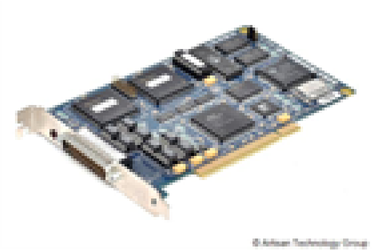


LP429-3 and LC429-3
USER'S MANUAL

PCI /cPCI to ARINC 429/717
INTERFACE BOARD AND SOFTWARE

April 5, 1999
Rev. C

Ballard Technology
3229A Pine Street
Everett, WA 98201 USA

Phone: (800) 829-1553 (425) 339-0281 Fax: (425) 339-0915
E-mail: *support@ballardtech.com*
Web: *www.ballardtech.com*

MA114-040599

TABLE OF CONTENTS

LIST OF FIGURES

LIST OF TABLES

COPYRIGHT NOTICE

LIMITED WARRANTY

Ballard Technology (the "Manufacturer") warrants its product to be free from defects in materials and workmanship under normal use for a period of one year from delivery. Should the product fail to be in good working order during the one year period the Manufacturer's sole obligation shall be at its option, to repair, correct or replace the product, or, at Manufacturer's sole discretion, to refund amounts paid for such product upon its return to a location designated by Manufacturer in the United States. Manufacturer makes no other warranties, expressed or implied, including, but not limited to, the implied warranties of merchantability and fitness for a particular purpose. Manufacturer's entire liability and Customer's exclusive remedy for damages from any cause whatsoever, including, but not limited to, nonperformance or misrepresentation, and regardless of the form of action, shall be limited to the purchase price for the specific product that caused the damages or that are the subject matter of, or directly related to, the cause of action. In no event will Manufacturer be liable for damages caused by Customer's negligence, or for any lost profits, lost savings or other incidental or consequential damages, even if Manufacturer has been advised of the possibility of such damages, or for any other claim against Customer by any other party. This limited warranty does not cover products damaged by accident, disaster, misuse, abuse or unauthorized modifications.

SAFETY WARNING

Ballard products are of commercial grade and therefore are neither designed, manufactured, nor tested to standards required for use in critical applications where a failure or deficiency of the product may lead to injury, death, or damage to property. Without prior specific approval in writing by the president of Ballard Technology, Inc., Ballard products are not authorized for use in such critical applications.

INTERFERENCE

This equipment generates, uses, and can radiate radio frequency energy and if not installed and used in accordance with the instructions manual, may cause interference to radio communications. Operation of this equipment in a residential area is likely to cause interference in which case the user at their own expense will be responsible for taking whatever measures may be required to correct the interference.

TRADEMARKS

Ballard Technology 3229A Pine Street, Everett, WA 98201, USA
Phone: (800) 829-1553 or (425) 339-0281 Fax: (425) 339-0915
E-mail: *support@ballardtech.com* Web: *www.ballardtech.com*

1. INTRODUCTION

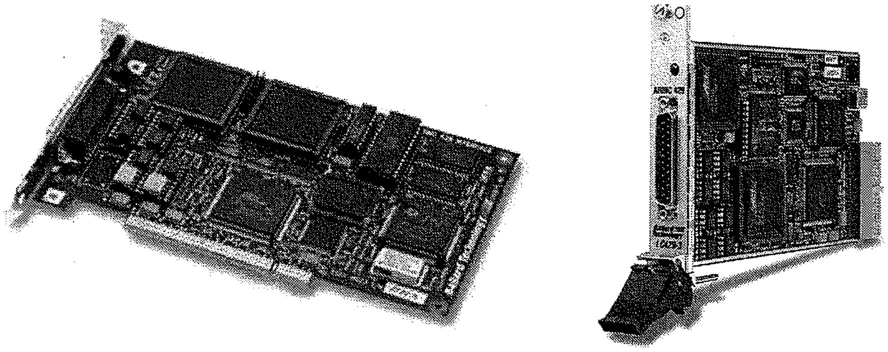
This manual covers both the LP429-3 and the LC429-3. Electrically they are nearly identical. Software written for one will work with the other. Throughout this manual together they are referred to as the Lx429-3. Wherever explicit differences apply each product is referred to individually as either the LP429-3 or the LC429-3.

1.1 ARINC 429

ARINC 429 is the specification that defines a local area network used on commercial aircraft and is the industry's standard for transfer of digital data between avionics system elements. The specification describes how an avionics system transmits information over a single twisted and shielded pair of wires (the databus) to as many as 20 receivers connected to that databus. Bi-directional data flow on a given bus is not permitted.

1.2 The LP429-3 and LC429-3

The LP429-3 is a PCI card that interfaces a computer to as many as eight ARINC 429 or ARINC 717 databuses. The LC429-3 is identical to the PCI LP429-3 except the LC429-3 is in a *CompactPCI* (cPCI) form factor. Software written for one card runs the other card without any code modifications provided they have the same channel configurations. Different versions of the Lx429-3 are available with varying numbers and combinations of ARINC 429 and ARINC 717 receive and transmit channels. For instance, an LP429-3/4R4T provides four receive and four transmit ARINC 429 channels. A version called the Lx429-3/717 is equivalent to an Lx429-3/2R2T with the addition of four ARINC 717 channels. All channels may be operated simultaneously. An on-board digital signal processor (DSP) allows the Lx429-3 to run autonomously, so it requires little or no supervision from the host PC.



a. The LP429-3 PCI card

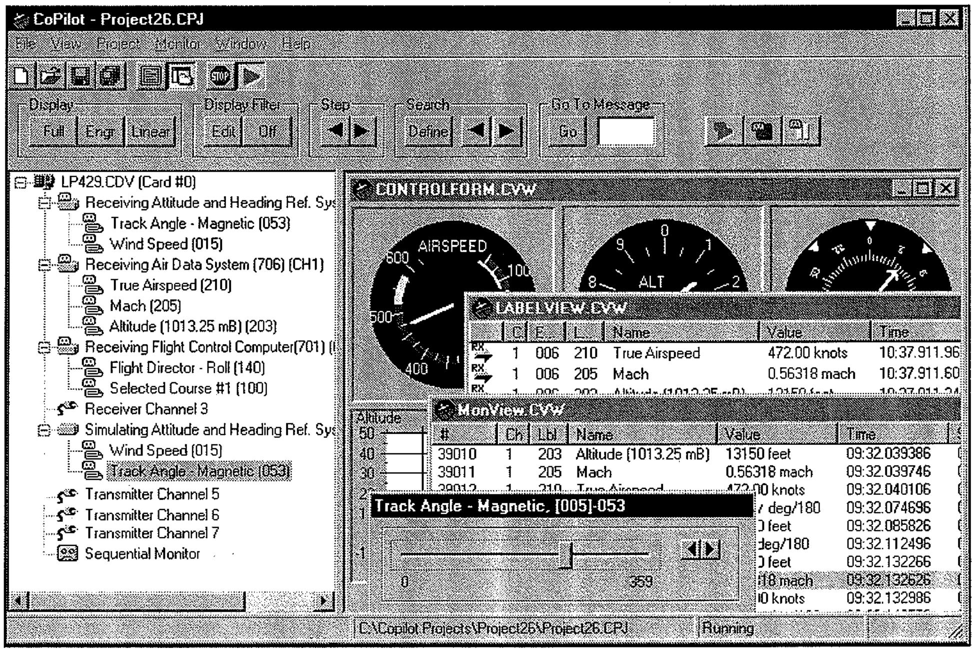
b. The LC429-3 cPCI card

Fig. 1.1 Photographs of the LP429-3 and LC429-3

Slots for expansion PCI cards are provided in most desktop personal computers (PCs). The PCI expansion bus is plug-and-play and has higher performance than the older ISA expansion bus. *CompactPCI* uses a front access card cage and offers increased shielding and electrostatic protection over the regular PCI form factor. The standard LC429-3 is a 3U-sized card. However, the LC429-3 may also be used in a 6U chassis with an optional 6U front panel adapter.

Software is used to configure the Lx429-3 and to access data. A library of driver functions is provided with the card for custom software development. This library has been designed to simplify application development for the Lx429-3 by relieving the user of most hardware details. The software also includes utility programs and sample application programs.

Available separately is a powerful, easy-to-use GUI software application called CoPilot 429. Although some specialized applications may require custom software, CoPilot 429 drastically simplifies the simulation and testing of 429 databuses. Using CoPilot 429 and your Lx429-3, you can transmit, monitor and record ARINC 429 messages with a few clicks of the mouse. You can also observe and change data in engineering units while the bus is running. The tools and filters built into the monitor mode of CoPilot 429 assist you in locating and analyzing bus activity. With CoPilot 429 there is little need to code custom software to access the capabilities of the new generation of 429 boards, which includes the Lx429-3. The best way to discover how CoPilot 429 can increase your productivity is to try it. For a free evaluation copy of CoPilot 429 please contact customer support at (800) 829-1553.



LP429.CDV (Card #0)
- Receiving Attitude and Heading Ref. Sys
  - Track Angle - Magnetic (053)
  - Wind Speed (015)
- Receiving Air Data System (706) (CH1)
  - True Airspeed (210)
  - Mach (205)
  - Altitude (1013.25 mB) (203)
- Receiving Flight Control Computer(701) (
  - Flight Director - Roll (140)
  - Selected Course #1 (100)
- Receiver Channel 3
- Simulating Attitude and Heading Ref. Sys
  - Wind Speed (015)
  - Track Angle - Magnetic (053)
- Transmitter Channel 5
- Transmitter Channel 6
- Transmitter Channel 7
- Sequential Monitor

	C	E...	L...	Name	Value	Time
RX	1	006	210	True Airspeed	472.00 knots	10:37.911.96
RX	1	006	205	Mach	0.56318 mach	10:37.911.60

#	Ch	Lbl	Name	Value	Time
39010	1	203	Altitude (1013.25 mB)	13150 feet	09:32.039386
39011	1	205	Mach	0.56318 mach	09:32.039746
				472.00 knots	09:32.040106
				deg/180	09:32.074696
				feet	09:32.085826
				deg/180	09:32.112496
				feet	09:32.132266
				18 mach	09:32.132626
				0 knots	09:32.132986

Track Angle - Magnetic, [005]-053

C:\Copilot Projects\Project26\Project26.CPJ | Running

Fig. 1.2 A sample CoPilot 429 screen

1.3 ARINC 717

The ARINC 717 specification defines the communication protocol used by the Digital Flight Data Acquisition Unit (DFDAU) and the Digital Flight Data Recorder (DRDR). The DFDAU accumulates data from the aircraft systems and transmits it to a DFDR on an ARINC 717 biphase databus. A similar bipolar databus can connect to the Quick Access Recorder (QAR). Ballard Technology's Lx429-3/717 may be used to monitor or simulate both of these ARINC 717 databuses.

Though custom versions are available, the standard Lx429-3/717 version includes two ARINC 429 receive channels, two ARINC 429 transmit channels, two ARINC 717 biphase channels (each configurable for receive or transmit), and two ARINC 717 bipolar channels (fixed at one receive and one transmit). Users of the Lx429-3/717 can interface simultaneously to both ARINC 429 and ARINC 717 databuses. Its ARINC 429 operation is identical to an Lx429-3/2R2T.

The ARINC 717 Monitor GUI (graphical user interface) is included with the Lx429-3/717. This program is a Windows 95/98/NT application developed by Ballard Technology to simplify the monitoring of the ARINC 717 databus. For more information on the Lx429-3/717, ARINC 717, or the 717 Monitor GUI please see Appendices C and D, or call (800) 829-1553.

1.4 Updates

At Ballard Technology we take pride in a high-quality, reliable product that meets the needs of its users, and improvement is a continuing process. Periodically, updates to documentation (in the form of Application Notes) may be issued as well as updates to software. Please fill out and return the Product Registration Card included in the front of this manual, so that we can keep you, the end user, informed of documentation and software updates, customer services, and new product information.

1.5 How to use this manual

If you intend to use the Lx429-3 with CoPilot 429, then you only need to read and follow the installation instructions in Chapter 2 before turning to the documentation provided with CoPilot 429.

The bulk of this manual is for users who intend to write their own software to operate the Lx429-3. After reading Chapter 2 (Installation) and Chapter 3 (Programming Basics) and referring to Appendix A (ARINC 429 Function Reference) you should be able to install the Lx429-3 and write simple computer programs to operate it. Refer to subsequent chapters for more complex applications. Operation with ARINC 717 is described in Appendices C and D.

The following conventions are observed throughout this manual:

- Driver function names are in **bold** type and are all prefixed by "`L43_`" (e.g., `L43_ChConfig`)
- A small 'h' suffix indicates hexadecimal values (e.g., F01Ch)
- Constants defined in the Lx429-3 driver software are written in all capital letters (e.g., CHCFG_DEFAULT)

1.6 Technical Support

Ballard Technology offers technical support before and after the sale. Our hours are 9:00 AM to 5:00 PM Pacific Time, though support and sales engineers are often available outside those hours. We invite your questions and comments on any of our products. You may reach us by phone at (800) 829-1553 or (425) 339-0281, by fax at (425) 339-0915, on the Web at www.ballardtech.com, or through email at *support@ballardtech.com*.

2. INSTALLATION

This chapter explains the procedures for installing your Lx429-3. There are four phases to installing the Lx429-3:

1. Insert the Lx429-3 into an empty slot in your computer
2. Install drivers for your Lx429-3
3. Test installation of the card and drivers by running L43TST32.EXE
4. Connect the Lx429-3 to your ARINC 429 and 717 databus(es)

The following system configuration is the minimum required for Lx429-3 installation:

- Microsoft Windows 95, Windows 98, or Windows NT operating system
- Personal computer with at least one free expansion bus slot
- One 3.5-inch high-density disk drive (or other media device)
- 1 MB of free hard-disk space

When you have completed the installation steps in this chapter, your Lx429-3 is ready to communicate on the 429 databus using CoPilot 429 or your software application. Chapter 3 outlines the tasks an application program performs to configure and control the Lx429-3, and Chapter 4 presents sample programs. If you plan to use CoPilot 429, please consult the documentation provided with that program. Appendices C and D discuss the ARINC 717 capability of the Lx429-3/717.

WARNING

Static Discharge

As with most electronic devices, static discharge may damage or degrade components on the board. When handling the board, the user should be grounded (e.g., through a wrist strap). The board is shipped in an anti-static bag, and should be stored in a similar container when not installed in the computer.

2.1 Card Insertion

The Lx429-3 is a Plug-and-Play device, so no jumpers or switches are used to configure it. If you are installing more than one Lx429-3 into a single computer, please refer to Section 2.4 . Install the cards as follows:

1. Shut down your computer.
2. **For LP429-3:** Insert the LP429-3 card(s) into an empty PCI slot and secure the card bracket to the case of your computer.
   For LC429-3: Insert the LC429-3 card(s) with the injector handle down into an empty peripheral slot (marked with a circle) in your CompactPCI system. When the bottom of the handle is pressed against the subrack's horizontal rail, move the injector handle up to complete the insertion of the card. Secure the LC429-3 front panel collar screw (located at the top of the front panel) to the chassis.
3. Restart your computer.

Your operating system is now prepared to have the Lx429-3 drivers installed. The installation process may begin automatically or may require manual installation as mentioned in the next section.

2.2 Driver Installation

This section explains the procedures for installing drivers into a computer running Windows 95, Windows 98 or Windows NT (4.0 and older). The Lx429-3 requires the following files from the Lx429-3 driver installation disk:

File Name	Windows 95	Windows 98	Windows NT
BTIKRNL.DLL	✓	✓	✓
BTIUNIV.SYS			✓
BTIUNIV.VXD	✓	✓	
L43W32.DLL	✓	✓	✓
L43WNT.REG			✓

Windows 95 and Windows 98 installation of the Lx429-3 card is described in Sections 2.2.1 and 2.2.2. Windows NT driver installation of the Lx429-3 card is described in Section 2.2.3. Refer to Section 2.4 after completing the steps in this section for information on installing multiple cards.

Note: If you are not installing from a floppy disk, refer to the readme.txt file for driver file locations.

2.2.1 Windows 95 and Windows 98 Automatic Driver Installation

Since the Lx429-3 is a Plug-and-Play device, the first time Windows 95 or Windows 98 is started with your Lx429-3 installed, the Lx429-3 card is detected and the driver installation process should begin automatically. If you are not prompted to install drivers for the Lx429-3 before Windows finishes loading and the drivers are not yet installed, skip the rest of this section and go to Section 2.2.2. When Windows detects the Lx429-3 and prompts you to install the drivers, follow the appropriate steps for either Windows 95 or Windows 98. When driver installation is complete, skip to Section 2.3.

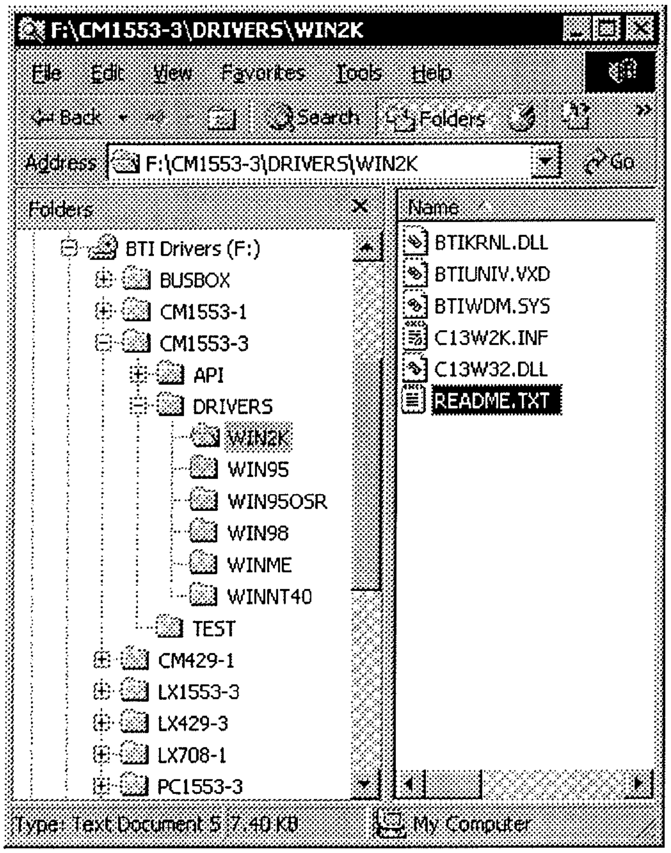
DRIVER INSTALLATION SUPPLEMENT

Drivers and driver installation instructions for Ballard Technology, Inc. hardware products are now contained on a single distribution CD. These are kept on disk so they can be easily updated as operating systems evolve. This Driver Installation Supplement supersedes all other driver installation instructions in Ballard publications.

To install drivers for your Ballard board, you must find, print, and follow the instructions on the software distribution disk. The instructions vary depending on the type of board and your computer's operating system. Find and print the installation procedure before installing your board.

The driver installation instructions are in a README.TXT file on the distribution disk in a folder specific to your board and operating system. Follow these steps to locate and print the instructions:

1. Insert the disk in your drive and browse to the folder for your Ballard board (e.g., CM1553-3).
2. Open the DRIVERS subfolder under the board folder (e.g., CM1553-3→DRIVERS).
3. Open the subfolder for your operating system (e.g., CM1553-3→DRIVERS→WIN2K).
4. Print the README.TXT file in the operating system subfolder.



F:\CM1553-3\DRIVERS\WIN2K

Address: F:\CM1553-3\DRIVERS\WIN2K

```
BTI Drivers (F:)
  BUSBOX
  CM1553-1
  CM1553-3
    API
    DRIVERS
      WIN2K
      WIN95
      WIN95OSR
      WIN98
      WINME
      WINNT40
    TEST
  CM429-1
  LX1553-3
  LX429-3
  LX708-1
  PC1553-3
```

```
BTIKRNL.DLL
BTIUNIV.VXD
BTIWDM.SYS
C13W2K.INF
C13W32.DLL
README.TXT
```

Type: Text Document 5 7.40 KB   My Computer

To install the driver software, follow the instructions printed from the README.TXT file. The installation procedure copies several files into the host computer system and modifies the system registry. If you encounter problems, have installation questions, or cannot find a folder for your operating system, contact Ballard Technology customer support at (800) 829-1553.

Ballard Technology
3229A Pine Street, Everett, WA 98201
1-800-829-1553 or 425-339-0281
e-mail: *support@ballardtech.com*
web: *www.ballardtech.com*

MA139-031402
March 14, 2002

Windows 95

1. Select the **Driver Provided by Manufacturer** option.
2. Click **Next**.
3. Insert the Lx429-3 installation disk into drive A.
4. Type the path containing the driver files for the Lx429-3 (**A:\WIN95** for floppy disk installation) in the **Copy manufacturer's files from** field.
5. Click **OK** to complete the installation process.

Windows 98 (or Windows 95 OSR2)

1. Click **Next** when the **Add New Hardware Wizard** starts.
2. Select the **Display a list of all the drivers in a specific location** option and click **Next**.
3. When prompted to select the device type, ignore this step and press **Next**.
4. Click **Have Disk** and insert the media containing the Lx429-3 installation files (if the data is not located on the hard disk).
5. Type the path containing the driver files for the Lx429-3 (**A:\WIN95** for floppy disk installation) and press **OK**.
6. Select the Lx429-3 and press **Next**.
7. Press **Next** again to search the specified location for the driver files.
8. Click **Finish** to complete the installation process.
9. Skip to Section 2.3 to test installation.

2.2.2 Windows 95 and Windows 98 Manual Driver Installation

Manual installation of drivers is only required if the Windows 95 or 98 installation procedure did not automatically begin, the installation was unsuccessful, or drivers need reinstallation. Perform the following steps to manually install the drivers:

1. Click the **Start** button, point to **Settings** and click **Control Panel**.
2. Double-click the **Add New Hardware** icon.
3. When the first **Add New Hardware Wizard** window opens, click **Next**.
4. For Windows 98 installations only, click **Next** to automatically search for Plug and Play hardware.
5. Select the **No** option so Windows does not search for new hardware and click **Next**.
6. Select **Other devices** from the **Hardware types** list and click **Next**.
7. Click **Have Disk** and insert the media containing the Lx429-3 installation files (if the data is not located on the hard disk).
8. Type the path containing the driver files for the Lx429-3 (**A:\WIN95** for floppy disk installation) in the **Copy manufacturer's files from** field of the **Install From Disk** window.
9. Click **OK** to return to the **Add New Hardware Wizard**.
10. Select whichever applies, either **LP429-3 (ARINC 429 Interface Card)** or **LC429-3 (cPCI ARINC 429 Interface Card),** from the **Models** list box and click **Next**.
11. Click **Next** to install drivers.
12. Click **Finish** to complete driver installation.
13. Skip to Section 2.3 to test installation.

2.2.3 Windows NT Driver Installation

To install drivers for Windows NT, you will need the Ballard Technology "L43 Windows NT Driver" disk. Perform the following steps to manually install the drivers for your Windows NT operating system:

1. Install the required registry keys by opening the WINNT folder on the driver disk containing the L43WNT.REG file and double click on it. Windows will update the required registry keys.
2. Copy the BTIKRNL.DLL and L43W32.DLL files from the WINNT folder on the driver disk into the C:\WINNT\SYSTEM32 folder.
3. Copy the BTIUNIV.SYS file from the WINNT folder on the driver disk into the C:\WINNT\SYSTEM32\DRIVERS folder.
4. Restart the computer.

After completing the driver installation, the card may be tested by running the program L43TST32.EXE located on the driver disk. A complete description of testing the Lx429-3 is described in Section 2.3.

2.3 Testing the Installation

You can test the installation and functionality of your Lx429-3 by double-clicking the L43TST32.EXE program provided on the installation disk. This program completes execution within several seconds. If the program does not detect any faults with the interface or the Lx429-3 hardware, it displays the Card Number, Slot Number and a "passed" test message. The Lx429-3 is now ready to be connected to the ARINC 429 and ARINC 717 databuses as described in Section 2.5.

If the L43TST32 test detects a fault, it displays relevant fault information. When the fault message displays, follow the instructions on screen.

Other useful information such as the board model, channel configuration, memory size and firmware version can be determined through the L43TST32 utility. Run it with the ? switch (i.e., L43TST32 ?) to view your options. If you need further assistance, call Ballard Technology at (800) 829-1553. A customer support engineer will interpret the fault and guide you through corrective steps.

2.4 Multiple Card Installation

The installation of multiple Lx429-3 cards into your computer requires a few more steps than a single card installation. These extra steps are needed to differentiate between the installed cards by determining their Slot and Card Numbers.

"Slot Numbers" are logical numbers assigned by the operating system. They are constant for a given system and identify each physical PCI (cPCI) slot. Note, the logical Slot Number is usually not equal to the physical PCI (cPCI) motherboard slot. Card Numbers are used by application software to uniquely identify each Lx429-3. The Card Numbers may change depending on the position and number of cards used. Card Numbers usually start at zero on the power supply connector side of the motherboard and increment by one for each successively installed Lx429-3 card. **If only one Lx429-3 is used, it is Card Number 0**.

The association of the Card Number to each physical card depends on the Slot Number occupied by a particular card. Changing slots or removing cards when multiple cards are installed may change the Card Number assigned to a particular Lx429-3 card. As long as the number of cards is constant and the cards are not moved, the associated Card Numbers remain constant for all cards.

The following procedure is used to determine Slot Numbers of the computer and relate these Slot Numbers to the Card Number of each Lx429-3. Table 2.1 may be useful when installing multiple Lx429-3's. Make a copy and fill in the information as needed.

1. Insert all cards in the computer as described in Section 2.1.
2. Install the drivers as described in Section 2.2.
3. Run the L43TST32 test utility with the –LED option:
   (i.e., A:\L43TST32 –LED).
4. Follow the on-screen instructions. L43TST32 will sequentially blink the LEDs to identify each Lx429-3. Record the Card Number and Slot Number displayed on screen. To help ensure proper wiring, you may want to externally mark each card with its Card Number.

Physical Slot # (may be marked on chassis)	**1**	**2**	**3**	**4**	**5**	**6**	**7**	**8**
System Slot #								
Card #								
Lx429-3 config. (4R4T, 2R6T, etc.)								

Table 2.1 Multiple Lx429-3 card installation

2.5 Connecting to the ARINC 429 and 717 Databuses

The ARINC 429 and 717 databuses are connected through the male 25-pin D-subminiature connector on the Lx429-3 bracket. Your cable should have a mating 25-pin female D-subminiature connector. The pin assignments depend on the channel configuration of the Lx429-3.

Using the suffix /xRyT, the LP429-3 and LC429-3 may be ordered with different configurations, where x and y are the number of receive and transmit channels respectively. Only the Lx429-3/717 has ARINC 717 capability; it is equivalent to an Lx429-3/2R2T with 717 added. Common channel configurations are shown in Table 2.2 with the more popular ones printed in bold. If you do not know which model Lx429-3 is installed in a computer, you can use the L43TST32 program described in Section 2.3 above to display the channel configuration.

Channel→ Version↓	CH0	CH1	CH2	CH3	CH4	CH5	CH6	CH7
1R1T	R	-	T	-	-	-	-	-
2R2T	**R**	**R**	**T**	**T**	**-**	**-**	**-**	**-**
717	**R**	**R**	**T**	**T**	**R#**	**T#**	**R/T~**	**R/T~**
4R4T	**R**	**R**	**R**	**R**	**T**	**T**	**T**	**T**
4R0T	R	R	R	R	-	-	-	-
0R4T	T	T	T	T	-	-	-	-
2R6T	R	R	T	T	T	T	T	T
6R2T	R	R	R	R	R	R	T	T
8R0T	R	R	R	R	R	R	R	R
0R8T	T	T	T	T	T	T	T	T

The most popular configurations are in **bold.** R = Receive, T = Transmit, R/T = Receive or Transmit,
= Bipolar 717, ~ = Biphase 717 (output on 717A/B), otherwise 429

Table 2.2 Channel configurations for the Lx429-3

Physical Channel	Designation	Pins

0
(429 only)
	CH0(P)
	CH0(N)
		3
		15

1
(429 only)
	CH1(P)
	CH1(N)
		2
		14

2
(429 only)
	CH2(P)
	CH2(N)
		4
		16

3
(429 only)
	CH3(P)
	CH3(N)
		5
		17

4
(429 or 717 Bipolar
	CH4(P)
	CH4(N)
		6
		18

5
(429 or 717 Bipolar)
	CH5(P)
	CH5(N)
		8
		19

6
(429 only)
	CH6(P)
	CH6(N)
		9
		20

7
(429 only)
	CH7(P)
	CH7(N)
		10
		21

717A
(717 Biphase CH6)
	717A(P)
	717A(N)
		11
		23

717B
(717 Biphase CH7)
	717B(P)
	717B(N)
		12
		13

	GND GND EXTRIG* SYNCOUT PIN25	1 7 22 24 25

Table 2.3 Pin assignments on the Lx429-3 connector

Pin assignments are shown in Table 2.3. Note that the channel designations (e.g., CH0) used in Tables 2.2 and 2.3 are also the names of the constants defined in the driver library. These constants are used to refer to the physical channels in software.

After completing the above steps, your Lx429-3 should be ready to communicate on the 429 databuses under the control of an application program such as CoPilot 429. If you are planning to write your own programs, you should proceed to Chapter 3 which outlines the tasks an application program performs to configure and control the Lx429-3. You can also run the example programs included on the Lx429-3 installation disk, but you may need to modify them for the channel configuration of your Lx429-3.

This page intentionally blank.

3. PROGRAMMING BASICS

This chapter illustrates basic ARINC 429 operation of the Lx429-3 using the driver library. Users programming custom applications for ARINC 717 should refer to Appendices C and D. Examples demonstrate:

- operation of ARINC 429 receive channels
- operation of ARINC 429 transmit channels
- monitoring and selectively recording ARINC 429 bus traffic

The examples provide code fragments in the C language. Libraries for other standard languages are available on request from Ballard Technology. All driver functions are prefixed by "**L43_**" and are printed in bold for clarity. The examples are given with the assumption that CH0 is a receive channel and CH4 is a transmit channel, as is the case with an Lx429-3/4R4T.

This chapter describes the operational elements relevant to most applications. After reading this chapter you will be familiar with the essentials of an application program for the Lx429-3, and you will recognize the most important driver functions. Complete descriptions of all driver functions may be found in Appendix A.

3.1 Terminology

The basic unit of information defined by the ARINC 429 Specification is a 32-bit word made up of several bit fields. The most important bit fields are the label (8-bits), SDI (2-bits), and data (variable length). Given the source of the data, the label determines how the data field is interpreted, including:

- data format (binary, BCD, ASCII character, etc.)
- data type (temperature, pressure, etc.)
- units (Newtons, degrees Celsius, etc.)

