R2D3

LVDS & RS-422A Data Capture Cards

PROGRAMMER'S REFERENCE

Scope

This User Guide covers *R2D3* Data Capture Cards for LVDS and RS-422A data + clock applications.

Part Numbers

R2D3-ASSY-0020
R2D3-ASSY-0040

Document Reference

R2D3-MAN-PR01

Publication Date

22 July 2005

Published by

Somerdata Ltd.
1 Riverside Business Park
St Annes Road, Bristol, BS4 4ED
UK

Sales & Customer Support

Phone: +44 (0)1179-634050
Fax: +44 (0)1173 302929
E-Mail: *sales@somerdata.com*
support@somerdata.com
Website: *www.somerdata.com*

REVISION HISTORY

Issue	Date	Notes
1	22 July 2005	Initial
2	Nov 2007	Reformatted
3	Jan 2011	Linux Kernel V2.6 support added

NOTICES

The information in this User Guide has been carefully checked and is believed to be accurate. However, SomerData assumes no responsibility for any inaccuracies that may be contained in this publication.

In no event will SomerData be liable for direct, indirect, special, exemplary, incidental, or consequential damages resulting from any defect or omission in this manual, even if advised of the possibility of such damages.

In the interest of continued product development, SomerData reserves the right to make improvements in this publication and the products it describes at any time, without notice or obligation.

All product names mentioned herein are used for identification purposes only, and may be the trademarks or registered trademarks of their respective companies.

This page intentionally blank

CONTENTS

1. INTRODUCTION

In this Section

About this User Guide

This document covers the SomerData R2D3 Data Capture Cards for LVDS and RS-422A data + clock applications.

This User Guide is intended for software developers who wish to control the R2D3 card in for the capture, recording, replay or processing of digital signals for monitoring, logging, analysis, simulation or testing applications.

The Programmer's Reference Guide includes details of control, status and alarm commands, programming notes, device driver information and the code samples that are supplied on CD-ROM.

Applicable Versions

R2D3-ASSY-0020	RS-422A interface
R2D3-ASSY-0040	LVDS interface
R2D3-ASSY-0420	RS-422A interface
R2D3-ASSY-0440	LVDS interface

What's in this User Guide

Section 2 – PRODUCT DESCRIPTION is a summary of your card's capabilities, features and applications.

Section 3 – CONTROLLING THE CARD is an overview of the card's control and status mechanisms, explanation of the terminology used in this Guide and a control / status command summary.

Section 4 – COMMANDS describes bit-level detail of commands sent to the card for controlling data transfers, interface settings and status requests.

Section 5 –STATUS AND ALARMS describes bit-level detail of the status and alarms returned by the card for data transfers, interface settings, memory states and alarms.

Section 6 – PCI INTERFACE CONTROL gives information on the operational register addresses for the PCI interface device.

Section 7 – PROGRAMMING NOTES (Windows) provides step-by-step guidance for programmers under Windows NT™ using the recommended device driver kit.

Section 8 – CODE SAMPLES (Windows) describes the sample programs and code that are included on the CD-ROM that was supplied with the R2D3 card.

Section 9 – LINUX DEVICE DRIVER describes installation and operation of the supplied LINUX driver.

Section 10 – COMMAND AND STATUS REFERENCE This provides a bit-level reference list for Control and Status commands.

Section 11 – SUPPORT describes the procedure and contact details for obtaining customer support on this product.

Section 12 – INDEX

Related User Guides

Installation and Operation

R2D3-MAN-S101	*LVDS Data Capture Card*
R2D3-MAN-S102	*RS-422A Data Capture Card*

User Guide Availability

Printed copies of Hardware and Software User Guides are supplied with the original products.

Additional printed copies, including the Programmer's Reference Guide can be supplied on request. Please contact your local supplier or SomerData for ordering details.

Electronic copies (Adobe Acrobat files) are included on the SomerData CD-ROM that is supplied with the original products.

The electronic User Guide library, which also includes product data sheets, can be accessed by selecting by opening the file *Document Index.htm*, or by browsing the *Documents*\ folder for the required document.

Additional and updated copies of the CD-ROM can be supplied on request. Please contact your local supplier or SomerData for ordering details.

Utility Programs

The SomerData CD-ROM includes some useful utility programs

- Remote control application
- File viewer/editor application
- Adobe Acrobat Reader

These programs are supplied "as-is" and with no warranty of any kind whatsoever.

Use of these programs is entirely at your own risk.

SomerData will not be held liable for any losses howsoever incurred,

If you do not agree to these terms, do not attempt to install or use these programs!

2. PRODUCT DESCRIPTION

In this Section

Introduction

R2D3 is a real-time data capture and recording card for PCI-bus Personal Computers and other compatible platforms.

The RS-422A card supports input/output data transfers at rates between 1024 kbits/s and 15 Mbits/s.

The LVDS card supports input/output data transfers at rates between 1024 kbits/s and 60 Mbits/s.

There is sufficient on-board memory to allow user processing time and ensure no loss of data.

An AMCC PCI interface handles all communication required of the PCI bus. The PCI interface conforms to Revision 2.1 of the PCI Local Bus Specification and requires 3.3 Volts DC power from the PCI bus.

R2D3 Data Capture Card

Data Handling

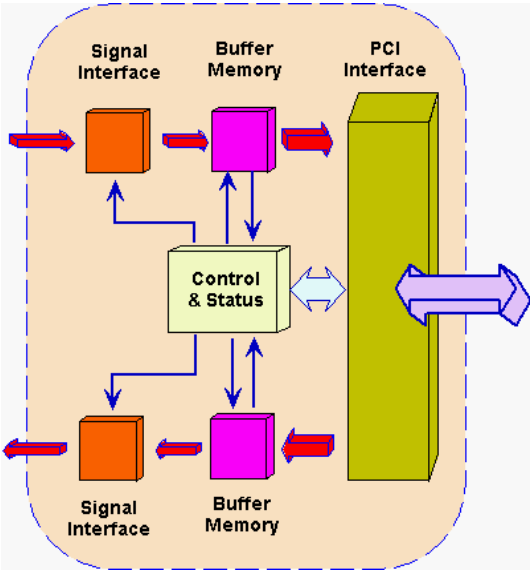
For card inputs, R2D3 accepts a digital bitstream, which is buffered and made available to the user's application software for transfer over the PCI bus.

For card outputs, data streams transferred across the PCI bus to the card are buffered, and output under the control of an external clock source.

On-board Buffer

The card uses on-board memory to buffer data to and from the PCI Bus, with 16 Mbits of SDRAM.

Data-Flow Block Diagram



3. CONTROLLING THE CARD

R2D3 uses a command and status structure based on the mailboxes of an AMCC S5935 PCI interface device.

In this Section

PCI Bus Transfer

The PCI local bus uses a proprietary interface device from AMCC.

A series of control and status commands are implemented, using the mailbox registers of the AMCC PCI interface device.

These allow control and interrogation of the on-board buffer. This buffer is implemented as fast memory cascaded to form the equivalent of a very large FIFO (first-in, first-out) memory.

Data transfers take place by reading data into or writing data out of the Buffer, across the PCI bus.

For fast (burst) transfers, the memory is divided into two areas, with independent status and control of each.

Settable thresholds are provided to indicate the fullness and emptiness of the Buffer.

Interrupt control is provided to enable the card to signal its status, including data states.

Data transfers into and out of the card are in serial form. Transfers across the PCI bus take place in Double Word (DWORD) 32-bit form.

Some status DWORDs can have multiple meaning. The meaning is defined by the offset address. The offset address occupies the 4 bits of the most significant nibble of each DWORD. Command A needs to contain an offset address to request offset status.

Data Structure

The bitstream is transferred in little endian format as a default.

This is normally suitable for all Intel manufactured processors but may be changed by sending a suitable command to the PCI interface device.

Buffer Thresholds

The on board buffer contains 16 Mbits (16777216 bits) of data. Requests to transfer data from the buffer occurs at a predetermined threshold.

This threshold has a default setting of 12 Mbits but can be set by the user.

The threshold is held until a predetermined empty point, which by default is at 4 Mbits but can be set by the user. This allows sufficient time for the PCI latency to be accommodated.

Terminology

The following terms are used with reference to controlling the card.

