DYNAMIC ENGINEERING

435 Park Dr., Ben Lomond, Calif. 95005
831-336-8891 **Fax** 831-336-3840
http://www.dyneng.com
sales@dyneng.com
Est. 1988

User Manual

PCI LVDS 8R Driver Documentation

Revision A
Corresponding Hardware: Revision C
10-2001-0203

PCI LVDS 8R
PCI based 8 channel LVDS Receiver card

Dynamic Engineering
435 Park Drive
Ben Lomond, CA 95005
831-336-8891
831-336-3840 FAX

Manual Revision A. Revised May 1, 2002

This document contains information of proprietary interest to Dynamic Engineering. It has been supplied in confidence and the recipient, by accepting this material, agrees that the subject matter will not be copied or reproduced, in whole or in part, nor its contents revealed in any manner or to any person except to meet the purpose for which it was delivered.

Dynamic Engineering has made every effort to ensure that this manual is accurate and complete. Still, the company reserves the right to make improvements or changes in the product described in this document at any time and without notice. Furthermore, Dynamic Engineering assumes no liability arising out of the application or use of the device described herein.

The electronic equipment described herein generates, uses, and can radiate radio frequency energy. Operation of this equipment in a residential area is likely to cause radio interference, in which case the user, at his own expense, will be required to take whatever measures may be required to correct the interference.

Dynamic Engineering's products are not authorized for use as critical components in life support devices or systems without the express written approval of the president of Dynamic Engineering.

Connection of incompatible hardware is likely to cause serious damage.

Table of Contents

List of Figures

no figures in this document

Introduction

The LV8R driver is a Windows NT driver for the PCI LVDS 8R board from Dynamic Engineering. This driver can control up to 30 boards in a system. The PCI LVDS 8R board receives 8 channels of LVDS input. A separate "Device Object" controls each LVDS channel, and a separate handle references each Device Object. IO Control calls (IOCTLs) are used to configure the hardware and the driver. ReadFile() calls are used to obtain LVDS data from the device. IOCTLs refer to a single Device Object instance, and therefore refer to a single channel on a device (except the IOCTL_LV8R_RESET_ALL_INPUT_FIFOS call, which affects all 8 channels on a given board).

Handles can be opened to specific channels on the device in Win32 by using the CreateFile() function call and passing in a Symbolic Link name. A Symbolic Link is the name of the device recognized by Windows. For the LV8R driver, Symbolic Link names are formed as LV8Rx_n where x indicates the zero based board number and n indicates the zero based channel number on the board. E.g. the second channel on the third board is LV8R2_1.

ReadFile() is used to read LVDS data from a specific channel specified by passing the appropriate handle opened via the CreateFile() function call. The driver has several modes in which it can execute a ReadFile(). The drivers ReadFile() mode can be changed by calling the IOCTL_LV8R_SET_TRIGGER_MODE. The most common mode is LOCAL_ACQUIRE mode. In this mode, the driver reads directly from the SDRAM bank on the device. LOCAL_ACQUIRE mode ReadFile() calls are limited to channels 0 and 4 on the device. In STREAMING_ACQUIRE_ONESHOT mode, the driver reads data directly from the LVDS input for that channel. The ReadFile() call will complete as soon as the device has received enough LVDS data through this channel to fill the buffer passed into ReadFile(). In STREAMING_ACQUIRE mode, the buffer passed into ReadFile() is treated as a circular buffer. When the device has completely filled the buffer it will immediately start filling again at the beginning of the buffer. This ReadFile() call will not complete until an IOCTL_LV8R_CHANNEL_STOP_ACQ is sent to the channels handle. Both STREAMING_ACQUIRE_ONESHOT and STREAMING_ACQUIRE mode ReadFile() calls can be made to any one of the 8 channels of the device.

Authors Note

This documentation is provided to supplement the PCI LVDS 8R device documentation provided by Dynamic Engineering. This documentation will provide information about all calls made to the driver, and how the driver interacts with the device for each of these calls.

Driver Installation

There are several files provided in each driver drop. These files include lv8r.sys, lv8r.reg, ddlv8r.h, lv8rtest.exe, and driver source files.

