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

PMC-BiSerial-S311
Driver Documentation

Revision A
Corresponding Hardware: Revision 1
10-2000-0101

PMC-BiSerial-S311
Bi-directional Serial Data Interface
PMC Module

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

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.

This product has been designed to operate with PMC Module carriers and compatible user-provided equipment. Connection of incompatible hardware is likely to cause serious damage.

Manual Revision A. Revised July 24, 2002.

Table of Contents

List of Figures

no figures in this document

Introduction

The Pb_s311 driver is a Windows NT driver for the PMC-Biserial-S311 board from Dynamic Engineering. This driver can control up to 10 boards in a system. Each PMC-Biserial-S311 board transmits and receives a single channel of serial data using the Northrop Grumman S-311 interface protocol with EIA-RS-485 differential drivers and receivers. A separate "Device Object" controls each PMC-Biserial-S311 board, and a separate handle references each Device Object. IO Control calls (IOCTLs) are used to configure the hardware and ReadFile() and WriteFile() calls are used to transfer data to and from the device over the PCI bus.

A handle can be opened to a specific board 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 Pb_s311 driver, Symbolic Link names are formed as Pb_s311n where n indicates the zero based board number. E.g. the third board is Pb_s3112.

ReadFile() and WriteFile() are used to transfer data to/from a specific board specified by passing the appropriate handle opened via the CreateFile() function call. The amount of data transferred by either of these calls is limited by the FIFO size, which is automatically detected when the driver initializes.

Note

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. For more detailed information on the hardware implementation, refer to the PMC-Biserial-S311 device user manual.

Driver Installation

There are several files provided in each driver drop. These files include Pb_s311.sys, Pb_s311.reg, ddPb_s311.h, Pb_s311test.exe, and Pb_s311test source files.

The Pb_s311.sys file is the binary driver file. In order to install the driver, place this file in your Winnt\system32\drivers directory.

The Pb_s311.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 PB_S311 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 ddPb_s311.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 PMC-Biserial-S311 device. It is not needed by the driver installation.

The Pb_s311test.exe file is a sample Windows NT console application that makes calls into the PB_S311 driver. It is not required during the driver installation.

Driver Startup

There are several tasks the PB_S311 driver must do when it is started. It must scan all possible PCI buses to detect every PMC-Biserial-S311 device in the system. It must create a “Device Object” for every board it finds. 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 PMC-Biserial-S311.

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 board. 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_PB_S311_GET_STATUS

Function: Returns the board status.
Input: none
Output: ULONG
Notes: Returns Status information for a given board obtained from the PB_S311_STAT0 register. This includes FIFO flags indicating the amount of data in the transmit and receive FIFOs and latched interrupt status bits indicating the cause of an interrupt. After the status is read, a value is written back to this register to clear only the specific interrupt conditions that were read. This will insure that no interrupt cause is missed due to being asserted between the read and write cycles. See the bit definitions in the DDPb_s311.h header file for more information.

IOCTL_PB_S311_GET_SW_ID

Function: Returns the user switch value.
Input: none
Output: ULONG
Notes: Returns the value set in the eight-position DIP switch on the PMC-Biserial-S311. Note that only the lower six bits are connected due to pin limitations of the Xilinx FPGA.

IOCTL_PB_S311_SET_CONFIG

Function: Sets the base configuration of the board.
Input: ULONG
Output: none
Notes: Controls the clock source and divisor for determining the transmit reference clock frequency and controls the master interrupt enable. See the bit definitions in the DDPb_s311.h header file for more information.

IOCTL_PB_S311_GET_CONFIG

Function: Returns the base configuration of the board.
Input: none
Output: ULONG
Notes: Returns the base configuration register value, excluding the Force Interrupt and FIFO Id and enable bits. See the bit definitions in the DDPb_s311.h header file for more information.

IOCTL_PB_S311_SET_FIFO_LEVELS

Function: Sets receive almost full and transmit almost empty FIFO levels.
Input: FIFO_LEVELS
Output: none
Notes: Sets the almost full level for the receive FIFO; the value is the number of words below full that the PAF flag becomes asserted. Sets the almost empty level for the transmit FIFO; the value is the number of words above empty for which the PAE flag is asserted. The transmit and receive state machines are stopped by this command, since normal FIFO data accesses are disabled when these level registers are accessed. Values are checked to not exceed the FIFO sizes.

IOCTL_PB_S311_GET_FIFO_LEVELS

Function: Returns receive almost full and transmit almost empty FIFO levels.
Input: none
Output: FIFO_LEVELS
Notes: Returns the almost full level for the receive FIFO and the almost empty level for the transmit FIFO. The transmit and receive state machines are stopped by this command, since normal FIFO data accesses are disabled when these level registers are accessed.

IOCTL_PB_S311_SET_TX_CONFIG

Function: Sets the transmitter configuration of the board.
Input: ULONG
Output: none
Notes: Controls the enabling of the transmit and almost empty Interrupts and whether the ready signal is ignored or processed. See the bit definitions in the DDPb_s311.h header file for more information.

IOCTL_PB_S311_GET_TX_CONFIG

Function: Returns the transmitter configuration of the board.
Input: none
Output: ULONG
Notes: Returns the transmit configuration register value including the start bit. See the bit definitions in the DDPb_s311.h header file for more information.