Memory Configuration

Buffer	*SRAM* = 524288 DWORDS
Upper Cache	XFIFO = 64 DWORDS
Lower Cache	AFIFO = 8 DWORDS

Command Data Structure

Bits 0 - 31	Data

Data Transfers

Input	Record / Write
Output	Replay / Read
DWORD	Data transfers are formatted as 32-bit words

PCI (AMCC) Mailboxes

Outgoing	Commands
Incoming	Status

PCI (AMCC) Bus Registers

INTSCR	Interrupt control / status register
MBEF	Mailbox empty / full status
MRTC	Bus master read transfer count register
MRAR	Bus master read address register
MWTC	Bus master write count register
MWAR	Bus master write address register
MCSR	Master control / status register

Command Summary

Commands are divided into 4 sets of DWORDs, each DWORD being a PCI outgoing mailbox register.

The first relates to interrupt controls and requests for status, the second is the activity control, the third sets up data flow, and the fourth sets up the thresholds.

Commands

Commands may be sent at any time.

Command A	Buffer interrupt controls status request
Command B	stream settings
Command C	Data interface settings
Command D	buffer threshold control

Status

More status is available than will fit into the 4 registers, so an offset scheme is used that changes the meaning of the status mailboxes.

Status A	memory states
Status B	stream settings data interface settings memory states
Status C	input signal alarms firmware variant
Status D	Buffer DWORDS remaining

4. COMMANDS

Introduction

Commands are divided into 4 sets of DWORDs, each DWORD being a PCI outgoing mailbox register.

The first relates to interrupt controls and requests for status, the second is the activity control, the third sets up data flow, and the fourth sets up the thresholds.

Commands may be sent at any time.

Command A	Buffer interrupt controls status request
Command B	stream settings
Command C	Data interface settings
Command D	buffer threshold control

Command A

Command A (the first command DWORD) contains interrupt control and the offset addresses for status requests.

Bit Number	Active State	Data
0 - 15	0	Reserved
16	1	Enable Interrupt on Buffer Full
17	1	Enable Interrupt on Buffer Empty
18	1	Enable Interrupt on Buffer Upper Threshold
19	1	Enable Interrupt on Buffer Lower Threshold
20 - 24	0	Reserved
25	1	1 = Status request
26 - 27		Status request
28 - 31		Offset Address

Interrupts must be enabled prior to the event.

For example, if the Buffer is full, then then Bit-16 (Enable Interrupt on Buffer Full) is set, there will be no subsequent interrupt, until the Buffer goes below full, then becomes full again.

Some status requests are not valid.

For example: Status A offset address greater than 0000_2. These are undefined.

All status should be requested. Only status A offset address 0000_2 can be received under interrupt control, and then only if one or more of bits 16 to 19 are set in Command A.

If the offset address of the command A is greater than 0000_2, the default status will be returned.

If an invalid offset is sent, the default status will be returned.

Default Offsets

Status	Default Offset
A	0
B	0
C	0
D	0

Command B

Command B controls stream setup and options.

Bit Number	Active State	Data
0	1	Active
1 – 3	0	Reserved
4	1	Memory Reset
5 – 7	0	Reserved
8	1	Downstream Monitor
9 – 11	0	Not used
12	1	Input data transfer
13 – 15	0	Not used
16	1	Unframed input
17 – 23	0	Not used
24	0	External Clock (Internal Clock = 1)
25 – 27	0	Not used
28 – 31	0	Reserved

Bit 4 - Memory Reset

Resets both the Buffer and Upper Cache pointers.

Bit 8 - Downstream Monitor

Routes the incoming data through the data interface module and the first stage of logic on the card, and then back out through the data interface module to the output connector.

Downstream Monitor is disabled during output data transfers.

Bit 16 - Unframed Input

Must be set to Unframed for normal operation.

Framed input is used for factory testing only.

Bit 24 - External Clock

That output signal will be at the same frequency as the signal that is being applied to the input connector.

With Internal Clock selected, the output signal will be at 2048 kbits/s.

Command C

Command C sets the hardware state.

Bit Number	Active State	Data
0	1	Output Disconnect
1 - 3	0	Reserved
4	1	Input Disconnect
5 - 7	0	Reserved
8	1	Input clock invert
9 – 11	0	Not used
12	1	Output clock invert
13 – 31	0	Reserved

Bit 0 - Output Disconnect

Sets outbound signal transfers to all-zeros.

Bit 4 - Input Disconnect

Disables input data transfers.

Command D

Command D controls buffer thresholds.

Bit Number	Data
0 – 3	Lower Buffer threshold
4 – 7	Upper Buffer threshold
8 – 31	Reserved

Each nibble represents the highest significant bits of the memory. Thus 1111 will set a threshold representing a full buffer and 0000 will set a threshold that represents an empty buffer.

For input data transfers, it is not unreasonable to set the lower threshold to 0000. However setting the thresholds to the maximum or minimum can cause unpredictable results. The power-on default setting is 1100_2 (75%) and 0100_2 (25%) for upper and lower thresholds respectively.

Setting the lower threshold higher than the upper threshold is not recommended!

This page intentionally blank

5. STATUS AND ALARMS

Introduction

All four mailboxes are used for status, each DWORD being a PCI incoming mailbox register.

There are 10 DWORDs of status capable of being reported. To enable the extra status, offset addresses are used.

These offset addresses are in integral part of the status, which results in the the width of the data in the status being 28 bits.

More status is available than will fit into the 4 registers, so an offset scheme is used that changes the meaning of the status mailboxes.

Status A	memory states
Status B	stream settings data interface settings memory states
Status C	input signal alarms firmware variant
Status D	Buffer DWORDS remaining

Status A

Status A provides Buffer and Cache states.

Bit Number	Active State	Power-on State	Data
0	1	0	Upper Cache full
1	1	1	Upper Cache empty
2	1	0	Upper Cache half-full
3	0	0	Reserved
4	1	0	Buffer Full
5	1	0	Buffer half-full
6	1	1	Buffer empty
7	1	1	Buffer Threshold triggered
8 – 15	0	0	Reserved
16	1	0	Interrupt on Buffer Full enabled
17	1	0	Interrupt on Buffer Empty enabled
18	1	0	Interrupt on Buffer Upper Threshold enabled
19	1	0	Interrupt on Buffer Lower Threshold enabled
20	0	0	Alarm
21 – 27	0	0	Reserved
28 - 31			0000_2 = offset address

Bits 1, 2 and 3 - Upper Cache status

The Cache is split between the PCI interface device and the FPGA. This empty flag indicates that the Cache FIFO in the FPGA is empty.

To determine if the Cache is completely empty it is also necessary to read the PCI interface device's status registers. Refer to the Section – *PCI Interface Control* for further details.

Bit 7 - Buffer Threshold Triggered

For input data transfers, the threshold is triggered when the upper threshold is reached. The threshold is reset when the lower threshold is reached.

Bit 20 - Alarm

This is a logical OR of the data error bits associated with the card variant, and the memory full and empty flags.

The Alarm bit, when set by a data event, will stay set until Statuc C is read. When set by a buffer event, it will stay set until the causing condition clears (e.g. data being read from the memory after a Buffer-full event).

Bits 21 - 27 Reserved

These bits are reserved for diagnostic data, which are assigned for production testing and fault finding. They should not be used by application software.

Status B

Status B provides a reflection of the actioned commands.

There are 3 offset addresses.

