OX16PCI952 DATA SHEET

Integrated High Performance Dual UARTs, Parallel Port and 5.0v PCI interface

DS_B008A_00

FEATURES

- Two 16C950 High performance UART channels
- IEEE1284 Compliant SPP/EPP/ECP parallel port
- Multi-function target PCI controller. Fully compliant to PCI bus specification 2.2 and PCI Power Management 1.0.
- Function access to pre-configure each UART and the parallel port, prior to handover to generic device drivers.
- UARTs fully software compatible with 16C550-type devices.
- Baud rates up to 15Mbps in asynchronous mode and 60Mbps in external 1x clock 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#
- 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
- 5, 6, 7, 8 and 9-bits data framing
- Global Interrupt Status and readable FIFO levels to facilitate implementation of efficient device drivers
- Detection of bad data in the receiver FIFO
- 2 multi-purpose IO pins which can be configured as interrupt inputs or 'wake-up' pins (via local registers).
- Auto-detection of a range of optional Microwire™ compatible EEPROMs, to reconfigure device.
- Operation via IO or memory mapping.
- 5.0V operation
- 128 pin TQFP package

DESCRIPTION

The OX16PCI952 is a single chip solution for PCI-based serial and parallel expansion add-in cards. It is a dual function device, offering IO or memory mapped access to the two ultra-high performance OX16C950 UARTs and the bi-directional parallel port. These functions are defined by Function 0 and Function 1, respectively. Serial port cards with 2 serial ports and a parallel port can be designed without redefining any device parameters.

Each UART channel in the OX16PCI952, is the fastest available PC-compatible UART, offering data rates up to 15Mbps and 128-byte deep transmitter and receiver FIFOs. The deep FIFOs reduce CPU overhead and allow utilisation 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, etc.

The parallel port is an IEEE 1284 compliant SPP, EPP and ECP parallel port that fully supports the existing Centronics interface. For legacy applications, the PCI resources have been arranged so that the parallel port can be located at standard I/O addresses

A set of local registers is available to enhance device driver efficiency and reduce interrupt latency. Each internal UART has features such as shadowed FIFO fill levels, an interrupt source register and Good-Data Status, readable in consecutive DWORD registers and is visible to logical function0 in both IO space and memory space. The local registers also provide additional controls for each UART and the parallel port, to customise the device for the end-users application.

The efficient 32-bit, 33MHz target-only interface is compliant with the PCI bus specification version 2.2 and version 1.0 of PCI Power Management Specification.

For full flexibility, all the default configuration register values can be overwritten using an optional Microwire™ compatible serial EEPROM.

This EEPROM can also be used to provide *function access* to pre-configure each UART into enhanced modes or pre-configure the parallel port, prior to any PCI configuration accesses and before control is handed to generic device drivers.

Microwire™ is a trade mark of National Semiconductor.

Oxford Semiconductor Ltd.
25 Milton Park, Abingdon, Oxon, OX14 4SH, UK
Tel: +44 (0)1235 824900 Fax: +44(0)1235 821141

Ó Oxford Semiconductor 2001
OX16PCI952 Datasheet rev 1.1 – June 2001
Part No. OX16PCI952-TQFP-A

CONTENTS

This page is intentionally left blank

1 PERFORMANCE COMPARISON

Feature	OX16PCI952	16C552 + PCI Bridge	16C652 + PCI Bridge
Internal serial channels	2	0	0
Integral 1284 Compliant parallel port	yes	no	no
Multi-function PCI device	yes	no	no
Support for PCI Power Management	yes	no	no
Zero wait-state read/write operation	yes	no	no
No. of available external interrupt pins	2	2	2
DWORD access to UART Interrupt Source Registers & FIFO Levels	yes	no	no
Good-Data status	yes	no	no
Full Plug and Play with external EEPROM	yes	yes	yes
External 1x baud rate clock	yes	no	no
Max baud rate in normal mode	15 Mbps	115 Kbps	1.5 Mbps
Max baud rate in 1x clock mode	60 Mbps	n/a	n/a
FIFO depth	128	16	64
Sleep mode	yes	no	yes
Auto Xon/Xoff flow	yes	no	yes
Auto CTS#/RTS# flow	yes	no	yes
Auto DSR#/DTR# flow	yes	no	no
No. of Rx interrupt thresholds	128	4	4
No. of Tx interrupt thresholds	128	1	4
No. of flow control thresholds	128	n/a	4
Transmitter empty interrupt	yes	no	no
Readable status of flow control	yes	no	no
Readable FIFO levels	yes	no	no
Clock prescaler options	248	n/a	2
Rx/Tx disable	yes	no	no
Software reset	yes	no	no
Device ID	yes	no	no
9-bit data frames	yes	no	no
RS485 buffer enable	yes	no	no
Infra-red (IrDA)	yes	no	yes

Table 1: OX16PCI952 performance compared with PCI Bridge + generic UART/Parallel Port Combinations.

1.1 Improvements of the OX16PCI952 over discrete solutions:

Higher degree of integration:
The OX16PCI952 offers two internal ultra-high performance 16C950 UARTs and one IEEE1284 compliant bi-directional parallel port.

UART device driver efficiency is increased by using each channel's features such as the 128-byte deep transmitter & receiver FIFOs, flexible clock options, automatic flow control, programmable interrupt and flow control trigger levels and readable FIFO levels. Data rates of each UART is up to 60Mbps.

Improved access timing:
Access to the internal UARTs require zero or one PCI wait states. 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 OX16PCI952 offers shadowed FIFO levels and Interrupt status registers of the internal UARTs, as well as general device interrupt status, to reduce the device driver interrupt latency.

Power management:
Both functions of the OX16PCI952 comply with the PCI Power Management Specification 1.0 and the PC98/99 Power Management specifications, by offering the extended capabilities for Power Management and supporting the power states D0, D2 and D3. This achieves significant power savings by allowing device drivers to power down the PCI functions and disable the UART channels and the parallel port.

A 'wake-up' event (the 'power management event') is requested via the PME# pin from either of the power states D2 or D3. For the UART channels, this wake-up request is generated through the UART line RI (for power state D3), and any modem line and the Serial Data In (for power state D2). For the parallel port, this wake-up request is generated through the multi-purpose IO pins, MIO(1:0).

Optional EEPROM:
The OX16PCI952 can be reconfigured from an external Microwire™ based EEPROM. However, this is not required in many applications as default values are provided for typical applications. Features available via the use of the EEPROM include redefining device ID's and vendor/sub-vendor ID fields in the PCI header space, selectively enabling/disabling interrupts, powerdown and wakeup requests, and performing function access to pre-configure the UARTs and the parallel port.

Multi-function device:
The OX16PCI952 is a multi-function device to enable users to load individual device drivers for the internal serial ports and the internal parallel port.

2 BLOCK DIAGRAM

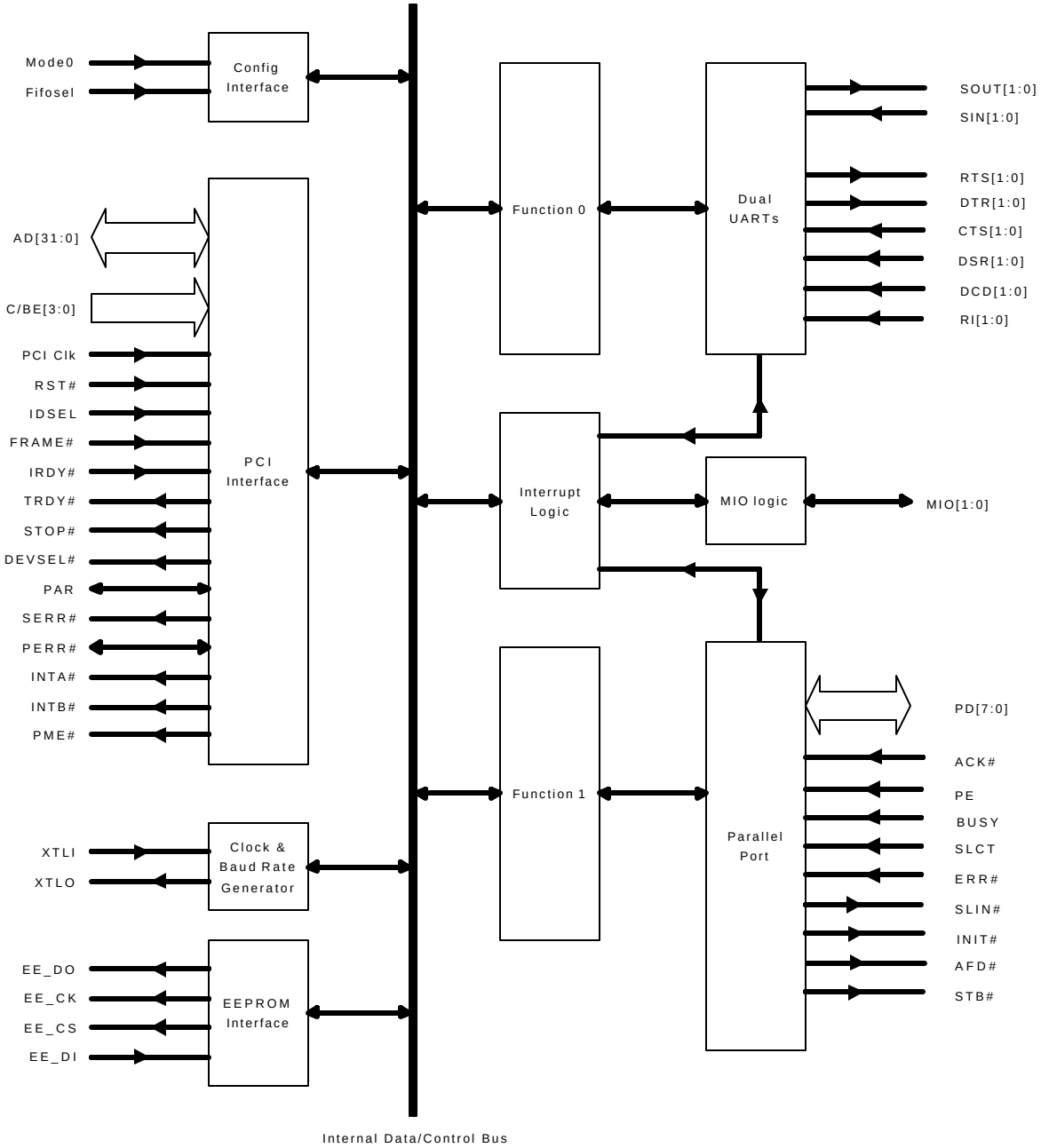
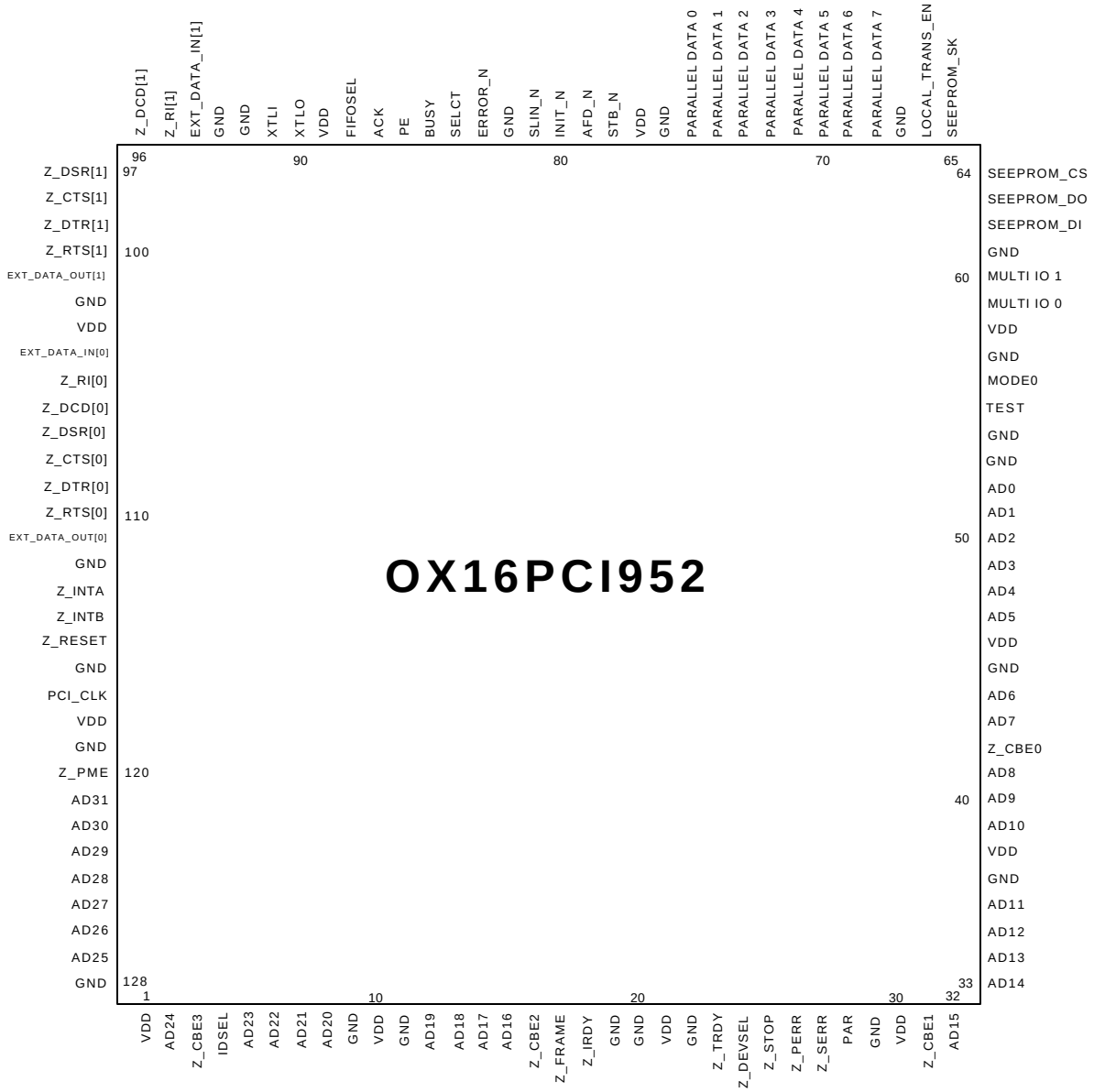


Figure 1: OX16PCI952 Block Diagram

3 PIN INFORMATION

Package : 128 TQFP. (14mm x 14mm)



4 PIN DESCRIPTION

Please refer to Section "Pin Information" for actual Signal Name to Pin Number assignments.

