OXuPCI952 Data Sheet

DS-0059

August 17 2009

Website	www.plxtech.com
Technical Support	www.plxtech.com/support

August 17 2009

Document number: DS-0059

Contents

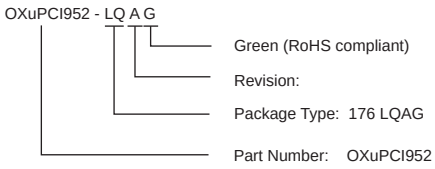
Features 1
Description 2
Improvements of the OXuPCI952 over Discrete Solutions 2
OXuPCI952 Device Modes 3
Pin Information—176-pin LQFP 5
Pinouts 5
Pin Descriptions 7
Configuration and Operation 14
PCI Target Controller 14
Configuration Space 16
Accessing Logical Functions 18
Accessing Local Configuration Registers 22
PCI Interrupts 32
Power Management 33
Internal OX16C950 UARTs 38
Operation – Mode Selection 38
Register Description Tables 41
UART Reset Configuration 45
Transmitter and receiver FIFOs 46
Line Control and Status 49
Interrupts and Sleep Mode 52
Modem Interface 56
Other Standard Registers 59
Automatic Flow Control 59
Baud Rate Generation 62
Additional Features 67
Local Bus 76
Operation 76
Configuration and Programming 78
Bidirectional Parallel Port 79
Operation and Mode Selection 79
Parallel Port Interrupt 80
Register Description 81
Serial EEPROM 86

Power Consumption 93
DC Electrical Characteristics 94
AC Electrical Characteristics 98
PCI Bus Timings 99
Local Bus 99
Serial Ports 102
Timing Waveforms 103
Package Information 118
176-Pin LQFP Package 118

This guide gives Information on the OXuPCI952.

Ordering Information

The following conventions are used to identify products:



Typographic Conventions

In this document, the following conventions apply.

Convention	Meaning
Italic Letters With Initial Capital Letters	A cross-reference to another publication
Title	A cross-reference to another section within the document
1, 2, 3	A numbered list where the order of list items is significant
■	A list where the order of items is not significant
	Significant additional information
`Courier`	Software code
Bold	Significant names, for example of files or directories Text you type

Revision Information

The following table documents the revisions of this guide.

Revision	Modification
August 17 2009	Rebranded and reformatted, pin number correction in Configuration and Operation on page 14
January 26 2009	Removal of redundant operating conditions table
November 06 2008	Minor change to Block Diagram, updates to pin descriptions, 5V added to I/O direction for pin descriptions, addition of PCI SIG logo
September 2007	Feature revision, including removal of D3cold
May 2007	First publication.

This page is intentionally blank

Features

- Dual 16C950 high performance UART channels
- 8-bit pass-through local bus (PCI bridge)
- IEEE1284 compliant SPP/EPP/ECP parallel port
- Efficient 32-bit, 33 MHz, multi-function target-only PCI controller, fully compliant to the *PCI Local Bus Specification 3.0*, and the *PCI Power Management Specification 1.1*
- Software-compatible with OXmPCI952
- UARTs fully software compatible with 16C550-type devices
- UART operation up to 60 MHz using external clock source. Up to 20 MHz with the crystal oscillator
- Baud rates up to 60 Mbps in external 1x clock mode and 15 Mbps in asynchronous mode
- 128-byte deep FIFO per transmitter and receiver
- Flexible clock prescaler, from 1 to 31.875
- Automated in-band flow control using programmable Xon/Xoff in both directions
- Automated out-of-band flow control using CTS#/RTS# and/or DSR#/DTR#
- Programmable RS485 turnaround delay
- Arbitrary trigger levels for receiver and transmitter FIFO interrupts and automatic in-band and out-of-band flow control
- Infra-red (IrDA) receiver and transmitter operation
- 9-bit data framing, as well as 5 ,6, 7 and 8 bits
- Detection of bad data in the receiver FIFO
- Global Interrupt Status and readable FIFO levels to facilitate implementation of efficient device drivers
- Local registers to provide status/control of device functions
- 11 multi-purpose I/O pins, which can be configured as input interrupt pins or ‘wake-up’
- Auto-detection of a wide range of MICROWIRE™ compatible EEPROMs, to configure device parameters
- Function access, to pre-configure each function prior to handover to generic device drivers
- Operation using I/O or memory mapping
- 3.3 V or 5 V PCI universal voltage operation
- Extended operating temperature range: -40° C to 85° C
- 176-pin LQFP package

Description

The OXuPCI952 is a single-chip solution for PCI-based serial and parallel expansion add-in cards. It is a dual function PCI device, where function 0 offers two ultra-high performance OX16C950 UARTs, and function 1 is configurable either as an 8-bit local bus or a bi-directional parallel port.

Each UART channel in the OXuPCI952 is the fastest available PC-compatible UART, offering data rates up to 15 Mbps and 128-byte deep transmitter and receiver FIFOs. The deep FIFOs reduce CPU overhead and allow use of higher data rates. Each UART channel is software compatible with the widely used industry-standard 16C550 devices (and compatibles), as well as the OX16C95x family of high performance UARTs. In addition to increased performance and FIFO size, the UARTs also provide the full set of OX16C95x enhanced features including automated in-band flow control, readable FIFO levels and similar.

To enhance device driver efficiency and reduce interrupt latency, internal UARTs have multi-port features such as shadowed FIFO fill levels, a global interrupt source register and Good-Data Status, readable in four adjacent DWORD registers visible to logical functions in I/O space and memory space.

Expansion of serial cards beyond two channels is possible using the 8-bit pass-through local bus function. This provides a general address/data bus and interrupt capability to a discrete UART part, such as the PLX Technology OX16C954. Other controllers can be used to provide capabilities beyond additional UART ports. The addressable space can be increased up to 256 bytes, and divided into four chip-select regions. This flexible expansion scheme caters for cards with up to 18 serial ports using external 16C950, 16C954 or compatible devices, or composite applications such as combined serial and parallel port expansion cards.

The parallel port is an IEEE 1284 compliant SPP/EPP/ECP parallel port that fully supports the existing Centronics interface. The parallel port can be enabled in place of the local bus. An external bus transceiver is required for 5V parallel port operation if the device is 3.3V sourced.

The configuration register values are programmed using an external Microwire compatible serial EEPROM. This EEPROM can also be used to provide function access, to pre-configure devices on the local bus/parallel port prior to any PCI configuration accesses and before control is handed to (generic) device drivers.

The OXuPCI952 can replace the OXmPCI952 in PCI applications, with the added benefit that the former device does not require an EEPROM to define a 952 part explicitly.

Improvements of the OXuPCI952 over Discrete Solutions

Higher degree of integration

The OXuPCI952 device offers two internal 16C950 high-performance UARTs and an 8-bit local bus or a bi-directional parallel port.

Multi-function device

The OXuPCI952 is a multi-function device to enable you to load individual device drivers for the internal serial ports, drivers for the peripheral devices connected to the local bus or drivers for the internal parallel port.

Dual Internal OX16C950 UARTs

The OXuPCI952 device contains two ultra-high performance UARTs, which can increase driver efficiency by using features such as the 128-byte deep transmitter and receiver FIFOs, flexible clock options, automatic flow control, programmable interrupt and flow control trigger levels and readable FIFO levels. Data rates are up to 60 Mbps.

Improved access timing

Access to the internal UARTs require zero or one PCI wait state. A PCI read transaction from an internal UART can complete within five PCI clock cycles and a write transaction to an internal UART can complete within four PCI clock cycles.

Reduces interrupt latency

The OXuPCI952 offers shadowed FIFO levels and Interrupt status registers on the internal UARTs and the MIO pins. This reduces the device driver interrupt latency.

Power management

The OXuPCI952 complies with the *PCI Power Management Specification 1.1* and the Microsoft *Communications Device-class Power Management Specification 2.0 (2000*). Both functions offer the extended capabilities for power management. This achieves significant power savings by enabling device drivers to power down the PCI functions. For function 0, this is through switching off the channel clock, in power state D3. Wake-up (PME# generation) can be requested by either function. For function 0, this is using the RI# inputs of the UARTs in the power-state D3 or any modem line and SIN inputs of the UARTs in power-state D2. For function 1, this is using the MIO[2] input.

Optional EEPROM

The OXuPCI952 can be reconfigured from an external EEPROM to your requirements. However, this is not required for many applications as the default values are sufficient for typical applications. An overrun detection mechanism built into the EEPROM controller prevents the PCI system from ‘hanging’ due to an incorrectly programmed EEPROM.

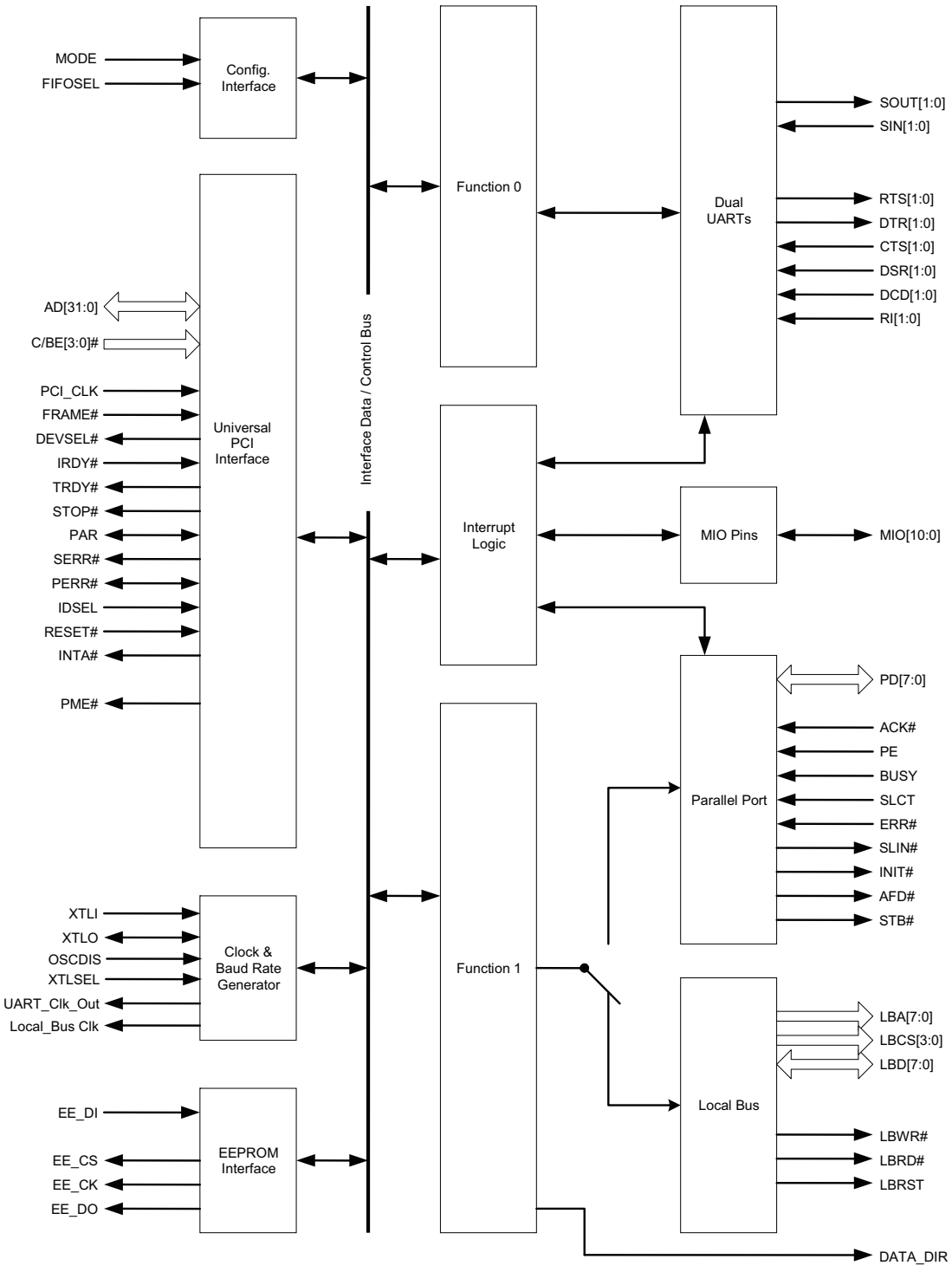
OXuPCI952 Device Modes

The OXuPCI952 supports two modes of operation, as summarized in the following table.

Device Mode	Mode Pin Selection	Functionality
0	MODE = 0	Function 0: Dual UARTs Function 1: 8-bit local bus
1	MODE = 1	Function 0: Dual UARTs Function 1: Parallel Port

The OXuPCI952 is not pin-compatible with the OX16PCI952 or the OXmPCI952, but is the same in all other aspects.

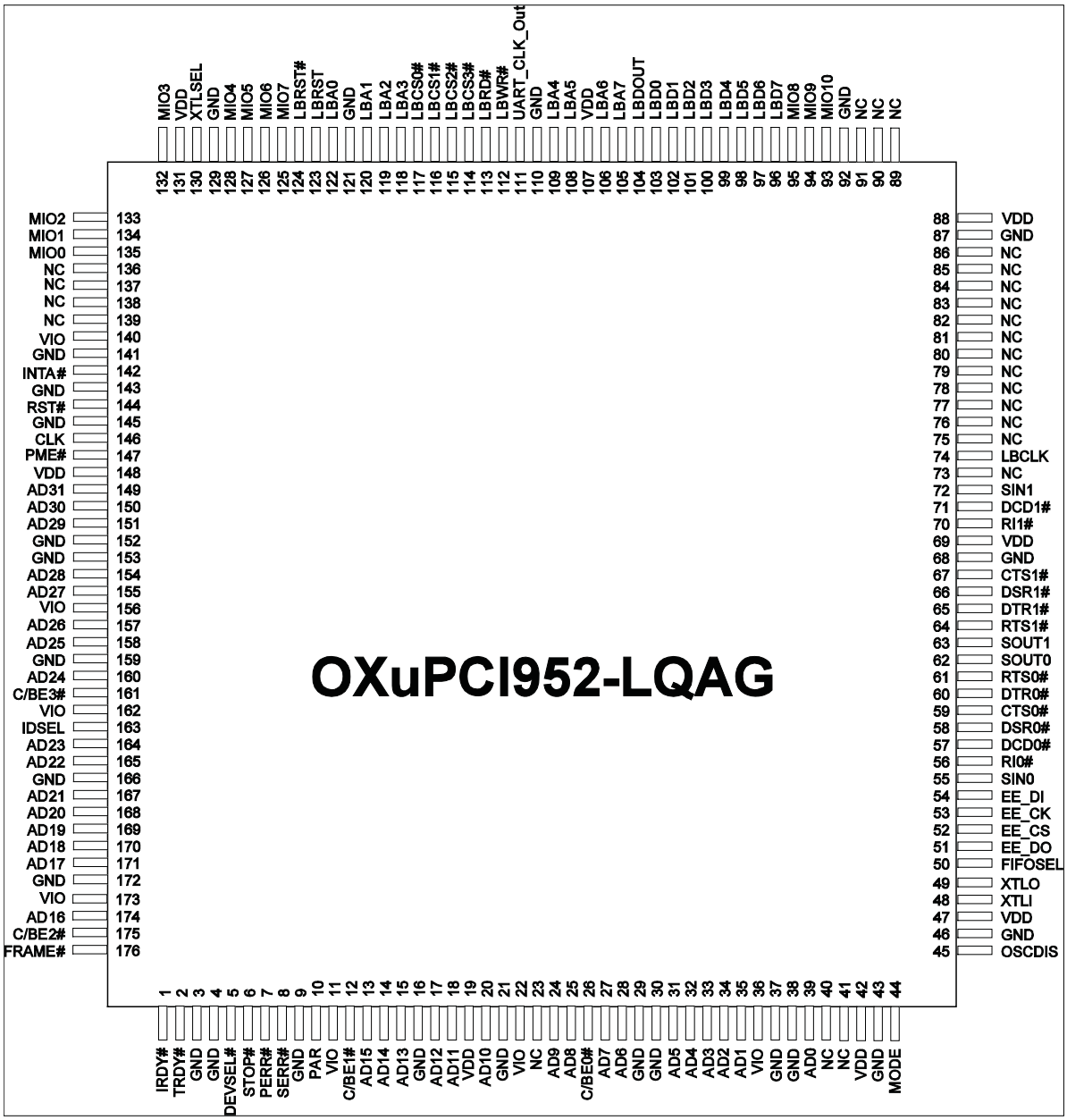
Figure 1 Block Diagram



Pin Information—176-pin LQFP

Pinouts

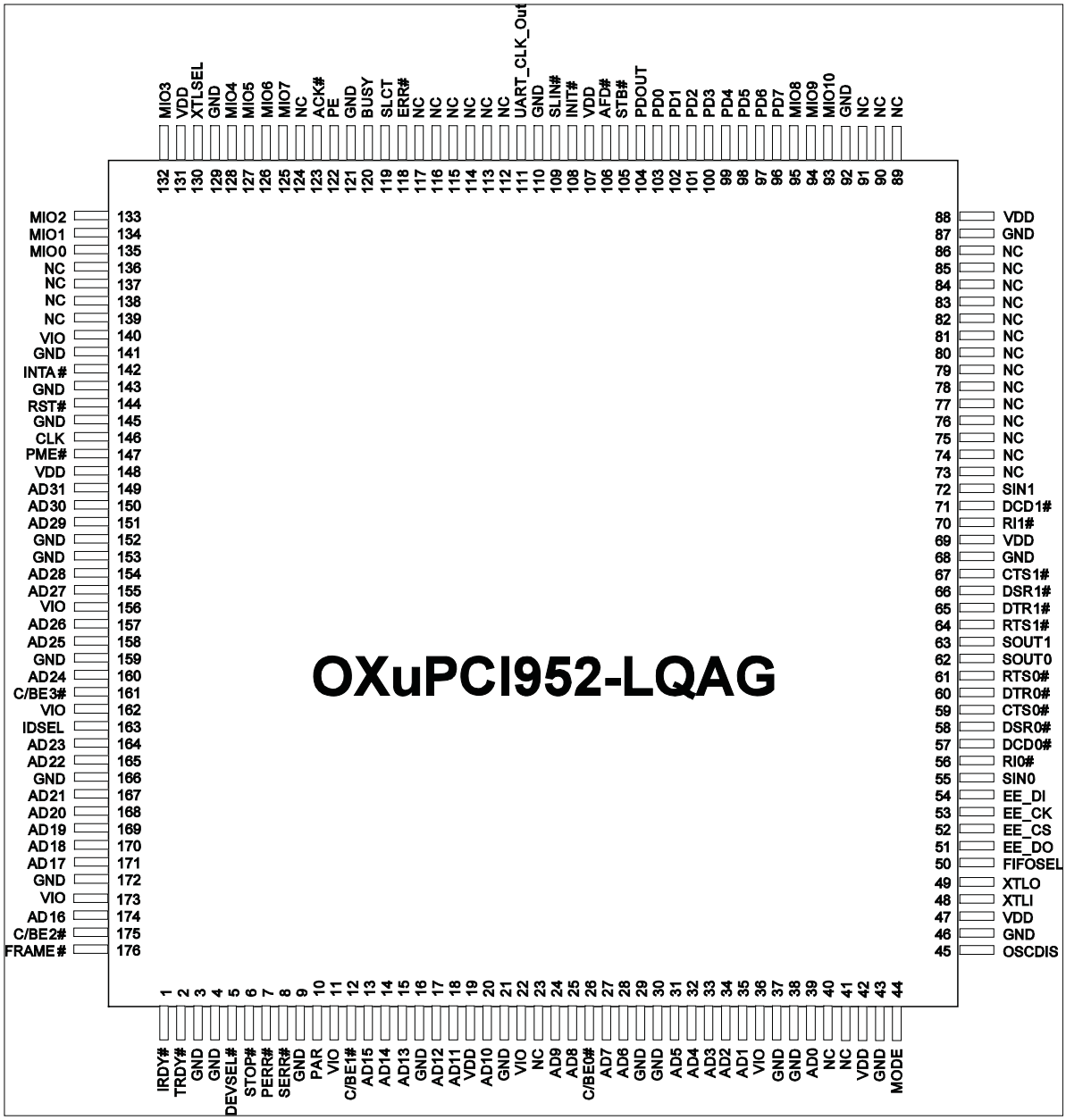
Figure 2 Dual UARTs + 8-bit Local Bus (Mode = 0)



13 NC. Do not connect these pins:

23, 40, 41, 79, 80, 81, 82, 83, 84, 136, 137, 138, 139

Figure 3 Dual UARTs + Parallel Port (Mode = 1)



21 NC. Do not connect these pins:

23, 40, 41, 74, 79, 80, 81, 82, 83, 84, 112, 113, 114, 115, 116, 117, 124, 136, 137, 138, 139

Pin Descriptions

For the I/O direction key, see page 13.

Table 1 PCI Interface – Mode 0 and Mode 1			
Pin	Dir[1]	Name	Description
149, 150, 151, 154, 155, 157, 158, 160, 164, 165, 167, 168, 169, 170, 171, 174, 13, 14, 15, 17, 18, 20, 24, 25, 27, 28, 31, 32, 33, 34, 35, 39	P_I/O	AD[31:0]	Multiplexed PCI address/data bus
161, 175, 12, 26	P_I	C/BE[3:0]#	PCI command/Byte enable
146	P_I	CLK	PCI system clock (33MHz)
176	P_I	FRAME#	Cycle frame
5	P_O	DEVSEL#	Device select
1	P_I	IRDY#	Initiator ready
2	P_O	TRDY#	Target ready
6	P_O	STOP#	Target stop request
10	P_I/O	PAR	Parity
8	P_O	SERR#	System error
7	P_I/O	PERR#	Parity error
163	P_I	IDSEL	Initialization device select
144	P_I	RST#	PCI system reset
142	P_OD	INTA#	Default PCI interrupt line. For Function 0 and Function 1
147	P_OD	PME#	Power management event

Table 2 Serial Port Pins – Mode 0 and Mode 1 (Sheet 1 of 2)

Pin	Dir[1]	Name	Description
50	I	FIFOSEL	FIFO select. For backward compatibility with 16C550, 16C650 and 16C750 devices, the UARTs' FIFO depth is 16 when FIFOSEL is low. The FIFO size is increased to 128 when FIFOSEL is high. The unlatched state of this pin is readable by software. The FIFO size may also be set to 128 by setting FCR[5] when LCR[7] is set, or by putting the device into Enhanced mode.
63, 62	O(h)	SOUT[1:0] IrDA_Out[1:0]	These two pins are present in all modes but they can serve one of two functions: ■ UART serial data outputs. ■ UART IrDA data output when MCR[6] of the corresponding channel is set in Enhanced mode.
72, 55	I(h) I(h)	SIN[1:0] IrDA_In[1:0]	These two pins are present in all modes but they can serve one of two functions: ■ UART serial data inputs ■ UART IrDA data input when IrDA mode is enabled (see above).
71, 57	I(h)	DCD[1:0]#	Active-low modem data-carrier-detect input
65, 60	O(h) O(h) O(h)	DTR[1:0]# 485_En[1:0] Tx_Clk_Out[1:0]	These two pins are present in all modes but they can serve one of three functions: ■ Active-low modem data-terminal-ready output. If automated DTR# flow control is enabled, the DTR# pin is asserted and deasserted if the receiver FIFO reaches or falls below the programmed thresholds, respectively ■ In RS485 half-duplex mode, the DTR# pin may be programmed to reflect the state of the transmitter empty bit to automatically control the direction of the RS485 transceiver buffer (see register ACR[4:3]) ■ Transmitter 1x clock (baud rate generator output). For isochronous applications, the 1x (or Nx) transmitter clock may be asserted on the DTR# pins (see register CKS[5:4])
64, 61	O(h)	RTS[1:0]#	Active-low modem request-to-send output. If automated RTS# flow control is enabled, the RTS# pin is deasserted and reasserted whenever the receiver FIFO reaches or falls below the programmed thresholds, respectively
67, 59	I(h)	CTS[1:0]#	Active-low modem clear-to-send input. If automated CTS# flow control is enabled, upon deassertion of the CTS# pin, the transmitter completes the current character and enters the idle mode until the CTS# pin is reasserted. Note that flow control characters are transmitted regardless of the state of the CTS# pin

Table 2 Serial Port Pins – Mode 0 and Mode 1 (Sheet 2 of 2)			
Pin	Dir[1]	Name	Description
66, 58	I(h) I(h)	DSR[1:0]# Rx_Clk_In[1:0]	These two pins are present in all modes but they can serve one of two functions: ■ Active-low modem data-set-ready input. If automated DSR# flow control is enabled, upon deassertion of the DSR# pin, the transmitter completes the current character and enters the idle mode until the DSR# pin is reasserted. Note that flow control characters are transmitted regardless of the state of the DSR# pin ■ External receiver clock for isochronous applications. The Rx_Clk_In is selected when CKS[1:0] = '01'
70, 56	I(h) I(h)	RI[1:0]# Tx_Clk_In[1:0]	Active-low modem Ring-Indicator input External transmitter clock. This clock can be used by the transmitter (and indirectly by the receiver) when CKS[6]='1'

Table 3 Clock Interface Pins – Mode 0 and Mode 1			
Pin	Dir[1]	Name	Description
49	I/O	XTLO	Crystal oscillator output when OSCDIS = '0'. External clock source input when OSCDIS = '1'
48	I	XTLI	Crystal oscillator input when OSCDIS = '0', up to 20MHz. N/C when OSCDIS = '1'
45	I	OSCDIS	Oscillator disable. 0 = the internal crystal oscillator is enabled and a crystal needs to be attached to XTLI/XTLO. XTLSEL must be set according to the crystal frequency that is used (up to 20Mhz). 1 = the internal crystal oscillator is disabled and an external oscillator source (up to 60MHz) can be input to XTLO. XTLI is N/C and XTLSEL must be 0
130	I	XTLSEL	Defines the frequency of the crystal attached to XTLI/XTLO (when OSCDIS = '0') 0 = 1 MHz – 12 MHz 1 = 12 MHz – 20 MHz

Table 4 8-bit Local Bus – Mode 0			
Pin	Dir[1]	Name	Description
111	O	UART_Clk_Out	Buffered crystal output. This clock can drive external UARTs connected to the local bus. Can be enabled and disabled by software
123	O(h)	LBRST	Local bus active-high reset
124	O	LBRST#	Local bus active-low reset
104	O	LBDOUT	Local bus data out enable. This pin can be used by external transceivers; it is high when LBD[7:0] are in output mode and low when they are in input mode
74	O	LBCLK	Buffered PCI clock. Can be enabled and disabled by software
114, 115, 116, 117	O(h) O(h)	LBCS[3:0]# LBDS[3:0]#	Local bus active-low chip-select (Intel mode). Local bus active-low data-strobe (Motorola mode)
112	O O	LBWR# LBRDWR#	Local bus active-low write-strobe (Intel mode) Local bus read-not-write control (Motorola mode)
113	O Z	LBRD# Hi-Z	Local bus active-low read-strobe (Intel mode) Permanent high impedance (Motorola mode)
105, 106, 108, 109, 118, 119, 120, 122	O(h)	LBA[7:0]	Local bus address signals
96, 97, 98, 99 100, 101, 102, 103	I/O(h)	LBD[7:0]	Local bus data signals
23, 40, 41, 79, 80, 81, 82, 83, 84, 136, 137, 138, 139		NC	Do not connect

Table 5 Parallel Port – Mode 1			
Pin	Dir[1]	Name	Description
123	I(h) I(h)	ACK# INTR#	Acknowledge (SPP mode). ACK# is asserted (low) by the peripheral to indicate that a successful data transfer has taken place Identical function to ACK# (EPP mode)
122	I(h)	PE	Paper empty. Activated by printer when it runs out of paper
120	I(h) I(h)	BUSY WAIT#	Busy (SPP mode). BUSY is asserted (high) by the peripheral when it is not ready to accept data Wait (EPP mode). Handshake signal for interlocked IEEE 1284 compliant EPP cycles
109	OD(h) O(h)	SLIN# ADDRSTB#	Select (SPP mode). Asserted by host to select the peripheral. Address strobe (EPP mode) provides address read and write strobe
119	I(h)	SLCT	Peripheral selected. Asserted by peripheral when selected
118	I(h)	ERR#	Error. Held low by the peripheral during an error condition
108	OD(h) O(h)	INIT# INIT#	Initialize (SPP mode). Commands the peripheral to initialize. Initialize (EPP mode). Identical function to SPP mode
106	OD(h) O(h)	AFD# DATASTB#	Auto Feed (SPP mode, open-drain). Data strobe (EPP mode) provides data read and write strobe
105	OD(h) O(h)	STB# WRITE#	Strobe (SPP mode). Used by peripheral to latch data currently available on PD[7:0]. Write (EPP mode). Indicates a write cycle when low and a read cycle when high
96, 97, 98, 99 100, 101, 102, 103	I/O(h)	PD[7:0]	Parallel data bus
104	O	PDOUT	Parallel port data out enable. This pin should be used by external transceivers for 5 V signaling; it is high when PD[7:0] are in output mode and low when they are in input mode
23, 40, 41, 74, 79, 80, 81, 82, 83, 84, 112, 113, 114, 115, 116, 117, 124, 136, 137, 138, 139		NC	Do not connect

Table 6 Multi-purpose and External Interrupt Pins – Mode 0 and Mode 1				
Pin		Dir[1]	Name	Description
Mode 0	Mode 1			
135	-	I/O(h)	MIO0	Multi-purpose I/O 0. Can drive high or low, or assert a PCI interrupt
-	135	O	NC	Output Driving '0'. Can be left as a no-connect
134	134	I/O(h)	MIO1	Multi-purpose I/O 1. Can drive high or low, or assert a PCI interrupt (as long as LCC[6:5] = "00")
134	134	0	NC	Output Driving '0' (when LCC[6:5] ≠ '00') Can be left as a no-connect
133	133	I/O(h)	MIO2	Multi-purpose I/O 2. When LCC[7] = 0, this pin can drive high or low, or assert a PCI interrupt
133	133	I	PME_In	Input power management event. When LCC[7] is set this input pin can assert a function 1 PME#
93, 94, 95, 125, 126, 127, 128, 132		I/O(h)	MIO[10:3]	Multi-purpose I/O pins. Can drive high or low, or assert a PCI interrupt

Table 7 EEPROM Pins – All Modes			
Pin	Dir[1]	Name	Description
53	O	EE_CK	EEPROM clock
52	O	EE_CS	EEPROM active-high chip select
54	IU(h)	EE_DI	EEPROM data in, with internal pull-up. When the serial EEPROM is connected, this pin should be pulled up using a 1-10k resistor. Pin to be connected to the external EEPROM's EE_DO pin
51	O	EE_DO	EEPROM data out. Pin to be connected to the external EEPROM's EE_DI pin

Table 8 Miscellaneous Pins			
Pin	Dir[1]	Name	Description
44	I	MODE	Mode selection pin: 0: Function 0: Dual UART Function 1: 8-bit local bus 1: Function 0: Dual UART Function 1: Parallel port
73, 75, 76, 77, 78, 85, 86, 89, 90, 91	N/A	N/A	Not connected

Table 9 Power and Ground			
Pin	Type	Name	Description
19, 42, 47, 69, 88, 107, 131, 148	V	VDD	Power Supply (3.3 V)
11, 22, 36, 140, 156, 162, 173	V	VIO	PCI I/O universal voltage Defines the (clamping) voltage of the PCI I/O buffers. To be connected to the VIO pin of the PCI connector
3, 4, 9, 16, 21, 29, 30, 37, 38, 43, 46, 68, 87, 92, 110, 121, 129, 141, 143, 145, 152, 153, 159, 166, 172	G	GND	Power supply ground (0 V)

I/O Direction Key

P_I	PCI input	3.3 V and 5 V
P_O	PCI output / PCITristates	3.3 V and 5 V
P_I/O	PCI bi-directional	3.3 V and 5 V
P_OD	PCI open drain	3.3 V and 5 V
I	Input	LVTTL level
I(h)	Input	LVTTL level, 5 V tolerant
IU(h)	Input with internal pull-up	LVTTL level, 5 V tolerant
I/O(h)	Bi-directional	LVTTL level, 5 V tolerant
O	Output	Standard Output
O(h)	Output	5 V tolerant (high voltage bi-direct in output mode)
OD	Open drain	Standard open-drain output
OD(h)	Open drain	5 V tolerant (high voltage bi-direct in open-drain mode)
NC	No connect	
G	Ground	
V	Voltage	

Configuration and Operation

The OXuPCI952 is a multi-function, target-only PCI device, compliant with the *PCI Local Bus Specification Revision 3.0*, and the *PCI Power Management Specification Revision 1.1*.