The lv8r.sys file is the binary driver file. In order to install the driver, place this file in your Winnt\system32\drivers directory.

The lv8r.reg file is the Windows NT registry entry file. This file contains the modifications to the Windows registry required to allow Windows to recognize the driver. In order to install the driver, double click on this file (or right click and select the Merge option in the context menu). This will merge the LV8R entries required by the driver into the Windows NT registry. Windows must be restarted after merging this file into the registry for the driver to work.

The ddlv8r.h file is the C header file that defines the Application Interface (API) to the driver. This file is required at compile time by any application that wishes to interface with the PCI LVDS 8R device. It is not needed by the driver installation.

The lv8rtest.exe file is a sample Windows NT console application that makes calls into the LV8R driver. It is not required during the driver installation.

Driver Startup

There are several tasks the LV8R driver must do when it is started. It must scan all possible PCI buses to detect every PCI LVDS 8R device in the system. It must create 8 “Device Objects” for every board it finds, one per channel. It must initialize each of these Device Objects. It must register callbacks (Interrupt Service Routines and Deferred Procedure Calls) with Windows. Finally it must initialize the PCI LVDS 8R board.

IO Controls

The driver uses IO Control calls (IOCTLs) to configure the device. IOCTLs refer to a single Device Object in the driver, which controls a single channel. IOCTLs are called using the Win32 function DeviceIoControl(), and passing in the handle to the device opened with CreateFile(). IOCTLs generally have input parameters, output parameters, or both. Often a custom structure is used.

IOCTL_LV8R_GET_STATUS

Function: Returns status of a specified channel.
Input: none
Output: LV8R_STATUS
Notes: Returns Status information for a given channel obtained from the DMA Status register, the Address Generator SDRAM Status registers, and the FE X Counter Read-back register. The LV8R_STATUS structure returned contains three ULONG fields: CardID, ChannelCount, and ChannelState. The CardID field will contain bits 23-16 of the DMA Status register. These bits reflect the settings of the user-defined dip-switch on the board. The ChannelCount field will contain bits 26-0 of the FE X Counter Read-back register for the specified channel. ChannelState will contain information about the state of the channel and board FIFOs. See the definition of LV8R_STATUS for more information.

IOCTL_LV8R_SET_MEMORY_CONFIG

Function: Sets the specified channel's memory configuration.
Input: LV8R_MEMORY
Output: None
Notes: Sets the Address Generator memory configuration for a given channel. The LV8R_MEMORY input structure contains two ULONG fields: Length and StartOffset. The Length field is used to set the Address Generator SDRAM Length register for the specified channel. The StartOffset field is used to set the Address Generator SDRAM Start Address register for the specified channel.

IOCTL_LV8R_GET_MEMORY_CONFIG

Function: Retrieves the memory configuration for all channels.
Input: None.
Output: LV8R_MEMORY_LAYOUT
Notes: Retrieves the Address Generator memory configurations for all channels. The LV8R_MEMORY_LAYOUT output structure contains an array of 8 LV8R_MEMORY structures. These correspond to the 8 channels on the board. The Length field in each LV8R_MEMORY output structure will be bits 24-0 of the Address Generator SDRAM Length register of the appropriate channel. The StartOffset field will be bits 24-0 of the Address Generator SDRAM Start Address register for the appropriate channel.

IOCTL_LV8R_SET_CHANNEL_CONFIG

Function: Sets up the FE TAG DEF register for the appropriate channel.
Input: LV8R_CHANNEL_CONFIG
Output: None
Notes: Sets up the specified channel configuration by setting up the specified channel's FE Tag Bit Definition register. The input struct contains a UCHAR field and several bool fields. The UCHAR field, Tag, is used to set the Tag Start and Save bits (bits 7-0). For more information on the Tag field see the definition of LV8R_CHANNEL_CONFIG structure. The bool fields are used to set the other bits in the register. The Parity field is used to set the Channel Parity Odd/Even bit (bit 8). The ParityEnable field is used to set the Channel Parity On/Off bit (bit 9). The ClockSelect field is used to set the ClkSel bit (bit 12). The TagMask field is used to set the Tag Mask bit (bit 13). The DeSerializeEnable field is used to set the CH A/B De-serializer enable bit (bit 15). The ContinuousMode field is used to set the Continuous mode bit (bit 11). The CountEnable field is used to set the Count Enable bit (bit 14).