Pins	Dir[1]	Name	Description
PCI interface			
2, 5, 6, 7, 8, 12, 13, 14, 15, 32, 33, 34, 35, 36, 39, 40, 41, 43, 44, 47, 48, 49, 50, 51, 52, 121, 122, 123, 124, 125, 126, 127	P_I/O	AD[31:0]	Multiplexed PCI Address/Data bus
3, 16, 31, 42	P_I	C/BE[3:0]#	PCI Command/Byte enable
117	P_I	PCI CLK	PCI system clock
17	P_I	FRAME#	Cycle Frame
24	P_O	DEVSEL#	Device Select
18	P_I	IRDY#	Initiator ready
23	P_O	TRDY#	Target ready
25	P_O	STOP#	Target Stop request
28	P_I/O	PAR	Parity
27	P_O	SERR#	System error
26	P_I/O	PERR#	Parity error
4	P_I	IDSEL	Initialization device select
115	P_I	RST#	PCI system reset
113 114	P_OD	INTA#, INTB#	PCI interrupts
120	P_OD	PME#	Power management event
Serial port pins			
88	I	FIFOSEL	FIFO select. For backward compatibility with 16C550, 16C650 and 16C750 devices the FIFO depth of both UARTs is 16 when FIFOSEL is low. The FIFO size of both UARTs is increased to 128 when FIFOSEL is high. The FIFO size of each UART may also be set to 128 by setting the UARTs FCR[5] when LCR[7] is set, or by putting the device into enhanced mode. The unlatched state of this pin is readable by software.
111 101 111 101	O O O O	EXT_DATA_OUT[0] EXT_DATA_OUT[1] IrDA_Out[0] IrDA_Out[1]	Serial data output, Uart 0 Serial data output, Uart 1. UART IrDA data outputs, for UART 0 and 1. Serial data output pins redefined as IrDA data outputs when MCR[6] of the corresponding UART channel is set in enhanced mode
104 94 104 94	I I I I	EXT_DATA_IN[0] EXT_DATA_IN[1] IrDA_In[0] IrDA_In[1]	Serial data input, UART 0. Serial data input, UART 1. UART IrDA data inputs, for UART 0 and 1. Serial data input pins redefined as IrDA data inputs when MCR[6] of the corresponding UART channel is set in enhanced mode

Pins	Dir[1]	Name	Description
Serial port pins (Contd)			
106 96	I I	DCD[0]# DCD[1]#	Active-low modem "data-carrier-detect" input, for UART 0 and UART 1.
109 99	O O	DTR[0]# DTR[1]#	Active-low modem "data-terminal-ready output", for UART 0 and UART 1. If automated DTR# flow control is enabled for the corresponding UART channel, the DTR# pin is asserted and deasserted if the receiver FIFO reaches or falls below the channel's programmed thresholds, respectively.
109 99	O O	485_En[0] 485_En[1]	In RS485 half-duplex mode, the DTR# pin of each UART channel may be programmed to reflect the state of the channel's transmitter empty bit to automatically control the direction of the RS485 transceiver buffer (see register ACR[4:3])
109 99	O O	Tx_Clk_Out[0] Tx_Clk_Out[1]	Transmitter 1x clock (baud rate generator output). For isochronous applications, the 1x (or Nx) transmitter clock of each UART channel may be asserted on the DTR# pins (see register CKS[5:4])
110 100	O O	RTS[0]# RTS[1]#	Active-low modem "request-to-send" output, for UART 0 and UART 1. If automated RTS# flow control is enabled for the corresponding UART channel, the RTS# pin is deasserted and reasserted whenever the receiver FIFO reaches or falls below the programmed thresholds, respectively.
108 98	I I	CTS[0]# CTS[1]#	Active-low modem "clear-to-send" input, for UART 0 and UART 1. If automated CTS# flow control is enabled for the corresponding UART channel, upon deassertion of the CTS# pin, the channel's transmitter will complete the current character and enter the idle mode until the CTS# pin is reasserted. Note: flow control characters are transmitted regardless of the state of the CTS# pin.
107 97	I I	DSR[0]# DSR[1]#	Active-low modem "data-set-ready" input, for UART 0 and UART 1. If automated DSR# flow control is enabled for the corresponding UART channel, upon deassertion of the channel's DSR# pin, the transmitter will complete the current character and enter the idle mode until the DSR# pin is reasserted. Note: flow control characters are transmitted regardless of the state of the DSR# pin
107 97	I I	Rx_Clk_In[0] Rx_Clk_In[0]	External receiver clock input, for isochronous applications. The DSR Uart pins are redefined as Rx_Clk_In, when the corresponding UART channel's CKS[1:0] register bits = '01'.
105 95	I I	RI[0]# RI[1]#	Active-low modem "Ring-Indicator" input, for UART 0 and UART 1.
105 95	I I	Tx_Clk_In[0] Tx_Clk_In[0]	External transmitter clock. The RI Uart pins are redefined as transmitter clk pins (and thus used indirectly by the receiver) when the UART channel's CKS[6] register bit ='1'.

Pins	Dir[1]	Name	Description
Serial port pins (Contd)			
91	XI	XTLI	Crystal oscillator input or external clock pin, for the UART channels. Crystal oscillator frequency maximum 40MHz. Maximum frequency 60MHz, via external clock source.
90	XO	XTLO	Crystal oscillator output

Pins	Dir[1]	Name	Description
Parallel port			
87	I	ACK#	Acknowledge Signal in SPP mode. ACK# is asserted (low) by the peripheral to indicate that a successful data transfer has taken place.
87	I	INTR#	INTR# pin in EPP mode. Function identical to ACK#.
86	I	PE	Paper Empty. Activated by printer when it runs out of paper.
85	I	BUSY	Busy Signal in SPP mode. BUSY is asserted (high) by the peripheral when it is not ready to accept data
85	I	WAIT#	Wait signal in EPP mode. Handshake signal for interlocked IEEE 1284 compliant EPP cycles.
81	OD [2]	SLIN#	Select signal in SPP mode. Asserted by host to select the peripheral.
81	O	ADDRSTB#	Address strobe in EPP mode. Provides address read and write strobe.
84	I	SLCT	Peripheral selected. Asserted by peripheral when selected.
83	I	ERR#	Error. Held low by the peripheral during an error condition.
80	OD [2]	INIT#	Initialize signal in SPP mode. Commands the peripheral to initialize.
80	O	INIT#	Initialize signal in EPP mode. Function identical to SPP.
79	OD [2]	AFD#	Auto Feed signal in SPP mode.
79	O	DATASTB#	Data strobe signal in EPP mode. Provides data read and write strobe.
78	OD [2]	STB#	Strobe signal in SPP mode. Used by the peripheral to latch data currently available on the PD[7:0] lines.
78	O	WRITE#	Write signal in EPP mode. Indicates a write cycle when low and a read cycle when high.
68, 69, 70, 71, 72, 73, 74, 75	I/O [2]	PD[7:0]	Parallel port bi-directional data bus
66	O	LOCAL_TRANS_EN	Parallel Port Data Output Enable. This pin can be used by external transceivers. It is high when PD[7:0] are in output mode, and low when they are in input mode.

Pins	Dir[1]	Name	Description
Multi-purpose & External interrupt pins			
59	I/O	MIO 0	Multi-purpose I/O 0. Can be driven high or low, or be used to assert PCI interrupts or power management events (PME).
60	I/O	MIO 1	Multi-purpose I/O 1. Can be driven high or low, or be used to assert PCI interrupts or power management events (PME).
Microwire EEPROM pins			
65	O	EE_CK	EEPROM clock signal
64	O	EE_CS	EEPROM active-high Chip Select Signal
62	IU	EE_DI	EEPROM data in (To be connected to the external EEPROM's DO pin). When the optional serial EEPROM is connected, this pin should be pulled up using an external 1-10k resistor. When the external EEPROM is not required, this external pull-up is not necessary as the internal pull-up is sufficient.
63	O	EE_DO	EEPROM data out. (To be connected to the external EEPROM's DI pin)
Miscellaneous pins			
55	ID	TEST	TESTPIN. This must be connected to GND.
56	I	MODE 0	MODE selector. MODE0 = 0. Device operates as a dual function device, where function 0 is the Dual UARTs and Function 1 is the parallel port. MODE0 = 1. Device operates only as a single function device, where function 0 is the Dual UARTs. Function 1 does not exist, so the parallel port is not visible[2] to PCI accesses.
Power and ground[3]			
1, 10, 21, 30, 38, 46, 58, 77, 89, 103, 118	V	VDD	Device Power
9, 11, 19, 20, 22, 29, 37, 45, 53, 54, 57, 61, 67, 76, 82, 92, 93, 102, 112, 116, 119, 128	G	GND	Device Gnd.

Table 2: Pin Descriptions

Note 1: Direction key:

I	5v TTL Input
ID	5v TTL Input, with internal pull-down
IU	5v TTL input, with internal pull-up
I/O	5v Tristate Bi-directional
O	5v Output
OD	5v Open drain
X1	Crystal Oscillator (or clock) input
XO	Crystal Oscillator output
P_I	5v PCI input
P_I/O	5v PCI bi-directional
P_O	5v PCI output
P_OD	5v PCI open drain
G	Ground
V	5.0V power

Note 2 : Parallel Port pins when MODE0 = 1

When the device operates as a single function device (MODE0 pin = '1'), in addition to function 1 (the parallel port) not being made available to PCI configuration accesses, all of the parallel port bi-directional pins and open-drain output pins are forced into their input modes. This means that when MODE0 = 1, as well as the standard parallel port inputs needing a tie to ground to prevent floating inputs, all of the bi-directional and open-drain parallel port pins also require a tie to ground. This affects all of the pins of the parallel port with the exception of the pin "local_trans_en" that remains as an output.

Note 3: Power & Ground

There are several types of VDD and VSS in this design, providing not only power for the internal (core) and I/O pad area but also special power lines to the PCI I/O buffers. These power rails are not connected internally. This precaution reduces the effects of simultaneous switching outputs and undesirable RF radiation from the chip. Further precaution is taken by segmenting the GND and VDD rails to isolate the PCI and UART pins.

5 CONFIGURATION & OPERATION

The OX16PCI952 is a multi-function, target-only PCI device, compliant with the PCI Local Bus Specification, Revision 2.2 and PCI Power Management Specification, Revision 1.0.

The OX16PCI952 affords maximum configuration flexibility by treating the internal UARTs and the parallel port as separate logical functions (function 0 and function 1, respectively). Each function has its own PCI configuration space and is therefore recognised and configured by the PCI BIOS separately (each function operates as though it were a separate device). The device can also be configured to operate as a single function device by making available only the internal UARTs. This is controlled by the Mode pin, as shown in Table 3.

The OX16PCI952 is configured by system start-up software during the bootstrap process that follows bus reset. The system scans the PCI bus and reads the vendor and device identification codes from any devices/functions it finds and the resources being requested. It then loads the 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.

A set of *local configuration registers* have been provided that can be used to control the device's characteristics (such as interrupt handling) and report internal functional status. This is on top of the UART and the Parallel Port registers, and the registers contained in each of the 2 function's PCI configuration Space. These local registers are common to both functions and can be set up by the device drivers of function 0 and function 1, or from the optional EEPROM.

The EEPROM can also be used to redefine the reset values of most register areas to tailor the device to the end users requirements if the default values do not meet the specific requirements of the manufacturer, such as the identification registers. As an additional enhancement, the EEPROM can be used to pre-program each UART and the Parallel Port, allowing pre-configuration, without requiring device driver changes. This, for example, does allow the enhanced features of the integrated UARTs to be in place prior to handover to any generic device drivers. A similar arrangement is available for the parallel port.

Mode	Configuration
0	Dual Function Device Mode. Function 0 is the Dual UART. Function 1 is the parallel port
1	Single Function Device Mode. Function 0 is the Dual UART. Function 1 (Parallel Port) is not available.

Table 3: Mode configuration

6 PCI TARGET CONTROLLER

6.1 Operation

The OX16PCI952 responds to the following PCI transactions:-

- *Configuration access*: The OX16PCI952 responds to type 0 configuration reads and writes if the IDSEL signal is asserted and the bus address selects the configuration registers for function 0 or 1. The device will respond to these configuration transactions by asserting DEVSEL#. Data transfer then follows. Any other configuration transaction will be ignored by the OX16PCI952.

- *IO reads/writes*: The address is compared with the addresses reserved in the I/O Base Address Registers (BARs), of each available function. If the address falls within one of the assigned ranges, the device will respond to the IO transaction by asserting DEVSEL#. Data transfer follows this address phase. For all modes, only byte accesses are possible to the function BARs (excluding the local configuration registers for which WORD, DWORD access is supported). For IO 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 will have no effect.

- *Memory reads/writes*: These are treated in the same way as I/O transactions, except that the memory ranges are used. With the exception of Memory accesses to the local configuration registers, memory access to single-byte regions such as the UART and parallel port registers is always expanded to DWORDs in the OX16PCI952. In other words, the OX16PCI952 reserves a DWORD per byte in single-byte regions. The device allows the user 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 OX16PCI952 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 will complete normally on the PCI bus, but it will have no effect on the UART or the Parallel Port (if available).

- All other cycles (64-bit, special cycles, reserved encoding etc.) are ignored.

The OX16PCI952 will complete all transactions as disconnect-with-data, ie the device will assert 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 will be signalled in response to any access while the OX16PCI952 is reading from the serial EEPROM.

The OX16PCI952 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. Fast back-to-back transactions (to both functions) are supported by the OX16PCI952 as a target, so a bus master can perform faster sequences of write transactions (when an inter-frame turn-around cycle is not required) to the UARTs, the Parallel Port, the PCI configuration space and the local configuration registers. The internal UARTs are accessed with zero wait states inserted.

The device supports any combination of byte-enables for accesses 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 OX16PCI952 performs parity generation and checking on all PCI bus transactions as defined by the PCI local Bus 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 will report 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 will still complete the transaction as though the parity check was correct.

The OX16PCI952 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, data in the device cannot be pre-fetched because there may be side-effects on reads.