The OXuPCI952 affords maximum configuration flexibility by treating the internal UARTs, the local bus and the parallel port as separate logical functions. Each function has its own configuration space and is therefore recognized and configured by the PCI BIOS separately. The functions used are configured by the mode selection pin (pin 44).

The OXuPCI952 is configured by system start-up software during the bootstrap process that follows bus reset. The system scans the bus and reads the vendor and device identification codes from any devices it finds. It then loads device-driver software according to this information and configures the I/O, memory and interrupt resources. Device drivers can then access the functions at the assigned addresses in the usual fashion, with the improved data throughput provided by PCI.

Each function operates as though it is a separate device. However there are a set of Local Configuration Registers that can be used to enable signals and interrupts, configure timings, and improve the efficiency of multi-port drivers. This architecture enables separate drivers to be installed for each function. Generic port drivers can be hooked to use the functions individually, or more efficient multi-port drivers can hook both functions, accessing the Local Configuration Registers from either.

All registers default after reset to suitable values for typical applications such as 2/6 port serial, or combo 2-port serial/1-port parallel add-in cards. However, all identification, control and timing registers can be redefined using the external serial EEPROM.

PCI Target Controller

The OXuPCI952 responds to the following PCI transactions:

- Configuration access: the OXuPCI952 responds to type 0 configuration reads and writes if the IDSEL signal is asserted and the bus address is selecting the configuration registers for function 0 or 1. The device responds to the configuration transaction by asserting DEVSEL#. Data transfer then follows. Any other configuration transaction is ignored by the OXuPCI952
- I/O reads/writes: the address is compared with the addresses reserved in the I/O Base Address Registers (BARs). If the address falls within one of the assigned ranges, the device responds to the I/O transaction by asserting DEVSEL#. Data transfer follows this address phase. For the UARTs and 8-bit local bus controller, only byte accesses are possible. For I/O accesses to these regions, the controller compares AD[1:0] with the byte-enable signals as defined in the PCI specification. The access is always completed; however if the correct BE signal is not present the transaction has no effect

- Memory reads/writes: these are treated in the same way as I/O transactions, except that the memory ranges are used. Memory access to single-byte regions is always expanded to DWORDs in the OXuPCI952. In other words, OXuPCI952 reserves a DWORD per byte in single-byte regions. The device allows you to define the active byte lane using LCC[4:3] so that in big-endian systems the hardware can swap the byte lane automatically. For memory mapped access in single-byte regions, the OXuPCI952 compares the asserted byte-enable with the selected byte-lane in LCC[4:3] and completes the operation if a match occurs, otherwise the access completes normally on the PCI bus, but it has no effect on either the internal UARTs or the local bus controller
- All other cycles (64-bit, special cycles, reserved encoding etc.) are ignored

The OXuPCI952 completes all transactions as disconnect-with-data. This means that the device asserts the STOP# signal alongside TRDY#, to ensure that the bus master does not continue with a burst access. The exception to this is Retry, which is signaled in response to any access while the OXuPCI952 is reading from the serial EEPROM.

The OXuPCI952 performs medium-speed address decoding as defined by the PCI specification. It asserts the DEVSEL# bus signal two clocks after FRAME# is first sampled low on all bus transaction frames which address the chip. The internal UARTs are accessed with zero wait states inserted. Fast back-to-back transactions are supported by the OXuPCI952 as a target, so a bus master can perform faster sequences of write transactions to the UARTs or local bus when an inter-frame turn-around cycle is not required.

The device supports any combination of byte-enables to the PCI Configuration Registers and the Local Configuration Registers. If a byte-enable is not asserted, that byte is unaffected by a write operation and undefined data is returned upon a read.

The OXuPCI952 performs parity generation and checking on all PCI bus transactions as defined by the standard. Note that this is entirely unrelated to serial data parity which is handled within the UART functional modules themselves. If a parity error occurs during the PCI bus address phase, the device reports the error in the standard way by asserting the SERR# bus signal. However if that address/command combination is decoded as a valid access, it still completes the transaction as though the parity check is correct.

The OXuPCI952 does not support any kind of caching or data buffering in addition to that already provided within the UARTs by the transmit and receive data FIFOs. In general, registers in the UARTs and on the local bus cannot be pre-fetched because there may be side-effects on read.

Configuration Space

The OXuPCI952 is a dual-function device, where each logical function has its own configuration space. All required fields in the standard header are implemented, plus the Power Management Extended Capability register set. The format of the configuration space is shown in the following tables.

In general, writes to any registers that are not implemented are ignored, and all reads from unimplemented registers return 0.

PCI Configuration Space Register Map

Predefined PCI Region

Configuration Register Description				Offset
31	16	15	0	Address
Device ID		Vendor ID		00h
Status		Command		04h
Class Code			Revision ID	08h
BIST	Header Type	Reserved	Reserved	0Ch
Base Address Register 0 (BAR 0)				10h
Base Address Register 1 (BAR 1)				14h
Base Address Register 2 (BAR 2)				18h
Base Address Register 3 (BAR 3)				1Ch
Base Address Register 4 (BAR 4)				20h
Base Address Register 5 (BAR 5)				24h
Reserved				28h
Subsystem ID		Subsystem Vendor ID		2Ch
Reserved				30h
Reserved			Cap_Ptr	34h
Reserved				38h
Reserved	Reserved	Interrupt Pin	Interrupt Line	3Ch

User Defined Region

Power Management Capabilities (PMC)		Next Ptr	Cap_ID	40h
Data	Reserved	PMC Control/Status Register (PMCSR)		44h

PCI Configuration Space Default Values

Following use of the external EEPROM.

Register Name	Mode 0		Mode 1			
	Function 0	Function 1	Function 0	Function 1	EEPROM	PCI
	DUAL UART	8-Bit LOCAL BUS	DUAL UART	PARALLEL PORT		
Vendor ID	0x1415				W	R
Device ID	0x9505	0x9511	0x9505	0x9513	W	R
Command	0x0000				-	R/W
Status	0x0290				W(Bit 4)	R/W
Revision ID	0x01				-	R
Class Code	0x070006	0x068000	0x070006	0x070101	W	R
Header	0x80				-	R
BAR 0	0x00000001	0x00000001	0x00000001	0x00000001	-	R
BAR 1	0x00000001	0x00000000	0x00000001	0x00000001	-	R
BAR 2	Unused	0x00000001	Unused	0x00000001	-	R
BAR 3	Unused	0x00000000	Unused	0x00000000	-	R
BAR 4	0x00000001	Unused	0x00000001	Unused	-	R
BAR 5	0x00000000	Unused	0x00000000	Unused	-	R
Subsystem Vendor ID	0x1415				W	R
Subsystem ID	0x0000				W	R
Cap. Ptr	0x40				-	R
Interrupt Line	0x00				-	R
Interrupt Pin	0x01				W	R
Cap ID	0x01				-	R
Next Ptr	0x00				-	R
PM Capabilities	0x6C02				W	R
PMC Control/ Status Register	0x0000				W (Data Scale)	R/W
PM Data Register	0x00				W	R

Accessing Logical Functions

Access to the two UARTs, the local bus and the parallel port is achieved using standard I/O and memory mapping, at addresses defined by the Base Address Registers (BARs) in the PCI configuration space. The BARs are configured by the system to allocate blocks of I/O and memory space to the logical functions, according to the size required by the function. The addresses allocated can then be used to access the functions. The mapping of these BARs, which is dependent upon the mode of the device, is shown by the following tables.

BAR	Function 0
	Dual UARTs (Mode 0, Mode 1)
0	Internal UART0 (I/O Mapped)
1	Internal UART1 (I/O Mapped)
2	Unused
3	Unused
4	Local configuration registers (I/O mapped)
5	Internal UARTs/ Local configuration registers (Memory mapped)

BAR	Function 1	
	Local bus (Mode 0)	Parallel port (Mode 1)
0	Local bus (I/O mapped)	Parallel port base registers
1	Local bus (Memory mapped)	Parallel port extended registers
2	Local configuration registers (I/O mapped)	
3	Local configuration registers (Memory mapped)	
4	Unused	
5	Unused	

PCI Access to Internal UARTs

I/O and Memory Space

BAR0 and BAR1 are used to address the two UARTs individually in I/O space, and BAR5 is used to address the UARTs in memory space. The function reserves an 8-byte block of I/O space for BAR0 and BAR1, and a 4Kbyte block of memory space for BAR5. Once the I/O access and the memory access enable bits in the Command register (configuration space) are set, the UARTs can accessed according to the following tables.

UART Register	PCI Offset from UARTs Base Address for Function0 in I/O Space (hex)	
	UART0 (BAR0)	UART1 (BAR1)
000	00	00
001	01	01
002	02	02
003	03	03
004	04	04
005	05	05
006	06	06
007	07	07

UART Register	PCI Offset from Base Address 5 for Function0 in Memory Space (hex)	
	UART0	UART1
000	00	20
001	04	24
002	08	28
003	0C	2C
004	10	30
005	14	34
006	18	38
007	1C	3C

Note that the local registers in memory space occupy the same Base Address Register (BAR5) as the internal dual UARTs in memory space. Bit 7 selects the region to be accessed. Access to addresses 00h to 3Ch is directed to the internal UARTs, and access to addresses 80h to FCh are directed to the local registers. When accessing the local registers using BAR5 (bit 7 set), fields 6:2 define the BYTE offset for the local registers; e.g. 00000b for the LCC register, 00100b for the MIC register.

In both cases, fields 1:0 are not used and are to be set to zeros. This is because a DWORD is used to hold a single Byte, in memory space.

PCI Access to 8-bit Local Bus

When the local bus is enabled (mode = 0), access to the bus works in a similar fashion to the internal UARTs. The function reserves a block of I/O space and a block of memory space. The I/O block size is user definable in the range of 4 to 256 bytes, and the memory range is fixed at 4Kbytes.

I/O space

In order to minimize the use of I/O space, the block size for BAR0 of Function1 is user definable in the range of 4 to 256 bytes. Having assigned the address range, you can define two adjacent address bits to decode up to four chip selects internally. This facility allows glueless implementation of the local bus connecting to four external peripheral chips. The address range and the lower address bit for chip-select decoding (Lower-Address-CS-Decode) are defined in the Local Bus Configuration register.

The 8-bit local bus has eight address lines (LBA[7:0]) that correspond to the maximum I/O address space. If the maximum allowable block size is allocated to the I/O space (i.e. 256 bytes), then as access in I/O space is byte aligned, LBA[7:0] equal PCI AD[7:0] respectively. When you select an address range which is less than 256 bytes, the corresponding upper address lines are set to logic zero.

The region can be divided into four chip-select regions when you select the second uppermost non-zero address bit for chip-select decoding. For example, if 32-bytes of I/O space are reserved, the local bus address lines A[4:0] are active and the remaining address lines are set to zero. To generate four chip-selects, select A3 as the Lower-Address-CS-Decode. In this case A[4:3] are used internally to decode chip-selects, asserting LBCS0# when the address offset is 00-07h, LBCS1# when the offset is 08-0Fh, LBCS2# when the offset is 10-17h, and LBCS3# when the offset is 18-1Fh.

The region can be divided into two chip-select regions by selecting the uppermost address bit to decode chip selects. In the above example, you can select A4 as the Lower-Address-CS-Decode, thus using A[5:4] internally to decode chip selects. As in this example LBA5 is always zero, only chip-select lines LBCS0# and LBCS1# are decoded into, asserting LBCS0# when address offset is 00-0Fh and LBCS1# when offset is 10-1Fh.

The region can be allocated to a single chip-select region by assigning an address bit beyond the selected range to Lower-Address-CS-Decode (but not above A8). In the above example, if you select A5 as the Lower-Address-CS-Decode, A[6:5] are used to internally decode chip-selects. As in this example LBA[7:5] are always zero, only the chip select line LBCS0# may be selected. In this case address offset 00-1Fh asserts LBCS0# and the other chip-select lines remain inactive permanently.

Memory Space

The memory base address registers have an allocated fixed size of 4Kbytes in the address space. Since the local bus has 8 address lines and the OXuPCI952 only implements DWORD aligned accesses in memory space, the 256 bytes of addressable space per chip select is expanded to 1K. Unlike an I/O access, for a memory access the upper address lines are always active and the internal chip-select decoding logic ignores the user setting for Lower-Address-CS-Decode (LT2[26:23]) and uses PCI AD[11:10] to decode into 4 chip-select regions. When the local bus is accessed in memory space, A[9:2] are asserted on LBA[7:0]. The chip-select regions are defined below.

Local Bus Chip-Select (Data-Strobe)	PCI Offset from BAR 1 in Function1 (Memory Space)	
	Lower Address	Upper Limit
LBCS0# (LBDS0#)	000h	3FCh
LBCS1# (LBDS1#)	400h	7FCh
LBCS2# (LBDS2#)	800h	BFCh
LBCS3# (LBDS3#)	C00h	FFCh

Note that the description given for I/O and memory accesses is for an Intel-type configuration for the local bus. For Motorola-type configuration, the chip select pins are redefined as data strobe pins. In this mode the local bus offers up to 8 address lines and four data-strobe pins.

PCI Access to Parallel Port

When the parallel port is enabled (mode = 1), access to the parallel port works using BAR definitions as usual, except that there are two I/O BARs corresponding to the two sets of registers defined to operate an IEEE1284 EPP/ ECP and bi-directional parallel port.

You can change the I/O space block size of BAR0 by over-writing the default values in LT2[25:20] using the serial EEPROM. For example, you can reduce the allocated space for BAR0 to 4 bytes by setting LT2[22:20] to ‘001’. The I/O block size allocated to BAR1 is fixed at 8 Bytes.

Legacy parallel ports expect the upper register set to be mapped 0x400 above the base block, therefore if the BARs are fixed with this relationship, generic parallel port drivers can be used to operate the device in all modes.

Example:
BAR0 = 0x00000379 (8 bytes at address 0x378)
BAR1 = 0x00000779 (8 bytes at address 0x778)

If this relationship is not used, custom drivers are needed.

Accessing Local Configuration Registers

The local configuration registers are a set of device specific registers which can be accessed from either function. The local configuration registers exist behind BAR4 and BAR5 for function 0, and behind BAR2 and BAR3 for function 1. For I/O transactions, access is limited to byte reads/writes. For memory transactions, accesses can be Word or Dword accesses; however, on little-endian systems such as Intel 80x86 the byte order is reversed.

The following table lists the definitions of the local registers, with the offsets (from the Base Address Register) defined for each local register.

Local Configuration and Control Register 'LCC' (Offset 0x00)

This register defines control of ancillary functions such as power management, external clock reference signals and the serial EEPROM. The individual bits are described below.

Bits	Description	Read/Write		Reset
		EEPROM	PCI	
0	Mode Status. This bit returns the state of the mode pin	-	R	X
1	Reserved. Returns 'X'	-	R	X
2	Enable UART clock output. When this bit is set, a buffered version of the UART clock is output on the pin "UART_Clk_Out". When this bit is low, the UART_Clk_Out is permanently low	W	RW	0
4:3	Endian Byte-Lane Select for memory access to 8-bit peripherals. 00 = Select Data[7:0] 01 = Select Data[15:8] 10 = Select Data[23:16] 11 = Select Data[31:24] Memory access to OXuPCI952 is always DWORD aligned. When accessing 8-bit regions like the internal UARTs, the 8-bit local bus and the parallel port, this option selects the active byte lane. As both PCI and PC architectures are little-endian, the default value is used by systems. However, some non-PC architectures may need to select the byte lane	W	RW	00
6:5	Power-down filter time. These bits define a value of an internal filter time for a power-down interrupt request in power management circuitry in Function0. Once Function0 is ready to go into power-down mode, the OXuPCI952 waits for the specified filter time and if Function0 is still in power-down request mode, it can assert a PCI interrup	W	RW	00
	00 = Power-down request disabled 01 = 4 seconds 10 = 129 seconds 11 = 518 seconds			
7	Function1 MIO2_PME Enable. A value of '1' enables MIO2 pin to set the PME_Status in PMCSR register, and hence assert the PME# pin if enabled. A value of '0' disables MIO2 from setting the PME_Status bit)	W	RW	0
23:8	Reserved. These bits are used for test purposes. The device driver must write zeros to these bits	-	R	0000h

Bits	Description	Read/Write		Reset
		EEPROM	PCI	
24	EEPROM Clock. For PCI read or write to the EEPROM, toggle this bit to generate an EEPROM clock (EE_CK pin)	-	W	0
25	EEPROM Chip Select. When 1 the EEPROM chip-select pin EE_CS is activated (high). When 0 EE_CS is de-active (low)	-	W	0
26	EEPROM Data Out. For writes to the EEPROM, this output bit is the input-data for the EEPROM. This bit is output on EE_DO and clocked into the EEPROM by EE_CK	-	W	0
27	EEPROM Data In. For reads from the EEPROM, this input bit is the output-data of the EEPROM connected to EE_DI pin	-	R	X
28	EEPROM Valid. A 1 indicates that a valid EEPROM program is present	-	R	X

Bits	Description	Read/Write		Reset
		EEPROM	PCI	
29	Reload configuration from EEPROM. Writing a 1 to this bit re-loads the configuration from EEPROM. This bit is self-clearing after EEPROM read	-	W	0
30	EEPROM Overrun Indication (when set). In conjunction with Bit 28 (Valid EEPROM) this bit indicates whether a successful EEPROM download had taken place. Successful download has EEPROM_VALID = 1 and EEPROM OVERRUN = 0	-	R	0
31	Reserved	-	R	1

Multi-purpose I/O Configuration register ‘MIC’ (Offset 0x04)

This register configures the operation of the multi-purpose I/O pins ‘MIO[10:0], as well as providing further device controls and status, as follows.

Bits	Description	Read/Write		Reset
		EEPROM	PCI	
1:0	MIO0 Configuration Register (when device mode ≠ ‘001’/’101’). 00 = MIO0 is a non-inverting input pin 01 = MIO0 is an inverting input pin 10 = MIO0 is an output pin driving ‘0’ 11 = MIO0 is an output pin driving ‘1’ When parallel port is enabled, (device mode = ‘001’/’101’), MIO[0] pin is unused and remains in forcing output mode	W	RW	00
3:2	MIO1 Configuration Register (when LCC[6:5] = ‘00’). 00 = MIO1 is a non-inverting input pin 01 = MIO1 is an inverting input pin 10 = MIO1 is an output pin driving ‘0’ 11 = MIO1 is an output pin driving ‘1’ When power-down mode in Function 0 is enabled (LCC[6:5] ≠‘00’), MIO1 pin is unused and remains in forcing output mode	W	RW	00
5:4	MIO2 Configuration Register (when LCC[7]=’0’). 00 = MIO2 is a non-inverting input pin 01 = MIO2 is an inverting input pin 10 = MIO2 is output pin driving ‘0’ 11 = MIO2 is output pin driving ‘1’ When LCC[7] is set, the MIO2 pin is re-defined to a PME_Input. Its polarity is controlled by MIC[4]. It sets the sticky PME_Status bit in Function1	W	RW	00
7:6	MIO3 Configuration Register. 00 = MIO3 is a non-inverting input pin 01 = MIO3 is an inverting input pin 10 = MIO3 is an output pin driving ‘0’ 11 = MIO3 is an output pin driving ‘1’	W	RW	00
9:8	MIO4 Configuration Register. 00 = MIO4 is a non-inverting input pin 01 = MIO4 is an inverting input pin 10 = MIO4 is an output pin driving ‘0’ 11 = MIO4 is an output pin driving ‘1’	W	RW	00
11:10	MIO5 Configuration Register. 00 = MIO5 is a non-inverting input pin 01 = MIO5 is an inverting input pin 10 = MIO5 is an output pin driving ‘0’ 11 = MIO5 is an output pin driving ‘1’	W	RW	00
13:12	MIO6 Configuration Register. 00 = MIO6 is a non-inverting input pin 01 = MIO6 is an inverting input pin 10 = MIO6 is an output pin driving ‘0’ 11 = MIO6 is an output pin driving ‘1’	W	RW	00

Bits	Description	Read/Write		Reset
		EEPROM	PCI	
15:14	MIO7 Configuration Register. 00 = MIO7 is a non-inverting input pin 01 = MIO7 is an inverting input pin 10 = MIO7 is an output pin driving ‘0’ 11 = MIO7 is an output pin driving ‘1’	W	RW	00
17:16	MIO8 Configuration Register. 00 = MIO8 is a non-inverting input pin 01 = MIO8 is an inverting input pin 10 = MIO8 is an output pin driving ‘0’ 11 = MIO8 is an output pin driving ‘1’	W	RW	00
19:18	MIO9 Configuration Register. 00 = MIO9 is a non-inverting input pin 01 = MIO9 is an inverting input pin 10 = MIO9 is an output pin driving ‘0’ 11 = MIO9 is an output pin driving ‘1’	W	RW	00
21:20	MIO10 Configuration Register. 00 = MIO10 is a non-inverting input pin 01 = MIO10 is an inverting input pin 10 = MIO10 is an output pin driving ‘0’ 11 = MIO10 is an output pin driving ‘1’	W	RW	00
Bit 25:22	Reserved. The device driver must write zeros to these bits	W	R/W	0000
Bit 26	Reserved	–	–	1
Bit 27	Reserved	-	R	X
Bit 28	Reserved	-	R	1
Bit 29	Reserved	–	–	0
Bit 30	Reserved	–	–	0
Bit 31	Parallel Port Filter Disable When set (1) disables the noise filters on the parallel port data lines and status lines. Filters are enabled by default. This bit is only relevant for the parallel port	W	R/W	0

Local Bus Timing Parameter Register 1 ‘LT1’ (Offset 0x08)

The Local Bus Timing Parameter registers (LT1 and LT2) define the operation and timing parameters used by the local bus. The timing parameters are programmed in 4-bit registers to define the assertion/de-assertion of the local bus control signals. The value programmed in these registers defines the number of PCI clock cycles after a reference cycle when the events occur, where the reference cycle is defined as two clock cycles after the master asserts the IRDY# signal. The following arrangement provides a flexible approach for you to define the desired bus timing of your peripheral devices. The timings refer to I/O or memory mapped access to BAR0 and BAR1 of Function1.

Bits	Description	Read/Write		Reset
		EEPROM	PCI	
3:0	Read Chip-select Assertion (Intel-type interface). Defines the number of clock cycles after the reference cycle when the LBCS[3:0]# pins are asserted (low) during a read operation from the local bus.[1] These bits are unused in Motorola-type interface	W	RW	0h
7:4	Read Chip-select De-assertion (Intel-type interface). Defines the number of clock cycles after the reference cycle when the LBCS[3:0]# pins are de-asserted (high) during a read from the local bus. [1] These bits are unused in Motorola-type interface	W	RW	3h (2h for parallel port)
11:8	Write Chip-select Assertion (Intel-type interface). Defines the number of clock cycles after the reference cycle when the LBCS[3:0]# pins are asserted (low) during a write operation to the local bus. [1] These bits are unused in Motorola-type interface	W	RW	0h
15:12	Write Chip-select De-assertion (Intel-type interface). Defines the number of clock cycles after the reference cycle when the LBCS[3:0]# pins are de-asserted (high) during a write operation to the local bus. [1] Read-not-write de-assertion during write cycles (Motorola-type interface). Defines the number of clock cycles after the reference cycle when the LBRDWR# pin is de-asserted (high) during a write to the local bus [1]	W	RW	2h
19:16	Read Control Assertion (Intel-type interface). Defines the number of clock cycles after the reference cycle when the LBRD# pin is asserted (low) during a read from the local bus. [1] Read Data-strobe Assertion (Motorola-type interface). Defines the number of clock cycles after the reference cycle when the LBDS[3:0]# pins are asserted (low) during a read from the local bus [1]	W	RW	0h (1h for parallel port)
23:20	Read Control De-assertion (Intel-type interface). Defines the number of clock cycles after the reference cycle when the LBRD# pin is de-asserted (high) during a read from the local bus. [1] Read Data-strobe De-assertion (Motorola-type interface). Defines the number of clock cycles after the reference cycle when the LBDS[3:0]# pins are de-asserted (high) during a read from the local bus [1]	W	RW	3h (2h for parallel port)
27:24	Write Control Assertion (Intel-type interface). Defines the number of clock cycles after the reference cycle when the LBWR# pin is asserted (low) during a write to the local bus. [1] Write Data-strobe Assertion (Motorola-type interface). Defines the number of clock cycles after the reference cycle when the LBDS[3:0]# pins are asserted (low) during a write to the local bus [1]	W	RW	0h (1h for parallel port)
31:28	Write Control De-assertion (Intel-type interface). Defines the number of clock cycles after the reference cycle when the LBWR# pin is de-asserted (high) during a write to the local bus. [1] Write Data-strobe De-assertion (Motorola-type interface). Defines the number of clock cycles after the reference cycle when the LBDS[3:0]# pins are de-asserted (high) during a write cycle to the local bus [1]	W	RW	2h

Note 1: Only values in the range of 0h to Ah (0-10 decimal) are valid. Other values are reserved. See notes on the following page.

Local Bus Timing Parameter Register 2 'LT2' (Offset 0x0C)

Bits	Description	Read/Write		Reset
		EEPROM	PCI	
3:0	Write Data Bus Assertion. This register defines the number of clock cycles after the reference cycle when the LBD pins actively drive the data bus during a write operation to the local bus [1]	W	RW	0h
7:4	Write Data Bus De-assertion. This register defines the number of clock cycles after the reference cycle when the LBD pins go high-impedance during a write operation to the local bus [1,2]	W	RW	Fh
11:8	Read Data Bus Assertion. This register defines the number of clock cycles after the reference cycle when the LBD pins actively drive the data bus at the end of a read operation from the local bus [1]	W	RW	4h (2h for parallel port)
15:12	Read Data Bus De-assertion. This register defines the number of clock cycles after the reference cycle when the LBD pins go high-impedance during at the beginning of a read cycle from the local bus [1]	W	RW	0h
19:16	Reserved	-	R	0h
22:20	I/O Space Block Size of BAR0 in Function1 000 = Reserved 001 = 4 Bytes 010 = 8 Bytes 011 = 16 Bytes 100 = 32 Bytes 101 = 64 Bytes 110 = 128 Bytes 111 = 256 Bytes	W	R	'100' (='010' for parallel port)
26:23	Local Bus Chip–select Parameter 'Lower-Address-CS-Decode'. [2] I/O space in 8-bit local bus 0000 = A2 0001 = A3 0010 = A4 0011 = A5 0100 = A6 0101 = A7 0110 = A8 0111 = A9 I/O space in 8-bit local bus 1000 = Res 1001 = Res 1010 = Res 1011 = Res 1100 = Res 1101 = Res 1110 = Res 1111 = Res	W	RW	'0001' (='0010' for parallel port)
28:27	Reserved	W	R	00
29	Local Bus Software Reset. When this bit is a 1 the Local Bus Reset pin is activated. When this bit is 0 the Local Bus Reset pin is de-activated [3]	-	RW	0
30	Local Bus Clock Enable. When this bit is a 1 the local bus clock (LBCK) pin is enabled. When this bit is a 0 LBCK pin is permanently low. The local bus clock is a buffered PCI clock	W	RW	0
31	Bus Interface Type. When low (=0) the local bus is configured to Intel-type operation, otherwise it is configured to Motorola-type operation. Note that when Mode[1:0] is '01', this bit is hard wired to 0	W	RW	0

Note 1: Only values in the range of 0 to Ah (0-10 decimal) are valid. Other values are reserved as writing higher values causes the PCI interface to retry all accesses to the local bus as it is unable to complete the transaction in 16 PCI clock cycles.

Note 2: The Lower-Address-CS-Decode parameter bits are unused for memory access to the 8-bit local bus, which uses a fixed decoding to allocate 1K regions to 4 chip selects.

Note 3: Local bus, UARTs and the parallel port are all reset with PCI reset. In addition,you can issue the software reset command.

LT2[15:0] enable you to control the data bus during the idle periods. The default values configure the local bus data pins to remain forcing (LT2[7:4] = Fh). LT[15:8] are programmed to place the bus in high-impedance at the beginning of a read cycle and set it back to forcing at the end of the read cycle. For systems that require the data bus to stay in high-impedance, write an appropriate value in the range of 0h to Ah to LT2[7:4]. This places the data bus in high impedance at the end of the write cycle. Whenever the value programmed in LT2[7:4] does not equal Fh, the local bus controller ignores the setting of LT2[15:8] as the data bus is high-impedance outside write cycles. In this case, place external pull-ups on the data bus pins LBD[7:0].

While the configuration data is read from the external EEPROM, the LBD pins remain in the high-impedance state.

The timing registers define the local bus timing parameters based on signal changes relative to a reference cycle which is defined as two PCI clock cycles after IRDY# is asserted for the first time in a frame. The following parameters are fixed relative to the reference cycle.

The local bus address pins LBA[7:0] are asserted during the reference cycle. In a write operation, the local bus data is available during the reference cycle, however I/O buffers change direction as programmed in LT2[3:0].

In a Motorola type bus write operation, the read-not-write pin (LBRDWR#) is asserted (low) during the reference cycle. In a read cycle this pin remains high throughout the duration of the operation.

The default settings in LT1 and LT2 registers provide one PCI clock cycle for address and chip-select to control signal set-up time, one clock cycle for address and chip-select from control signal hold time, two clock cycles of pulse duration for read and write control signals and one clock cycle for data bus hold time. These parameters are acceptable for using external OX16C950 and OX16C954 devices connected to the local bus, in Intel mode. Some redefinition is required if the bus is to be operated in Motorola mode.

Take great care when programming the local bus timing parameters. For example, defining a value for chip-select assertion which is larger that the value defined for chip-select de-assertion or defining a chip-select assertion value which is greater than control signal assertion results in obvious invalid local bus cycles.

UART Receiver FIFO Levels 'URL' (Offset 0x10)

The receiver FIFO level of the two internal UARTs are shadowed in local configuration registers as follows:

Bits	Description	Read/Write		Reset
		EEPROM	PCI	
7:0	UART0 Receiver FIFO Level (RFL[7:0])	-	R	0x00h
15:8	UART1 Receiver FIFO Level (RFL[7:0])	-	R	0x00h
31:16	Reserved	-	R	0x0000h

UART Transmitter FIFO Levels 'UTL' (Offset 0x14)

The transmitter FIFO level of the two UARTs are shadowed in Local configuration registers as follows:

Bits	Description	Read/Write		Reset
		EEPROM	PCI	
7:0	UART0 Transmitter FIFO Level (TFL[7:0])	-	R	0x00h
15:8	UART1 Transmitter FIFO Level (TFL[7:0])	-	R	0x00h
31:16	Reserved	-	R	0x0000h

UART Interrupt Source Register 'UIS' (Offset 0x18)

The UART Interrupt Source register is described below:

Bits	Description	Read/Write		Reset
		EEPROM	PCI	
5:0	UART0 Interrupt Source Register (ISR[5:0])	-	R	01h
11:6	UART1 Interrupt Source Register (ISR[5:0])	-	R	01h
26:12	Reserved	-	R	XXXh
27	UART0 good-data status	-	R	1
28	UART1 good-data status	-	R	1
30:29	Reserved	-	R	3h
31	Global Good-data Status. This bit is the logical AND of bits 27 to 28, i.e. it is set if Good-Data Status of all internal UARTs is set.	-	R	1

Good-data status for a given internal UART is set when all of the following conditions are met:

- ISR reads a level0 (no-interrupt pending), a level 2a (receiver data available), a level 2b (receiver time-out) or a level 3 (transmitter THR empty) interrupt
- LSR[7] is clear so there is no parity error, framing error or break in the FIFO
- LSR[1] is clear so no over-run error has occurred

If the device driver software reads the receiver FIFO levels (URL) followed by this register, then if good-data status for a given channel is set, the driver can remove the number of bytes indicated by the FIFO level without the need to read the line status register for that channel. This feature enhances the driver efficiency.

For a given channel, if the good-data status bit is not set, then the software driver should examine the corresponding ISR bits. For example if bit 28 is low, then the driver should examine bits 11 down to 6 to obtain the ISR[5:0] for UART1. If the ISR indicates a level 4 or higher interrupt, the interrupt is due to a change in the state of modem lines or detection of flow control characters. The device driver software should then take appropriate measures as would in any other 550/950 driver. When ISR indicates a level 1 (receiver status) interrupt then the driver can examine the Line Status Register (LSR) of the relevant channel. Since reading the LSR clears LSR[7], the device driver-software should either flush or empty the contents of the receiver FIFO, otherwise the good-data status is no longer valid.

The UART Receiver FIFO Level (URL), UART Transmitter FIFO Level (UTL), UART Interrupt Source register (UIS) and Global Interrupt Status register (GIS) are allocated adjacent address offsets (10h to 1Ch) in the Base Address Register. The device driver-software can read all of the above registers in single burst read operation. The location offset of the registers is such that the FIFO levels are usually read before the status registers so that the status of the N characters indicated in the receiver FIFO levels are valid.