Status B - Offset Address 0000

Bit Number	Active State	Power-on State	Data
0	1	0	Active
1 - 3	0	0	Reserved
4	1	0	Buffer reset
5 - 7	0	0	Reserved
8	1	0	Downstream monitor
9 - 11	0	0	Reserved
12	1	0	Input Data transfer
13 - 15	0	0	Reserved
16	1	0	Unframed input
16 - 23	0	0	Reserved
24	0	0	External Clock
25 - 27	0	0	Reserved
28 - 31			0000_2 = offset address

Status B - Offset Address 0001

Bit Number	Active State	Power-on State	Data
0	1	0	Output disconnect
1 - 3	0	0	Reserved
4	1	0	Input disconnect
5 - 7	0	0	Reserved
8	1	0	Input clock invert
9 - 11	0	0	Reserved
12	1	0	Output clock invert
13 - 27	0	0	Reserved
28 - 31			0001_2 = offset address

Status B - Offset Address 0010

Bit Number	Active State	Power-on State	Data
0	1	0	Upper Cache full
1	1	1	Upper Cache empty
2	1	0	Upper Cache half-full
3	0	0	Reserved
4	1	0	Buffer Full
5	1	0	Buffer half-full
6	1	1	Buffer empty
7	1	1	Buffer Threshold triggered
8 - 15	0	0	Reserved
16	1	0	Interrupt on Buffer Full enabled
17	1	0	Interrupt on Buffer Empty enabled
18	1	0	Interrupt on Buffer Upper Threshold enabled
19	1	0	Interrupt on Buffer Lower Threshold enabled
20	0	0	Alarm
21 - 27	0	0	Reserved
28 - 31			0010_2 = offset address

Notes are as for Status A.

The whole of this status (excluding offset) is a reflection of Status A to enable this data to be read while Status A is under interrupt control.

Status C

Status C provides a indication of data alarms, and card type.

There are 2 offset addresses.

Status C - Offset Address 0000

Bit Number	Active State	Power-on State	Data
0	1	1	Signal Not Present Alarm (all zeros)
1 - 3	0	0	Reserved
4	1	0	Input Signal Alarm (all ones)
5 - 7	0	0	Reserved
8	1	0	No frame Alignment Word Alarm
9 - 15	0	0	Reserved
16	1	0	Input Clock Alarm <8kHz
17 - 26	0	0	Reserved
27	1	0	Hardware Error
28 - 31			0000_2 = offset address

Bit 0 - Signal Not Present alarm
This alarm indicates that the number of consecutive zeros is 175 bits.

Bit 4 - Input Signal Alarm
This alarm indicates that the number of consecutive ones is 175 bits.

Bit 8 - No frame Alignment Word Alarm
This alarm indicates that frame alignment word has not been detected when the card is set for Framed input. Note that Framed input is used for factory testing only.

Bit 16 - Input Clock Alarm
This alarm indicates that the input signal frequency is less than 8 kHz.

Bit 27 - Hardware Error

This alarm indicates incompatibility between the card's hardware and firmware.
This is a diagnostic aid only and if this Status bit is flagged the power should be removed from the card immediately and the card returned to SomerData.

Status C - Offset Address 0001

RS-422A Card Part Number R2D3-ASSY-0040

RS-422A Card Part Number R2D3-ASSY-0440

(Current Card Versions)

The following applies to RS-422A cards with serial numbers higher than 2003/48/7 (Manufactured from December 2003).

Bit No.	Active State	Power-on State	Data
0 - 3			Card Variant
4 - 15			Card type
16 - 18			Card Serial Number: Sequence Code (LSB = 16)
19 - 27			Card Serial Number: Date Code (LSB = 19)
28 - 31			0001_2 = offset address

This status provides hardware and firmware version control information in the following priority:

Card Type, Variant, Serial Number

Bits-0 to 3 - Card Variant

R2D3-ASSY-0040.4 = 0100
R2D3-ASSY-0440.0 = 0000

Used for hardware changes that do not have an impact on specification or functionality.

Bits-4 to 15 - Card Type

R2D3-ASSY-0040 = 000001000000
R2D3-ASSY-0400 = 010001000000

Identifies the top-level build configuration of the card, defining the data interface, number of streams, connector type, etc.

Bits-16 to 27 - Serial Number

Identifies the serial number of the card - coded to 12 bits.

The Serial Number is based on a Date Code plus a Sequence Code.

Week 45, Year 2002 will return the Date Code *000000000*.

This feature is for use by SomerData application programs.

RS-422A Card Part Number R2D3-ASSY-0040

(Previous Card Version)

The following applies to RS-422A cards with serial numbers lower than 2003/48/7 (manufactured prior to December 2003).

Bit Number	Active State	Power-on State	Data
0 - 3	2	2	0010_2 = Firmware Variant
4 - 7	0	0	0000_2 = Firmware type
8 - 15	0	0	00000000_2 = Interface type
16 - 19	1	1	0001_2 = Number of Streams
20 - 26	0	0	Not Used
27	0	0	Hardware Error
28 - 31			0001_2 = offset address

LVDS Card Part Number R2D3-ASSY-0020

LVDS Card Part Number R2D3-ASSY-0420

The following applies to all LVDS cards.

Bit Number	Active State	Power-on State	Data
0 - 3	3	3	0011_2 = Firmware Variant
4 - 7	0	0	0000_2 = Firmware type
8 - 15	3	3	00000011_2 = Interface type
16 - 19	1	1	0001_2 = Number of Streams
20 - 26	0	0	Not Used
27	0	0	Hardware Error
28 - 31			0001_2 = offset address

This status provides hardware and firmware version control information in the following priority:

Firmware Variant, Firmware Type, Interface Type, Number of Streams

Bits-0 to 3 - Firmware Variant
R2D3-ASSY-0020 = 0011
R2D3-ASSY-0420 = 0100

Bits-4 to 7 - Firmware Type
R2D3-ASSY-0020 = 0000
R2D3-ASSY-0420 = 0000

Bits-8 to 15 – Interface Type
R2D3-ASSY-0020 = 00000011
R2D3-ASSY-0420 = 00000011

Bits-16 to 19 – Number of Streams
R2D3-ASSY-0040 = 0001
R2D3-ASSY-0440 = 0001

Bit-27 – Hardware Error

This alarm indicates incompatibility between the card's hardware and firmware.

This is a diagnostic aid only and if this Status bit is flagged the power should be removed from the card immediately and the card returned to SomerData.

Status D

Status D provides a indication of words remaining in the Buffer.

Status D - Offset Address 0000

Bit Number	Active State	Power-on State	Description
0 - 17	1	0	Binary representation of the number of DWORDs remaining in the Buffer
18 - 27	0	0	Reserved
28 - 31			0000_2 = offset address

6. PCI INTERFACE CONTROL

For detailed register information, it is recommended that programmers refer to the *AMCC S5935 Data Book, a* copy of which is included on the SomerData CD-ROM in the *\Documents* folder.

In this Section

PCI Configuration

Operational register addresses.

There are 16 operational registers that are mapped as 16 consecutive DWORD registers located at the address space specified by the base address register 0.

The address offset and abbreviation columns refer to those of the AMCC PCI interface device, the register names are those of either the AMCC PCI interface device, or the terminology used for controlling the whole card (as defined in the main text).

Address Offset	Abbreviation	Register Name
00	OMB1	COMMAND A
04	OMB2	COMMAND B
08	OMB3	COMMAND C
0C	OMB4	COMMAND D
10	IMB1	STATUS A
14	IMB2	STATUS B
18	IMB3	STATUS C
1C	IMB4	STATUS D
20	FIFO	DATA PORT
24	MWAR	Master Write Address Register
28	MWTC	Master Write Transfer Count Register
2C	RRAR	Master Read Address Register
30	MRTC	Master Read Transfer Count Register
34	MBEF	Mailbox Empty/Full Status Register
38	INTCSR	Interrupt Control/Status Register.
3C	MCSR	Bus Master Control/Status Register.

Refer to the AMCC PCI Products Data Book for bit by bit detail of these registers.

The following is the status reported in the MCSR:

Bit	Status
0	AFIFO from PCI is Full
1	AFIFO from PCI is half Empty (or less)
2	AFIFO from PCI is Empty
3	AFIFO to PCI is Full
4	AFIFO to PCI is Half Full (or greater)
5	AFIFO to PCI is Empty
6	PCI to Card, Transfer count is Zero
7	PCI from Card, Transfer count is Zero

Initially the AFIFO (Lower Cache) should be empty in both directions, and the transfer counts should be zero, therefore this status should be = $E6_{16}$.

A full AFIFO on input data transfers will give = DE_{16} or $5E_{16}$.

An almost full AFIFO on input data transfers will give = $D6_{16}$ or 56_{16}.

An empty AFIFO on input data transfers will give = $E6_{16}$ or 66_{16}.

A full AFIFO on output data transfers will give = $E1_{16}$ or $A1_{16}$.

An almost empty AFIFO on output data transfers will give = $E2_{16}$ or $A2_{16}$.