6.2 Configuration space

The OX16PCI952 is a dual-function (or single function) device, where each logical function has its own PCI configuration space.

All the required fields in the predefined PCI header region have been implemented. The device dependant region of the PCI configuration space contains the PCI Power Management Extended Capability register set.

The format of the configuration space, for both function 0 and function 1, is as shown in Table 4, below.

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

6.2.1 PCI Configuration Space Register map

Configuration Register Description 31	16	15	0	Offset Address
Device ID		Vendor ID		00h
Status		Command		04h
Class Code			Revision ID	08h
BIST[1]	Header Type	Reserved	Reserved	0Ch
Base Address Register 0 (BAR 0) *Function 0 – UART0 in I/O Space* *Function 1 – Parallel Port Base Register Set in I/O Space*				10h
Base Address Register 1 (BAR 1) *Function 0 – UART1 I/O Space* *Function 1 – Parallel Port Extended Register Set in I/O space*				14h
Base Address Register 2 (BAR 2) Local Configuration Registers in IO space, for function 0 and function 1				18h
Base Address Register 3 (BAR 3) Local Configuration Registers in Memory space, for function 0 and function 1				1Ch
Base Address Register 4 (BAR 4) *Function 0 – UART0 and UART1 in Memory Space* *Function 1 – Not Implemented.*				20h
Base Address Register 5 (BAR5) *Not Implemented, for function 0 and function 1*				24h
Reserved				28h
Subsystem ID		Subsystem Vendor ID		2Ch
Reserved				30h
Reserved			Cap_Ptr	34h
Reserved				38h
Reserved	Reserved	Interrupt Pin	Interrupt Line	3Ch

Predefined PCI Header Region

Device Dependent PCI Region

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

Table 4: PCI Configuration space (for UART and Parallel Port function)

Register name	Reset value		Program read/write	
	Function 0	Function 1[1]	EEPROM	PCI
	Dual UART	Parallel Port		
Vendor ID	0x1415	0x1415	W	R
Device ID	0x9521	0x9523	W	R
Command	0x0000	0x0000	-	R/W
Status	0x0290	0x0290	W (bit 4)	R/W
Revision ID	0x00	0x00	-	R
Class code	0x070006	0x070101	W	R
Header type	0x80	0x80	W (bit 7)	R
BAR 0	0x00000001	0x00000001	-	R/W
BAR 1	0x00000001	0x00000001	-	R/W
BAR 2	0x00000001	0x00000001	-	R/W
BAR 3	0x00000000	0x00000000	-	R/W
BAR 4	0x00000000	-	-	R/W
Subsystem VID	0x1415	0x1415	W	R
Subsystem ID	0x0001	0x0001	W	R
Cap ptr.	0x40	0x40	-	R
Interrupt line	0x00	0x00	-	R/W
Interrupt pin	0x01[2]	0x01[2]	W	R
Cap ID	0x01	0x01	-	R
Next ptr.	0x00	0x00	-	R
PM capabilities	0x6C01	0x6C01	W	R
PMC control/ status register	0x0000	0x0000	-	R/W

Table 5: PCI configuration space Reset Values

NOTE 1: Function 1 PCI Configuration Space is available only when the OX16PCI952 is operating in the dual-function device mode (MODE0 pin = '0'). Configuration accesses to Function 1 in the single function device mode (MODE0 pin = '1') will result in 'Master-Aborts'.

NOTE 2 : Default Interrupt pins values for function 0 and function 1 result in both functions asserting interrupts on the pin INTA#. The default values can be re-programmed by the serial controller so that the 2 functions can assert interrupts onto separate (function dependent) interrupt pins.

6.3 Accessing Function 0 and Function 1

Access to the internal UARTs and the Parallel Port is achieved (at addresses defined by the Base Address Registers in the PCI configuration space) via standard I/O and memory mapping. These 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 base addresses that have been allocated can then be used to access the functions. The mapping of these BARs is shown inTable 6.

BAR	Function 0	Function 1[1]
0	Internal UART 0 (I/O Mapped)	Parallel Port Base Registers (I/O Mapped)
1	Internal UART 1 (I/O Mapped)	Parallel Port Extended Registers (I/O Mapped)
2	Local configuration registers (I/O Mapped)	
3	Local configuration registers (Memory Mapped)	
4	Internal UART 0 and UART 1 (Memory Mapped)	Not Implemented
5	Not Implemented	

Table 6: Base Address Register definition
NOTE 1. Function 1 is only accessible in the Dual Function mode (MODE0 = '0')

6.3.1 PCI access to the internal UARTs

IO and memory space

BAR 0, BAR 1, and BAR 4 of function 0 are used to access the internal UARTs through I/O and Memory transactions. The function reserves 8-byte blocks of I/O space for each UART (total of 16-bytes) and a 4K byte block of memory space for both UARTs.

Once the I/O and/or the Memory access enable bits in the Command register (of this function's PCI configuration space) are set, the internal UARTs can be accessed using the mappings shown in the following tables.

UART 0 Address	PCI Offset from Base Address 0 , for UART 0 in I/O space
000	00
001	01
002	02
003	03
004	04
005	05
006	06
007	07

UART 1 Address	PCI Offset from Base Address 1, for UART 1 in I/O space
000	00
001	01
002	02
003	03
004	04
005	05
006	06
007	07

Base Address mapping for UART0 and UART 1 registers, for I/O accesses.

UART Address	PCI Offset from Base Address 4, for UART0 and UART 1 in Memory space (hex)	
	UART 0	UART 1
000	00	20
001	04	24
002	08	28
003	0C	2C
004	10	30
005	14	34
006	18	38
007	1C	3C

Base Address mapping for UART 0 and UART1 registers, for memory accesses

Note 1: Since 4K of memory space is reserved to map both UARTs and the full bus address is not used for decoding, there are a number of aliases of the UARTs in the allocated memory region

6.3.2 PCI access to parallel port

IO space

When the dual-mode of the device is utilised, then BAR 0 and BAR 1 of function 1 are used to access the internal Parallel Port through I/O transactions. *Memory accesses to the parallel port are not possible.*

These I/O BARs correspond to the two sets of registers defined to operate an IEEE1284 EPP and bi-directional Parallel Port. BAR0 reserves an 8-byte block of I/O space and BAR1 reserves a 4 byte-block of I/O space.

Once the I/O access enable bits in the Command register (of this function's PCI configuration space) are set, the internal parallel port can be accessed using the mappings shown in the following table. *See section "Bi-Directional Parallel Port" for more details.*

Register Name	BAR 0 I/O Address Offset, in Function 1
PDR	000h
ecpAFifo	000h
DSR (EPP mode) (Other modes)	001h 001h
DCR	002h
EPPA	003h
EPPD1	004h
EPPD2	005h
EPPD3	006h
EPPD4	007h

Register Name	BAR1 I/O Address Offset, in Function 1
EcpDFifo	400h
TFifo	400h
CnfgA	400h
CnfgB	401h
ECR	402h
-	403h

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 of I/O at address 0x378)
BAR1 = 0x00000779 (4 bytes of I/O at address 0x778)

If this relationship is not used, custom drivers will be needed.

6.4 Accessing the Local Configuration Registers

The local configuration registers are a set of device specific registers that can be accessed from either function 0 or function 1. They are mapped to the I/O and memory addresses set up in BAR2 and BAR3 of each function, with the offsets defined for each register. I/O or memory accesses can be byte, word or dword accesses, however on little-endian systems such as Intel 80x86 the byte order will be reversed.

6.4.1 Local Configuration and Control register 'LCC' (Offset 0x00)

This register defines control of ancillary functions such as Power Management, Endian selection and the serial EEPROM. The individual bits are described below.

Bits	Description	Read/Write		Reset
		EEPROM	PCI	
0	**Mode**. This bit returns the state of the MODE0 pin.	-	R	X
1	**Test**. This bit returns the state of the TEST pin. Applications must tie the TEST pin to GND, so this bit will always return '0'.	-	R	0
2	**Parallel Port Filter Enable.** This controls the noise filters on the parallel ports input control lines and the data lines, of the parallel port and has meaning only when the parallel port is available (Dual-mode device operation). 1 => Enable filters, 0 => disable filters.	W	RW	1
4:3	**Endian Byte-Lane Select for memory access to 8-bit peripherals**. 00 = Select Data[7:0] 10 = Select Data[23:16] 01 = Select Data[15:8] 11 = Select Data[31:24] Memory access to OX16PCI952 is always DWORD aligned. When accessing 8-bit regions like the internal UARTs and the parallel port, this option selects the active byte lane. As both PCI and PC architectures are little endian, the default value will be 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 time value for the internal powerdown filter, part of the power management circuitry of Function 0 (only). Once Function0 is ready to go into the power down mode, the OX16PCI952 will wait for the specified filter time and if Function0 is still indicating a power-down, it will assert a powerdown request and a PCI interrupt (if the latter is enabled). 00 = power-down request disabled 10 = 129 seconds 01 = 4 seconds 11 = 515 seconds	W	RW	00
23:7	**Reserved**. These bits are used for test purposes. The device driver must write zeros to these bits.	-	R	0000h
24	**EEPROM Clock**. For reads or writes to the external EEPROM , toggle this bit to generate an EEPROM clock (EE_CK pin).	-	W	0
25	**EEPROM Chip Select**. When set to 1, the EEPROM chip-select pin EE_CS is activated (high). When set to 0, EE_CS is de-active (low).	-	W	0
26	**EEPROM Data Out**. For writes to the EEPROM, this output feeds the data-input of the external EEPROM. This bit is output on the devices EE_DO pin 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 (D0) of the external EEPROM connected to EE_DI pin.	-	R	X
28	**EEPROM Valid**. A 1 indicates that a valid EEPROM program header 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 an EEPROM read	-	W	0
30	**Reserved**	-	-	0
31	**Reserved**	-	R	0

6.4.2 Multi-purpose I/O Configuration register 'MIC' (Offset 0x04)

This register configures the operation of the multi-purpose I/O pins 'MIO[1:0]' as follows.

Bits	Description	Read/Write		Reset
		EEPROM	PCI	
1:0	**MIO0 Configuration Register** 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'	W	RW	00
3:2	**MIO1 Configuration Register** 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'	W	RW	00
4	**MIO0 Power Management Event (PME) Enable**. A value of '1' enables the MIO0 pin to set the PME_Status bit in the Power Management Register PMCSR of the selected function, and hence assert the PME# pin if this option has been enabled. A value of '0' prevents MIO0 from setting the PCI PME_Status bit. This pin can affect function 0 or function 1, through the control defined in the GIS (local configuration) register.	W	RW	0
5	**MIO1 Power Management Event (PME) Enable**. A value of '1' enables the MIO1 pin to set the PME_Status bit in the Power Management Register PMCSR of the selected function, and hence assert the PME# pin if this option has been enabled. A value of '0' prevents MIO1 from setting the PCI PME_Status bit. This pin can affect function 0 or function 1, through the control defined in the GIS (local configuration) register.	W	RW	0
31:6	**Reserved**	-	R	0000h

6.4.3 UART FIFO Levels 'UFL' (Offset 0x08)

The receiver and transmitter FIFO levels of both UARTs is mirrored (shadowed) in this local configuration register, 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
23:16	**UART0 Transmitter FIFO Level** (TFL[7:0])	-	R	0x00h
31:24	**UART1 Transmitter FIFO Level** (TFL[7:0])	-	R	0x00h

6.4.4 UART Interrupt Source Register 'UIS' (Offset 0x0C)

The Interrupt Source Register of each UART and the general data status, is mirrored (shadowed) in this local configuration register, as follows.

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
15:12	**Reserved**	-	R	0h
16	**UART0 Good-Data Status**	-	R	1
17	**UART1 Good-Data Status**	-	R	1
30:18	**Reserved**	-	R	00h
31	**Global Good-Data Status**. This bit is the logical AND of bits 16 and 17, 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 a given channel's receiver FIFO levels (from the UFL register) followed by the UIS register, and the Good-Data status for that 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 of 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. 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, for that channel. 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 will no longer be valid.

The UART FIFO Level (UFL), the UART Interrupt Source register (UIS) and the Global Interrupt Status register (GIS) are allocated adjacent address offsets (08h to 10h) in the Base Address Register. The device driver-software can read all of the above registers in a single burst read operation. The location offset of the registers are 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.

6.4.5 Global Interrupt Status and Control Register 'GIS' (Offset 0x10)

This register controls the assertion of interrupts and power management events, as well as returning the internal status of all interrupt sources and power management events.