Global Interrupt Status and Control Register 'GIS' (Offset 0x1C)

Bits	Description	Read/Write		Reset
		EEPROM	PCI	
1:0	UART Interrupt Status. These bits reflect the internal interrupt states of UART1 to UART0 respectively [1]	-	R	0x0h
3:2	Reserved	-	R	0x0h
4	MIO0 Status (when device mode = 0). This bit reflects the state of the internal MIO[0]. The internal MIO[0] reflects the non-inverted or inverted state of MIO0 pin.[2] When device mode = 1, this reflects state of the parallel port interrupt	- -	R R	X 0
5	MIO1 Status (LCC[6:5]='00'). This bit reflects the state of the internal MIO[1]. The internal MIO[1] reflects the non-inverted or inverted state of MIO1 pin.[2] Function 0 Power-down Interrupt (LCC[6:5] ≠'00'). In this mode this is a sticky bit. When set, it indicates a power-down request issued by Function 0 and would normally have asserted a PCI interrupt if bit 21 was set. Reading this bit clears it		R	X 0
14:6	MIO[10:2] Status. These bits reflect the state of the internal MIO[10:2]. The internal MIO[10:2] reflect the non-inverted or inverted state of MIO[10:2] pins respectively [2]	-	R	XXXh
15	Reserved. Returns 'X'	-	R	X
17:16	UART Interrupt Mask. When set (1) these bits enable the two internal UARTs to assert a PCI interrupt respectively. When cleared (=0) they prevent the respective UART from asserting a PCI interrupt [3]	W	RW	3h
19:18	Reserved	W	RW	3h
20	MIO[0] Interrupt Mask (When device mode = 0). When set (=1) this bit enables MIO0 pin to assert a PCI interrupt. When cleared (=0) it prevents MIO0 pin from asserting a PCI interrupt. [2] Parallel Port Interrupt Mask (When device mode = 1). When set (=1) this bit enables the parallel port to assert a PCI interrupt. When cleared (=0) it prevents the parallel port from asserting a PCI interrupt	W W	RW RW	1 1
21	MIO[1] Interrupt Mask (LCC[6:5]='00'). When set (=1) this bit enables MIO1 pin to assert a PCI interrupt. When cleared (=0) it prevents MIO1 pin from asserting a PCI interrupt. [2] Function 0 Power-down Interrupt Mask (LCC[6:5] ≠'00'). When set (=1) this bit enables the power-down logic in Function0 to assert a PCI interrupt. When cleared (=0) it prevents the power-down logic in Function 0 from asserting a PCI interrupt	W W	RW RW	1 0
30:22	MIO Interrupt Mask. When set (=1) these bits enable each MIO[10:2] pin to assert a PCI interrupt respectively. When cleared (=0) they prevent the respective pins from asserting a PCI interrupt [2]	W	RW	1FFh
31	Reserved	W	RW	1

Note 1: GIS[1:0] are the inverse of UIS[6] and UIS[0] respectively. Systems that do not require the local bus or parallel port need not read this register to identify the source of the interrupt as long as they read the UIS (offset 18h) register.

Note 2: The returned value is either the direct state of the corresponding MIO pin or its inverse as configured by the Multi-purpose I/O Configuration register 'MIC' (offset 0x04). As the internal MIO can assert a PCI interrupt, the inversion feature can define each external interrupt to be defined as active-low or active-high, as controlled by the MIC register.

When the MIO[0] pin has been set-up as an input or output, this can be made to generate an interrupt when the MIO[0] Interrupt Mask (bit 20) is set (=1). This bit enables MIO[0] pin to assert a PCI interrupt.

When the MIO[1] pin has been set-up as an input or output, this can be made to generate an interrupt when the MIO[1] Interrupt Mask (bit 21) is set (=1). This bit enables MIO[1] pin to assert a PCI interrupt.

Note 3: The UART Interrupt Mask register bits are all set after a hardware reset to enable the interrupt from both the internal UARTs. This caters for generic device driver software that does not access the Local Configuration Registers. The default setting for UART Interrupt Mask bits can be changed using the serial EEPROM. Note that even though by default the UART interrupts are enabled in this register, since after a reset the IER registers of individual UARTs disables all interrupts, a PCI interrupt is not asserted after a hardware reset.

PCI Interrupts

Interrupts in PCI systems are level-sensitive and can be shared. There are up to 13 sources of interrupts in the OXuPCI952, one using each UART channel and up to 11 from the multi-purpose I/O pins (MIO10 to MIO0). The parallel port and MIO[0] pin share the same interrupt status bit (GIS[4]). The PCI power management power-down interrupt for the internal UARTs (Function0) and the MIO[1] pin share the status bit GIS[5]. The local bus uses the MIO pins to pass interrupts to the PCI controller.

Function 0 and Function 1 interrupts are set to assert on the INTA# line, by default. These default routings may be modified by writing to the interrupt pin field in the configuration registers using the serial EEPROM facility, for example, to disable an interrupt pin. But functional interrupts can only happen on the INTA line. The interrupt pin field is normally considered a hard-wired read-only value in PCI. It indicates to system software which PCI interrupt pin (if any) is used by a function. The interrupt pin may only be modified using the serial EEPROM facility, and you must not invoke any combination which violates the PCI specification. If in doubt, use the default routings. The following table relates the interrupt pin field to the device pin used.

Interrupt Pin	Device Pin Used
0	None
1	INTA#
2 to 255	Reserved

During the system initialization process and PCI device configuration, system-specific software reads the interrupt pin field to determine which (if any) interrupt pin is used by each function. It programs the system interrupt router to logically connect this PCI interrupt pin to a system-specific interrupt vector (IRQ). It then writes this routing information to the interrupt line field in the function's PCI configuration space. Device driver software must then hook the interrupt using the information in the interrupt line field.

Interrupt status for all thirteen sources of interrupt is available using the GIS register in the Local Configuration Register set, which can be accessed using I/O or memory accessed from both logical functions. This facility enables each function to snoop on interrupts asserted from the other function, regardless of the interrupt routing.

The interrupt from each UART channel is enabled using the IER register and the MCR register for that UART. If the interrupt is enabled and active, then the device drives the PCI interrupt pin low. Generic device driver software uses the IER register to enable interrupts. The OXuPCI952 offers additional interrupt masking ability using GIS[17:16]. An internal UART channel may assert a PCI interrupt if the interrupt is enabled by IER and GIS[17:16].

All interrupts can be enabled or disabled individually using the GIS register set in the local configuration registers. When an MIO pin is enabled, an external device can assert a PCI interrupt by driving that pin. The sense of the MIO external interrupt pins (active-high or active-low) is defined in the MIC register. The parallel port can also assert an interrupt (but this effectively disables the MIO[0] interrupt).

Power Management

The OXuPCI952 is compliant with the *PCI Power Management Specification Revision 1.1*. This is indicated by both functions by their Power Management Capabilities Register (PMC), which defaults to 6C02(h).

Each logical function implements its own set of Power Management registers and supports the power states D0, D2 and D3hot. The power state D1 is not supported by any functions. PME# generation from D2 and D3hot is available, but wakeup from D3cold is not supported.

These features are summarized in the following table.

Power State	State Supported	PME# Generation Supported
D0 uninitialized	Yes	No [1]
D0 initialized	Yes	No [1]
D1	No	No
D2	Yes	Yes
D3hot	Yes	Yes
D3cold	No	No

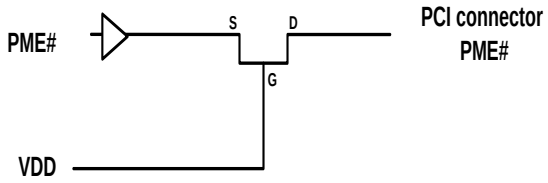
Note 1: PMSCR register indicates yes to satisfy OS value requirements.

Power management is accomplished by handling the power-down and power-up ("power management event") requests. These are asserted on the relevant function's interrupt pin and the PME# pin respectively. Each function can assert the PME# pin independently.

Power-down requests are not defined by any of the PCI power management specifications. They are device-specific features and require a custom device driver implementation. The device driver can either implement the power-down itself or use the special interrupt and power-down features offered by the device to determine when the function or device is ready for power-down.

It is worth noting that the PME# pin can, in certain cases, activate the PME# signal when all power is removed from the device. This causes the PC to wake up from low-power state D3(cold). To ensure full cross-compatibility with system board implementations, the use of an isolator FET is recommended (see diagram). If power management capabilities are not required, the PME# pin can be treated as no-connect.

Figure 4 PME# Isolator Circuitry



Power Management of Function 0

Provided that the necessary controls are set in the device's local configuration registers (LCC and GIS), the two internal UARTs can be programmed to issue power-down requests and/or 'wakeup' requests (power management events), for function 0.

Function 0 can be configured to monitor the activity of the 2 serial channels, and issue a power-down interrupt when both of the UARTs are inactive (no interrupts pending and both transmitters and receivers are idle).

Only when both the serial channels are indicating a power-down request, does the internal power management circuitry wait for a period of time as programmed into the power-down filter time. This time is defined by the local configuration register, LCC[7:5]). If the power-down requests remain valid for this time (this means that both serial channels are still inactive) then the OXuPCI952 issues a power-down interrupt on this function's interrupt pin, if this option is enabled. Alternatively, the device driver can poll function 0's power-down status field in the local configuration register GIS[5] to determine a power-down request. The power-down filter stops the UARTs from issuing too many power-down interrupts whenever the UARTs activity is intermittent.

Upon a power-down interrupt, the device driver can change the power state of the device (function 0) as required. Note that the power state of function 0 is only changed by the device driver and at no point does the OXuPCI952 change its own power state. The power-down interrupt merely informs the device driver that this logical function is ready for power down. Before placing the device into

the lower power states, the driver must provide the means for the function to generate a ‘wakeup’ (power management) event.

Whenever the device driver changes function 0’s power state to state D2 or D3, the device takes the following actions:

- The internal clock to the internal UARTs is shut down
- PCI interrupts are disabled regardless of the values
- contained in the GIS registers
- Access to I/O or Memory BARs is disabled

However, access to the configuration space is still enabled.

The device driver can optionally assert/de-assert any of its selected (design dependent) MIO pins to switch off VCC, disable other external clocks, or activate shut-down modes.

The device can only issue a wakeup request (a power management event, PME#) if it is enabled by this function’s PME_En bit, bit-8 of the PCI Power Management Register PMCSR. PME# assertion is immediate and does not use the power-down filter timer. It operates even if the power-down filter time is set to disabled.

Like power down, wake-up requests for function 0 can be generated by each serial channel. The means to generate wake-up events from these sources must be set up prior to placing this function into the power-down states D2 or D3 (including setting of the PME_En bit).

For each of the two UARTs, when the device (function 0) is in the power state D3, only activity on the serial channel’s RI# line (the trailing edge of a pulse) generate a wake-up event. When the device (function 0) is in the power state D2, then wake-ups are configurable. In this case, a change in the state of any modem line (which is enabled by a 16C950-specific mask bit) or a change in the state of the serial input line (again, if enabled by a 16C950-specific mask bit) can issue a wake up request on the PME# pin. It is worth noting that after a hardware reset all of these mask bits are cleared to enable wake up assertion from all modem lines and the SIN line when in the power state D2. As the wake up operation from D2 requires at least one mask bit to be enabled, the device driver can for example disable the masks with the exception of the ring indicator, so only a modem ring can wake up the computer. In the case of a wake up request from the serial input line EXT_DATA_IN (from the power state D2), the clock for that channel is turned on so serial data framing can be maintained.

When function 0 issues a wake up request from the serial channels, the PME_Status bit in this function’s PCI power management registers (PMCSR[15]) is set. This is a sticky bit which is only cleared by writing a ‘1’ to it. While PME_En (PMCSR[8]) remains set, the PME_Status continues to assert the PME# pin to inform the device driver that a power management wake up event has occurred. After a wake up event is signaled, the device driver is expected to return this function to the D0 power state.

Power Management of Function 1

Provided that the necessary controls are set in the device's local configuration registers (MIC and GIS), up to 8 multi-purpose pins (MIO[10:3]) can be programmed to issue power-down requests but only MIO[2] can be set to generate wake-up requests (power management events), for function 1. The parallel port or the local bus function are not capable of issuing power-down requests or power management events but can be placed in a low power state through power management involving the MIO pins.

The state of the MIO pin(s) that issues a power-down request is controlled by the MIC register. This can be active high or active low. This state can be the same MIO state that asserts function 1's interrupt pin for normal functionality. The assertion of the MIO pins results in a function 1 power-down request being made immediately. There is no power-down filtering time associated with function 1.

The power-down request can be issued on the function's interrupt pin, if this option is enabled.

Upon a power-down interrupt, the device driver can change the power state of the device (function 1) as required. Note that the power state of function 1 is only changed by the device driver and at no point does the OXuPCI952 change its own power state. The power-down interrupt merely informs the device driver that this logical function is ready for power down. Before placing the device into the lower power states, the driver must provide the means for the function to generate a wakeup (power management) event.

Whenever the device driver changes function 1's power state to state D2 or D3, the device takes the following actions:

- Parallel port placed in a low power mode
- Local bus function placed in a low power mode
- PCI interrupts are disabled regardless of the values contained in the GIS registers
- Access to I/O or memory BARs is disabled

However, access to the configuration space is still enabled.

Function 1 can only issue a wakeup request (power management event) if it is enabled by this function's PME_En bit, bit-8 of the PCI Power Management Register PMCSR.

Wakeup requests for function 1 can only be generated by the multi-purpose I/O pin MIO[2]. Prior to placing this function into the power-down states D2 or D3, wake-up events from this source must be enabled.

The state of the MIO[2] pin that results in wake-up requests is determined by the settings in the local configuration register MIC. As soon as the correct logic is invoked then a power management event (wakeup) is asserted. The PME# event is immediate.

When function 1 issues a wakeup request, the PME_Status bit in this function's PCI power management registers (PMCSR[15]) is set. This is a sticky bit which is only cleared by writing a '1' to it. While PME_En (PMCSR[8]) remains set, the

PME_Status continues to assert the PME# pin to inform the device driver that a power management wake up event has occurred. After a wake up event is signaled, the device driver is expected to return this function to the D0 power state

Table 10 Function 0 (UARTs) Power Down Interrupt Settings

LCC[6:5]	GIS[21]	Power-down Filter Time	Operation
00	X	N/A	Function 0 power-down interrupt is disabled. MIO[1] can assert a PCI interrupt if GIS[21] is set
01	0	4 s	Function 0 power-down interrupt is disabled. GIS[5] reflects the state of the internal power-down mode
10	0	129 s	Polling operation. MIO[1] interrupt is disabled
11	0	518 s	
01	1	4 s	Function 0 power-down is enabled. GIS[5] reflects the state of the internal power-down mode
10	1	129 s	MIO[1] interrupt is disabled
11	1	515 s	

Table 11 Function 1 (Local Bus) Wake-up Configuration

LCC[7]	MIC[5:4]	MIO2 Rising	MIO2 Falling	Function1 PME_Status
0	XX	X	X	Remains unchanged
1	00	yes	X	Gets set
1	00	no	X	Remains unchanged
1	01	X	Yes	Gets set
1	01	X	No	Remains unchanged

Universal Voltage

The OXuPCI952 is fully compatible with both 3.3V and 5V PCI interfaces. To support universal voltage (3.3V and 5V), the PCI I/O buffers on the OXuPCI952 must be supplied with power from the VCC (IO) pins on the PCI connector.

Internal OX16C950 UARTs

Each of the two UART channels in the OXuPCI952 operates individually as an OX16C950 high-performance serial port. Each channel has its own full set of registers, but all share a common clock and FIFOSEL pin. After a device reset, a common configuration state is loaded into both channels, but after this time each channel can be operated individually through its own 8-byte block of addressable space.

Operation – Mode Selection

Each channel is backward compatible with the 16C450, 16C550, 16C654 and 16C750 UARTs. The operation of the ports depends on a number of mode settings, which are referred to throughout this section. The modes, conditions and corresponding FIFO depth are given below:

Table 12 UART Mode Configuration

UART Mode	FIFO Size	FCR[0]	Enhanced Mode (EFR[4]=1)	FCR[5] (guarded with LCR[7] = 1)	FIFOSEL Pin
450	1	0	X	X	X
550	16	1	0	0	0
Extended 550	128	1	0	X	1
650	128	1	1	X	X
750	128	1	0	1	0
950 [1]	128	1	1	X	X

Note 1: 950 mode configuration is identical to a 650 configuration.

450 Mode

After a hardware reset, bit 0 of the FIFO Control Register (‘FCR’) is cleared, hence the UARTs are compatible with the 16C450. The transmitter and receiver FIFOs (referred to as the Transmit Holding Register and Receiver Holding Register respectively) have a depth of one. This is referred to as Byte mode. When FCR[0] is cleared, all other mode selection parameters are ignored.

550 Mode

Connect FIFOSEL to GND. After a hardware reset, writing a 1 to FCR[0] increases the FIFO size to 16, providing compatibility with 16C550 devices.

Extended 550 Mode

Connect FIFOSEL to VDD. Writing a 1 to FCR[0] increases the FIFO size to 128, thus providing a 550 device with 128 deep FIFOs.

750 Mode

For compatibility with 16C750, connect FIFOSEL to GND. Writing a 1 to FCR[0] increases the FIFO size to 16. In a similar fashion to 16C750, the FIFO size can be further increased to 128 by writing a 1 to FCR[5]. Note that access to FCR[5] is protected by LCR[7]. i.e., to set FCR[5], software should first set LCR[7] to temporarily remove the guard. Once FCR[5] is set, the software should clear LCR[7] for normal operation.

The 16C750 additional features are available as long as the UART is not put into Enhanced mode; i.e. ensure EFR[4] = '0'. These features are:

- Deeper FIFOs
- Automatic RTS/CTS out-of-band flow control
- Sleep mode

650 Mode

The 950 UART is compatible with the 16C650 when EFR[4] is set, i.e. the device is in Enhanced mode. As 650 software drivers usually put the device in Enhanced mode, running 650 drivers on the one of the UART channels results in 650 compatibility with 128 deep FIFOs, as long as FCR[0] is set. This is regardless of the state of the FIFOSEL pin. Note that the 650 emulation mode of the OXuPCI952 provides 128-deep FIFOs rather than the 32 provided by a legacy 16C654.

In Enhanced (650) mode the device has the following features available over those provided by a generic 550. (Note that some of these are similar to those provided in 750 mode, but are enabled using different registers.)

- Deeper FIFOs
- Sleep mode
- Automatic in-band flow control
- Special character detection
- Infra-red "IrDA-format" transmit and receive mode
- Transmit trigger levels
- Optional clock prescaler

950 Mode

The additional features offered in 950 mode generally only apply when the UART is in Enhanced mode (EFR[4]='1'). Provided FCR[0] is set, in Enhanced mode the FIFO size is 128 regardless of the state of FIFOSEL.

Note that 950 mode configuration is identical to that of 650 mode, however additional 950 specific features are enabled using the Additional Control Register 'ACR'.

In addition to larger FIFOs and higher baud rates, the enhancements of the 950 mode over 650 emulation mode are:

- Selectable arbitrary trigger levels for the receiver and transmitter FIFO interrupts
- Improved automatic flow control using selectable arbitrary thresholds
- DSR#/DTR# automatic flow control
- Transmitter and receiver can be optionally disabled
- Software reset of device
- Readable FIFO fill levels
- Optional generation of an RS-485 buffer enable signal
- Four-byte device identification (0x16C9500A)
- Readable status for automatic in-band and out-of-band flow control
- External 1x clock modes
- Flexible “M+N/8” clock prescaler
- Programmable sample clock to allow data rates up to 15 Mbps
- 9-bit data mode
- Readable FCR register

The 950 trigger levels are enabled when ACR[5] is set where bits 4 to 7 of FCR are ignored. Then arbitrary trigger levels can be defined in RTL, TTL, FCL and FCH registers. The Additional Status Register (‘ASR’) offers flow control status for the local and remote transmitters. FIFO levels are readable using RFL and TFL registers.

The UART has a flexible prescaler capable of dividing the system clock by any value between 1 and 31.875 in steps of 0.125. It divides the system clock by an arbitrary value in “M+N/8” format, where M and N are 5- and 3-bit binary numbers programmed in CPR[7:3] and CPR[2:0] respectively. This arrangement offers a great deal of flexibility when choosing an input clock frequency to synthesize arbitrary baud rates. The default division value is 4 to provide backward compatibility with 16C650 devices.

You can apply an external 1x (or Nx) clock for the transmitter and receiver to the RI# and DSR# pin respectively. The transmitter clock may instead be asserted on the DTR# pin. The external clock options are selected through the CKS register (offset 0x02 of ICR).

It is also possible to define the over-sampling rate used by the transmitter and receiver clocks. The 16C450/16C550 and compatible devices employ 16x over-sampling, where there are 16 clock cycles per bit. However the 95x UART channels can employ any over-sampling rate from 4 to 16 by programming the TCR register. This allows the data rates to be increased to 460.8 Kbps using a 1.8432 MHz clock, or 15 Mbps using a 60 MHz clock. The default value after a reset for this register is 0x00, which corresponds to a 16 cycle sampling clock. Writing 0x01, 0x02 or 0x03 also results in a 16 cycle sampling clock. To program the value to any value from 4 to 15 it is necessary to write this value into the TCR. This means that to set the device to a 13 cycle sampling clock, you must write 0x0D to TCR.

The UARTs also offer 9-bit data frames for multi-drop industrial applications.

Register Description Tables

Each UART is accessed through an 8-byte block of I/O space (or through memory space). Since there are more than 8 registers, the mapping is also dependent on the state of the Line Control Register 'LCR' and Additional Control Register 'ACR':

- LCR[7]=1 enables the divider latch registers DLL and DLM
- LCR specifies the data format used for both transmitter and receiver. Writing 0xBF (an unused format) to LCR enables access to the 650 compatible register set. Writing this value sets LCR[7] but leaves LCR[6:0] unchanged. Therefore, the data format of the transmitter and receiver data is not affected. Write the desired LCR value to exit from this selection
- ACR[7]=1 enables read access to the 950 specific status registers
- ACR[6]=1 enables read access to the Indexed Control Register set (ICR) registers

Table 13 Standard 550 Compatible Registers (Sheet 1 of 2)

Register Name	Address	R/W	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
THR [1]	000	W	Data to be transmitted							
RHR [1]	000	R	Data received							
IER [1,2] 650/950 Mode	001	R/W	CTS interrupt mask	RTS interrupt mask	Special Char. Detect	Sleep mode	Modem interrupt mask	Rx Stat interrupt mask	THRE interrupt mask	RxRDY interrupt mask
550/750 Mode			Unused		Alternate sleep mode					
FCR [3] 650 mode	010	W	RHR Trigger Level		THR Trigger Level		Tx Trigger Enable	Flush THR	Flush RHR	Enable FIFO
750 mode			RHR Trigger Level		FIFO Size	Unused				
950 mode			Unused							
ISR [3]	010	R	FIFOs enabled		Interrupt priority (Enhanced mode)		Interrupt priority (All modes)			Interrupt pending
LCR [4]	011	R/W	Divisor latch access	Tx break	Force parity	Odd / even parity	Parity enable	Number of stop bits	Data length	

Table 13 Standard 550 Compatible Registers (Sheet 2 of 2)

Register Name	Address	R/W	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
MCR [3,4] 550/750 Mode	100	R/W	Unused		CTS & RTS Flow Control	Enable Internal Loop Back	OUT2 (Int En)	OUT1	RTS	DTR
650/950 Mode			Baud prescale	IrDA mode	XON-Any					
LSR [3,5] Normal	101	R	Data Error	Tx Empty	THR Empty	Rx Break	Framing Error	Parity Error	Overrun Error	RxRDY
9-bit data mode								9^{th} Rx data bit		
MSR [3]	110	R	DCD	RI	DSR	CTS	Delta DCD	Trailing RI edge	Delta DSR	Delta CTS
SPR [3] Normal	111	R/W	Temporary data storage register and Indexed control register offset value bits							
9-bit data mode			Unused							9^{th} Tx data bit
Additional Standard Registers – These registers require divisor latch access bit (LCR[7]) to be set to 1.										
DLL	000	R/W	Divisor latch bits [7:0] (Least significant byte)							
DLM	001	R/W	Divisor latch bits [15:8] (Most significant byte)							

Table 14 650 Compatible Registers

Register Name	Address	R/W	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
To access these registers LCR must be set to 0xBF										
EFR	010	R/W	CTS flow control	RTS Flow control	Special char detect	Enhance mode	In-band flow control mode			
XON1	100	R/W	XON Character 1							
9-bit mode			Special Character 1							
XON2	101	R/W	XON Character 2							
9-bit mode			Special Character 2							
XOFF1	110	R/W	XOFF Character 1							
9-bit mode			Special Character 3							
XOFF2	111	R/W	XOFF Character 2							
9-bit mode			Special Character 4							

Table 15 950 Specific Registers										
Register Name	Address	R/W	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
ASR [1,6,7]	001	R/W [7]	Tx Idle	FIFO size	FIFO-SEL	Special Char Detect	DTR	RTS	Remote Tx Disabled	Tx Disabled
RFL [6]	011	R	Number of characters in the receiver FIFO							
TFL [3,6]	100	R	Number of characters in the transmitter FIFO							
ICR [3,8,9]	101	R/W	Data read/written depends on the value written to the SPR prior to the access of this register							

Register access notes:

Note 1: Requires LCR[7] = 0.

Note 2: Requires ACR[7] = 0.

Note 3: Requires that last value written to LCR was not 0xBF.

Note 4: To read this register ACR[7] must be = 0.

Note 5: To read this register ACR[6] must be = 0.

Note 6: Requires ACR[7] = 1.

Note 7: Only bits 0 and 1 of this register can be written.

Note 8: To read this register ACR[6] must be = 1.

Note 9: This register acts as a window through which to read and write registers in the Indexed Control Register set.

Table 16 Indexed Control Register Set

Register Name	SPR Offset [10]	R/W	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
ACR	0x00	R/W	Additional Status Enable	ICR Read Enable	950 Trigger Level Enable	DTR definition and control		Auto DSR Flow Control Enable	Tx Disable	Rx Disable
CPR	0x01	R/W	5 Bit “integer” part of clock prescaler					3 Bit “fractional” part of clock prescaler		
TCR	0x02	R/W	Unused				4 Bit N-times clock selection bits [3:0]			
CKS	0x03	R/W	Tx 1x Mode	Tx CLK Select	BDOUT on DTR	DTR 1x Tx CLK	Rx 1x Mode	0	Receiver Clock Sel[1:0]	
TTL	0x04	R/W	Unused	Transmitter Interrupt Trigger Level (0-127)						
RTL	0x05	R/W	Unused	Receiver Interrupt Trigger Level (1-127)						
FCL	0x06	R/W	Unused	Automatic Flow Control Lower Trigger Level (1-127)						
FCH	0x07	R/W	Unused	Automatic Flow Control Higher Trigger level (1-127)						
ID1	0x08	R	Hardwired ID byte 1 (0x16)							
ID2	0x09	R	Hardwired ID byte 1 (0xC9)							
ID3	0x0A	R	Hardwired ID byte 1 (0x50)							
REV	0x0B	R	Hardwired revision byte (0x12)							
CSR	0x0C	W	Writing 0x00 to this register will reset the UART (except the CKS register)							
NMR	0x0D	R/W	Unused		9^{th} Bit SChar 4	9^{th} Bit SChar 3	9^{th} Bit SChar 2	9^{th} Bit SChar 1	9^{th}-bit Int. En.	9 Bit Enable
MDM	0x0E	R/W	0	0	SIN Wakeup disable	Modem Wakeup Disable	Δ DCD Wakeup disable	Trailing RI edge disable	Δ DSR Wakeup disable	Δ CTS Wakeup disable
RFC	0X0F	R	FCR[7]	FCR[6]	FCR[5]	FCR[4]	FCR[3]	FCR[2]	FCR[1]	FCR[0]
GDS	0X10	R	0	0	0	0	0	0	0	Good data status
PIX	0x12	R	Hardwired Port Index (0x00, 0x01, 0x02, 0x03 according to which UART)							
CKA	0x13	R/W	Unused			Res. Write ‘0’	Res. Write ‘0’	Invert DTR signal	Invert internal tx clock	Invert internal rx clock
RS485_DLYEN	0x14	R/W	Unused							RS-DEL
RS485_DLYCNT	0x15	R/W	RS485 Delay Bit Count				RS485 Delay Phase Count			

Note 10: The SPR offset column indicates the value that must be written into SPR prior to reading/writing any of the Indexed Control Registers using ICR. Offset values not listed in the table are reserved for future use and must not be used.

To write to any of the Indexed Controlled Registers:

- Ensure that the last value written to LCR was not 0xBF (reserved for 650 compatible register access value)
- Write the desired offset to SPR (address 111b)
- Write the desired value to ICR (address 101b)

To read from ICR registers:

- Ensure that the last value written to LCR was not 0xBF (see above)
- Write 0x00 offset to SPR to select ACR
- Set bit 6 of ACR (ICR read enable) by writing x1xxxxxxb to address 101b. Ensure that other bits in ACR are not changed
- (Software drivers should keep a copy of the contents of the ACR elsewhere since reading ICR involves overwriting ACR!)
- Write the desired offset to SPR (address 111b)
- Read the desired value from ICR (address 101b)
- Write 0x00 offset to SPR to select ACR
- Clear bit 6 of ACR bye writing x0xxxxxxb to ICR, thus enabling access to standard registers again

UART Reset Configuration

Hardware Reset

After a hardware reset, all writable registers are reset to 0x00, with the following exceptions:

- DLL, which is reset to 0x01
- CPR, which is reset to 0x20

The state of read-only registers following a hardware reset is as follows:

- RHR[7:0]: Indeterminate
- RFL[6:0]: 0000000_2
- TFL[6:0]: 0000000_2
- LSR[7:0]: 0x60, signifying that both the transmitter and the transmitter FIFO are empty
- MSR[3:0]: 0000_2
- MSR[7:4]: Dependent on modem input lines DCD, RI, DSR and CTS respectively
- ISR[7:0]: 0x01, i.e. no interrupts are pending
- ASR[7:0]: $1xx00000_2$

The reset state of output signals are given below:

Signal	Reset State
SOUT	Inactive High
RTS#	Inactive High
DTR#	Inactive High

Software Reset

An additional feature available in the OX16C950 UART is software resetting of the serial channel. This command has the same effect on a single channel as a hardware reset except it does not reset the clock source selections (i.e. CKS register). To reset the UART, write 0x00 to the Channel Software Reset register ‘CSR’.

Transmitter and receiver FIFOs

Both the transmitter and receiver have associated holding registers (FIFOs), referred to as the transmitter holding register (THR) and receiver holding register (RHR) respectively.

In normal operation, when the transmitter finishes transmitting a byte it removes the next data from the top of the THR and proceeds to transmit it. If the THR is empty, it waits until data is written into it. If THR is empty and the last character being transmitted has been completed (i.e. the transmitter shift register is empty) the transmitter is said to be idle. Similarly, when the receiver finishes receiving a byte, it transfers it to the bottom of the RHR. If the RHR is full, an overrun condition occurs.

Data is written into the bottom of the THR queue and read from the top of the RHR queue completely asynchronously to the operation of the transmitter and receiver.

The size of the FIFOs is dependent on the setting of the FCR register. When in Byte mode, these FIFOs only accept one byte at a time before indicating that they are full; this is compatible with the 16C450. When in a FIFO mode, the size of the FIFOs is either 16 (compatible with the 16C550) or 128.

Data written to the THR when it is full is lost. Data read from the RHR when it is empty is invalid. The empty or full status of the FIFOs is indicated in the Line Status Register ‘LSR’. Interrupts are generated when the UART is ready for data transfer to/from the FIFOs. The number of items in each FIFO may also be read back from the transmitter FIFO level (TFL) and receiver FIFO level (RFL) registers.

FIFO Control Register ‘FCR’

FIFO setup

- FCR[0]: Enable FIFO mode
  - logic 0 ⇒Byte mode
  - logic 1 ⇒ FIFO mode

  This bit should be enabled before setting the FIFO trigger levels.
- FCR[1]: Flush RHR
  - logic 0 ⇒ No change
  - logic 1 ⇒ Flushes the contents of the RHR

  This is only operative when already in a FIFO mode. The RHR is automatically flushed whenever changing between Byte mode and a FIFO mode. This bit returns to zero after clearing the FIFOs.
- FCR[2]: Flush THR
  - logic 0 ⇒ No change
  - logic 1 ⇒ Flushes the contents of the THR, in the same manner as FCR[1] does for the RHR

THR Trigger levels:

- FCR[3]: Tx trigger level enable
  - logic 0 ⇒ Transmit trigger levels enabled
  - logic 1 ⇒ Transmit trigger levels disabled

  When FCR[3]=0, the transmitter trigger level is always set to 1, thus ignoring FCR[5:4]. Alternatively, 950-mode trigger levels can be set using ACR[5]
- FCR[5:4]: Compatible trigger levels

450, 550 and extended 550 modes

The transmitter interrupt trigger levels are set to 1 and FCR[5:4] are ignored.