An empty FIFO on output data transfers will give = $E6_{16}$ or $A6_{16}$.

Beware of other status values, they either indicate an error, or that something has not been reset, for example the FIFO status flags (bits 25 and 26). Both FIFOs full will give 19_{16}, 59_{16}, 99_{16} or even $D9_{16}$.

AMCC Bus Registers - Quick Reference

MCSR – Master Control / Status Register

Bit Number	Type	Function	Comment
0	RO	PCI to add-on FIFO full	Lower FIFO status - output data transfers
1	RO	PCI to add-on FIFO 4+ spaces	Lower FIFO status - output data transfers
2	RO	PCI to add-on FIFO empty	Lower FIFO status - output data transfers
3	RO	Add-on to PCI FIFO full	Lower FIFO status - input data transfers
4	RO	Add-on to PCI FIFO 4+ spaces	Lower FIFO status - input data transfers
5	RO	Add-on to PCI FIFO empty	Lower FIFO status - input data transfers
6	RO	PCI to add-on transfer count = 0	Output data transfers
7	RO	Add-on to PCI transfer count = 0	Input data transfers
8	RW	Write vs Read priority	
9	RW	Write FIFO management scheme	Input data transfers
10	RW	Write transfer enable	Input data transfers
11		0	
12	RW	Read vs Write priority	
13	RW	Read FIFO management scheme	Output data transfers
14	RW	Read transfer enable	Output data transfers
15		0	
16 - 23		NV operation address / data	
24	RWC	Reset controls	Add-on reset
25	RWC	Reset controls	PCI to add-on FIFO status flags reset
26	RWC	Reset controls	Add-on to PCI FIFO status flags reset
27	RWC	Reset controls	Mailbox flags reset
28			
29 - 31		nv RAM access control	

INTCSR – Interrupt Control / Status Register

Bit Number	Type	Function	Comment
0 - 1	RW	Outgoing Mailbox Byte number	Command mailboxes
2 - 3	RW	Outgoing Mailbox number	Command mailboxes
4	RW	Enable Interrupt	Command mailboxes
5 – 7	RO	Not used	
8 - 9	RW	Incoming Mailbox Byte number	Status mailboxes
10 - 11	RW	Incoming Mailbox number	Status mailboxes
12	RW	Enable Interrupt	Status mailboxes
13	RO	Not used	
14	RWC	Interrupt on Write Transfer complete	Input data transfers
15	RW	Interrupt on Read Transfer complete	Output data transfers
16	RWC	Outgoing Mailbox interrupt	Command mailboxes
17	RWC	Incoming Mailbox interrupt	Status mailboxes
18	RWC	Write Transfer complete	Input data transfers
19	RWC	Read Transfer complete	Output data transfers
20	RWC	Master Abort	
21	RWC	Target Abort	
22	RO	Not used	
23	RO	Interrupt Asserted	
24 – 25	RO	Endian Control	
26 – 27	RO	Lower FIFO Advance Control PCI interface	
28 – 29	RO	Lower FIFO Advance Control add-in interface	
30	RO	Inbound (card output) Lower FIFO toggle	
31	RO	Outbound (card input) Lower FIFO toggle	

Bits 24 / 25 – Endian Control
- 00 = no conversion (default)
- 01 = 16-bit endian
- 10 = 32-bit endian
- 11 = 64-bit endian

Bits 26 / 27 and Bits 28 / 29 – Lower FIFO Advance Control
- 00 = Byte 0
- 01 = Byte 1
- 10 = Byte 2
- 11 = Byte 3

Bit 30 and Bit 31 – Lower FIFO Toggle
- 0 = Bytes 0-3
- 1 = Bytes 4-7

MBEF – Mailbox Empty / Full Status

Bit Number	Type	Function	Comment
0 - 3	RO	Outgoing Mailbox 1	Command A
4 – 7	RO	Outgoing Mailbox 2	Command B
8 – 11	RO	Outgoing Mailbox 3	Command C
12 - 15	RO	Outgoing Mailbox 4	Command D
16 - 19	RO	Incoming Mailbox 1	Status A
20 – 23	RO	Incoming Mailbox 2	Status B
24 – 27	RO	Incoming Mailbox 3	Status C
28 - 31	RO	Incoming Mailbox 4	Status D

MWAR – Bus Master Write Address Register

Bit Number	Type	Function	Comment
0 - 1	RO	DWord address	
2 - 31	RW	Write transfer address	Input data transfers

MWTC – Bus Master Write Transfer Count Register

Bit Number	Type	Function	Comment
0 - 25	RW	Write transfer count	Input data transfers
26 - 31	RO	Reserved	

MRAR – Bus Master Read Address Register

Bit Number	Type	Function	Comment
0 - 1	RO	DWord address	
2 - 31	RW	Read transfer address	Output data transfers

MRTC – Bus Master Read Transfer Count Register

Bit Number	Type	Function	Comment
0 - 25	RW	Read transfer count	Output data transfers
26 - 31	RO	Reserved	

This page intentionally blank

7. PROGRAMMING NOTES (Windows)

In this Section

Device Driver

Windows

SomerData recommends the use of the WinRT™ Device Driver Kit (V3.5 or higher for Windows NT, V4.0 or higher for Windows 2000, V4.1 for Windows XP).

This is a toolkit that allows software developers to control hardware devices from Win32 applications.

The tools allow port I/O, memory I/O and interrupts to be handled directly from C or C++ code without using the DDK (Device Driver Kit).

WinRT information is included on the SomerData CD-ROM in the *Documents*\ folder.

The SomerData CD-ROM includes code examples for use with the WinRT device driver kit. See Section – *CODE SAMPLES (WINDOWS)*

Programming the Card – Windows NT

The R2D3 card uses the AMCC S5935 PCI bus master control device to provide interfacing between the PCI bus and R2D3 card operation.

The WinRT™ device driver development kit provides a useful way of interfacing the AMCC device to the NT kernel without the need for a bespoke device driver.

Depending on the operating system, the following WinRT™ Device Driver Kit versions are required:

Windows NT™	V3.5 or higher
Windows 2000™	V4.0 or higher
Windows XP™	V4.1

User applications can make indirect calls to the NT kernel, and the WinRT development tools provide useful debugging and tracing functions to enable the programmer to follow the progress of code in near real-time.

WinRT Functions

A number of pre-defined functions are provided for use with the WinRT device driver.

These provide application access to commonly used control functions of the AMCC PCI controller, as well as the ability to control and move data.

These should all be treated as samples, although all have been tested with the R2D3 card. For more details, including debugging and modification information, refer to the WinRT driver development kit and manual.

Working programs can be built using these functions to control the R2D3 card which together with the supplied header file R2D3Ctl.h provide the programmer with useful shortcuts.

AMCC Control and Setup

DWORD OpenPciAmcc(void)

Checks for the presence of the WinRT driver device, initialises and sets up the common buffer used for DMA transfers, creates interrupt events and initialises the interrupt service routine.

DWORD ClosePciAmcc(void)

Stops interrupts, frees resources and closes handles. Used before exiting the program.

DWORD WriteMailBox(DWORD Address, DWORD Data)

Sets up the Mailbox interrupt register, writes data (Data) to a mailbox (at Address), then waits for an interrupt to show that the mailbox has been acknowledged.

DWORD ReadMailBox(DWORD Address)

Sets up the Mailbox interrupt register, waits for an incoming interrupt indicating that a mailbox (at Address) is ready to be read then fetches the value.

DWORD DumpAmccRegister(void)

Used to provide global access to fetch and print the state of the AMCC registers.

DWORD GetAmccRegisters(PAMCC_REGISTERS pAmccRegisters)

Utility to fetch the state of all AMCC registers including Mailboxes.

VOID PrintAmccRegisters(PAMCC_REGISTERS pAmccRegisters)

Used to print/display the contents of the AMCC registers. User code is required to direct output to the appropriate display device.

DWORD ResetAmcc(void)

Resets the AMCC MasterControl Status Register FIFO flags, and disables all interrupts.

AMCC Interrupt Control

DWORD DisableAmccInterrupts(void)

Disables interrupt processing, can be used to suspend interrupts while data is being processed.

DWORD EnableAmccInterrupts(DWORD InterruptEnable)

Enables interrupts, usually used to pass AMCC parameters for mailbox/write/read interrupts, etc.