IOCTL_LV8R_GET_CHANNEL_CONFIG

Function: Retrieves information from the specified channel's FE TAG DEF register
Input: None
Output: LV8R_CHANNEL_CONFIG
Notes: Retrieves information from the specified channel's FE Tag Bit Definition register. The output structure contains a UCHAR field and several bool fields. The UCHAR field, Tag, is used to retrieve the Tag Start and Save bits (bits 7-0). For more information on the Tag field see the definition of LV8R_CHANNEL_CONFIG structure. The bool fields are used to retrieve the other bits in the register. The Parity field is used to retrieve the Parity Channel Odd/Even bit (bit 8). The ParityEnable field is used to retrieve the Parity Channel On/Off bit (bit 9). The ClockSelect field is used to retrieve the Clk Sel bit (bit 12). The TagMask field is used to retrieve the Tag Mask bit (bit 13). The DeSerializeEnable field is used to retrieve the CH A/B De-serializer enable bit (bit 15). The ContinuousMode field is used to retrieve the continuous mode bit (bit 11). The CountEnable field is used to retrieve the Count Enable bit (bit 14).

IOCTL_LV8R_SET_TRIGGER_MODE

Function: Sets the configuration of the driver for ReadFile calls
Input: LV8R_TRIGGER_CONFIG
Output: None
Notes: Sets up the driver for a ReadFile. Also sets the FE XYZ Stop registers for the specified channel. Also determines if the "Load" bit in the Address Generator SDRAM Control register is set for the specified channel when a capture takes place. The input structure has five fields. The Mode field is an LV8R_TRIGGER_MODE, and determines the acquisition mode of the driver. See the definition of LV8R_TRIGGER_MODE for more details. The Length field is used to set the appropriate channels FE X Stop register. The Skip field is used to set the FE Y Stop register. The Repeat field is used to set the FE Z Stop register. The ResetMemoryCounter field is used to set a software variable. This variable is used to determine the state of the Address Generator SDRAM Control register Load bit (bit 0) when an acquisition is started on the specified channel (See definition of IOCTL_LV8R_CHANNEL_START_ACQ for more details). If the ResetMemoryCounter field is TRUE then the Load bit (bit 0) will be set to 0. If the ResetMemoryCounter field is FALSE then the Load bit (bit 0) will be set to 1.

IOCTL_LV8R_GET_TRIGGER_MODE

Function: Retrieves the configuration of the driver for ReadFile calls.
Input: None
Output: LV8R_TRIGGER_CONFIG
Notes: Retrieves the settings of the driver for a ReadFile. Also retrieves the FE XYZ Stop registers for the specified channel. Also determines if the "Load" bit in the Address Generator SDRAM Control register is set for the specified channel when a capture takes place. The output structure has five fields. The Mode field is an LV8R_TRIGGER_MODE, and determines the acquisition mode of the driver. See the definition of LV8R_TRIGGER_MODE for more details. The Length field is used to retrieve the appropriate channels FE X Stop register. The Skip field is used to retrieve the FE Y Stop register. The Repeat field is used to retrieve the FE Z Stop register. The ResetMemoryCounter field is used to retrieve a software variable. This variable is used to determine the state of the Address Generator SDRAM Control register Load bit (bit 0) when an acquisition is started on the specified channel (See definition of IOCTL_LV8R_CHANNEL_START_ACQ for more details). If the ResetMemoryCounter field is TRUE then the Load bit (bit 0) will be set to 0. If the ResetMemoryCounter field is FALSE then the Load bit (bit 0) will be set to 1.