The SDI (Source/Destination Identifier) identifies the source of a word when more than one source exists on the aircraft (e.g. left engine temperature versus right engine temperature). Often, however, the SDI bits are ignored. **In this manual we use the term *message* to refer to the 32-bit ARINC word associated with a *label and SDI combination*.**

3.2 Getting started

The first step in developing an application for the Lx429-3 is identifying the ARINC 429 messages to be handled. For the examples in this chapter, suppose we want to simultaneously:

- receive selected messages from a Global Positioning System (GPS)
- simulate transmission of messages from an Air Data System (ADS)
- record all above activity to disk for later analysis

Specifically, assume that among all messages that GPS and ADS units can transmit, we are interested only in those shown in Table 3.1.

Equipment	ARINC 429 Word Name	Octal Label
GPS	Present Pos - Latitude	310
GPS	Present Pos - Longitude	311
ADS	Computed Airspeed	206
ADS	Total Air Temperature	211
ADS	Altitude Rate	212

Table 3.1 Example messages

Part of the configuration process is associating messages with Message Records. Each Message Record contains a single 32-bit ARINC 429 word, and possibly some extra data related to that message (e.g., the time-tag). When the Lx429-3 receives a given message, the full 32-bit word is stored in a designated Message Record. When the Lx429-3 transmits a given message, the word to be transmitted is retrieved from a predetermined Message Record. More detailed information on Message Records may be found in Section 4.2.

3.3 Steps a program must perform

Before continuing these examples, we present an overview of the sequence of steps a program performs to operate the Lx429-3.

Operating the Lx429-3 with the driver library involves seven general steps, four of which require only a single function call. Three of the steps involve many options, so the number of function calls required depends on which options are desired. The steps are as follows:

1. Open: Software gains access to Lx429-3 hardware through the host computer's operating system. The `L43_CardOpen` function requests access and returns a "handle" by which all subsequent functions refer to the card. An input parameter to `L43_CardOpen` is the Card Number discussed in Section 2.4.

2. Configure: The required Lx429-3 channels and capabilities are enabled. A Schedule is created for transmitter operation. Filter Tables are configured for receiver operation. Special features of the Lx429-3 may require extra configuration. Other low-level options such as interrupt generation and bus speed are also set at this point.

3. Initialize data: The transmit channel begins transmitting messages from Lx429-3 memory immediately after activation. Data associated with these messages should be initialized to prevent transmission of invalid messages.

4. Activate: `L43_CardStart` activates all configured channels simultaneously. Once activated, the Lx429-3 transmits and/or receives from the databuses independently of the host computer.

5. Handle data: The Lx429-3 transmits and receives messages according to its configuration without requiring any host supervision. A program running on the host can update data for transmission or read received data at any time. To reduce data latency and/or host overhead, applications may access data in response to bus events detected by polling or interrupts. Driver functions are provided to simplify data exchange between the host and Lx429-3. Additionally, a library of utility functions is provided to insert and extract the bit fields of an ARINC 429 word. These functions may be used independently of the Lx429-3.

6. Deactivate: **`L43_CardStop`** deactivates the card. Unless the Lx429-3 is explicitly deactivated, it continues operating even if the application software halts.

7. Close: Whether or not the Lx429-3 is deactivated, **`L43_CardClose`** must be called before an application terminates. Failure to do so can cause unpredictable results. **`L43_CardClose`** does not deactivate the Lx429-3.

The following examples show how easy it is to perform these steps using the driver functions.

3.4 Receiver example

In this section we describe a Lx429-3 example program that receives Latitude and Longitude messages (Table 3.1).

The primary task in configuring receive channels is telling the Lx429-3 in which Message Records to store specific messages. The Lx429-3 ignores messages not assigned a Message Record unless a Default Record has been defined. If a Default Record is defined, all messages not assigned their own Message Records are written to the Default Record. The Default Record makes it possible to receive all bus traffic while isolating the messages of interest.

The code in Fig. 3.1 configures the Lx429-3 to receive all traffic. The two GPS messages of interest are stored in separate Message Records. The Lx429-3 writes all other messages to the Default Record.

```
MSGSTRUCT msgdefault, latitude, longitude;

handleval = L43_CardOpen(cardnum);                      //Open resources

L43_ChConfig(CHCFG_DEFAULT,CH0,handleval);              //Configure channel
                                                        //Rcv Filters
msgdefault.addr = L43_FilterDefault(MSGCRT_DEFAULT,CH0,handleval);
latitude.addr = L43_FilterSet(MSGCRT_DEFAULT,0310,SDIALL,CH0,handleval);
longitude.addr = L43_FilterSet(MSGCRT_DEFAULT,0311,SDIALL,CH0,handleval);

L43_CardStart(handleval);                               //Start the card

while (!done)       //Read data as required by application
{
   msgdefault.data = L43_MsgDataRd(msgdefault.addr,handleval);
   latitude.data = L43_MsgDataRd(latitude.addr,handleval);
   longitude.data = L43_MsgDataRd(longitude.addr,handleval);
}

L43_CardStop(handleval);                                //Stop the card
L43_CardClose(handleva);                                //Close resources
```

Fig. 3.1 Example receiver program

The code starts by creating message structures named msgdefault, latitude and longitude. These structures have members for the message address and the value of the data. **L43_CardOpen** is always the first driver function called since it returns the handle needed by subsequent functions. An input parameter to **L43_CardOpen** is the Card Number discussed in Section 2.4.

L43_ChConfig sets up an empty filter table for a receive channel and enables or disables selected options for that channel. This function sets the bus speed to low and disables all options by default. Constants can be included to turn on specific options. Note that this function does not activate the channel. CH0 is a pre-defined constant referring to physical channel 0 of the Lx429-3. (See Tables 2.2 and 2.3).

A Filter Table is an array of pointers to Message Records. There is one pointer for every possible message type (i.e., every possible Label/SDI combination). All pointers are initially zero. A received message is recorded if and only if its entry in the Filter Table points to a valid Message Record. The **L43_FilterDefault** function fills the Filter Table such that all messages received on that channel are written to the Default Record. If **L43_FilterDefault** is used, it must precede any calls to **L43_FilterSet**. The two **L43_FilterSet** functions assign receive messages to Message Records by placing entries in the Filter Table. The **L43_FilterDefault** and **L43_FilterSet** functions return the address of the Message Record.

The SDIALL constant tells the Lx429-3 to accept all messages with the given label and any SDI. Different constants may be used to specify that only messages with specific SDIs are to be received. Messages with different SDIs could be assigned to separate Message Records.

The Lx429-3 begins receiving messages only after **L43_CardStart** is called. As discussed above, when a message is received, it is stored in its assigned Message Record. The previous value in that Message Record is over-written. **L43_MsgDataRd** may be called at any time to return the full 32-bit ARINC word from a specified Message Record. A library of functions is provided to extract the relevant bit fields from the ARINC word.

The program ends with the required **L43_CardStop** and **L43_CardClose** functions. Receiving messages on other receive channel(s) only requires extra **L43_ChConfig**, **L43_FilterDefault** and **L43_FilterSet** function calls.

3.5 Transmitter example

We now develop a separate program to transmit the Airspeed, Temperature and Altitude Rate messages.

Normally, ARINC 429 sources transmit messages periodically at repetition rates prescribed by the ARINC 429 specification. The specification defines a minimum and maximum *transmit interval* for each message. Many different transmit intervals are called out in the specification, so transmitting many different messages may involve complex timing requirements. The Lx429-3 driver functions handle timing requirements automatically.

To illustrate, recall that our transmission example is simulating an Air Data System transmitting three messages. Their transmit intervals are shown in Table 3.2. The code in Fig. 3.2 configures the Lx429-3 to transmit these messages with proper timing.

Word	Octal Label	Min transmit interval (ms)	Max transmit interval (ms)
Computed Airspeed	206	62.5	125
Total Air Temperature	211	250	500
Altitude Rate	212	31.3	62.5

Table 3.2 Transmit intervals

```
MSGSTRUCT comp_airspeed, tot_air_temp, altitude_rate;
MSGADDR msgaddr[3];
INT     min_intrvls[3];
INT     max_intrvls[3];

handleval = L43_CardOpen(cardnum);                      //Open resources

L43_ChConfig(CHCFG_DEFAULT,CH4,handleval);              //Configure channel

//Create 3 messages
comp_airspeed.addr = L43_MsgCreate(MSGCRT_DEFAULT,handleval);
tot_air_temp.addr = L43_MsgCreate(MSGCRT_DEFAULT,handleval);
altitude_rate.addr = L43_MsgCreate(MSGCRT_DEFAULT,handleval);

//Set up arrays needed to build schedule
msgaddr[0] = comp_airspeed.addr; msgaddr[1] = tot_air_temp.addr;
msgaddr[2] = altitude_rate.addr;
min_intrvl[0] = 63; min_intrvl[1] = 250; min_intrvl[2] = 32; //Tx intervals
max_intrvl[0] = 125; max_intrvl[1] = 500; max_intrvl[2] = 62;
//Build schedule
L43_SchedBuild(3,msgaddr,min_intrvl,max_intrvl,CH4,handleval);

L43_MsgDataWr(0206,comp_airspeed.addr,handleval);       //Initialize data
L43_MsgDataWr(0211,tot_air_temp.addr,handleval);
L43_MsgDataWr(0212,altitude_rate.addr,handleval);

L43_CardStart(handleval);                               //Start the card

while (!done)
{             //Update data as required by the application
    L43_MsgDataWr(comp_airspeed.data,comp_airspeed.addr,handleval);
    L43_MsgDataWr(tot_air_temp.data,tot_air_temp.addr,handleval);
    L43_MsgDataWr(altitude_rate.data,altitude_rate.addr,handleval);
}

L43_CardStop(handleval);                                  //Stop the card
L43_CardClose(handleval);                                 //Close resources
```

Fig. 3.2 Example transmitter program

This example starts out similar to the previous one. We first define message structures for the messages of interest. Three arrays to be used for building a schedule are also created. As always, the `L43_CardOpen` function must be called first to obtain the handle of the card. `L43_ChConfig` automatically determines that CH4 is a transmit channel and configures it accordingly.

The easiest way to create a schedule that transmits the three messages at their proper transmit intervals is to use the **L43_SchedBuild** function. To do this we first create the three messages using **L43_MsgCreate** which returns their addresses. **L43_MsgCreate** in transmit is equivalent to **L43_FilterSet** in receive. Next, we create three arrays: one for the message addresses, one for minimum transmit intervals, and one for maximum transmit intervals. Information on a given message is contained in the same position in each of these arrays. The minimum and maximum transmit intervals (rounded to integer values in milliseconds) are defined as specified in Table 3.2.

The **L43_SchedBuild** function constructs a *Schedule* in Lx429-3 memory. A Schedule is a sequence of messages separated by timed gaps. The gaps are calculated so that the timing requirements of all messages are satisfied. The parameters in the example indicate that **L43_SchedBuild** is to schedule transmission of three messages on CH4 from the information in the three arrays. An alternative to using **L43_SchedBuild** is to explicitly define your own schedule as described in Section 4.4.

The Message Records are initialized using **L43_MsgDataWr** before **L43_CardStart** commands the Lx429-3 to begin transmitting. Here we initialized all three messages to zero except for their labels. Notice that the label is the least significant part of the the data and that it is entered in octal. See Appendix A for more information regarding the order of bits in an ARINC word. Also using **L43_MsgDataWr,** we can update the data value of a message at any time .

The three messages are repeatedly transmitted at the proper rates until the Lx429-3 is halted by **L43_CardStop**. As always, the program ends with **L43_CardClose**.

3.6 Monitor example

In a sense, any receiver is a monitor that holds the most recent value of the received data. However, the Lx429-3 has another type of monitor called a Sequential Record, which records a time tagged history of user selected 429 bus activity. A Sequential Record is useful for analyzing and reconstructing the bus activity. It may be configured to capture all or selective bus activity. Additional information on the Sequential Record may be found in Section 4.5.

The code in Fig. 3.3 combines the previous receive and transmit examples while configuring the Sequential Record to capture only the five message types defined in the examples. Specifically, the Sequential Record will capture only the latitude and longitude messages from the receive channel and all words transmitted by the Lx429-3.

```
MSGSTRUCT msgdefault, latitude, longitude;
MSGSTRUCT comp_airspeed, tot_air_temp, altitude_rate;
MSGADDR msgaddr[3];
INT     min_intrvls[3];
INT     max_intrvls[3];
SEQRECORD seqbuf;

handleval = L43_CardOpen(cardnum);                    //Open resources

L43_ChConfig(CHCFG_DEFAULT,CH0,handleval);            //Configure RCV Chan
L43_ChConfig(CHCFG_SEQALL,CH4,handleval);             //Configure XMT Chan

//Create 3 transmit messages
comp_airspeed.addr = L43_MsgCreate(MSGCRT_DEFAULT,handleval);
tot_air_temp.addr = L43_MsgCreate(MSGCRT_DEFAULT,handleval);
altitude_rate.addr = L43_MsgCreate(MSGCRT_DEFAULT,handleval);

//Set up arrays needed to build schedule
msgaddr[0] = comp_airspeed.addr; msgaddr[1] = tot_air_temp.addr;
msgaddr[2] = altitude_rate.addr;
min_intrvl[0] = 63; min_intrvl[1] = 250; min_intrvl[2] = 32; //Tx intervals
max_intrvl[0] = 125; max_intrvl[1] = 500; max_intrvl[2] = 62;

//Build schedule
L43_SchedBuild(3,msgaddr,min_intrvl,max_intrvl,CH4,handleval);

L43_MsgDataWr(0206,comp_airspeed.addr,handleval);      //Initialize data
L43_MsgDataWr(0211,tot_air_temp.addr,handleval);
L43_MsgDataWr(0212,altitude_rate.addr,handleval);

//Define filters
msgdefault.addr = L43_FilterDefault(MSGCRT_DEFAULT,CH0,handleval);
latitude.addr = L43_FilterSet(MSGCRT_SEQ,0310,SDIALL,CH0,handleval);
longitude.addr = L43_FilterSet(MSGCRT_SEQ,0311,SDIALL,CH0,handleval);

//Configure monitor
L43_SeqConfig(SEQCFG_32K | SEQCFG_CONTINUOUS,handleval);

L43_CardStart(handleval);                              //Start the card

while (!done)
{        //Depending upon the application
   if (L43_SeqRd(&seqbuf,handleval))         //Read monitor
   {
       //Write to disk, display data, etc
   }
   //Also read and write message data as required
}

L43_CardStop(handleval);                               //Stop the card
L43_CardClose(handleval);                              //Close resources
```

Fig. 3.3 Example monitor program

Filtering irrelevant messages out of the bus traffic conserves Sequential Record memory and makes the Sequential Record easier to analyze. Filtering for the Sequential Record can be established either at the channel level or the message level. If the CHCFG_SEQALL constant is used in **`L43_ChConfig`**, then all messages on that channel are saved in the Sequential Record. If only specific messages are to be saved then the MSGCRT_SEQ constant is used either in **`L43_FilterSet`** for receive or **`L43_MsgCreate`** for transmit. These are summarized in Table 3.3.

Use these functions & parameters → **to record ↓**	**L43_ChConfig**	**Receive** **L43_Filter Set**	**Transmit** **L43_MsgCreate**
from all messages on selected channels	CHCFG_SEQALL	don't care	don't care
from selected messages on selected channels	CHCFG_SEQSEL	MSGCRT_SEQ	MSGCRT_SEQ

Table 3.3 Sequential Record filtering options

Much of the code in Fig. 3.3 is a combination of modified fragments from the previous examples. In addition to the others, a structure (seqbuf) is created of type SEQRECORD with members matching the record fields in a Sequential Record. Since we want to monitor all transmitted messages the CHCFG_SEQALL constant is used in place of the CHCFG_DEFAULT constant in `L43_ChConfig` for CH4. To save the Latitude and Longitude messages in the Sequential Monitor the MSGCRT_SEQ constant is used in their respective `L43_FilterSet` functions. `L43_SeqConfig` sets up the Sequential Record with a 32KB circular (continuous) buffer. Notice how the two constants, SEQCFG_32K and SEQCFG_CONTINUOUS are OR-ed together. `L43_Start` activates all configured channels on the Lx429-3. After activation, the Lx429-3 begins simultaneously receiving, transmitting, and recording the specified traffic in its Sequential Record.

The while() loop in the code polls the Sequential Record. If new messages have been received, `L43_SeqRd` returns a non-zero value, places the oldest message in the seqbuf structure, and automatically increments an internal pointer. `L43_SeqRd` returns zero if no new messages have been received. If the application requires it, the received and transmitted messages may be read/written as in the previous examples. As in previous examples, the program ends with `L43_CardStop` and `L43_CardClose`.

The preceding examples illustrated the most important driver functions. Your programs for the Lx429-3 can be modeled after the code in these examples. Complete source code for these examples may be found on the distribution disk. Detailed function descriptions are found in Appendix A. More information may be contained in the README.TXT files on the distribution disks.

4. ADVANCED OPERATION

The purpose of this chapter is to provide you with a deeper understanding of how the Lx429-3 works internally. Such insight will help you use the driver functions for more advanced applications. This chapter describes how the capabilities of the Lx429-3 are implemented in its internal data structures. Some operational details omitted from the previous chapter are also included. Note: This chapter applies only to ARINC 429 operation; see Appendices C and D for ARINC 717 operation.

4.1 Overview

A Digital Signal Processor (DSP) is the heart of the Lx429-3. Resident firmware executed by the DSP implements many of the Lx429-3 capabilities. The speed of the DSP allows simultaneous operation of all eight channels. Gate arrays manage the interface to the host and arbitrate access to the on-board memory and provide ARINC 429 encoding and decoding. The on-board memory contains all configuration data structures described in the following sections. Part of the memory is a true dual port memory that may be used in situations where intensive host access would otherwise bog down internal processing.

All exchanges between the host and ARINC 429 databuses are buffered by various data structures in Lx429-3 memory. The host writes messages for transmission to designated data structures, and the Lx429-3 transmits them according to its configuration. Similarly, the Lx429-3 receives messages from the 429 databuses and stores them in its memory. The host can then read the received messages from designated locations. User software accesses these structures through calls to driver functions.

Figure 4.1 provides an overview of the flow of messages between the primary data structures in the Lx429-3. These data structures and their interactions are explained in detail in following sections. It will be helpful to refer back to this diagram.

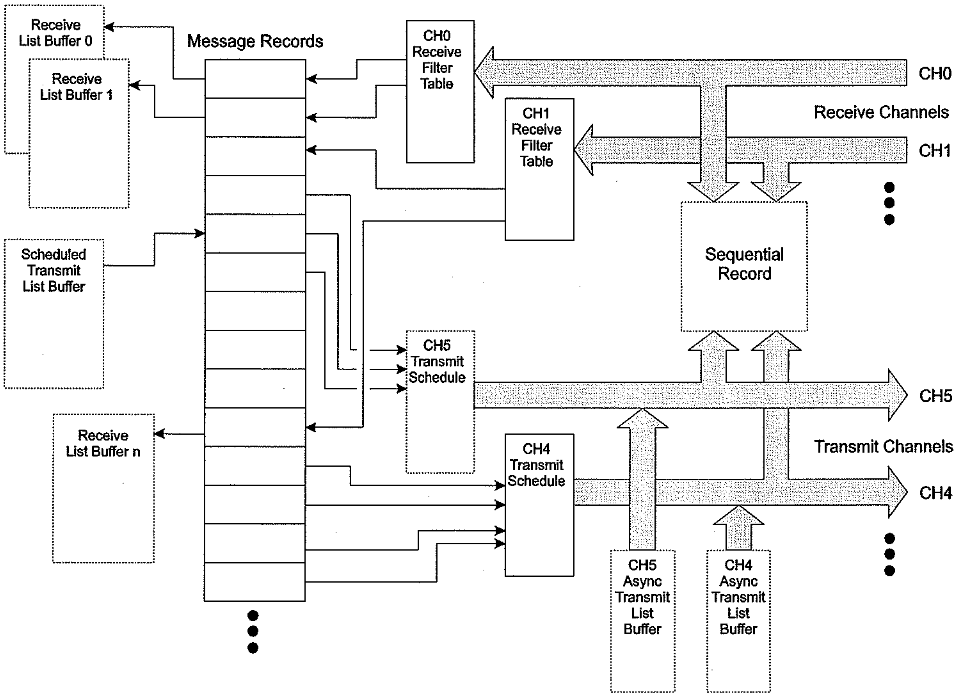


Fig. 4.1 Message flow between primary data structures in a typical Lx429-3

4.2 Message Records

Message Records are used by both receive and transmit channels to buffer incoming and outgoing messages. (See Chapter 3.) A Message Record structure (Figure 4.2) includes space for additional information that may be used if the corresponding options are enabled when the card is configured. By default, only the Reserved, Activity, and Message fields are used. The Lx429-3 updates the fields continuously while it is activated (i.e., between `L43_CardStart` and `L43_CardStop`). The following describes these data fields. Note that some options are mutually exclusive.

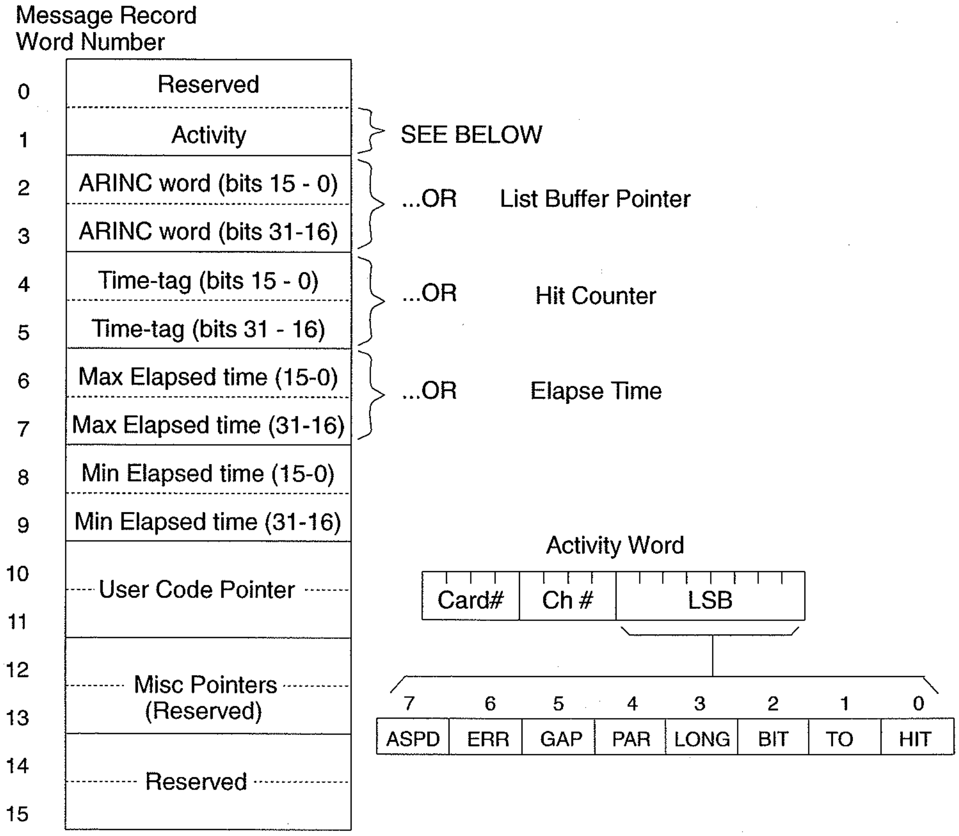


Fig. 4.2 Message Record structure

4.2.1 Reserved

The Reserved words may be used by internal processes and are not intended for end users.

4.2.2 Activity

The Activity word provides a variety of useful information about the message. The following describes the fields within the Activity word:

Card #: The card number is a four-bit field assigned by the user. It is intended to identify the source of data when, for example, data from several cards is being streamed to a single disk file from the Sequential Record. See Section 4.5. The value of Card # is assigned using `L43_BrdNumWr`.

Ch #: The channel number is a four-bit field identifying the source or destination channel of a message according to Lx429-3 channel numbering. The DSP senses the channel and automatically fills in this value, which is in the range of 0 to 7 as defined by Table 2.3.

ASPD: This bit indicates the speed of a received word when the decoder is configured for automatic speed detection. A zero indicates low speed (12.5Kbps) and a one indicates high speed (100Kbps). This bit only has meaning for received words.

ERR: This bit indicates an error was detected in word, and is the logical OR of EGAP, EPAR, LONG, EBIT, and TO. This bit only has meaning for received words.

EGAP: Indicates that the word was not preceded by a gap of at least four bit times. This bit only has meaning for received words.

EPAR: Indicates that the word was received with a parity error. This bit only has meaning for received words.

LONG: Indicates that the word was received with more than 32 bits, of which only 32 bits are represented in the Message field. This bit only has meaning for received words.

EBIT: Indicates that the word was received with some kind of timing error in at least one bit. This bit only has meaning for received words.

TO: Indicates that a time-out occurred, probably caused by a short word (less than 32 bits) or a noise burst that looked like the start of a word. This bit only has meaning for received words.

HIT: Indicates that the Message Record has been processed either by the transmission or reception of a message, so this bit is normally set when the Lx429-3 is operating. The **`L43_MsgIsAccessed`** function returns the value of the HIT bit and then clears it. When the HIT bit is set, the user knows the message has been processed at least once since the previous call to **`L43_MsgIsAccessed`**. This bit has meaning for both received and transmitted words.

4.2.3 ARINC Word

The ARINC Word is the 32 bit data value of the ARINC 429 message. Bits seven through zero contain the label. See Appendix A for a complete interpretation of an ARINC 429 message.

4.2.4 List Buffer Pointer

When a List Buffer is associated with a Message Record, the ARINC Word field is replaced with a pointer to the List Buffer. See Section 4.6 for more information on List Buffers.

4.2.5 Time-tag

The time-tag is a 32-bit value derived from an internal clock. The resolution (and resulting range) of time-tag values may be adjusted with the **`L43_TimerResolution`** function. This function does not affect the internal counter. It only determines which bits are extracted from the 48-bit counter to form the 32-bit time-tag. The selected resolution applies to all channels.

Note that the time-tag of a transmitted word represents when the word is loaded into the encoder, *not* when the word is actually transmitted. Similarly, the time-tag of a received word represents when the word was read from the decoder. Thus, the time-tag does not necessarily reflect the exact time of the 429 bus activity.

4.2.6 Hit Counter

This number is incremented every time the Message Record is accessed by the DSP, so it represents the number of times a message has been received or transmitted. This option is enabled either for a single message by **`L43_MsgConfig`** or for all messages on the channel by **`L43_ChConfig`**. Since the Hit Counter uses the same field as the Time Tag, the Hit Counter may not be used concurrently with any of the time related fields.

4.2.7 Elapsed Time

The Elapsed Time is the most recent transmit interval (i.e., the time between the two most recent receptions or transmissions of the message.) The DSP calculates this value by subtracting the previous time-tag from the current time-tag. Resolution of the elapsed time is the same as the time-tag.

4.2.8 Min/Max Elapsed Time

The Minimum and Maximum Elapsed Times are the worst case Elapsed Times since the Lx429-3 was last activated. The DSP calculates the Elapsed Time, and if it is not between the current minimum and maximum, the appropriate value is updated. The Elapsed Time and Min/Max Elapsed Time options are enabled by the **`L43_ChConfig`** function. This function also initializes the Minimum time and the Maximum time to zero.

4.2.9 User Code Pointer

The user can specify individual Message Records to be serviced by custom DSP code. Contact Ballard Technology for further information on use of custom DSP code.

Sections 4.3 and 4.4 describe how the receive and transmit channels use Message Records.

4.3 Filter Tables

Each receive channel has a Filter Table as shown in Figure 4.3. A Filter Table is an array of pointers to Message Records. It contains one pointer for every Label/SDI combination (256 X 4 pointers). The Label/SDI bits of an ARINC word form an index into the Filter Table.

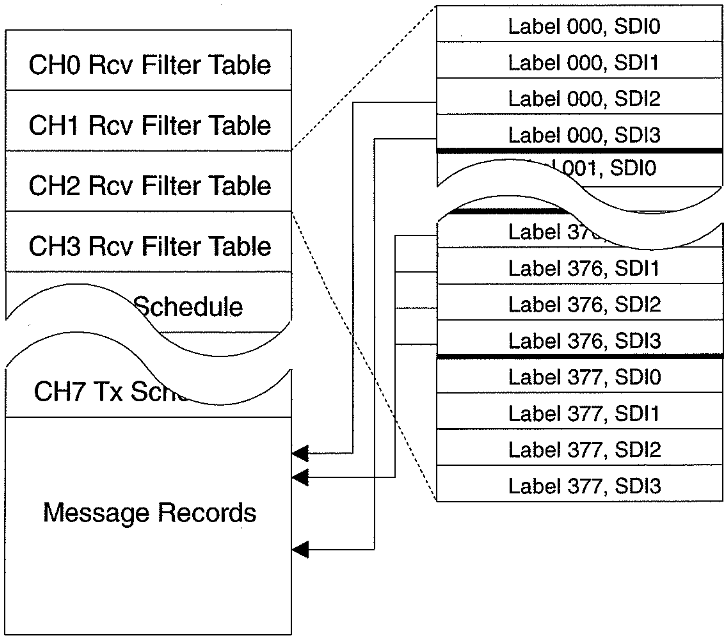


Fig. 4.3 Filter Tables

4.3.1 How Filter Tables work

When a word is received from the ARINC 429 databus, the Lx429-3 examines the label and SDI bits and retrieves the corresponding pointer from the Filter Table. If the pointer is zero, the Lx429-3 quits processing the word. Otherwise, the Lx429-3 writes the word to the Message Record to which the Filter Table points for that label/SDI. Other processing may follow depending on which options are enabled. For example, if the Elapsed Time option is enabled, the Lx429-3 calculates the time elapsed since the last reception of the word and writes this to the Message Record.

4.3.2 Configuring the Filter Tables

The user sets Filter Table entries to point to Message Records with the **L43_FilterSet** and **L43_FilterDefault** functions. **L43_ChConfig** sets up the Filter Table and therefore must precede any calls to **L43_FilterSet** or **L43_FilterDefault**. Any number of Filter Table entries may point to the same Message Record. For example, the SDIALL constant used in the **L43_FilterSet** function sets the pointers for all four SDIs of a particular label to the same Message Record as shown for Label 376 in Fig. 4.3.

4.4 Transmit Schedules

A Schedule (as described in Chapter 3) is a program executed by the Lx429-3 firmware that controls the transmission of words onto the ARINC 429 databus. There are two ways to create a Schedule. Chapter 3 demonstrated the easiest way: using the **L43_SchedBuild** function to automatically construct a Schedule. This section describes the Schedule in more detail and tells how to explicitly construct a Schedule.