Bits	Description	Read/Write		Reset
		EEPROM	PCI	
0	**UART 0 internal interrupt status**. This bit reflects the state of UART 0's internal interrupt line[1].	-	R	0x0h
1	**UART 1 internal interrupt status**. This bit reflects the state of UART 1's internal interrupt line[1].	-	R	0x0h
2	**MIO0 Internal State.** This bit reflects the state of the internal MIO[0] signal. The internal MIO[0] signal reflects the non-inverted or inverted state of MIO0 pin. [2]	-	R	X
3	**MIO1 Internal State** This bit reflects the state of the internal MIO[1] signal. The internal MIO[1] reflects the non-inverted or inverted state of MIO1 pin. [2]	-	R	X
15:4	**Reserved**.	-	R	000h
16	**UART 0 Interrupt Mask**. When set (=1) this bit enables UART 0 to assert a PCI interrupt on function 0's interrupt pin (INTA# by default). When cleared (=0), UART 0 is prevented from asserting a PCI interrupt.[3]	W	RW	1
17	**UART 1 Interrupt Mask**. When set (=1) this bit enables UART 1 to assert a PCI interrupt on function 0's interrupt pin (INTA# by default). When cleared (=0), UART 1 is prevented from asserting a PCI interrupt.[3]	W	RW	1
18	**MIO 0 Interrupt Mask**. When set (=1) this bit enables the MIO 0 pin to assert a PCI interrupt, on the selected function's interrupt pin. When cleared (=0) this prevents MIO 0 from asserting a PCI interrupt. The function that is affected is controlled by GIS, bit 26. The MIO 0 pin is active high, unless inversion has been set in the MIC register	W	RW	0
19	**MIO 1 Interrupt Mask**. When set (=1) this bit enables the MIO 1 pin to assert a PCI interrupt, on the selected function's interrupt pin. When cleared (=0) this prevents MIO 1 from asserting a PCI interrupt. The function that is affected is controlled by GIS, bit 27. The MIO 1 pin is active high, unless inversion has been set in the MIC register	W	RW	0
20	**MIO 0 Power-down Mask**. When set (=1) this bit enables the MIO 0 pin to issue a powerdown event by setting the selected function's power-down sticky bit (GIS, bits 22 or 23). The function whose powerdown sticky bit is affected is controlled by GIS, bit 26. Note that if the MIO 0 pin is routed to Function 0, then the pin uses the UART power-down filtering algorithm. Both the UARTs and the MIO 0 pin must indicate a power-down for the filter period before any powerdown requests are issued, for function 0. However, when the MIO 0 pin is routed to Function 1, then a powerdown state on the pin MIO 0 will immediately issue a powerdown request, for function 1, without any filters.	W	RW	0

21	**MIO 1 Power-down Mask**. When set (=1) this bit enables the MIO 1 pin to issue a powerdown event by setting the selected function's power-down sticky bit (GIS, bits 22 or 23). The function whose powerdown sticky bit is affected is controlled by GIS, bit 27. Note that if the MIO 1 pin is routed to Function 0, then the pin uses the UART power-down filtering algorithm. Both the UARTs and the MIO 1 pin must indicate a power-down for the filter period before any powerdown requests are issued, for function 0. However, when the MIO 1 pin is routed to Function 1, then a powerdown state on the pin MIO 1 will immediately issue a powerdown request, for function 1, without any filters.	W	RW	0
22	**Function 0 Powerdown interrupt Status.** This is a sticky bit. When set, it indicates a power-down request issued by Function0. Normally this would have asserted a PCI interrupt on Function 0's interrupt pin (INTA# by default) if GIS bit24 were set. This bit is cleared on reading.	-	R	0
23	**Function 1 Powerdown interrupt Status.** This is a sticky bit. When set, it indicates a power-down request issued by Function1. Normally this would have asserted a PCI interrupt on Function 1's interrupt pin (INTA# by default) if GIS bit25 were set. This bit is cleared on reading.	-	R	0
24	**Function0 Power-down interrupt mask**. When set (=1), this enables Function 0 powerdown requests to assert a PCI interrupt, on Function 0's interrupt pin (INTA# by default)	W	RW	0
25	**Function1 Power-down interrupt mask**. When set (=1), this enables Function 1 powerdown requests to assert a PCI interrupt, on Function 1's interrupt pin (INTA# by default)	W	RW	0
26	**MIO0 Function selection**. When reset (=0), all functional and powerdown interrupt requests, and Power Management Events (PME) due to the MIO0 pin will affect function 0. When set '1', these requests and events will affect function1. [4]	W	RW	1
27	**MIO1 Function selection**. When reset (=0), all functional and powerdown interrupt requests, and Power Management Events (PME) due to the MIO1 pin will affect function 0. When set '1', these requests and events will affect function1. [4]	W	RW	1
28	**Parallel Port Interrupt Status**. When set (='1'), an internal parallel port interrupt is present. This would have issued a PCI interrupt on Function 1's interrupt pin (INTA# by default) if GIS, bit 29 was set. When reset (=0), no internal parallel port interrupts are present.	-	R	0
29	**Parallel Port Interrupt Enable**. When set (=1), an internal parallel port interrupt will assert a PCI interrupt on Function 1's interrupt pin (INTA# by default). When reset (=0), the parallel port will not be able to issue a PCI interrupt.	W	RW	1
31:30	**Reserved**	-	R	0h

Note 1: GIS[1:0] are the inverse of UIS[6] and UIS[0] respectively.

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.

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

Note 4 : Powerdown behaviour of the MIO pins is different when associated with function 0 or function 1. See section "Power Management".

6.5 PCI Interrupts

Interrupts in PCI systems are level-sensitive and can be shared. In the OX16PCI952, there are five sources of interrupts - one from each UART channel, two from Multi-Purpose IO pins (MIO0 to MIO1), and one from the parallel port.

All interrupts can be routed to the PCI interrupt pins, INTA# or INTB#. The default values assigned to the interrupt pin field within each function's PCI configuration space results in all Function 0 interrupts to be made available on the INTA# pin, and all Function1 interrupts also to be made available on the INTA# pin (The *interrupt pin* fields are the same).

These default interrupt pin values may be modified (for example, to map function 1 interrupts on the INTB# line or disable all interrupts altogether) by writing to the Interrupt Pin field in each function's PCI configuration Space using the serial EEPROM facility. 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 card developers must not set any value which violates the PCI local bus specification on this issue. If in doubt, the default values should be used.

Table 7 relates the Interrupt Pin field to the device pin used.

Interrupt Pin Field Value[1]	Interrupt Pin used
0	None
1	INTA#
2	INTB#
3 to 255	Reserved

Table 7: 'Interrupt pin' definition

NOTE 1:
The OXPCI952 has only 2 interrupt pins : INTA# and INTB#. Interrupt Pin values other than 1 or 2, will result in that function not being able to assert an interrupt.

During the system initialisation 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 programmes 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.

The Interrupt status for all sources of interrupts are available using the GIS register in the Local Configuration Register set, which can be accessed using I/O or Memory accesses from both logical functions. This facility enables each function to snoop on interrupts asserted from the other function regardless of the interrupt routing. *This is valid only when the device is operating in the dual-function mode.*

The 5 sources of interrupts on the OX16PCI952, can be enabled/disabled individually using the options in the local configuration register "GIS".

By default, the interrupt options for the UARTs and the parallel port are enabled in the GIS register. Following the initial PCI configuration process, any UART based interrupts that are generated as a result of 1st enabling interrupts in the UART's interrupt register (the ISR register) will result in the assertion of the UART interrupt on the interrupt pin of function 0. By the same token, any parallel port based interrupts will result in the assertion of the parallel port interrupt on the interrupt pin of function 1.

Once the interrupt options for the Multi-purpose MIO pins (MIO0, MIO1) are enabled in the GIS register, the assertion of these pins will, following the initial PCI configuration process, assert an interrupt on the selected function's interrupt pin. The sense of the MIO pin (active-high or active-low) that generates an interrupt is controlled by the options in the MIC register.

For greater flexibility, the MIO pins can be individually associated with either function0 or function1, thereby affecting the selected functions interrupt, power-down, or power management event logic.

Once an interrupt has been asserted, this interrupt can only be removed by the device driver either by disabling the relevant controls in the GIS register or by removing the conditions on the 5 interrupt sources. For the UARTs, this will require reads of the relevant register to clear any UART based interrupts.

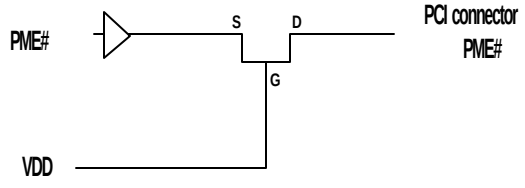
6.6 Power Management

The OX16PCI952 is compliant with the PCI Power Management Specification Revision 1.0. Each logical function implements its own set of Power Management registers and supports the power states D0, D2 and D3.

Power management is accomplished by handling the power-down and power-up ("power management event") requests, that 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. It is a device-specific feature and requires a bespoke 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 power is removed from the device. This will cause 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.



PME# Isolator Circuitry

6.6.1 Power Management of Function 0

Provided that the necessary controls have been set in the device's local configuration registers (LCC, MIC, and GIS), the internal UARTs and the 2 multi_purpose (MIO) pins can be programmed to issue powerdown requests and/or 'wakeup' requests (power management events), for function 0.

For the case of the 2 internal UARTs, function 0 can be configured to monitor the activity of the 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).

For the case when either or both of the MIO pins are associated with function 0, the state of the MIO pin that governs powerdown is the inverse of the MIO state that asserts an interrupt on function 0's interrupt pin (the INTA# line) for normal functionality (if that option were to be enabled). This means that when the MIO pins are not interrupting, the MIO state will be taken as a powerdown state.

When all powerdown sources of function 0 are indicating a powerdown request (this means, both UARTs are indicating a powerdown and each MIO pin associated with function 0 is indicating a powerdown) only then will 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 all function 0 powerdown requests remain valid for this time (for the UARTs, this means that both channels are still inactive) then the OX16PCI952 will issue a powerdown interrupt on this function's interrupt pin, if this option is enabled. *Alternatively, the device driver can poll function 0's powerdown status field in the local configuration register GIS[22] to determine a powerdown request.* The powerdown filter stops the UARTs and any MIO pins associated with function 0 from issuing too many powerdown interrupts whenever the UARTs and MIO pin 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 will the OX16PCI952 change its own power state. The powerdown 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 powerdown filter timer. It operates even if the powerdown filter time is set to disabled.

Like powerdown, wakeup requests for function 0 can be generated by up to 4 sources: by each of the internal UARTs and either of the 2 Multi-purpose MIO pins (when they are associated with function 0). The means to generate wakeup events from these sources will have been setup prior to placing this function into the powerdown states D2 or D3 (including setting of the PME_En bit).

For the case of each UART, when the device (function 0) is in the powerstate D3, only activity on the channel's RI line (the trailing edge of a pulse) will generate a wakeup 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 powerstate 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 for a wake up request from the serial input line EXT_DATA_IN (from the power state D2) then the clock for that channel is turned on so serial data framing can be maintained.

For the case of the MIO pins (associated with function 0), the state of the MIO pins that results in wakeup requests is determined by the settings in the local configuration register MIC. The wakeup behaviour for these pins, unlike the UARTs, is not dependent upon the powerstates D2 or D3. As soon as the correct logic is invoked then a power management event (wakeup) is asserted.

When function 0 issues a wake up request, either from the UARTs or the MIO pins, the PME_Status bit in this function's PCI power management registers (PMCSR[15]) will be set. This is a sticky bit which will only be cleared by writing a '1' to it. While PME_En (PMCSR[8]) remains set, the PME_Status will continue 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 signalled, the device driver is expected to return this function to the D0 power-state.

6.6.2 Power Management of Function 1

Provided that the necessary controls have been set in the device's local configuration registers (MIC, and GIS), only the 2 multi_purpose (MIO) pins can be programmed to issue powerdown requests and/or 'wakeup' requests (power management events), for function 1. *The parallel port is not capable of issuing a powerdown request or power management events but can be placed in a low power state through power management involving the MIO pins.*

When either or both of the MIO pins are associated with function 1, then the state of the MIO pin(s) that issues a powerdown request is controlled by the MIC register. *This state can be the same MIO state that asserts function 1's interrupt pin for normal functionality.* The assertion of the MIO pins will result in a function 1 powerdown request being made immediately. *Unlike the case when the MIO pins are associated with function 0, there is no powerdown filtering time associated with function 1.*

The powerdown request can be issued on the function's interrupt pin, if this option is enabled. *Alternatively, the device driver can poll function 1's powerdown status field in the local configuration register GIS[23] to determine a powerdown request.*

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 will the OX16PCI952 change its own power state. The powerdown 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 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.

The device 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.

Like powerdown, wakeup requests for function 1 can be generated by either of the 2 Multi_purpose MIO pins when they are associated with function 1. The means to generate wakeup events from these sources will have been setup prior to placing this function into the powerdown states D2 or D3.

The state of the MIO pins (when associated with function 1) that results in wakeup requests is determined by the settings in the local configuration register MIC. As soon as the correct logic is invoked than a power management event (wakeup) is asserted. The PME# event is immediate.

When function 1 issues a wake up request, from the MIO pins, the PME_Status bit in this function's PCI power management registers (PMCSR[15]) will be set. This is a sticky bit which will only be cleared by writing a '1' to it. While PME_En (PMCSR[8]) remains set, the PME_Status will continue 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 signalled, the device driver is expected to return this function to the D0 power-state

7 INTERNAL OX16C950 UART

Each of the internal UARTs in the OX16PCI952 is an OX16C950 rev B specification high-performance serial port. The features of this UART are described in this section.

7.1 Operation – mode selection

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

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

Table 8: UART Mode Configuration

Note 1: 950 mode configuration is identical to 650 configuration

7.1.1 450 Mode

After a hardware reset, bit 0 of the FIFO Control Register ('FCR') is cleared, hence the UART is 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.

7.1.2 550 Mode

After a hardware reset, writing a 1 to FCR[0] will increase the FIFO size to 16, providing compatibility with 16C550 devices.

7.1.3 750 Mode

Writing a 1 to FCR[0] will increase 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

7.1.4 650 Mode

The OX16PCI952 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 will result in 650 compatibility with 128 deep FIFOs, as long as FCR[0] is set. Note that the 650 emulation mode of the OX16PCI952 provides 128-deep FIFOs rather than the 32 provided by a legacy 16C650.

In enhanced (650) mode the device has the following features available over those provided by a generic 550. (Note: some of these are similar to those provided in 750 mode, but 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

7.1.5 950 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.

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' (see section 7.11.3). 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 (0x16C95004)
- Readable status for automatic in-band and out-of-band flow control
- External 1x clock modes (see section 0)
- Flexible "M+N/8" clock prescaler (see section 7.10.2)
- Programmable sample clock to allow data rates up to 15 Mbps (see section 7.10.3).
- 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 (see section 7.11). 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 synthesise arbitrary baud rates. The default division value is 4 to provide backward compatibility with 16C650 devices.

The user may 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 16 times over-sampling, where there are 16 clock cycles per bit. However the 950 UART 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 1.8432MHz 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 will also result 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 i.e. to set the device to a 13 cycle sampling clock it would be necessary to write 0x0D to TCR. For further information see section 7.10.3

The UART also offers 9bit data frames for multi-drop industrial applications.

7.2 Register description tables

The 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':

1. LCR[7]=1 enables the divider latch registers DLL and DLM.
2. 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 will set 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.
3. ACR[7]=1 enables access to the 950 specific registers.
4. ACR[6]=1 enables access to the Indexed Control Register set (ICR) registers as described on page 32.

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	
MCR [3,4] 550/750 Mode	100	R/W	Unused		CTS & RTS Flow Control	Enable Internal Loop Back	Unused		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								9th 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							9th 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 9: Standard 550 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 10: 650 Compatible 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 (see Table 12)							

Table 11: 950 Specific Registers

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

Register Name	SPR Offset [10]	R/W	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
Indexed Control Register Set										
ACR	0x00	R/W	Addit-ional 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 (0-127)						
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 (0x04)							
CSR	0x0C	W	Writing 0x00 to this register will reset the UART (Except the CKS and CKA registers)							
NMR	0x0D	R/W	Unused		9th Bit SChar 4	9th Bit SChar 3	9th Bit SChar 2	9th Bit SChar 1	9th-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	Unused							Good Data Status
DMS	0x11	R/W	Force TxRdy inactive	Force RxRdy inactive	Unused				TxRdy status (R)	RxRdy status (R)
PIDX	0x12	R	Hardwired Port Index (0x00)							
CKA	0x13	R/W	Unused					Invert DTR signal	Invert internal tx clock	Invert internal rx clock

Table 12: Indexed Control Register Set

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 via ICR. Offset values not listed in the table are reserved for future use and must not be used.