IOCTL_LV8R_FILL_RAM

Function: Fills the specified channels memory with a pattern.
Input: ULONG
Output: None
Notes: Fills the specified Channels memory (determined by the channels Address Generator SDRAM Start Address and Length registers) with the input ULONG pattern. Uses the FE Data Holding Register for the appropriate channel, selects the PCI clock, starts the channel, and waits for the channel to complete in order to fill the channel's memory. In order to not block the CPU when polling, this IOCTL uses a system work item to poll the channels done bit at IRQL PASSIVE_LEVEL. This IOCTL could fail if another FillRam is taking place or if for some other reason the driver cannot start a system work item. No acquisitions should be taking place when this IOCTL is called. If any acquisitions are taking place, they will be stopped. This IOCTL will reset the channel configuration and the FE X Stop settings, so IOCTL_LV8R_SET_CHANNEL_CONFIG and IOCTL_LV8R_SET_TRIGGER_MODE should be again called after this IOCTL is complete.

IOCTL_LV8R_SELF_TEST

Function: Not intended to be used.
Input: None
Output: None
Notes: This IOCTL is not intended for use. It is intended for use by the SELFTEST C Library released with the driver. This IOCTL fills the channels RAM with an incrementing pattern from the FE counter. This IOCTL only works on channel 0. This IOCTL does not use a system work item to poll, so will hang the system until complete (maximum of about 3-4 seconds, depending on the size of the channels memory configuration).

IOCTL_LV8R_STREAM_POSITION

Function: Determines approximately where the DMA is working during a ReadFile call in STREAMING_ACQUIRE mode.
Input: LV8R_STREAM_SET_INFO
Output: LV8R_STREAM_INFO
Notes: Provides the application with information and notifications about where the DMA is currently transferring data. This IOCTL will only work if the channel is in STREAMING_ACQUIRE mode. The LV8R_STREAM_SET_INFO input struct contains two fields. If the bool "Wait" field is set to FALSE, the IOCTL will complete immediately. If the Wait field is set to TRUE the IOCTL will complete when the buffer half specified in the LV8R_SAMODE_BUFFER_HALF "SetPosition" field is completed. The LV8R_STREAM_INFO output struct contains three fields. The LV8R_SAMODE_BUFFER_HALF "Position" field represents the half of the application buffer that the DMA is currently working on. The ULONG "HalfwayOffset"
Field is the exact offset in the application buffer where the driver defined Second Half begins. The ULONG "Total" field is the total amount transferred by the current ReadFile call.

IOCTL_LV8R_SET_READ_OFFSET

Function: Sets the starting offset in SDRAM for a ReadFile in LOCAL_ACQUIRE mode
Input: ULONG
Output: None
Notes: Sets a software variable that will be used to set the Address Generator SDRAM Start Address register at the beginning of a ReadFile call in LOCAL_ACQUIRE mode. This IOCTL will succeed only if called on channel 0 or channel 4.

IOCTL_LV8R_GET_READ_OFFSET

Function: Retrieves the starting offset in SDRAM for a ReadFile in LOCAL_ACQUIRE mode.
Input: None
Output: ULONG
Notes: Retrieves the software variable that is used to set the Address Generator SDRAM Start Address register at the beginning of a ReadFile call in LOCAL_ACQUIRE mode. This IOCTL will succeed only if called on channel 0 or channel 4.

IOCTL_LV8R_SET_TIME_OUT

Function: Sets a system timer that cancels a ReadFile call when it expires.
Input: ULONG
Output: None
Notes: This function is used to set a system timer that will cancel a ReadFile after a certain amount of time. The ULONG input is the amount of time, in Milliseconds, after which the timer will expire. The timer is started when a ReadFile call is made, and is canceled when a ReadFile call is completed successfully. If the time limit expires, the timer will cancel the Read IO Request. The driver's timeout value is set to zero when the driver is started. If the timeout value is zero, a ReadFile call will not start a timer, and the timeout value effectively becomes infinite. Set the timeout to 0 if you do not want a timer to be invoked during a ReadFile IO Request.

IOCTL_LV8R_GET_DATA_COUNTER

Function: Retrieves bits 26-0 of the FE X Counter Read-back register
Input: None
Output: ULONG
Notes: Retrieves the FE X Counter Read-back register for the appropriate channel. Only bits 26-0 are valid.