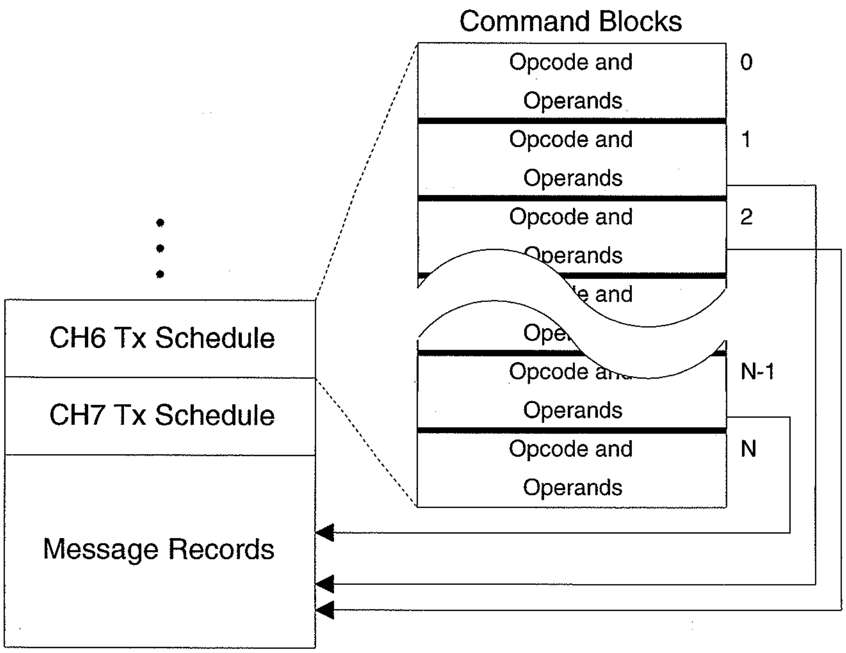


Fig. 4.4 Transmit Schedule

Opcode Name	Function	Description
MESSAGE	L43_SchedMsg	Transmits the ARINC 429 word from the Message Record
GAP	L43_SchedGap	Inserts a timed gap between transmissions (allows asynch msgs)
ENTRY	L43_SchedEntry	Indicates the starting point for the schedule
HALT	L43_SchedHalt	Halts the Schedule
PAUSE	L43_SchedPause	Halts processing of schedule until L43_ChResume is called
RESTART	L43_SchedRestart	Restarts the Schedule at beginning (resets channel)
INTERRUPT	L43_SchedInt	Generates interrupt/interrupt log list entry (See Section 4.7)
USERCODE	L43_SchedUser	Calls custom DSP code
Branch	L43_SchedBranch	Jumps to specified Command Block and resumes execution.
CALL	L43_SchedCall	Jumps to specified Cmd Block, saving the return address on stack.
RETURN	L43_SchedReturn	Returns to address following last Call.
NOP	L43_SchedNop	No operation.
PULSE	L43_SchedPulse	Pulse an external discrete signal
LISTGAP	L43_SchedGapList	Inserts a conditional timed gap for asynchronous msgs
FIXEDGAP	L43_SchedGapFixed	Inserts a timed gap that does not allow asynchronous msgs

Table 4.1 Command Blocks in the Transmit Schedule

4.4.1 How Schedules work

A Schedule is required for any transmission from the Lx429-3. Even if the Lx429-3 is to transmit only one word one time, a Schedule must be created. In this extreme case the Schedule would consist of a *Message* Command Block followed by a dummy *Gap* Command Block and a *Halt* Command Block.

The Schedule must be loaded by the host computer into Command Blocks in the memory on the Lx429-3. By default each transmit channel is allocated 512 Command Blocks as shown in Figure 4.4. Each Command Block contains one of the opcodes shown in Table 4.1.

Though very complex Schedules may be created using the special commands listed in Table 4.1, the typical Schedule consists of a loop of *Message* and *Gap* Command Blocks. When the DSP processes a *Message* Command Block, it retrieves the ARINC 429 word to be transmitted from the Message Record pointed to by the operand in the Command Block (see Figure 4.4). The DSP loads the word into the encoder and may update fields of the Message Record, depending on which options are enabled for that message. It then proceeds to the next Command Block.

A *Gap* Command Block triggers transmission of the current contents of the encoder and specifies period of the transmitter must wait before starting another transmission. The DSP services other channels during transmissions and gap times.

A few rules must be followed when developing a Transmit Schedule:

1. A *Message* Command Block should be followed by either another *Message* Command Block or a *Gap* Command Block. No transmission of the message occurs until a subsequent *Message* or *Gap* Command Block is processed.
2. The parameter in a *Gap* Command Block is the total time from the end of one message to the start of the next messages. The gap is measured in bit times at the speed set for that channel. For long gap times *Gap* Command Blocks may be strung together.
3. If a *Message* Command Block follows another *Message* Command Block, then a four (4) bit time gap is automatically inserted between the two messages.
4. There is an implicit *Restart* Command at the end of every Schedule, so the Schedule runs in a loop unless directed otherwise (e.g., by a *Halt* Command Block).
5. A subroutine (the destination for any *Call* or *Branch)* should precede the use of the *Call* or *Branch* in the schedule. The main starting point should follow subroutines and is indicated by the *Entry* Command Block.

4.4.2 Creating a Schedule

The Schedule is programmed explicitly using **`L43_SchedEntry`**, **`L43_SchedMsg`**, and **`L43_SchedGap`**, and other scheduling driver functions shown in Table 4.1. Each of these functions makes an entry into the next available Command Block at the end of the current Schedule. The opcode in the Command Block corresponds to the name of the function, and the operand is specified in the function parameters. Only the Message and Gap opcodes specifically relate to the transmission of ARINC 429 words. The other opcodes controlling the processing of the Schedule are explained for the individual functions in Appendix A. The following section demonstrates the use of the scheduling functions.

4.4.3 Example: Creating a Schedule with Explicit Timing

The example in Chapter 3 used `L43_SchedBuild` to automatically construct the Transmit Schedule required to meet given timing requirements. In contrast, this example specifies timing explicitly. Additionally, it verifies the transmitter's timing performance by configuring an Lx429-3 receive channel to receive data from the transmitter through its internal wrap around self-test feature and recording the minimum and maximum transmit intervals.

Suppose for this example that we want to transmit the two messages shown in Table 4.2. We choose target transmit intervals that are the approximate averages of the minimum and maximum transmit intervals given in the ARINC 429 specification. A Transmit Schedule that meets these timing requirements (accounting for word length and interword gaps) is shown in Fig. 4.5. If transmission of this sequence of words and gaps is repeated, the timing requirements for each word will be met. At slow speed (12.5Kbps) a bit-time is 0.08ms, and at high speed (100Kbps) it is 0.01ms, so each word takes either 2.56ms or 0.32ms.

ARINC Word (octal label)	Min Transmit Interval (ms)	Max Transmit Interval (ms)	Target Interval (ms)
Computed Airspeed (206)	62.5	125	90
Altitude Rate (212)	31.3	62.5	45

Table 4.2 Example transmit intervals

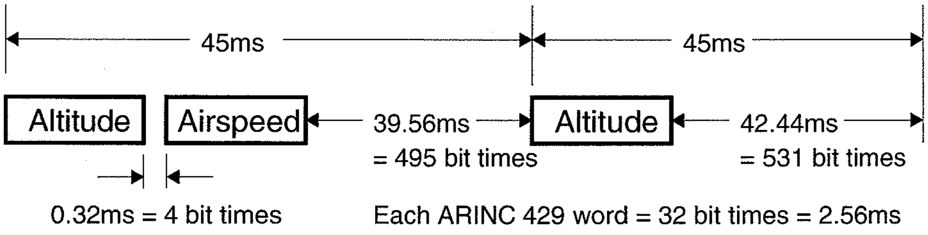


Fig. 4.5 Transmit Schedule for the low-speed timing requirements of Table 4.2

```
MSGSTRUCT xmt_airspeed, xmt_altitude, rcv_airspeed, rcv_altiutde;

MSGFIELDS altitude_record;
MSGFIELDS airspeed_record;


handleval = L43_CardOpen(cardnum);

// CONFIGURE RECEIVE CHANNEL 0 (CH0) AND FILTERS
L43_ChConfig(CHCFG_MAXMIN , CH0, handleval);
rcv_airspeed.addr =
L43_FilterSet(MSGCRT_DEFAULT,0206,SDIALL,CH0,handleval);
rcv_altitude.addr =
L43_FilterSet(MSGCRT_DEFAULT,0212,SDIALL,CH0,handleval);

// CONFIGURE TRANSMIT CHANNEL 4 (CH4) AND CREATE MESSAGES
L43_ChConfig(CHCFG_SELFTEST,CH4,handleval);
xmt_altitude.addr = L43_MsgCreate(MSGCRT_DEFAULT,handleval);
xmt_airspeed.addr = L43_MsgCreate(MSGCRT_DEFAULT,handleval);

// CREATE THE TRANSMIT SCHEDULE
L43_SchedMsg(xmt_altitude.addr,CH4,handleval);
L43_SchedMsg(xmt_airspeed.addr,CH4,handleval);
L43_SchedGap(495,CH4,handleval);
L43_SchedMsg(xmt_altitude.addr,CH4,handleval);
L43_SchedGap(531,CH4,handleval);

// INITIALIZE THE MESSAGE RECORDS FROM WHICH Lx429-3 WILL TRANSMIT
L43_MsgDataWr(0x0000008A,xmt_altitude.addr,handleval);
L43_MsgDataWr(0x00000086,xmt_airspeed.addr,handleval);

L43_CardStart(handleval);

//WAIT FOR A NUMBER OF TRANSMISSIONS
L43_CardStop(handleval);

L43_MsgBlockRd(&altitude_record,rcv_altitude.addr,handleval);
L43_MsgBlockRd(&airspeed_record,rcv_airspeed.addr,handleval);

printf("Max xmt interval of Altitude is %u \n", altitude_record.maxtime);
printf("Min xmt interval of Altitude is %u \n", altitude_record.mintime);
printf("Max xmt interval of AirSpeed is %u \n", airspeed_record.maxtime);
printf("Min xmt interval of AirSpeed is %u \n", airspeed_record.mintime);

L43_CardClose(handleval);
```

Fig. 4.6 Example with explicit transmit schedule.

The code for this example (Fig. 4.6) begins by creating message structures. MSGSTRUCT has members for the address of the Message Record and the value of the data; it is used as in previous examples. We will transmit from two Message Records, and we will receive these transmissions into a different pair of Message Records. MSGFIELDS is a structure with members for each of the fields in a Message Record. Its use is explained later.

After obtaining the handle of the card with **L43_CardOpen**, we use **L43_ChConfig** to configure a receive channel (CH0) with the MIN/MAX Elapsed Time enabled. The MIN/MAX Elapsed Time option is explained in Section 4.2.8.

Next, with the **L43_FilterSet** functions we allocate Message Records in which to save messages for the given labels. As in the examples of Chapter 3, we are ignoring SDI bits.

The next L43_ChConfig configures a transmit channel (CH4) with the self-test option enabled. The self-test option sends the output from CH4 back into all of the receive channels. Only one transmit channel should be connected to the self-test bus at any given time. L43_MsgCreate is used to create the two messages to be transmitted.

The L43_SchedMsg and L43_SchedGap functions create the schedule using the sequence of messages and gaps shown in Fig. 4.5. Notice that the minimum (four-bit) interword gap need not be explicitly scheduled. Also the time parameter of the L43_SchedGap function is in units of bit times. At low-speed a bit time is 0.08ms.

The L43_MsgDataWr functions initialize the values to be transmitted for Airspeed and Altitude Rate. The octal labels 206 and 212 translate to hexadecimal 86 and 8A, respectively. We do not care what the data values are for this example, so we have zeroed all but the label bits.

After L43_CardStart commands the Lx429-3 to begin operating, the two messages are transmitted in accordance with the Schedule. Meanwhile, the receiver records these transmissions along with the minimum and maximum times between successive receptions of each message. The Lx429-3 is allowed to run long enough for the minimum and maximum elapsed time values to stabilize. These minimum and maximum measurements are then available in the Message Records. Though it need not be, the card is stopped using L43_CardStop before the measurements are read from the card.

We use L43_MsgBlockRd to read the entire Message Record into the application. A pointer to the destination structure, which we declared at the beginning of the code, is a parameter of this function. The minimum and maximum elapsed time values are members of that structure as shown in Fig. 4.2. The *printf* functions print the measured ranges of the transmit intervals, which should correlate very closely to our expected values.

Schedules involving more messages may be implemented the same way, though, as mentioned in Chapter 3, calculating the timing requirements becomes substantially more difficult. Other actions such as interrupts to the host may also be scheduled.

4.5 Sequential Record

The Sequential Record is a buffer that holds a time-tagged history of the ARINC 429 words transmitted and received by the Lx429-3 (Figure 4.7). The Sequential Record can store messages from any or all Lx429-3 channels. Individual channels and/or individual messages within a channel may be selectively recorded. The filtering of desired messages is controlled as described in Section 3.6.

By default, recording halts when the Sequential Record is full. However, configuring the Sequential Record with the SEQCFG_CONTINUOUS flag in L43_SeqConfig makes the buffer circular. In this mode it automatically wraps around and continues recording, over-writing old messages. The Lx429-3 issues an interrupt (if this option is enabled) when the Sequential Record either

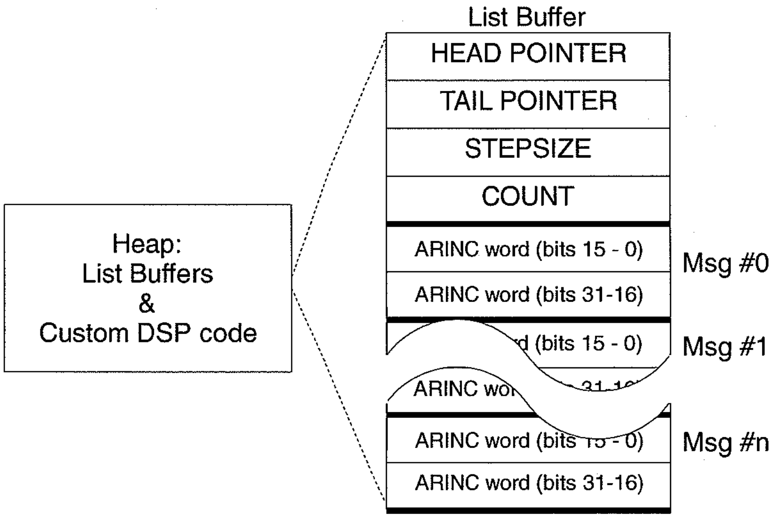


Fig. 4.9 Receive and Transmit List Buffer structure

4.6.1 Receive List Buffers

There are two types of list buffers (FIFO and ping-pong) that can be associated with a receive message. The type is specified using predefined constants when the list is created with `L43_ListRcvCreate`.

FIFO List Buffer:

When the value of received data in a particular label/SDI is rapidly changing and it is important not to lose any of the data, then a FIFO Receive List Buffer may be used. This may be especially true in communications or file transfer protocols such as ARINC 615 where there are streams of words with identical label fields separated by minimum (4-bit time) gaps. Thus, an interface unit participating in such a protocol must be capable of buffering long sequences of words.

The Lx429-3 adds messages to a FIFO Receive List Buffer as they are received, and the user retrieves them sequentially with the `L43_ListDataRd` function. The user is not required to maintain the pointers as long as the `L43_ListDataRd` function is used to access the Buffer. The pointers are automatically changed when a new word is entered into the list (i.e., when it is received) and when a word is read from the list using `L43_ListDataRd`. If messages are not read from the Receive List Buffer fast enough, the buffer wraps around and over-writes old messages. The Lx429-3 can issue an interrupt to signal this occurrence.

Ping-Pong List Buffer:

The ping-pong list buffer guaranties data integrity by preventing a problem that can occur when only a single buffer is used. The problem happens when the host computer and the Lx429-3 processor simultaneously access the same message in memory, causing the data being read by the host to contain part of the old message and part of the new message. The ping-pong list buffer solves this problem by using multiple memory locations, so that **`L43_ListDataRd`** always reads the most recent complete copy of a received message.

4.6.2 Scheduled Transmit List Buffers

There are three types of list buffers (FIFO, ping-pong, and circular) that may be associated with scheduled transmit messages. A Scheduled Transmit List Buffer is attached to a Message Record and is created using **`L43_ListXmtCreate`**. The type is specified using predefined constants. The user writes words sequentially to the List Buffer using the **`L43_ListDataWr`** function. The user is not required to maintain the pointers as long as the **`L43_ListDataWr`** function is used. Transmission timing is controlled by the transmit schedule. When the schedule indicates that a message should be transmitted, the next message in the list is used.

FIFO List Buffer:

As words in the Schedule are to be transmitted, they are sequentially obtained from the FIFO list buffer rather than directly from the Message Record. Whenever the Schedule indicates that a word should be transmitted from a Message Record that is associated with a FIFO list buffer, the Lx429-3 transmits the next available word in the buffer. If messages are not updated fast enough and all words have been transmitted at least once, then the last (most recent) word written by the host to the FIFO list buffer is the word that is transmitted until another word is written by the host. The Lx429-3 can issue an interrupt to signal that more data needs to be written by the host.

Ping-Pong List Buffer:

As in receive, the transmit ping-pong list buffer guaranties data integrity when the host computer and the Lx429-3 processor simultaneously access the same message in memory, which could cause the data being transmitted by the host to contain part of the old message and part of the new message. With a ping-pong transmit list buffer the Lx429-3 transmits the last complete message loaded using **`L43_ListDataWr`**.

Circular List Buffer:

With a circular list buffer transmissions repeatedly loop through the entire list buffer. This feature would greatly simplify the transmission of a data pattern, for example a sine wave or ramp, since the whole pattern could be preloaded into the List Buffer rather than requiring the host computer to update the data value for each transmission.

4.6.3 Asynchronous Transmit List Buffers

Although there may be other uses, asynchronous transmit list buffers are intended to support communications protocols such as ARINC 615. The Lx429-3 is capable of transmitting the asynchronous bursts of words typically involved in communications protocols without disrupting the timing of periodic words (the regular scheduled messages). A bi-directional communications protocol would involve the use of receive list buffers and asynchronous transmit list buffers.

Operation of the asynchronous transmit list buffers is very simple. When an allocated asynchronous transmit list buffer contains words that have not been transmitted (i.e., is not empty), the Lx429-3 automatically transmits words from the asynchronous transmit list buffer by interleaving them in the gaps of a running Schedule. The Lx429-3 fills any scheduled gap with as many words as will fit from the asynchronous transmit list buffer. The gap is then adjusted to preserve the timing of scheduled messages as shown in Figure 4.10. This architecture allows an application program to load conveniently sized packets of data from a file and let the Lx429-3 automatically manage their transmission. **`L43_ListStatus`** may be used to determine whether or not a List Buffer is empty.

Unlike other list buffers, an asynchronous transmit list buffer is associated with a channel, not a Message Record. To implement an asynchronous transmit list buffer use **`L43_ListAsyncCreate`** to create the list and associate it with a channel, and then create a transmit schedule using **`L43_SchedBuild`** (Section 3.5) or explicitly using **`L43_SchedMsg`** and **`L43_SchedGap`** (Section 4.4). If only asynchronous messages are to be transmitted and no messages are to be scheduled, then a dummy schedule must be created explicitly with a single large gap. Once the schedule is running use **`L43_ListDataWr`** to pass the data for transmission. See the example programs on the API distribution disk.

Asynchronous transmit list buffers are FIFO buffers. Words written to the list with **`L43_ListDataWr`** are transmitted only once in the order they are written. When the list is empty, no words are transmitted from the list until the host writes new words to it. As with other List Buffers, the user is not required to maintain the pointers as long as **`L43_ListDataWr`** is used to add words to the list.

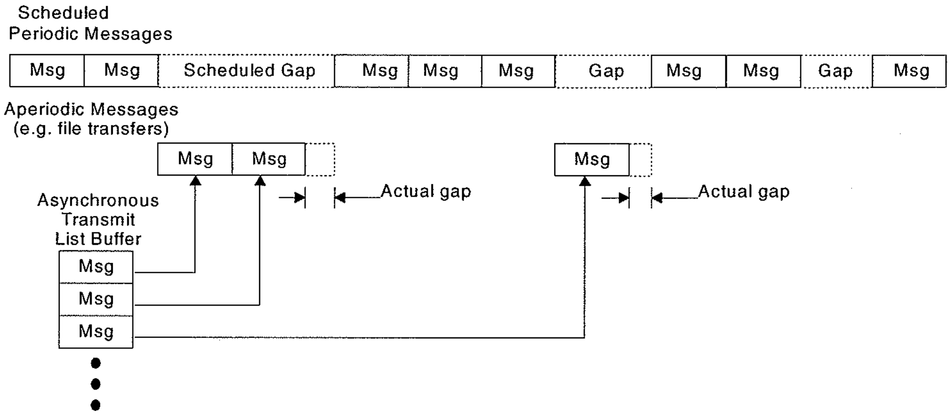


Fig. 4.10 Operation of Asynchronous Transmit List Buffer

4.7 True Dual Port Ram

The Lx429-3 has 32K bytes of true dual port RAM (DPRAM). This DPRAM is in addition to the standard RAM, which has dual port capability but requires more time to resolve contention. The DPRAM allows simultaneous access by both ports: the internal bus of the Lx429-3 and the host computer. The DPRAM is useful for time-critical applications since it minimizes the time used in the contention circuit, thus increasing data throughput. To further enhance the speed of the Lx429-3, a small portion of DPRAM is reserved for internal variables and the communication process. The remainder is available to the user.

The user can specify whether Message Records, the Sequential Record, and/or list buffers are located in DPRAM. Constants in various functions cause these structures to be set up in RAM or DPRAM. By default, all are located in standard RAM. The constant in **`L43_ChConfig`** sets the location for Message Records on that channel unless overridden by constants in **`L43_MsgCreate`**, **`L43_FilterSet`**, or **`L43_FilterDefault`**. The constants in **`L43_SeqConfig`**, **`L43_ListAsyncCreate`**, **`L43_ListRcvCreate`**, and **`L43_ListXmtCreate`** determine the location of the Sequential Record and list buffers. If the DPRAM is specified but is full, the structure is placed in standard RAM.

4.8 Error Injection

In some situations a user may want to evaluate a receiver's response to protocol errors in the transmitted message. For such situations the user can selectively direct the Lx429-3 to inject parity errors or gap errors.

4.8.1 Parity Errors

ARINC 429 specifies that messages (32-bit words) should be transmitted with odd parity, so the Lx429-3 defaults to odd parity. However, any transmit or receive channel on the Lx429-3 can be configured using **`L43_ChConfig`** so the parity is odd, even or used as data. When used as data, the parity circuits do not generate or check parity. Also, transmit channels and transmit messages can be made to invert the parity bit (send a parity error) using other constants in **`L43_ChConfig`** and **`L43_MsgCreate`**. Please refer to the functions in Appendix A for the appropriate constants to set these options.

4.8.2 Gap Errors

According to ARINC 429 the gap between messages should be at least four (4) bit-times. This requirement is automatically met when transmit schedules are created using **`L43_SchedBuild`** or the explicit scheduling described in Section 4.4. Normally, two sequential messages in a schedule are automatically separated by a four bit-time gap. If a gap of less than four bit-times is desired, then an explicit schedule must be created with an **`L43_SchedMsg`** replaced with an **`L43_SchedMsgEx`** for the message preceding the short gap. A parameter in **`L43_SchedMsgEx`** specifies the length of the gap.

4.9 Interrupts

Interrupts are versatile features of the Lx429-3. They can be used to signal that user specified events have occurred. Examples of when an interrupt may be generated are: when a specified message is received, when the transmit schedule reaches a certain point, when an error occurs, or when the Sequential Record or a list buffer needs service. The different conditions or events that can generate an interrupt are listed in Table 4.3. Interrupts can be configured for polling or generating a hardware interrupt to the host. This section explains how to incorporate interrupts into an application program.

4.9.1 Lx429-3 interrupt architecture

The user configures the Lx429-3 to generate interrupts on specific events. The Lx429-3 automatically records every interrupt event in the Interrupt Log List as shown in Fig. 4.11 and discussed in Section 4.9.3. The Interrupt Log List may be polled (read and evaluated) whether or not hardware interrupts are enabled. If hardware interrupts are enabled, then an interrupt service routine must be installed in the host. The Lx429-3 has one hardware interrupt signal into the host computer.

4.9.2 Using interrupts

Each interrupt event must be explicitly enabled in order to generate an interrupt. As shown in Table 4.3, each interrupt event is enabled by inclusion of a constant in one of the functions. In addition, the `L43_IntConfig` function must be called in order to create the Interrupt Log List.

When the program is running, the Interrupt Log List may be polled using `L43_IntRd`. This function returns a zero if the Interrupt Log List is empty, otherwise it may be evaluated to determine the source of the interrupt. Each entry in the Interrupt Log List may be read only once, since `L43_IntRd` automatically increments the list pointers each time it is called.

Using hardware interrupts rather than polling, requires an interrupt service routine and an understanding of the computer's operating system. The `L43_IntInstall` function is used to enable the hardware interrupt and to associate the interrupt service routine with the interrupts from the Lx429-3. The interrupt service routine must call `L43_IntReset` to clear the hardware interrupt. More discussion may be found under `L43_IntInstall` in Appendix A. Also, see the interrupt examples on the Lx429-3 software distribution disks.

4.9.3 Interrupt Log List

The Interrupt Log List shown in Fig. 4.11 is a circular buffer which records all interrupt events in order of occurrence. An entry is added to this list for each interrupt event. Since different events on any Lx429-3 channel can trigger an interrupt, it is up to the handler to identify the source of an interrupt in order to respond appropriately. An interrupt is identified by reading and evaluating its entry from the Interrupt Log List.

Each Interrupt Log List entry contains three fields. The Card#/Ch# field is identical to the field of the same name in a Message Record (see Fig. 4.2). The meaning of the Interrupt Information Word depends on the Interrupt Type. Table 4.3 summarizes the meanings and values of these parameters.

The `L43_IntRd` function returns the oldest entry from this list and updates the tail pointer. As long as this function is used to read the Interrupt Log List the user need not maintain the head and tail pointers shown in Fig. 4.11.

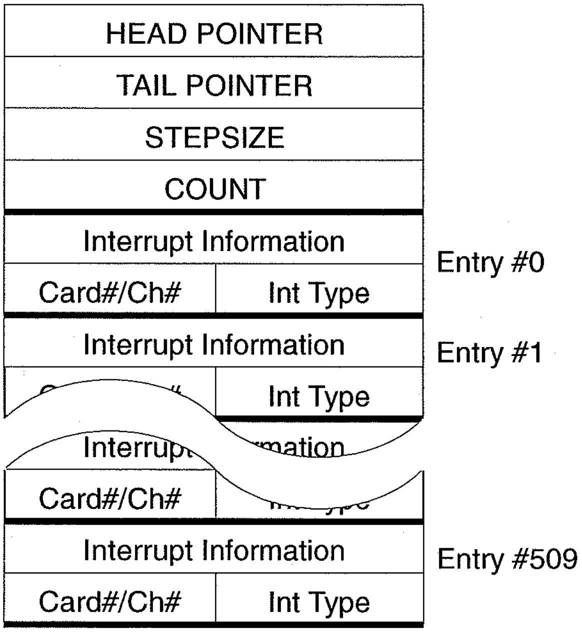


Fig. 4.11 Interrupt Log List

Interrupt Type[1]	Related to...	Condition/Event	Interrupt Information[2]	Enabling function(s)	Enabling constant
1	Message Records	Message Received	Msg Rec address	L43_MsgConfig L43_ChConfig	MSGCRT_INT CHCFG_INT
2	Schedule	*Interrupt* Command Block processed	User-assigned tag value	L43_SchedInt	
3	Schedule	*Halt* Command Block processed	Command Block address	L43_ChConfig L43_SchedHalt	CHCFG_INTHALT
4	Schedule	*Pause* Command Block processed	Command Block address	L43_ChConfig L43_SchedPause	CHCFG_INTPAUSE
5	Sequential Record	Sequential Record full (or wrapped around)	Address of last entry	L43_SeqConfig	SEQCFG_INTFULL
6	List Buffers	List Buffer empty/full (underflow or overflow)	List Buffer address	L43_ListAsyncCreate L43_ListRcvCreate L43_ListXmtCreate	LISTCRT_INT
7	Decoder	Error in received word	Message address	L43_ChConfig	CHCFG_INTERR
8	Sequential Record	n^{th} entry (user specified)	Address of last entry	L43_SeqConfig	SEQCFG_INTFREQ
9	717 Word	Processed 717 word	Word address	L43_SubFrmWordConfig L43_SuperFrmWordConfig	WRDCFG_WRDINT
A	717 Sub-frame	Processed 717 subframe	Subframe number	L43_SubFrmWordConfig L43_SuperFrmWordConfig	WRDCFG_SFINT
B	717 Out of sync	717 receive channel lost sync	Channel number	L43_ChConfig	CHCFG717_INTERR

Table 4.3 Lx429-3 Interrupts summary

[1] This number identifies the interrupt in the Interrupt Log List entry. It does NOT correspond to the ISA bus interrupt numbers.

[2] This is found in the Interrupt Log List entry.

APPENDIX A: ARINC 429 FUNCTION REFERENCE

This appendix provides detailed information on the primary ARINC 429 driver functions of the LP429-3 and LC429-3 (together referred to as Lx429-3). Information pertaining to ARINC 717 related functions is found in Appendix D. The descriptions and examples discussed here are intended for use with programs written in the C language. Users of other languages should contact Ballard Technology for assistance.

Overview of the Lx429-3 API

The API is divided into several functional categories. All function names contain a prefix (Ch, Sched, Mon, Msg, etc.) indicating the category to which they belong. The general naming convention for the API functions consists of the function prefix followed by an action. For example, the function **`L43_CardReset`** is a member of the "Card" function category, and will cause the card to be reset.

"hCard" parameters

Nearly all functions require a card "handle" parameter. **`L43_CardOpen`**, which is always the first Lx429-3 function called, returns this handle. The handle uniquely identifies a card when more than one Lx429-3 is used in a single computer. The handle is always the last parameter in any function that requires it.

"ctrlflags" parameters

Many functions have a *ctrlflags* parameter. Each bit controls an option in this bitmask parameter. Constants are defined in the header (.H) file for these parameters. The name of a constant reflects the function in which it is used (e.g., CHCFG_DEFAULT is used in the **`L43_ChConfig`** function). Option parameters are always first in the parameter list of a function that accepts them. The default options can always be selected by using the ??_DEFAULT constant where ?? depends on the function in which it is used (e.g., CHCFG_DEFAULT). When multiple options are selected, the constants should be bitwise OR-ed together. The default options are shown in bold in this appendix. Since the default constants are *#defined* to 0, only non-default constants actually need to be included in the OR-ing. The constants defined in the header file should be used by name (not value) in your code since the values are subject to change.