To read or write to any of the Indexed Controlled Registers use the following procedure:

Writing to ICR 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).

Reading 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 x1xxxxxb 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 x0xxxxxb to ICR, thus enabling access to standard registers again.

7.3 Reset Configuration

7.3.1 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]: $1xx00000_2$
RFC[7:0]: 00000000_2
GDS[7:0]: 00000001_2
DMS[7:0]: 00000010_2
CKA[7:0]: 00000000_2

The reset state of output signals are tabulated below:

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

Table 13: Output Signal Reset State

7.3.2 Software Reset

An additional feature available in the OX16PCI952 UARTs 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 and CKA register). To reset the UART write 0x00 to the Channel Software Reset register 'CSR'.

7.4 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 will remove the next data from the top of the THR and proceed to transmit it. If the THR is empty, it will wait 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 will transfer it to the bottom of the RHR. If the RHR is full, an overrun condition will occur (see section 7.5.3).

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 are indicated in the Line Status Register 'LSR' (see section 7.5.3). 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 (see section 7.11.2).

7.4.1 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 will return 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:
In 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

Table 14: Transmit Interrupt Trigger Levels

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 will be generated (if enabled) if the TFL falls below the trigger level.

750 Mode:
In 750 compatible 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 (see section 7.11.4), 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 hysteresis element in in-band and out-of-band flow control (see section 7.9). 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

Table 15: Compatible Receiver Trigger Levels

A receiver data interrupt will be generated (if enabled) if the Receiver FIFO Level ('RFL') reaches the upper trigger level.

7.5 Line Control & Status

7.5.1 False Start Bit Detection

On the falling edge of a start bit, the receiver will wait for 1/2 bit and re-synchronise 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 will proceed 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 will then wait for a low transition on SIN (signifying the next start bit).

The receiver will continue receiving data even if the RHR is full or the receiver has been disabled (see section 7.11.3) in order to maintain framing synchronisation. The only difference is that the received data does not get transferred to the RHR.

7.5.2 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 will set 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 however, 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

Table 16: LCR Data Length Configuration

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

Table 17: LCR Stop Bit Number Configuration

LCR[5:3]: Parity type
The selected parity type will be 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

Table 18: LCR Parity Configuration

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 recognised 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.

7.5.3 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 9th 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 9th bit of received data is '1' in 9-bit mode.

The Parity error flag will be 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 9th 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 will try to re-synchronise by assuming that the error was 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 will be transferred to the RHR and the receiver will then wait until the SIN line returns high. The LSR[4] break flag will be set when this data item gets to 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 will be 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].

7.6 Interrupts & Sleep Mode

The serial channel interrupts are asserted on the PCI pin, INTA#, by default. The interrupts can be enabled or disabled using the GIS register interrupt mask (see section 6.4.5) and the IER register. Unlike generic 16C550 devices, the interrupt can not be disabled using the implementation-specific MCR[3].

7.6.1 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. 9th 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.

Sleep mode is described in section 7.6.4.

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 (see map on page 31). 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 via EFR[3:0]) and the detection of the XOFF2 special character (when enabled via 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.

7.6.2 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 Table 19.

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

Table 19: Interrupt Status Identification Codes

Note1: ISR[0] indicates whether any interrupts are pending.
Note2: Interrupts of priority levels 5 and 6 cannot occur unless the UART is in Enhanced mode.
Note3: ISR[5] is only used in 650 & 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

7.6.3 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 (9th 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. 9th 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, will occur 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 will only be 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 or 9th 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 (see section 7.11.9).

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.

7.6.4 Sleep Mode

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

- Sleep mode enabled (IER[4]=1 in 650/950 modes, or IER[5]=1 in 750 mode):
- The transmitter is idle, i.e. the transmitter shift register and FIFO are both empty.
- SIN is high.
- The receiver is idle.
- The receiver FIFO is empty (LSR[0]=0).
- The UART is not in loopback mode (MCR[4]=0).
- Changes on modem input lines have been acknowledged (i.e. MSR[3:0]=0000).
- 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 will retain its programmed state whilst in power-down mode.
The channel will automatically exit 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.

7.7 Modem Interface

7.7.1 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] (see section 7.11.3).

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] (see section 7.9.4).

MCR[2]: OUT1
logic 0 ⇒ Force OUT1# output low when loopback mode is disabled.
logic 1 ⇒ Force OUT1# output high.

MCR[3]: OUT2/External interrupt enable
logic 0 ⇒ Force OUT2# output low when loopback mode is disabled. If INT_SEL# is low the external interrupt is in high-impedance state when MCR[3] is cleared. If INT_SEL# is high MCR[3] does not affect the interrupt.
logic 1 ⇒ Force OUT2# output high. If INT_SEL# is low the external interrupt is enabled and operating in normal active (forcing) mode when MCR[3] is high. If INT_SEL# is high MCR[3] does not affect the interrupt.

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 in-active (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 will be accepted as a valid XON (see in-band flow control, section 7.9.3).

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 & 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 will only take effect in Enhanced mode. See section 7.9.1.

7.7.2 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.

7.8 Other Standard Registers

7.8.1 Divisor Latch Registers 'DLL & 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. See section 7.10 for full details.

7.8.2 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. For more information on Indexed Control registers see sections 7.2 and 7.11.

7.9 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.

7.9.1 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 will set 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: 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 will disable 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 will re-enable transmission (see section 7.11.6).

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, recognising XON2 as the XON character and XOFF2 as the XOFF character.
logic [10] ⇒ Single character in-band receive flow control enabled, recognising XON1 as the XON character and XOFF1 and the XOFF character.
logic [11] ⇒ The behaviour 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 setting 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 will be transferred to the RHR and a level 5 interrupt (XOFF or special character) will be 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 will be forced inactive high if the RFL reaches the upper flow control threshold. This will be 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.

7.9.2 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] will be set and a level 5 interrupt is asserted, if enabled by IER[5]. This flag will be 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.

7.9.3 Automatic In-band Flow Control

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

Any valid XON/XOFF characters will not be 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 will be 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 will be re-enabled. The received data will be transferred to the RHR.

When in-band transmit flow control is enabled, the RFL will be sampled whenever the transmitter is idle (briefly, between characters, or when the THR is empty) and an XON/XOFF character will be 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 will be sent and ASR[1] will be set. If ASR[1] is set and the RFL falls below the lower trigger level, XON will be sent and ASR[1] will be cleared.

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

7.9.4 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 will disable 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 will be forced inactive when the RFL reaches the upper trigger level and will return 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 or not the UART is in Enhanced mode.

7.10 Baud Rate Generation

7.10.1 General Operation

The UART contains a programmable baud rate generator that is capable of taking any clock input from 1.8432MHz to 60MHz 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. Also, 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 60MHz). 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

Table 20: Standard PC COM Port Baud Rate Divisors (assuming a 1.8432MHz crystal)

7.10.2 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. 14.7456MHz, 18.432MHz, 32MHz, 40MHz or 60.0MHz. The flexible prescaler allows system designers 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 1.8432MHz.

Table 22 on the following page gives 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.

7.10.3 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, each UART of the OX16PCI952 is designed in a manner to enable 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 will be 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 Table 23 on the following page. 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 OX16PCI952 has the facility to operate at baud-rates up to 15 Mbps in normal mode.

Table 26 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 will return "0000" if read.

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

Table 21: TCR Sample Clock Configuration

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 will be able to access this register. Writing 0x01 to the TCR will 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 will operate in 16x mode.

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

Clock Frequency (MHz)	CPR value	Effective crystal frequency	Error from 1.8432MHz (%)	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

Table 22: Example clock options and their associated maximum baud rates

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

Table 23: Maximum Baud Rates Available at all 'TCR' Sampling Clock Values

7.10.4 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 (see section 7.11.8). 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 will use 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 is synchronised to the 1x clock. In this mode asynchronous data may be transmitted at baud rates up to 60Mbps. The local 1x clock source can be asserted on the DTR# pin.

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 will cause a continuous change to the modem status (unless masked in MDM register, see section 7.11.10).

7.10.5 Crystal Oscillator Circuit

The UARTs reference 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 2.

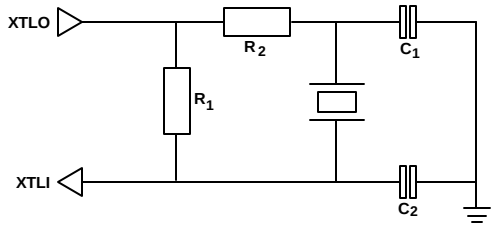


Figure 2: Crystal Oscillator Circuit

Frequency Range (MHz)	C_1 (pF)	C_2 (pF)	R_1 (W)	R_2 (W)
1.8432 - 8	68	22	220k	470R
8-60	33-68	33 – 68	220k-2M2	470R

Table 24: Component values

Note: 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 16pF).

7.11 Additional Features

7.11.1 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 was 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: 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.

7.11.2 FIFO Fill levels 'TFL & 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:

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

7.11.3 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 synchronisation with the receive data stream but received data is not stored into the RHR. In-band flow control characters continue to be detected and acted upon. Special characters will not be detected.
Changes to this bit will only be recognised 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 will only be recognised 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 will be forced inactive high if the Receiver FIFO Level ('RFL') reaches the upper flow control threshold. DTR# line will be re-activated (=0) when the RFL drops below the lower threshold (see FCL & 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 will be 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 will be forced high whenever the transmitter is not empty (LSR[6]=0), otherwise DTR# pin is low.

If the user sets 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# would go inactive once the SOUT line returns to it's 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 ⇒ 950 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 will map 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.

7.11.4 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.

7.11.5 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 the system designer to optimise the interrupt performance hence minimising 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.

7.11.6 Flow Control Levels 'FCL' & '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 optimising 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] as described in section 7.4.1. 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, one XOFF may be 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 one 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].

7.11.7 Device Identification Registers

The identification registers is 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 950 core. This register returns 0x04 for the UART core in this device.

7.11.8 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 [00] ⇒ The RCLK pin is selected for the receiver clock (550 compatible mode).

logic [01] ⇒ The DSR# pin is selected for the receiver clock.

logic [10] ⇒ The output of baud rate generator (internal BDOUT#) 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.

7.11.9 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 9th 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. The user should write the 9th (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 9th 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 9th bit set. As multi-drop systems often use the 9th 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 will result 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, the user 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 (See below).

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 9th bit is set
logic 0 ⇒ Receiver interrupt for detection of an 'address' character (i.e. 9th bit set) is disabled.
logic 1 ⇒ Receiver interrupt for detection of an 'address' character (i.e. 9th 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] will enable 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 9th 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.

7.11.10 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 the user 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. In can not 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. In can not 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. In can not 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. In can not generate an interrupt, assert a PME# or wake up the UART.

MDM[4]: Reserved
This bit must be set to '0'

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

7.11.11 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.

7.11.12 Good-data status register 'GDS'

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

For the definition of Good-data status refer to section 6.4.3

GDS[0]: Good Data Status
GDS[7:1]: Reserved

7.11.13 DMA Status Register 'DMS'

The DMS register is located at offset 0x11 of the ICR. This allows the internal TXRDY# and RXRDY# lines to be permanently deasserted, and the current internal status to be monitored. This mainly has applications for testing.

DMS[0]: RxRdy Status
Read Only: set when RxRdy is asserted (pin driven low).

DMS[1]: TxRdy Status
Read Only: set when TxRdy is asserted (pin driven low).

DMS[5:2] Reserved

DMS[6]: Force RxRdy Inactive
logic 0 ⇒ RxRdy# acts normally
logic 1 ⇒ RxRdy# is permanently inactive (high) regardless of FIFO thresholds

DMA[7]: Force TxRdy Inactive
logic 0 ⇒ TxRdy# acts normally
logic 1 ⇒ TxRdy# is permanently inactive (high) regardless of FIFO thresholds.

7.11.14 Port Index Register 'PIX'

The PIX register is located at offset 0x12 of the ICR. This read-only register gives the UART index. For a single channel device such as the OX16C950 this reads '0'.

7.11.15 Clock Alteration Register 'CKA'

The CKA register is located at offset 0x13 of the ICR. This register adds 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 the user to select a clock mode and then reset the channel to work-around any timing glitches.

8 BI-DIRECTIONAL PARALLEL PORT

8.1 Operation and Mode selection

The OX16PCI952 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.

The system can access the parallel port registers via two blocks of I/O space as defined by 2 BARs (Base Address Registers) in the PCI Configuration space of function 1(dual-mode device operation only). BAR0 (8-bytes of I/O space) contains the address of the basic parallel port registers and BAR1 (4-bytes of I/O space) contains the address of the upper (extended) registers. These register blocks are referred to as the 'lower block' and 'upper block', respectively, in this section.

If the upper block is located at an address 0x400 above the lower block, as shown below, then generic PC parallel port device drivers can be used to configure the port, as the addressable registers of legacy parallel ports always have this relationship.

BAR0 = 0x00000379 (8 bytes of I/O at address 0x378)
BAR1 = 0x00000779 (4 bytes of I/O at address 0x778)

If this relationship is not used, then a custom driver will be needed.

8.1.1 SPP mode

SPP (output-only) mode is the standard implementation of a simple parallel port, and is the default mode of the OX16PCI952 following a reset.

In this mode, the parallel port data lines (PD[7:0] lines) always drive the value written to the PDR register. All data transfers are done under software control using the registers DCR and DSR. Input must be performed in nibble mode.

Generic device driver-software may use the address in I/O space defined by BAR0 (of function 1) to access these parallel port registers.

8.1.2 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 IEEE1284 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 & clearing DCR[5], the data direction bit.