IOCTL_LV8R_CHANNEL_STOP_ACQ

Function: Stops a channel acquisition in LOCAL_ACQUIRE mode. Sets a flag to complete ReadFile IO Request in STREAMING_ACQUIRE mode.
Input: None
Output: LV8R_STATUS
Notes: In LOCAL_ACQUIRE mode, this function is used to stop the current acquisition taking place on the given channel. It will stop the acquisition by reading the FE Tag Bit Definition register, setting the "Start Channel" bit (bit 10) to zero, and writing back the result to the appropriate FE Tag Bit Definition register. Also reads the Address Generator SDRAM Control register "Start for channel x" bit (bit 7) to determine if the channel's transfer has been completed (The Address Generator Length requirement is met). If the channels transfer has not been completed, this function will write the value OxFFFFFFFF to the appropriate FE Channel Done register. This is intended to force the Address Generator to read the remaining data from the input FIFO into the channel's designated memory. In STREAMING_ACQUIRE mode, this function sets a flag to complete the currently running ReadFile IO Request. The ReadFile IO Request will then complete successfully when the DMA chip reaches the middle or the end of the user buffer. This function does nothing in

STREAMING_ACQUIRE_ONESHOT mode. This function currently does not return valid data in the output LV8R_STATUS structure.

IOCTL_LV8R_CHANNEL_START_ACQ

Function: Starts an acquisition on the appropriate channel in LOCAL_ACQUIRE mode
Input: None
Output: LV8R_STATUS
Notes: In LOCAL_ACQUIRE mode, this function will start an acquisition on the given channel. It will start the acquisition through three steps. It will reset the DMA Status register "Done" bit for the appropriate channel. It will start the Address Generator by setting the "Start for channel x" bit (bit 7) and the "IO" bit (bit 2) to one for the appropriate channel. The function will then read the FE Tag Bit Definition register, setting the "Start Channel" bit (bit 10) to one, and writing back the result to the appropriate FE Tag Bit Definition register.

IOCTL_LV8R_RESET_ALL_INPUT_FIFOS

Function: Resets all 8 input FIFOs
Input: None
Output: None
Notes: This function resets all 8 input FIFOs. In order to accomplish this, the function sets all FE Tag Bit Definition registers "Clk Sel" bits (bit 12) to one. It then resets the input FIFOs by twiddling the DMA Base Control register "Reset_1" bit (bit 1) to zero then to one. The FE Tag Bit Definition registers are then restored to their original values. This function will affect all 8 channels. No Reads, channel acquisitions, or FillRams should be taking place when this function is called.

IOCTL_LV8R_IDENTIFY

Function: Flashes user LED three times
Input: None
Output: None
Notes: This function lights the user LED three times for approximately 0.1 seconds with approximately 150 msec spaces between the flashes. It is used for identification purposes.

Read

LVDS data is read from the device using read driver calls. A read call refers to a single Device Object in the driver, which controls a single channel. Reads are executed using the Win32 function ReadFile() and passing in the handle to the device opened with CreateFile(). ReadFile() takes as an input parameter a pointer to a pre allocated buffer and a DWORD that represents the size of the buffer. ReadFile() takes as an output parameter a pointer to a DWORD that represents the number of bytes read by ReadFile().

ReadFile can be done in three separate modes: LOCAL_ACQUIRE mode, STREAMING_ACQUIRE mode, and STREAMING_ACQUIRE_ONESHOT mode (defined in the _LV8R_TRIGGER_MODE enum in the DDLV8R.H file). The mode of the driver is set using the IOCTL_LV8R_SET_TRIGGER_MODE IOCTL call. The default mode for the driver when it first starts up is LOCAL_ACQUIRE mode. The mode determines from what source the data comes. The mode also determines if the ReadFile call will complete as soon as the buffer is filled. In LOCAL_ACQUIRE mode, a ReadFile() call will read data directly from the SDRAM bank on the device. The ReadFile will complete when the specified buffer is filled or the ReadFile reaches the limit of the SDRAM bank. In STREAMING_ACQUIRE mode, the data is read directly from the channel, bypassing the SDRAM on the device. The ReadFile will complete after an IOCTL_LV8R_CHANNEL_STOP_ACQ IOCTL is passed to the driver. In STREAMING_ACQUIRE_ONESHOT mode, the data is also read directly from the channel, bypassing the SDRAM bank on the device. The ReadFile will complete when the specified buffer is filled.