VOID StopInterrupt(void)

Stops the deferred interrupt process and frees resources. Usually used at the close of an application program.

VOID RejectInterrupt(void)

Used by the interrupt service routine to reject unwanted interrupts.

BOOLEAN SetUpAmccIsr(void)

Sets up a repeating interrupt service routine which runs in background. The routine services all hardware interrupts generated by the AMCC device and sets events to signal the calling application that they have occurred.

AMCC Data Transfer

VOID DumpBuffer(PUCHAR pBuffer, DWORD Length)

Utility to display the contents of the common DMA buffer. This will normally need to be re-written to accommodate the appropriate display method.

DWORD WriteToMWTC(DWORD val)

Sets transfer value for AMCC write operations. Note this function is not normally called directly but is used as part of data send/request routines.

DWORD DmaReadFifo(DWORD Length);

Sets up the AMCC for DMA reading (Length) Bytes, then waits for an interrupt to show that the expected transfer is complete.

DWORD DmaWriteFifo(DWORD Length)

Sets up the AMCC for DMA writing (Length) Bytes, then waits for an interrupt to show that the expected transfer is complete.

Program Development Sequence

Sample code, provided as a Borland CBuilder5 project, demonstrates most of the functionality of the card and is a good starting point for program development.

A normal development sequence would be:

Establish the WinRT device driver and operating parameters on the development system. This can be done programmatically (see WinRT manual) or more quickly by using the WinRT console program.

Open the WinRT device driver and establish that communication has been established. Verify the size of the common (DMA) buffer.

Establish the Interrupt service routine, if required. Note that the card can be operated in polling mode at the expense of processor overhead.

Establish the initial conditions for the card (operational mode, input/output connections, etc.)

Establish a status gathering routine to read registers and mailboxes. It is often useful to use a separate thread of operation for this, but be sure to include suitable synchronisation mechanisms.

Write code for required command functions, using the pre-built WinRT AMCC functions as a guide.

A fully-working sample is provided on the SomerData CD-ROM, in the folder *\R2D3\Support\LVDS\,* and can be used as an aid to program development.

Card Initialisation

On power up the card will be in the following initial state: (assumes no signal connected)

Status Registers

Status A = Bits 7, 6 and 1 set all other bits reset

Status B = all Bits reset,
Offset 1 Bit 28 set, all other bits reset

Status C = Bits 0-3, 8-11 set, all others reset
Offset 1 Bit 28 set, bits 0-3, 4-7 and 16-19 depend on firmware code

Status D = Bit 28 set, all other bits reset

No other command initialisation is required.

Programming Essentials

The following notes apply to Borland CBuilder/Delphi and the supplied sample program, but the principles apply to all code development.

If you use multi-threaded operation (see sample), you must:

a) ensure that your code is thread-safe. Use Critical sections to protect global variables, and Events to signal between threads. Make sure that all calls to the main VCL thread are made using the Synchronise method.

b) ensure that there are separate (shared) handles opened to the WinRT device driver for each thread.

c) establish a buffer to copy the contents of the shared DMA buffer (refer to *Optimising for High Transfer Rates* below).

d) check the returned offset addresses to make sure that the status returned is current. During data transfers, status information can be delayed by the card, hence returned information can be 'stale'.

The use of offsets allows more status to be returned than 4 mailboxes allow, but relies on the requested status being returned accurately. During intense data transfer periods, the card may not update status immediately in response to a request.

If the incoming mailbox is read before the status has been updated, the information read will be old.

If your program does not check for offset address accuracy, the data could belong to a previous offset.

Optimising for High Transfer Rates

High data transfer rates (up to 60 Mbits/s) can only usefully be exploited if the PCI platform and operating software are fast enough to sustain such high rates.

WindowsNT enables control of most timings and priority scheduling, but still requires a regular amount of time for internal housekeeping tasks. The in-built data buffer of the R2D3 allows these timing inconsistencies to be smoothed out without imposing system overhead but has limited capability. If the buffer service request is not actioned within a known time, buffer overflow (input data transfers) or buffer underflow (output data transfers) can occur.

The amount of time required depends on the system configuration, other programs that are running, the intrinsic speed of the platform and, for disk transfer operations, the speed of the disk. The last of these is especially important if large amounts of data are to be transferred to and from the disk. As a minimum, the disk interface should be fast (SCSI or ATA-100) and the disk should be capable of supporting these fast interfaces.

In order to assist the determination of speed requirements, the WinRT development kit includes the capability to record a trace of calls into the driver, with timings.

The supplied sample test program includes Trace statements that can be viewed in the WinRT console program and used to benchmark the system. It is strongly recommended that any controlling software also include trace statements to allow code speed optimisation.

Timing Optimisation Example

As an example of timing calculation, consider the following case at 60 Mbits/s serial input. The platform is 700MHz PIII processor with an ATA-100 disk, running SomerData's sample test program under Windows NT4.

Total R2D3 buffer size = 16777216 bits

Time to fill at 60 Mbits/s = Capacity / Rate = 279 ms

therefore

Time to half-fill = 139 ms

For sustained data transfer rate of 60 Mbits/s, the total time to transfer data from the card, process and write to disk must never exceed an average of the half-fill time. To prevent buffer overflowing, any individual transfer that takes longer must be compensated for by a transfer that is shorter by the excess amount.

Trace Buffer Examples

The following example of an actual trace buffer serves to illustrate. Additional comments are added in the rightmost column.

This sequence represents the process of setting up an interrupt to wait for a mailbox to indicate that a threshold has been reached, then setting up a DMA bus transfer, in this case of 262144 Bytes, in to a common buffer.

The AMCC S5935 sets an interrupt to show that all the requested data has been transferred. This cycle can be repeated as many times as is necessary to maintain an average data buffer position of approximately halfway.

Note that in this case it has taken approximately 17 ms to transfer data from the card to the buffer.

Time Index (100ns)	Time Delta (100ns)	Error Mask	Value	Message	Comment
42270279337	100	0x4	0x0	Enter Deferred Processing	Interrupt received
42270279412	75	0x4	0x20000	Processing Interrupt bits 0x%X	Determine that this is a mailbox interrupt
42270279496	83	0x4	0x0	Exit Deferred Processing	Leave service routine
42270375910	96414	0x4	0xC410	Read MailBox register 0x%X	Read mailbox to get data
42270375952	41	0x4	0x50400A1	Read data 0X%X	Determine that threshold has been set
42270376212	259	0x4	0x4000	EnableAmccInterrupts(): Wrote ICSR with 0x%X	Enable Write Transfer Complete interrupt
42270495255	118632	0x4	0x40000	Read: Set DMA transfer length 0x%X	Set transfer length (262144 Bytes)
42270495314	58	0x4	0x56D000	Read: Set DMA transfer address 0x%X	Set DMA address
42270495448	134	0x4	0x400	Read: Start DMA transfer, Wrote MCSR 0x%X	Transfer 262144 Bytes
42270674113	178665	0x4	0x0	Enter ISR	Service interrupt
42270674172	58	0x4	0x40000	ISR current Interrupt bits 0x%X	Determine that Write Transfer Complete has interrupted
42270674247	75	0x4	0x0	Exit ISR	End of transfer
42270674364	117	0x4	0x0	Enter Deferred Processing	

The next few sequences are a repeat of the previous, except that the program enters a loop to get 3 further sets of data and transfer it to disk. These loops can be used to determine the speed of processing and disk transfer.

In this case, the disk transfer time is approximately 11 ms.

Note that the other times are quite insignificant in comparison with the time taken to fetch the data and transfer it to disk. For example, the entire interrupt determination and servicing delay is only approximately 40 μs.

Overall, for each loop to receive 262144 Bytes and transfer to disk takes approximately 28 ms. Since there are 4 loops plus other small delays, the entire process is just complete in approximately 116 ms, within the target of 152 ms to empty half a buffer of data.

Clearly this system has some extra capacity and is not likely to fail to complete transfers if the process is interrupted for more than a few milliseconds. However if external delays of more than this are expected (in this case an average of approximately 20 ms), the system may require further upgrading.

Here, the solution might be to employ a faster disk, probably SCSI, and probably to increase the processing capability, either by adding a second processor or increasing the speed of the existing processor.