8.1.3 EPP mode

To use the Enhanced Parallel Port ('EPP') mode, the mode field of the Extended Control Register (ECR[7:5]) must be set to '100' using the negotiation steps as defined by the IEEE1284 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 failed to respond to an EPP cycle the host would never see a rising edge on WAIT#, and subsequently lock up. A built-in time-out facility is provided in order to prevent this from happening. It uses a fixed 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 EPP registers are found at offset 000-007h and 400-402h.

8.1.4 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 IEEE1284 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.

8.2 Parallel port interrupt

The parallel port interrupt is asserted on function 1's interrupt pin (INTA# by default, on the OX16PCI952). This interrupt is enabled by setting bit 4 of the DCR register.

When the interrupt is enabled, a rising edge of the ACK# (INTR#) pin results in a parallel port interrupt to be asserted on function 1's interrupt pin and this interrupt state to be mirrored in the register DSR, bit 2. This condition is maintained until the status register (DSR) is read, which clears the interrupt and clears the interrupt status field in this register (DSR[2]).

8.3 Register Description

The parallel port registers are described below.
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
SPP (Compatibility Mode) Registers										
PDR	000h	R/W	Parallel Port Data Register							
ecpAFifo	000h	R/W	ECP FIFO : Address / RLE							
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							

Lower Address Block (BAR0)

Upper Address Block (BAR1)

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]		Must write '00001'
-	403h	-	Reserved		

Table 25: Parallel port register set

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 signalling

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 '00000001' respectively.

8.3.1 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) will drive data onto the parallel port data lines. In all the other modes, the drivers may be tri-stated by setting the direction bit in the DCR. Reads from this register return the actual logic values on the parallel port data lines.

8.3.2 ECP FIFO Address / RLE

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

8.3.3 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 (see section 8.1.3)

DSR[0]:
EPP mode only: Timeout
logic 0 ⇒ EPP Timer Timeout has not occurred.
logic 1 ⇒ EPP Timer Timeout has occurred (Reading this bit clears it).

Other Parallel Port modes: Unused
This bit returns a '1'.

DSR[1]: Unused
This bit returns a '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.

8.3.4 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 the STB#, AFD#, INIT# and SLIN# pins, respectively.

When in the EPP mode, the WRITE#, DATASTB# and the ADDRSTB# pins are normally controlled by the EPP controller, for EPP bus signalling, but writes to the corresponding bits in the DCR register will override the EPP controller's states for these 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 pins in output mode.
logic 1 ⇒ PD pins in input mode.

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)

*Note : 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 drivers must not utilise the values associated with these bits. The OX16PCI952 returns "00" for these bits, for all parallel port modes.

8.3.5 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 will be transferred to the peripheral as an EPP address by the hardware. A read from this register will transfer an address from the peripheral under hardware control.

8.3.6 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.

8.3.7 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.

8.3.8 Test FIFO

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

8.3.9 Configuration A register

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

8.3.10 Configuration B register

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

8.3.11 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 – read / write
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

9 SERIAL EEPROM SPECIFICATION

The OX16PCI952 can be configured using an optional serial electrically-erasable programmable read only memory (EEPROM). If the EEPROM is not present, the device will remain in its default configuration after reset. Although this may be adequate for some applications, many will benefit from the degree of programmability afforded by this feature. The EEPROM also allows accesses to the integrated UARTs and the parallel port, which can be useful for default setups.

The EEPROM interface supports a variety of serial EEPROM devices that have a proprietary serial interface known as Microwire™. This 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 address, then input serially the data. The interface controller has been designed to handle (autodetect) the following list of compatible devices that have a 16-bit data word format but differ in memory size (and hence the number of address bits). NM93C46 (64 WORDS), NM93C56 (128 WORDS), devices with 256 WORDS, 512 WORDs and 1024 WORDS.

The OX16PCI952 incorporates a controller module which reads data from the serial EEPROM and writes data into the relevant register space. It performs this operation in a sequence which starts immediately after a PCI bus reset and ends either when the controller finds no EEPROM is present or when it reaches the end of the eeprom data. *Note that any attempted PCI access while the eeprom is being sensed or while data is being downloaded from the serial EEPROM will result in a "retry" response*. The operation of this controller is described below.

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 as desired. Any changes to the eeprom contents, however, will not take effect until a PCI bus reset takes place or until the eeprom reload option (LCC[29]) is set.

A Windows® based utility to program the EEPROM is available. For further details please contact Oxford Semiconductor (see back cover).

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

9.1 EEPROM Data Organisation

The serial EEPROM data is divided into five zones. The size of each zone is an exact multiple of 16-bit WORDs. Zone0 is allocated to the header. An EEPROM program must contain a valid header before any further data is interrogated.

The general EEPROM data structure is shown in Table 26.

DATA Zone	Size (Words)	Description
0	One	Header
1	Multiples of 2	Function Access
2	One to more	Local Configuration Registers
3	One to four	Identification Registers
4	Two or more	PCI Configuration Registers

Table 26: EEPROM data format

9.1.1 Zone0: Header

The header identifies the EEPROM program as valid.

Bits	Description
15:4	These bits should return 0x950 to identify a valid program. Once the OX16PCI952 reads 0x950 from these bits, it sets LCC[28] to indicate that a valid EEPROM program is present.
3	1 = Zone1 (Function Access) exists 0 = Zone1 does not exist
2	1 = Zone2 (Local Configuration Registers) exists 0 = Zone2 does not exist
1	1 = Zone3 (Identification Registers) exists 0 = Zone3 does not exist
0	1 = Zone4 (PCI Configuration Registers) exists 0 = Zone4 does not exist

The programming data for each zone follows the proceeding zone if it exists. For example a Header value of 0x950F indicates that all zones exist and they follow one another in sequence (from Zone1 to Zone4), while 0x950A indicates that only Zones 1 and 3 exist where the header data is followed by Zone1 WORDs, and since Zone2 is missing Zone1 WORDs are followed by Zone3 WORDs.

9.1.2 Zone 1: Function Access

Zone 1 allows each UART and the Parallel Port of the OX16PCI952 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 utilising the enhanced features/modes of these logical units. For example, the 950 mode of the UARTs offer high performance. Generic UART drivers will be unable to enable 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) via 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 UART and/or the parallel port through its assigned I/O BAR (base address register), with the exception that a function read access does not return any data (it is discarded internally). The UARTs and the parallel port behave as though these function accesses via the eeprom were corresponding read/write pci accesses.

Each entry for Function Access Zone comprises 2 16 bit words (word pairs), with the exception when ending this zone that requires a single word (all 0's) for 'termination'. The format is as shown.

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 000 for BAR 0 (UART0 or parallel port I/O BAR) 001 for BAR 1 (UART1 or parallel port IO BAR) *Others reserved*.
1	11	'0' : Read access required (data will be discarded) '1' : Write access required
1	10:8	Function Number, requiring access. 000 – Function 0 (UARTs) 001 – Function 1 (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).

1st WORD of FUNCTION ACCESS PAIR

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)

2nd WORD of FUNCTION ACCESS PAIR

Function Access Examples

1) 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

2) (Continue). 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

3) (Continue). 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. See example 3.*

9.1.3 Zone2: Local Configuration Register Zone

Zone2 of the EEPROM allows the end-user to override the default settings of the Local Configuration Registers to meet their specific applications. Accesses require 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 (see section 6.4). *Note: Not all of the registers in the Local Configuration Register set are writable by the EEPROM. (See section "Accessing the Local Configuration Registers").*

The format of configuration WORDs for the Local Configuration Registers in Zone 2 are described in the following table.

Bits	Description
15	'0' = There are no more Configuration WORDs to follow in Zone2. 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 LCC[23:16] is 0x02.
7:0	8-bit value of the register to be programmed

EEPROM Word Format for Zone 2

9.1.4 Zone 3 : Identification Registers

Zone 3 of the EEPROM allows the end-user to change the Vendor and/or Subsystem Vendor ID's of both logical functions, to suit their requirements. Writes to this zone affects the corresponding fields in the PCI Configuration Space of both function 0 and function 1.

The format of Device Identification configuration WORDs are described in Table 27.

Bits	Description
15	'0' = There are no more Zone 3 (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 Zone 3 (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

Table 27: Zone 3 data format

9.1.5 Zone 4 : PCI Configuration Registers

Zone 4 of the EEPROM allows the end-user to alter the default values/settings of the PCI Configuration Space registers of both functions, independently. This excludes writes to the Vendor ID and Subsystem Vendor ID fields that can be written only through Zone 3).

This zone is divided into two groups, one for the PCI Configuration Space of function 0 and one for function 1. Each group 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 4. '1' = Define this function header.
14:3	Reserved. Write zeros.
2:0	Function number for the following configuration WORD(s). '000' = Function 0 (UARTs) '001' = Function 1 (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 are 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. Offset values are tabulated in section 6.2.
7:0	8-bit value of the register to be programmed

Table 28: Zone 4 data format (data)

Table 29 defines which PCI Configuration registers are writable from the EEPROM.

Offset	Bits	Description
0x00	7:0	Vendor ID bits 7 to 0.
0x01	7:0	Vendor ID bits 15 to 8.
0x02	7:0	Device ID bits 7 to 0.
0x03	7:0	Device ID bits 15 to 8.
0x06	3:0 4 7:5	Must be '0000'. Extended Capabilities 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.
0x0E	7	Header Type (Multi/single function bit)
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.

Table 29: EEPROM-writable PCI configuration registers

10 OPERATING CONDITIONS

Symbol	Parameter	Min	Max	Units
V_{DD}	DC supply voltage	-0.3	7.0	V
V_{IN}	DC input voltage	-0.3	V_{DD} + 0.3	V
I_{IN}	DC input current		+/- 10	mA
T_{STG}	Storage temperature	-40	125	°C

Table 30: Absolute maximum ratings

Symbol	Parameter	Min	Max	Units
V_{DD}	DC supply voltage	4.75	5.25	V
T_C	Temperature – Commercial	0	70	°C

Table 31: Recommended operating conditions

11 DC ELECTRICAL CHARACTERISTICS

11.1 5v Standard (non-PCI) I/O Buffers

Vdd = 5.0v +/- 5%, Ta = 0 to 70c

Symbol	Parameter	Condition	Min	Typ	Max	Units
V_{IH}	High level input voltage	TTL Interface	2.0			V
V_{IL}	Low level input voltage	TTL Interface			0.8	V
V_T	Switching Threshold	TTL		1.4		V
V_{T+}	Schmitt Trigger, positive-going threshold	TTL			2.0	V
V_{T-}	Schmitt Trigger, negative-going threshold	TTL	0.8			V
I_{IH}	High level input current	$V_{in} = V_{DD}$, no pull-downs. $V_{in} = V_{DD}$, buffer with pull-down.	-10 10	 50	10 100	µA
I_{IL}	Low level input current	$V_{in} = V_{SS}$, no pull-ups $V_{in} = V_{SS}$, buffer with pull-up.	-10 -100	 -50	10 -10	µA
V_{OH}	High level output voltage	I_{OH} = -1 µA I_{OH} = -1mA to -12mA	$V_{DD} - 0.05$ 2.4			V
V_{OL}	Low level output voltage	I_{OL} = 1 µA I_{OL} = 1mA to 12 mA			0.05 0.4	V
I_{OZ}	Tri-state output leakage current	Vout = Vss or Vdd	-10		10	µA
Ios	Output Short Circuit Current	Vdd = 5.25v, Vo = Vdd Vdd = 5.25v, Vo = Vss	 -180		200	mA
I_{DD}	Quiescent Supply Current	Vin = Vss or Vdd			100[1]	uA
C_{IN}	Cap of input buffers[2]	Any input and bi-directional Buffers			4.0	pF
C_{OUT}	Cap of output buffers[2]	Any Output Buffer			4.0	pF

Table 32: Characteristics of 5v tolerant I/O buffers

Note 1 : This value depends upon the customer design.
Note 2: This value excludes package parasitics

11.2 PCI I/O Buffers

Symbol	Parameter	Condition	Min	Max	Unit
DC Specifications					
V_{CC}	Supply voltage		4.75	5.25	V
V_{IL}	Input low voltage		-0.5	0.8	V
V_{IH}	Input high voltage		2.0	V_{CC} + 0.5	V
I_{IL}	Input low leakage current	V_{IN} = 0.5V		-70	µA
I_{IH}	Input high leakage current	V_{IN} = 2.7V		70	µA
V_{OL}	Output low voltage	I_{OUT} = -2 mA		0.55	V
V_{OH}	Output low voltage	I_{OUT} = 3 mA, 6mA	2.4		V
C_{IN}	Input pin capacitance			10	pF
C_{CLK}	CLK pin capacitance		5	12	pF
C_{IDSEL}	IDSEL pin capacitance			8	pF
L_{PIN}	Pin inductance			10	nH

12 AC ELECTRICAL CHARACTERISTICS

12.1 PCI I/O Buffers

All PCI I/O buffers are compliant to the AC Electrical Characteristics of the 5.0v signalling environment, as defined in the PCI Local Bus Specification, revision 2.2. (Section 4.2.1.2 of this specification)

AC Specifications					
	Switching current	$0 < V_{OUT}$ 1.4	-44		
$I_{OH(AC)}$	high	$1.4 < V_{OUT}$ 2.4	$-44 (V_{OUT} - 1.4)/0.024$		mA
		$3.1 < V_{OUT}$ V_{CC}		Eq. A	
	(Test point)	$V_{OUT} = 3.1$		-142	
	Switching current	V_{OUT} 2.2	95		
$I_{OL(AC)}$	low	$2.2 > V_{OUT} > 0.55$	$V_{OUT} / 0.023$		mA
		$0.71 > V_{OUT} > 0$		Eq. B	
	(Test point)	$V_{OUT} = 0.71$		206	
I_{CL}	Low clamp current	$-5 < V_{IN} < -1$	$-25 + (V_{IN} +1)/ 0.015$		mA
I_{HL}	High clamp current	$V_{CC}+4 < V_{IN} < V_{CC}+1$	$25+ (V_{IN} -V_{CC} -1)/ 0.015$		mA
$Slew_R$	Output rise slew rate	0.4V to 2.4V	1	5	V/nS
$Slew_F$	Output fall slew rate	2.4V to 0.4V	1	5	V/nS

Table 32: Characteristics of PCI I/O buffers

Eq. A : $I_{OH} = 11.9 * (V_{OUT} - 5.25) * (V_{OUT} + 2.45)$ for $3.1 < V_{OUT}$ V_{CC}

Eq. B : $I_{OL} = 78.5 * V_{OUT} * (4.4 - V_{OUT})$ for $0.71 > V_{OUT} > 0$

12.2 Serial Ports

Isochronous (x1 Clock) Timing:

Symbol	Parameter	Min	Max	Units
t_{irs}	SIN set-up time to Isochronous input clock 'Rx_Clk_In rising [1]	TBD	10	ns
t_{irh}	SIN hold time after Isochronous input clock 'Rx_Clk_In' rising [1]	TBD	0.1	ns
t_{its}	SOUT valid after Isochronous output clock 'Tx_Clk_Out' falling [1]	TBD	6	ns

Table 33: Isochronous mode timing

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. The system designer is should ensure that mark-to-space ratio of the x1 clock is such that the required set-up and hold timing constraint are met. 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.