ReadFile should only be called when no data acquisitions are taking place. When a ReadFile begins, it will stop any currently ongoing transfer on any channel. This includes a ReadFile in STREAMING_ACQUIRE or STREAMING_ACQUIRE_ONESHOT modes. In these modes, an acquisition will be automatically started at the beginning of the ReadFile call.

ReadFile calls can be canceled. ReadFile calls are canceled by calling the CancelIO() Win32 API function. ReadFile calls can also fail because of a timeout. If the channel's timeout value is set and the specified time elapses the ReadFile will automatically cancel. In both cases, the ReadFile will complete with an ERROR_OPERATION_ABORTED error.

LOCAL_ACQUIRE mode

A ReadFile() in LOCAL_ACQUIRE mode is the most basic ReadFile done with the LV8R driver. This ReadFile mode is intended to read data from one of the SDRAM banks on the device. ReadFile in this mode can only be used with channel 0 or channel 4. Use channel 0 to access the 256MB SDRAM

bank used for channels 0-3. Use channel4 to access the 256MB SDRAM bank used for channels 4-7. A ReadFile in this mode can accept a buffer with length anywhere from 4 bytes to 256M. However, the buffer must fit into the physical memory on the system (DMA requirement).

To use this ReadFile mode, first set up the Read Offset. The Read Offset is the offset in the SDRAM bank at which you wish to start your ReadFile. The default Offset when the driver first loads will be zero. Valid values for a Read Offset are between 0 and 0x1ffffff. The ReadFile will begin at the Read Offset and continue filling the buffer until either the buffer is full or the end of the SDRAM bank is reached. If the end of the SDRAM bank is reached, then the ReadFile call will return successfully. The "lpNumberOfBytesRead" ReadFile output parameter will be set to the number of bytes between the Read Offset and the end of the SDRAM bank. A ReadFile call will not ultimately alter the channel's memory configuration. It will temporarily set the Address Generator SDRAM Start and Length registers to appropriate values to complete the ReadFile call, but when the ReadFile call completes these registers will be returned to their original settings.

STREAMING_ACQUIRE mode

A ReadFile in STREAMING_ACQUIRE mode is intended to give the application direct access to one of the input channels, bypassing the SDRAM buffer entirely. This mode is also intended to be continuous. In other words, the ReadFile will not stop when the Applications buffer is filled. It will instead stop when explicitly told to stop. When it reaches the upper limit of the application buffer, the DMA engine will begin again at the beginning of the buffer. In order to successfully complete this type of ReadFile call, the application must make an IOCTL_LV8R_CHANNEL_STOP_ACQ call. This will flag the driver DMA engine to stop when it reaches the next buffer division and complete the IO request. A ReadFile in this mode will work on any of the 8 channels on the PCI LVDS 8R device. A ReadFile in this mode must take a buffer of length greater than 8k and less than 16M.

In STREAMING_ACQUIRE mode, the driver breaks the application buffer up into two halves. The driver does this in order to provide information to the application as to which half the DMA is currently filling. The reasoning behind this design decision is the device will be filling one half of the application buffer while the application is using the data from the other half. The two halves will generally not be equal in size. The exact offset of the second half can be obtained by the IOCTL_LV8R_STREAM_POSITION function. The driver uses the IOCTL_LV8R_STREAM_POSITION function to provide information and notifications to the application about the current position (half) the DMA is filling. The DMA is broken up into discrete

transfers; one transfer for each time the DMA fills one half of the buffer. The DMA will actually stop when it completes a half. The driver will then tell it to begin again after checking for “stop” flags and updating its “position” variables. However, there is a finite amount of time between when the DMA stops (completes the half) and when it is started again. This may result in a loss of small amounts of data from the stream due to the hardware FIFOs overflowing before the DMA is started again. The probability of and amount of data lost depends on the latency of the Interrupt Service Routine. Several factors affect this, including the overall speed of the system and traffic on the PCI bus. Windows NT is not a Real Time Operating System (RTOS), and such offers no guarantees about the latency of its interrupts (the amount of time between when the INT line is asserted and when the ISR begins).