Schedule indices (SCHNDX)

All of the scheduling functions (**`L43_Sched??`**) return a value of type SCHNDX (schedule index). These functions append the Command Block index to the schedule. This index is a parameter of some of the advanced scheduling functions (e.g., **`L43_SchedCall`**, **`L43_SchedBranch`**).

"channel" parameters

Some functions take a channel parameter to specify which ARINC 429 channel applies to the function. The predefined constants listed below can be used for this parameter (configurations shown are for example only, see Table 2.2 for a more complete listing of channel configurations).

Constants	Description	Configuration of 1R/1T	Configuration of 2R/2T	Configuration of 4R/4T
CH0	Channel 0	Receiver	Receiver	Receiver
CH1	Channel 1	-	Receiver	Receiver
CH2	Channel 2	Transmitter	Transmitter	Receiver
CH3	Channel 3	-	Transmitter	Receiver
CH4	Channel 4	-	-	Transmitter
CH5	Channel 5	-	-	Transmitter
CH6	Channel 6	-	-	Transmitter
CH7	Channel 7	-	-	Transmitter

"message" parameters

Message data and related information such as the time-tag are stored in individual message records in the Lx429-3. All of the message manipulation functions (e.g., `L43_MsgDataRd`) require a message address parameter that uniquely identifies a message record. The `L43_MsgCreate` function creates these records and returns their addresses. The message address parameter always immediately precedes the card handle in Lx429-3 function calls.

The message values are 32-bit parameter specifying the value of an ARINC 429 word. The ARINC 429 word may contain a label, parity bit, SDI, SSM, and data bits. The functions expect the message value to be in a "reversed label" ARINC 429 format. The relationship between the two formats is shown below:

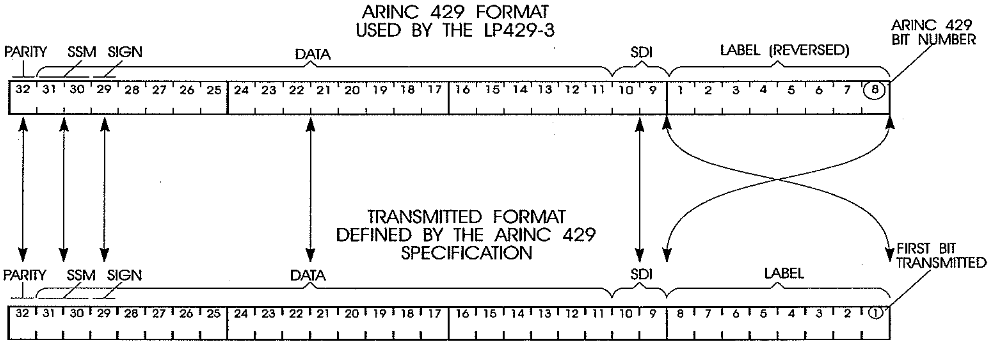


Table A.1 is a summary of the driver functions. As illustrated in the body of this manual, most applications can be implemented using just the bolded functions.

CARD Functions	
Function	Description
L43_CardClose	Disables access to a Lx429-3 card and releases its hardware resources
L43_CardOpen	Enables access to a Lx429-3 card and secure hardware resources
L43_CardReset	Resets the Lx429-3 hardware; destroys all existing configuration data
L43_CardResume	Starts the card without resetting the transmitter schedules
L43_CardStart	Starts operation of the Lx429-3 card
L43_CardStop	Stops operation of the Lx429-3 card
L43_CardTest	Performs a hardware test on the Lx429-3
CHANNEL Functions	
Function	Description
L43_ChClear	Clears either a transmit schedule or a receiver filter table
L43_ChConfig	Configures either a transmit or a receive channel
L43_ChPause	Pauses operation of a channel
L43_ChResume	Resumes operation of channel
FILTER Functions	
Function	Description
L43_FilterDefault	Creates a default message, and points the entire table to that message
L43_FilterRd	Reads the message address associated with a filter location
L43_FilterSet	Creates a message, and points the specified filter location to that message
L43_FilterWr	Writes a message address to the specified filter location
SCHEDULE Functions	
Function	Description
L43_SchedBranch	Inserts a BRANCH command into the schedule
L43_SchedBuild	Builds a schedule based on minimum and maximum transmit intervals
L43_SchedCall	Inserts a CALL command into the schedule
L43_SchedEntry	Sets the starting point of the schedule
L43_SchedGap	Inserts an inter-message gap into the schedule (calls either SchedGapFixed or SchedGapList based on the configuration of an Asynchronous List Buffer).
L43_SchedGapFixed	Inserts a fixed gap into the schedule.
L43_SchedGapList	Inserts a conditional inter-message gap into the schedule (used for Asynchronous List Buffers)
L43_SchedHalt	Inserts a HALT command into the schedule
L43_SchedInt	Inserts an INT command into the schedule
L43_SchedMsg	Inserts a message into the schedule
L43_SchedMsgEx	Inserts a message with a specified gap value into the schedule (used for inserting a gap time of less than four bit times)
L43_SchedPause	Inserts a PAUSE command into the schedule
L43_SchedRestart	Restarts the schedule at the beginning
L43_SchedReturn	Inserts a RET command into the schedule
MESSAGE Functions	
Function	Description
L43_MsgBlockRd	Reads an entire message record from the Lx429-3
L43_MsgBlockWr	Writes an entire message record on the Lx429-3
L43_MsgCreate	Creates and initializes a message record
L43_MsgDataRd	Reads the data associated with a message
L43_MsgDataWr	Writes the data associated with a message

Table A.1 ARINC 429 function summary

SEQUENTIAL RECORD Functions	
Function	**Description**
L43_SeqConfig	Configures the sequential record.
L43_SeqInterval	Sets the interval value if using interval mode
L43_SeqIntFrequency	Sets an interrupt frequency if using interrupt frequency mode
L43_SeqIsRunning	Determines whether the Sequential Record is in the process of recording
L43_SeqRd	Reads the next message record from the Sequential Record
LIST Functions	
Function	**Description**
L43_ListAsyncCreate	Creates an asynchronous transmit list buffer
L43_ListDataRd	Reads the next data value associated with a list buffer
L43_ListDataWr	Writes the next data value associated with a list buffer
L43_ListRcvCreate	Creates a receive list buffer
L43_ListXmtCreate	Creates a transmit list buffer
INTERRUPT Functions	
Function	**Description**
L43_IntConfig	Enables interrupts and initializes the interrupt log list
L43_IntInstall	Installs an interrupt handler (OS-dependent)
L43_IntRd	Reads an entry from the interrupt log list
L43_IntUninstall	Removes an interrupt handler (OS-dependent)
I/O Functions	
Function	**Description**
L43_ExtDIORd	Reads the value of the specified Digital I/O pin
L43_ExtDIOWr	Sets the value of the specified Digital I/O pin
L43_ExtLEDRd	Reads the On/Off value of the LED.
L43_ExtLEDWr	Sets the On/Off value of the LED.
TIMER Functions	
Function	**Description**
L43_TimerClear	Clears the time-tag timer
L43_TimerRd	Reads the current value of the time-tag timer
L43_TimerResolution	Selects a time-tag timer resolution
UTILITY Functions	
Command	**Description**
L43_BCDGetData	Extracts data value from BCD word.
L43_BCDGetMant	Extracts mantissa from a BCD word.
L43_BCDGetSign	Extracts the sign from a BCD word.
L43_BCDGetSSM	Extracts SSM field from BCD word.
L43_BCDGetVal	Calculates the value of a BCD word.
L43_BCDPutData	Inserts data value into BCD word.
L43_BCDPutMant	Inserts mantissa value into BCD word.
L43_BCDPutSign	Inserts the sign into a BCD word.
L43_BCDPutSSM	Inserts SSM field into BCD word.
L43_BCDPutVal	Inserts the value into a BCD word.
L43_BNRGetData	Extracts data value from BNR word.
L43_BNRGetMant	Extracts mantissa from a BNR word.
L43_BNRGetSign	Extracts the sign from a BNR word.
L43_BNRGetSSM	Extracts SSM field from BNR word.
L43_BNRGetVal	Calculates the value of a BNR word.
L43_BNRPutData	Inserts data value into BNR word.
L43_BNRPutMant	Inserts mantissa value into BNR word.
L43_BNRPutSign	Inserts the sign into a BNR word.
L43_BNRPutSSM	Inserts the SSM field into a BNR word.
L43_BNRPutVal	Inserts the value into a BNR word.

Table A.1 ARINC 429 function summary (continued)

UTILITY Functions (continued)	
Command	**Description**
L43_FldGetData	Extracts data field from ARINC 429 word.
L43_FldGetLabel	Extracts label field from ARINC 429 word.
L43_FldGetParity	Extracts parity bit from ARINC 429 word.
L43_FldGetSDI	Extracts SDI field from ARINC 429 word.
L43_FldGetValue	Extracts specified field from ARINC 429 word.
L43_FldPutData	Inserts the data field into an ARINC 429 word.
L43_FldPutLabel	Inserts the label field into an ARINC 429 word.
L43_FldPutSDI	Inserts the SDI field into an ARINC 429 word.
L43_FldPutValue	Inserts the specified field into an ARINC 429 word.
L43_GetChanCount	Gets the receiver and transmitter channel count.
L43_IsRcvChan	Checks if the specified channel is a receiver.
L43_IsXmtChan	Checks if the specified channels is a transmitter.
L43_ValFromAscii	Creates an integer value from an ASCII string
L43_ValGetBits	Extracts a bit field from an integer value
L43_ValPutBits	Puts a bit field into an integer value
L43_ValToAscii	Creates an ASCII string from an integer

Table A.1 ARINC 429 function summary (continued)

Error values

Functions that return an address return a zero if an error is encountered. Functions that return a value other than an address return a negative value if an error is encountered. The negative error values are listed in Table A.2 as well as the predefined constants that can be used to test for them. See the function description to determine if a particular function can return these error values.

ERROR Values		
Name	**Value**	**Description**
ERR_NONE	0	No error
ERR_UNKNOWN	-1	An unexpected error occurred
ERR_BADVER	-2	A bad version was encountered
ERR_BADPARAMS	-11	L43_CardOpen() called with bad parameters
ERR_NOHANDLES	-12	L43_CardOpen() already has allocated too many handles
ERR_NOCARD	-13	L43_CardOpen() could not find a Lx429-3 card at the specified address
ERR_NOIO	-14	L43_CardOpen() could not find the I/O ports
ERR_NOMEM	-15	L43_CardOpen() could not find the memory
ERR_WRONGMODEL	-17	Card does not support this feature
ERR_NOSEL	-18	L43_CardOpen() could not allocate a memory selector
ERR_LOCK	-19	The communication process is locked up
ERR_TOOMANY	-20	Too many channels have been configured
ERR_BADHANDLE	-21	A bad handle was specified
ERR_NOTCHAN	-23	Not a valid channel
ERR_NOTXMT	-24	The Transmitter has not been configured
ERR_NOTRCV	-25	The Receiver has not been configured
ERR_NOTMON	-26	The monitor has not been configured
ERR_ALLOC	-27	There is not enough memory to allocate
ERR_VXD	-28	An error occurred in the VXD
ERR_BADLABEL	-29	The specified label value is not valid
ERR_BADSDI	-30	The specified SDI value is not valid
ERR_BADMSG	-31	The specified command block is not a message block
ERR_BADSCHNDX	-32	Specified command index is out of range
ERR_BUFSIZE	-33	Insufficient space in user buffer
ERR_NOCONFIG	-34	The card has not been properly configured
ERR_CONFLICTS	-35	Unable to resolve conflicts in schedule
ERR_RANGE	-36	Schedule is out of range
ERR_FACTOR	-37	A bad factor value was specified

Table A.2 Errors summary (continued on next page)

ERROR Values (continued)		
Name	**Value**	**Description**
ERR_BOOTFULL	-41	No space to add boot code
ERR_BOOTNUM	-42	There is no boot code with the specified number
ERR_ACCESS	-43	Unable to write to access register
ERR_ROMVERIFY	-44	Unable to verify the value written to the ROM
ERR_COUNT	-45	An invalid count was specified
ERR_CRC	-46	There was a bad checksum in the HEX file
ERR_FNAME	-47	Bad filenames were specified
ERR_FRDWR	-48	There was an error reading or writing the HEX file
ERR_HEX	-49	There was a bad hex character in the HEX file
ERR_INDEX	-51	The command block index was invalid or the schedule is full
ERR_NOMSGS	-52	No messages specified
ERR_TYPE	-54	There was a bad type value in the HEX file
ERR_ZEROLEN	-55	Zero length was specified
ERR_BADADDRESS	-56	A bad address was specified
ERR_SELFIOFAIL	-71	I/O self-test failed
ERR_SELFMEMFAIL	-72	Memory self-test failed
ERR_SELFCOMMFAIL	-73	Communication self-test failed
ERR_SELFXMTFAIL	-74	Transmit self-test failed
ERR_PLXBUG	-75	PLX bug is causing problems
ERR_NOT717CHAN	-100	Specified channel is not a 717 channel
ERR_SUBFRMNUM	-101	Invalid 717 SubFrame number was specified
ERR_WORDNUM	-102	Invalid 717 Word number was specified
ERR_NOTINSYNC	-103	Not Synchronized to 717 databus
ERR_SUPERFRM	-104	717 SuperFrame not configured
ERR_SUPERFRMNUM	-105	Invalid 717 SuperFrame number was specified

Table A.2 Errors summary (continued)

The following pages contain descriptions of the Lx429-3 driver functions. The constants in bold in the tables on the following pages are the default options. Note that the “L43_” prefix has been omitted from the headings for easier reading, but all Lx429-3 functions must begin with “L43_” in source code.

BCDGetData .... Extracts the data value from a BCD word.

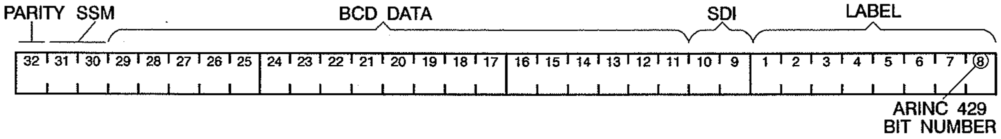
SYNOPSIS: ULONG L43_BCDGetData(msg, msb, lsb)

ULONG msg	BCD word to extract data from
USHORT msb	Most significant bit of BCD field
USHORT lsb	Least significant bit of BCD field

RETURNS: 32-bit value of data field.

DESCRIPTION: Extracts the data field of the BCD word in *msg*. *msb* and *lsb* specify the most significant and least significant bits of the BCD field respectively. The specified bits are converted to a 32-bit unsigned value. No other conversion is made.

The function assumes the BCD word has the following format:



Note: This is a utility function and does not access any Lx429-3 hardware.

WARNINGS: None.

SEE ALSO: L43_BCDPutData.

BCDGetMant .... Extracts the mantissa value from a BCD word.

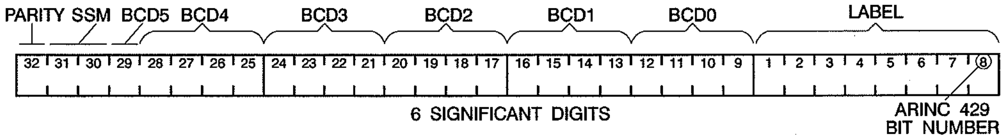
SYNOPSIS: ULONG L43_BCDGetMant(msg, sigdig)

ULONG msg	BCD word to extract data from
USHORT sigdig	Number of significant digits

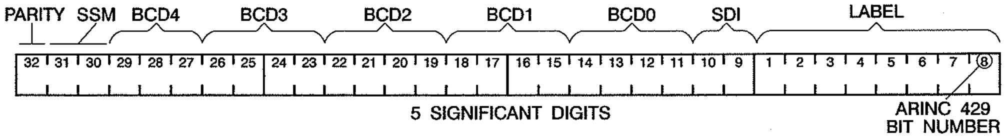
RETURNS: The data field mantissa.

DESCRIPTION: Extracts the data field of the BCD word specified in *msg*. *sigdig* specifies the number of BCD digits in the data field. The result is converted to a 32-bit unsigned value and returned. No other conversion is made.

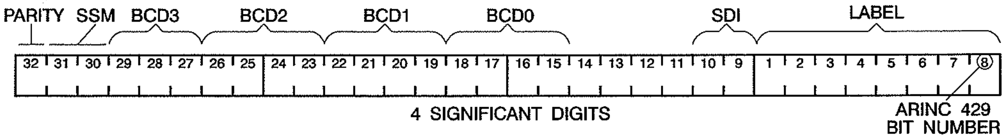
The function assumes the BCD data field is divided into the following fields:



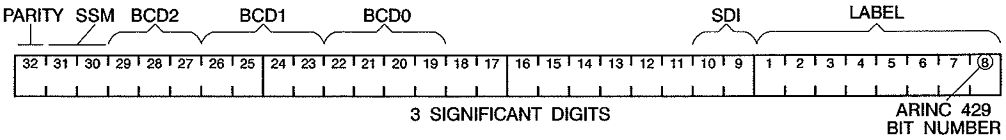
6 SIGNIFICANT DIGITS



5 SIGNIFICANT DIGITS



4 SIGNIFICANT DIGITS



3 SIGNIFICANT DIGITS

Note: This is a utility function and does not access any Lx429-3 hardware.

WARNINGS: None.

SEE ALSO: L43_BCDPutMant.

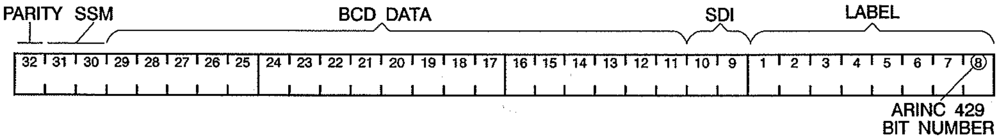
BCDGetSign .... Extracts the sign from a BCD word.

SYNOPSIS: USHORT L43_BCDGetSign(msg)
ULONG msg BCD word to extract data from

RETURNS: A non-zero value if sign of BCD word is negative, otherwise zero if the sign is positive.

DESCRIPTION: Returns the sign of the BCD word specified in *msg*. The result is non-zero if the sign of the BCD word is negative (SSM field equals 11 binary). Otherwise, the function returns a zero value.

The function assumes the SSM field is located at bits 30 through 31 as shown below:



Note: This is a utility function and does not access any Lx429-3 hardware.

WARNINGS: None.

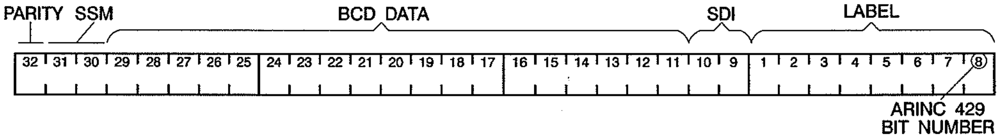
SEE ALSO: L43_BCDPutSign.

BCDGetSSM .... Extracts the SSM field from a BCD word.

SYNOPSIS: USHORT L43_BCDGetSSM(msg)
ULONG msg BCD word to extract data from

RETURNS: Value of the 2-bit SSM field.

DESCRIPTION: Extracts the SSM field of the BCD word in *msg*. The function assumes the SSM field is located at bits 30 through 31 as shown below:



Note: This is a utility function and does not access any Lx429-3 hardware.

WARNINGS: None.

SEE ALSO: L43_BCDPutSSM.

BCDGetVal............................. Calculates the value of a BCD word.

SYNOPSIS: VOID L43_BCDGetVal(resultstr, msg, sigdig, resolstr)

LPCSTR resultstr	Pointer to resulting ASCII string
ULONG msg	BCD word to extract data from
USHORT sigdig	Number of significant digits
LPCSTR resolstr	Pointer to resolution string

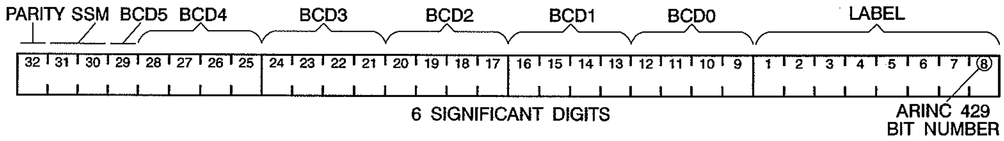
RETURNS: None.

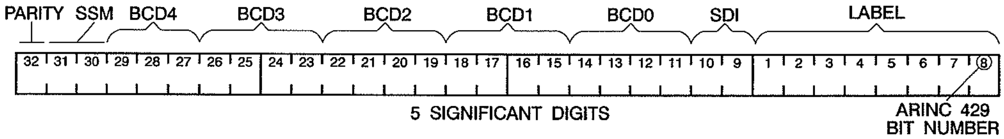
DESCRIPTION: Extracts the data field of the BCD word specified in *msg*. The resulting ASCII string is written to *resultstr*. This string may contain a decimal point and may be signed.

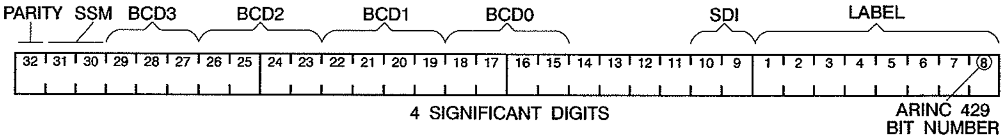
msg contains the 32-bit BCD word to extract data from. *sigdig* specifies the number of BCD digits in the data field.

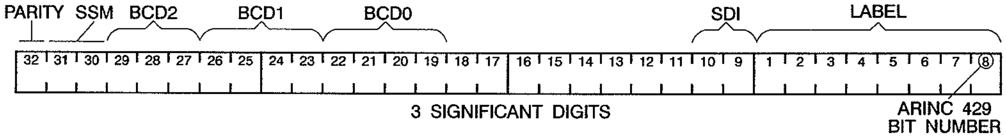
resolstr must point to an ASCII string specifying the resolution of the BCD data. This string may contain a decimal point if needed, but should not have a sign.

The function assumes the BCD data field is divided into the following fields:









Note: This is a utility function and does not access any Lx429-3 hardware.

WARNINGS: None.

SEE ALSO: L43_BCDPutVal.

BCDPutData.....................................Inserts the data value into a BCD word.

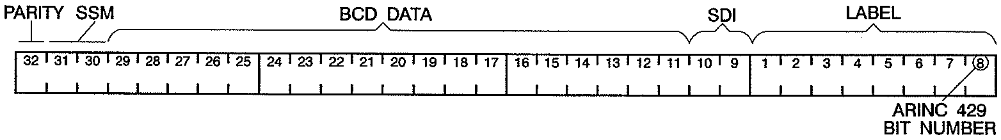
SYNOPSIS: ULONG L43_BCDPutData(msg, value, msb, lsb)

ULONG msg	32-bit BCD word
ULONG value	New data value
USHORT msb	Most significant bit of BCD field
USHORT lsb	Least significant bit of BCD field

RETURNS: The new 32-bit BCD word with the data field inserted.

DESCRIPTION: Inserts *value* into the data field of the BCD word specified by *msg*. *msb* and *lsb* specify the most significant and least significant bits of the field respectively. *value* is converted to BCD and inserted into the specified bits. No other conversion is made.

The function assumes the BCD word has the following format:



Note: This is a utility function and does not access any Lx429-3 hardware.

WARNINGS: None.

SEE ALSO: L43_BCDGetData.

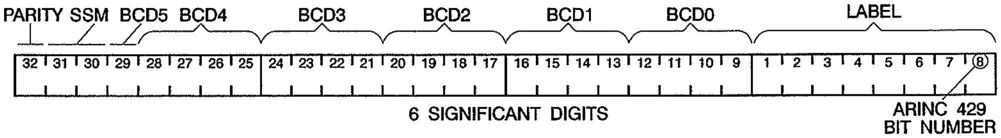
BCDPutMant ....Inserts the mantissa value into a BCD word.

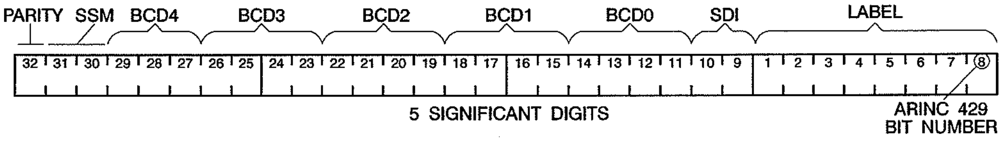
SYNOPSIS: ULONG L43_BCDPutMant(msg, value, sigdig, sign)

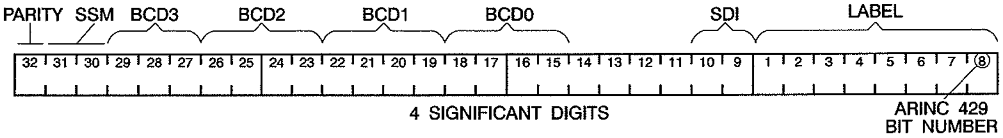
ULONG msg	32-bit BCD word
ULONG value	New data value
USHORT sigdig	Number of significant digits
USHORT sign	Sign for SSM field

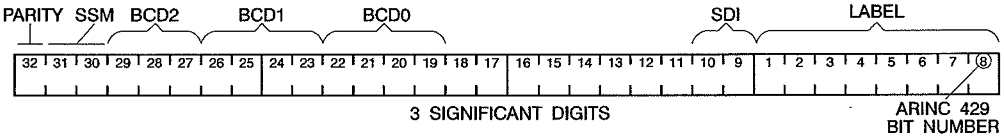
RETURNS: The new 32-bit BCD word with the data field inserted.

DESCRIPTION: Inserts the *value* of the data field into the BCD word specified in *msg*. *sigdig* specifies the number of digits in the BCD field. *value* is converted to BCD and inserted into the data field. The BCD data field is assumed to be divided into the following fields:









If *sign* is non-zero, the value 11 binary is inserted into the SSM field to denote a signed value. Otherwise, the value 00 binary is inserted into the SSM field. No other conversion is made.

Note: This is a utility function and does not access any Lx429-3 hardware.

WARNINGS: None.

SEE ALSO: L43_BCDGetMant.

BCDPutSign ....Inserts the sign into a BCD word.

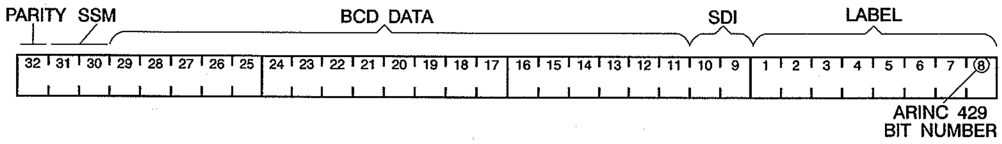
SYNOPSIS: ULONG L43_BCDPutSign(msg, sign)

ULONG msg	32-bit BCD word
USHORT sign	Sign for SSM field

RETURNS: The new 32-bit BCD word with the SSM field inserted.

DESCRIPTION: Inserts *sign* into the SSM field of the BCD word in *msg*. If *sign* is non-zero, the value 11 binary is inserted into the SSM field to specify a signed value. Otherwise, the value 00 binary is inserted into the SSM field.

The function assumes the SSM field is located at bits 30 through 31 as shown below:



Note: This is a utility function and does not access any Lx429-3 hardware.

WARNINGS: None.

SEE ALSO: L43_BCDGetSign.

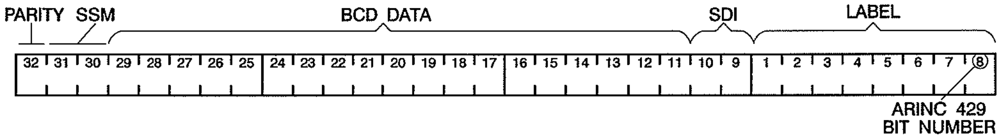
BCDPutSSM..........................Inserts the SSM field into a BCD word.

SYNOPSIS: ULONG L43_BCDPutSSM(msg, value)

ULONG msg	32-bit BCD word
USHORT value	2-bit value of SSM field

RETURNS: The new 32-bit BCD word with the SSM field inserted.

DESCRIPTION: Inserts *value* into the SSM field of the BCD word in *msg*. The function assumes the SSM field is located at bits 30 through 31 as shown below:



Note: This is a utility function and does not access any Lx429-3 hardware.

WARNINGS: None.

SEE ALSO: L43_BCDGetSSM.

BCDPutVal .... Inserts the value into a BCD word.

SYNOPSIS: ULONG L43_BCDPutVal(valuestr, msg, sigdig, resolstr)

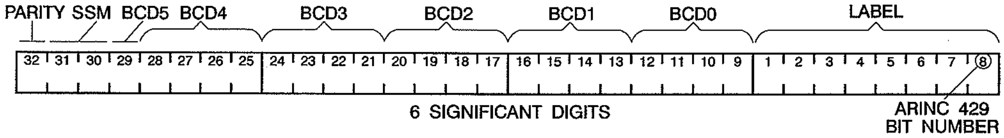
LPCSTR valuestr	Pointer to value string
ULONG msg	32-bit BCD word
USHORT sigdig	Number of significant digits
LPCSTR resolstr	Pointer to resolution string

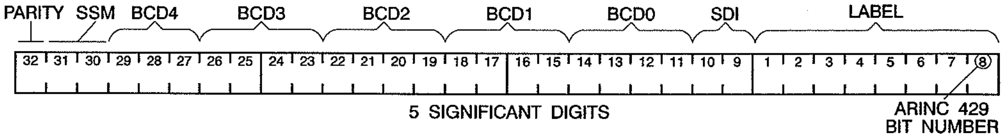
RETURNS: The new 32-bit BCD word with the data field inserted.

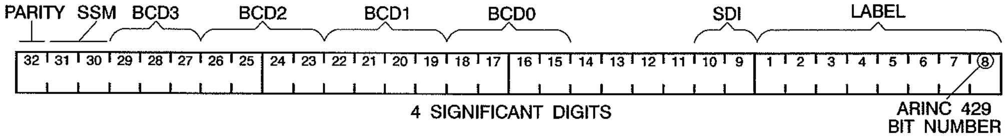
DESCRIPTION: Inserts a new value of the data field into the BCD word specified by *msg*. *valuestr* must be an ASCII string which contains the value to insert. It may contain a decimal point and may be signed. *sigdig* specifies the number of BCD digits in the data field.

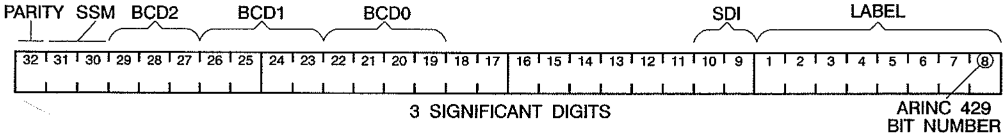
resolstr must point to an ASCII string specifying the resolution of the BCD data. This string may contain a decimal point if needed, but should not have a sign.