It can therefore be seen that benchmarking the platform and application software is crucial to obtaining the best transfer performance.

Time Index (100ns)	Time Delta (100ns)	Error Mask	Value	Message	Comment
42270954690	100	0x4	0x0	Exit Deferred Processing	End of previous transfer
42271055195	100504	0x4	0x40000	Read: Set DMA transfer length 0x%X	Disk transfer and processing delay
42271055245	50	0x4	0x56D000	Read: Set DMA transfer address 0x%X	Card transfer as before
42271055396	150	0x4	0x400	Read: Start DMA transfer, Wrote MCSR 0x%X	
42271233877	178480	0x4	0x0	Enter ISR	
42271233927	50	0x4	0x40000	ISR current Interrupt bits 0x%X	
42271234002	75	0x4	0x0	Exit ISR	
42271234111	108	0x4	0x0	Enter Deferred Processing	
42271234195	83	0x4	0x40000	Processing Interrupt bits 0x%X	
42271234296	100	0x4	0x0	Exit Deferred Processing	Transfer from card complete
42271337373	103077	0x4	0x40000	Read: Set DMA transfer length 0x%X	Disk transfer and processing delay

Note that the card's command structure includes the capability to generate a further interrupt if the data buffer becomes full (input data transfers) or empty (output data transfers). This can be used to flag data error conditions to the application, and if necessary to recover from over / underflow situations.

This page intentionally blank

8. CODE SAMPLES (Windows)

In this Section

Introduction

The supplied sample test program demonstrates the capability of the card to accept and provide data.

Test software and code samples are supplied without warranty or support as an aid to software developers and do not represent the normal commercial quality of SomerData software products.

The WinRT™ device driver development kit provides a useful way of interfacing the AMCC device to the NT kernel without the need for a bespoke device driver.

Depending on the operating system, the following WinRT™ Device Driver Kit versions are required:

Windows NT™	V3.5 or higher
Windows 2000™	V4.0 or higher
Windows XP™	V4.1

Sample Program Functions

The program demonstrates the following functions:

use of WinRT device driver to control the AMCC interface

use of interrupts to control data flow

multi-threaded operation

common status tellbacks

input data transfer and store to disk

retrieve from disk and output data transfer

individual register / mailbox reading and writing

How the Sample Program works

The program contains a main screen that is used to display primary status and control functions. The Open button causes the driver to be initialised, and the card initial conditions to be set. Some error checking is included that will identify that the driver is inoperative if not installed correctly.

Note that only limited error checking is provided throughout the sample program. This avoids clutter and allows the program flow to be followed more easily. The program is also broken down into separate functions and lines to enable easier insertion of breakpoints during debugging and also makes more than normal use of global variables.

Events and Threads

The program creates various events and the threads that will be needed for each main function.

Status Display thread

Runs in the background polling the card for status and checking the offset addresses. Status registers are only updated if the requested offset and the returned offsets match.

Interrupt thread

Runs at time critical priority, waiting for events from the WinRT deferred interrupt process, and creates events used by other threads in response.

Buffer thread

Controls the transfer of data to and from the card.

Disk thread

Controls the writing and reading to and from disk storage.

Thread Interoperation

Each thread is started in a permanent loop, controlled by Wait events. Each loop waits for another thread to signal that it is either ready to process data or has finished processing data. This signalling ensures that common buffer areas are not accessed simultaneously.

The interrupt thread runs at TimeCritical priority to ensure that interrupts are serviced quickly. The status thread runs at lowest priority.

The sample displays the device configuration information in the main window. This list item box could also be used to display other information.

Status is generally displayed in the status window, but all registers are also displayed in the main window in hex format.

Static setup information can be changed in the Controls window. These generally set the card environment and modes of operation. Use of global register variables ensures that these can be kept updated if other actions change the status, e.g. writing or reading to/from disk.

If all is operating correctly, the main display will show:

WinRT configuration information

WinRT Version information

Status of incoming mailboxes, outgoing mailboxes and AMCC registers (in hex).

The other menu items will be enabled, allowing selection of the Controls and File menus.

Input Data Transfer and Store to Disk (Record)

Before you can write to disk, you must connect a suitable signal source, open the device (as above), and select a file to record.

The default filename is *C:\\Test.r2d*.

Open the Buffer window. Select a suitable transfer size. The default is 262144 Bytes. This is the amount of data that will be transferred across the PCI bus during each data transfer.

The maximum data size that can be transferred is shown next to the selection box, this represents the DMA buffer size that was allocated during driver start, on NT bootup. This will normally be the DMA buffer size set up in the WinRT console or setup program.

Data can only be transferred in 32-bit (DWORD) increments, so the number you enter must always be divisible by 4. A catch in the program will prevent any other number.

The size of increment required will be a compromise between minimising the latency of data in the card buffer and minimising the PCI bus traffic. The default transfer size setting of 262144 Bytes is suitable for recording data to disk at the highest transfer rate.

There is no check on available disk space, or limit to the amount of data that can be written, so these need to be controlled by the user.

Note that the test program and associated code does not support file sizes greater than 2 GBytes. A label indicates the number of Bytes that have been written to disk.

The sample code includes the writing of a standard SomerData header to allow interchange with other R2D programs. This adds 1024 Bytes to the file. If a pure binary file is required, remove the sections WriteHeader and WriteHeaderEnd.

Retrieve from Disk and Output Data Transfer (Replay)

Reading from disk is similar to writing. You need to open a known *R2D* file, and set transfer size increments as above. The default transfer size setting of 65536 Bytes is suitable for replaying data from disk at the highest transfer rate.

Replay continues in a loop forever until stopped.

With Internal Clock selected, the output signal will be at 2048 kbits/s.

When External Clock is selected, the output signal will be at the same frequency as the signal that is being applied to the input connector.

Displaying Data

You can examine the contents of some parts of the data using the Display option.

This sets the card into Record mode, and fetches data in increments set by the transfer size edit box.

Set the number of lines of data to be displayed, the display will show the hex version of 32 columns x number of lines of beginning of the fetched data.

The display shows 32 characters per line.

Debug information

There are a number of debug messages in the code that are commented out. These can be re-instated to provide information or breakpoints, but will slow operation.

In order to view WinRT debug information, use the Trace function from the WinRT console program. Trace statements are included in the WinRT code sections. These can be removed to speed code further.

Controlling Setups

Hardware selections are available from the Controls window.

Bypass - selects a mode that connects input to output without going through the card buffer. This Downstream Monitor enables system connection integrity checks to be carried out before recording or replaying data.

Input Clock Inversion

Changes the edge on which data is clocked in.

This can be useful if clock and data cables are uneven or unknown length, or the source does not allow clock/data control.

Note that for best performance, clock and data cable should be of the correct impedance and similar propagation delay.

Impedance discontinuities can cause reflections, which will give frequency zones of errored performance.

Bypass (Downstream Monitor) allows this to be checked before recording.

Input Isolation

Prevents data reaching the card.

Framing - is used only for internal test purposes and allows a structured data pattern to be checked.

For user applications, this should always be set to unframed.

Mode - selection is for use with manual register control and is simply a convenient way of changing mode from Record to Play.

Selection during Recording/Replaying is automatic.

Threshold - Slider bar selection of upper and lower thresholds.

An interlock prevents selection of lower thresholds greater than upper thresholds (and upper thresholds smaller than lower thresholds).

Known Limitations

The program does not trap conditions which are illegal combinations, or where manual register control conflicts with automatic settings.

The program does not trap buffer overflow of underflow conditions.

User code would normally expect to deal with these conditions, but the circumstances will vary with each application.

Note that interrupts can be enabled to signal that these conditions have occurred.

The program does not support recording after replaying. This is a known limitation. Exit the program and restart.

Some program hang-ups can be fixed by using the various reset buttons, otherwise re-starting will be necessary.

9. LINUX DEVICE DRIVER

In this Section

Introduction

Linux device drivers are included on the SomerData CD-ROM in the folder:

*R2D3**Support**Linux*\

The SomerData CD-ROM includes installation notes, header file and source code.

One driver supports input data transfers and another supports output data transfers. Both drivers support the single-stream LVDS card as well as the dual-stream R2D3 E1/HDB3 cards.

