systems design operations and the Pascall Electronics factory are located in the UK.

DDC Microelectronics, a division of Data Device Corporation and formerly the space microelectronics division of Maxwell Technologies, is a leading developer and manufacturer of innovative, cost-effective, space-qualified microelectronics solutions for satellites and spacecraft. DDC Microelectronics has provided space-qualified radiation-tolerant and radiation-shielded products, including semiconductors and single-board computers, to the space industry for more than two decades. DDC radiation mitigated power modules, memory modules, and single board computers incorporate powerful commercial silicon for superior performance and high reliability in space applications. DDC Microelectronics specializes in understanding the radiation performance of commercial semiconductors, qualifying selected components for use in space, integrating them with proprietary radiation mitigation technologies, and manufacturing and screening these products in a DLA approved MIL-PRF-38534 facility, located in southern California.

Your Solution Provider for... Connectivity, Power, and Control

Connectivity

Data Bus Solutions
DDC is the market leader in high reliability data bus solutions for MIL-STD-1553/1760, ARINC 429, Fibre Channel, Ethernet, CANbus, Serial I/O and other protocols, and is one of the few companies able to provide a full range of computers, boards, hybrids and ASIC solutions for aerospace, defense and space applications.

Power

Power Supplies
DDC supplies highly customized power products to the aerospace, defense, maritime and satellite communications industries.

Solid-State Power Controllers
DDC's programmable solid-state power controllers provide simple and reliable power management for aerospace and defense systems.

Control

Motor Controllers and Drives
DDC is the world leader in high reliability torque, speed, and position controllers and drives engineered to operate in demanding environments.

Motion Feedback
DDC is the world leader in the design and manufacture of Synchro/Resolver-to-Digital and Digital-to-Synchro/Resolver converters.

Certifications

Data Device Corporation is ISO 9001:2008, AS 9100 Rev C, EN 9100, and JIS Q9100 certified. DDC has been granted certification by the Defense Logistics Agency, Land & Maritime (DLA) for manufacturing Class D, G, H, and K hybrid products in accordance with MIL-PRF-38534. Industry documents used to support DDC's certifications and Quality system are MIL-STD-883, ANSI/NCSL Z540-1, IPC-A-610, MIL-STD-202, JESD-22, and J-STD-020.

Beta Transformer Technology Corporation (BTTC) and its subsidiaries are ISO 9001:2008 and AS 9100 Rev C certified. BTTC has been granted certification as a qualified source of transformers by the Defense Logistics Agency, Land & Maritime (DLA) and is listed on the QPL for products MIL-PRF 21038/27-01 through -31 Product Levels C, M and T.

DDC Electronics, Ltd.'s XCEL Power Systems and Pascall Electronics manufacturing operations are ISO 9001:2008, AS 9100 Rev C, EN9100 and ISO 14001:2004 certified.

DDC®

Connectivity Power Control

Your Solution Provider for... Connectivity, Power, and Control

DDC Electronics Ltd. | BETA TRANSFORMER TECHNOLOGY CORPORATION | NHi National Hybrid Incorporated | North Hills™ | Pascall | XCEL Power Systems

The first choice for more than 50 years—DDC

DDC is the world leader in the design and manufacture of high reliability data interface products, motion control, and solid-state power controllers for aerospace, defense, and industrial automation.

Inside the U.S. : Call Toll-Free 1-800-DDC-5757

©Data Device Corporation

UL REGISTERED FIRM
DATA DEVICE CORPORATION
REGISTERED TO:
ISO 9001:2008, AS9100C:2009-01
EN9100:2009, JIS Q9100:2009
FILE NO. 10001296 ASH09

Certified System Environment ISO 14001 SAI GLOBAL

Certified System SAI GLOBAL AS9100 Aerospace

Certified System SAI GLOBAL ISO 9001 Quality

DDC Headquarters and Main Factory
105 Wilbur Place, Bohemia, NY 11716-2426
Tel: (631) 567-5600
Toll-Free, Customer Service: 1-800-DDC-5757
www.ddc-web.com

DDC Microelectronics
13000 Gregg Street, Suite C, Poway, CA 92064
Tel: (631) 567-5600
Toll-Free, Customer Service: 1-800-DDC-5757

Beta Transformer Technology Corporation
40 Orville Drive, Bohemia, NY 11716-2426
Tel: (631) 244-7393
www.BTTC-Beta.com

Beta Transformer Mexico, S. DE R L. DE C.V.
Avedina 20 De Noviembre
959 Zona Centro, Ensenada, Baja Mexico
Tel: (631) 244-7393

North Hills Signal Processing Corporation
6851 Jericho Turnpike, Syosset, NY 11791
Tel: (631) 244-7393
www.nhsignal.com

North Hills Signal Processing Corporation
Avedina Jose Escandon y Helquera No. 21
Km. 8.5 Carretera Lauro Villar
H. Matamoros Tamaulipas, Mexico
Tel: (631) 244-7393

Outside the U.S. : Call 1-631-567-5600

DDC Electronics Ltd.

BSI UKAS

BSI UKAS

ADS 21ST CENTURY SUPPLY CHAINS SC21 Bronze Award

DDC Electronics Ltd Headquarters and Pascall Electronics Ltd. Factory
Westbridge Business Park, Cothey Way
Ryde, Isle of Wight, PO33 1QT, UK
Tel: +44 (0) 1983 817300
www.pascall.co.uk

United Kingdom: DDC U.K., Ltd
James House, 27-35 London Road, Newbury,
Berkshire RG14 1JL, England
Tel: +44 1635 811140

XCEL Power Systems Ltd
Brunswick Road, Cobbs Wood Industrial Estate
Ashford, Kent, TN 23 1EH, UK
Tel: +44 (0) 1233 656800
www.xcelpower.com

bsi. AS/EN 9100 Series Aviation, Space and Defense | ISO 9001 Quality Management | ISO 14001 Environmental Management

©Data Device Corporation

UL REGISTERED FIRM
DATA DEVICE CORPORATION
REGISTERED TO:
ISO 9001:2008, AS9100C:2009-01
EN9100:2009, JIS Q9100:2009
FILE NO. 10001296 ASH09

France: DDC Electronique
84-88 Bld de la Mission Marchand
92411 Courbevoie Cedex, France
Tel: +33-1-41-16-3424

Germany: DDC Elektronik GmbH
Triebstrasse 3, D-80993 München, Germany
Tel: +49 (0) 89-15 00 12-11

Japan: DDC Electronics K.K.
Suidobashi Sotobori-dori Bldg, 8F, 1-5,
Koraku 1-chome,
Bunkyo-ku, Tokyo 112-0004, Japan
Tel: 81-3-3814-7688
Web site: www.ddcjapan.co.jp

Asia: DDC - RO Registered in Singapore
Blk-327 Hougang Ave 5 #05-164
Singapore 530327
Tel: +65 6489 4801

India: DDC Electronics Private Limited
C-31, C/O Quest Offices Pvt. Ltd.
10th Floor, Raheja Towers
M.G Road, Bangalore 560001, India
Tel: +91 080 301 10 200

Data Device Corporation

The information in this Manual is believed to be accurate; however, no responsibility is assumed by Data Device Corporation for its use, and no license or rights are granted by implication or otherwise in connection therewith. Specifications are subject to change without notice.