13 Timing Waveforms

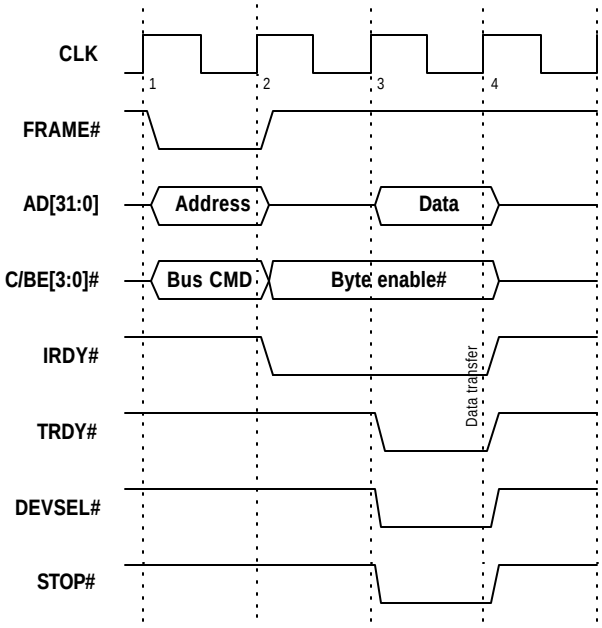


Figure 1: PCI Read transaction from the PCI Configuration Space (Function 0 or Function 1)

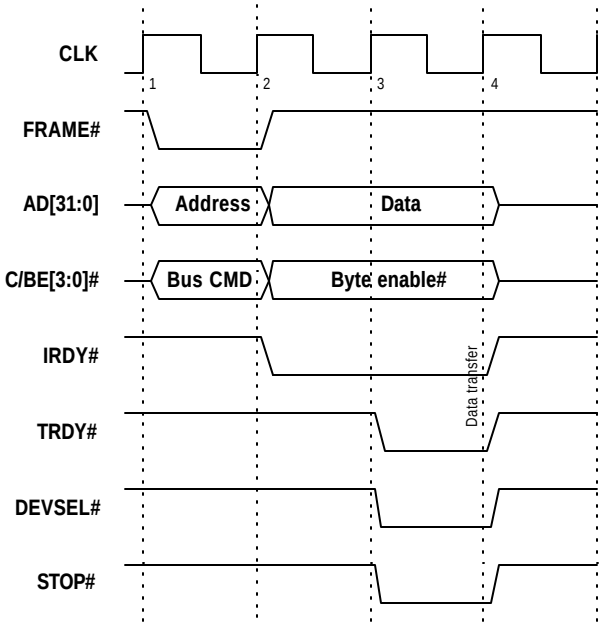


Figure 2: PCI Write transaction to the PCI Configuration Space (Function 0 or Function 1)

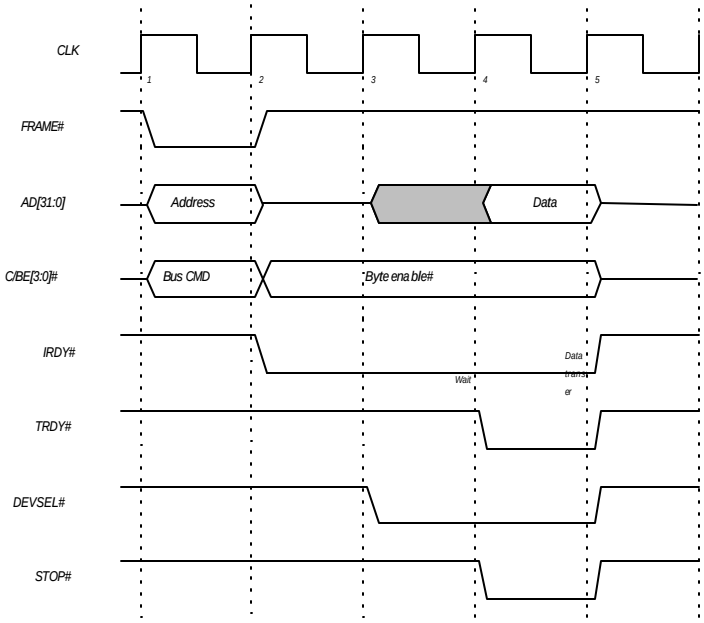


Figure 3. PCI read from the internal UARTs (UART 0 or UART 1)

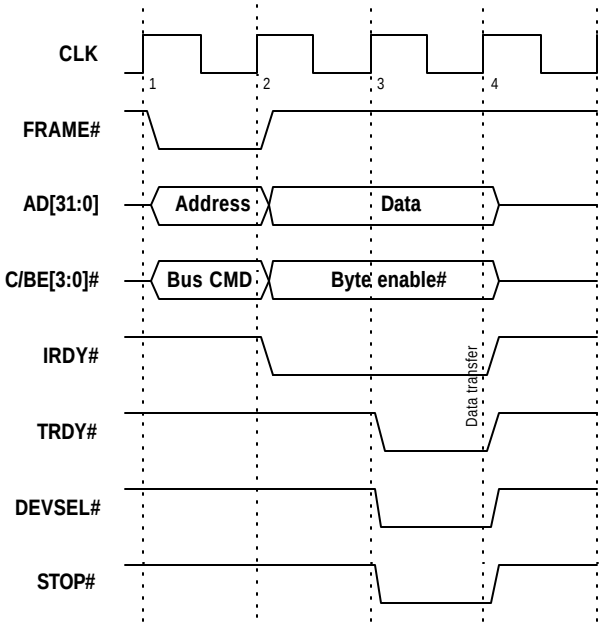
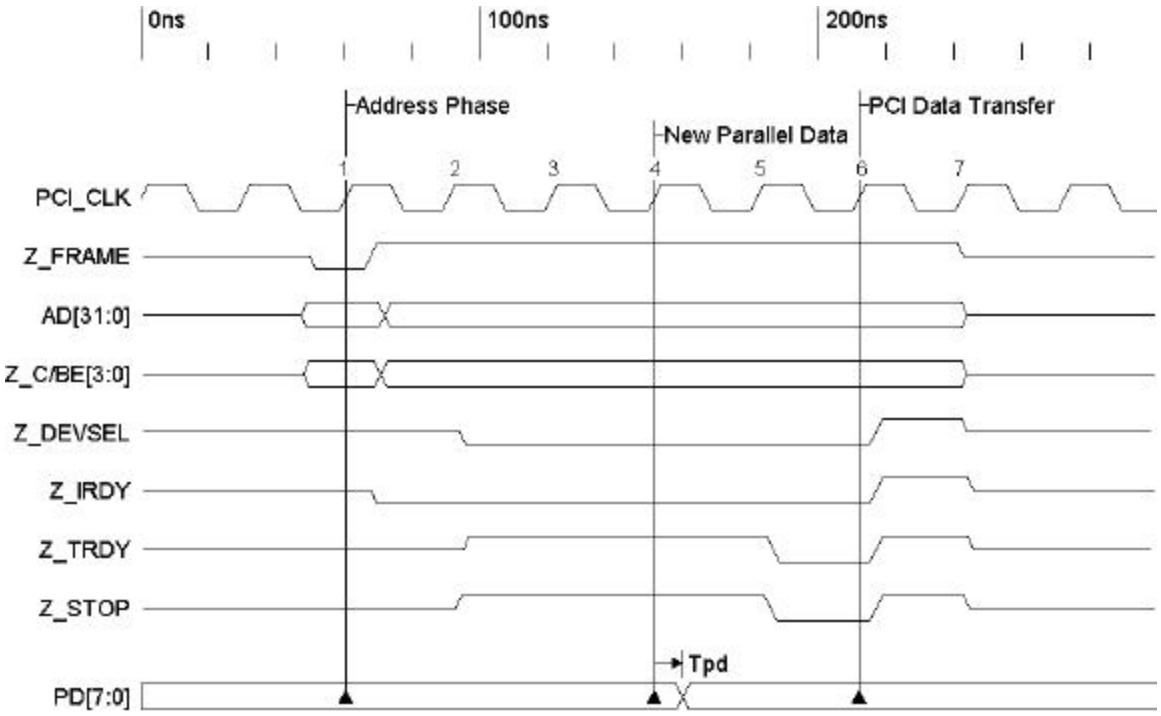


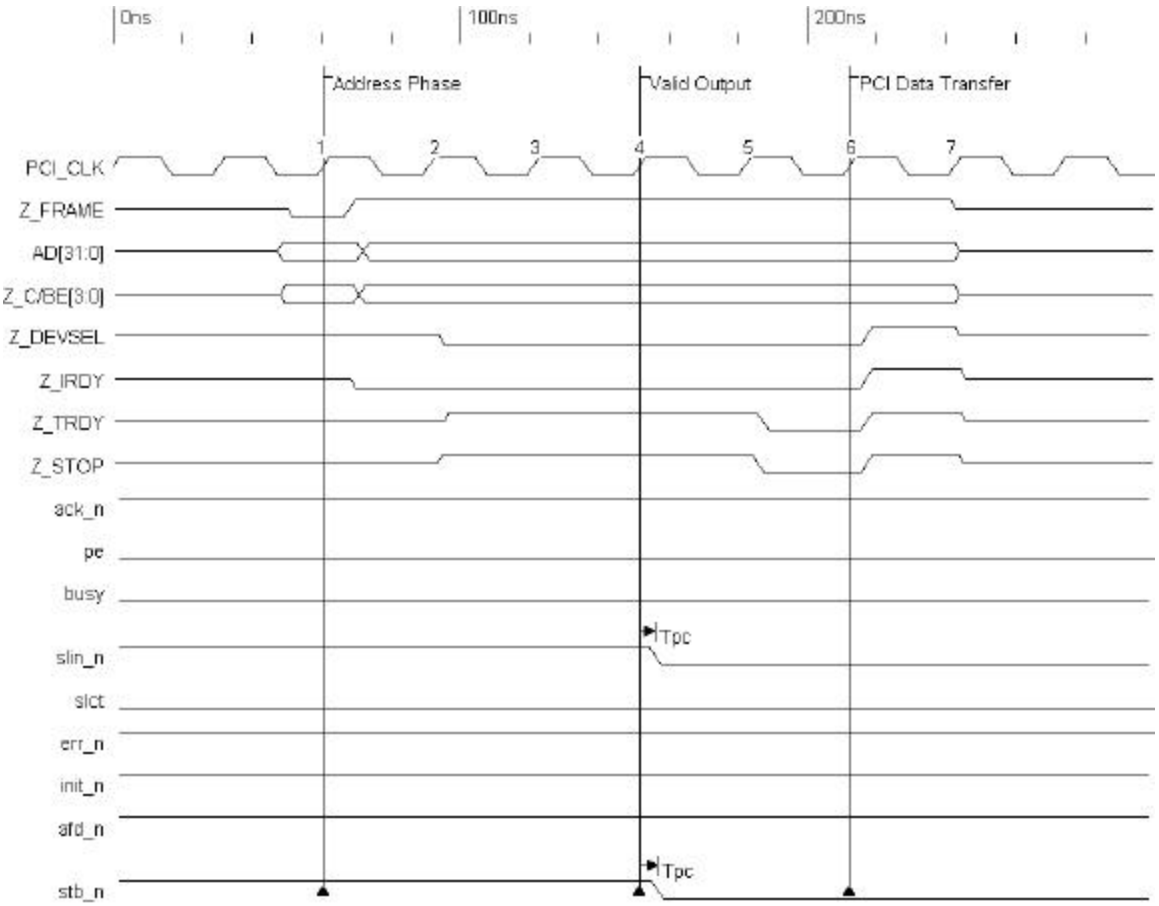
Figure 4. PCI Write to the internal UARTs (UART0 or UART 1)



PCI Write to SPP Data register

Tpd (PCI CLK to Valid Parallel Port Data) : 16 ns max*

** These values exclude the effects of external parasitic (board) capacitances.*

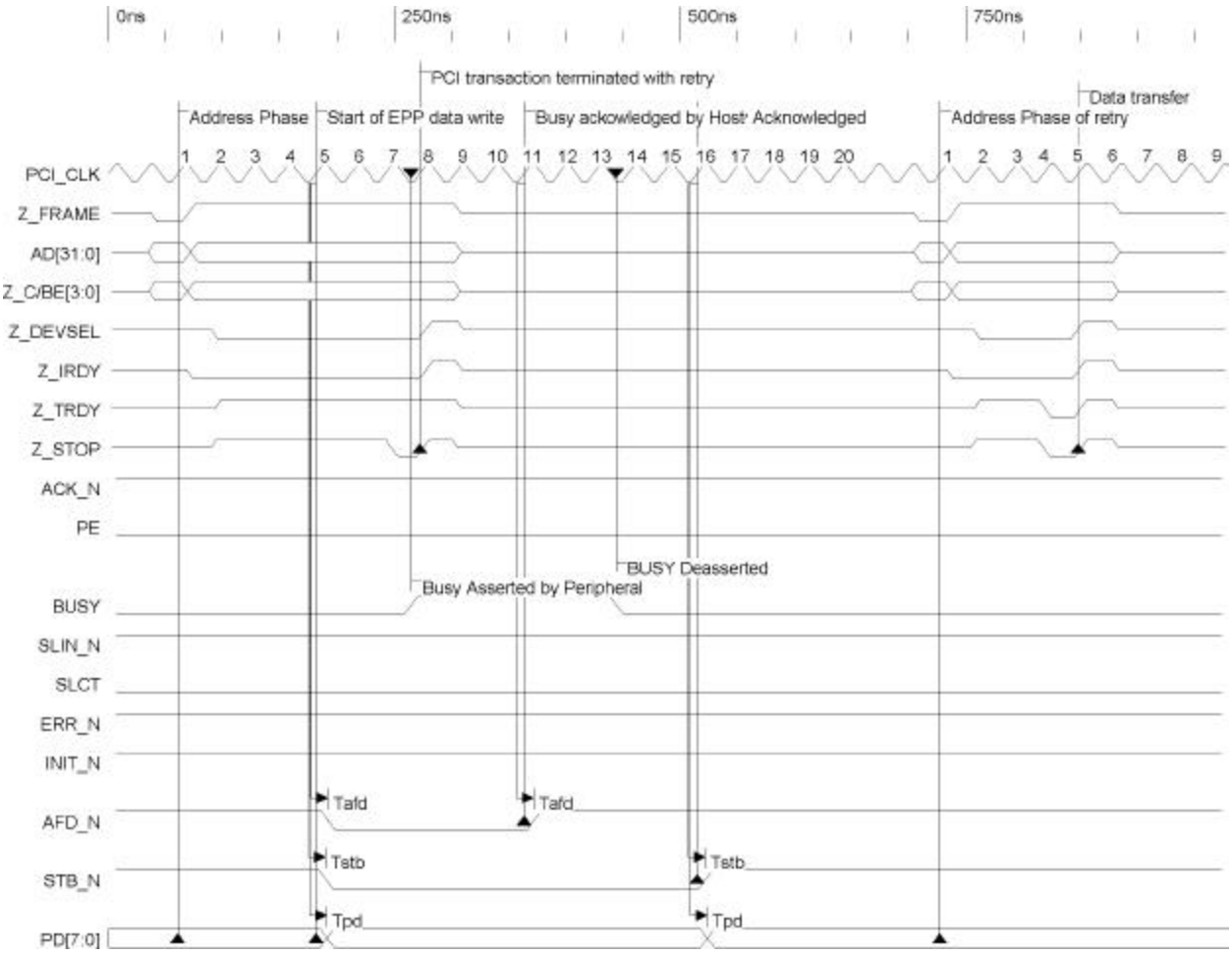


PCI Write to SPP DCR Register

Tpc : PCI CLK to Valid Parallel Port Control lines.