The BCD data field is assumed to be divided into the following fields:









Note: This is a utility function and does not access any Lx429-3 hardware.

WARNINGS: None.

SEE ALSO: L43_BCDGetVal.

BNRGetData .... Extracts the data value from a BNR word.

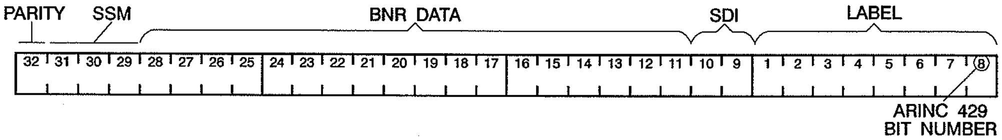
SYNOPSIS: ULONG L43_BNRGetData(msg, msb, lsb)

ULONG msg	BNR word to extract data from
USHORT msb	Most significant bit of BNR field
USHORT lsb	Least significant bit of BNR field

RETURNS: 32-bit value of data field.

DESCRIPTION: Extracts the data field from the BNR word specified in *msg*. *msb* and *lsb* specify the most significant and least significant bits of the BNR data field respectively.

The function assumes the BNR word has the following format:



Note: This is a utility function and does not access any Lx429-3 hardware.

WARNINGS: None.

SEE ALSO: L43_BNRPutData.

BNRGetMant .... Extracts the mantissa value from a BNR word.

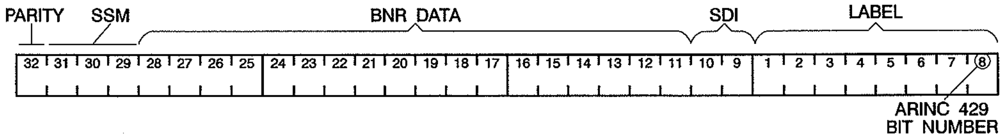
SYNOPSIS: ULONG L43_BNRGetMant(msg, sigdig)

ULONG msg	BNR word to extract data from
USHORT sigdig	Number of significant digits

RETURNS: 32-bit value of data field.

DESCRIPTION: Extracts the data field of the BNR word specified in *msg*. sigdig specifies the number of significant digits in the data field. If the SSM field of *msg* specifies a signed value, then the two's complement of the data field is returned. No other conversion is made.

The BNR data field is signed if bit 29 in the SSM field is non-zero. The BNR data field is assumed to be left-adjusted at bit 28 as shown below.



Note: This is a utility function and does not access any Lx429-3 hardware.

WARNINGS: None.

SEE ALSO: L43_BNRPutMant.

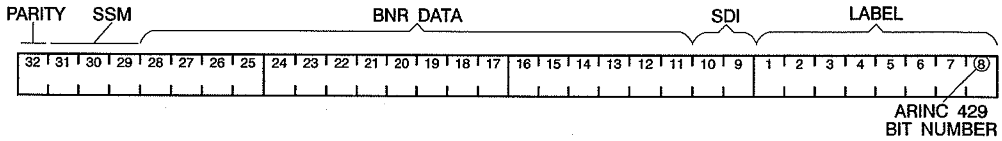
BNRGetSign............................Extracts the sign from a BNR word.

SYNOPSIS: USHORT L43_BNRGetSign(msg)
ULONG msg   BNR word to extract data from

RETURNS: A non-zero value if sign of BNR word is negative.

DESCRIPTION: Returns the sign of the BNR word in *msg*. The result is non-zero if the sign of the BNR word is negative (bit 29 of the SSM field is non-zero). Otherwise, the function returns a zero value.

The function assumes the SSM field is located at bits 29 through 31 as shown below:



Note: This is a utility function and does not access any Lx429-3 hardware.

WARNINGS: None.

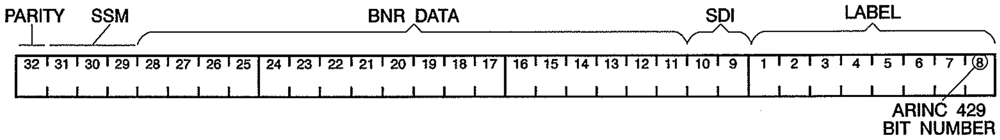
SEE ALSO: L43_BNRPutSign.

BNRGetSSM .... Extracts the SSM field from a BNR word.

SYNOPSIS: USHORT L43_BNRGetSSM(msg)
ULONG msg BNR word to extract SSM field from

RETURNS: Value of SSM field.

DESCRIPTION: Extracts the SSM field from the BNR word in *msg*. The function assumes the SSM field is located at bits 29 through 31 as shown below:



Note: This is a utility function and does not access any Lx429-3 hardware.

WARNINGS: None.

SEE ALSO: L43_BNRPutSSM.

BNRGetVal............................ Calculates the value of a BNR word.

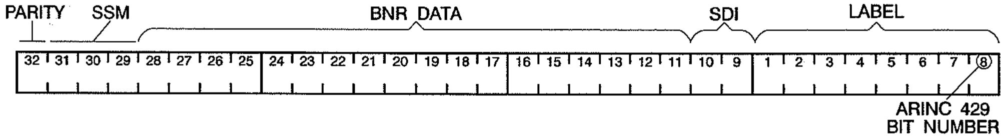
SYNOPSIS: VOID L43_BNRGetVal(resultstr, msg, sigdig, resolstr)

LPCSTR resultstr	Pointer to resulting ASCII string
ULONG msg	BNR word to extract data from
USHORT sigdig	Number of significant digits
LPCSTR resolstr	Pointer to resolution string

RETURNS: None.

DESCRIPTION: Extracts the data field from the BNR word in *msg*. *resultstr* is an ASCII string containing the results. This string may contain a decimal point and a negative sign.

sigdig specifies the number of significant digits in the data field. The function assumes the data field is left-adjusted at bit 28 as shown below:



resolstr points to an ASCII string specifying the resolution of the BNR data. This string may contain a decimal point if needed, but should not have a negative sign.

Note: This is a utility function and does not access any Lx429-3 hardware.

WARNINGS: None.

SEE ALSO: L43_BNRPutVal.

BNRPutData........................Inserts the data value into a BNR word.

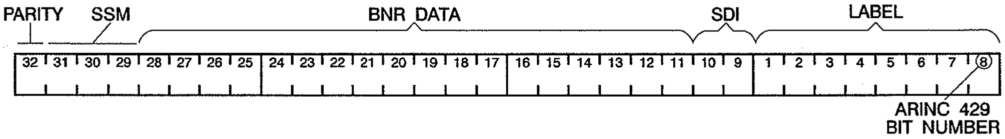
SYNOPSIS: ULONG L43_BNRPutData(msg, value, msb, lsb)

ULONG msg	32-bit BNR word
ULONG value	New data value
USHORT msb	Most significant bit of BNR field
USHORT lsb	Least significant bit of BNR field

RETURNS: The new 32-bit BNR word with the data field inserted.

DESCRIPTION: Inserts *value* into the data field of the BNR word specified in *msg*. *msb* and *lsb* specify the most significant and least significant bits of the BNR field respectively. No other conversion is made.

The function assumes the BNR word has the following format:



Note: This is a utility function and does not access any Lx429-3 hardware.

WARNINGS: None.

SEE ALSO: L43_BNRGetData.

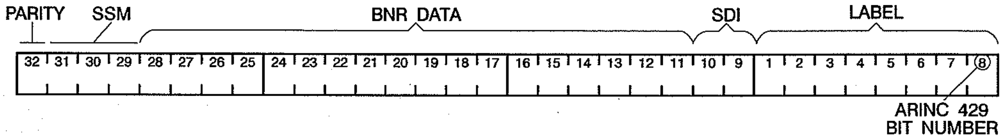
BNRPutMant .... Inserts the mantissa value into a BNR word.

SYNOPSIS: ULONG L43_BNRPutMant(msg, value, sigdig, twos)

ULONG msg	32-bit BNR word
ULONG value	New data value
USHORT sigdig	Number of significant digits
USHORT twos	Two's complement field

RETURNS: The new 32-bit BNR word with the data field inserted.

DESCRIPTION: Inserts *value* into the data field of the BNR word specified in *msg*. *sigdig* specifies the number of significant digits in the BNR field. The function assumes the BNR data field is left-adjusted at bit 28 as shown below.



If *twos* is non-zero, then the two's complement of *value* is inserted into the data field, and bit 29 of the SSM field is set to one. Otherwise, bit 29 of the SSM field is set to zero.

Note: This is a utility function and does not access any Lx429-3 hardware.

WARNINGS: None.

SEE ALSO: L43_BNRGetMant.

BNRPutSign .... Inserts the sign into a BNR word.

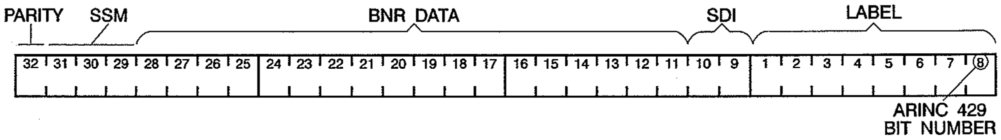
SYNOPSIS: ULONG L43_BNRPutSign(msg, sign)

ULONG msg	32-bit BNR word
USHORT sign	Sign value

RETURNS: The new 32-bit BNR word with the SSM field inserted.

DESCRIPTION: Inserts *sign* into the SSM field of the BNR word specified in *msg*. If *sign* is non-zero, then bit 29 of the SSM field is set to one. Otherwise, bit 29 of the SSM field is set to zero.

The function assumes the SSM field is located at bits 29 through 31 as shown below:



Note: This is a utility function and does not access any Lx429-3 hardware.

WARNINGS: None.

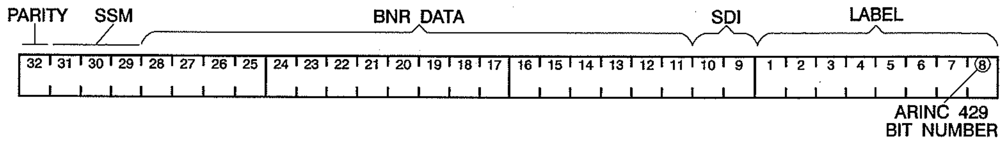
SEE ALSO: L43_BNRGetSign.

BNRPutSSM .... Inserts the SSM field into a BNR word.

SYNOPSIS: ULONG L43_BNRPutSSM(msg, value)
ULONG msg 32-bit BNR word
USHORT value 3-bit value of SSM field

RETURNS: The new 32-bit BNR word with the SSM field inserted.

DESCRIPTION: Inserts *value* into the 3-bit SSM field of the BNR word specified by *msg*. The function assumes the SSM field is located at bits 29 through 31 as shown below:



Note: This is a utility function and does not access any Lx429-3 hardware.

WARNINGS: None.

SEE ALSO: L43_BNRGetSSM.

BNRPutVal .... Inserts the value into a BNR word.

SYNOPSIS: ULONG L43_BNRPutVal(valuestr, msg, sigdig, resolstr)

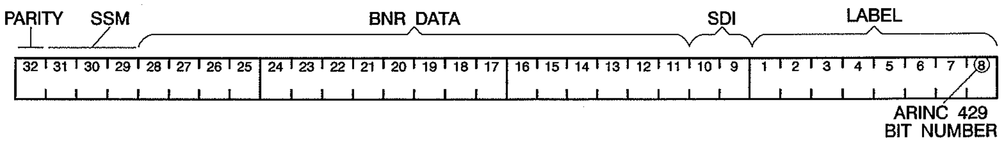
LPCSTR valuestr	Pointer to value string
ULONG msg	32-bit BNR word
USHORT sigdig	Number of significant digits
LPCSTR resolstr	Pointer to resolution string

RETURNS: The new 32-bit BNR word with the data field inserted.

DESCRIPTION: Inserts *valuestr* into the data field of the BNR word specified in *msg*. *valuestr* is an ASCII string containing the value to insert. This string may contain a decimal point and may be signed. *sigdig* specifies the number of significant digits in the BNR data field.

resolstr points to an ASCII string specifying the resolution of the BNR data. This string may contain a decimal point if needed, but should not have a sign.

The function assumes the data field is left-adjusted at bit 28 as shown below:



Note: This is a utility function and does not access any Lx429-3 hardware.

WARNINGS: None.

SEE ALSO: L43_BNRGetVal.

CardClose.. Disables access to a Lx429-3 card.

SYNOPSIS: ERRVAL L43_CardClose(hCard)

HCARD hCard	Handle of Lx429-3 card

RETURNS: A negative value if an error occurs, or zero if successful.

DESCRIPTION: Disables access to the specified Lx429-3 card and releases the associated hardware resources (memory and I/O space, interrupt number and DMA channel). This function does not stop the Lx429-3 from operating.

WARNINGS: This function MUST be called before a program terminates to prevent the associated hardware resources from being lost. This is especially important in Microsoft Windows operating systems.

SEE ALSO: L43_CardOpen

CardOpen .... Enables access to a Lx429-3 card.

SYNOPSIS: HCARD L43_CardOpen(cardnum)
INT cardnum Card number of Lx429-3 card

RETURNS: A card handle to the specified card if successful, otherwise a negative value to indicate an error.

DESCRIPTION: L43_CardOpen checks for the presence of a Lx429-3 that has been assigned *cardnum* by the operating system. If L43_CardOpen finds the card, it performs a quick hardware self-test of the card. Since this function returns the card handle parameter required by all other functions, this function is always the first Lx429-3 API called by a program.

See Section 2.0 (Installation) of the Lx429-3 manual for more information on card numbers.

WARNINGS: See the README.TXT files accompanying your driver software for latest information.

L43_CardClose must be called to release the hardware resources before the program terminates.

SEE ALSO: L43_CardClose

CardReset .... Resets the Lx429-3 hardware.

SYNOPSIS: VOID L43_CardReset(hCard)

HCARD hCard	Handle of Lx429-3 card

RETURNS: None.

DESCRIPTION: Stops the specified Lx429-3 and clears all configuration information on the card. If a 429 message is being processed, the processing is allowed to finish before the card is halted.

WARNINGS: None.

SEE ALSO: L43_CardStart, L43_CardStop

CardResume............................Resumes operation of the specified Lx429-3.

SYNOPSIS: ERRVAL L43_CardResume(hCard)
HCARD hCard Handle of Lx429-3 card

RETURNS: A negative value if an unable to resume, otherwise zero.

DESCRIPTION: Reactivates the specified Lx429-3 from the point at which it was stopped using L43_CardStop. The following list compares the difference between calling L43_CardResume and L43_CardStart.

Feature	**When CardStart is called....**	**When CardResume is called....**
Transmit Schedule	Execution starts at the start of the transmit schedule.	Execution resumes at the point the transmit schedule was stopped.
Interrupt Log List	Any unread entries in the interrupt log list are cleared before the card is started.	Any unread records in the interrupt log list are preserved before the card is started.
Sequential Record	Any unread records in the sequential record are cleared before the card is started.	Any unread records in the sequential record are preserved before the card is resumed.

WARNINGS: A call to L43_CardStop must precede this function.

SEE ALSO: L43_CardStart, L43_CardStop

CardStart..Starts operation of the specified Lx429-3.

SYNOPSIS: ERRVAL L43_CardStart(hCard)
HCARD hCard Handle of Lx429-3 card

RETURNS: Returns a negative value if unable to start, otherwise zero.

DESCRIPTION: Activates all configured channels of the specified Lx429-3. A configured channel will do the following:
Transmitters: Starts processing its schedule.
Receivers: Start receiving filtered data.

The sequential record and interrupt log list are cleared and begin operation at the start of their allocated buffers.

WARNINGS: The Lx429-3 continues operating even after an application program ends unless L43_CardStop halts it.

SEE ALSO: L43_CardStop

CardStop .... Stops operation of the specified Lx429-3.

SYNOPSIS: BOOL L43_CardStop(hCard)
HCARD hCard Handle of Lx429-3 card

RETURNS: TRUE if the card was active.

DESCRIPTION: Stops operation of the specified Lx429-3. If a 429 message is being processed, the processing is allowed to finish before the card is halted.

WARNINGS: None.

SEE ALSO: L43_CardStart

CardTest .... Performs a hardware test on the Lx429-3.

SYNOPSIS:

ERRVAL L43_CardTest(level, hCard)	
USHORT level	Level of tests to perform
HCARD hCard	Handle of Lx429-3 card

RETURNS: A negative value if an error occurs, or zero if successful.

DESCRIPTION: Executes the test specified by the *level* constant. When the test completes, the card is left in the same state as after a L43_CardReset.

level	
Constant	**Description**
TEST_LEVEL_0	Tests the port I/O interface of the Lx429-3. The test reads and writes each I/O port with a walking-bit pattern.
TEST_LEVEL_1	In addition to testing Level 0 tests, this level tests the memory interface of the Lx429-3. The test performs a pattern test of the RAM.
TEST_LEVEL_2	In addition to testing Level 1 tests, this level tests the communication process of the Lx429-3. The test performs a pattern test of the RAM using the communication process.
TEST_LEVEL_3	In addition to testing Level 2 tests, this level tests the encoders and decoders of the Lx429-3. The card transmits from all the transmit channels and verifies they are received on all the receive channels. (This utilizes a self-test wrap around on the Lx429-3.)

WARNINGS: This function disrupts normal operation of the card. Do not use this function when the receivers are connected to an active databus, as the results will be unpredictable.

SEE ALSO: L43_CardReset

ChClear ....Clear the contents of a channel.

SYNOPSIS: ERRVAL L43_ChClear(channel, hCard)

INT channel	Number of the channel
HCARD hCard	Handle of Lx429-3 card

RETURNS: A negative value if an error occurs, or zero if successful.

DESCRIPTION: Clears the contents of the specified *channel*. If the channel is a transmitter, all command blocks in the transmit schedule are deleted. If the channel is a receiver, all filters are deleted. The contents of the message records are unaffected. The configuration options for the channel previously specified by L43_ChConfig are unchanged.

WARNINGS: None.

SEE ALSO: L43_CardReset, L43_ChConfig

ChConfig .... Configure transmit or receive channel.

SYNOPSIS: ERRVAL L43_ChConfig(ctrlflags, channel, hCard)

ULONG ctrlflags	Selects channel options
INT channel	Number of channel
INT hCard	Handle of Lx429-3 card

RETURNS: A negative value if an error occurs, or zero if successful.

DESCRIPTION: Configures the *channel* of the specified card by performing the following steps:

- Stops the channel.
- Clears transmit schedule for the specified channel (filter tables are not affected).
- Writes Lx429-3 options defined by *ctrlflags*.
- Restarts the channel if previously started.

	ctrlflags		
Constant	**Description**	**Rcv**	**Xmt**
CHCFG_DEFAULT	Select all default settings (**bold** below)	✔	✔
CHCFG_LOWSPEED	Select low speed messages (12.5kHz)	✔	✔
CHCFG_HIGHSPEED	Select high speed messages (100kHz)	✔	✔
CHCFG_AUTOSPEED	Select auto speed detection of messages	✔	
CHCFG_PARODD	Select odd parity	✔	✔
CHCFG_PAREVEN	Select even parity	✔	✔
CHCFG_PARDATA	Select parity bit as data (ignores parity)	✔	✔
CHCFG_SEQSEL	Sequential monitoring selected at message level	✔	✔
CHCFG_SEQALL	Every message will be recorded in the Sequential Record	✔	✔
CHCFG_NOINTHALT	No interrupt will be issued on a HALT command		✔
CHCFG_INTHALT	An interrupt will be issued on a HALT command		✔
CHCFG_NOINTPAUSE	No interrupt will be issued on a PAUSE command		✔
CHCFG_INTPAUSE	An interrupt will be issued on a PAUSE command		✔
CHCFG_NOINTERR	No interrupt will be issued when a decoder detects an error	✔	
CHCFG_INTERR	An interrupt will be issued when a decoder detects an error	✔	
CHCFG_TIMETAGOFF	The timetag option is selected at the message level	✔	✔
CHCFG_TIMETAG	All messages will record a timetag.	✔	✔
CHCFG_ELAPSEOFF	The elpase timing option is selected at the message level	✔	✔
CHCFG_ELAPSE	All messages will record an elapsed time.	✔	✔
CHCFG_MAXMINOFF	Max and min repetition rates are selected at the message level	✔	✔
CHCFG_MAX	All messages will record a max time	✔	✔
CHCFG_MIN	All messages will record a min time	✔	✔
CHCFG_MAXMIN	All messages will record a max and a min time	✔	✔

ctrlflags (continued)			
Constant	**Description**	**Rcv**	**Xmt**
CHCFG_NOHIT	The hit counter is selected at the message level (can't have hit counter with time tag, elapse, or max/min timing)	✔	✔
CHCFG_HIT	The hit counter is selected for all messages (can't have hit counter with time tag, elapse, or max/min timing)	✔	✔
CHCFG_SELFTESTOFF	This channel will not transmit on the self-test bus	✔	✔
CHCFG_SELFTEST	This channel will transmit on the self-test bus (only one transmitter can be on the self-test bus at a time; therefore, only the last transmit channel configured to use the self-test bus will use this option).	✔	✔
CHCFG_SYNCOFF	The SYNCOUT signal option will be selected at the message level		✔
CHCFG_SYNC	The SYNCOUT signal will be active for all messages of this channel		✔
CHCFG_EXTOFF	Messages are selected for an external trigger at the message level		✔
CHCFG_EXTTRIG	All messages for this channel will be externally triggered		✔
CHCFG_NOERR	Error injection is selected at the message level		✔
CHCFG_PARERR	All messages will have a parity error (inverts the parity bit)		✔
CHCFG_UNPAUSE	The channel will initially be un-paused	✔	✔
CHCFG_PAUSE	The channel will initially be paused	✔	✔
CHCFG_RAM	All messages will default to RAM for this channel (can be over-ridden at message level)	✔	✔
CHCFG_DPRAM	All messages will default to Dual Port RAM for this channel (can be over-ridden at message level)	✔	✔

WARNINGS: The function clears any previous contents of the channel.

SEE ALSO: L43_CardStart, L43_CardStop

ChPause .... Pause the operation of a channel.

SYNOPSIS:

```
VOID L43_ChPause(channel, hCard)
INT channel              Number of the channel
HCARD hCard              Handle of Lx429-3 card
```

RETURNS: None.

DESCRIPTION: Pauses the operation of the channel specified by *channel*. All activity on the channel ceases. The channel remains paused until the channel is resumed by L43_ChResume, or the board is restarted by L43_CardStart or L43_CardResume.

Note: A transmit channel can also be paused when the Lx429-3 encounters a Pause Command Block in the transmit schedule.

WARNINGS: None.

SEE ALSO: L43_ChResume, L43_SchedPause

ChResume .... Resume the operation of a channel.

SYNOPSIS: VOID L43_ChResume(channel, hCard)

INT channel	Number of the channel
HCARD hCard	Handle of Lx429-3 card

RETURNS: None.

DESCRIPTION: Resumes the operation of the channel specified by *channel* after it has been paused by L43_ChPause, or the Lx429-3 has encountered a Pause Command Block in the transmit schedule. If the card has been started, all activity on the channel will begin. If the card is stopped, channel activity will begin when the card is started.

WARNINGS: None.

SEE ALSO: L43_ChPause

ExtDIORd .... Read the value of a digital I/O pin.

SYNOPSIS: BOOL L43_ExtDIORd(dionum, hCard)

INT dionum	Specifies DIO number
HCARD hCard	Handle of Lx429-3 card

RETURNS: Status of the digital I/O pin. Returns a zero if the pin is low (0V), or a one if the pin is high (5V).

DESCRIPTION: Reads the status of the digitail I/O pin specifed by *dionum*. There are four digital I/O pins on the Lx429-3, numbered 1-4.

WARNINGS: None.

SEE ALSO: L43_ExtDIOWr

ExtDIOWr .... Set the value of a digital I/O pin.

SYNOPSIS: VOID L43_ExtDIOWr(dionum, dioval, hCard)

INT dionum	Specifies DIO number
BOOL dioval	The value to set
HCARD hCard	Handle of Lx429-3 card

RETURNS: None.

DESCRIPTION: Sets the digital I/O pin specified by *dionum* to the value specified by *dioval.* A *dioval* of zero sets the pin low (0V), and a *dioval* of one sets the pin high (5V).

WARNINGS: When using the digital I/O as an output (as this function does), do not drive the digital I/O pin from an external source as this may damage the board.

SEE ALSO: L43_ExtDIORd

ExtLEDRd .... Read the value of the LED.

SYNOPSIS: BOOL L43_ExtLEDRd(hCard)

HCARD hCard   Handle of Lx429-3 card

RETURNS: Returns a zero if the LED is off, or a one if the LED is on.

DESCRIPTION: Reads the status of the on-board LED. The LED can be seen above the 25-pin D-Sub connector at the bracket end of the card.

WARNINGS: None.

SEE ALSO: L43_ExtLEDWr

ExtLEDWr ....Sets the value of the LED.

SYNOPSIS: VOID L43_ExtLEDWr(ledval, hCard)

BOOL ledval	The value to set
HCARD hCard	Handle of Lx429-3 card

RETURNS: None.

DESCRIPTION: Sets the state of the on-board LED. An *ledval* of zero sets the LED off, and an *ledval* of one sets the LED on. The LED can be seen above the 25-pin D-Sub connector at the bracket end of the card.

WARNINGS: None.

SEE ALSO: L43_ExtLEDWr

FilterDefault..................... Creates a default filter for a receive channel.

SYNOPSIS: MSGADDR L43_FilterDefault(ctrlflags, channel, hCard)

ULONG ctrlflags	Selects message options
INT channel	Number of receive channel
HCARD hCard	Handle of Lx429-3 card

RETURNS: Address of the message the function created and placed in the filter table.

DESCRIPTION: Creates a message record with the options specified in *ctrlflags*, and then sets it as the default message record for the channel specified by *channel*. Received messages that do not meet the criteria of specific filters are saved in the default message record. When a message does not match a specific filter and no default filter has been created, the message is skipped and not saved in memory.

The options that can be used in *ctrlflags* are listed below. Please note that only the receiver options can be used with this function.

	ctrlflags		
Constant	**Description**	**Rcv**	**Xmt**
MSGCRT_DEFAULT	Select all default settings (**bold** below)	✓	✓
MSGCRT_NOSEQ	This message will not get recorded in the sequential record	✓	✓
MSGCRT_SEQ	This message will get recorded in the sequential record	✓	✓
MSGCRT_NOINT	This message will not generate an interrupt	✓	✓
MSGCRT_INT	This message will generate an interrupt	✓	✓
MSGCRT_NOTIMETAG	This message will not record a time tag	✓	✓
MSGCRT_TIMETAG	This message will record a time tag	✓	✓
MSGCRT_NOELAPSE	This message will not record an elapsed time	✓	✓
MSGCRT_ELAPSE	This message will record an elapsed time	✓	✓
MSGCRT_NOMAXMIN	This message will not record maximum and minimum repetition rates	✓	✓
MSGCRT_MAX	This message will record maximum repetition rates	✓	✓
MSGCRT_MIN	This message will record minimum repetition rates	✓	✓
MSGCRT_MAXMIN	This message will record maximum and minimum repetition rates	✓	✓
MSGCRT_NOHIT	This message will not record a hit counter	✓	✓
MSGCRT_HIT	This message will record a hit counter (can't have hit counter with time tag, elapse, or max/min timing)	✓	✓
MSGCRT_NOSKIP	This message will not be skipped	✓	✓
MSGCRT_SKIP	This message will be skipped, and none of the options will be processed	✓	✓
MSGCRT_NOSYNC	This message will not generate a SYNCOUT signal		✓
MSGCRT_SYNC	This message will generate a SYNCOUT signal		✓

ctrlflags (continued)			
Constant	**Description**	**Rcv**	**Xmt**
MSGCRT_NOEXTRIG	This message will be triggered immediately		✓
MSGCRT_EXTRIG	This message will wait for an EXTRIG* pulse to be triggered		✓
MSGCRT_NOERR	This message will not have a parity error		✓
MSGCRT_PARERR	This message will have a parity error		✓
MSGCRT_WIPE	This data fields of this message will initially be wiped to a value	✓	✓
MSGCRT_NOWIPE	The data fields of this message will initially be left unchanged	✓	✓
MSGCRT_WIPE0	The data fields of this message will be wiped with a value of zeros (this option does not get used if MSGCRT_NOWIPE is used)	✓	✓
MSGCRT_WIPE1	The data fields of this message will be wiped with a value of ones (this option does not get used if MSGCRT_NOWIPE is used)	✓	✓
MSGCRT_CHAN	The memory device usage is determined by the channel configuration (L43_ChConfig)	✓	✓
MSGCRT_RAM	This message will reside in RAM	✓	✓
MSGCRT_DPRAM	This message will reside in Dual Port RAM	✓	✓

WARNINGS: This function initializes all filters (all label/SDI combinations) for the specified *channel*. Any filters previously created for the channel are overwritten. Therefore, L43_FilterDefault must be used BEFORE L43_FilterSet.

SEE ALSO: L43_FilterSet, L43_FilterRd, L43_FilterWr

FilterRd .... Read an entry from a filter table.

SYNOPSIS: MSGADDR L43_FilterRd(label, sdi, channel, hCard)

INT label	Label value
INT sdi	SDI pattern
INT channel	Number of receive channel
HCARD hCard	Handle of Lx429-3 card

RETURNS: Address of the message the filter table is set to for the given parameters.

DESCRIPTION: Reads the address of the message pointed to by the filter table for the specified *channel*, *label* and *sdi* values.

WARNINGS: This value reads the address of the message record that a filter is pointing to, not the ARINC 429 data word. Use L43_MsgDataRd to read the ARINC429 data from a message record.

SEE ALSO: L43_FilterWr, L43_FilterSet, L43_FilterDefault, L43_MsgDataRd

FilterSet.................................Creates a filter for a receive channel.

SYNOPSIS: MSGADDR L43_FilterSet(ctrlflags, label, sdi, channel, hCard)

ULONG ctrlflags	Selects message options
INT label	Label value to receive
INT sdi	SDI patterns to receive
INT channel	Number of receive channel
HCARD hCard	Handle of Lx429-3 card

RETURNS: Address of the message the function created and placed in the filter table.