650 mode

The transmitter interrupt trigger levels can be set to the following values:

FCR[5:4]	Transmit Interrupt Trigger Level
00	16
01	32
10	64
11	112

These levels only apply when in Enhanced mode and when FCR[3] is set, otherwise the trigger level is set to 1. A transmitter empty interrupt is generated (if enabled) if the TFL falls below the trigger level.

750 Mode

Transmitter trigger level is set to 1, FCR[4] is unused and FCR[5] defines the FIFO depth as follows:

- FCR[5]=0: FIFO size is 16 bytes
- FCR[5]=1: FIFO size is 128 bytes

In non-Enhanced mode and when FIFOSEL pin is low, FCR[5] is writable only when LCR[7] is set. Note that in Enhanced mode, the FIFO size is increased to 128 bytes when FCR[0] is set.

950 mode

Setting ACR[5]=1 enables 950-mode trigger levels set using the TTL register, FCR[5:4] are ignored.

RHR trigger levels

FCR[7:6]: Compatible Trigger levels

450, 550, extended 550, 650 and 750 modes

The receiver FIFO trigger levels are defined using FCR[7:6]. The interrupt trigger level and upper flow control trigger level where appropriate are defined by L1 in the table below. L2 defines the lower flow control trigger level. Separate upper and lower flow control trigger levels introduce a hysteretic element in in-band and out-of-band flow control. In Byte mode (450 mode) the trigger levels are all set to 1.

950 mode

In similar fashion to for transmitter trigger levels, setting ACR[5]=1 enables 950-mode receiver trigger levels. FCR[7:6] are ignored.

FCR [7:6]	Mode					
	550 FIFO Size 16		Ext. 550 / 750 FIFO Size 128		650 FIFO Size 128	
	L1	L2	L1	L2	L1	L2
00	1	n/a	1	1	16	1
01	4	n/a	32	1	32	16
10	8	n/a	64	1	112	32
11	14	n/a	112	1	120	112

A receiver data interrupt is generated (if enabled) if the Receiver FIFO Level (‘RFL’) reaches the upper trigger level.

Line Control and Status

False Start Bit Detection

On the falling edge of a start bit, the receiver waits for 1/2 bit and re-synchronies the receiver's sampling clock onto the centre of the start bit. The start bit is valid if the SIN line is still low at this mid-bit sample and the receiver proceeds to read in a data character. Verifying the start bit prevents noise generating spurious character generation. Once the first stop bit has been sampled, the received data is transferred to the RHR and the receiver waits for a low transition on SIN (signifying the next start bit).

The receiver continues receiving data even if the RHR is full or the receiver has been disabled in order to maintain framing synchronization. The only difference is that the received data is not transferred to the RHR.

Line Control Register 'LCR'

The LCR specifies the data format that is common to both transmitter and receiver. Writing 0xBF to LCR enables access to the EFR, XON1, XOFF1, XON2 and XOFF2, DLL and DLM registers. This value (0xBF) corresponds to an unused data format. Writing the value 0xBF to LCR sets LCR[7] but leaves LCR[6:0] unchanged. Therefore, the data format of the transmitter and receiver data is not affected. Write the desired LCR value to exit from this selection.

LCR[1:0]: Data length

LCR[1:0] Determines the data length of serial characters. Note that these values are ignored in 9-bit data framing mode, i.e. when NMR[0] is set.

LCR[1:0]	Data Length
00	5 bits
01	6 bits
10	7 bits
11	8 bits

LCR[2]: Number of stop bits

LCR[2] defines the number of stop bits per serial character.

LCR[2]	Data Length	No. Stop Bits
0	5,6,7,8	1
1	5	1.5
1	6,7,8	2

LCR[5:3]: Parity type

The selected parity type is generated during transmission and checked by the receiver, which may produce a parity error as a result. In 9-bit mode, parity is disabled and LCR[5:3] is ignored.

LCR[5:3]	Parity type
xx0	No parity bit
001	Odd parity bit
011	Even parity bit
101	Parity bit forced to 1
111	Parity bit forced to 0

LCR[6]: Transmission break

- logic 0 ⇒ Break transmission disabled
- logic 1 ⇒ Forces the transmitter data output SOUT low to alert the communication terminal, or send zeros in IrDA mode

It is the responsibility of the software driver to ensure that the break duration is longer than the character period for it to be recognized remotely as a break rather than data.

LCR[7]: Divisor latch enable

- logic 0 ⇒ Access to DLL and DLM registers disabled
- logic 1 ⇒ Access to DLL and DLM registers enabled

Line Status Register ‘LSR’

This register provides the status of data transfer to CPU.

LSR[0]:RHR data available

- logic 0 ⇒ RHR is empty: no data available
- logic 1 ⇒ RHR is not empty: data is available to be read

LSR[1]: RHR overrun error

- logic 0 ⇒ No overrun error
- logic 1 ⇒ Data was received when the RHR was full. An overrun error has occurred. The error is flagged when the data would normally have been transferred to the RHR

LSR[2]: Received data parity error

- logic 0 ⇒ No parity error in normal mode or ninth bit of received data is ‘0’ in 9-bit mode
- logic 1 ⇒ Data has been received that did not have correct parity in normal mode or ninth bit of received data is ‘1’ in 9-bit mode

The parity error flag is set when the data item in error is at the top of the RHR and cleared following a read of the LSR. In 9-bit mode, LSR[2] is no longer a flag and corresponds to the ninth bit of the received data in RHR.

LSR[3]: Received data framing error

- logic 0 ⇒ No framing error
- logic 1 ⇒ Data has been received with an invalid stop bit

This status bit is set and cleared in the same manner as LSR[2]. When a framing error occurs, the UART tries to re-synchronies by assuming that the error is due to sampling the start bit of the next data item.

LSR[4]: Received break error

- logic 0 ⇒ No receiver break error
- logic 1 ⇒ The receiver received a break

A break condition occurs when the SIN line goes low (normally signifying a start bit) and stays low throughout the start, data, parity and first stop bit. (Note that the SIN line is sampled at the bit rate). One zero character with associated break flag set is transferred to the RHR and the receiver waits until the SIN line returns high. The LSR[4] break flag is set when this data item reaches the top of the RHR and it is cleared following a read of the LSR.

LSR[5]: THR empty

- logic 0 ⇒ Transmitter FIFO (THR) is not empty
- logic 1 ⇒ Transmitter FIFO (THR) is empty

LSR[6]: Transmitter and THR empty

- logic 0 ⇒ The transmitter is not idle
- logic 1 ⇒ THR is empty and the transmitter has completed the character in shift register and is in idle mode. (i.e. set whenever the transmitter shift register and the THR are both empty)

LSR[7]: Receiver data error

- logic 0 ⇒ Either there are no receiver data errors in the FIFO or it was cleared by a read of LSR
- logic 1 ⇒ At least one parity error, framing error or break indication in the FIFO

In 450 mode LSR[7] is permanently cleared, otherwise this bit is set when an erroneous character is transferred from the receiver to the RHR. It is cleared when the LSR is read. **Note that in 16C550 this bit is only cleared when all of the erroneous data are removed from the FIFO**. In 9-bit data framing mode parity is permanently disabled, so this bit is not affected by LSR[2].

Interrupts and Sleep Mode

The serial channel interrupts are asserted on the PCI INTA# pin. The interrupts can be enabled or disabled using the GIS register interrupt mask and the IER register. Unlike generic 16C550 devices, the interrupt cannot be disabled using the implementation-specific MCR[3].

Interrupt Enable Register 'IER'

Serial channel interrupts are enabled using the Interrupt Enable Register ('IER').

IER[0]: Receiver data available interrupt mask

- logic 0 ⇒ Disable the receiver ready interrupt
- logic 1 ⇒ Enable the receiver ready interrupt

IER[1]: Transmitter empty interrupt mask

- logic 0 ⇒ Disable the transmitter empty interrupt
- logic 1 ⇒ Enable the transmitter empty interrupt

IER[2]: Receiver status interrupt

Normal mode

- logic 0 ⇒ Disable the receiver status interrupt
- logic 1 ⇒ Enable the receiver status interrupt

9-bit data mode

- logic 0 ⇒ Disable receiver status and address bit interrupt
- logic 1 ⇒ Enable receiver status and address bit interrupt

In 9-bit mode (i.e. when NMR[0] is set), reception of a character with the address-bit (i.e. ninth bit) set can generate a level 1 interrupt if IER[2] is set.

IER[3]: Modem status interrupt mask

- logic 0 ⇒ Disable the modem status interrupt
- logic 1 ⇒ Enable the modem status interrupt

IER[4]: Sleep mode

- logic 0 ⇒ Disable sleep mode
- logic 1 ⇒ Enable sleep mode whereby the internal clock of the channel is switched off

IER[5]: Special character interrupt mask or alternate sleep mode

9-bit data framing mode

- logic 0 ⇒ Disable the received special character interrupt
- logic 1 ⇒ Enable the received special character interrupt

In 9-bit data mode, The receiver can detect up to four special characters programmed in the Special Character Registers . When IER[5] is set, a level 5 interrupt is asserted when the receiver character matches one of the values programmed.

650/950 modes (non-9-bit data framing)

- logic 0 ⇒ Disable the special character receive interrupt
- logic 1 ⇒ Enable the special character receive interrupt

In 16C650 compatible mode when the device is in Enhanced mode (EFR[4]=1), this bit enables the detection of special characters. It enables both the detection of XOFF characters (when in-band flow control is enabled using EFR[3:0]) and the detection of the XOFF2 special character (when enabled using EFR[5]).

750 mode (non-9-bit data framing)

- logic 0 ⇒ Disable alternate sleep mode
- logic 1 ⇒ Enable alternate sleep mode whereby the internal clock of the channel is switched off

In 16C750 compatible mode (i.e. non-Enhanced mode), this bit is used an alternate sleep mode and has the same effect as IER[4].

IER[6]: RTS interrupt mask

- logic 0 ⇒ Disable the RTS interrupt
- logic 1 ⇒ Enable the RTS interrupt

This enable is only operative in Enhanced mode (EFR[4]=1). In non-Enhanced mode, RTS interrupt is permanently enabled.

IER[7]: CTS interrupt mask

- logic 0 ⇒ Disable the CTS interrupt
- logic 1 ⇒ Enable the CTS interrupt

This enable is only operative in Enhanced mode (EFR[4]=1). In non-Enhanced mode, CTS interrupt is permanently enabled.

Interrupt Status Register ‘ISR’

The source of the highest priority interrupt pending is indicated by the contents of the Interrupt Status Register ‘ISR’. There are nine sources of interrupt at six levels of priority (1 is the highest) as shown in the following table.

Level	Interrupt Source	ISR[5:0] see note 3
-	No interrupt pending [1]	000001
1	Receiver status error or Address-bit detected in 9-bit mode	000110
2a	Receiver data available	000100
2b	Receiver time-out	001100
3	Transmitter THR empty	000010
4	Modem status change	000000
5 [2]	In-band flow control XOFF or Special character (XOFF2) or Special character 1, 2, 3 or 4 or bit 9 set in 9-bit mode	010000
6 [2]	CTS or RTS change of state	100000

Note 1: SR[0] indicates whether any interrupts are pending.

Note 2: Interrupts of priority levels 5 and 6 cannot occur unless the UART is in Enhanced mode.

Note 3: SR[5] is only used in 650 and 950 modes. In 750 mode, it is ‘0’ when FIFO size is 16 and ‘1’ when FIFO size is 128. In all other modes it is permanently set to 0.

ISR[6] and ISR[7] indicate whether the FIFOs are enabled. When you enable FIFOs both bits are set when either 16 or 128, note they also mirror fcr[0].

Interrupt Description

Level 1—Receiver status error interrupt (ISR[5:0]=’000110’)

Normal (non-9-bit) mode: this interrupt is active whenever any of LSR[1], LSR[2], LSR[3] or LSR[4] are set. These flags are cleared following a read of the LSR. This interrupt is masked with IER[2].

9-bit mode: this interrupt is active whenever any of LSR[1], LSR[2], LSR[3] or LSR[4] are set. The receiver error interrupt due to LSR[1], LSR[3] and LSR[4] is masked with IER[3]. The ‘address-bit’ received interrupt is masked with NMR[1]. The software driver can differentiate between receiver status error and received address-bit (ninth data bit) interrupt by examining LSR[1] and LSR[7]. In 9-bit mode LSR[7] is only set when LSR[3] or LSR[4] is set and it is not affected by LSR[2] (i.e. ninth data bit).

Level 2a—Receiver data available interrupt (ISR[5:0]=’000100’)

This interrupt is active whenever the receiver FIFO level is above the interrupt trigger level.

Level 2b—Receiver time-out interrupt (ISR[5:0]='001100')

A receiver time-out event, which may cause an interrupt, occurs when all of the following conditions are true:

- The UART is in a FIFO mode
- There is data in the RHR
- There has been no read of the RHR for a period of time greater than the time-out period
- There has been no new data written into the RHR for a period of time greater than the time-out period. The time-out period is four times the character period (including start and stop bits) measured from the centre of the first stop bit of the last data item received

Reading the first data item in RHR clears this interrupt.

Level 3—Transmitter empty interrupt (ISR[5:0]='000010')

This interrupt is set when the transmit FIFO level falls below the trigger level. It is cleared on an ISR read of a level 3 interrupt or by writing more data to the THR so that the trigger level is exceeded. Note that when 16C950 mode trigger levels are enabled (ACR[5]=1) and the transmitter trigger level of zero is selected (TTL=0x00), a transmitter empty interrupt is only asserted when both the transmitter FIFO and transmitter shift register are empty and the SOUT line has returned to idle marking state.

Level 4—Modem change interrupt (ISR[5:0]='000000')

This interrupt is set by a modem change flag (MSR[0], MSR[1], MSR[2] or MSR[3]) becoming active due to changes in the input modem lines. This interrupt is cleared following a read of the MSR.

Level 5
Receiver in-band flow control (XOFF) detect interrupt
Receiver special character (XOFF2) detect interrupt
Receiver special character 1, 2, 3 or 4 interrupt
Ninth Bit set interrupt in 9-bit mode (ISR[5:0]='010000')

A level 5 interrupt can only occur in Enhanced-mode when any of the following conditions are met:

- A valid XOFF character is received while in-band flow control is enabled.
- A received character matches XOFF2 while special character detection is enabled, i.e. EFR[5]=1.
- A received character matches special character 1, 2, 3 or 4 in 9-bit mode.

It is cleared on an ISR read of a level 5 interrupt.

Level 6—CTS or RTS changed interrupt (ISR[5:0]='100000')

This interrupt is set whenever any of the CTS# or RTS# pins changes state from low to high. It is cleared on an ISR read of a level 6 interrupt.

Sleep Mode

For a channel to go into sleep mode, all of the following conditions must be met:

1 Sleep mode enabled (IER[4]=1 in 650/950 modes, or IER[5]=1 in 750 mode).

2 The transmitter is idle, i.e. the transmitter shift register and FIFO are both empty.

3 SIN is high.

4 The receiver is idle.

5 The receiver FIFO is empty (LSR[0]=0).

6 The UART is not in loopback mode (MCR[4]=0).

7 Changes on modem input lines have been acknowledged (i.e. MSR[3:0]=0000).

8 No interrupts are pending.

A read of IER[4] (or IER[5] if a 1 was written to that bit instead) shows whether the power-down request was successful. The UART retains its programmed state whilst in power-down mode.

The channel automatically exits power-down mode when any of the conditions 1 to 7 becomes false. It may be woken manually by clearing IER[4] (or IER[5] if the alternate sleep mode is enabled).

Sleep mode operation is not available in IrDA mode.

Modem Interface

Modem Control Register 'MCR'

MCR[0]: DTR

- logic 0 ⇒ Force DTR# output to inactive (high)
- logic 1 ⇒ Force DTR# output to active (low)

Note that DTR# can be used for automatic out-of-band flow control when enabled using ACR[4:3].

MCR[1]: RTS

- logic 0 ⇒ Force RTS# output to inactive (high)
- logic 1 ⇒ Force RTS# output to active (low)

Note that RTS# can be used for automatic out-of-band flow control when enabled using EFR[6].

MCR[2]: OUT1

Note that OUT1# is not bonded out on the OXuPCI952.

- logic 0 ⇒ Force OUT1# output low when loopback mode is disabled
- logic 1 ⇒ Force OUT1# output high

MCR[3]: OUT2/Internal interrupt enable

Note that OUT2# is not bonded out on the OXuPCI952.

- logic 0 ⇒ Force OUT2# output low when loopback mode is disabled
- logic 1 ⇒ Force OUT2# output high

MCR[4]: Loopback mode

- logic 0 ⇒ Normal operating mode
- logic 1 ⇒ Enable local loop-back mode (diagnostics)

In local loop-back mode, the transmitter output (SOUT) and the four modem outputs (DTR#, RTS#, OUT1# and OUT2#) are set inactive (high), and the receiver inputs SIN, CTS#, DSR#, DCD#, and RI# are all disabled. Internally the transmitter output is connected to the receiver input and DTR#, RTS#, OUT1# and OUT2# are connected to modem status inputs DSR#, CTS#, RI# and DCD# respectively.

In this mode, the receiver and transmitter interrupts are fully operational. The modem control interrupts are also operational, but the interrupt sources are now the lower four bits of the Modem Control Register instead of the four modem status inputs. The interrupts are still controlled by the IER.

MCR[5]: Enable XON-Any in Enhanced mode or enable out-of-band flow control in non-Enhanced mode

650/950 (Enhanced) modes:

- logic 0 ⇒ XON-Any is disabled
- logic 1 ⇒ XON-Any is enabled

In Enhanced mode (EFR[4]=1), this bit enables the Xon-Any operation. When Xon-Any is enabled, any received data are accepted as a valid XON.

750 (normal) mode:

- logic 0 ⇒ CTS/RTS flow control disabled
- logic 1 ⇒ CTS/RTS flow control enabled

In non-Enhanced mode, this bit enables the CTS/RTS out-of-band flow control.

MCR[6]: IrDA mode

- logic 0 ⇒ Standard serial receiver and transmitter data format
- logic 1 ⇒ Data will be transmitted and received in IrDA format

This function is only available in Enhanced mode. It requires a 16x clock to function correctly.

MCR[7]: Baud rate prescaler select

- logic 0 ⇒ Normal (divide by 1) baud rate generator prescaler selected
- logic 1 ⇒ Divide-by-“M+N/8” baud rate generator prescaler selected

where M and N are programmed in CPR (ICR offset 0x01). After a hardware reset, CPR defaults to 0x20 (divide-by-4) and MCR[7] is reset. User writes to this flag only take effect in Enhanced mode.

Modem Status Register ‘MSR’

MSR[0]: Delta CTS#

Indicates that the CTS# input has changed since the last time the MSR was read.

MSR[1]: Delta DSR#

Indicates that the DSR# input has changed since the last time the MSR was read.

MSR[2]: Trailing edge RI#

Indicates that the RI# input has changed from low to high since the last time the MSR was read.

MSR[3]: Delta DCD#

Indicates that the DCD# input has changed since the last time the MSR was read.

MSR[4]: CTS

This bit is the complement of the CTS# input. It is equivalent to RTS (MCR[1]) in internal loop-back mode.

MSR[5]: DSR

This bit is the complement of the DSR# input. It is equivalent to DTR (MCR[0]) in internal loop-back mode.

MSR[6]: RI

This bit is the complement of the RI# input. In internal loop-back mode it is equivalent to the internal OUT1.

MSR[7]: DCD

This bit is the complement of the DCD# input. In internal loop-back mode it is equivalent to the internal OUT2.

Other Standard Registers

Divisor Latch Registers 'DLL and DLM'

The divisor latch registers are used to program the baud rate divisor. This is a value between 1 and 65535 by which the input clock is divided by in order to generate serial baud rates. After a hardware reset, the baud rate used by the transmitter and receiver is given by:

$$Baudrate = \frac{InputClock}{16 * Divisor}$$

Where divisor is given by DLL + (256 x DLM). More flexible baud rate generation options are also available.

Scratch Pad Register 'SPR'

The scratch pad register does not affect operation of the rest of the UART in any way and can be used for temporary data storage. The register may also be used to define an offset value to access the registers in the Indexed Control Register set.

Automatic Flow Control

Automatic in-band flow control, automatic out-of-band flow control and special character detection features can be used when in Enhanced mode (flow control is software compatible with the 16C654). Alternatively, 750-compatible automatic out-of-band flow control can be enabled when in non-Enhanced mode. In 950 mode, in-band and out-of-band flow controls are compatible with 16C654 with the addition of fully programmable flow control thresholds.

Enhanced Features Register 'EFR'

Writing 0xBF to LCR enables access to the EFR and other Enhanced mode registers. This value corresponds to an unused data format. Writing 0xBF to LCR sets LCR[7] but leaves LCR[6:0] unchanged. Therefore, the data format of the transmitter and receiver data is not affected. Write the desired LCR value to exit from this selection.

Note that In-band transmit and receive flow control is disabled in 9-bit mode.

EFR[1:0]: In-band receive flow control mode

When in-band receive flow control is enabled, the UART compares the received data with the programmed XOFF character(s). When this occurs, the UART disables transmission as soon as any current character transmission is complete. The UART then compares the received data with the programmed XON character(s). When a match occurs, the UART re-enables transmission.

For automatic in-band flow control, bit 4 of EFR must be set. The combinations of software receive flow control can be selected by programming EFR[1:0] as follows:

- logic [00] ⇒ In-band receive flow control is disabled
- logic [01] ⇒ Single character in-band receive flow control enabled, recognizing XON2 as the XON character and XOFF2 as the XOFF character
- logic [10] ⇒ Single character in-band receive flow control enabled, recognizing XON1 as the XON character and XOFF1 and the XOFF character
- logic [11] ⇒ The behavior of the receive flow control is dependent on the configuration of EFR[3:2]. Single character in-band receive flow control is enabled, accepting XON1 or XON2 as valid XON characters and XOFF1 or XOFF2 as valid XOFF characters when EFR[3:2] = “01” or “10”. EFR[1:0] should not be set to “11” when EFR[3:2] is ‘00’

EFR[3:2]: In-band transmit flow control mode

When in-band transmit flow control is enabled, XON/XOFF character(s) are inserted into the data stream whenever the RFL passes the upper trigger level and falls below the lower trigger level respectively.

For automatic in-band flow control, bit 4 of EFR must be set. The combinations of software transmit flow control can then be selected by programming EFR[3:2] as follows:

- logic [00] ⇒ In-band transmit flow control is disabled
- logic [01] ⇒ Single character in-band transmit flow control enabled, using XON2 as the XON character and XOFF2 as the XOFF character
- logic [10] ⇒ Single character in-band transmit flow control enabled, using XON1 as the XON character and XOFF1 as the XOFF character
- logic[11] ⇒ The value EFR[3:2] = “11” is reserved for future use and should not be used

EFR[4]: Enhanced mode

- logic 0 ⇒ Non-Enhanced mode. Disables IER bits 4-7, ISR bits 4-5, FCR bits 4-5, MCR bits 5-7 and in-band flow control. Whenever this bit is cleared, the settings of other bits of EFR are ignored
- logic 1 ⇒ Enhanced mode. Enables the enhanced mode functions. These functions include enabling IER bits 4-7, FCR bits 4-5, MCR bits 5-7. For in-band flow control the software driver must set this bit first. If this bit is set, out-of-band flow control is configured with EFR bits 6-7; otherwise out-of-band flow control is compatible with 16C750

EFR[5]: Enable special character detection

- logic 0 ⇒ Special character detection is disabled
- logic 1 ⇒ While in Enhanced mode (EFR[4]=1), the UART compares the incoming receiver data with the XOFF2 value. Upon a correct match, the received data are transferred to the RHR and a level 5 interrupt (XOFF or special character) is asserted if level 5 interrupts are enabled (IER[5] set to 1)

EFR[6]:Enable automatic RTS flow control.

- logic 0 ⇒ RTS flow control is disabled (default)
- logic 1 ⇒ RTS flow control is enabled in Enhanced mode (i.e. EFR[4] = 1), where the RTS# pin is forced inactive high if the RFL reaches the upper flow control threshold. This is released when the RFL drops below the lower threshold. 650 and 950-mode drivers should use this bit to enable RTS flow control

EFR[7]: Enable automatic CTS flow control.

- logic 0 ⇒ CTS flow control is disabled (default)
- logic 1 ⇒ CTS flow control is enabled in Enhanced mode (i.e. EFR[4] = 1), where the data transmission is prevented whenever the CTS# pin is held inactive high. 650 and 950-mode drivers should use this bit to enable CTS flow control

A 750-mode driver should set MCR[5] to enable RTS/CTS flow control.

Special Character Detection

In Enhanced mode (EFR[4]=1), when special character detection is enabled (EFR[5]=1) and the receiver matches received data with XOFF2, the 'received special character' flag ASR[4] is set and a level 5 interrupt is asserted, if enabled by IER[5]. This flag is cleared following a read of ASR. The received status (i.e. parity and framing) of special characters does not have to be valid for these characters to be accepted as valid matches.

Automatic In-band Flow Control

When in-band receive flow control is enabled, the UART compares the received data with XOFF1 or XOFF2 characters to detect an XOFF condition. When this occurs, the UART disables transmission as soon as any current character transmission is complete. Status bits ISR[4] and ASR[0] are set. A level 5 interrupt occurs (if enabled by IER[5]). The UART compares all received data with XON1 or XON2 characters to detect an XON condition. When this occurs, the UART re-enables transmission and status bits ISR[4] and ASR[0] are cleared.

Any valid XON/XOFF characters are not written into the RHR. An exception to this rule occurs if special character detection is enabled and an XOFF2 character is received that is a valid XOFF. In this instance, the character is written into the RHR.

The received status (i.e. parity and framing) of XON/XOFF characters does not have to be valid for these characters to be accepted as valid matches.

When the 'XON Any' flag (MCR[5]) is set, any received character is accepted as a valid XON condition and the transmitter is re-enabled. The received data is transferred to the RHR.

When in-band transmit flow control is enabled, the RFL is sampled whenever the transmitter is idle (briefly, between characters, or when the THR is empty) and an XON/XOFF character is inserted into the data stream if needed. Initially, remote transmissions are enabled and hence ASR[1] is clear. If ASR[1] is clear

and the RFL has passed the upper trigger level (i.e. is above the trigger level), XOFF is sent and ASR[1] is set. If ASR[1] is set and the RFL falls below the lower trigger level, XON is sent and ASR[1] is cleared.

If transmit flow control is disabled after an XOFF has been sent, an XON is sent automatically.

Automatic Out-of-band Flow Control

Automatic RTS/CTS flow control is selected by different means, depending on whether the UART is in Enhanced or non-Enhanced mode. When in non-Enhanced mode, MCR[5] enables both RTS and CTS flow control. When in Enhanced mode, EFR[6] enables automatic RTS flow control and EFR[7] enables automatic CTS flow control. This allows software compatibility with both 16C650 and 16C750 drivers.

When automatic CTS flow control is enabled and the CTS# input becomes active, the UART disables transmission as soon as any current character transmission is complete. Transmission is resumed whenever the CTS# input becomes inactive.

When automatic RTS flow control is enabled, the RTS# pin is forced inactive when the RFL reaches the upper trigger level and returns to active when the RFL falls below the lower trigger level. The automatic RTS# flow control is ANDed with MCR[1] and hence is only operational when MCR[1]=1. This allows the software driver to override the automatic flow control and disable the remote transmitter regardless by setting MCR[1]=0 at any time.

Automatic DTR/DSR flow control behaves in the same manner as RTS/CTS flow control but is enabled by ACR[3:2], regardless of whether the UART is in Enhanced mode.

Baud Rate Generation

The UART contains a programmable baud rate generator that is capable of taking any clock input from 1.8432 MHz to 60 MHz and dividing it by any 16-bit divisor number from 1 to 65535 written into the DLM (MSB) and DLL (LSB) registers. In addition to this, a clock prescaler register is provided which can further divide the clock by values in the range 1.0 to 31.875 in steps of 0.125. A further feature is the Times Clock Register ‘TCR’ which allows the sampling clock to be set to any value between 4 and 16.

These clock options allow for highly flexible baud rate generation capabilities from almost any input clock frequency (up to 60 MHz).

The actual transmitter and receiver baud rate is calculated as follows:

$$BaudRate = \frac{InputClock}{SC * Divisor * prescaler}$$

Where:

- SC = Sample clock values defined in TCR[3:0]
- Divisor = DLL + (256 x DLM)
- Prescaler = 1 when MCR[7] = '0'; else = M + (N / 8) where:
  - M = CPR[7:3] (Integer part – 1 to 31)
  - N = CPR[2:0] (Fractional part – 0.000 to 0.875)

After a hardware reset, the prescaler is bypassed (set to 1) and TCR is set to 0x00 (i.e. SC = 16). Assuming this default configuration, the following table gives the divisors required to be programmed into the DLL and DLM registers in order to obtain various standard baud rates:

DLM:DLL Divisor Word	Baud Rate (bits per second)
0x0900	50
0x0300	110
0x0180	300
0x00C0	600
0x0060	1,200
0x0030	2,400
0x0018	4,800
0x000C	9,600
0x0006	19,200
0x0004	28,800
0x0003	38,400
0x0002	57,600
0x0001	115,200

Clock Prescaler Register 'CPR'

The CPR register is located at offset 0x01 of the ICR.

The prescaler divides the system clock by any value in the range of 1 to "31 7/8" in steps of 1/8. The divisor takes the form "M+N/8", where M is the 5 bit value defined in CPR[7:3] and N is the 3 bit value defined in CPR[2:0].

The prescaler is by-passed and a prescaler value of '1' is selected by default when MCR[7] = 0.

Note that since access to MCR[7] is restricted to Enhanced mode only, EFR[4] should first be set and then MCR[7] set or cleared as required.

For higher baud rates use a higher frequency clock, e.g. 14.7456 MHz, 18.432 MHz, 32 MHz, 40 MHz or 60.0 MHz. The flexible prescaler allows you to use clocks that are not integer multiples of popular baud rates. When using a non-standard clock frequency, compatibility with existing 16C550 software drivers may be maintained with a minor software patch to program the on-board prescaler to divide the high frequency clock down to 1.8432 MHz.

Tables in Times Clock Register ‘TCR’ give the prescaler values required to operate the UARTs at compatible baud rates with various different crystal frequencies. Also given is the maximum available baud rates in TCR = 16 and TCR = 4 modes with CPR = 1.

Times Clock Register ‘TCR’

The TCR register is located at offset 0x02 of the ICR.

The 16C550 and other compatible devices such as 16C650 and 16C750 use a 16 times (16x) over-sampling channel clock. The 16x over-sampling clock means that the channel clock runs at 16 times the selected serial bit rate. It limits the highest baud rate to 1/16 of the system clock when using a divisor latch value of unity. However, the 16C950 UART enables it to accept other multiplications of the bit rate clock. It can use values from 4x to 16x clock as programmed in the TCR as long as the clock (oscillator) frequency error, stability and jitter are within reasonable parameters. Upon hardware reset the TCR is reset to 0x00, which means that a 16x clock is used, for compatibility with the 16C550 and compatibles.

The maximum baud-rates available for various system clock frequencies at all of the allowable values of TCR are indicated in the following table. These are the values in bits-per-second (bps) that are obtained if the divisor latch = 0x01 and the prescaler is set to 1.

The OXuPCI952 has the facility to operate at baud-rates up to 15 Mbps in normal mode.

The following table indicates how the value in the register corresponds to the number of clock cycles per bit. TCR[3:0] is used to program the clock. TCR[7:4] are unused and returns “0000” if read.

TCR[3:0]	Clock Cycles per Bit
0000 to 0011	16
0100 to 1111	4-15

Use of the TCR does not require the device to be in 650 or 950 mode although only drivers that have been written to take advantage of the 950 mode features are able to access this register. Writing 0x01 to the TCR does not switch the device into 1x isochronous mode. This is explained in the following section. (TCR has no effect in isochronous mode). If 0x01, 0x10 or 0x11 is written to TCR the device operates in 16x mode.

Reading TCR always returns the last value that was written to it irrespective of mode of operation.

The following table gives example clock options and their associated maximum baud rates.