The following sections only apply to the input data transfer driver. At the time of writing, the output data transfer driver is incomplete, although it is supplied on the CD-ROM.

Please note that fully documenting the driver is beyond the scope of this manual. For details please refer to the source code.

Installation

This device driver can currently only be compiled as a module, Kernel 2.4 and 2.6 are supported in separate source files. Kernel 2.6 support is provided as a NetBeans project, but can be used as stand-alone source witj your own makefile. Module loading varies between kernels for 2. 4:

insmod r2d3.o

or insmod r2d3Serial.ko

for Kernel 2.6

It works with UP and SMP (only tested with SMP). Kernel 2.6 has been tested with Ubuntu 10.10, 2.4 with RedHat Version 7.2.

Device Files

PCI device enumeratiohn and loading changed significantly between kernels 2.4 and 2.6. In 2.4 the device driver dynamically grabs a major device number for the devices.

Thus, you will not know the device number until the driver is loaded, so you cannot create the */dev* files in advance.

You have these options:

- Use the device files which are dynamically created in */proc/r2d3* when the device driver is loaded Create a link from */dev/r2d3* to */proc/r2d3* if you want the device files to be under */dev*
- Use devfs - device files are created automatically in *{devfsroot}/r2d3*
  Currently, *devfs* is "experimental" so use is not recommended
- Create */dev* files at boot time using information in */proc/devices* and */proc/r2d3/status*

- In 2.6 a complete new source tree has been providied and is designated R2D3Serial. On loading, /proc/r2d3Serial/status file is created, access to the driver is via dev/r2d3Serial/

Compilation

Makefiles are provided for both versions. Just type *make* using gcc to compile. .

Operation

When loaded, it dynamically allocates itself a major device numbers and creates device files and diagnostic files under */proc/r2d3*.

Devices are enumerated in order they are found on the PCI device starting from 0. For each device X, the following device files are created:

DMA transfers are supported by writing address locations to the card MWAR/MRAR registers

Further documentation is available by examining the source.

This page intentionally blank

10. COMMAND AND STATUS REFERENCE

This Section provides a bit-level reference list for Control, Status and Alarms.

In this Section

Hex Conversion

The following conversion table may be useful for interpreting the Mailbox information when running the Windows code sample programs.

HEX	f	e	d	c	b	a	9	8	7	6	5	4	3	2	1	0
BINARY	1111	1110	1101	1100	1011	1010	1001	1000	0111	0110	0101	0100	0011	1101	0001	0000
DECIMAL	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0

Command A

Nibble	Bit No.	Active State	Data
7	31		Offset Address
7	30		
7	29		
7	28		
6	27		Status request (00 = A, 01 = B, 10 = C, 11 = D)
6	26		
6	25	1	Status request
6	24	0	Reserved
5	23	0	
5	22	0	
5	21	0	
5	20	0	
4	19	1	Enable Interrupt on Buffer Lower Threshold
4	18	1	Enable Interrupt on Buffer Upper Threshold
4	17	1	Enable Interrupt on Buffer Empty
4	16	1	Enable Interrupt on Buffer Full
3	15	0	Reserved
3	14	0	
3	13	0	
3	12	0	
2	11	0	
2	10	0	
2	9	0	
2	8	0	
1	7	0	
1	6	0	
1	5	0	
1	4	0	
0	3	0	
0	2	0	
0	1	0	
0	0	0	

Command B

Nibble	Bit No.	Active State	Data
7	31		Reserved
7	30		
7	29		
7	28		
6	27		Nor used
6	26		
6	25		
6	24	0	External Clock
5	23		Not used
5	22		
5	21		
5	20		
4	19		
4	18		
4	17		
4	16	1	Unframed input
3	15		Not used
3	14		
3	13		
3	12	1	Input data Transfer
2	11		Not used
2	10		
2	9		
2	8	1	Downstream Monitor
1	7		Reserved
1	6		
1	5		
1	4	1	Memory Reset
0	3		Reserved
0	2		
0	1		
0	0	1	Active

Command C

Nibble	Bit No.	Active State	Data
7	31		Reserved
7	30		
7	29		
7	28		
6	27		
6	26		
6	25		
6	24		
5	23		
5	22		
5	21		
5	20		
4	19		
4	18		
4	17		
4	16		
3	15		
3	14		
3	13		
3	12	1	Output Clock invert
2	11		Not used
2	10		
2	9		
2	8	1	Input Clock invert
1	7		Reserved
1	6		
1	5		
1	4	1	Input disconnect
0	3		Reserved
0	2		
0	1		
0	0	1	Output disconnect

Command D

Nibble	Bit No.	Active State	Data
7	31		Reserved
7	30		
7	29		
7	28		
6	27		
6	26		
6	25		
6	24		
5	23		
5	22		
5	21		
5	20		
4	19		
4	18		
4	17		
4	16		
3	15		
3	14		
3	13		
3	12		
2	11		
2	10		
2	9		
2	8		
1	7		Upper Buffer threshold
1	6		
1	5		
1	4		
0	3		Lower Buffer threshold
0	2		
0	1		
0	0		

Status A

Nibble	Bit No.	Active State	Power-on State	Data
7	31	0	0	Offset Address
7	30	0	0	
7	29	0	0	
7	28	0	0	
6	27	0	0	Reserved
6	26	0	0	
6	25	0	0	
6	24	0	0	
5	23	0	0	
5	22	0	0	
5	21	0	0	
5	20	1	0	Alarm
4	19	1	0	Interrupt on Buffer Lower Threshold enabled
4	18	1	0	Interrupt on Buffer Upper Threshold enabled
4	17	1	0	Interrupt on Buffer Empty enabled
4	16	1	0	Interrupt on Buffer Full enabled
3	15	0	0	Reserved
3	14	0	0	
3	13	0	0	
3	12	0	0	
2	11	0	0	
2	10	0	0	
2	9	0	0	
2	8	0	0	
1	7	1	1	Buffer Threshold triggered
1	6	1	1	Buffer empty
1	5	1	0	Buffer half-full
1	4	1	0	Buffer full
0	3	0	0	Reserved
0	2	1	0	Upper Cache (XFIFO) half-full
0	1	1	1	Upper Cache (XFIFO) empty
0	0	1	0	Upper Cache (XFIFO) full

Status B - Offset 0000

Nibble	Bit No.	Active State	Power-on State	Data
7	31	0	0	Offset Address
7	30	0	0	
7	29	0	0	
7	28	0	0	
6	27	0	0	Reserved
6	26	0	0	
6	25	0	0	
6	24	0	0	External clock
5	23	0	0	Reserved
5	22	0	0	
5	21	0	0	
5	20	0	0	
4	19	0	0	
4	18	0	0	
4	17	0	0	
4	16	1	0	Unframed input
3	15	0	0	Reserved
3	14	0	0	
3	13	0	0	
3	12	1	0	Input Data transfer
2	11	0	0	Reserved
2	10	0	0	
2	9	0	0	
2	8	1	0	Downstream monitor
1	7	0	1	Reserved
1	6	0	1	
1	5	0	0	
1	4	1	0	Buffer reset
0	3	0	0	Reserved
0	2	0	0	
0	1	0	1	
0	0	1	0	Active

Status B - Offset 0001

Nibble	Bit No.	Active State	Power-on State	Data
7	31	0	0	Offset Address
7	30	0	0	
7	29	0	0	
7	28	1	0	
6	27	0	0	Reserved
6	26	0	0	
6	25	0	0	
6	24	0	0	
5	23	0	0	
5	22	0	0	
5	21	0	0	
5	20	0	0	
4	19	0	0	
4	18	0	0	
4	17	0	0	
4	16	0	0	
3	15	0	0	
3	14	0	0	
3	13	0	0	
3	12	1	0	Output Clock invert
2	11	0	0	
2	10	0	0	
2	9	0	0	
2	8	1	0	Input Clock invert
1	7	0	0	
1	6	0	0	
1	5	0	0	
1	4	1	0	Input disconnect
0	3	0	0	
0	2	0	0	
0	1	0	0	
0	0	1	0	Output disconnect