DESCRIPTION: Filters are used to sort and save (by label and SDI) messages that are received over a given databus. During operation, when the label and SDI in a received message match the label and SDI information in a specific filter, the message is stored in a specific message record location. This function creates a message record with the options specified in *ctrlflags*, and then sets it as the message record for the specified *channel*, *label*, and *sdi* values.

sdi allows one or more SDI combinations to be specified. A filter is created for each specified SDI. The predefined constants listed below can be used to specify the SDI. When a combination of SDIs are selected, the constants should be ORed together.

sdi	
Constants	**Value**
SDI00	1
SDI01	2
SDI10	4
SDI11	8
SDIALL	15

The options that can be used in *ctrlflags* are listed below. Please note that only the receiver options can be used with this function.

ctrlflags			
Constant	**Description**	**Rcv**	**Xmt**
MSGCRT_DEFAULT	Select all default settings (**bold** below)	✔	✔
MSGCRT_NOSEQ	This message will not get recorded in the sequential record	✔	✔
MSGCRT_SEQ	This message will get recorded in the sequential record	✔	✔
MSGCRT_NOINT	This message will not generate an interrupt	✔	✔
MSGCRT_INT	This message will generate an interrupt	✔	✔
MSGCRT_NOTIMETAG	This message will not record a time tag	✔	✔
MSGCRT_TIMETAG	This message will record a time tag	✔	✔

ctrlflags (continued)			
Constant	**Description**	**Rcv**	**Xmt**
MSGCRT_NOELAPSE	This message will not record an elapsed time	✔	✔
MSGCRT_ELAPSE	This message will record an elapsed time	✔	✔
MSGCRT_NOMAXMIN	This message will not record maximum and minimum repetition rates	✔	✔
MSGCRT_MAX	This message will record maximum repetition rates	✔	✔
MSGCRT_MIN	This message will record minimum repetition rates	✔	✔
MSGCRT_MAXMIN	This message will record maximum and minimum repetition rates	✔	✔
MSGCRT_NOHIT	This message will not record a hit counter	✔	✔
MSGCRT_HIT	This message will record a hit counter (can't have hit counter with time tag, elapse, or max/min timing)	✔	✔
MSGCRT_NOSKIP	This message will not be skipped	✔	✔
MSGCRT_SKIP	This message will be skipped, and none of the options will be processed	✔	✔
MSGCRT_NOSYNC	This message will not generate a SYNCOUT signal		✔
MSGCRT_SYNC	This message will generate a SYNCOUT signal		✔
MSGCRT_NOEXTRIG	This message will be triggered immediately		✔
MSGCRT_EXTRIG	This message will wait for an EXTRIG* pulse to be triggered		✔
MSGCRT_NOERR	This message will not have a parity error		✔
MSGCRT_PARERR	This message will have a parity error		✔
MSGCRT_WIPE	This data fields of this message will initially be wiped to a value	✔	✔
MSGCRT_NOWIPE	The data fields of this message will initially be left unchanged	✔	✔
MSGCRT_WIPE0	The data fields of this message will be wiped with a value of zeros (this option does not get used if MSGCRT_NOWIPE is used)	✔	✔
MSGCRT_WIPE1	The data fields of this message will be wiped with a value of ones (this option does not get used if MSGCRT_NOWIPE is used)	✔	✔
MSGCRT_CHAN	The memory device usage is determined by the channel configuration (L43_ChConfig)	✔	✔
MSGCRT_RAM	This message will reside in RAM	✔	✔
MSGCRT_DPRAM	This message will reside in Dual Port RAM	✔	✔

WARNINGS: None.

SEE ALSO: L43_FilterDefault, L43_FilterRd, L43_FilterWr

FilterWr .... Write an entry to a filter table.

SYNOPSIS: ERRVAL L43_FilterWr(message, label, sdi, channel, hCard)

MSGADDR message	Message address to write to filter
INT label	Label value
INT sdi	SDI pattern
INT channel	Number of receive channel
HCARD hCard	Handle of Lx429-3 card

RETURNS: A negative value if an error occurs, or zero if successful.

DESCRIPTION: Writes the message address of *message* into the filter table position specified by *channel*, *label*, and *sdi*.

This function is most useful to assign multiple labels to point to one message record. After calling L43_FilterSet for the first label of interest, this function could be called with the meassge address that was returned by L43_FilterSet for each of the remaining labels. This would point all the labels of interest to one message record.

WARNINGS: None.

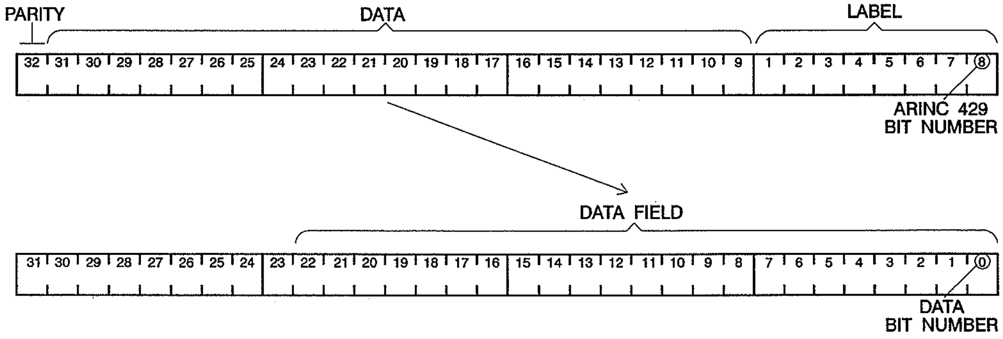
SEE ALSO: L43_FilterRd, L43_FilterSet, L43_FilterDefault

FldGetData...................... Extracts data field from a 32-bit ARINC 429 word.

SYNOPSIS: ULONG L43_FldGetData(msg)
ULONG msg 32-bit ARINC 429 word

RETURNS: The 23-bit data field of an ARINC 429 word.

DESCRIPTION: Extracts the 23-bit data field of the ARINC 429 word in *msg*. The extracted 23-bit data field is right shifted and zero filled as shown below:



Note: This is a utility function and does not access any Lx429-3 hardware.

WARNINGS: None.

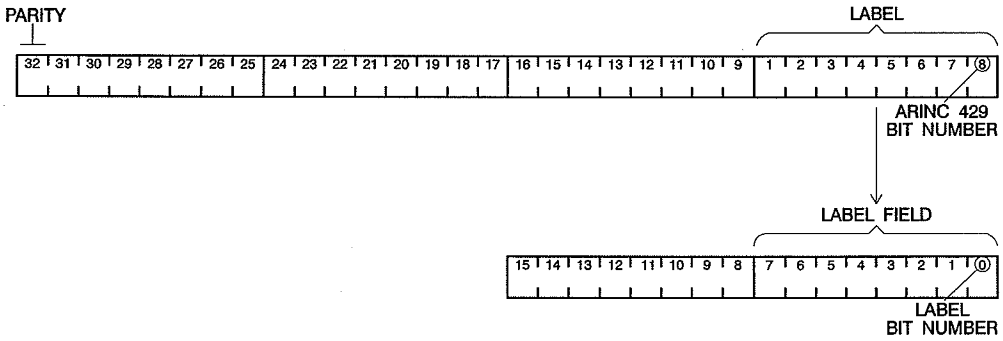
SEE ALSO: L43_FldPutData.

FldGetLabel.............Extracts the label field from a 32-bit ARINC 429 word.

SYNOPSIS: ULONG L43_FldGetLabel(msg)
ULONG msg 32-bit ARINC 429 word

RETURNS: The 8-bit label field of an ARINC 429 word.

DESCRIPTION: Extracts the 8-bit label field of the ARINC 429 word in *msg*. The extracted 8-bit label field and zero filled as shown below:



Note: This is a utility function and does not access any Lx429-3 hardware.

WARNINGS: None.

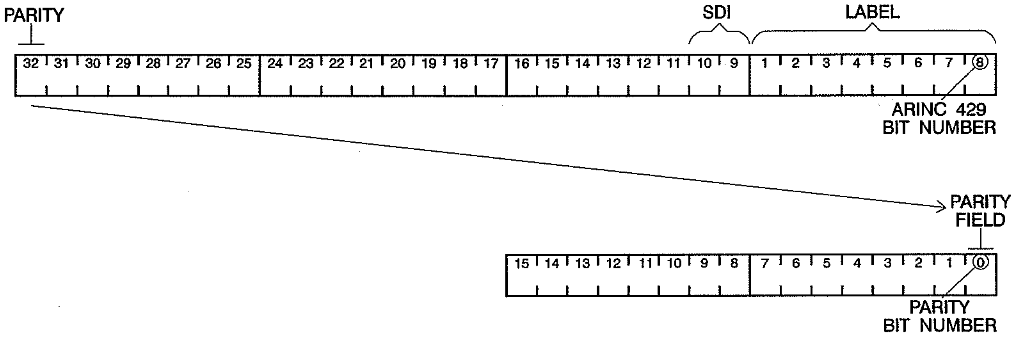
SEE ALSO: L43_FldPutLabel.

FldGetParity ....Extracts the parity bit from a 32-bit ARINC 429 word.

SYNOPSIS: USHORT L43_FldGetParity(msg)
ULONG msg 32-bit ARINC 429 word

RETURNS: The parity bit of an ARINC 429 word.

DESCRIPTION: Extracts the parity bit of the ARINC 429 word in *msg*. The extracted parity bit is right shifted and zero filled as shown below:



Note: This is a utility function and does not access any Lx429-3 hardware.

WARNINGS: None.

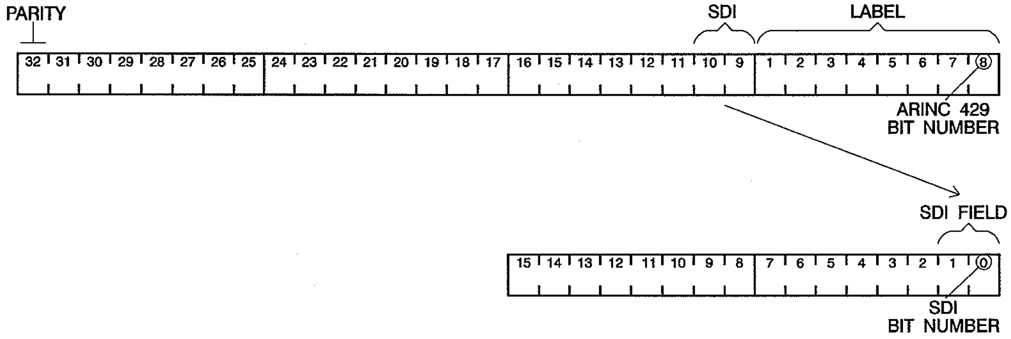
SEE ALSO: L43_FldGetData.

FldGetSDI .... Extracts the SDI field from a 32-bit ARINC 429 word.

SYNOPSIS: USHORT L43_FldGetSDI(msg)
ULONG msg 32-bit ARINC 429 word

RETURNS: The 2-bit SDI field of an ARINC 429 word.

DESCRIPTION: Extracts the 2-bit SDI field of the ARINC 429 word in *msg*. The extracted SDI field is right shifted and zero filled as shown below:



Note: This is a utility function and does not access any Lx429-3 hardware.

WARNINGS: None.

SEE ALSO: L43_FldPutSDI.

FldGetValue ....Gets value of specified field of ARINC 429 word.

SYNOPSIS: ULONG L43_FldGetValue(msg, startbit, endbit)

ULONG msg	32-bit ARINC 429 word
USHORT startbit	Starting bit number of BCD field
USHORT endbit	Ending bit number of BCD field

RETURNS: The specified field of an ARINC 429 word.

DESCRIPTION: Extracts a bit field from the ARINC 429 word in *msg*. *startbit* and *endbit* determine the lowest and highest bit position of the field to extract. The extracted field is right shifted and zero filled.

Note: This is a utility function and does not access any Lx429-3 hardware.

WARNINGS: None.

SEE ALSO: L43_FldPutValue.

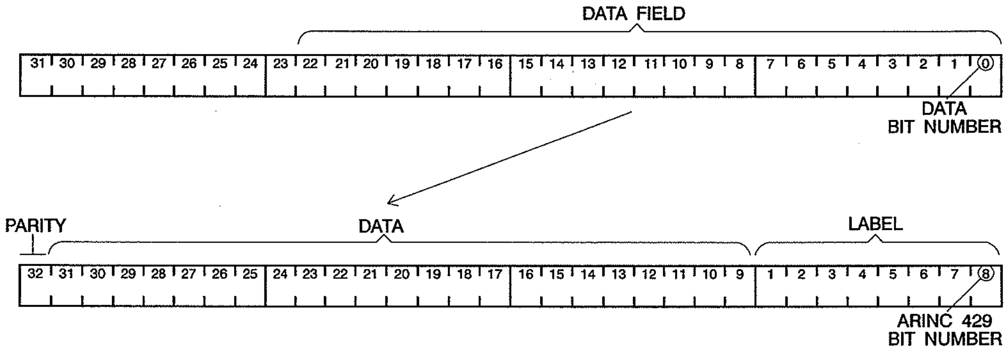
FldPutData .... Inserts the data field into an ARINC 429 word.

SYNOPSIS: ULONG L43_FldPutData(msg, data)

ULONG msg	32-bit ARINC 429 word
ULONG data	New 23-bit data field value

RETURNS: The new 32-bit ARINC 429 word with the data field inserted.

DESCRIPTION: Inserts a 23-bit data field value into the ARINC 429 word in *msg*. *data* is left shifted and packed into *msg* as shown below:



Note: This is a utility function and does not access any Lx429-3 hardware.

WARNINGS: None.

SEE ALSO: L43_FldGetData.

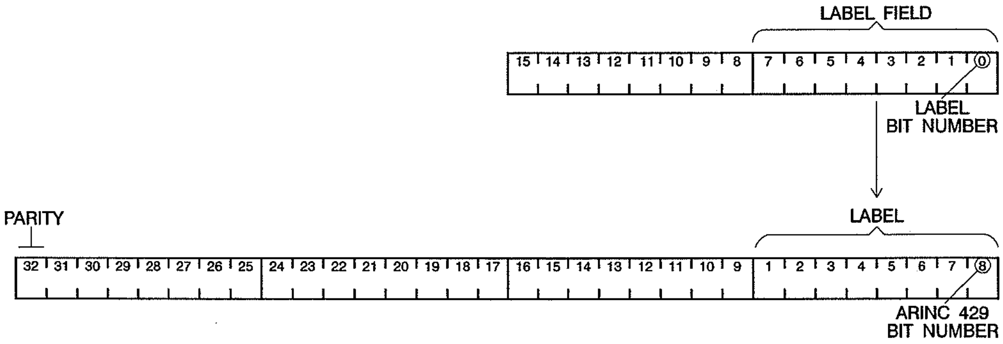
FldPutLabel ....Inserts the label field into an ARINC 429 word.

SYNOPSIS: ULONG L43_FldPutLabel(msg, label)

ULONG msg	32-bit ARINC 429 word
USHORT label	New 8-bit label field value

RETURNS: The new 32-bit ARINC 429 word with the *label* field inserted.

DESCRIPTION: Inserts an 8-bit *label* field value into the ARINC 429 word in *msg*. *label* is packed into *msg* as shown below:



Note: This is a utility function and does not access any Lx429-3 hardware.

WARNINGS: None.

SEE ALSO: L43_FldGetLabel.

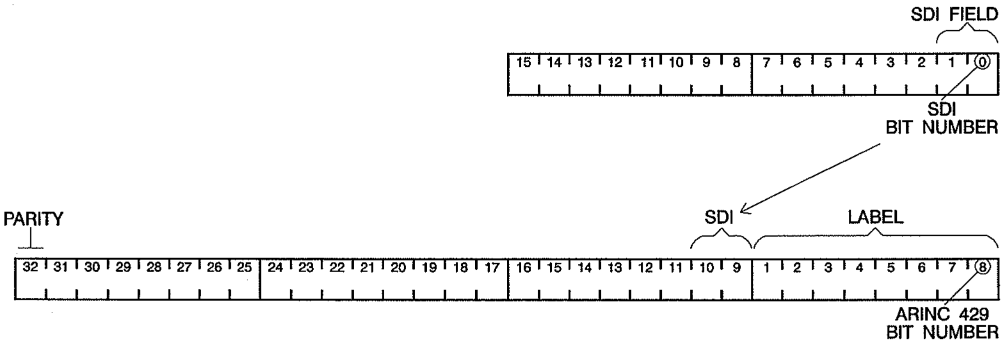
FldPutSDI .... Inserts the SDI field into a ARINC 429 word.

SYNOPSIS: ULONG L43_FldPutSDI(msg, sdi)

ULONG msg	32-bit ARINC 429 word
USHORT sdi	New 2-bit SDI field value

RETURNS: The new 32-bit ARINC 429 word with the SDI field inserted.

DESCRIPTION: Inserts a 2-bit SDI field value into the ARINC 429 word in *msg*. *sdi* is left shifted and packed into *msg* as shown below:



Note: This is a utility function and does not access any Lx429-3 hardware.

WARNINGS: None.

SEE ALSO: L43_FldGetSDI.

FldPutValue ....Inserts the specified field into a ARINC 429 word.

SYNOPSIS: ULONG L43_FldPutValue(msg, data, startbit, endbit)

ULONG msg	32-bit ARINC 429 word
ULONG data	New 23-bit data field value
USHORT startbit	Starting bit position of field
USHORT endbit	Ending bit position of field

RETURNS: The new 32-bit ARINC 429 word with the specified field inserted.

DESCRIPTION: Inserts a bit field value into the ARINC 429 word in *msg*. *startbit* and *endbit* specify the low and high bit positions of the field.

Note: This is a utility function and does not access any Lx429-3 hardware.

WARNINGS: None.

SEE ALSO: L43_FldGetValue.

GetChanCount..Gets the board channel count.

SYNOPSIS: VOID L43_GetChanCount(rcvcount, xmtcount, hCard)

LPINT rcvcount	Pointer to variable to hold receiver count
LPINT xmtcount	Pointer to variable to hold transmitter count
HCARD hCard	Handle of Lx429-3 card

RETURNS: None.

DESCRIPTION: Determines the transmitter and receiver channel count, and puts them in the variables pointed to by *rcvcount* and *xmtcount*.

WARNINGS: None.

SEE ALSO: L43_IsRcvChan, L43_IsXmtChan

IntConfig .... Configures the interrupt log list.

SYNOPSIS: ERRVAL L43_IntConfig(ctrlflags, count, hCard)

USHORT ctrlflags	Selects the configuration options
USHORT count	Number of entries in the interrupt log list
HCARD hCard	Handle of Lx429-3 card

RETURNS: A negative value if an error occurs, or zero if successful.

DESCRIPTION: Enables the interrupt log list of the Lx429-3 card. *count* sets the number of entries in the card's interrupt log list. *count* must be a positive value specified by the user. *ctrlflags* can be one of the following constants.

ctrlflags	
Constant	**Condition**
INTCFG_DEFAULT	Select all default **(bold)** settings
INTCFG_ENABLE	Enable interrupt log list
INTCFG_DISABLE	Disable interrupt log list

WARNINGS: This function does not enable interrupts to the host. The interrupt log list can be polled by an application without installing an interrupt handler.

SEE ALSO: L43_IntInstall

IntInstall (WIN32) .... Installs an interrupt handler under WIN32.

SYNOPSIS: ERRVAL L43_IntInstall(hEvent, hCard)

LPVOID hEvent	Handle of a WIN32 event object
HCARD hCard	Handle of Lx429-3 card

RETURNS: A negative value if an error occurs, or zero if successful.

DESCRIPTION: Associates a WIN32 event object with interrupts from the card specified by the handle. If the function is successful, any interrupt issued from the Lx429-3 causes the event object specified by *hEvent* to be set to the signaled state.

The user's application must ensure that the event object is set to the unsignaled state before the Lx429-3 issues the first interrupt. This can be done when creating the event object with the WIN32 API function CreateEvent.

Note: Event objects are never polled. Create a worker thread which immediately goes to sleep by calling a WIN32 API wait function like WaitForSingleObject. When the Lx429-3 issues an interrupt, the event object is signaled, and the worker thread wakes up to respond to the interrupt.

It is the user's responsibility to clear the interrupt from the Lx429-3 by calling L43_IntReset. The user must also set the event object to the unsignaled state.

WARNINGS: If this function is used, L43_IntUninstall MUST be called before the user's program terminates. It removes the association between the Lx429-3 and the event object.

SEE ALSO: L43_IntRd, L43_IntUninstall

IntRd .... Reads the next entry from the interrupt log list.

SYNOPSIS: ULONG L43_IntRd(typeval, infoval, hCard)

LPUSHORT typeval	Pointer to variable to receive type value
LPULONG infoval	Pointer to variable to receive info value
HCARD hCard	Handle of Lx429-3 card

RETURNS: The address of the entry in the interrupt log list, or zero if it is empty and there are no entries to read.

DESCRIPTION: Reads a single entry from the interrupt log list. Each entry contains two values – the type of interrupt that generated the entry, and an information word associated with the interrupt. The type of interrupt is passed through *typeval*, and the information word is passed through *infoval.*

typeval is a 16-bit value with the following two fields:

Board Number				Channel Number				Type Value							
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0

The value returned in *infoval* depends upon the *typeval* values listed below.

typeval	Description	*infoval*
INTTYPE_MSG	The interrupt option was enabled on a message that was processed	Address of the message record
INTTYPE_OPCODE	An interrupt was encountered in a transmit schedule	Tag value passed in L43_SchedInt()
INTTYPE_HALT	A transmit schedule encountered a halt command	Address of the schedule entry
INTTYPE_PAUSE	A transmit schedule encountered a pause command	Address of the schedule entry
INTTYPE_SEQFULL	The sequential record is full	Address of the last sequential record entry
INTTYPE_LIST	A list buffer was full or empty.	Address of the list buffer
INTTYPE_ERR	A decoder error was detected	Address of the message that contained the error
INTTYPE_SEQFREQ	The sequential record reached the interrupt frequency value	Address of the last sequential record entry
INTTYPE_717WORD	A specific 717 word was encountered	Address of the word record
INTTYPE_717SUBFRM	A specific 717 subframe was encountered	Subframe number
INTTYPE_717ERRSYNC	A 717 receive channel lost sync	Channel number

Note: *typeval* must each be large enough to store a 16-bit word, and *infoval* must be large enough to store a 32-bit word. If *typeval* or *infoval* is a NULL pointer, that parameter is not returned.

WARNINGS: This function should be preceded by a call to L43_IntConfig. To use this function, it is not necessary to install an interrupt handler.

SEE ALSO: L43_IntConfig, L43_IntInstall

IntUninstall .... Removes an interrupt handler.

SYNOPSIS: ERRVAL L43_IntUninstall(hCard)
HCARD hCard Handle of Lx429-3 card

RETURNS: A negative value if an error occurs, or zero if successful.

DESCRIPTION: Removes an interrupt handler installed by the L43_IntInstall function. The Lx429-3's interrupt log list remains unchanged.

WARNINGS: This function must be called before the user's application terminates if L43_IntInstall has been called.

SEE ALSO: L43_IntConfig, L43_IntInstall

IsRcvChan ....Checks for a receive channel.

SYNOPSIS: BOOL L43_IsRcvChan(channel, hCard)

INT channel	Channel number to test
HCARD hCard	Handle of Lx429-3 card

RETURNS: A non-zero value if the channel is a receiver, or zero if it is not a reciever.

DESCRIPTION: Checks to see if the channel number specified by *channel* is a receive channel.

WARNINGS: Do not assume that if this function returns a value of zero that the channel must then be a transmitter, because the channel may not exist at all. A call to L43_IsXmtChan must be made to be sure that the channel is a transmitter.

SEE ALSO: L43_IsXmtChan, L43_GetChanCount

IsXmtChan .... Checks for a transmit channel.

SYNOPSIS: BOOL L43_IsXmtChan(channel, hCard)

INT channel	Channel number to test
HCARD hCard	Handle of Lx429-3 card

RETURNS: A non-zero value if the channel is a transmitter, or zero if it is not a transmitter.

DESCRIPTION: Checks to see if the channel number specified by *channel* is a transmit channel.

WARNINGS: Do not assume that if this function returns a value of zero that the channel must then be a receiver, because the channel may not exist at all. A call to L43_IsRcvChan must be made to be sure that the channel is a receiver.

SEE ALSO: L43_IsRcvChan, L43_GetChanCount

ListAsyncCreate .... Creates an asynchronous transmit list buffer.

SYNOPSIS: LISTADDR L43_ListAsyncCreate(ctrlflags, count, channel, hCard)

ULONG ctrlflags	Selects list options
INT count	Number of entries in list
INT channel	Channel number to associate with the list buffer
HCARD hCard	Handle of Lx429-3 card

RETURNS: The Lx429-3 address of the list if successful, otherwise zero.

DESCRIPTION: Creates an asynchronous transmit list buffer the size of *count* entries. The list buffer is connected with the channel specified by *channel.* Every time a gap is encountered in a transmit schedule, if data exists in the list it will be transmitted during the gap time. The *ctrlflags* specifies what type of list buffer and the options associated with the list buffer. Only FIFO mode is valid for an asynchronous list buffer.

ctrlflags	
Constant	**Description**
LISTCRT_DEFAULT	Select all default settings (**bold** below)
LISTCRT_FIFO	Selects FIFO mode
LISTCRT_PINGPONG	Selects ping-pong mode
LISTCRT_CIRCULAR	Selects circular mode
LISTCRT_NOINT	An interrupt will not be issued when the list buffer is empty/full
LISTCRT_INT	An interrupt will be issued when the list buffer is empty/full
LISTCRT_RAM	List is created in static RAM
LISTCRT_DPRAM	List is created in dual-port RAM

WARNINGS: After connecting a message record with a list buffer, the functions L43_ListDataRd and L43_ListDataWr must be used to read the data. Do not use L43_MsgDataRd and L43_MsgDataWr as they will return incorrect results.

SEE ALSO: L43_ListRcvCreate, L43_ListAsyncCreate, L43_ListDataWr, L43_ListDataRd

ListDataRd .... Reads the next data associated with a list.

SYNOPSIS: ULONG L43_ListDataRd(list, hCard)

LISTADDR list	List from which to read data
HCARD hCard	Handle of Lx429-3 card

RETURNS: The 32-bit value of the ARINC word if there is a word in the list, or zero if it is empty.

DESCRIPTION: Similar to L43_MsgDataRd except it reads from a list. This function reads one message from the list, and automatically updates the pointers to point to the next message. The *list* parameter is the value returned when the list was created using L43_ListRcvCreate. The position of the message to be read is determined by the mode of the list as follows:

CIRCULAR MODE: Not valid for a receive list buffer.

FIFO MODE: Reads the oldest complete message received.

PINGPONG MODE: Reads the newest complete message received.

WARNINGS: *list* must be configured as a receive list buffer using L43_ListRcvCreate.

SEE ALSO: L43_MsgDataRd, L43_ListDataWr, L43_ListRcvCreate, L43_ListXmtCreate,L43_ListAsyncCreate

ListXmtCreate .... Creates a transmit list buffer.

SYNOPSIS: LISTADDR L43_ListXmtCreate(ctrlflags, count, message, hCard)

ULONG ctrlflags	Selects list options
INT count	Number of entries in list
MSGADDR message	Message record address to associate with the list buffer
HCARD hCard	Handle of Lx429-3 card

RETURNS: The Lx429-3 address of the list if successful, otherwise zero.

DESCRIPTION: Creates a transmit list buffer the size of *count* entries. The list buffer is connected with a message record so that the data is processed in the list instead of in the messag record. The *ctrlflags* specifies what type of list buffer and the options associated with the list buffer.

ctrlflags	
Constant	**Description**
LISTCRT_DEFAULT	Select all default settings (**bold** below)
LISTCRT_FIFO	Selects FIFO mode
LISTCRT_PINGPONG	Selects ping-pong mode
LISTCRT_CIRCULAR	Selects circular mode
LISTCRT_NOINT	An interrupt will not be issued when the list buffer is empty/full
LISTCRT_INT	An interrupt will be issued when the list buffer is empty/full
LISTCRT_RAM	List is created in static RAM
LISTCRT_DPRAM	List is created in dual-port RAM

WARNINGS: After connecting *message* with a list buffer, the functions L43_ListDataRd and L43_ListDataWr must be used to read the data. Do not use L43_MsgDataRd and L43_MsgDataWr as they will return incorrect results.

SEE ALSO: L43_ListRcvCreate, L43_ListAsyncCreate, L43_ListDataWr, L43_ListDataRd

MsgBlockRd .... Reads an entire message record from the Lx429-3.

SYNOPSIS: MSGADDR L43_MsgBlockRd(structptr, message, hCard)

LPMSGFIELDS structptr	Pointer to destination structure
MSGADDR message	Message from which to read
HCARD hCard	Handle of Lx429-3 card

RETURNS: The message record address that was read.

DESCRIPTION: Reads an entire message record from the Lx429-3.

MSGFIELDS structure		
Field	**Size**	**Description**
msgopt	USHORT	Message options fields - do not modify these fields.
msgact	USHORT	Message activity - see table below for detail.
msgdata	ULONG	Message data value - 32-bit ARINC data word value.
listptr	ULONG	List buffer pointer - used instead of msgdata when in list buffer mode.
timetag	ULONG	Time tag value - 32-bits with resolution set by L43_TimerResolution.
hitcount	ULONG	Hit counter value - used instead of timetag when in hit counter mode.
maxtime	ULONG	Maximum repetition rate - 32-bits with resolution equal to time tag resolution.
Elspsetime	ULONG	Elapsed time - 32-bits with resolution equal to time tag resolution. Used instead of maxtime when in elapsed time mode.
mintime	ULONG	Minimum repetition rate - 32-bits with resolution equal to time tag resolution.
userptr	ULONG	Reserved
miscptr	ULONG	Reserved
rsvd	ULONG	Reserved

The msgact field may be tested by AND-ing the values returned with the constants from the following table.

msgact field	
Constant	**Description**
MSGACT_BRD	The board number (bits 12-15).
MSGACT_ CHAN	The channel number (bits 8-11).
MSGACT_ ASPD	Decoder auto speed detected. When a receiver is in auto speed detection mode, this bit reflects the speed detected. A one signifies high speed, and a zero signifies low speed.
MSGACT_ ERR	If set, it signifies that an error occurred in receiving this word. The type of error is defined by the following bits.
MSGACT_ GAP	Gap Error - a gap of less than four bit times preceded the word.
MSGACT_ PAR	Parity error - a parity error was detected in the word.
MSGACT_ LONG	Long word error - a word of more than 32-bits was detected.
MSGACT_ BIT	Bit timing error - an error occurred while decoding the bits of the word (short bits or long bits).
MSGACT_ TO	Time out error - the decoder timed out while receiving a word (short word).
MSGACT_ HIT	Signifies that the message has been processed by the firmware (the "hit bit").

WARNINGS: None.

SEE ALSO: L43_MsgBlockWr, L43_MsgDataRd, L43_MsgDataWr

MsgBlockWr.......................Writes an entire message record to the Lx429-3.

SYNOPSIS: MSGADDR L43_MsgBlockWr(structptr, message, hCard)

LPMSGFIELDS structptr	Pointer to source structure
MSGADDR message	Message to write to
HCARD hCard	Handle of Lx429-3 card

RETURNS: The message record address that was written to.