Clock Frequency (MHz)	CPR value	Effective Crystal Frequency	Error from 1.8432 MHz (%)	Max. Baud rate with CPR = 1, TCR = 16	Max. Baud Rate with CPR = 1, TCR = 4
1.8432	0x08 (1)	1.8432	0.00	115,200	460,800
7.3728	0x20 (4)	1.8432	0.00	460,800	1,843,200
14.7456	0x40 (8)	1.8432	0.00	921,600	3,686,400
18.432	0x50 (10)	1.8432	0.00	1,152,000	4,608,000
32.000	0x8B (17.375)	1.8417	0.08	2,000,000	8,000,000
33.000	0x8F (17.875)	1.8462	0.16	2,062,500	8,250,000
40.000	0xAE (21.75)	1.8391	0.22	2,500,000	10,000,000
50.000	0xD9 (27.125)	1.8433	0.01	3,125,000	12,500,000
60.000	0xFF (31.875)	1.8824	2.13	3,750,000	15,000,000

The following table gives maximum baud rates available at all ‘tcr’ sampling clock values.

Sampling Clock	TCR Value	System Clock (MHz)							
		1.8432	7.372	14.7456	18.432	32	40	50	60
16	0x00	115,200	460,750	921,600	1.152M	2.00M	2.50M	3.125M	3.75M
15	0x0F	122,880	491,467	983,040	1,228,800	2,133,333	2,666,667	3,333,333	4.00M
14	0x0E	131,657	526,571	1,053,257	1,316,571	2,285,714	2,857,143	3,571,429	4,285,714
13	0x0D	141,785	567,077	1,134,277	1,417,846	2,461,538	3,076,923	3,846,154	4,615,384
12	0x0C	153,600	614,333	1,228,800	1,536,000	2,666,667	3,333,333	4,166,667	5.00M
11	0x0B	167,564	670,182	1,340,509	1,675,636	2,909,091	3,636,364	4,545,455	5,454545
10	0x0A	184,320	737,200	1,474,560	1,843,200	3.20M	4.00M	5.00M	6.00M
9	0x09	204,800	819,111	1,638,400	2,048,000	3,555,556	4,444,444	5,555,556	6,666,667
8	0x08	230,400	921,500	1,843,200	2,304,000	4.00M	5.00M	6.25M	7.50M
7	0x07	263,314	1,053,143	2,106,514	2,633,143	4,571,429	5,714,286	7,142,857	8,571428
6	0x06	307,200	1,228,667	2,457,600	3,072,000	5,333,333	6,666,667	8,333,333	10.00M
5	0x05	368,640	1,474,400	2,949,120	3,686,400	6.40M	8.00M	10.00M	12.00M
4	0x04	460,800	1,843,000	3,686,400	4,608,000	8.00M	10.00M	12.50M	15.00M

External 1x Clock Mode

The transmitter and receiver can accept an external clock applied to the RI# and DSR# pins respectively. The clock options are selected using the CKS register. The transmitter and receiver may be configured to operate in 1x (i.e. isochronous mode) by setting CKS[7] and CKS[3] respectively. In isochronous mode, transmitter or receiver uses the 1x clock (usually, but not necessarily, an external source) where asynchronous framing is maintained using start-, parity- and stop-bits. However, serial transmission and reception are synchronized to the 1x clock. In this mode asynchronous data may be transmitted at baud rates up to 60 Mbps. The local 1x clock source can be asserted on the DTR# pin.

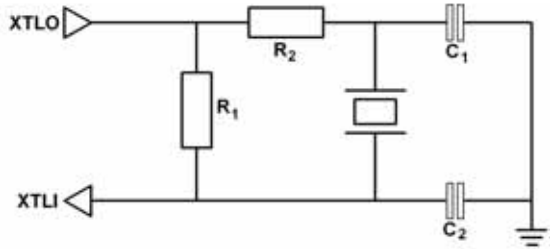
The OXuPCI952 supports crystal frequencies in two bands, 0–12 MHz and 12–20 MHz. The frequency for the device is selected by the XTLSEL pin (pin 130), which should be set to select the correct band for the chosen crystal.

Note that line drivers need to be capable of transmission at data rates twice the system clock used (as one cycle of the system clock corresponds to 1 bit of serial data). Also note that enabling modem interrupts is illegal in isochronous mode, as the clock signal causes a continuous change to the modem status (unless masked in MDM register).

Crystal Oscillator Circuit

The UART's reference clock may be provided by its own crystal oscillator or directly from a clock source connected to the XTLI pin. The circuit required to use the internal oscillator is shown in Figure 5.

Figure 5 Crystal Oscillator Circuit



The following table gives component values.

Frequency Range (MHz)	C_1 (pF)	C_2 (pF)	R_1 (Ω)	R_2 (Ω)
1.8432 – 20	22	22	1M	470R

Note that for better stability use a smaller value of R_1. Increase R_1 to reduce power consumption.

The total capacitive load (C1 in series with C2) should be that specified by the crystal manufacturer (nominally 18 pF).

Note that for frequency operation above 20 MHz, an external clock source is required.

Additional Features

Additional Status Register ‘ASR’

ASR[0]: Transmitter disabled

- logic 0 ⇒ The transmitter is not disabled by in-band flow control
- logic 1 ⇒ The receiver has detected an XOFF and has disabled the transmitter

This bit is cleared after a hardware reset or channel software reset. The software driver may write a 0 to this bit to re-enable the transmitter if it is disabled by in-band flow control. Writing a 1 to this bit has no effect.

ASR[1]: Remote transmitter disabled

- logic 0 ⇒ The remote transmitter is not disabled by in-band flow control
- logic 1 ⇒ The transmitter has sent an XOFF character, to disable the remote transmitter (cleared when subsequent XON is sent)

This bit is cleared after a hardware reset or channel software reset. The software driver may write a 0 to this bit to re-enable the remote transmitter (an XON is transmitted). Note that writing a 1 to this bit has no effect.

ASR[2]: RTS

This is the complement of the actual state of the RTS# pin when the device is not in loopback mode. The driver software can determine if the remote transmitter is disabled by RTS# out-of-band flow control by reading this bit. In loopback mode this bit reflects the flow control status rather than the pin’s actual state.

ASR[3]: DTR

This is the complement of the actual state of the DTR# pin when the device is not in loopback mode. The driver software can determine if the remote transmitter is disabled by DTR# out-of-band flow control by reading this bit. In loopback mode this bit reflects the flow control status rather than the pin’s actual state.

ASR[4]: Special character detected

- logic 0 ⇒ No special character has been detected
- logic 1 ⇒ A special character has been received and is stored in the RHR

This can be used to determine whether a level 5 interrupt was caused by receiving a special character rather than an XOFF. The flag is cleared following the read of the ASR.

ASR[5]: FIFOSEL

This bit reflects the unlatched state of the FIFOSEL pin.

ASR[6]: FIFO size

- logic 0 ⇒ FIFOs are 16 deep if FCR[0] = 1
- logic 1 ⇒ FIFOs are 128 deep if FCR[0] = 1

ASR[7]: Transmitter Idle

- logic 0 ⇒ Transmitter is transmitting
- logic 1 ⇒ Transmitter is idle

This bit reflects the state of the internal transmitter. It is set when both the transmitter FIFO and shift register are empty.

FIFO Fill levels ‘TFL and RFL’

The number of characters stored in the THR and RHR can be determined by reading the TFL and RFL registers respectively. When data transfer is in constant operation, the values should be interpreted as follows:

- The number of characters in the THR is no greater than the value read back from TFL
- The number of characters in the RHR is no less than the value read back from RFL

Additional Control Register ‘ACR’

The ACR register is located at offset 0x00 of the ICR.

ACR[0]: Receiver disable

- logic 0 ⇒ The receiver is enabled, receiving data and storing it in the RHR
- logic 1 ⇒ The receiver is disabled. The receiver continues to operate as normal to maintain the framing synchronization with the receive data stream but received data is not stored in the RHR. In-band flow control characters continue to be detected and acted upon. Special characters are not detected

Changes to this bit are only recognized following the completion of any data reception pending.

ACR[1]: Transmitter disable

- logic 0 ⇒ The transmitter is enabled, transmitting any data in the THR
- logic 1 ⇒ The transmitter is disabled. Any data in the THR is not transmitted but is held. However, in-band flow control characters may still be transmitted

Changes to this bit are only recognized following the completion of any data transmission pending.

ACR[2]: Enable automatic DSR flow control

- logic 0 ⇒ Normal. The state of the DSR# line does not affect the flow control
- logic 1 ⇒ Data transmission is prevented whenever the DSR# pin is held inactive high

This bit provides another automatic out-of-band flow control facility using the DSR# line.

ACR[4:3]: DTR# line configuration

When bits 4 or 5 of CKS (offset 0x03 of ICR) are set, the transmitter 1x clock or the output of the baud rate generator (Nx clock) are asserted on the DTR# pin, otherwise the DTR# pin is defined as follows:

- logic [00] ⇒ DTR# is compatible with 16C450, 16C550, 16C650 and 16C750 (i.e. normal)
- logic [01] ⇒ DTR# pin is used for out-of-band flow control. It is forced inactive high if the Receiver FIFO Level ('RFL') reaches the upper flow control threshold. DTR# line is re-activated (=0) when the RFL drops below the lower threshold (see FCL and FCH)
- logic [10] ⇒ DTR# pin is configured to drive the active-low enable pin of an external RS485 buffer. In this configuration the DTR# pin is forced low whenever the transmitter is not empty (LSR[6]=0), otherwise DTR# pin is high.
- logic [11] ⇒ DTR# pin is configured to drive the active-high enable pin of an external RS485 buffer. In this configuration, the DTR# pin is forced high whenever the transmitter is not empty (LSR[6]=0), otherwise DTR# pin is low.

If you set ACR[4], then the DTR# line is controlled by the status of the transmitter empty bit of LCR. When ACR[4] is set, ACR[3] is used to select active high or active low enable signals. In half-duplex systems using RS485 protocol, this facility enables the DTR# line to directly control the enable signal of external 3-state line driver buffers. When the transmitter is empty the DTR# goes inactive once the SOUT line returns to its idle marking state.

ACR[5]: 950 mode trigger levels enable

- logic 0 ⇒ Interrupts and flow control trigger levels are as described in FCR register and are compatible with 16C650/16C750 modes
- logic 1 ⇒ 16C950 specific enhanced interrupt and flow control trigger levels defined by RTL, TTL, FCL and FCH are enabled

ACR[6]: ICR read enable

- logic 0 ⇒ The Line Status Register is readable
- logic 1 ⇒ The Indexed Control Registers are readable

Setting this bit maps the ICR set to the LSR location for reads. During normal operation this bit should be cleared.

ACR[7]: Additional status enable

- logic 0 ⇒ Access to the ASR, TFL and RFL registers is disabled
- logic 1 ⇒ Access to the ASR, TFL and RFL registers is enabled

When ACR[7] is set, the MCR, LCR and IER registers are no longer readable but remain writable, and the registers ASR, TFL and RFL replace them in the register map for read operations. The software driver may leave this bit set during normal operation, since MCR, LCR and IER do not generally need to be read.

Transmitter Trigger Level 'TTL'

The TTL register is located at offset 0x04 of the ICR.

Whenever 950 trigger levels are enabled (ACR[5]=1), bits 4 and 5 of FCR are ignored and an alternative arbitrary transmitter interrupt trigger level can be defined in the TTL register. This 7-bit value provides a fully programmable transmitter interrupt trigger facility. In 950 mode, a priority level 3 interrupt occurs indicating that the transmitter buffer requires more characters when the interrupt is not masked (IER[1]=1) and the transmitter FIFO level falls below the value stored in the TTL register. The value 0 (0x00) has a special meaning. In 950 mode when the user writes 0x00 to the TTL register, a level 3 interrupt only occurs when the FIFO and the transmitter shift register are both empty and the SOUT line is in the idle marking state. This feature is particularly useful to report back the empty state of the transmitter after its FIFO has been flushed away.

Receiver Interrupt. Trigger Level 'RTL'

The RTL register is located at offset 0x05 of the ICR.

Whenever 950 trigger levels are enabled (ACR[5]=1), bits 6 and 7 of FCR are ignored and an alternative arbitrary receiver interrupt trigger level can be defined in the RTL register. This 7-bit value provides a fully programmable receiver interrupt trigger facility as opposed to the limited trigger levels available in 16C650 and 16C750 devices. It enables you to optimize the interrupt performance hence minimizing the interrupt overhead.

In 950 mode, a priority level 2 interrupt occurs indicating that the receiver data is available when the interrupt is not masked (IER[0]=1) and the receiver FIFO level reaches the value stored in this register.

Flow Control Levels 'FCL' and 'FCH'

The FCL and FCH registers are located at offsets 0x06 and 0x07 of the ICR respectively

Enhanced software flow control using XON/XOFF and hardware flow control using RTS#/CTS# and DTR#/DSR# are available when 950 mode trigger levels are enabled (ACR[5]=1). Improved flow control threshold levels are offered using Flow Control Lower trigger level ('FCL') and Flow Control Higher trigger level ('FCH') registers to provide a greater degree of flexibility when optimizing the flow control performance. Generally, these facilities are only available in Enhanced mode.

In 650 mode, in-band flow control is enabled using the EFR register. An XOFF character may be transmitted when the receiver FIFO exceeds the upper trigger level defined by FCR[7:6]. An XON is then sent when the FIFO is read down the lower fill level. The flow control is enabled and the appropriate mode is selected using EFR[3:0].

In 950 mode, the flow control thresholds defined by FCR[7:6] are ignored. In this mode, threshold levels are programmed using FCL and FCH. When flow control is enabled by EFR[3:0] and the receiver FIFO level ('RFL') reaches the value programmed in the FCH register, an XOFF is transmitted to stop the flow of serial data as defined by EFR[3:0]. When the receiver FIFO level falls below the value programmed in FCL the flow is resumed by sending an XON character (as defined in EFR[3:0]). The FCL value of 0x00 is illegal.

CTS/RTS and DSR/DTR out-of-band flow control use the same trigger levels as in-band flow control. When out-of-band flow control is enabled, RTS# (or DTR#) line is de-asserted when the receiver FIFO level reaches the upper limit defined in the FCH and is re-asserted when the receiver FIFO is drained below a lower limit defined in FCL. When 950 trigger levels are enabled (ACR[5]=1), the CTS# flow control functions as in 650 mode and is configured by EFR[7]. However, RTS# is automatically de-asserted and re-asserted when EFR[6] is set and RFL reaches FCH and drops below FCL. DSR# flow control is configured with ACR[2]. DTR# flow control is configured with ACR[4:3].

Device Identification Registers

The identification registers are located at offsets 0x08 to 0x0B of the ICR.

The UARTs offer four bytes of device identification. The device ID registers may be read using offset values 0x08 to 0x0B of the Indexed Control Register. Registers ID1, ID2 and ID3 identify the device as an OX16C950 and return 0x16, 0xC9 and 0x50 respectively. The REV register resides at offset 0x0B of ICR and identifies the revision of OX16C950. This register returns 0x12 for the OXuPCI952.

Clock Select Register 'CKS'

The CKS register is located at offset 0x03 of the ICR.

This register is cleared to 0x00 after a hardware reset to maintain compatibility with 16C550, but is unaffected by software reset. This allows the user to select a clock source and then reset the channel to work-around any timing glitches.

CKS[1:0]: Receiver Clock Source Selector

- logic [x0] ⇒ The output of baud rate generator (internal BDOUT#) is selected for the receiver clock
- logic [01] ⇒ The DSR# pin is selected for the receiver clock
- logic [11] ⇒ The transmitter clock is selected for the receiver. This allows RI# to be used for both transmitter and receiver

CKS[2]: Reserved

CKS[3]: Receiver 1x clock mode selector

- logic 0 ⇒ The receiver is in Nx clock mode as defined in the TCR register. After a hardware reset the receiver operates in 16x clock mode, i.e. 16C550 compatibility
- logic 1 ⇒ The receiver is in isochronous 1x clock mode

CKS[5:4]: Transmitter 1x clock or baud rate generator output (BDOUT) on DTR# pin

- logic [00] ⇒ The function of the DTR# pin is defined by the setting of ACR[4:3]
- logic [01] ⇒ The transmitter 1x clock (bit rate clock) is asserted on the DTR# pin and the setting of ACR[4:3] is ignored
- logic [10] ⇒ The output of baud rate generator (Nx clock) is asserted on the DTR# pin and the setting of ACR[4:3] is ignored
- logic [11] ⇒ Reserved

CKS[6]: Transmitter clock source selector

- logic 0 ⇒ The transmitter clock source is the output of the baud rate generator (550 compatibility)
- logic 1 ⇒ The transmitter uses an external clock applied to the RI# pin

CKS[7]: Transmitter 1x clock mode selector

- logic 0 ⇒ The transmitter is in Nx clock mode as defined in the TCR register. After a hardware reset the transmitter operates in 16x clock mode, i.e. 16C550 compatibility
- logic 1 ⇒ The transmitter is in isochronous 1x clock mode

Nine-bit Mode Register ‘NMR’

The NMR register is located at offset 0x0D of the ICR.

The UART offers 9-bit data framing for industrial multi-drop applications. The 9-bit mode is enabled by setting bit 0 of the Nine-bit Mode Register (NMR). In 9-bit mode, the data length setting in LCR[1:0] is ignored. Furthermore, as parity is permanently disabled, the setting of LCR[5:3] is also ignored.

The receiver stores the ninth bit of the received data in LSR[2] (where parity error is stored in normal mode). Note that the UART provides a 128-deep FIFO for LSR[3:0].

The transmitter FIFO is 9 bits wide and 128 deep. Write the ninth (MSB) data bit in SPR[0] first and then write the other 8 bits to THR.

As parity mode is disabled, LSR[7] is set whenever there is an overrun, framing error or received break condition. It is unaffected by the contents of LSR[2] (now the received ninth data bit).

In 9-bit mode, in-band flow control is disabled regardless of the setting of EFR[3:0] and the XON1/XON2/XOFF1 and XOFF2 registers are used for special character detection.

Interrupts in 9-Bit Mode: while IER[2] is set, upon receiving a character with status error, a level 1 interrupt is asserted when the character and the associated status are transferred to the FIFO.

The UART can assert an optional interrupt if a received character has its ninth bit set. As multi-drop systems often use the ninth bit as an address bit, the receiver is able to generate an interrupt upon receiving an address character. This feature is enabled by setting NMR[2]. This results in a level 1 interrupt being asserted when the address character is transferred to the receiver FIFO.

In this case, as long as there are no errors pending, i.e. LSR[1], LSR[3], and LSR[4] are clear, '0' can be read back from LSR[7] and LSR[1], thus differentiating between an ‘address’ interrupt and receiver error or overrun interrupt in 9-bit mode. Note however, that should an overrun or error interrupt actually occur, an address character may also reside in the FIFO. In this case, the software driver should examine the contents of the receiver FIFO as well as process the error.

The above facility produces an interrupt for recognizing any ‘address’ characters. Alternatively, you can configure the UART to compare the receiver data stream with up to four programmable 9-bit characters and assert a level 5 interrupt after detecting a match. The interrupt occurs when the character is transferred to the FIFO.

NMR[0]: 9-bit mode enable

- logic 0 ⇒ 9-bit mode is disabled
- logic 1 ⇒ 9-bit mode is enabled

NMR[1]: Enable interrupt when ninth bit is set

- logic 0 ⇒ Receiver interrupt for detection of an ‘address’ character (i.e. ninth bit set) is disabled
- logic 1 ⇒ Receiver interrupt for detection of an ‘address’ character (i.e. ninth bit set) is enabled and a level 1 interrupt is asserted

Special Character Detection

While the UART is in both 9-bit mode and Enhanced mode, setting IER[5] enables detection of up to four ‘address’ characters. The least significant eight bits of these four programmable characters are stored in special characters 1 to 4 (XON1, XON2, XOFF1 and XOFF2 in 650 mode) registers and the ninth bit of these characters are programmed in NMR[5] to NMR[2] respectively.

NMR[2]: Bit 9 of Special Character 1
NMR[3]: Bit 9 of Special Character 2
NMR[4]: Bit 9 of Special Character 3
NMR[5]: Bit 9 of Special Character 4

NMR[7:6]: Reserved

Bits 6 and 7 of NMR are always cleared and reserved for future use.

Modem Disable Mask ‘MDM’

The MDM register is located at offset 0x0E of the ICR.

This register is cleared after a hardware reset to maintain compatibility with 16C550. It allows you to mask interrupts, sleep operation and power management events due to individual modem lines or the serial input line.

MDM[0]: Disable delta CTS

- logic 0 ⇒ Delta CTS is enabled. It can generate a level 4 interrupt when enabled by IER[3]. In power state D2, delta CTS can assert the PME# line. Delta CTS can wake up the UART when it is asleep under auto-sleep operation
- logic 1 ⇒ Delta CTS is disabled. It cannot generate an interrupt, assert a PME# or wake up the UART

MDM[1]: Disable delta DSR

- logic 0 ⇒ Delta DSR is enabled. It can generate a level 4 interrupt when enabled by IER[3]. In power state D2, delta DSR can assert the PME# line. Delta DSR can wake up the UART when it is asleep under auto-sleep operation
- logic 1 ⇒ Delta DSR is disabled. It cannot generate an interrupt, assert a PME# or wake up the UART

MDM[2]: Disable Trailing edge RI

- logic 0 ⇒ Trailing edge RI is enabled. It can generate a level 4 interrupt when enabled by IER[3]. In power state D2, trailing edge RI can assert the PME# line. Trailing edge RI can wake up the UART when it is asleep under auto-sleep operation
- logic 1 ⇒ Trailing edge RI is disabled. It cannot generate an interrupt, assert a PME# or wake up the UART

MDM[3]: Disable delta DCD

- logic 0 ⇒ Delta DCD is enabled. It can generate a level 4 interrupt when enabled by IER[3]. In power state D2, delta DCD can assert the PME# line. Delta DCD can wake up the UART when it is asleep under auto-sleep operation
- logic 1 ⇒ Delta DCD is disabled. It cannot generate an interrupt, assert a PME# or wake up the UART

MDM[4]: Modem Wakeup Disable

- logic 1 ⇒ Prevents modem status interrupts from taking the UART out of sleep mode. For low power this may be useful
- logic 0 ⇒ Allows the modem status interrupts to take the UART out of sleep mode

MDM[5]: Disable SIN wake up

- logic 0 ⇒ When the device is in power-down state D2, a change in the state of the serial input line (i.e. start bit) can assert the PME# line
- logic 1 ⇒ When the device is in power-down state D2, a change in the state of the serial input line cannot assert the PME# line

MDM[7:6]: Reserved

Readable FCR 'RFC'

The RFC register is located at offset 0x0F of the ICR.

This read-only register returns the current state of the FCR register (note that FCR is write-only). This register is included for diagnostic purposes.

Good-data Status Register 'GDS'

The GDS register is located at offset 0x10 of the ICR.

GDS[0]: Good Data Status

GDS[7:1]: Reserved

Port Index Register 'PIX'

The PIX register is located at offset 0x12 of the ICR. This read-only register gives the UART index. For the OXuPCI952 this reads 0, 1, 2 or 3 depending on the UART being accessed.

Clock Alteration Register 'CKA'

The CKA register is located at offset 0x13 of the ICR. This register provides additional clock control mainly for isochronous and embedded applications. The register is effectively an enhancement to the CKS register.

This register is cleared to 0x00 after a hardware reset to maintain compatibility with 16C550, but is unaffected by software reset. This allows you to select a clock mode and then reset the channel to work around any timing glitches.

CKA[0]: Invert internal RX Clock

This allows the sense of the receiver clock to be inverted. The main use for this is to invert an isochronous input clock so the falling edge is used for sampling rather than the rising edge.

CKA[1]: Invert internal TX clock

This allows the sense of the transmitter clock to be inverted. The main use for this is to invert an isochronous input clock so the rising edge is used for data output rather than the falling edge.

CKA[2]: Invert DTR

This allows the DTR output signal to be inverted, which is most likely to be useful when DTR is selected as being the transmitter clock for isochronous applications input clock, so the falling edge is used for sampling rather than the rising edge.

RS485 Delay Enable ‘RS485_DLYEN’

The RS485_DLYEN register is located at offset 0x14 of the ICR. This register enables the programmable RS485 switch off delay time.

RS485_DLYEN[0]: Enable RS485 Delay

- logic 0 ⇒ Normal 950-style operation
- logic 1 ⇒ When RS485 signaling is enabled, the turn-off time at the end of transmission can be programmed from the commencement of the stop bit. There is a minimum delay of 2 clock cycles (of the transmitter phase clock) plus the delay programmed into the RS485_DLYEN register. The delay comprises number of bits and a phase. The phase represents the oversampling from the TCR register. Take care that the phase programmed is possible. For example, if TCR is set to 4 clocks/bit, then only phases 0, 1, 2 and 3 are valid

RS485 Delay Count ‘RS485_DLYCNT’

The RS485_DLYCNT register is located at offset 0x15 of the ICR. This register set the programmable RS485 switch off delay time, provided the feature has been enabled using RS485_DLYEN[0].

RS485_DLCNT[3:0]: RS485 Phase Delay

The number of phases required for the delay. A phase is the count of transmitter clock cycles per bit (programmable in the TCR). Each bit commences transmission on phase 0. It should be ensured that this phase is consistent with the TCR for correct operation.

RS485_DLCNT[7:4]: RS485 Bit Delay

The number of bit periods required for the delay. Zero indicates the final stop bit.

Local Bus

The OXuPCI952 incorporates a bridge from PCI to the local bus. It allows you to expand the capabilities of your products by adding peripherals to this bus.

The local bus comprises a bi-directional 8-bit data bus, an 8-bit address bus, up to four chip selects, and a number of control signals that allow for easy interfacing to standard peripherals. It also provides eleven active-high or active-low interrupt inputs.

The local bus is configured by LT1 and LT2 in the Local Configuration Register space. By programming these registers, you can alter the characteristics of the local bus to suit the characteristics of the peripheral devices being used.

Operation

The local bus can be accessed using I/O and memory space, in similar fashion to the internal UARTs. The mapping to the devices varies with the application, but the bus is fully configurable to facilitate simple development.

The operation of the local bus is synchronized to the PCI bus clock. The clock signal is output on pin LBCLK if it has been enabled by setting LT2[30].

The eight bit bi-directional pins LBD[7:0] drive the output data onto the bus during local bus write cycles. For reads, the device latches the data read from these pins at the end of the cycle.

The local bus address is placed on pins LBA[7:0] at the start of each local bus cycle and remain latched until the start of the subsequent cycle. If the maximum allowable block size (256 bytes) is allocated to the local bus in I/O space, then as access in I/O space is byte aligned, AD[7:0] are asserted on LBA[7:0]. If a smaller address range is selected, the corresponding upper address lines are set to logic zero.

The control bus comprises up to four chip-select signals LBCS[3:0]#, a read strobe LBRD# and a write strobe LBWR#, in Intel-type interfaces. For Motorola-type interfaces, LBWR# is redefined to perform read/write control signal (LBRDWR#) and the chip-select signals (LBCS[3:0]#) are redefined to data strobe (LBDS[3:0]#).

A reference cycle is defined as two PCI clock cycles after the master asserts the IRDY# signal for the first time within a frame. In general, all the local bus control signals change state in the first cycle after the reference cycle, with offsets to provide suitable setup and hold times for common peripheral devices. However, all the timings can be increased or decreased independently in multiples of PCI clock cycles. This feature enables the card designer to override the length of read or write operations, the address and chip-select set-up and hold timing, and the data bus hold timing so that add-in cards can be configured to suit different speed peripheral devices connected to the local bus. You can also program the data bus to remain in the high impedance state or actively drive the bus during idle periods.

The local bus always returns to an idle state, where no chip-select (data strobe in Motorola mode) signal is active, between adjacent accesses. During read cycles, the local bus interface latches data from the bus on the rising edge of the clock where LBRD# (LBDS[3:0]# in Motorola mode) goes high. Ensure that your peripherals provide the OXuPCI952 with the specified data set-up and hold times for this clock edge.

The local bus cannot accept burst transfers from the PCI bus. If a burst transfer is attempted, the PCI interface signals 'disconnect with data' on the first data phase. The local bus does accept 'fast back-to-back' transactions from PCI.

A PCI target must complete the transaction within 16 PCI clock cycles from assertion of the FRAME# signal otherwise it should signal a retry. During a read operation from the local bus, the OXuPCI952 waits for the master-ready signal (IRDY#) and computes the number of remaining cycles to the de-assertion of the read control signal. If the total number of PCI clock cycles for that frame is greater than 16 clock cycles, the OXuPCI952 posts a retry. The master normally returns immediately and completes the operation in the following frame.

Configuration and Programming

The values of the configuration registers for the local bus controller after reset allow the host system to identify the function and configure its base address registers. Alternatively many of the default values can be re-programmed during device initialization using of the optional serial EEPROM.

The I/O space block can be varied in size from 4 bytes to 256 bytes (32 bytes is the default) by setting LT2[22:20] accordingly. Varying the block size means the I/O space can be allocated efficiently by the system, whatever the application.

The I/O block can then be divided into one, two or four chip-select regions, depending on the setting in LT2[26:23]. To divide the area into a four chip-select region, select the second uppermost non-zero address bit as the Lower-Address-CS-decode. To divide into two regions, select the uppermost address bit. If an address bit beyond the selected range is selected, the entire I/O space is allocated to CS0#. For example, if 32 bytes of I/O space are reserved, the active address lines are A[4:0]. To divide this into four regions, set the Lower Address CS Decode parameter to A3, by programming the value ‘0001’ into LT2[26:23]. To select two regions, choose A4, and to maintain one region, select any value greater than A4.

The memory space block is always 4Kbytes, and always divided into four chip-select regions of 1Kbyte each.

A soft reset facility is provided so software can independently reset the peripherals on the local bus. The local bus reset signals, LBRST and LBRST#, are always active during a PCI bus reset and also when the configuration register bit LT2[29] is set to 1.

The clock enable bit, when set, enables a copy of the PCI bus clock output on the local bus pin LBCLK. A buffered UART clock can also be asserted on the UART_Clk_Out pin. This means that a single oscillator can be used to drive serial ports on the local bus as well as the internal UARTs.

Bidirectional Parallel Port

Operation and Mode Selection

The OXuPCI952 offers a compact, low power, IEEE 1284 compliant host-interface parallel port, designed to interface to many peripherals such as printers, scanners and external drives. It supports compatibility modes, SPP, NIBBLE, PS2, EPP and ECP modes. The register set is compatible with the Microsoft register definition. To enable the parallel port function, the device mode must be set to '001' or '101'. The system can access the parallel port using two 8-byte blocks of I/O space; BAR0 contains the address of the basic parallel port registers, BAR1 contains the address of the upper registers. These are referred to as the 'lower block' and 'upper block' in this section. If the upper block is located at an address 0x400 above the lower block, generic PC device drivers can be used to configure the port, as the addressable registers of legacy parallel ports always have this relationship. If not, a custom driver is needed.

SPP Mode

SPP (output-only) is the standard implementation of a simple parallel port. In this mode, the PD lines always drive the value in the PDR register. All transfers are done under software control. Input must be performed in nibble mode.

Generic device driver software may use the address in I/O space encoded in BAR0 of function 1 to access the parallel port. The default configuration allocates 8 bytes to BAR0 in I/O space.

PS2 Mode

This mode is also referred to as bi-directional or compatible parallel port. To use the PS2 mode, the mode field of the Extended Control Register (ECR[7:5]) must be set to '001', using the negotiation steps as defined by the IEEE 1284 specification.

PS2 operation is similar to SPP mode but, in this mode, directional control of the parallel port data lines (PD[7:0]) is possible by setting and clearing DCR[5], the data direction bit.

EPP Mode

To use the Enhanced Parallel Port 'EPP' the mode bits (ECR[7:5]) must be set to '100' 100' using the negotiation steps as defined by the IEEE 1284 specification.

The EPP address and data port registers are compatible with the IEEE 1284 definition. A write or read to one of the EPP port registers is passed through the parallel port to access the external peripheral. In EPP mode, the STB#, INIT#, AFD# and SLIN# pins change from open-drain outputs to active push-pull (totem pole) drivers (as required by IEEE 1284) and the pins ACK#, AFD#, BUSY, SLIN# and STB# are redefined as INTR#, DATASTB#, WAIT#, ADDRSTB# and WRITE# respectively.

An EPP port access begins with the host reading or writing to one of the EPP port registers. The device automatically buffers the data between the I/O registers and the parallel port depending on whether it is a read or a write cycle. When the peripheral is ready to complete the transfer it takes the WAIT# status line high. This allows the host to complete the EPP cycle.