Status B - Offset 0010

Nibble	Bit No.	Active State	Power-on State	Data
7	31	0	0	Offset address
7	30	0	0	
7	29	1	0	
7	28	0	0	
6	27	0	0	Reserved
6	26	0	0	
6	25	0	0	
6	24	0	0	
5	23	0	0	
5	22	0	0	
5	21	0	0	
5	20	0	0	Alarm
4	19	1	0	Interrupt on Buffer Lower Threshold enabled
4	18	1	0	Interrupt on Buffer Upper Threshold enabled
4	17	1	0	Interrupt on Buffer empty enabled
4	16	1	0	Interrupt on Buffer full enabled
3	15	0	0	Reserved
3	14	0	0	
3	13	0	0	
3	12	0	0	
2	11	0	0	
2	10	0	0	
2	9	0	0	
2	8	0	0	
1	7	1	1	Buffer threshold triggered
1	6	1	1	Buffer empty
1	5	1	0	Buffer half-full
1	4	1	0	Buffer full
0	3	0	0	Reserved
0	2	1	0	Upper Cache (XFIFO) half-full
0	1	1	1	Upper Cache (XFIFO) empty
0	0	1	0	Upper Cache (XFIFO) full

Status C - Offset 0000

Nibble	Bit No.	Active State	Power-on State	Data
7	31	0	0	Offset Address
7	30	0	0	
7	29	0	0	
7	28	0	0	
6	27	1	0	Hardware Error
6	26	0	0	Reserved
6	25	0	0	
6	24	0	0	
5	23	0	0	
5	22	0	0	
5	21	0	0	
5	20	0	0	
4	19	0	0	
4	18	0	0	
4	17	0	0	
4	16	1	0	Input Clock Alarm <8kHz
3	15	0	0	Reserved
3	14	0	0	
3	13	0	0	
3	12	0	0	
2	11	0	0	
2	10	0	0	
2	9	0	0	
2	8	1	0	Input Signal no Frame Alignment Word Alarm
1	7	0	0	Reserved
1	6	0	0	
1	5	0	0	
1	4	1	0	Input Signal Alarm (all-ones)
0	3	0	0	Reserved
0	2	0	0	
0	1	0	0	
0	0	1	1	Input Signal Not Present Alarm (all-zeros)

Status C - Offset 0001

Current RS-422A Card Version

The following applies to RS-422A cards with serial numbers higher than 2003/48/7 (manufactured from December 2003).

Nibble	Bit No.	Active State	Power-on State	Data
7	31	0	0	Offset Address
7	30	0	0	
7	29	0	0	
7	28	1	0	
6	27		0	Card Serial Number: Date Code 000000000 = Week 45, Year 2002
6	26		0	
6	25		0	
6	24		0	
5	23		0	
5	22		0	
5	21		0	
5	20		0	
4	19		0	
4	18		0	Card Serial Number: Sequence Code
4	17		0	
4	16		0	
3	15		0	Card Type R2D3-ASSY-0040 = 000001000000
3	14		0	
3	13		0	
3	12		0	
2	11		0	
2	10		0	
2	9		0	
2	8		0	
1	7		0	
1	6		0	
1	5		0	
1	4		0	
0	3		0	Card Variant R2D3-ASSY-0040.5 = 0100
0	2		0	
0	1		0	
0	0		0	

LVDS Card and previous RS-422A Card Version

The following applies to all LVDS cards and also to RS-422A cards with serial numbers higher than 2003/48/7 (manufactured prior to December 2003).

Nibble	Bit No.	Active State	Power-on State	Data
7	31	0	0	Offset Address
7	30	0	0	
7	29	0	0	
7	28	1	0	
6	27	0	0	Hardware Error
6	26	0	0	Not used
6	25	0	0	
6	24	0	0	
5	23	0	0	
5	22	0	0	
5	21	0	0	
5	20	0	0	
4	19	0	0	Number of streams
4	18	0	0	
4	17	0	0	
4	16	1	0	
3	15	0	0	Interface Type
3	14	0	0	
3	13	0	0	
3	12	0	0	
2	11	0	0	
2	10	0	0	
2	9	0 or 1	0 or 1	
2	8	0 or 1	0 or 1	
1	7	0	0	Firmware Type
1	6	0	0	
1	5	0	0	
1	4	0	0	
0	3	0	0	Firmware Variant
0	2	0	0	
0	1	1	1	
0	0	0 or 1	0 or 1	

Status D

Nibble	Bit No.	Active State	Power-on State	Data
7	31	0	0	Offset Address
7	30	0	0	
7	29	0	0	
7	28	0	0	
6	27	0	0	Reserved
6	26	0	0	
6	25	0	0	
6	24	0	0	
5	23	0	0	
5	22	0	0	
5	21	0	0	
5	20	0	0	
4	19	0	0	
4	18	0	0	
4	17	0	0	Binary representation of the number of DWORDs remaining in the Buffer
4	16	1	0	
3	15	0	0	
3	14	0	0	
3	13	0	0	
3	12	0	0	
2	11	0	0	
2	10	0	0	
2	9	1	0	
2	8	1	0	
1	7	0	0	
1	6	0	0	
1	5	0	0	
1	4	0	0	
0	3	0	0	
0	2	0	0	
0	1	1	0	
0	0	1	0	

This page intentionally blank

11. SUPPORT

In this Section

What to do if you have a problem

Firstly, please ensure that you have followed the installation, connection and operation instructions in the appropriate User Guide.

Also, check the Troubleshooting section (where appropriate) to eliminate common problems.

Servicing, Maintenance and Repairs

Please contact your supplier or SomerData for all questions relating to maintenance and repairs.

Any unauthorised attempt to open, modify or otherwise repair the product will invalidate the SomerData warranty and may result in the product being left in an irreparable condition.

If you need Support

For warranty, technical and application support issues, you should initially contact your supplier to check whether your SomerData product is covered by warranty, extended warranty or maintenance contract.

At SomerData, we will make our best efforts to provide prompt and friendly support by phone, fax and e-mail.

However, please do not expect us to be magicians or mind-readers!

Diagnosing a problem will require your co-operation and we expect you to provide a detailed description of the problem in the form of a detailed Fault Report.

SomerData Contact Information

Address:	Somerdata Limited Underwood Business Park Wookey Hole Road Wells Somerset BA5 1AF UK	
Phone:	UK	01749-671481
	International	+44 1749-671481
Fax:	UK	01749-671482
	International	+44 1749-671482
E-Mail:	*support@somerdata.com*	
Website:	*www.somerdata.com*	

Support Requests

When contacting SomerData for support, please provide as much information as possible about the problem or issue for which you require assistance.

We will be able to deal with your request more efficiently if you provide the following details (where available) in your Fault Report:

- Part Number or Model Number
  (for example R2D3-ASSY-0326.5)
- Serial Number (for example 2003/19/005)
- Software Version (for example 1480.1.0.0)
- Details of any symptoms or error messages
- Diagnostics information (if available)
- Sequence of events/actions or other circumstances that triggered the problem
- How you are able to identify that there is a problem
- How you have been able to measure, log or otherwise display the problem
- Details of the host PC (if appropriate) including: operating system; hardware configuration; other hardware devices (e.g. additional PCI cards); other software applications (e.g. analysis or processing programs) that are running at the same time
- Sample data files (if appropriate)

Returns

Please do not return any products to SomerData without first contacting SomerData and obtaining a Return Merchandise Authorisation (RMA) Number.

This will ensure that the processing of any repair or upgrade is handled efficiently and in accordance with any agreed action.

If the SomerData product is under warranty, repairs are free-of charge. If not, there will be a repair charge, which will comprise an initial evaluation fee and quotation, followed by repair and parts (if authority is given to carry-out the repair).

Pack the item in its original packaging. If the original packaging is not available, it must be packed in such a way to avoid transit damage. Damage sustained in transit is not covered under warranty.

Returned goods should be accompanied by documentation that indicates the RMA Number along with a detailed fault report and contact details (name, organisation, phone, fax and e-mail).

Mark the RMA Number on the outside of the package.

Ship the item by insured, prepaid carrier to the above address.

Items being returned from outside the European Community *must* be accompanied by a Commercial Invoice. This should include a description of the goods, value for Customs Purposes and state that the goods are being temporarily returned to the UK for repair. SomerData will not accept liability for UK importation costs resulting from inadequate documentation.

12. INDEX