DESCRIPTION: Writes an entire message record to the Lx429-3.

MSGFIELDS structure		
Field	**Size**	**Description**
msgopt	USHORT	Message options fields - do not modify these fields.
msgact	USHORT	Message activity - see table below for detail.
msgdata	ULONG	Message data value - 32-bit ARINC data word value.
listptr	ULONG	List buffer pointer - used instead of msgdata when in list buffer mode.
timetag	ULONG	Time tag value - 32-bits with resolution set by L43_TimerResolution.
hitcount	ULONG	Hit counter value - used instead of timetag when in hit counter mode.
maxtime	ULONG	Maximum repetition rate - 32-bits with resolution equal to time tag resolution.
elspsetime	ULONG	Elapsed time - 32-bits with resolution equal to time tag resolution. Used instead of maxtime when in elapsed time mode.
mintime	ULONG	Minimum repetition rate - 32-bits with resolution equal to time tag resolution.
userptr	ULONG	Reserved
miscptr	ULONG	Reserved
rsvd	ULONG	Reserved

The msgact field may be tested by AND-ing the values returned with the constants from the following table.

msgact field	
Constant	**Description**
MSGACT_BRD	The board number (bits 12-15).
MSGACT_ CHAN	The channel number (bits 8-11).
MSGACT_ ASPD	Decoder auto speed detected. When a receiver is in auto speed detection mode, this bit reflects the speed detected. A one signifies high speed, and a zero signifies low speed.
MSGACT_ ERR	If set, it signifies that an error occurred in receiving this word. The type of error is defined by the following bits.
MSGACT_ GAP	Gap Error - a gap of less than four bit times preceded the word.
MSGACT_ PAR	Parity error - a parity error was detected in the word.
MSGACT_ LONG	Long word error - a word of more than 32-bits was detected.
MSGACT_ BIT	Bit timing error - an error occurred while decoding the bits of the word (short bits or long bits).
MSGACT_ TO	Time out error - the decoder timed out while receiving a word (short word).
MSGACT_ HIT	Signifies that the message has been processed by the firmware (the "hit bit").

WARNINGS: None.

SEE ALSO: L43_MsgBlockRd, L43_MsgDataRd, L43_MsgDataWr

MsgCreate............................Creates and initializes a message record.

SYNOPSIS: MSGADDR L43_MsgCreate(ctrlflags, hCard)
ULONG ctrlflags Selects message options
HCARD hCard Handle of Lx429-3 card

RETURNS: The Lx429-3 address of the message record if successful, otherwise zero.

DESCRIPTION: Allocates memory for a message record and initializes the record with the options specified in *ctrlflags*.

The options that can be used in *ctrlflags* are listed below. Please note that only the transmitter options can be used with this function.

ctrlflags

Constant	Description	Rcv	Xmt
MSGCRT_DEFAULT	Select all default settings (**bold** below)	✓	✓
MSGCRT_NOSEQ	This message will not get recorded in the sequential record	✓	✓
MSGCRT_SEQ	This message will get recorded in the sequential record	✓	✓
MSGCRT_NOINT	This message will not generate an interrupt	✓	✓
MSGCRT_INT	This message will generate an interrupt	✓	✓
MSGCRT_NOTIMETAG	This message will not record a time tag	✓	✓
MSGCRT_TIMETAG	This message will record a time tag	✓	✓
MSGCRT_NOELAPSE	This message will not record an elapsed time	✓	✓
MSGCRT_ELAPSE	This message will record an elapsed time	✓	✓
MSGCRT_NOMAXMIN	This message will not record maximum and minimum repetition rates	✓	✓
MSGCRT_MAX	This message will record maximum repetition rates	✓	✓
MSGCRT_MIN	This message will record minimum repetition rates	✓	✓
MSGCRT_MAXMIN	This message will record maximum and minimum repetition rates	✓	✓
MSGCRT_NOHIT	This message will not record a hit counter	✓	✓
MSGCRT_HIT	This message will record a hit counter (can't have hit counter with time tag, elapse, or max/min timing)	✓	✓
MSGCRT_NOSKIP	This message will not be skipped	✓	✓
MSGCRT_SKIP	This message will be skipped, and none of the options will be processed	✓	✓
MSGCRT_NOSYNC	This message will not generate a SYNCOUT signal		✓
MSGCRT_SYNC	This message will generate a SYNCOUT signal		✓
MSGCRT_NOEXTRIG	This message will be triggered immediately		✓
MSGCRT_EXTRIG	This message will wait for an EXTRIG* pulse to be triggered		✓
MSGCRT_NOERR	This message will not have a parity error		✓
MSGCRT_PARERR	This message will have a parity error		✓

ctrlflags (continued)			
Constant	Description	Rcv	Xmt
MSGCRT_WIPE	This data fields of this message will initially be wiped to a value	✓	✓
MSGCRT_NOWIPE	The data fields of this message will initially be left unchanged	✓	✓
MSGCRT_WIPE0	The data fields of this message will be wiped with a value of zeros (this option does not get used if MSGCRT_NOWIPE is used)	✓	✓
MSGCRT_WIPE1	The data fields of this message will be wiped with a value of ones (this option does not get used if MSGCRT_NOWIPE is used)	✓	✓
MSGCRT_CHAN	The memory device usage is determined by the channel configuration (L43_ChConfig)	✓	✓
MSGCRT_RAM	This message will reside in RAM	✓	✓
MSGCRT_DPRAM	This message will reside in Dual Port RAM	✓	✓

WARNINGS: None.

SEE ALSO: L43_MsgDataRd, L43_MsgDataWr, L43_MsgBlockRd, L43_MsgBlockWr

MsgDataRd....................................Reads the data associated with a message.

SYNOPSIS: ULONG L43_MsgDataRd(message, hCard)

MSGADDR message	Message from which to read
HCARD hCard	Handle of Lx429-3 card

RETURNS: 32-bit value of the ARINC data word, or zero if no data has been received in the message record.

DESCRIPTION: Reads the 32-bit value of the ARINC data word from the message record specified by *message*.

WARNINGS: None.

SEE ALSO: L43_MsgDataWr, L43_MsgCreate, L43_MsgBlockRd, L43_MsgBlockWr

MsgDataWr .... Writes the data associated with a message.

SYNOPSIS: VOID L43_MsgDataWr(value, message, hCard)

ULONG value	Value of data to write to message
MSGADDR message	Message to receive new data
HCARD hCard	Handle of Lx429-3 card

RETURNS: None.

DESCRIPTION: Writes the 32-bit ARINC data value specified by *value* into the message record specified by *message*.

WARNINGS: None.

SEE ALSO: L43_MsgDataRd, L43_MsgCreate, L43_MsgBlockRd, L43_MsgBlockWr

SchedBranch .... Appends a conditional branch to the schedule.

SYNOPSIS: SCHNDX L43_SchedBranch(condition, destindex, channel, hCard)

USHORT condition	Condition for branch
SCHNDX destindex	Destination index for branch
INT channel	Channel number of transmitter
HCARD hCard	Handle of Lx429-3 card

RETURNS: The index of the appended command block if successful, or a negative value if an error occurs.

DESCRIPTION: Appends a conditional branch command block to the current end of the schedule. A conditional branch command block causes the Lx429-3 to branch to the index in the schedule specified by *destindex* if *condition* evaluates as TRUE.

The *condition* flags listed below may be used to specify the branch condition.

condition	
Constant	**Description**
COND_ALWAYS	Always branch.
COND_DIO1ACT	Branch if DIO1 is high (5V).
COND_DIO1NACT	Branch if DIO1 is low (0V).
COND_DIO2ACT	Branch if DIO2 is high (5V).
COND_DIO2NACT	Branch if DIO2 is low (0V).
COND_DIO3ACT	Branch if DIO3 is high (5V).
COND_DIO3NACT	Branch if DIO3 is low (0V).
COND_DIO4ACT	Branch if DIO4 is high (5V).
COND_DIO4NACT	Branch if DIO4 is low (0V).

WARNINGS: A call to L43_ChConfig must precede this function. When creating sub-routines, L43_SchedEntry needs to be called to point to the main section.

SEE ALSO: L43_SchedEntry, L43_SchedCall, L43_SchedReturn

SchedBuild .... Sequences messages by given intervals.

SYNOPSIS: ERRVAL L43_SchedBuild(nummsgs, msgs, min, max, channel, hCard)

INT nummsgs	Number of messages to schedule
LPMSGADDR msgs[]	Array of message addresses
LPINT min[]	Array of message frequencies
LPINT max[]	Array of message frequencies
INT channel	Channel number of transmitter
HCARD hCard	Handle of Lx429-3 card

RETURNS: Zero if the schedule was successfully built, or a negative value if an error occurred.

DESCRIPTION: Clears any transmit schedule for the specified *channel* and creates a new transmit schedule. The new schedule will consist of *nummsgs* messages, each transmitted within an interval specified by the *min* and *max* interval arrays. *msgs* points to an array of message addresses, each previously generated by a call to L43_MsgCreate. *min* and *max* point to arrays of intervals in units of milliseconds.

The nth element of the *msgs* array uses to the nth element of the *min* and *max* intervals arrays to create the schedule.

The function schedules messages and gaps to generate the specified transmit intervals. If the schedule cannot be generated, an error is returned.

WARNINGS: A call to L43_ChConfig must precede this function as well as a call to L43_MsgCreate for each message to be scheduled.

SEE ALSO: L43_SchedMsg, L43_SchedMsgEx, L43_SchedGap

SchedCall ....Appends a conditional call to the schedule.

SYNOPSIS: SCHNDX L43_SchedCall(condition, destindex, channel, hCard)

USHORT condition	Condition for branch
SCHNDX destindex	Destination index for branch
INT channel	Channel number of transmitter
HCARD hCard	Handle of Lx429-3 card

RETURNS: The index of the appended command block if successful, or a negative value if an error occurs.

DESCRIPTION: Appends a conditional call command block to the current end of the schedule. A conditional call command block causes the Lx429-3 to call the index in the schedule specified by *destindex* if *condition* evaluates as TRUE.

The *condition* flags listed below may be used to specify the call condition.

condition	
Constant	**Description**
COND_ALWAYS	Always call.
COND_DIO1ACT	Call if DIO1 is high (5V).
COND_DIO1NACT	Call if DIO1 is low (0V).
COND_DIO2ACT	Call if DIO2 is high (5V).
COND_DIO2NACT	Call if DIO2 is low (0V).
COND_DIO3ACT	Call if DIO3 is high (5V).
COND_DIO3NACT	Call if DIO3 is low (0V).
COND_DIO4ACT	Call if DIO4 is high (5V).
COND_DIO4NACT	Call if DIO4 is low (0V).

WARNINGS: A call to L43_ChConfig must precede this function. When creating sub-routines, L43_SchedEntry needs to be called to point to the main section. L43_SchedReturn must be called for every use of L43_SchedCall.

SEE ALSO: L43_SchedEntry, L43_SchedBranch, L43_SchedReturn

SchedEntry ....Sets the beginning of the transmit schedule.

SYNOPSIS: SCHNDX L43_SchedEntry(channel, hCard)

INT channel	Channel number of transmitter
HCARD hCard	Handle of Lx429-3 card

RETURNS: The index of the appended command block if successful, or a negative value if an error occurs.

DESCRIPTION: Sets the next availible location in the schedule as the beginning of the schedule. This operation is only necessary if sub-routines are used in a schedule.

A schedule with sub-routines would need to define the sub-routines first by calling the desired schedule functions, while saving the returned schedule indicies. Then, this function must be called to set the starting point of the schedule. Now the main part of the schedule can be built by calling the other schedule functions. Finally, a branch or call command would be used to jump into the sub-routines.

WARNINGS: A call to L43_ChConfig must precede this function.

SEE ALSO: L43_SchedBranch, L43_SchedCall, L43_SchedReturn

SchedGap......................... Appends a gap command block into the schedule.

SYNOPSIS:

```
SCHNDX L43_SchedGap(gap, channel, hCard)
USHORT gap          Gap value in bit times
INT channel         Channel number of transmitter
HCARD hCard         Handle of Lx429-3 card
```

RETURNS: The index of the appended command block if successful, or a negative value if an error occurs.

DESCRIPTION: Appends a gap command block to the current end of the schedule. When a gap command block is encountered in the schedule, it triggers the transmission of any preceeding message command block as well as the specified *gap* (in bit times) before the next message can be transmitted.

Depending on the configuration of an asynchronous list buffer, this function will internally call either L43_SchedGapFixed, or L43_SchedGapList. This allows the user to include or leave out an asynchronous list buffer without re-building the schedule.

WARNINGS: A call to L43_ChConfig must precede this function.

SEE ALSO: L43_SchedMsg, L43_SchedGapFixed, L43_SchedGapList

SchedGapFixed .... Appends a fixed gap command block into the schedule.

SYNOPSIS: SCHNDX L43_SchedGapFixed(gap, channel, hCard)

USHORT gap	Gap value in bit times
INT channel	Channel number of transmitter
HCARD hCard	Handle of Lx429-3 card

RETURNS: The index of the appended command block if successful, or a negative value if an error occurs.

DESCRIPTION: Appends a fixed gap command block to the current end of the schedule. When a gap command block is encountered in the schedule, it triggers the transmission of any preceeding message command block as well as the specified *gap* (in bit times) before the next message can be transmitted. The gap time is fixed so no asynchronous messages can be transmitted during this time.

Note: This is an advanced function, and for most applications the L43_SchedGap function is preferred.

WARNINGS: A call to L43_ChConfig must precede this function.

SEE ALSO: L43_SchedGap, L43_SchedGapList

SchedGapList Appends a conditional gap command block into the schedule.

SYNOPSIS: SCHNDX L43_SchedGapList(gap, list, channel, hCard)

USHORT gap	Gap value in bit times
LISTADDR list	Address of asynchronous list buffer
INT channel	Channel number of transmitter
HCARD hCard	Handle of Lx429-3 card

RETURNS: The index of the appended command block if successful, or a negative value if an error occurs.

DESCRIPTION: Appends a conditional gap command block to the current end of the schedule. A conditional gap command block specifies *gap* (in bit times) before the next scheduled message can be transmitted. During this gap time, if a messgae exists in the asynchronous list buffer, it is transmitted in the gap time.

Note: This is an advanced function, and for most applications the L43_SchedGap function is preferred.

WARNINGS: A call to L43_ChConfig must precede this function. In addition, an asynchronous list buffer must be configured using L43_ListAsyncCreate before calling this function.

SEE ALSO: L43_SchedGap, L43_SchedGapFixed

SchedHalt .... Appends a halt command block into the schedule.

SYNOPSIS: SCHNDX L43_SchedHalt(channel, hCard)

INT channel	Channel number of transmitter
HCARD hCard	Handle of Lx429-3 card

RETURNS: The index of the appended command block if successful, or a negative value if an error occurs.

DESCRIPTION: Appends a halt command block to the current end of the schedule. A halt command block stops the schedule until the card is restarted using L43_CardStart.

Note: Execution of this function does NOT halt the schedule. The schedule is halted only when the resulting command block is executed after L43_CardStart starts the Lx429-3.

WARNINGS: A call to L43_ChConfig must precede this function.

SEE ALSO: L43_SchedPause, L43_SchedRestart

SchedInt ....Appends a conditional interrupt to the schedule.

SYNOPSIS: SCHNDX L43_SchedInt(condition, tagval, channel, hCard)

USHORT condition	Condition for branch
USHORT tagval	Interrupt tag value
INT channel	Channel number of transmitter
HCARD hCard	Handle of Lx429-3 card

RETURNS: The index of the appended command block if successful, or a negative value if an error occurs.

DESCRIPTION: Appends a conditional interrupt command block to the current end of the schedule. A conditional interrupt command block causes the Lx429-3 to generate a interrupt log list entry if *condition* evaluates as TRUE. The interrupt type placed in the log list is INTTYPE_OPCODE and the user specified value *tagval* is used as the info value. Entries are read out of the interrupt log list using L43_IntRd.

The *condition* flags listed below may be used to specify the interrupt condition.

condition	
Constant	**Description**
COND_ALWAYS	Always interrupt.
COND_DIO1ACT	Interrupt if DIO1 is high (5V).
COND_DIO1NACT	Interrupt if DIO1 is low (0V).
COND_DIO2ACT	Interrupt if DIO2 is high (5V).
COND_DIO2NACT	Interrupt if DIO2 is low (0V).
COND_DIO3ACT	Interrupt if DIO3 is high (5V).
COND_DIO3NACT	Interrupt if DIO3 is low (0V).
COND_DIO4ACT	Interrupt if DIO4 is high (5V).
COND_DIO4NACT	Interrupt if DIO4 is low (0V).

WARNINGS: A call to L43_ChConfig must precede this function.

SEE ALSO: L43_IntRd

SchedMsg.. Appends a message into the schedule.

SYNOPSIS: SCHNDX L43_SchedMsg(message, channel, hCard)

MSGADDR message	Address of message
INT channel	Channel number of transmitter
HCARD hCard	Handle of Lx429-3 card

RETURNS: The index of the appended command block if successful, or a negative value if an error occurs.

DESCRIPTION: Appends a message command block to the current end of the schedule. When a message command block is encountered in the schedule, the message value specified by *message* is loaded into the tranmsit channel's encoder. The message will be transmitted when the schedule subsequently encounters either a gap command block or another message command block.

Note: Execution of this function does NOT transmit the message. The message is transmitted only when the resulting schedule is executed after L43_CardStart starts the Lx429-3.

WARNINGS: A call to L43_ChConfig must precede this function. In addition, the message must have been created with L43_MsgCreate.

SEE ALSO: L43_SchedMsgEx, L43_MsgCreate

SchedMsgEx........................Appends a message and gap into the schedule.

SYNOPSIS: SCHNDX L43_SchedMsgEx(message, gap, channel, hCard)

MSGADDR message	Address of message
USHORT gap	Gap value to follow message
INT channel	Channel number of transmitter
HCARD hCard	Handle of Lx429-3 card

RETURNS: The index of the appended command block if successful, or a negative value if an error occurs.

DESCRIPTION: Appends a message command block to the current end of the schedule. A message command block loads the message value specified by *message* into the tranmsit channel's encoder. The message will be transmitted when the schedule encounters either a gap command block or another message command block.

gap specifies the gap value in bit time to follow the message. This gap value allows the user to explicitly specify a gap value of less than four bit times. This gap value is overwritten if followed by a gap command block.

Note: Execution of this function does NOT transmit the message. The message is transmitted only when the resulting schedule is executed after L43_CardStart starts the Lx429-3.

WARNINGS: A call to L43_ChConfig must precede this function. In addition, the message must have been created with L43_MsgCreate.

SEE ALSO: L43_SchedMsg, L43_MsgCreate

SchedPause .... Appends a pause command block into the schedule.

SYNOPSIS: SCHNDX L43_SchedPause(channel, hCard)

INT channel	Channel number of transmitter
HCARD hCard	Handle of Lx429-3 card

RETURNS: The index of the appended command block if successful, or a negative value if an error occurs.

DESCRIPTION: Appends a pasue command block to the current end of the schedule. A pause command block pauses the channel until the card is un-paused using L43_ChResume.

Note: Execution of this function does NOT pause the channel. The channel is paused only when the resulting command block is executed after L43_CardStart starts the Lx429-3.

WARNINGS: A call to L43_ChConfig must precede this function.

SEE ALSO: L43_SchedHalt, L43_SchedRestart

SchedRestart............... Appends a restart command block into the schedule.

SYNOPSIS: SCHNDX L43_SchedRestart(channel, hCard)

INT channel	Channel number of transmitter
HCARD hCard	Handle of Lx429-3 card

RETURNS: The index of the appended command block if successful, or a negative value if an error occurs.

DESCRIPTION: Appends a restart command block to the current end of the schedule. A restart command block restarts the schedule back at the beginning. A restart command block is automatically appended to the end of the schedule, so this function does not need to be called for simple schedules.

Note: Execution of this function does NOT restart the schedule. The schedule is restarted only when the resulting command block is executed after L43_CardStart starts the Lx429-3.

WARNINGS: A call to L43_ChConfig must precede this function.

SEE ALSO: L43_SchedHalt, L43_SchedPause

SchedReturn .... Appends a return command block into the schedule.

SYNOPSIS: SCHNDX L43_SchedReturn(channel, hCard)

INT channel	Channel number of transmitter
HCARD hCard	Handle of Lx429-3 card

RETURNS: The index of the appended command block if successful, or a negative value if an error occurs.

DESCRIPTION: Appends a return command block to the current end of the schedule. A resturn command block returns the schedule to the point at which the last call command was made. For every call command there must be a return command to insure proper operation.

WARNINGS: A call to L43_ChConfig must precede this function.

SEE ALSO: L43_SchedCall

SeqConfig................................ Configures the sequential record of the card.

SYNOPSIS: ERRVAL L43_SeqConfig(ctrlflags, hCard)

ULONG ctrlflags	Selects configuration options
HCARD hCard	Handle of Lx429-3 card

RETURNS: A negative value if an error occurs, otherwise zero.

DESCRIPTION: Configures the Lx429-3 sequential record by allocating an on-board buffer and initializing internal pointers associated with the buffer.

ctrlflags	
Constant	**Description**
SEQCFG_DEFAULT	Select all default settings (**bold** below)
SEQCFG_DISABLE	Disable sequential record
SEQCFG_FILLHALT	Enable sequential record in fill and halt mode
SEQCFG_CONTINUOUS	Enable sequential record in continuous mode
SEQCFG_DELTA	Enable sequential record in delta mode
SEQCFG_INTERVAL	Enable sequential record in interval mode
SEQCFG_16K	Allocate a 16K sequential record buffer
SEQCFG_ALLAVAIL	Allocate all available memory to the sequential record
SEQCFG_32K	Allocate a 32K sequential record buffer
SEQCFG_64K	Allocate a 64K sequential record buffer
SEQCFG_128K	Allocate a 128K sequential record buffer
SEQCFG_DPRAM	Allocate all available dual-port RAM to the sequential record
SEQCFG_NOINTFULL	Do not generate interrupt when the sequential record is full
SEQCFG_INTFULL	Generate interrupt when the sequential record is full
SEQCFG_NOINTFREQ	Do not generate interrupt at user defined frequency
SEQCFG_INTFREQ	Generate interrupt at user defined frequency

WARNINGS: If the SEQCFG_ALLAVAIL flag is used, L43_SeqConfig should be the last of the functions called that allocate memory before L43_CardStart is called.

SEE ALSO: L43_SeqRd, L43_SeqInterval, L43_SeqIntFrequency

SeqInterval ....Sets the interval time for the sequential record.

SYNOPSIS: INT L43_SeqInterval(interval, mode, hCard)

INT interval	Interval time (in seconds)
INT mode	Mode to determine interval value
HCARD hCard	Handle of Lx429-3 card

RETURNS: The actual interval value that the Lx429-3 will use.

DESCRIPTION: This function is used when the sequential record has been configured with the SEQCFG_INTERVAL flag. In this case, the sequential record will record only the first occurence of a message within the specified *interval*.

The Lx429-3 can not accomidate all interval values that could be passed through *interval*. The Lx429-3 operates on a 48-bit internal timer, which it will use for the interval processing. The mode specified will help determine the actual interval that will be used. The constants below should be used to set the mode.

mode	
Constant	**Description**
INTERVALMODE_CLOSEST	Uses the value closest to the specified interval
INTERVALMODE_LESS	Uses the value just less than specified interval
INTERVALMODE_GREATER	Uses the value just greater the specified interval

WARNINGS: None.

SEE ALSO: L43_SeqConfig, L43_SeqRd

SeqIntFrequency..... Sets the interrupt frequency for the sequential record.

SYNOPSIS: USHORT L43_SeqIntFrequency(intfreq, hCard)

USHORT intfreq	Interrupt frequency
HCARD hCard	Handle of Lx429-3 card

RETURNS: The previous interrupt frequency.

DESCRIPTION: This function is used when the sequential record has been configured with the SEQCFG_INTFREQ flag. In this case, the sequential record will generate an interrupt log list entry after it records *intfreq* amount of messages. It continues in this manner until the sequential record is stopped.

WARNINGS: None.

SEE ALSO: L43_SeqConfig, L43_SeqRd

SeqIsRunning................Determines whether the sequential record is active.

SYNOPSIS: BOOL L43_SeqIsRunning(hCard)
HCARD hCard Handle of Lx429-3 card

RETURNS: TRUE if the sequential record is still active, otherwise FALSE.

DESCRIPTION: This function is typically used when the sequential record has been configured with the SEQCFG_FILLHALT flag. In this case, the sequential record disables itself after filling its internal buffer. Thus, L43_SeqIsRunning effectively indicates whether the buffer is full.

WARNINGS: None.

SEE ALSO: L43_SeqConfig, L43_SeqRd

SeqRd ....Reads the next record from the sequential record.

SYNOPSIS: USHORT L43_SeqRd(seqbuf, hCard)

LPSEQRECORD seqbuf	Pointer to structure
HCARD hCard	Handle of Lx429-3 card

RETURNS: The number of (16-bit) words read from the sequential record, or zero if there was no new record to read.

DESCRIPTION: Copies the next entry from the sequential record on the Lx429-3 to the user-supplied buffer. Using the pre-defined sequential record structure SEQRECORD allows for easy handling of the data.

SEQRECORD structure

Field	Size	Description
vercount	USHORT	Version and count information.
timestampl	USHORT	Low word of time stamp - resolution set by L43_TimerResolution.
timestamph	USHORT	High word of time stamp - resolution set by L43_TimerResolution.
activity	USHORT	Activity - see table below for details.
data	ULONG	Data value - 32-bit ARINC 429 data word value, or 12-bit ARINC 717 data word with MSBs zero filled.
wordnum	USHORT	Word number (717 only)
subframe	USHORT	Subframe number (717 only)
superframe	USHORT	Superframe number (717 only)

vercount consists of the version (high byte) and the count of the number of words in the record (low byte). If it is an ARINC 429 record, then the version is one (1) and the count is six (6). If it is an ARINC 717 record, then the version is two (2) and the count is nine (9).

The activity field is different for ARINC 429 and ARINC 717. It may be tested by AND-ing the values returned with the constants from the corresponding table below.

The last three words in the sequential record structure are used only by ARINC 717 to to identify the data word. The *superframe* field contains the value of the superframe counter when the word was recorded, but this value may be invalid if superframes had not been configured.

ARINC 429 activity field	
Constant	**Description**
MSGACT_BRD	The board number (bits 12-15).
MSGACT_ CHAN	The channel number (bits 8-11).
MSGACT_ ASPD	Decoder auto speed detected. When a receiver is in auto speed detection mode, this bit reflects the speed detected. A one signifies high speed, and a zero signifies low speed.
MSGACT_ ERR	If set, it signifies that an error occurred in receiving this word. The type of error is defined by the following bits.
MSGACT_ GAP	Gap Error - a gap of less than four bit times preceded the word.
MSGACT_ PAR	Parity error - a parity error was detected in the word.
MSGACT_ LONG	Long word error - a word of more than 32-bits was detected.
MSGACT_ BIT	Bit timing error - an error occurred while decoding the bits of the word (short bits or long bits).
MSGACT_ TO	Time out error - the decoder timed out while receiving a word (short word).
MSGACT_ HIT	Signifies that the message has been processed by the firmware (the "hit bit").

ARINC 717 activity field	
Constant	**Description**
MSGACT717_BRD	The board number (bits 12-15).
MSGACT717_ CHAN	The channel number (bits 8-11).
MSGACT717_ SPD	The speed of the bus (bits7-5). Where: 0 = 64 WPS 1 = 128 WPS 2 = 256 WPS 3 = 512 WPS 4 = 1024 WPS 5 = 2048 WPS 6 = 4096 WPS 7 = 8192 WPS
MSGACT717_ TO	Time out error - the decoder timed out while receiving a word (short word).
MSGACT717_ HIT	Signifies that the message has been processed by the firmware (the "hit bit").

WARNINGS: None.

SEE ALSO: L43_SeqConfig, L43_SeqIsRunning

TimerClear.. Clears the time-tag timer.

SYNOPSIS: VOID L43_TimerClear(hCard)

HCARD hCard Handle of Lx429-3 card

RETURNS: None.

DESCRIPTION: Clears the time-tag timer to zero on the specified Lx429-3 card.

WARNINGS: None.

SEE ALSO: L43_TimerRd, L43_TimerResolution

TimerRd .... Reads the time-tag timer.

SYNOPSIS: ULONG L43_TimerRd(hCard)

HCARD hCard	Handle of Lx429-3 card

RETURNS: The current time-tag timer value.

DESCRIPTION: Reads the current value of the time-tag timer on the specified Lx429-3 card.

WARNINGS: None.

SEE ALSO: L43_TimerClear, L43_TimerResolution

TimerResolution.................................Selects the time-tag timer resolution.

SYNOPSIS: INT L43_TimerResolution(timerresol, hCard)

INT timerresol	Selects the timer resolution
HCARD hCard	Handle of Lx429-3 card

RETURNS: A negative value if an error occurs, or zero if successful.

DESCRIPTION: Selects the resolution for the time-tag timer on the specified Lx429-3. *timerresol* must be one of the following predefined constants.

timerresol		
Constant	**Resolution**	**Range (h:m:s)**
TIMERRESOL_1US	1us	1:11:34
TIMERRESOL_16US	16us	19:05:19
TIMERRESOL_1024US	1024us	50 days
TIMERRESOL_16384US	16384us	814 days

WARNINGS: After changing the resolution, a call to L43_TimerClear should be made to clear the timer.

SEE ALSO: L43_TimerClear

ValFromAscii..Converts a string to a numeric value.

SYNOPSIS: ULONG L43_ValFromAscii(asciistr, radixval)

LPCSTR asciistr	ASCII string to convert
INT radixval	Radix of string

RETURNS: The converted integer numeric value.

DESCRIPTION: Converts a string representation of a 16-bit value with the specified radix to an integer. Processing stops at the first null-terminator.

Note: This is a utility function and does not access the Lx429-3 hardware.

WARNINGS: No check is made for invalid characters.

SEE ALSO: L43_ValToAscii

ValGetBits........................Extract the specified bit field from an integer value.

SYNOPSIS: USHORT L43_ValGetBits(oldval, startbit, endbit)

USHORT oldval	The old value
INT startbit	Position of starting bit of field
INT endbit	Position of ending bit of field

RETURNS: The value of the extracted bit field.

DESCRIPTION: Extracts the specified bit field from the 16-bit integer *oldval*. The result is obtained by masking the field and shifting the *endbit* to the LSB of the return value.

Note: This is a utility function and does not access the Lx429-3 hardware.

WARNINGS: None.

SEE ALSO: L43_ValPutBits

ValPutBits .... Inserts a bit field into an integer value.

SYNOPSIS: USHORT L43_ValPutBits(oldval, newfld, startbit, endbit)

USHORT oldval	The old value
USHORT newfld	The value of the new field
INT startbit	Position of starting bit of field
INT endbit	Position of ending bit of field

RETURNS: The integer value with the inserted bit field.