If a faulty or disconnected peripheral fails to respond to an EPP cycle, the host never sees a rising edge on WAIT#, and subsequently locks up. A built-in time-out facility is provided to prevent this. It uses an internal timer which aborts the EPP cycle and sets a flag in the DSR register to indicate the condition. When the parallel port is not in EPP mode, the timer is switched off to reduce current consumption. The host time-out period is 10μs as specified with the IEEE 1284 specification.

The register set is compatible with the Microsoft register definition. Assuming that the upper block is located 400h above the lower block, the registers are found at offset 000-007h and 400-402h.

The OXuPCI952 supports version 1.7 of the EPP protocol.

ECP Mode

To use the Extended Capabilities Port ('ECP') mode, the mode field of the Extended Control Register (ECR[7:5]) must be set to '011' using the negotiation steps as defined by the IEEE 1284 specification.

ECP mode is compatible with the Microsoft register definition for ECP, and the IEEE 1284 bus protocol and timing.

The ECP mode supports the decompression of Run-length encoded (RLE) data, in hardware. The RLE received data is expanded automatically by the correct number, into the ECP receiver FIFO. Run-length encoding on data to be transmitted is not available in hardware. This needs to be handled in software, if this feature is required.

Assuming that the upper block is located 400h above the lower block, the ECP registers are found at offset 000-007h and 400-402h.

Parallel Port Interrupt

The parallel port interrupt is asserted on the INTA#.

It is enabled by setting DCR[4]. When DCR[4] is set, an interrupt is asserted on the rising edge of the ACK# (INTR#) pin and is held until the status register is read, which resets the INT# status bit as held by the bit DSR[2].

Register Description

The parallel port registers are described below. (Note that it is assumed that the upper block is placed 400h above the lower block.)

Register Name	Address Offset	R/W	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
PDR	000h	R/W	Parallel Port Data Register							
DSR (EPP mode)	001h	R	nBUSY	ACK#	PE	SLCT	ERR#	INT#	1	Timeout
(Other modes)	001h	R	nBUSY	ACK#	PE	SLCT	ERR#	INT#	1	1
DCR	002h	R/W	0	0	DIR	INT_EN	nSLIN#	INIT#	nAFD#	nSTB#
EPPA [1]	003h	R/W	EPP Address Register							
EPPD1 [1]	004h	R/W	EPP Data 1 Register							
EPPD2 [1]	005h	R/W	EPP Data 2 Register							
EPPD3 [1]	006h	R/W	EPP Data 3 Register							
EPPD4 [1]	007h	R/W	EPP Data 4 Register							
EcpDFifo	400h	R/W	ECP Data FIFO							
TFifo	400h	R/W	Test FIFO							
CnfgA	400h	R	Configuration A Register – always 90h							
CnfgB	401h	R	0	int	‘000000’					
ECR	402h	R/W	Mode[2:0]			Reserved – Must write ‘00001’ Reads return FIFO status and Service Interrupt status				
-	403h	-	Reserved							

Note 1: These registers are only available in EPP mode.

Note 2: Prefix ‘n’ denotes that a signal is inverted at the connector. Suffix ‘#’ denotes active-low signaling.

The reset state of PDR, EPPA and EPPD1-4 is not determinable (i.e. 0xXX). The reset value of DSR is ‘XXXXX111’. DCR and ECR are reset to ‘0000XXXX’ and ‘00010101’ respectively.

Parallel Port Data Register ‘PDR’

PDR is located at offset 000h in the lower block. It is the standard parallel port data register. Writing to this register in mode 000 (SPP mode) drives data onto the parallel port data lines. In all other modes the drivers may be tri-stated by setting the direction bit in the DCR. Reads from this register return the value on the data lines.

ECP FIFO Address / RLE

A data byte written to this address is interpreted as an address if bit(7) is set, otherwise an RLE count for the next data byte. Count = bit(6:0) + 1.

Device Status Register ‘DSR’

DSR is located at offset 001h in the lower block. It is a read only register showing the current state of control signals from the peripheral. Additionally in EPP mode, bit 0 is set to ‘1’ when an operation times out.

DSR[0]:

EPP mode: Timeout:

- logic 0 ⇒ EPP Timer Timeout has not occurred
- logic 1 ⇒ Timeout has occurred (Reading this bit clears it)

Other modes: Unused.

This bit is permanently set to 1.

DSR[1]: Unused

This bit is permanently set to 1.

DSR[2]: INT#

- logic 0 ⇒ A parallel port interrupt is pending
- logic 1 ⇒ No parallel port interrupt is pending

This bit is activated (set low) on a rising edge of the ACK# pin. It is de-activated (set high) after reading the DSR.

DSR[3]: ERR#

- logic 0 ⇒ The ERR# input is low
- logic 1 ⇒ The ERR# input is high

DSR[4]: SLCT

- logic 0 ⇒ The SLCT input is low
- logic 1 ⇒ The SLCT input is high

DSR[5]: PE

- logic 0 ⇒ The PE input is low
- logic 1 ⇒ The PE input is high

DSR[6]: ACK#

- logic 0 ⇒ The ACK# input is low
- logic 1 ⇒ The ACK# input is high

DSR[7]: nBUSY

- logic 0 ⇒ The BUSY input is high
- logic 1 ⇒ The BUSY input is low

Device Control Register 'DCR'

DCR is located at offset 002h in the lower block. It is a read-write register which controls the state of the peripheral inputs and enables the peripheral interrupt. When reading this register, bits 0 to 3 reflect the actual state of STB#, AFD#, INIT# and SLIN# pins respectively. When in EPP mode, the WRITE#, DATASTB# AND ADDRSTB# pins are driven by the EPP controller, although writes to this register override the state of the respective lines. The bits in the DCR register must result in these lines to be in the inactive state, allowing the EPP controller to control these lines. This is also applicable for the ECP mode, to allow the ECP controller to control the parallel port pins.

DCR[0]: nSTB#

- logic 0 ⇒ Set STB# output to high (inactive)
- logic 1 ⇒ Set STB# output to low (active)

During an EPP address or data cycle the WRITE# pin is driven by the EPP controller, otherwise it is inactive.

DCR[1]: nAFD#

- logic 0 ⇒ Set AFD# output to high (inactive)
- logic 1 ⇒ Set AFD# output to low (active)

During an EPP address or data cycle the DATASTB# pin is driven by the EPP controller, otherwise it is inactive.

DCR[2]: INIT#

- logic 0 ⇒ Set INIT# output to low (active)
- logic 1 ⇒ Set INIT# output to high (inactive)

DCR[3]: nSLIN#

- logic 0 ⇒ Set SLIN# output to high (inactive)
- logic 1 ⇒ Set SLIN# output to low (active)

During an EPP address or data cycle the ADDRSTB# pin is driven by the EPP controller, otherwise it is inactive.

DCR[4]: ACK Interrupt Enable

- logic 0 ⇒ ACK interrupt is disabled
- logic 1 ⇒ ACK interrupt is enabled

DCR[5]: DIR (DIRECTION) *

- logic 0 ⇒ PD port is output
- logic 1 ⇒ PD port is input

This bit is overridden during an EPP address or data cycle, when the direction of the port is controlled by the bus access (read/write).

* Microsoft's ECP Specification states that all direction changes related to ECP mode must first be made in the PS/2 mode, for reliable operation.

DCR[7:6]: Reserved

These bits are reserved and always set to "00".

EPP Address Register 'EPPA'

EPPA is located at offset 003h in lower block, and is only used in EPP mode. A byte written to this register is transferred to the peripheral as an EPP address by the hardware. A read from this register transfers an address from the peripheral under hardware control.

EPP Data Registers 'EPPD1-4'

The EPPD registers are located at offset 004h-007h of the lower block, and are only used in EPP mode. Data written or read from these registers is transferred to/from the peripheral under hardware control.

ECP Data FIFO

Hardware transfers data from this 16-bytes deep FIFO to the peripheral when DCR(5) = '0'. When DCR(5) = '1' hardware transfers data from the peripheral to this FIFO.

Test FIFO

Used by the software in conjunction with the full and empty flags to determine the depth of the FIFO and interrupt levels.

Configuration A Register

ECR[7:5] must be set to '111' to access this register. Interrupts generated are always level, and the ECP port only supports an **implD** of '001'.

Configuration B Register

ECR[7:5] must be set to '111' to access this register. Read only, all bits are set to 0, except for bit[6] which reflects the state of the interrupt.

Extended Control Register 'ECR'

The Extended Control Register is located at offset 002h in upper block. It is used to configure the operation of the parallel port.

ECR[4:0]: Reserved—write

These bits are reserved and must always be set to "00001".

ECR[0]: Empty—read

When DCR[5] = ‘0’:

- logic 0 ⇒ FIFO contains at least one byte
- logic 1 ⇒ FIFO completely empty

When DCR[5] = ‘1’:

- logic 0 ⇒ FIFO contains at least one byte
- logic 1 ⇒ FIFO contains less than one byte

ECR[1]: Full - read

When DCR[5] = ‘0’:

- logic 0 ⇒ FIFO has at least one free byte
- FIFO completely full

When DCR[5] = ‘1’:

- logic 0 ⇒ FIFO has at least one free byte
- logic 1 ⇒ FIFO full

ECR[2]: serviceIntr - read

When DCR[5] = ‘0’:

- logic 1 ⇒ writeIntrThreshold (8) free bytes or more in FIFO

When DCR[5] = ‘1’:

- logic 1 ⇒ readIntrThreshold (8) bytes or more in FIFO

ECR[7:5]: Mode

These bits define the operational mode of the parallel port:

- logic ‘000’ SPP
- logic ‘001’ PS2
- logic ‘010’ Reserved
- logic ‘011’ ECR
- logic ‘100’ EPP
- logic ‘101’ Reserved
- logic ‘110’ Test
- logic ‘111’ Config

Serial EEPROM

The OXuPCI952 can be configured using an optional serial electrically-erasable programmable read only memory (EEPROM). If the EEPROM is not present, the device remains in its default configuration after reset. Although this may be adequate for some applications, many benefit from the degree of programmability afforded by this feature.

The EEPROM interface on the OXuPCI952 is based on the proprietary serial interface known as Microwire. The interface has four pins which supply the memory device with a clock, a chip-select and serial data input and output lines. In order to read from such a device, a controller has to output serially a read command and an address, and then input serially the data.

The EEPROM interface auto-detects a variety of Microwire compatible EEPROM devices attached to its interface, where the data has been organized as a 16-bit data word format. Devices supported include the NM93C46, C56, C66, C76, and C86.

The OXuPCI952 reads data from the serial EEPROM and writes data into the configuration register space immediately after a PCI bus reset. This sequence ends either when the controller finds no EEPROM is present or when it reaches the end of the EEPROM data.

Following device configuration, driver software can access the serial EEPROM through four bits in the device-specific Local Configuration Register LCC[27:24]. Software can use this register to manipulate the device pins in order to read and modify the EEPROM contents.

To prevent an incorrectly coded EEPROM from ‘hanging’, the PCI system (since all PCI accesses are met with a retry while the external EEPROM device is being accessed), the OXuPCI952 incorporates an EEPROM overrun detection mechanism that terminates the EEPROM download if any attempts are made to read beyond the size of the detected EEPROM. In this case, the overrun flag is set in the local registers (LCC reg, bit 30). The overrun condition does not reset the device to eliminate the effects of any downloaded (corrupt) values. Any overrun indications must be handled by reprogramming the device with the correct EEPROM data, as the state of the OXuPCI952 may be unknown.

A Windows based utility is available to help generate correct data for the EEPROM. For further details please contact you PLX Technology representative.

Microwire is a trade mark of National Semiconductor. For a description of Microwire, please refer to National Semiconductor data manuals.

EEPROM Data Organization

The serial EEPROM data is divided into six zones. The size of each zone is an exact multiple of 16-bit WORDs. Zone0 is allocated to the header. A valid EEPROM program must contain a header.

The EEPROM can be programmed from the PCI bus. Once the programming is complete, the device driver should either reset the PCI bus or set LCC[29] to reload the OXuPCI952 registers from the serial EEPROM.

The general EEPROM data structure is shown in the following table.

EEPROM Zones

DATA Zone	Size (Words)	Description
0	One	Header
1	One or more	Local Configuration Registers
2	One to four	Identification Registers
3	Two or more	PCI Configuration Registers
4	One or more	Power management data
5	Two or more	Function access

Zone 0: Header

The header identifies the EEPROM program as valid. The following table describes the EEPROM header.

Bits	Description
15:8	These bits should return 0x96 to identify a valid program. Once the OXuPCI952 reads 0x96 from these bits, it sets LCC[28] to indicate that a valid EEPROM program is present
7	Reserved. Write ‘0’ to this bit
6	Reserved. Write ‘0’ to this bit
5	Reserved. Write ‘0’ to this bit
4	1 = Zone1 (local configuration) exists 0 = Zone1 does not exist
3	1 = Zone2 (identification) exists 0 = Zone2 does not exist
2	1 = Zone3 (PCI configuration) exists 0 = Zone3 does not exist
1	1 = Zone4 (power management zone) exists 0 = Zone4 does not exist
0	1 = Zone5 (function access zone) exists 0 = Zone5 does not exist

The programming data for each zone follows the preceding zone if it exists.

For example, a header value of 0x961F indicates that all zones exist and they follow one another in sequence. A header value of 0x9605 indicates that only Zones 3 and 5 exist where the header data is followed by Zone3 WORDs, and since Zone4 is missing, Zone3 WORDs are followed by Zone5 WORDs.

Zone 1: Local Configuration Registers

The Zone1 region of EEPROM contains the program value of the vendor-specific Local Configuration Registers using one or more configuration WORDs. Registers are selected using a 7-bit byte-offset field. This offset value is the offset from Base Address Registers in I/O or memory space.

The format of the EEPROM words for this zone, is as shown. Note that not all of the registers in the Local Configuration Register set are writable by EEPROM.

Bits	Description
15	'0' = There are no more configuration WORDs to follow in Zone1. Move to the next available zone or end EEPROM program if no more zones are enabled in the header. '1' = There is another configuration WORD to follow for the Local Configuration Registers
14:8	These seven bits define the byte-offset of the Local Configuration Register to be programmed. For example the byte-offset for LT2[23:16] is 0x0E
7:0	8-bit value of the register to be programmed

The following table shows which Local Configuration Registers are writable from the EEPROM. Note that an attempt by the EEPROM to write to any other offset locations can result in unpredictable behavior.

Offset	Bits	Description	Reference
0x00	1:0	Must be '00'	
0x00	2	Enable UART clock-out	LCC[2]
0x00	4:3	Endian byte-lane select	LCC[4:3]
0x00	6:5	Power down filter	LCC[6:5]
0x00	7	MIO2_PME enable	LCC[7]
0x04	7:0	Multi-purpose I/O configuration	MIC[7:0]
0x05	7:0	Multi-purpose I/O configuration	MIC[15:8]
0x06	5:0	Multi-purpose I/O configuration	MIC[21:16]
0x07	26	Multi-purpose I/O configuration	MIC[26]
0x07	31:29	Clkrun features, filter disable	MIC[31:29]
0x08	7:0	Local bus timing parameters	LT1[7:0]
0x09	7:0	Local bus timing parameters	LT1[15:8]
0x0A	7:0	Local bus timing parameters	LT1[23:16]
0x0B	7:0	Local bus timing parameters	LT1[31:24]
0x0C	7:0	Local bus timing parameters	LT2[7:0]
0x0D	7:0	Local bus timing parameters	LT2[15:8]

Offset	Bits	Description	Reference
0x0E	3:0	Must be '0000'	
0x0E	6:4	I/O space block size	LT2[22:20]
0x0E	7	Lower-Address-CS-Decode	LT2[23]
0x0F	2:0	Lower-Address-CS-Decode	LT2[26:24]
0x0F	4:3	Must be '00'	LT2[28:27]
0x0F	5	Must be '0'	
0x0F	6	Local bus clock enable	LT2[30]
0x0F	7	Bus interface type	LT2[31]
0x1E	1:0	UART interrupt mask	GIS[17:16]
0x1E	4	MIO0/parallel port interrupt mask	GIS[20]
0x1E	7:5	Multi-purpose I/O interrupt mask	GIS[23:21]
0x1F	7:0	Multi-purpose I/O interrupt mask	GIS[30:24]

Zone 2: Identification Registers

The Zone2 region of the EEPROM contains the program value for vendor ID and subsystem vendor ID. The format of device identification configuration WORDs are described in the following table.

Bits	Description
15	'0' = There are no more Zone2 (Identification) bytes to program. Move to the next available zone or end EEPROM program if no more zones are enabled in the header. '1' = There is another Zone2 (identification) byte to follow
14:8	0x00 = Vendor ID bits [7:0] 0x01 = Vendor ID bits [15:8] 0x02 = Subsystem vendor ID [7:0] 0x03 = Subsystem vendor ID [15:8] 0x03 to 0x7F = Reserved
7:0	8-bit value of the register to be programmed

Zone 3: PCI Configuration Registers

The Zone3 region of the EEPROM contains any changes required to the PCI Configuration Registers (with the exception of the vendor ID and subsystem vendor ID which are programmed through Zone2). This zone is divided into two groups, each of which consists of a function header WORD and one or more configuration WORDs for that function. The function header is described in the following table.

Bits	Description
15	‘0’ = End of Zone 3 ‘1’ = Define this function header
14:3	Reserved. Write zeros
2:0	Function number for the following configuration WORD(s): ‘000’ = Function0 (internal UARTs) ‘001’ = Function1 (local bus/parallel port) Other values = Reserved

The subsequent WORDs for each function contain the address offset and a byte of programming data for the PCI configuration space belonging to the function number selected by the proceeding function header. The format of configuration WORDs for the PCI Configuration Registers is described below.

Bits	Description
15	‘0’ = This is the last configuration WORD in for the selected function in the function header ‘1’ = There is another WORD to follow for this function
14:8	These seven bits define the byte-offset of the PCI Configuration Register to be programmed. For example the byte-offset of the Interrupt Pin Register is 0x3D
7:0	8-bit value of the register to be programmed

The following table shows which PCI Configuration Registers are writable from the EEPROM for each function.

Offset	Bits	Description
0x02	7:0	Device ID bits 7 to 0
0x03	7:0	Device ID bits 15 to 8
0x06	3:0	Must be ‘0000’
0x06	4	Extended capabilities
0x06	7:5	Must be ‘000’
0x09	7:0	Class code bits 7 to 0
0x0A	7:0	Class code bits 15 to 8
0x0B	7:0	Class code bits 23 to 16
0x2E	7:0	Subsystem ID bits 7 to 0
0x2F	7:0	Subsystem ID bits 15 to 8
0x3D	7:0	Interrupt pin
0x42	7:0	Power management capabilities bits 7 to 0
0x43	7:0	Power management capabilities bits 15 to 8

Zone 4: Power Management DATA (and DATA_SCALE Zone)

Zone 4 of the EEPROM provides each function's Power Management Registers with user-defined values for the DATA and DATA_SCALE fields, which are provided to the system software during the initial configuration when requests are made through the DATA_SELECT field.

Since there are 16 possible values for the DATA_SELECT fields, the system can request up to 16 sets of DATA and DATA_SCALE values. The organization of the EEPROM data for this zone is shown in the following table. Each DATA and DATA_SCALE value being programmed is relevant to a particular value of the DATA_SELECT parameter.

Bits	Description
15	'0' = There are no more configuration WORDs to follow in Zone4. Move to the next available zone or end EEPROM program if no more zones are enabled in the header. '1' = There is another configuration WORD to follow for the power management data zone
14	Function select 0 = Values applicable for Function 0 1 = Values applicable for Function 1
13:10	DATA SELECT This indicates for which DATA_SELECT value the DATA and DATA_SCALE values in this particular word are associated with. DATA SELECT: 1 of 16 values (0h to Fh)
9:8	DATA SCALE value User value corresponding to chosen DATA SELECT
7:0	DATA register value User value corresponding to chosen DATA SELECT

Zone 5: Function Access

Zone 5 allows each of the two internal UARTs, devices on the 8-bit local bus, or the parallel port of the OXuPCI952 to be pre-configured, prior to any PCI accesses. This is very useful when these functions need to run with (typically generic) device drivers and these drivers are not capable of using the Enhanced features/modes of these logical units. For example, the 950 mode of the UARTs offer high performance. Generic UART drivers are unable to set the 950 mode since this requires setting of registers outside the register definition within the generic device drivers. By using function access, the relevant UART registers can be accessed (setup) using the EEPROM to enable/customize these features before control is handed to these device drivers.

Each 8-bit function access is equivalent to accessing each function through its assigned I/O Base Address Register (BAR), with the exception that a function read access does not return any data (it is discarded internally). The internal UARTs, 8-bit local bus or the parallel port behave as though these function accesses using the EEPROM are equivalent to PCI-based I/O read/write accesses.

Each entry for the function access zone comprises two 16 bit words (word pairs), with the exception that ending this zone only requires a single word (all 0's) for 'termination'. The format is as shown in the following table.

Word	Bits	Description
1	15	'1' = Function access word pair available '0' = End function access zone * (over to next available zone or end EEPROM download)
1	14:12	BAR number to access Acceptable BARs 000 to 100. Others reserved
1	11	'0' = Read access required (data is discarded) '1 = Write access required
1	10:8	Function number, requiring access: 000 = Function 0 (UARTs) 001 = Function 1 (8-bit local bus / parallel Port). *Others Reserved*.
1	7:0	I/O address to access This is the address that needs to be written/read and is the offset address from the specified BAR. E.g. to access SPR register of UART, address is 00000111 (7dec).

* When ending function access zone, only a single word is required (not word pairs) and all fields must be zeros.

First Word of Function Access Word Palr

Word	Bits	Description
2	15	'1'
2	14:8	Reserved—write 0's
2	7:0	Data to be written to specified address. Field is unused for function access READS (set to 0's for reads)

Second Word of Function Access Word Pair

When directing function access to each function, you need to ensure that the BAR number and I/O address range are compatible with the function's I/O make-up for the given device mode. The OXuPCI952 rejects accesses to BARs which do not hold the I/O base address of the function and any addresses outside the workable address range for that I/O BAR.

For example, when the UARTs BAR0 to BAR1 of function 0 are an 8-byte I/O Base Address Register that provide individual access to the two internal UARTs. In this case, function access only accepts accesses to BAR0 to BAR1 of Function 0, and with the I/0 address range 0 to 7. Accesses to other BARs or address ranges are rejected, for function 0.

Function access examples:

- Enable Internal loopback of UART 0 (enable bit 4 of UART 0's MCR register)

  1000100000000100

  1000000000010000

  - 1st Word: Function Number = 000, BAR No = 000 (UART0), Write Access, Address=00000100 (MCR reg)
  - 2nd Word: Data to be written=00010000

- Enable FIFO of UART 1 (enable bit 0 of UART 1's FCR register)

  1001100000000010

  1000000000000001

  - 1st Word: Function Number = 000, BAR No = 001 (UART1), Write access, address=00000010 (FCR reg)
  - 2nd Word: Data to be written=00000001

- Read IER Register, UART 0. End function access zone

  1000000000000001 (function access word pair)

  1000000000000000

  0000000000000000 (end function access zone *)

  - 1st Word: Function Number = 000, BAR No =000 (UART0), Read access, address=00000001 (IER reg)
  - 2nd Word: No data to be written (as read access). Read data is discarded

  * When ending function access zone, only a single word is required (not word pairs) and all fields must be zeros.

Power Consumption

The values in the following tables do not include any statistical spreads and are suitable for indications only.

The following table shows OXuPCI952 5V power consumption.

Symbol	Parameter	Condition	Typical	Max	Units
Icc	Operating supply current on power-on	Xtal = 1.8432Mhz crystal	25	63	mA
		Xtal = 14.745Mhz crystal	30	67	
		Xtal = 60Mhz Osc	55	85	
	Operating supply current in normal mode*	Xtal = 1.8432Mhz crystal	23.9	47	
		Xtal = 14.745Mhz crystal	27.5	51	
		Xtal = 60Mhz Osc	44.4	71	
	Operating supply current in power-down mode	Xtal = 1.8432Mhz crystal	21.6	65	
		Xtal = 14.745Mhz crystal	17.3	41	
		Xtal = 60Mhz Osc	27.4	47	

The following table shows OXuPCI952 3.3V power consumption.

Symbol	Parameter	Condition	Typical	Max	Units
Icc	Operating supply current on power-on	Xtal = 1.8432Mhz crystal	22	33	mA
		Xtal = 14.745Mhz crystal	20	31	
		Xtal = 60Mhz Osc	25	39	
	Operating supply current in normal mode*	Xtal = 1.8432Mhz crystal	23.9	47	
		Xtal = 14.745Mhz crystal	22.9	67	
		Xtal = 60Mhz Osc	39.5	83	
	Operating supply current in power-down mode	Xtal = 1.8432Mhz crystal	15.4	71	
		Xtal = 14.745Mhz crystal	17.7	73	
		Xtal = 60Mhz Osc	21.8	75	

* Values have been measured from a few samples running under the following conditions:

- PCI CLK active, SYSTEM CLK line driven by a crystal or clock module
- All PCI configuration accesses over
- Single UART channel (1 COM port) set to operate at the maximum data-transfer rate, with its internal loop-back enabled
- Values measured during file transfers (at the max rate)

DC Electrical Characteristics

The following table shows absolute maximum ratings.

Symbol	Parameter	Rating	Unit
V_{CC}	Core power supply	-0.3 to 3.9	V
V_{CC5}	Power supply of 5V I/O	-0.3 to 6.0	V
V_{IN3}	Input voltage of 3.3V I/O	-0.3 to V_{CC} + 0.3	V
	Input voltage of 3.3V with 5V tolerance I/O	-0.3 to V_{CC5} + 0.3	V
V_{IN5}	Input voltage of 5V I/O	-0.3 to V_{CC5} + 0.3	V
V_{OUT3}	Output voltage of 3.3V I/O	-0.3 to V_{CC} + 0.3	V
V_{OUT5}	Output voltage of 5V I/O	-0.3 to V_{CC5} + 0.3	V
T_{STG}	Storage temperature	-40 to 150	°C

The following table shows recommended operating conditions.

Symbol	Parameter		Min	Typ	Max	Unit
V_{CC}	Core power supply		3.0	3.3	3.6	V
$V_{CC}5$	Power supply of 5V I/O	Commercial	4.75	5.0	5.25	V
		Industrial	4.5	5.0	5.5	V
V_{IN3}	Input voltage of 3.3V I/O		0	3.3	3.6	V
	Input voltage of 3.3V I/O with 5V tolerance		0	3.3	5.25	V
V_{IN5}	Input voltage of 5V I/O		0	5	5.25	V
T_J	Junction operating temperature	Commercial	0	25	115	°C
		Industrial	–40	25	125	°C

PCI I/O Buffers (Universal Signaling)

All PCI I/O buffers are compliant with the PCI 3.3V/5.0V DC/AC specifications, as defined in the *PCI Local Bus Specification 2.3/3.0*.

The following table shows DC characteristics of 5V PCI I/O cell.

Symbol	Parameter	Condition	Min	Typ	Max	Unit
VCCK	Core power supply	Core area	3.0	3.3	3.6	V
VCC3I	Power supply	5V I/O	4.5	5.0	5.5	V
VCC3O	Power supply		4.5	5.0	5.5	V
V_{il}`	Input low voltage	PCI	-0.5		0.8	V
V_{ih}	Input high voltage		2.0		V_{cc5o} +0.5	V
V_t	Switching threshold	PCI		1.4		V
V_{ol}	Output low voltage	I_{out} = 3mA, 6mA			0.55	V
V_{oh}	Output high voltage	I_{out} = -2mA	2.4			V
I_{ol} (*)	Switching current low	$V_{out} \geq 2.2$	95			mA
		$2.2 > V_{out} > 0.55$	V_{out}/ 0.023			mA
		$0.71 > V_{out} \geq 0$			Eq. A(*)	mA
	Test point	V_{out} = 0.71			206	mA

Symbol	Parameter	Condition	Min	Typ	Max	Unit
I_{oh} (**)	Switching current high	$0 < V_{out} < 1.4$	-44			mA
		$1.4 < V_{out} < 2.4$	-44 + (V_{out}-1.4) / 0.024			mA
		$3.1 < V_{out} < V_{cc}$			Eq. B(**)	mA
	Test point	V_{out} = 3.1			-142	mA

(*) Eq. A = 78.5 * V_{out} * (4.4 –V_{out})

(**) Eq. B = 11.9 * (V_{out} – 5.25) * (V_{out} + 2.45)

The following table shows DC characteristics of 3.3V PCI I/O cells.

Symbol	Parameter	Condition	Min	Typ	Max	Units
VCCK	Core power supply	Core area	3.0	3.3	3.6	V
VCC3I	Power supply	3.3V I/O	3.0	3.3	3.6	V
VCC3O	Power supply		3.0	3.3	3.6	V
V_{il}`	Input low voltage	PCI	-0.5		$0.3V_{cc3o}$	V
V_{ih}	Input high voltage		$0.5V_{cc3o}$		V_{cc3o}+ 0.5	V
V_t	Switching threshold	PCI		1.3		V
V_{ol}	Output low voltage	I_{out} = 1500µA 6mA			0.1* V_{cc3o}	V
V_{oh}	Output high voltage	I_{out} = -500µA 6mA	0.9* V_{cc3o}			V
R_{pu}	Input pull-up resistance	V_{in} = 0	40	75	190	KΩ
R_{pd}	Input pull-down resistance	V_{in} = VCC3I	40	75	190	KΩ
I_{ol} (*)	Switching current low	$V_{cc3o} > V_{out} \geq 0.6V_{cc3o}$	$16V_{cc3o}$			mA
		$6V_{cc3o} > V_{out} \geq 0.1V_{cc3o}$	$26.7V_{out}$			mA
		$0.18V_{cc3o} > V_{out} \geq 0$			Eq. C(*)	mA
	Test point	$V_{out} = 0.18V_{cc3o}$			$38V_{cc3o}$	mA

Symbol	Parameter	Condition	Min	Typ	Max	Units
I_{oh} (**)	Switching current high	$0 < V_{out} < 0.3V_{cc3o}$	$-12V_{cc3o}$			mA
		$0.3V_{cc3o}<V_{out}< 0.9V_{cc3o}$	$-17.1*(V_{cc3o}-V_{out})$			mA
		$0.7V_{cc3o}<V_{out}< V_{cc3o}$			Eq. D(**)	mA
	Test point	$V_{out} = 0.7\ V_{cc3o}$			$-32V_{cc3o}$	mA

(*) Eq.C = $(256/V_{cc3o}) * V_{out} * (V_{cc3o} – V_{out})$

(**) Eq.D = $(98.0/V_{cc3o}) * (V_{out} – V_{cc3o}) * (V_{out} + 0.4\ V_{cc3o})$

Non-PCI I/O Buffers

The following table shows DC characteristics of 3.3V I/O cells.

Symbol	Parameter	Condition	Min	Typ	Max	Units
VCCK	Core power supply	Core area	3.0	3.3	3.6	V
VCC3I	Power supply	3.3V I/O	3.0	3.3	3.6	V
VCC3O	Power supply		3.0	3.3	3.6	V
V_{il}`	Input low voltage	CMOS			$0.3*V_{cc}$	V
V_{ih}	Input high voltage		$0.7*V_{cc}$			V
V_{il}`	Input low voltage	LVTTL			0.8	V
V_{ih}	Input high voltage		2.0			V
V_{ol}	Output low voltage	I_{ol} = 2 ~ 24mA			0.44	V
V_{oh}	Output high voltage	I_{oh} = -2 ~ -24mA	2.4			V

The following table shows DC characteristics of 5V-tolerant I/O cells.

Symbol	Parameter	Condition	Min	Typ	Max	Units
VCCK	Core power supply	Core area	3.0	3.3	3.6	V
VCC5I	Power supply	5V I/O	4.5	5.0	5.5	V
VCC5O	Power supply		4.5	5.0	5.5	V
V_{il},	Input low voltage	CMOS			$0.3*V_{cc}$	V
V_{ih}	Input high voltage		$0.7*V_{cc}$			V
V_{il},	Input low voltage	TTL			0.8	V
V_{ih}	Input high voltage		2.0			V
V_{ol}	Output low voltage	I_{ol} = 2 ~ 24mA			0.4	V
V_{oh}	Output high voltage	I_{oh} = -2 ~ -24mA	3.5			V
R_{pu}	Input pull-up resistance***	V_{in} = 0	30	50	125	KΩ

*** Applies to EEPROM Di pin only.

The following table shows DC leakage current.

Symbol	Parameter	Min	Typ	Max	Units
I_{IN}	Input leakage current	-10	±1	10	uA
I_{OZ}	Tri-state leakage current	-10	±1	10	uA

AC Electrical Characteristics

The timings for PCI pins comply with PCI specification for the 3.3 Volt signaling environment.

The following table shows AC specifications for the PCI bus.