To use ReadFile in STREAMING_ACQUIRE mode, first set up the channel FE Tag Bit Definition register using the IOCTL_LV8R_SET_CHANNEL_CONFIG IOCTL. It is recommended that the “Continuous mode” bit (bit 11) be set. This will allow the stream from the channel to be continuous. If this bit is not set the device will stop the stream when the XYZ values are met (See FE Definitions section in the PCI LVDS 8R device documentation provided by Dynamic Engineering). After the Channel is configured, the ReadFile can be started. At this point in time, the device is expecting input from the channel. Also at this point in time the application can obtain information and notifications from the driver as to which half the DMA is currently filling. This information and notifications can be obtained using the IOCTL_LV8R_STREAM_POSITION function. The application then calls the IOCTL_LV8R_STOP_CHANNEL_ACQ function when the application wants the ReadFile to complete. The ReadFile will complete after the IOCTL_LV8R_STOP_CHANNEL call when it reaches the end of the half it is currently filling. If there is no data coming in from the channel, the ReadFile will never complete. If this is the case the ReadFile must be canceled, or the ReadFile must timeout.

STREAMING_ACQUIRE_ONESHOT mode

A ReadFile in STREAMING_ACQUIRE mode is intended to give the application direct access to one of the input channels, bypassing the SDRAM buffer entirely. This mode is intended to fill up the application buffer once, the complete the IO Request. A ReadFile in this mode will work on any of the 8 channels on the PCI LVDS 8R device. A ReadFile in this mode is limited to a buffer length of less than 16M.

To use ReadFile in STREAMING_ACQUIRE_ONESHOT mode, first set up the FE Tag Bit Definition register using the IOCTL_LV8R_SET_CHANNEL_CONFIG IOCTL. After the Channel is configured, the ReadFile can be started. At this point, the device is expecting input from the channel. The ReadFile will complete when the device has received enough data from the channel to fill the application buffer. If there is no data coming in from the channel, the ReadFile will never complete. If this is the case, the ReadFile must be canceled, or the ReadFile must timeout.

Warranty and Repair

Dynamic Engineering warrants this product to be free from defects under normal use and service and in its original, unmodified condition, for a period of one year from the time of purchase. If the product is found to be defective within the terms of this warranty, Dynamic Engineering's sole responsibility shall be to repair, or at Dynamic Engineering's sole option to replace, the defective product.

Dynamic Engineering's warranty of and liability for defective products is limited to that set forth herein. Dynamic Engineering disclaims and excludes all other product warranties and product liability, expressed or implied, including but not limited to any implied warranties of merchandisability or fitness for a particular purpose or use, liability for negligence in manufacture or shipment of product, liability for injury to persons or property, or for any incidental or consequential damages.

Dynamic Engineering's products are not authorized for use as critical components in life support devices or systems without the express written approval of the president of Dynamic Engineering.

Service Policy

Before returning a product for repair, verify as well as possible that the driver is at fault. The driver has gone through extensive testing and in most cases it will be “cockpit error” rather than an error with the driver. When you are sure or at least willing to pay to have someone help then call the Customer Service Department and arrange to speak with an engineer. We will work with you to determine the cause of the issue. If the issue is one of a defective driver we will correct the problem and provide an updated module(s) to you [no cost]. If the issue is of the customer’s making [anything that is not the driver] the engineering time will be invoiced to customer. Pre-approval may be required in some cases depending on the customer’s invoicing policy.

Out of Warranty Repairs

Out of warranty support will be billed. The current minimum repair charge is $125. An open PO will be required.

For Service Contact:

Customer Service Department
Dynamic Engineering
435 Park Dr.
Ben Lomond, CA 95005
831-336-8891
831-336-3840 fax
support@dyneng.com