DESCRIPTION: Inserts a bit field into an 16-bit integer value. The *oldval* is masked and ORed with the shifted value of *newfld*.

Note: This is a utility function and does not access the Lx429-3 hardware.

WARNINGS: None.

SEE ALSO: L43_ValGetBits

ValToAscii..................................Creates a string representation of an integer.

SYNOPSIS: LPSTR L43_ValToAscii(value, asciistr, numbits, radixval)

ULONG value	The value to be converted
LPSTR asciistr	A string to receive the results.
INT numbits	The number of significant bits
INT radixval	The radix value

RETURNS: A string representing the integer.

DESCRIPTION: Creates a string representation of an integer in a specified radix. The string is copied to *asciistr* and is also returned. The string is always null-terminated. *asciistr* is assumed to be large enough to hold the resulting string.

The length of the string is determined by *numbits* and *radixval* and is padded by leading zeros. *radixval* can be any positive integer but is commonly 16 for hexadecimal, 8 for octal, or 10 for decimal. For example, a string representation of a value with 16 significant bits and a radix of 16 will always be 4 characters long followed by a null-terminator.

Note: This is a utility function and does not access the Lx429-3 hardware.

WARNINGS: None.

SEE ALSO: L43_ValFromAscii

This page intentionally blank.

APPENDIX B: CONNECTORS

D-Subminiature connector

The LP429-3 and LC429-3 use a standard 25-pin male D-subminiature connector as shown in Fig. B.1.

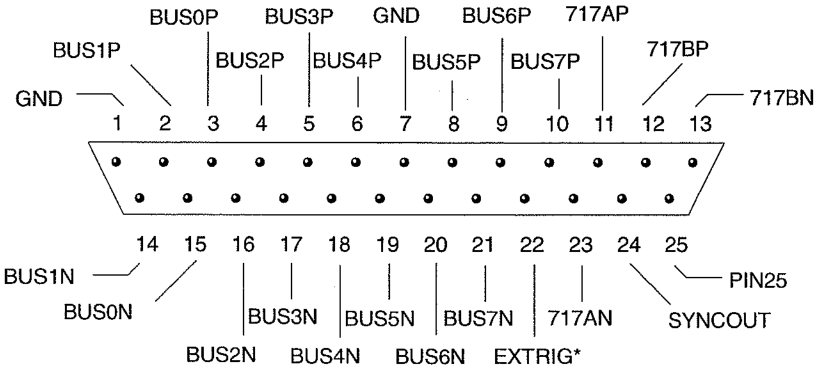


Fig. B.1 Pinouts of Lx429-3 D-sub connector

LP429-3 signal header

The LP429-3 provides several general purpose digital I/O signals. Possible uses for these signals include:

- triggering message transmissions
- triggering error injection
- allowing on-board events to trigger external processes

Some external signals are available at the D-Subminiature connector described above. Otherwise, these signals are available on the I/O header P10 as shown in Fig. B.2. A signal can be routed to the D-subminiature connector by placing a jumper on the header between the desired signal and PIN25. Contact Ballard Technology for more information on the use of these signals.

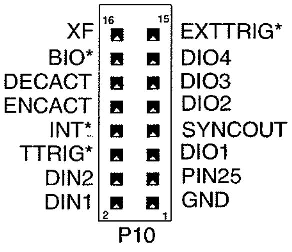


Fig. B.2 LP429-3 external signals header

LC429-3 signal header

Some of the signals on the LC429-3 header (Fig. B.3) are the same as those on the LP429-3 (Fig. B.2). Because the cPCI card cage makes it more difficult to access signals on a header, the header on the LC429-3 is configured so that additional jumpers on the header can be used to route other signals to unused pins on the D-subminiature connector.

The signals PIN25, 717AP, 717AN, 717BP, and 717BN are connected between the header and the D-Subminiature connector. PIN25 is always available, but 717AP, 717AN, 717BP, or 717BN may already be in use by the 717 biphase transceivers in U1 and U3 (as is the case with the LC429-3/717). **Warning: Verify that components are not installed at U1 and U3 before placing jumpers on 717AP, 717AN, 717BP, or 717BN.** Contact Ballard Technology for more information on the use of these signals.

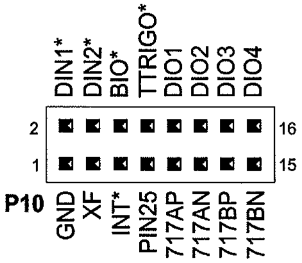


Fig. B.3 LC429-3 external signals header

APPENDIX C: ARINC 717 OVERVIEW

The Lx429-3/717 interfaces to both ARINC 429 and ARINC 717 databuses. Though custom versions are available with other mixes of ARINC 429 channels, the standard Lx429-3/717 is equivalent to an Lx429-3/2R2T with the addition of the ARINC 717 capability. There are two Harvard biphase ARINC 717 channels, each of which can be configured in software as either receive or transmit, and two bipolar ARINC 717 channels, which are fixed at one receive and one transmit. All ARINC 429 and 717 channels may be used concurrently.

ARINC 717

The ARINC 717 specification defines the communication protocol used by the Digital Flight Data Acquisition Unit (DFDAU) and the Digital Flight Data Recorder (DRDR). The DFDAU accumulates data from the aircraft systems and transmits it to the DFDR on a Harvard biphase ARINC 717 databus. A similar bipolar ARINC 717 databus can connect to a Quick Access Recorder (QAR). Ballard Technology's Lx429-3/717 card is used to monitor and simulate these databuses.

ARINC 717 data is organized by words, subframes, and frames. A subframe is a collection of received data words in a one second interval. There are four subframes in a frame. Although not defined by the ARINC 717 specification, frames are often logically grouped together to form superframes.

The basic unit of information defined by the ARINC 717 protocol is a 12-bit word. There are two types of words: sync words and data words. The sync (or synch) word is always the first word in a subframe. Data words contain the information collected by the DFDAU and transmitted to the DFDR.

Subframes are continually transmitted at a rate of once per second with no gaps between words or subframes. Subframes are numbered one through four. The standard sync word values used to identify the four subframes are shown in Table C.1. Other sync word values may be configured through software.

	Octal	Hex	Decimal
Sync word 1	1107	0247	583
Sync word 2	2670	05B8	1464
Sync word 3	5107	0A47	2631
Sync word 4	6670	0DB8	3512

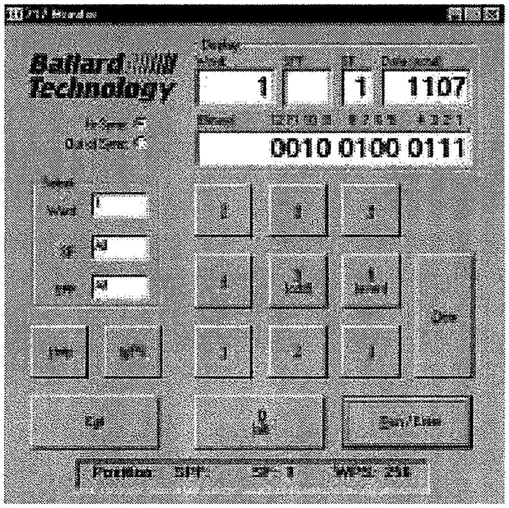
Table C.1 ARINC 717 sync word values

The length of a subframe is determined by the transmission speed of the bus. Although the ARINC 717 specifies bus speeds of 64 to 512 words per second (WPS), the Lx429-3/717 card is capable of running at speeds of 64, 128, 256, 512, 1024, 2048, 4096, or 8192 WPS.

Four subframes are grouped together into a single frame. Since a subframe is transmitted every second, a frame is transmitted every four seconds. Frames can be grouped together to form superframes. Superframes are an extension of the ARINC 717 specification and are used to organize large amounts of unique data. The number of frames in a superframe and the location of the superframe counter in each frame are user defined.

717 Monitor GUI

The 717 Monitor is the Windows 95/98/NT based GUI (graphical user interface) developed by Ballard Technology to simplify the monitoring of an ARINC 717 databus. The 717 Monitor GUI displays the specified data when it is received. Fig. C.1 shows a sample 717 Monitor screen. This software GUI and its documentation are included with the Lx429-3/717.



Display
Word 1
SF 1
Data (octal) 1107
Binary 0010 0100 0111

Fig. C.1 Ballard Technology's 717 Monitor GUI

Programming Basics

The Ballard Technology 717 Monitor has been included with the Lx429-3/717 so users may not need to develop their own custom applications. For those wishing to write their own application, sample ARINC 717 programs are located on the driver disks. A simple receive example is shown in Fig. C.2. Notice that it uses the required `L43_CardOpen, L43_ChConfig, L43_CardStart, L43_CardStop, and L43_CardClose` functions, similar to the ARINC 429 examples discussed in Chapter 3. A complete list of driver functions relevant to ARINC 717 operation may be found in Appendix D of this manual.

The capabilities of interrupts and the sequential record as described in Chapter 4 and Appendix A are also available to ARINC 717 programs. ARINC 717 words are marked for interrupts or entry into the sequential record using `L43_SubFrmWordConfig` or `L43_SuperFrmWordConfig`. Alternatively, all 717 words may be marked for entry into the sequential record using `L43_ChConfig.`

Superframes are unique to ARINC 717. A program using superframes should first use **L43_SuperFrmConfig** to establish the number of frames per superframe and to allocate space for superframe records. The location of the superframe counter is specified using **L43_SuperFrmCounterPos**. Then, **L43_SuperFrmDefine** is used to mark specific subframe words as superframe words. Using **L43_SuperFrmWordConfig**, individual superframe words can be configured to generate an interrrupt or to be entered into the sequential record. **L43_SuperFrmWordRd** and **L43_SuperFrmWordWr** are used to read and write the values of superframe words.

```
handle = L43_CardOpen (0);        // OPEN
L43_CardReset(handle);                                          // RESET
L43_ChConfig(CHCFG717_256WPS,CH6,handle);  // CONFIGURE
L43_CardStart(handle);                                          // START

while (!L43_SyncTest(CH6,handle));  // WAIT FOR SYNCHRONIZATION

//AS REQUIRED BY THE APPLICATION, HANDLE DATA...REPEAT

    if (!L43_SubFrmWordRd(&syncword1,1,1,CH6,handle))
        use the data;  // HANDLE DATA

L43_CardStop(handle);     // STOP
L43_CardClose (handle);       // CLOSE
```

Fig. C.2 Example ARINC 717 receive program

This page intentionally blank.

APPENDIX D: ARINC 717 FUNCTION REFERENCE

This appendix provides detailed information about each driver function for the ARINC 717 operation of the Lx429-3/717. The descriptions and examples discussed here are intended for use with programs written in the C language. Users of other languages should contact Ballard Technology for assistance.

There are two catagories of ARINC 717 driver functions: those that are specific to ARINC 717 and those that are shared with ARINC 429. A summary of ARINC 717 functions is shown in Table D.1. Although all the ARINC 429 functions documented in Appendix A are included with the Lx429-3/717 API, only those shown in Table D.1 apply to ARINC 717 operation. The bolded functions are documented in this Appendix. The unbolded functions are documented in Appendix A and may be useful in some ARINC 717 applications.

The following are commonly used parameters that are not described in detail under each command:

"hCard" parameters

Nearly all functions require a card "handle" parameter. **`L43_CardOpen`**, which is always the first Lx429-3 function called, returns this handle. The handle uniquely identifies a card when more than one Lx429-3 is used in a single computer. The handle is always the last parameter in any function that requires it.

"ctrlflags" parameters

Many functions have a *ctrlflags* parameter. Each bit controls an option in this bitmask parameter. Constants are defined in the header (.H) file for these parameters. The name of a constant reflects the function in which it is used (e.g., CHCFG_DEFAULT is used in the **`L43_ChConfig`** function). Option parameters are always first in the parameter list of a function that accepts them. The default options can always be selected by using the ??_DEFAULT constant where ?? depends on the function in which it is used (e.g., CHCFG_DEFAULT). When multiple options are selected, the constants should be bitwise OR-ed together. The default options are shown in bold in this appendix. Since the default constants are *#defined* to 0, only non-default constants actually need to be included in the OR-ing. The constants defined in the header file should be used by name (not value) in your code since the values are subject to change.

"channel" parameters

This is an integer that specifies which channel on the Lx429-3/717 to use. The predefined constants listed below can also be used for this parameter. On the standard Lx429-3/717 the 717 functionality is on channels 4 through 7. Nonstandard implementations can be determined by running the test program L43TST32.EXE as described in Chapter 2.

Constants	Value	Description
CH0	0	429 Receive channel 0.
CH1	1	429 Receive channel 1.
CH2	2	429 Transmit channel 2.
CH3	3	429 Transmit channel 3.
CH4	4	717 Bipolar Receive channel 4.
CH5	5	717 Bipolar Transmit channel 5.
CH6	6	717 Biphase channel 6.
CH7	7	717 Biphase channel 7.

"subframe" parameters

Subframe Number. This is an integer specifying the index of a subframe. It must be in the of range one to four (1 to 4).

"superfrmnum" parameters

Superframe Number. This is an interger specifying a particular frame within a superframe. It must be in the range of one to the number of frames per superframe (1 to frpersuperfr).

"wordnum" parameters

Word Number. This is an integer specifying the index of a word in a particular subframe. It must be in the range from one to the subframe size, where subframe size is the speed in words per second (WPS).

"error" parameters

Error. This is an integer returned by functions that provide error information to an application. Functions that return an address return a zero if an error is encountered. Functions that don't return an address return a negative value if an error is encountered. The negative error values and the predefined names that can be used to test for them are listed in Table A.2 in Appendix A. See the function description to determine if a particular function can return these error values.

The following pages contain descriptions of the various ARINC 717 driver functions for the Lx429-3/717.

CARD Functions	
Function	**Description**
L43_CardClose	Disables access to a Lx429-3 card and releases its hardware resources
L43_CardOpen	Enables access to a Lx429-3 card and secure hardware resources
L43_CardReset	Resets the Lx429-3 hardware; destroys all existing configuration data
L43_CardResume	Starts the card without resetting the transmitter schedules
L43_CardStart	Starts operation of the Lx429-3 card
L43_CardStop	Stops operation of the Lx429-3 card
L43_CardTest	Performs a hardware test on the Lx429-3

Table D.1 ARINC 717 API function summary (continued on next page)

CHANNEL Functions	
Function	Description
L43_ChConfig	Configures the channel, used with both 429 and 717
L43_Ch717Status	Gets the status of a channel
L43_Get717ChanCount	Gets the number of receive and transmit ARINC 717 channels
L43_Is717BiPhaseChan	Test whether the channel is Biphase 717
L43_Is717Chan	Tests whether the channel is ARINC 717
L43_Is717RcvChan	Tests whether the channel is ARINC 717 receive
L43_Is717XmtChan	Tests whether the channel is ARINC 717 transmit
SUBFRAME Functions	
Function	Description
L43_SubFrmBlkRd	Block reads the entire 717 subframe
L43_SubFrmBlkWr	Block writes the entire 717 subframe
L43_SubFrmStatus	Reads the status of a 717 subframe
L43_SubFrmWordConfig	Sets configuration options for a 717 word in a subframe
L43_SubFrmWordRd	Reads a 717 word from a subframe
L43_SubFrmWordWr	Write a 717 word to a subframe
SUPERFRAME Functions	
Function	Description
L43_SuperFrmConfig	Configures for use with superframes
L43_SuperFrmCounterPos	Specifies the location of the superframe counter
L43_SuperFrmDefine	Marks a word as a superframe word
L43_SuperFrmWordConfig	Sets configuration options for a 717 word in a superframe
L43_SuperFrm WordRd	Reads the specified superframe word
L43_SuperFrmWordWr	Writes a word to the superframe structure
717 SYNC Functions	
Function	Description
L43_SyncTest	Tests whether a receive channel is in-sync
L43_SyncValRd	Reads the values used as sync words
L43_SyncValWr	Writes the values to be used as sync words
SEQUENTIAL RECORD Functions	
Function	Description
L43_SeqConfig	Configures the sequential record.
L43_SeqIntFrequency	Sets an interrupt frequency if using interrupt frequency mode
L43_SeqIsRunning	Determines if Sequential Record is recording messages
L43_SeqRd	Reads the next message record from the Sequential Record
INTERRUPT Functions	
Function	Description
L43_IntConfig	Enables interrupts and initializes the interrupt log list
L43_IntInstall	Installs an interrupt handler (OS-dependent)
L43_IntRd	Reads an entry from the interrupt log list
L43_IntUninstall	Removes an interrupt handler (OS-dependent)
I/O Functions	
Function	Description
L43_ExtDIORd	Reads the value of the specified Digital I/O pin
L43_ExtDIOWr	Sets the value of the specified Digital I/O pin
L43_ExtLEDRd	Reads the On/Off value of the LED.
L43_ExtLEDWr	Sets the On/Off value of the LED.
TIMER Functions	
Function	Description
L43_TimerClear	Clears the time-tag timer
L43_TimerRd	Reads the current value of the time-tag timer
L43_TimerResolution	Selects a time-tag timer resolution
UTILITY Functions	
Function	Description
L43_ValFromAscii	Creates an integer value from an ASCII string
L43_ValGetBits	Extracts a bit field from an integer value
L43_ValPutBits	Puts a bit field into an integer value
L43_ValToAscii	Creates an ASCII string from an integer

Table D.1 ARINC 717 API function summary (continued)

This page intentionally blank.

ChConfig...Configures the channel.

SYNOPSIS: ERRVAL L43_ChConfig(ctrlflags, channel, hCard);

ULONG ctrlflags	Selects the channel options
INT channel	Channel number
HCARD hCard	Handle of the card

RETURNS: A negative value if an error occurs, or zero if successful.

DESCRIPTION: This is the same function as L43_ChConfig for ARINC 429 except that the following options apply to ARINC 717 operation. See Appendix A for ARINC 429 options.

ctrlflags			
Constant	**Description**	**Rcv**	**Xmt**
CHCFG717_DEFAULT	Select all default settings (**bold** below)	✔	✔
CHCFG717_AUTOSPEED	Select auto speed detection	✔	
CHCFG717_64WPS	Select speed of 64WPS	✔	✔
CHCFG717_128WP	Select speed of 128WPS	✔	✔
CHCFG717_256WPS	Select speed of 256WPS (default)	✔	✔
CHCFG717_512WPS	Select speed of 512WPS	✔	✔
CHCFG717_1024WPS	Select speed of 1024WPS	✔	✔
CHCFG717_2048WPS	Select speed of 2048WPS	✔	✔
CHCFG717_4096WPS	Select speed of 4096WPS	✔	✔
CHCFG717_8192WPS	Select speed of 8192WPS	✔	✔
CHCFG717_BIPHASERCV	Set Biphase channel to receive	✔	
CHCFG717_BIPHASEXMT	Set Biphase channel to transmit		✔
CHCFG717_BIPOLARHSPD	Set Bipolar transmit to high speed (1.5us) rise time. (10us default)		✔
CHCFG717_SELFTEST	Enable internal wraparound	✔	✔
CHCFG717_SELFTESTOFF	Disable internal wraparound	✔	✔
CHCFG717_SYNC	Enable sync output on 717 words	✔	✔
CHCFG717_SYNCOFF	Disable sync output	✔	✔
CHCFG717_ACTIVE	Enable channel activity	✔	✔
CHCFG717_INACTIVE	Disable channel activity	✔	✔
CHCFG717_EXTTRIG	Enable external trigger for all messages		✔
CHCFG717_EXTOFF	External trigger is enabled on message level		✔
CHCFG717_HIT	Hit counter is enabled for all sub frames	✔	✔
CHCFG717_NOHIT	Hit counter is disabled	✔	✔
CHCFG717_TIMETAG	Enable time-tag for all subframes	✔	✔
CHCFG717_TIMETAGOFF	Time-tag is disabled	✔	✔
CHCFG717_SEQALL	Record entire channel in sequential record	✔	✔
CHCFG717_SEQSEL	Sequential record recording is enabled at word level	✔	✔
CHCFG717_INTERR	Enable interrupts on Out of Sync Error	✔	
CHCFG717_NOINTERR	No interrupt on Out of Sync Error	✔	
CHCFG717_PAUSE	Mark channel as paused	✔	✔
CHCFG717_UNPAUSE	Mark channel as unpaused	✔	✔

WARNINGS: None.

SEE ALSO: L43_ChConfig in Appendix A

Ch717Status .... Gets the status of a channel

SYNOPSIS: ERRVAL L43_Ch717Status (buf, channel, hCard);

LPUSHORT buf	Array to receive the status
INT channel	Channel number
HCARD hCard	Handle of the Lx429-3/717 card

RETURNS: Returns a negative value if unable to read status, otherwise zero.

DESCRIPTION: The status of the specified *channel* is read and returned to *buf*. The current word, subframe and superframe indicate the current position in the cycle.

buf[0] = status bits
Bit 1: InSync – the channel is in-sync (receive only)
Bit 5: SPFValid – the Superframe Counter value is valid (receive only)
All other bits are reserved.
buf[1] = current word number (range of 1 to number of words per subframe).
buf[2] = current subframe number(range of 1 to 4)
buf[3] = current value of the Superframe Counter (range of 0 to the superframe modulo)

WARNINGS: None.

SEE ALSO: L43_SubFrmStatus

Get717ChanCount.............Gets the number of receive and transmit channels

SYNOPSIS: VOID L43_Get717ChanCount(rcvcount, xmtcount, hCard);

LPINT rcvcount	Number of receive channels
LPINT xmtcount	Number of transmit channels
HCARD hCard	Handle of the card

RETURNS: None.

DESCRIPTION: Determines the number of ARINC 717 receive and transmit channels on the card.

WARNINGS: None.

SEE ALSO: L43_Is717Chan

Is717BiPhaseChan .... Test whether a channel is biphase 717

SYNOPSIS:	BOOL L43_Is717BiPhaseChan(channel, hCard);	
	INT channel	Channel number
	HCARD hCard	Handle of the card

RETURNS: A non-zero value if the channel is biphase 717, or zero if it is not biphase 717.

DESCRIPTION: Determines whether the specified channel is an ARINC 717 biphase channel.

WARNINGS: None.

SEE ALSO: L43_Is717Chan

Is717Chan .... Tests whether a channel is ARINC 717.

SYNOPSIS: BOOL L43_Is717Chan(channel, hCard);

INT channel	Channel number
HCARD hCard	Handle of the card

RETURNS: A non-zero value if it is a 717 channel, or zero if it is not a 717 channel.

DESCRIPTION: Determines whether the specified channel is an ARINC 717 channel. If it returns a true value (non-zero), then the channel could be 717 biphase or bipolar, receive or transmit.

WARNINGS: None.

SEE ALSO: L43_Is717RcvChan, L43_Is717XmtChan

Is717RcvChan.....................Tests whether a channel is ARINC 717 receive.

SYNOPSIS: BOOL L43_Is717RcvChan(channel, hCard);

INT channel	Channel number
HCARD hCard	Handle of the card

RETURNS: A non-zero value if it is a 717 receive channel, or zero if it is not a 717 receive channel.

DESCRIPTION: Determines whether the specified channel is an ARINC 717 receive channel. If it returns a true value (non-zero), then the channel could be 717 biphase or bipolar.

WARNINGS: None.

SEE ALSO: L43_Is717XmtChan

Is717XmtChan .... Tests whether a channel is ARINC 717 transmit.

SYNOPSIS:	BOOL L43_Is717XmtChan(channel, hCard);	
	INT channel	Channel number
	HCARD hCard	Handle of the card

RETURNS: A non-zero value if it is a 717 transmit channel, or zero if it is not a 717 transmit channel.

DESCRIPTION: Determines whether the specified channel is an ARINC 717 transmit channel. If it returns a true value (non-zero), then the channel could be 717 biphase or bipolar.

WARNINGS: None.

SEE ALSO: L43_Is717RcvChan

SubFrmBlkRd .... Block reads an entire subframe.

SYNOPSIS: ERRVAL L43_SubFrmBlkRd(databuf, subframe, channel, hCard);

LPUSHORT databuf	Pointer to destination array
INT subframe	Subframe to read
INT channel	Channel to read from
HCARD hCard	Handle of the card

RETURNS: A negative value if an error occurs, or zero if successful.

DESCRIPTION: Reads the subframe header plus an entire subframe from the card and puts it in an array pointed to by *databuf.* There are eight words for the header and WPS subframe data words. Where WPS is the speed of the 717 bus in words per second. For example, 72 words would be read for a 64 WPS bus. The eight header words are defined in L43_SubFrmStatus.

The 717 values are in the 12 least significant bits of each data word read. The four most significant bits are reserved for system use.

WARNINGS: The array must be large enough to hold one seconds worth of data plus eight words. Typical speeds are 64 and 256 words per second.

SEE ALSO: L43_SubFrmStatus, L43_SubFrmBlkWr, L43_SubFrmWordRd

SubFrmBlkWr ....Block writes an entire subframe.

SYNOPSIS: ERRVAL L43_SubFrmBlkWr(databuf, subframe, channel, hCard);

LPUSHORT databuf	Pointer to array of data to write
INT subframe	Number of the subframe to write to
INT channel	Channel number
HCARD hCard	Handle of the card

RETURNS: A negative value if an error occurs, or zero if successful.

DESCRIPTION: Writes an entire subframe with its header to the card from an array pointed to by *databuf*. The number of words written is equal to eight (for the header) plus the speed of the 717 bus in words per second (WPS). The 717 values are in the 12 least significant bits of each data word. The header and the four most significant bits of data words are reserved for system use and must be preserved. See L43_SubFrmStatus for information on the header.

WARNINGS: When using this function, the header and the values of the four most significant bits of each word must be preserved. Use L43_SubFrmBlkRd and modify only the 12 least significant bits of data words before calling this function.

SEE ALSO: L43_SubFrmBlkRd, L43_SubFrmWordWr, L43_SubFrmStatus

SubFrmStatus .... Reads the status of a subframe.

SYNOPSIS: ERRVAL L43_SubFrmStatus(buf, subframe, channel, hCard)

LPUSHORT buf	Pointer to destination array
INT subframe	Subframe number to get status from
INT channel	Channel to get get status from
HCARD hCard	Handle of the card

RETURNS: A negative value if an error occurs, or zero if successful.

DESCRIPTION: Reads status information (header) of subframe structure and writes it to data buffer pointed to by *buf*. There are eight words in the header. The eight words in the subframe header are shown in the table below.

Subframe Header

Word	Name	Description
0	Option Word	Reserved
1	Activity Word	See L43_SeqRd (Appx A)
2	Subframe number	Of this subframe
3	Bus speed in WPS	Size/speed of this subframe
4,5	Timestamp	Time at end of subframe (Same as 429 timestamp)
6,7	Superframe Pointer	Reserved

WARNINGS: None.

SEE ALSO: L43_SubFrmBlkRd, L43_SubFrmBlkWr, L43_SeqRd

SubFrmWordConfig .... Configures a word in a subframe.

SYNOPSIS: ERRVAL L43_SubFrmWordConfig(ctrlflags, subframe,
word, channel, hCard);

ULONG ctrlflags	Selects configuration options
INT subframe	Subframe number
INT word	Word number
INT channel	Channel number
HCARD hCard	Handle of the card

RETURNS: A negative value if an error occurs, or zero if successful.

DESCRIPTION: Configures a specific word in a specific subframe to genereate an interrupt or to be entered in the sequential record. By default words do not generate interrupts and are not entered in the sequential record. Words can be marked for word interrupts and/or subframe interrupts. The only difference between word interrupts and subframe interrupts is the *typeval* returned by L43_IntRd.

ctrlflags	
Constant	**Description**
WRDCFG_DEFAULT	Select all default options (**bold** default)
WRDCFG_SFINT	Enable subframe interrupt
WRDCFG_SFINTOFF	Disable subframe interrupt
WRDCFG_WRDINT	Enable word interrupt
WRDCFG_WRDINTOFF	Disable word interrupt
WRDCFG_SEQ	Enable sequential record
WRDCFG_SEQOFF	Disable sequential record

WARNINGS: None.

SEE ALSO: L43_IntRd, L43_SeqRd

SubFrmWordRd ....Reads a 717 word from a subframe.

SYNOPSIS: ERRVAL L43_SubFrmWordRd(value, subframe, word, channel, hCard);

LPUSHORT value	Pointer to destination
INT subframe	Subframe number
INT word	Word number
INT channel	Channel number
HCARD hCard	Handle of the card

RETURNS: A negative value if an error occurs, or zero if successful.

DESCRIPTION: Reads the 12-bit value of the specified 717 word from the specified subframe. The most significant bits of the word read are always zero.

WARNINGS: Returns invalid data for a word defined as a superframe.

SEE ALSO: L43_SubBlkRd, L43_SubFrmWordWr

SubFrmWordWr .... Writes a 717 word to a subframe.

SYNOPSIS: ERRVAL L43_SubFrmWordWr(value, subframe, word, channel, hCard);

USHORT value	Value of word to write
INT subframe	Subframe number
INT word	Word number
INT channel	Channel number
HCARD hCard	Handle of the card

RETURNS: A negative value if an error occurs, or zero if successful.

DESCRIPTION: Writes a 12 bit ARINC 717 value to the specifed word in the specified subframe. The four most significan bits of *value* may be anything, since this function preserves the on-board state of those bits.

WARNINGS: None.

SEE ALSO: L43_SubFrmBlkWr

SuperFrmConfig.................... Configures channel for use with superframes.

SYNOPSIS: ERRVAL L43_SuperFrmConfig(count, frpersuperfr, channel, hCard);

INT count	Number of superframe records to allocate
INT frpersuperfr	Number of frames per superframe
INT channel	Channel number
HCARD hCard	Handle of the card

RETURNS: A negative value if an error occurs, or zero if successful.

DESCRIPTION: Configures the channel to handle superframes by setting the number of frames per superframe and by allocating space for *count* superframe records.

WARNINGS: *count* must be greater than the number words that are are to be marked as superframe words (i.e., *count* must be greater than the number of times L43_SuperFrmDefine is called). For a given channel L43_SuperFrmConfig must precede any use of L43_SuperFrmDefine.

SEE ALSO: L43_ SuperFrmDefine

SuperFrmCounterPos .. Specifies the location of the superframe counter.

SYNOPSIS: ERRVAL L43_SuperFrmCounterPos(subframe, word, channel, hCard);

INT subframe	Subframe number
INT word	Word number
INT channel	Channel number
HCARD hCard	Handle of the card

RETURNS: A negative value if an error occurs, or zero if successful.

DESCRIPTION: Specifies the location of the superframe counter by channel, subframe and word number. If superframes are used, this function must be called before L43_CardStart.

WARNINGS: None.

SEE ALSO: L43_SuperFrmConfig, L43_SuperFrmDefine

This page intentionally blank.