Symbol	Parameter	Condition	Min	Max	Unit	Note
Ioh(AC)	Switching current high	0<Vout<=0.3Vcc 0.3Vcc<Vout<0.9Vcc 0.7Vcc<Vout<Vcc	TBD	TBD	mA mA mA	1 1 1,2
Iol(AC)	Switching current low	Vcc>Vout>=0.6Vcc 0.6Vcc>Vout>0.1Vcc 0.18Vcc>Vout>0	TBD	TBD	mA mA mA	1 1 1,2
Slewr	Output rise slew rate	0.2Vcc-0.6Vcc load	TBD	TBD	V/ns	3
Slewf	Output fall slew rate	0.6Vxx-0.2Vcc load	TBD	TBD	V/ns	3

Note 1: This specification does not apply to PCI_CLK and RST# pins. “Switching Current High” specifications are not relevant to SERR#, PME#, INTA#.

Note 2:
Equation C Ioh = (98.0/Vcc)*(Vout-Vcc)*(Vout+0.4Vcc) for Vcc>Vout>0.7Vcc
Equation D Iol = (256/Vcc)*Vout*(Vcc-Vout) for 0V<Vout<0.18Vcc.

Note 3: For test circuit, see Section 4.2.2.2 of the *PCI Local Bus Specification*. This figure does not apply to open-drain outputs.

PCI Bus Timings

The PCI interface is compliant with Table 4-6 of the *PCI Local Bus Specification, Rev 2.3*, of which relevant portions are reproduced in the following table.

Symbol	Parameter	Min	Max	Units
T_{val}	CLK to signal valid delay - bused signals	2	11	ns
$T_{val}(ptp)$	CLK to signal valid delay - point to point	2	12	ns
T_{on}	Float to active delay	2		ns
T_{off}	Active to float delay		28	ns
T_{su}	Input setup time to CLK - bused signals	7		ns
$T_{su}(ptp)$	Input setup time to CLK - point to point	10, 12		ns
T_{h}	Input hold time from CLK	0		ns
T_{rst}	Reset active time after power stable	1		ms
$T_{rst-clk}$	Reset active time after CLK STABLE	100		µs
$T_{rst-off}$	Reset active to output float delay		40	ns
T_{rhfa}	RST# high to first configuration access	2^{25}		clocks
T_{rhff}	RST# high to first FRAME# assertion	5		clocks
T_{pvrh}	Power valid to RST# high	100		ms

Local Bus

By default, the local bus control signals change state in the cycle immediately following the reference cycle. The reference cycle is a marker positioned at the second edge of the PCI_CLK after the IRDY# signal is asserted (PCI_CLK 4 in the diagrams). All local bus timing parameters are with respect to the local bus reference (or marker), with offsets to provide set-up and hold times for common peripherals in INTEL mode.

The tables below show the local bus parameters using default values for LT1/LT2 local registers, however each of the parameters can be increased or decreased by a number of PCI clock cycles by adjusting the parameters in registers LT1 and LT2.

All timing parameters assume a loading of 50 pF on the local bus output pins.

The following table shows read operation from an Intel-type local bus

Symbol	Parameter	Min	Max	Units
t_{ref}	IRDY# falling to reference cycle	2 * PCI CLKS		
t_{za}	Reference to address valid	9	15	ns
t_{ard}	Address valid to LBRD# falling	3	5	ns
t_{zrcs1}	Reference to LBCS# falling	9	15	ns
t_{zrcs2}	Reference to LBCS# rising	(3 * PCI_CLK) + 11	(3 * PCI_CLK) + 20	ns
t_{csrd}	LBCS# falling to LBRD# falling	3	5	ns
t_{rdcs}	LBRD# rising to LBCS# rising	4	6	ns
t_{zrd1}	Reference to LBRD# falling	11	19	ns
t_{zrd2}	Reference to LBRD# rising	92	98	ns
t_{drd}	Data bus floating to LBRD# falling	4	7	ns
t_{zd1}	Reference to data bus floating at the start of the read transaction	8	13	ns
t_{zd2}	Reference to data bus driven by OXuPCI952 at the end of the read transaction	(4 * PCI_CLK) + 9	(4 * PCI_CLK) + 15	ns
t_{sd}	Data bus valid to LBRD# rising	17	-	ns
t_{hd}	Data bus valid after LBRD# rising	0	-	ns
t_{LBCLK}	Reference to LBCLK output delay	7	12	ns

The following table shows write operation to an Intel-type local bus.

Symbol	Parameter	Min	Max	Units
t_{ref}	IRDY# falling to reference	2 * PCI CLKS		
t_{za}	Reference to address valid	9	15	ns
t_{awr}	Address valid to LBWR# falling	2	3	ns
t_{zwcs1}	Reference to LBCS# falling	9	15	ns
t_{zwcs2}	Reference to LBCS# rising	(2 * PCI_CLK) + 11	(2 * PCI_CLK) + 20	ns
t_{cswr}	LBCS# falling to LBWR# falling	2	2	ns
t_{wrcs}	LBWR# rising to LBCS# rising	5	8	ns
t_{zwr1}	Reference to LBWR# falling	10	17	ns
t_{zwr2}	Reference to LBWR# rising	(2 * PCI_CLK) + 7	(2 * PCI_CLK) + 12	ns
t_{zdv}	Reference to data bus valid	13	22	ns
t_{zdf}	Reference to data bus high-impedance	Note1	Note1	ns
t_{wrdi}	LBWR# rising to data bus invalid	Note1	Note1	Ns
t_{LBCLK}	Reference to LBCLK output delay	7	12	ns

The following table shows read operation from a Motorola-type local bus.

Symbol	Parameter	Min	Max	Units
t_{ref}	IRDY# falling to reference	2 * PCI CLKS		
t_{za}	Reference to address valid	9	15	ns
t_{ads}	Address valid to LBDS# falling	3	5	ns
t_{zrds1}	Reference to LBDS# falling	12	20	ns
t_{zrds2}	Reference to LBDS# rising	(3 * PCI_CLK) + 8	(3 * PCI_CLK) + 15	ns
vt_{drd}	Data bus floating to LBDS# falling	4	8	ns
t_{zd1}	Reference to data bus floating at the start of the read transaction	8	13	ns
t_{zd2}	Reference to data bus driven by OXuPCI952 at the end of the read transaction	(4 * PCI_CLK) + 9	(4 * PCI_CLK) + 16	ns
t_{sd}	Data bus valid to LBDS# rising		-	ns
t_{hd}	Data bus valid after LBDS# rising		-	ns
t_{LBCLK}	Reference to LBCLK output delay	7	12	ns

The following table shows write operation to a Motorola-type local bus.

Symbol	Parameter	Min	Max	Units
t_{ref}	IRDY# falling to reference	2 * PCI CLKS		
t_{za}	Reference to address valid	9	15	ns
t_{ads}	Address valid to LBDS# falling	3	5	ns
t_{zw1}	Reference to LBRDWR# falling	7	12	ns
t_{zw2}	Reference to LBRDWR# rising	(2 * PCI_CLK) + 9	(2 * PCI_CLK) + 16	ns
t_{wds}	LBRDWR# falling to LBDS# falling	5	8	ns
t_{dsw}	LBDS# rising to LBRDWR# rising	1	2	ns
t_{zwds1}	Reference to LBDS# falling	12	20	ns
t_{zwds2}	Reference to LBDS# rising	(2 * PCI_CLK) + 9	(2 * PCI_CLK) + 15	ns
t_{zdv}	Reference to data bus valid	13	22	ns
t_{zdf}	Reference to data bus high-impedance	Note1	Note1	ns
t_{dsdi}	LBDS# rising to data bus invalid	Note1	Note1	Ns
t_{LBCLK}	Reference to LBCLK output delay	7	12	ns

Note 1: For local bus writes, values on the data bus persist until the next read/ write local bus transaction.

Serial Ports

The following table shows isochronous (x1 clock) timing.

Symbol	Parameter	Min	Max	Units
t_{irs}	SIN set-up time to isochronous input clock ‘Rx_Clk_In rising [1]	2	5	ns
t_{irh}	SIN hold time after isochronous input clock ‘Rx_Clk_In’ rising [1]	1	–	ns
t_{its}	SOUT valid after isochronous output clock ‘Tx_Clk_Out’ falling [1]	2	4	ns

Note 1: In Isochronous mode, transmitter data is available after the falling edge of the x1 clock and the receiver data is sampled using the rising edge of the x1 clock. Ensure that mark-to-space ratio of the x1 clock meets set-up and hold timing constraints. One way of achieving this is to choose a crystal frequency which is twice the required data rate and then divide the clock by two using the on-board prescaler. In this case the mark-to-space ratio is 50/50 for the purpose of set-up and hold calculations.

Note 2: Timing values are applicable to a load of 50pF.

Timing Waveforms

Figure 6 PCI Read Transaction from Internal UARTs

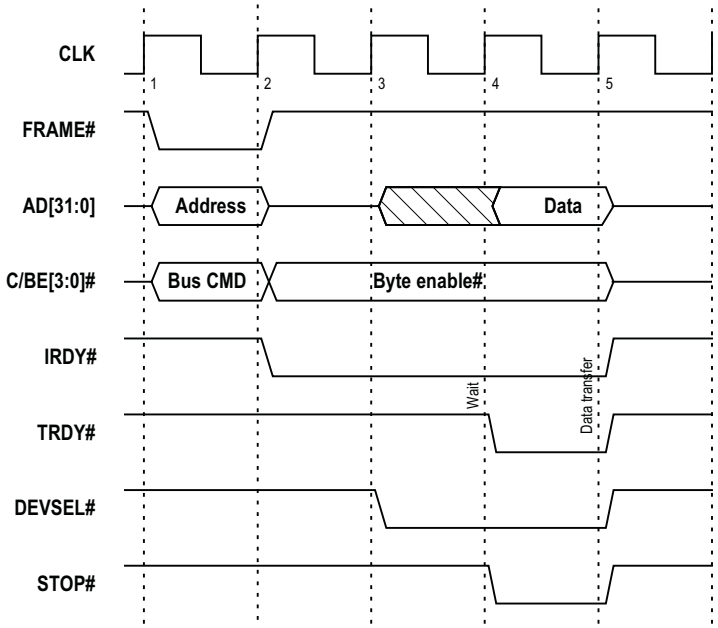


Figure 7 PCI Write Transaction to Internal UARTs

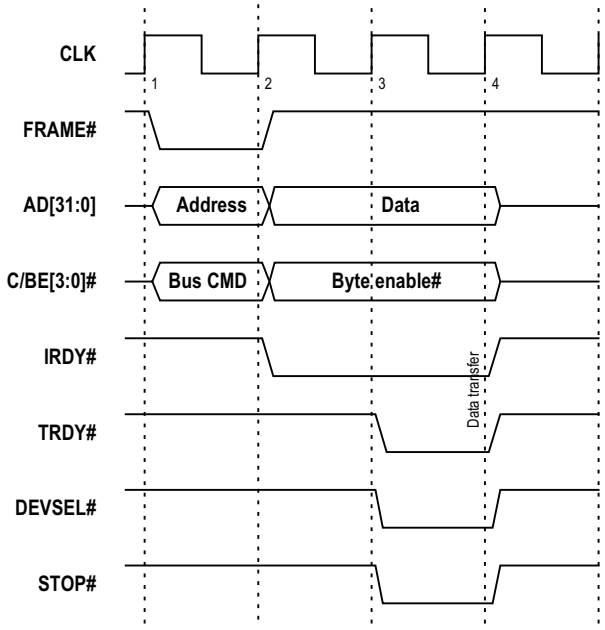


Figure 8 PCI Read Transaction from Local Configuration Registers

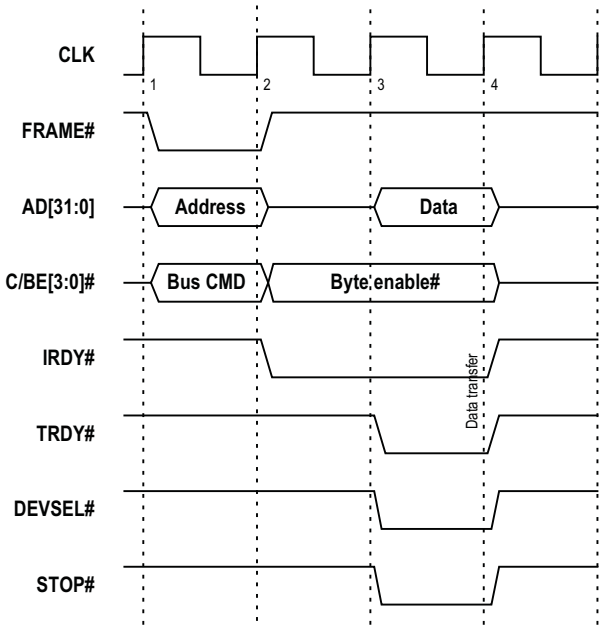


Figure 9 PCI Write Transaction to Local Configuration Registers

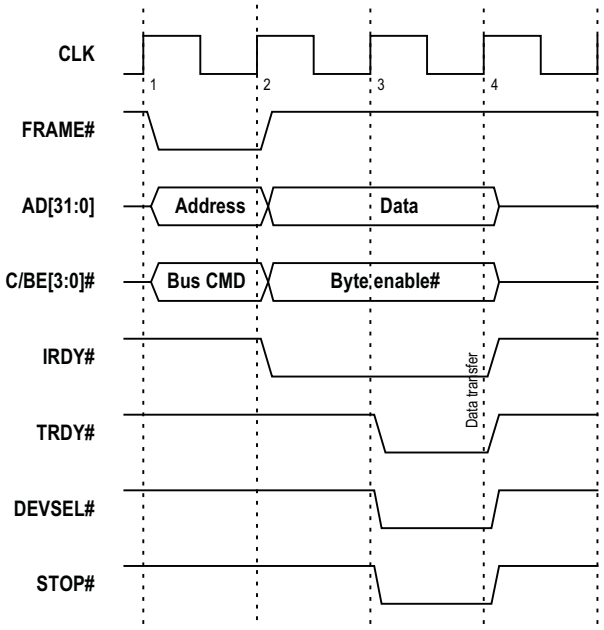


Figure 10 PCI Read Transaction from Intel-type Local Bus

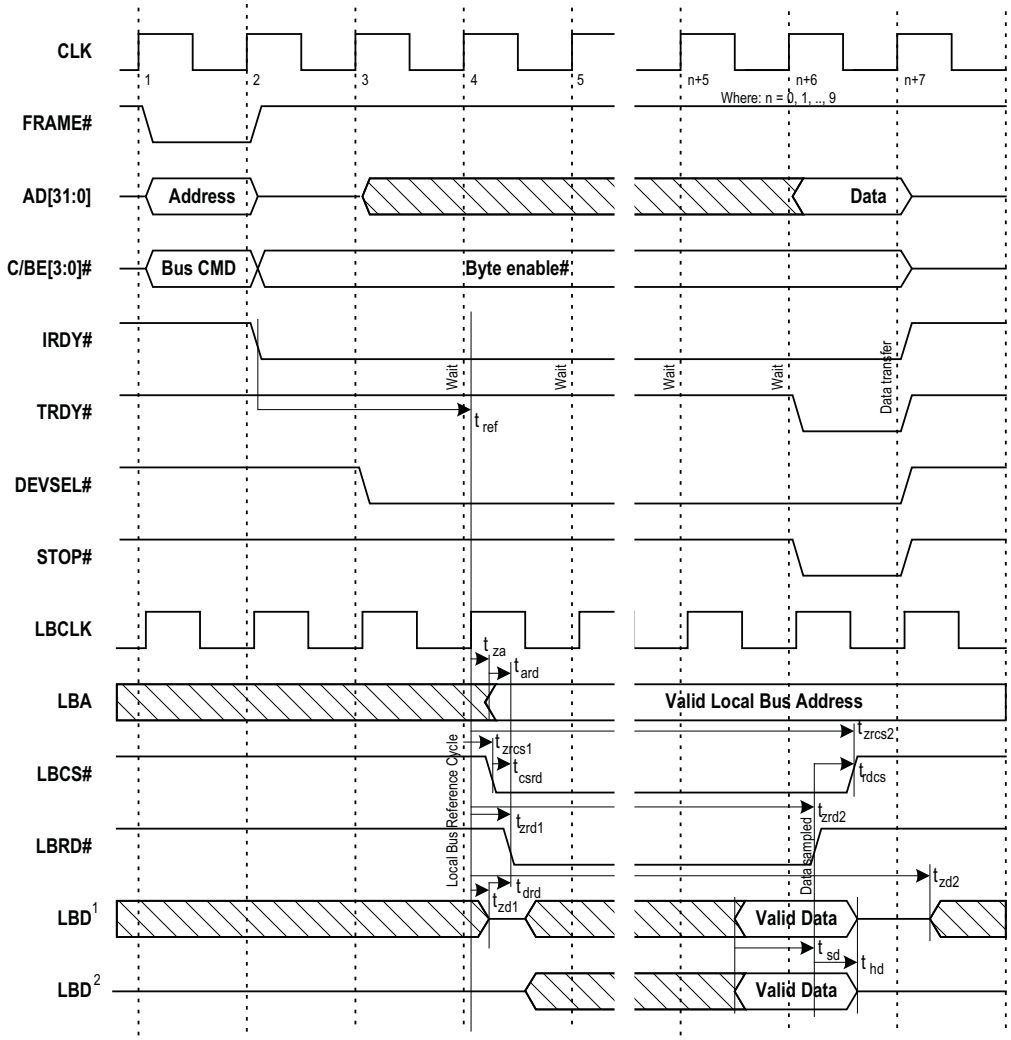


Figure 11 PCI Write Transaction to Intel-type Local Bus

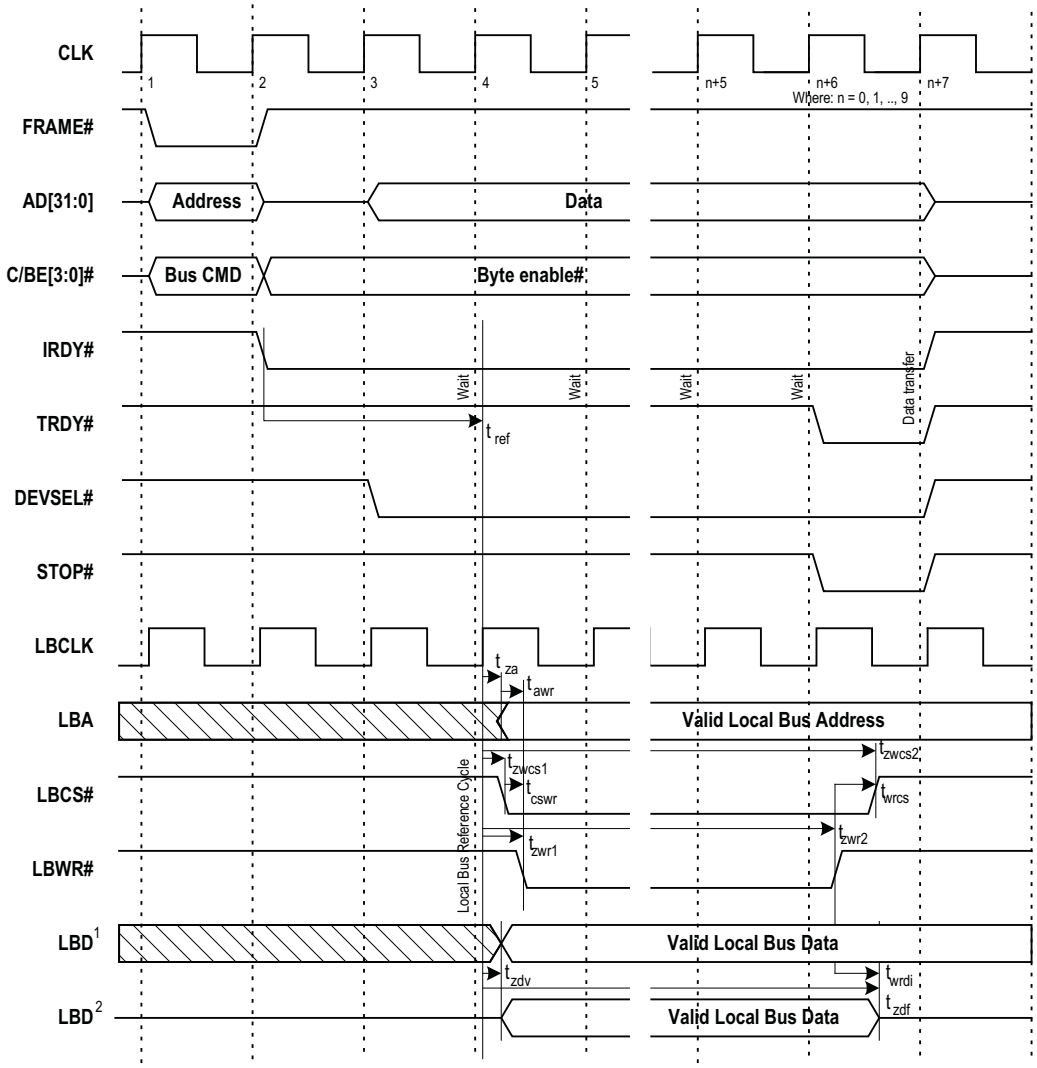


Figure 12 PCI Read Transaction from Motorola-type Local Bus

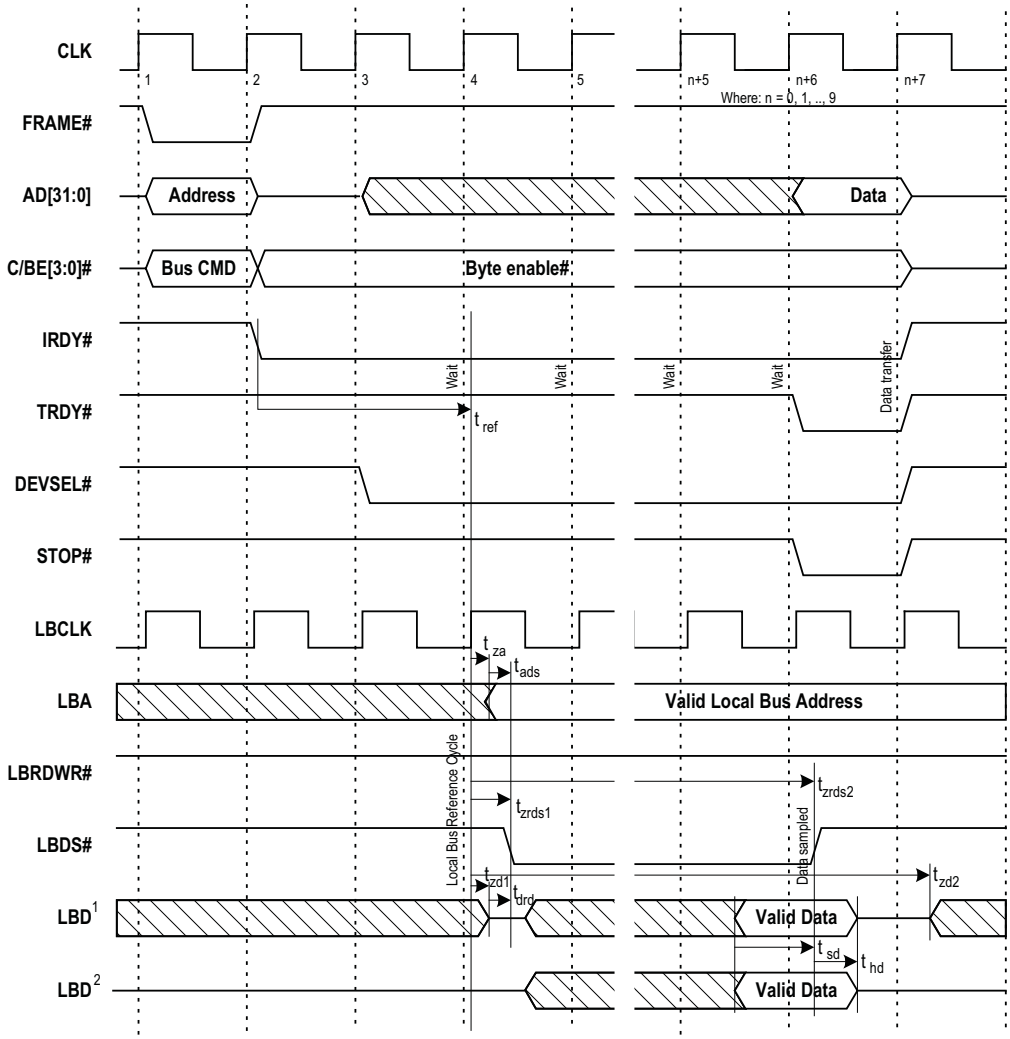


Figure 13 PCI Write Transaction to Motorola-type Local Bus

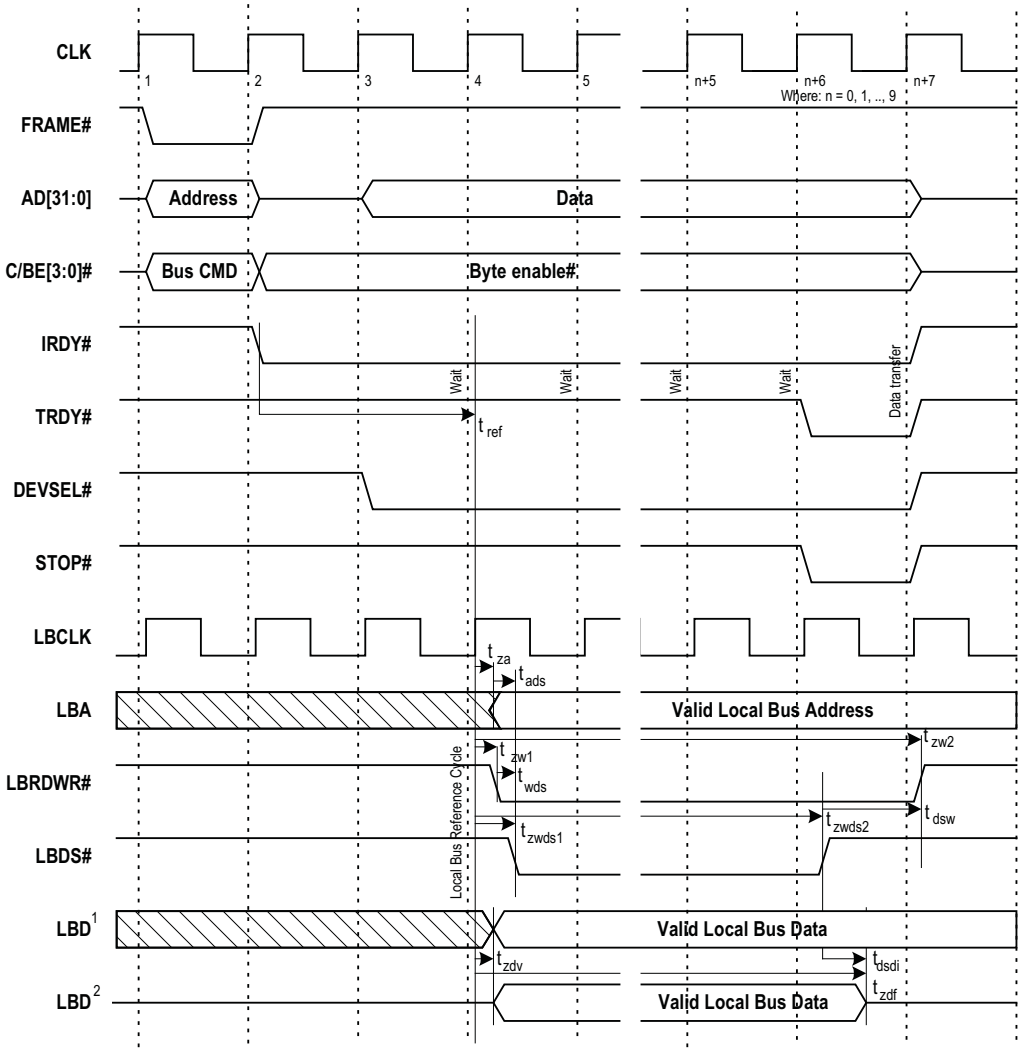


Figure 14 Isochronous (x1 clock) Timing Waveform

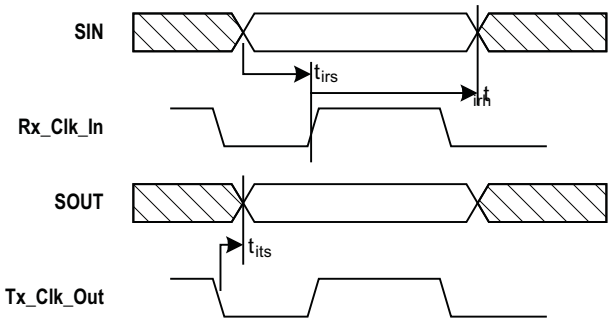
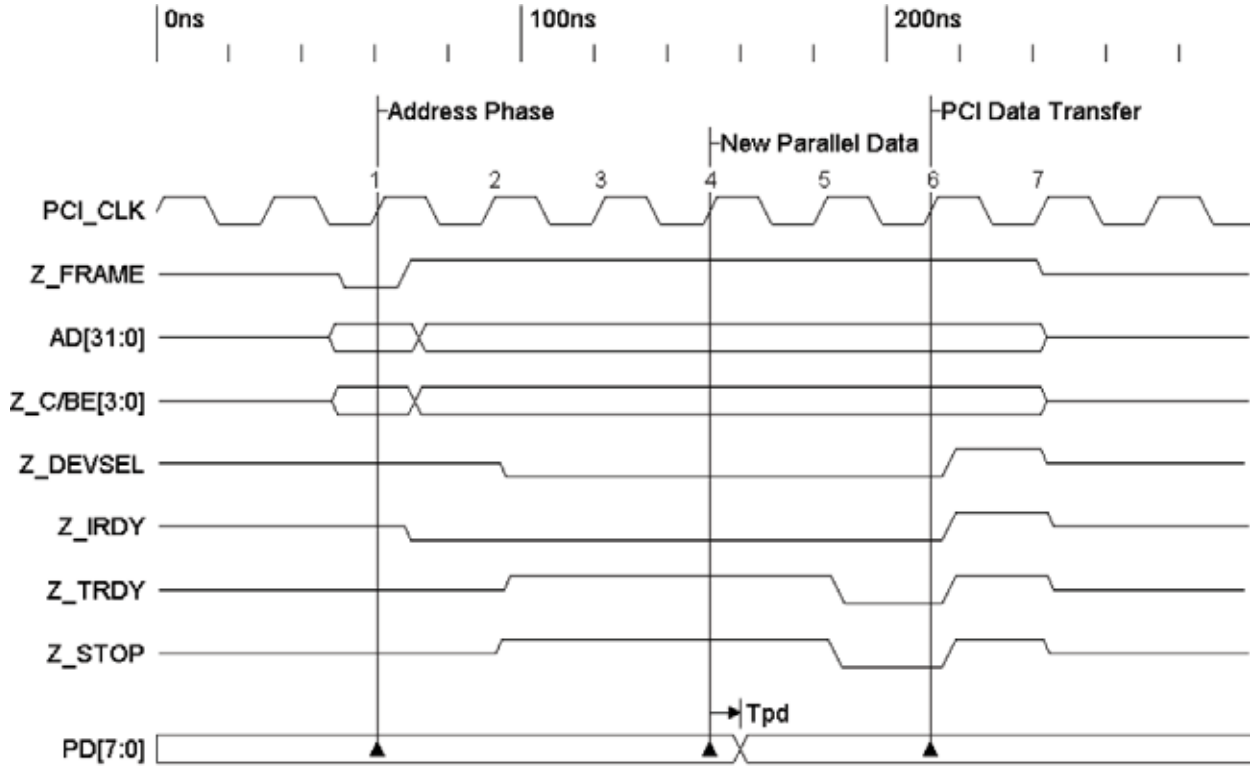


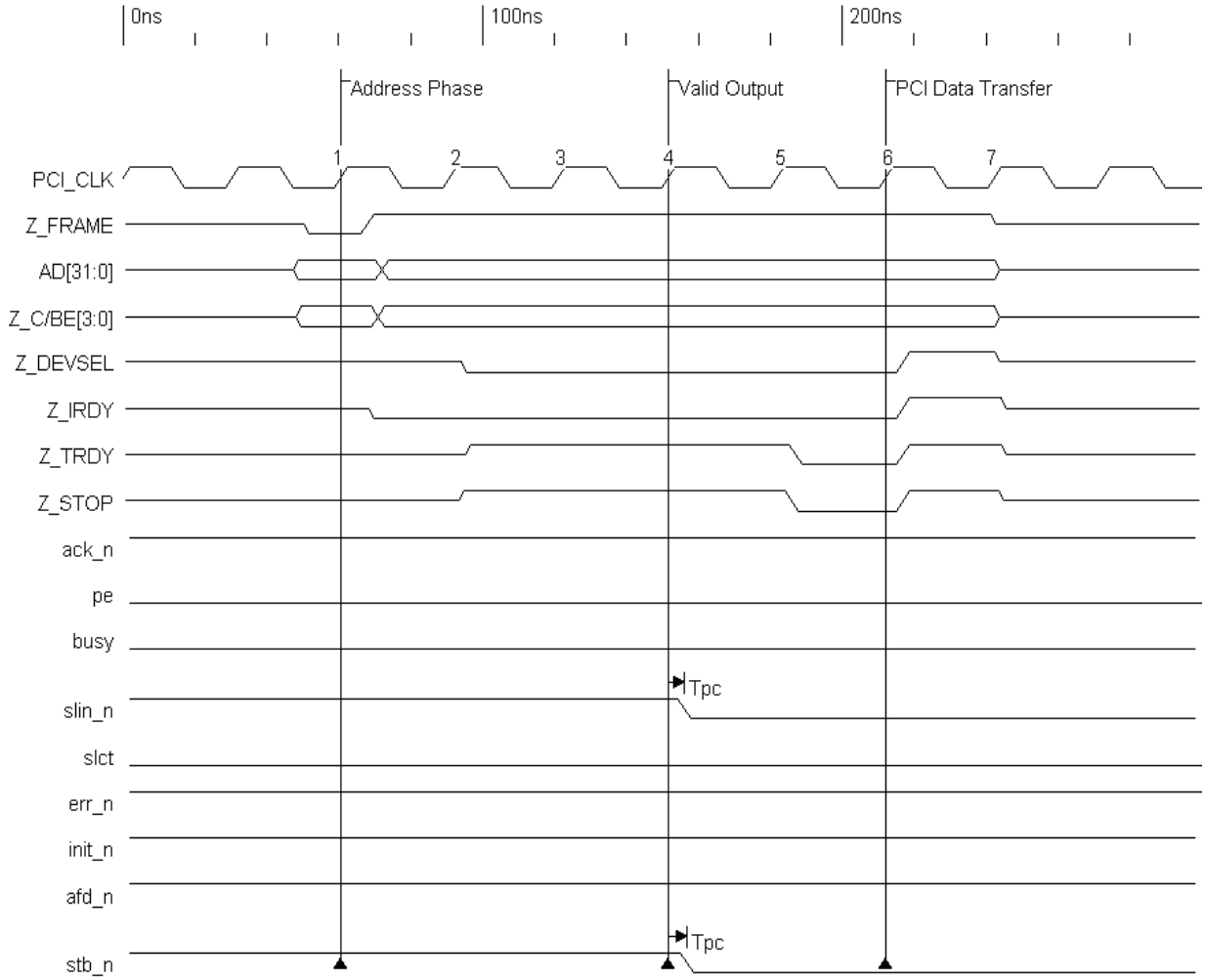
Figure 15 PCI Write to SPP Data Register



Tpd (PCI CLK to valid parallel port data): 15ns max *

* These values are applicable to a pin loading of 100 pF.