IOCTL_PB_S311_SET_RX_CONFIG

Function: Sets the receiver configuration of the board.
Input: ULONG
Output: none
Notes: Controls the enabling of the receive, overflow, and almost full interrupts and whether the receive interrupt is asserted on every word received or only words with a specific mode bit value. Also data filtering can be enabled using the same mode bit definition bit. The testmode enable needed for loading the receive FIFO from the PCI bus is also controlled with this IOCTL. See the bit definitions in the DDPb_s311.h header file for more information.

IOCTL_PB_S311_GET_RX_CONFIG

Function: Returns the receiver configuration of the board.
Input: none
Output: ULONG
Notes: Returns the receive configuration register value including the start bit. See the bit definitions in the DDPb_s311.h header file for more information.

IOCTL_PB_S311_SET_TERMINATIONS

Function: Sets the configuration of the driver terminations.
Input: ULONG
Output: none
Notes: Sets the configuration of the terminations for the IO lines. See the bit definitions in the DDPb_s311.h header file for more information.

IOCTL_PB_S311_GET_TERMINATIONS

Function: Returns the configuration of the driver terminations.
Input: none
Output: ULONG
Notes: Returns the configuration of the terminations for the IO lines. See the bit definitions in the DDPb_s311.h header file for more information.

IOCTL_PB_S311_START_TX

Function: Starts the transmitter state machine.
Input: none
Output: none
Notes: Sets the start bit for the transmitter, leaving all other configuration bits the same.

IOCTL_PB_S311_START_RX

Function: Starts the receiver state machine.
Input: none
Output: none
Notes: Sets the start bit for the receiver, leaving all other configuration bits the same.

IOCTL_PB_S311_STOP_TX

Function: Stops the transmitter state machine.
Input: none
Output: ULONG
Notes: Clears the start bit for the transmitter and returns the board Status. This is used to abort a transmission. See the bit definitions in the DDPb_s311.h header file for more information.

IOCTL_PB_S311_STOP_RX

Function: Stops the receiver state machine.
Input: none
Output: ULONG
Notes: Clears the start bit for the receiver and returns the received word count. This will also clear the word count register, although the old count remains latched until the next received word.

IOCTL_PB_S311_GET_RX_COUNT

Function: Returns the number of received words stored.
Input: none
Output: ULONG
Notes: Returns the received word count accumulated since the counter was last reset. If data filtering is enabled, only the words that are stored count towards the cumulative total. The counter is also cleared by this command although the old count will remain until the next word is stored in the FIFO.

IOCTL_PB_S311_PUT_TX_DATA

Function: Loads one data word into the transmit FIFO.
Input: ULONG
Output: none
Notes: Loads a single transmit data word into the transmit FIFO. This IOCTL was used mainly for development until WriteFile() call was implemented.

IOCTL_PB_S311_GET_RX_DATA

Function: Reads one data word from the receive FIFO.
Input: none
Output: ULONG
Notes: Reads a single receive data word from the receive FIFO. This IOCTL was used mainly for development until ReadFile() call was implemented.

IOCTL_PB_S311_GET_TX_DATA

Function: Reads one data word from the transmit FIFO.
Input: none
Output: ULONG
Notes: Used for transmit FIFO loop-back testing.

IOCTL_PB_S311_PUT_RX_DATA

Function: Loads one data word into the receive FIFO.
Input: ULONG
Output: none
Notes: Used for receive FIFO loop-back testing. The testmode bit must be set using the IOCTL_PB_S311_SET_RX_CONFIG call.

IOCTL_PB_S311_RESET_FIFOS

Function: Resets the transmit and receive FIFOs.
Input: none
Output: ULONG
Notes: Resets the transmit and receive FIFOs. This will clear all data and reset the almost full and empty values to the default value of seven. Returns the board status. See the bit definitions in the DDPb_s311.h header file for more information.

IOCTL_PB_S311_FORCE_INTERRUPT

Function: Causes a system interrupt to occur.
Input: none
Output: none
Notes: Causes an interrupt to be asserted on the PCI bus if the master interrupt enable is set. This IOCTL is used for development, to test interrupt processing.

IOCTL_PB_S311_REGISTER_EVENT

Function: Registers an event to be signaled when an interrupt occurs.
Input: Event Handle
Output: none
Notes: The caller creates an event with CreateEvent() and supplies the handle returned from that call as the input to this IOCTL. The driver then obtains a system pointer to the event and signals the event when an interrupt is serviced.

IOCTL_PB_S311_ENABLE_INTERRUPT

Function: Sets the master interrupt enable to true.
Input: none
Output: none
Notes: Sets the master interrupt enable, leaving all other bit values in the base configuration register the same. This IOCTL is used in the user interrupt processing function to re-enable the interrupts after they were disabled in the driver interrupt service routine. This allows that function to enable the interrupts without knowing the particulars of the other configuration bits.

Write

Data to be sent from the transmitter is written to the transmit FIFO using a WriteFile() call. The user supplies the device handle, a pointer to the buffer containing the data, the number of bytes to write, a pointer to a variable to store the amount of data actually transferred, and a pointer to an optional Overlapped structure for performing asynchronous IO. The number of bytes is checked to see if it exceeds the size of the FIFO and if not the command is executed with successive writes to the Tx FIFO port. See Win32 help files for details the of the WriteFile() call.

Read

Received data can be read from the receive FIFO using a ReadFile() call. The user supplies the device handle, a pointer to the buffer to store the data in, the number of bytes to read, a pointer to a variable to store the amount of data actually transferred, and a pointer to an optional Overlapped structure for performing asynchronous IO. The number of bytes is checked to see if it exceeds the size of the FIFO and if not the command is executed with successive reads from the Rx FIFO port. See Win32 help files for the details of the ReadFile() call.

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 the 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