Figure 16 PCI Write to SPP DCR Register

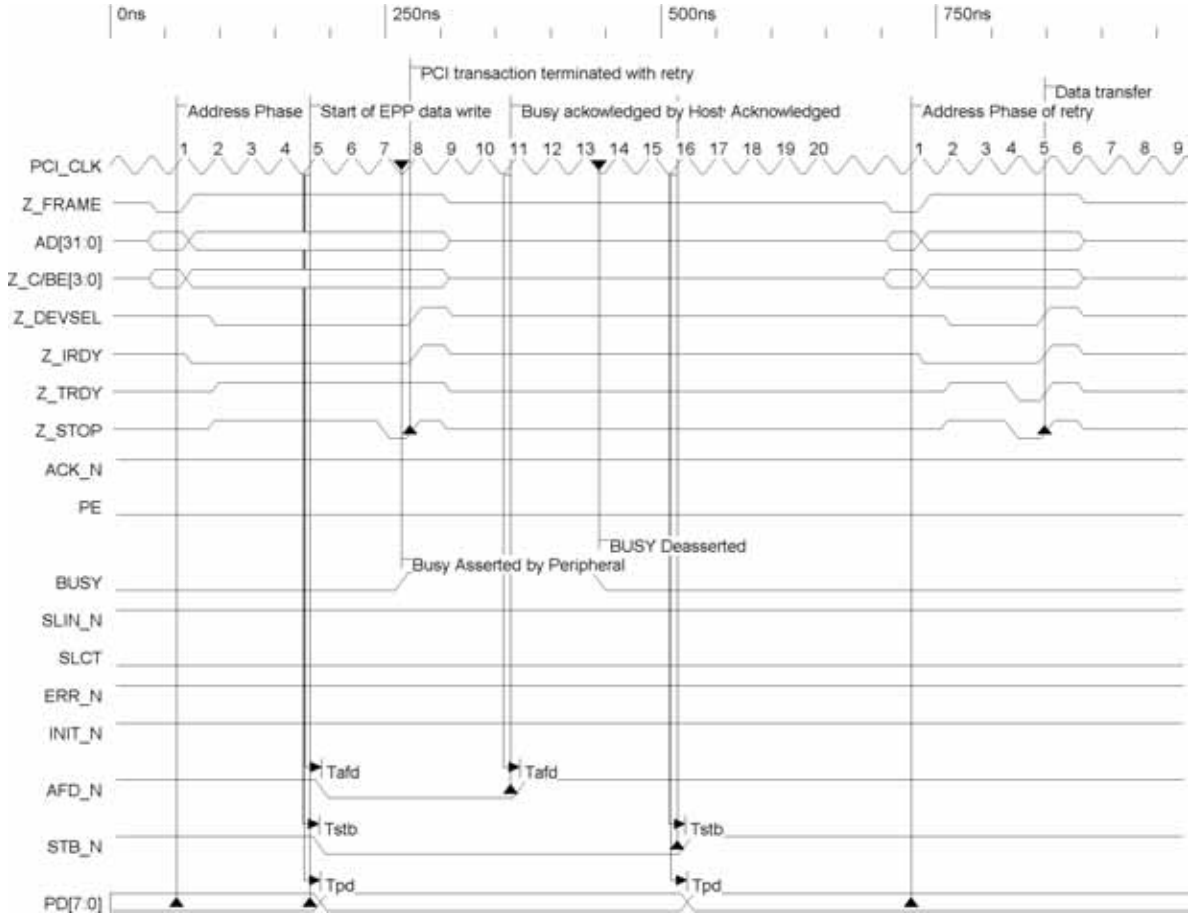


Tpc: PCI CLK to valid parallel port control lines.

- Tpc (Slin_N): 14ns max *
- Tpc (Stb_N): 14ns max *
- Tpc (Init_N): 14ns max *
- Tpc (Afd_N): 14ns max *

* These values are applicable to a pin loading of 50 pF.

Figure 17 Write to EPP Data Register (EPP Write Data Cycle)



PCI CLK no (1st transaction)
1—Start of PCI write to EPP Data Register
5—Start of EPP data write cycle on parallel port side.
8—PCI transaction completes with a 'retry' (without affecting ongoing EPP write data cycle) as EPP cycle cannot complete within 16 PCI CLK cycles
7-8—Peripheral asserts BUSY
11—Host responds to BUSY by de-asserting AFD_N (3 clock cycles after sampling BUSY)
13-14—Peripheral deasserts BUSY
16—Host responds to BUSY by asserting STB_N and the parallel port data lines (2 clock cycles after sampling BUSY).
—EPP cycle completed.

PCI CLK no (retry transaction)
1—Start of retry transaction, to the original write to EPP data register
5—Retry transaction completes with "Data Transfer", without initiating another EPP write data cycle.

Tafd (PCI clk to valid AFD_N): 13ns max *
Tstb (PCI clk to valid STB_N): 15ns max *
Tpd (PCI clk to valid Parallel Data): 15ns max *

* These values are applicable to a pin loading of 50 pF.

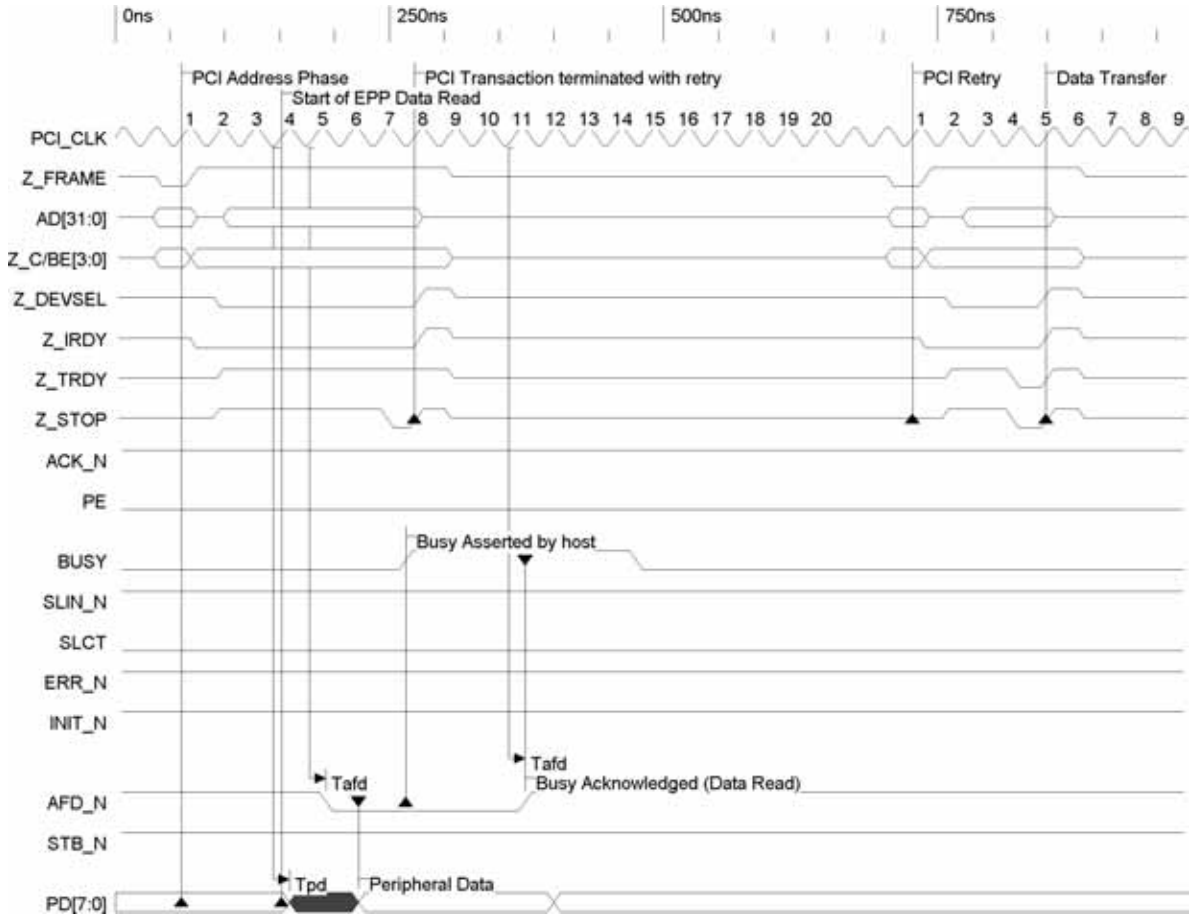
Input set-up of BUSY signal with respect to PCI_CLK: 0ns min
Input HOLD of BUSY signal with respect to PCI_CLK: 2ns min

If the set-up and hold of the BUSY signal is not met, BUSY is registered in the next PCI_CLK cycle.

EPP write data cycle duration is dependent on the timing response of the peripheral's BUSY line and the parallel port filters.

Example waveform has the parallel port filters disabled. An extra 2 PCI CLK cycles are incurred in the response of the host to the peripheral's BUSY line when the filters are enabled.

Figure 18 Read from EPP Data Register (EPP Read Data Cycle)



PCI CLK no (1st transaction)
1—Start of PCI read from EPP Data Register
4—Start of EPP data read cycle on parallel port side.
8—PCI transaction completes with a 'retry' (without affecting ongoing EPP read data cycle) as EPP cycle cannot complete within 16 PCI CLK cycles
7-8—Peripheral asserts BUSY in response to the host driving AFD_N low.
11—Host responds to BUSY by de-asserting AFD_N (3 clock cycles after sampling BUSY)
14-15—Peripheral deasserts BUSY
—EPP cycle completed.

PCI CLK no (retry transaction)

1—Start of retry transaction, to the original read from EPP data register
5—Retry transaction completes with “Data Transfer”, without initiating another EPP read data cycle.

Tafd (PCI clk to valid AFD_N): 13ns max *
Tstb (PCI clk to valid STB_N): N/A as always ‘1’

* These values are applicable to a pin loading of 50 pF.

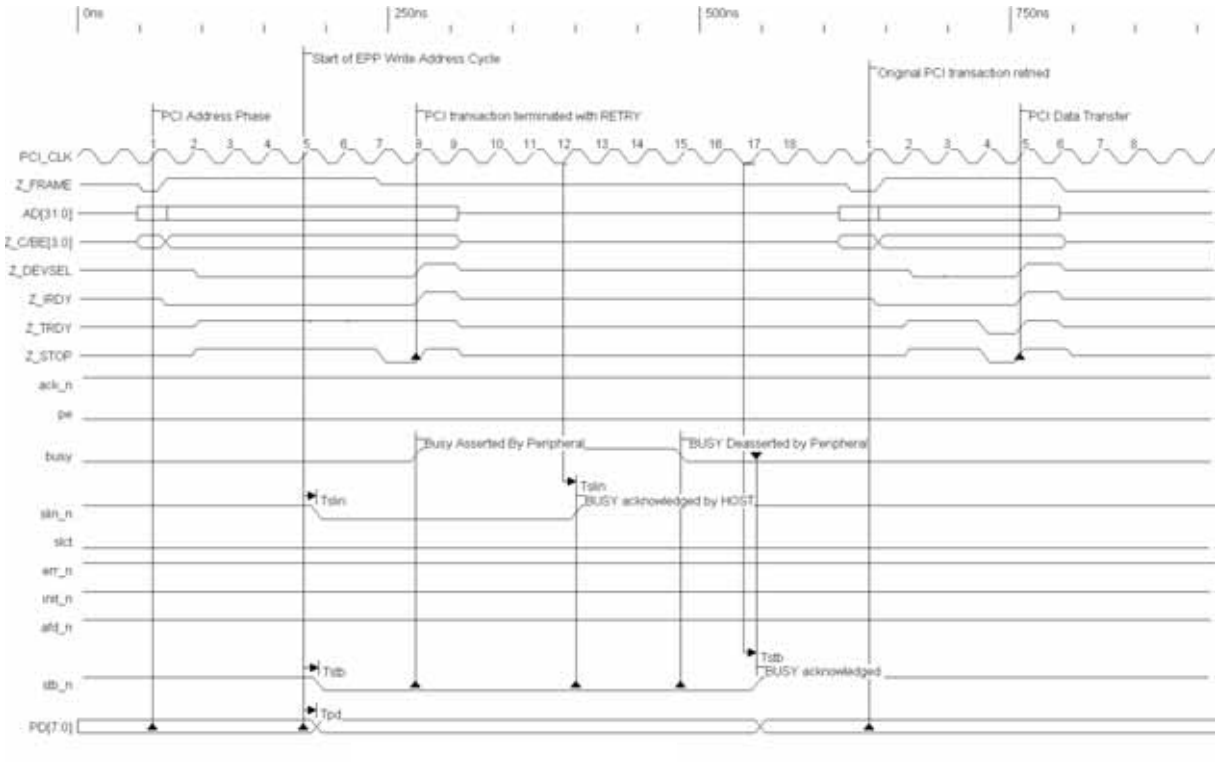
Input set-up of READ data signal with respect to PCI_CLK: 0ns min
Input HOLD of BUSY signal with respect to PCI_CLK: 2ns min
Tpd (PCI_CLK to Valid Parallel Read data: 17ns max

Input set-up of BUSY signal with respect to PCI_CLK: 0ns min
Input HOLD of BUSY signal with respect to PCI_CLK: 2ns min

EPP read data cycle duration is dependent upon the timing response of the peripheral’s BUSY line and the parallel port filters.

Example waveform has the parallel port filters disabled. An extra 2 PCI CLK cycles are incurred in the response of the host to the peripheral’s BUSY line when the filters are enabled.

Figure 19 Write to EPP Address Register (EPP Write Address Cycle)



PCI CLK no (1st transaction)
1—Start of PCI write to EPP Address Register
5—Start of EPP address write cycle on parallel port side.

8—PCI transaction completes with a ‘retry’ (without affecting current EPP write address cycle) as EPP cycle cannot complete within 16 PCI CLK cycles
7-8—Peripheral asserts BUSY
11—Host responds to BUSY by de-asserting SLIN_N (4 clock cycles after sampling BUSY)
14-15—Peripheral deasserts BUSY
17—Host responds to BUSY by asserting SLIN_N and the parallel port data lines (2 clock cycles after sampling BUSY).
—EPP cycle completed.

PCI CLK no (retry transaction)
1—Start of retry transaction, to the original write to EPP address register
5—Retry transaction completes with “Data Transfer”, without initiating another EPP write address cycle.

Tslin (PCI clk to valid SLIN_N): 13ns max *
Tstb (PCI clk to valid STB_N): 15ns max *
Tpd (PCI clk to valid port Address): 15ns max *

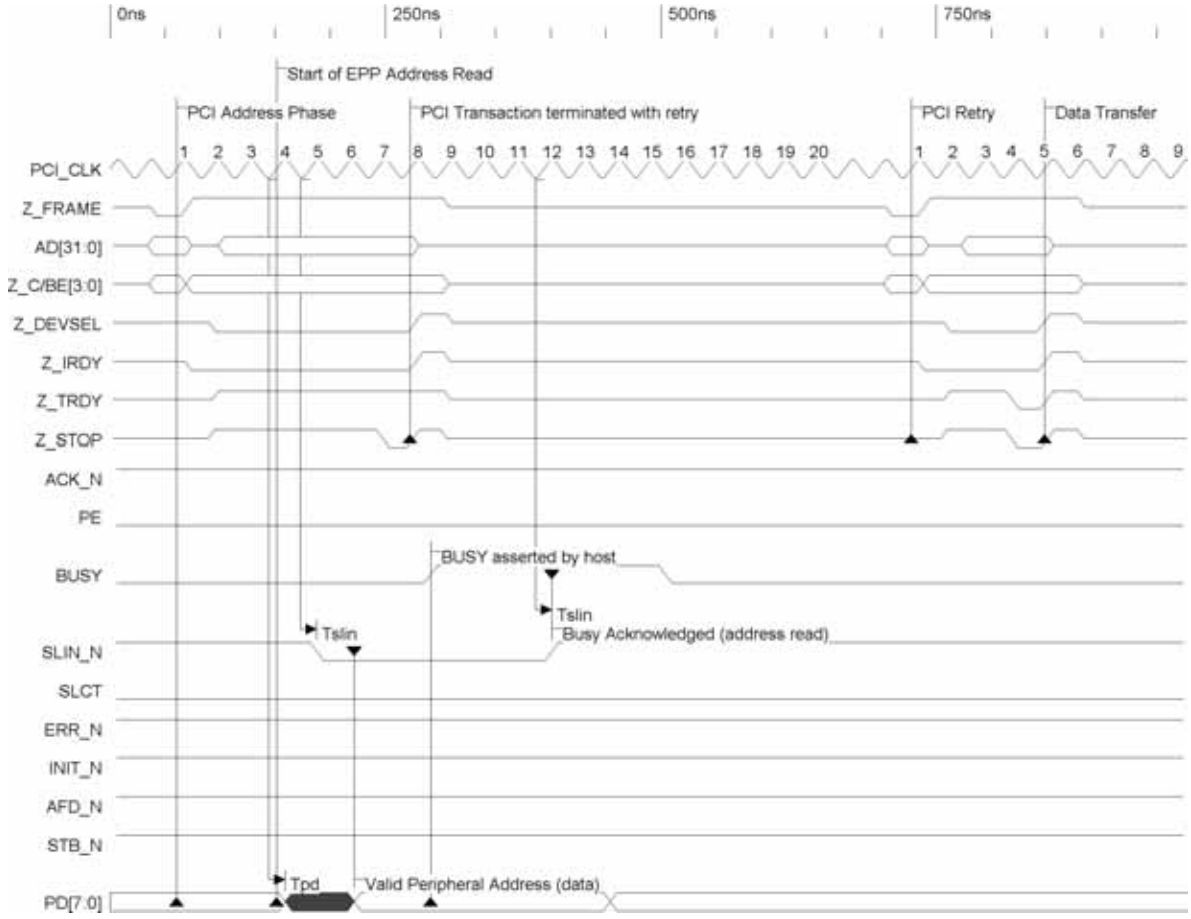
* These values are applicable to a pin loading of 50 pF.

Input set-up of BUSY signal with respect to PCI_CLK: 0ns min
Input HOLD of BUSY signal with respect to PCI_CLK: 2ns min

If the set-up and hold of the BUSY signal is not met, BUSY is registered in the next PCI_CLK cycle.

EPP write address cycle duration is dependent on the timing response of the peripheral’s BUSY line and the parallel port filters. Example waveform has the parallel port filters disabled. An extra 2 PCI CLK cycles are incurred in the response of the host to the peripheral’s BUSY line when the filters are enabled.

Figure 20 Read from EPP Address Register (EPP Read Address Cycle)



PCI CLK no (1st transaction)
1—Start of PCI read from EPP Address Register
4—Start of EPP address read cycle on parallel port side.
8—PCI transaction completes with a 'retry' (without affecting current EPP read address cycle) as EPP cycle cannot complete within 16 PCI CLK cycles
8-9—Peripheral asserts BUSY in response to the host driving SLIN_N low.
12— Host responds to BUSY by de-asserting SLIN_N (3 clock cycles after sampling BUSY)
15-16—Peripheral deasserts BUSY
—EPP cycle completed.

PCI CLK no (retry transaction)
1—Start of retry transaction, to the original read from EPP address register
5—Retry transaction completes with "Data Transfer", without initiating another EPP read address cycle.

Tslin (PCI clk to valid SLIN_N): 13ns max *

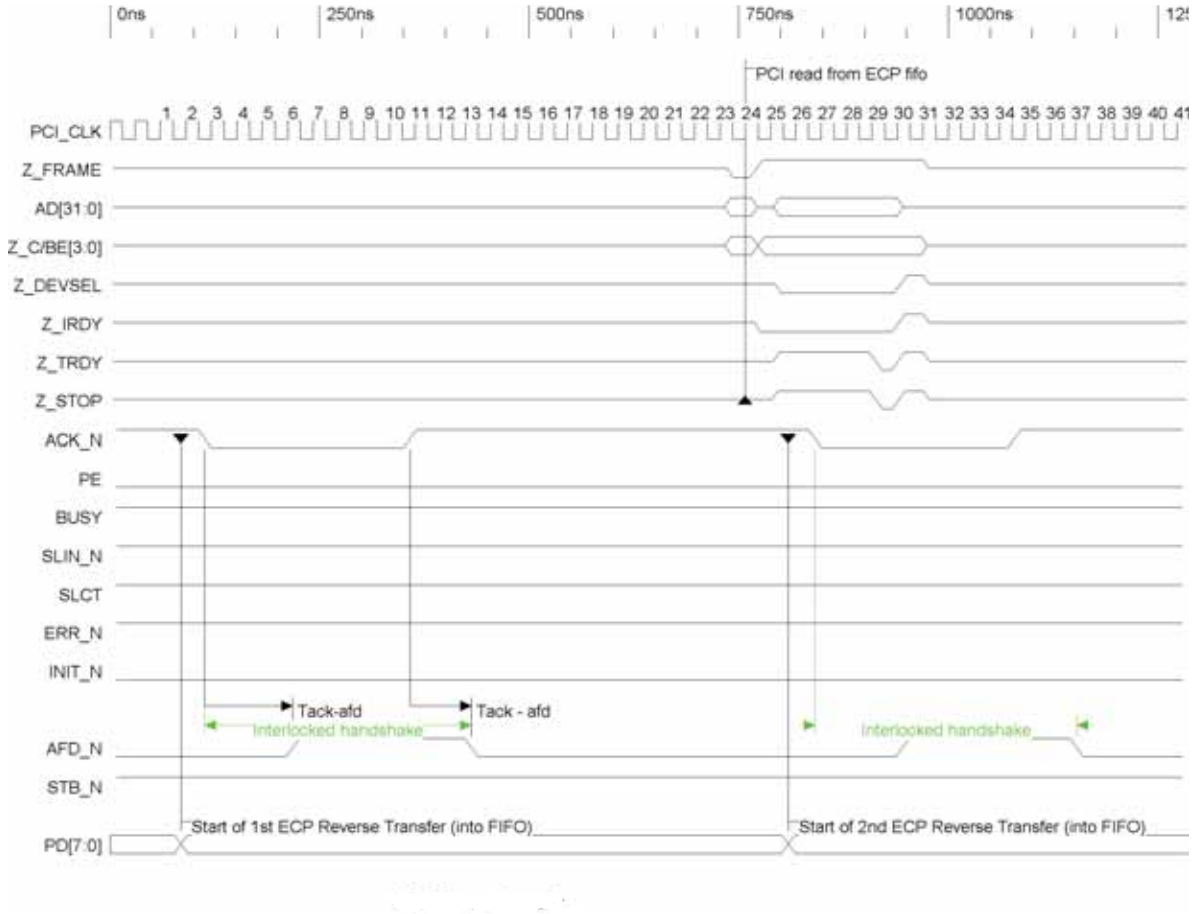
Input set-up of read data (address) with respect to PCI_CLK: 0ns min
Input HOLD of read data (address) with respect to PCI_CLK: 2.5ns min
Tpd (PCI_CLK to Valid port address: 17ns max *

* These values are applicable to a pin loading of 50 pF.

Input set-up of BUSY signal with respect to PCI_CLK: 0ns min
Input HOLD of BUSY signal with respect to PCI_CLK: 2ns min

EPP read address cycle duration is dependent upon the timing response of the peripheral's BUSY line and the parallel port filters. Example waveform has the parallel port filters disabled. An extra 2 PCI CLK cycles are incurred in the response of the host to the peripheral's BUSY line when the filters are enabled.

Figure 21 2 Consecutive Write Transactions to ECP DFIFO



PCI CLK no

1—Start of 1st PCI write to ECP DFIFO Register

6—Start of 1st ECP forward transfer cycle

—1st PCI transaction terminates with a "Data Transfer"

—Peripheral asserts BUSY in response to the host driving STB_N low

—Start of 2nd PCI write to ECP DFIFO Register. Current ECP forward transfer remains unaffected.

—Host responds to BUSY by de-asserting STB_N (2 clock cycles after sampling BUSY) – 1st ECP forward transfer.

—Peripheral deasserts BUSY

17—End of 1st ECP forward transfer (2 clock cycles after sampling BUSY)

18—Start of 2nd ECP forward transfer cycle

Tstb (PCI clk to valid STB_N): 15ns max*
Tpd (PCI clk to valid port data): 15ns max*

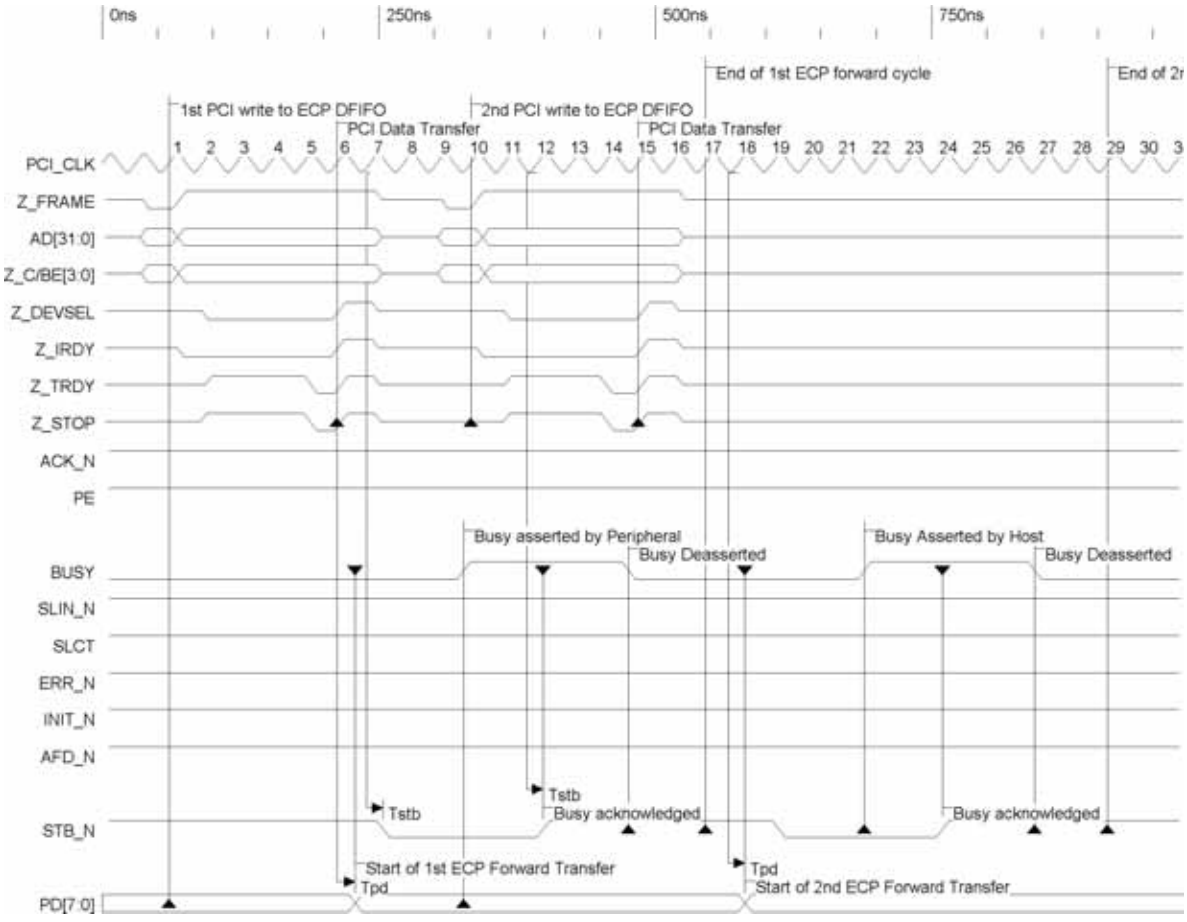
* These values are applicable to a pin loading of 50 pF.

Input set-up of BUSY signal with respect to PCI_CLK: 0ns min
Input HOLD of BUSY signal with respect to PCI_CLK: 1.5ns min

If BUSY set-up and hold is violated, the BUSY signal is registered in the next PCI_CLK cycle in the ECP transaction.

ECP forward transfer cycle duration is dependent on the timing response of the peripheral's BUSY line and the parallel port filters. Example waveform has the parallel port filters disabled. An extra 2 PCI CLK cycles are incurred in the response of the host to the peripheral's BUSY line when the filters are enabled.

Figure 22 ECP Reverse Transfer



PCI CLK no (parallel port already placed in the reverse ECP transfer mode)
2—Start of 1st ECP reverse transfer by peripheral
2-3—Peripheral asserts ACK_N low
6—Host responds by asserting AFD_N (3 clock cycles after sampling ACK_N low)
—Peripheral deasserts ACK_N
13—Host responds by de-asserting AFD_N (2 clock cycles after sampling ACK_N low)
—End of ECP reverse transfer (data already transferred to ECP DFIFO)

24—Start of PCI read from ECP DFIFO (does not initiate or affect any ECP reverse transfers that may be taking place)
26—Start of 2nd ECP reverse transfer by peripheral.

Tafd (PCI clk to AFD_N valid): 15ns max *

* These values are applicable to a pin loading of 50 pF.

Input set-up of ACK_N with respect to PCI_CLK: 0ns min
Input HOLD of ACK_N with respect to PCI_CLK: 2ns min
Input set-up of read data with respect to PCI_CLK: 0ns min
Input HOLD of read data with respect to PCI_CLK: 2ns min

ECP reverse transfer cycle duration is dependent on the timing response of the peripheral's ACK_N line and the parallel port filters. Example waveform has the parallel port filters disabled. An extra 2 PCI CLK cycles are incurred in the response of the host to the peripheral's ACK_N line when the filters are enabled.

Package Information

176-Pin LQFP Package

Figure 23 176-pin Low Profile Quad Flat Pack (LQFP) Package