Tpc (Slin_N)	: 16.0 ns max*
Tpc (Stb_N)	: 16.0 ns max*
Tpc (Init_N)	: 16.0 ns max*
Tpc (Afd_N)	: 16.0 ns max*

** These values exclude the effects of external parasitic (board) capacitances.*



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.

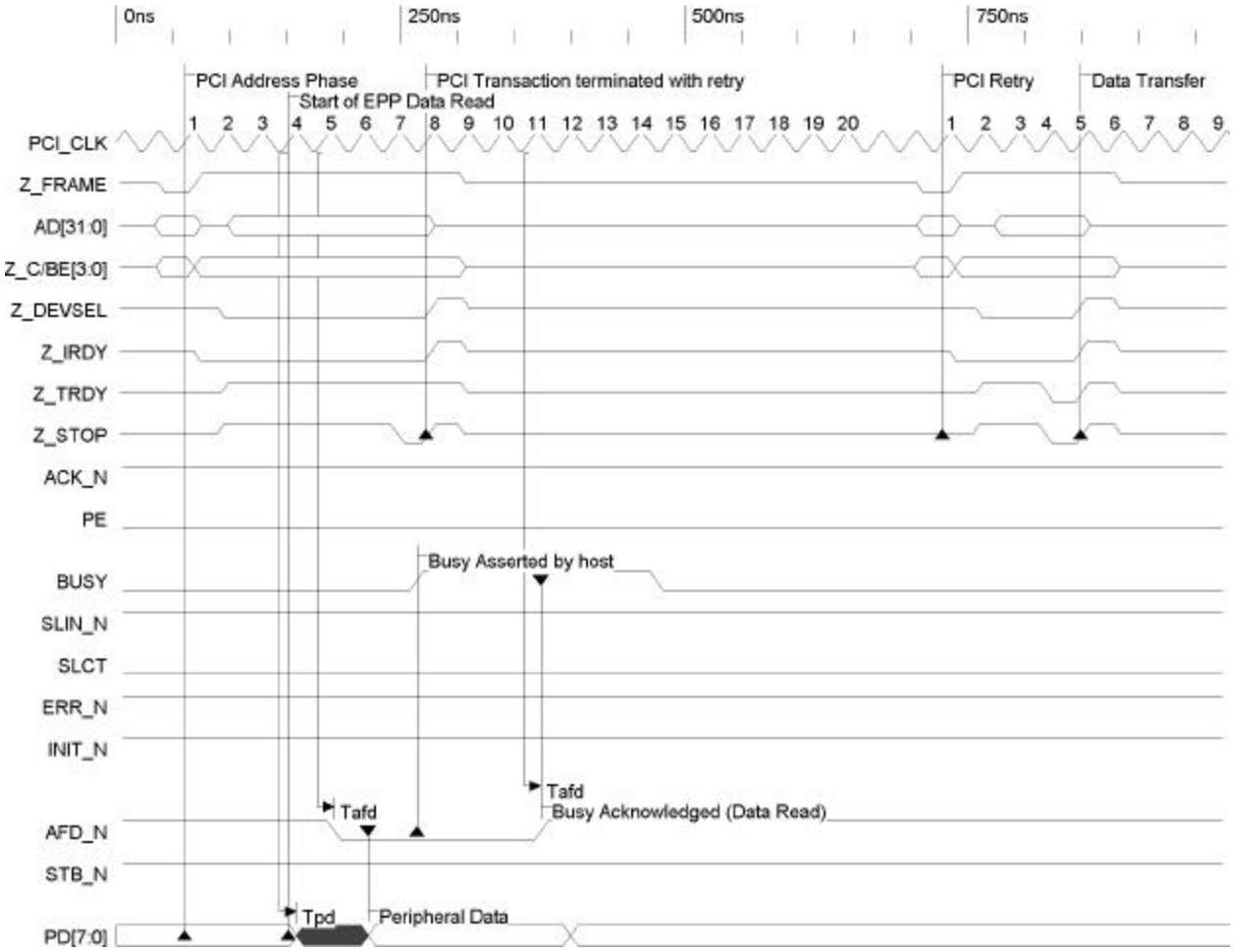
PCI CLK no (Retry Transaction)

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)	- 16ns max*
Tstb (PCI clk to valid STB_N)	- 16ns max*
Tpd (PCI clk to valid Parallel Data)	- 16ns max*

EPP Write Data Cycle duration is dependant 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 will be incurred in the response of the host to the peripheral's BUSY line when the filters are enabled.

** These values exclude the effects of external parasitic (board) capacitances.*



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.

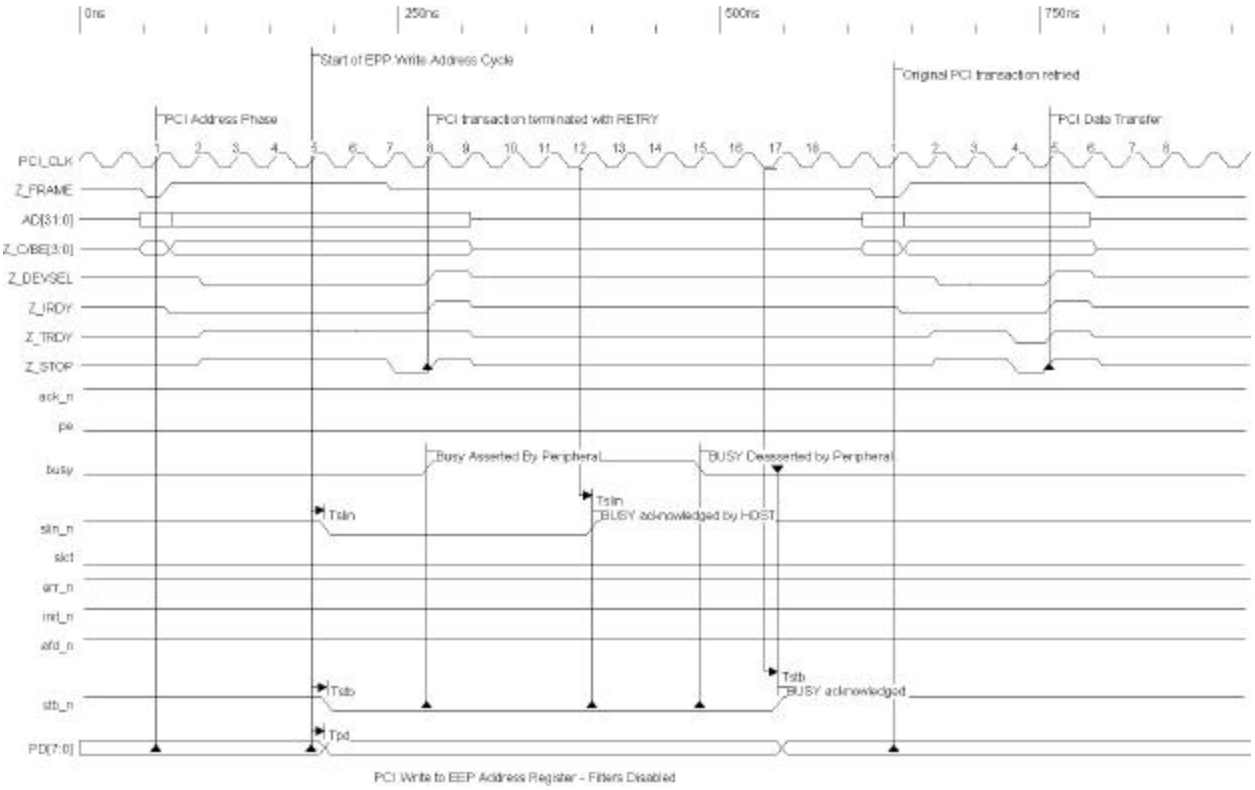
PCI CLK no (Retry Transaction)

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)	- 16ns max*
Tstb (PCI clk to valid STB_N)	- 16ns max*
Tpd (PCI clk to valid Parallel Data)	- 16ns max*

EPP Read Data Cycle duration is dependant 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 will be incurred in the response of the host to the peripheral's BUSY line when the filters are enabled.

** These values exclude the effects of external parasitic (board) capacitances.*



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.

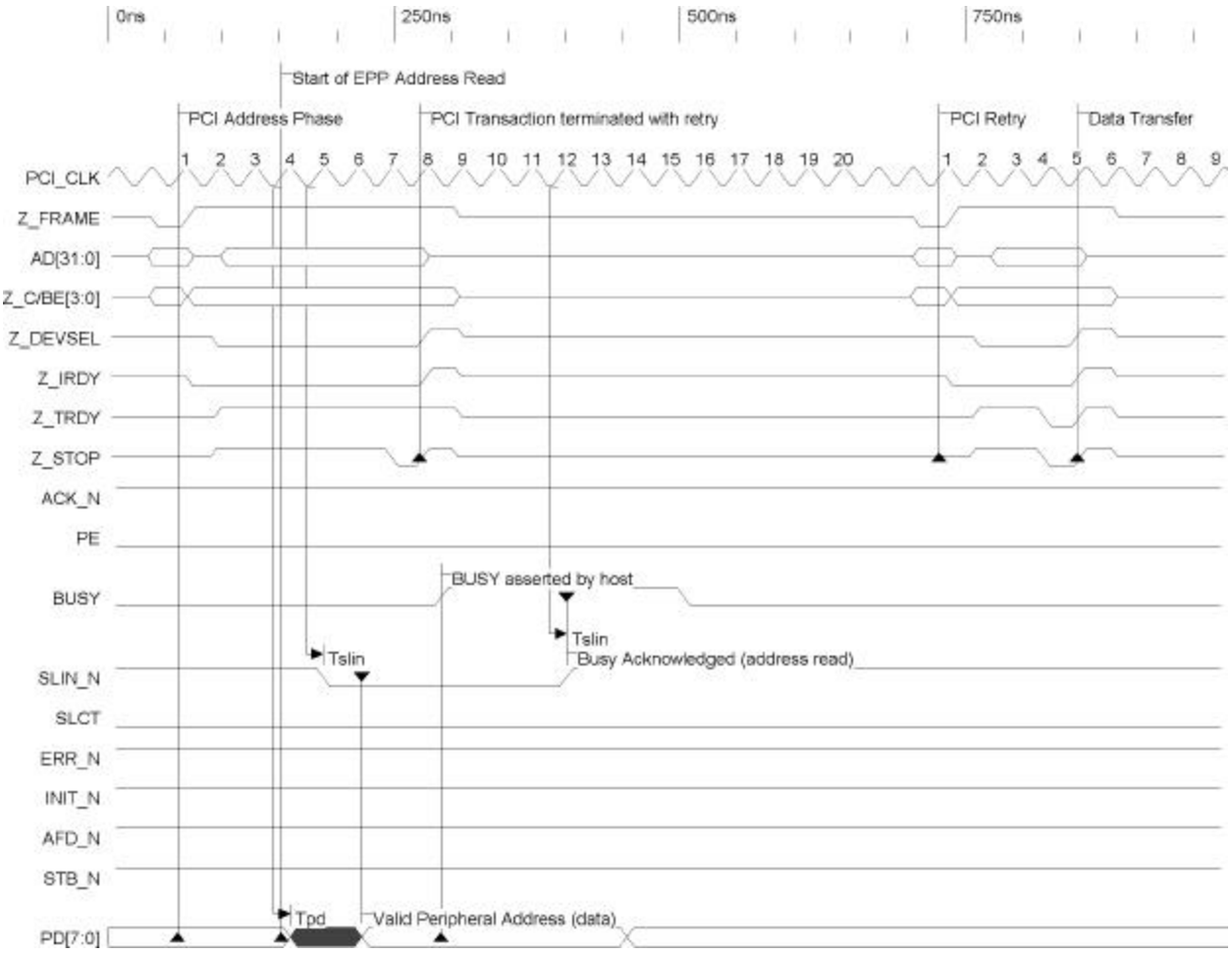
PCI CLK no (Retry Transaction)

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)	- 16ns max*
Tstb (PCI clk to valid STB_N)	- 16ns max*
Tpd (PCI clk to valid port Address)	- 16ns max*

EPP Write Address Cycle duration is dependant 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 will be incurred in the response of the host to the peripheral's BUSY line when the filters are enabled.

** These values exclude the effects of external parasitic (board) capacitances.*



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.

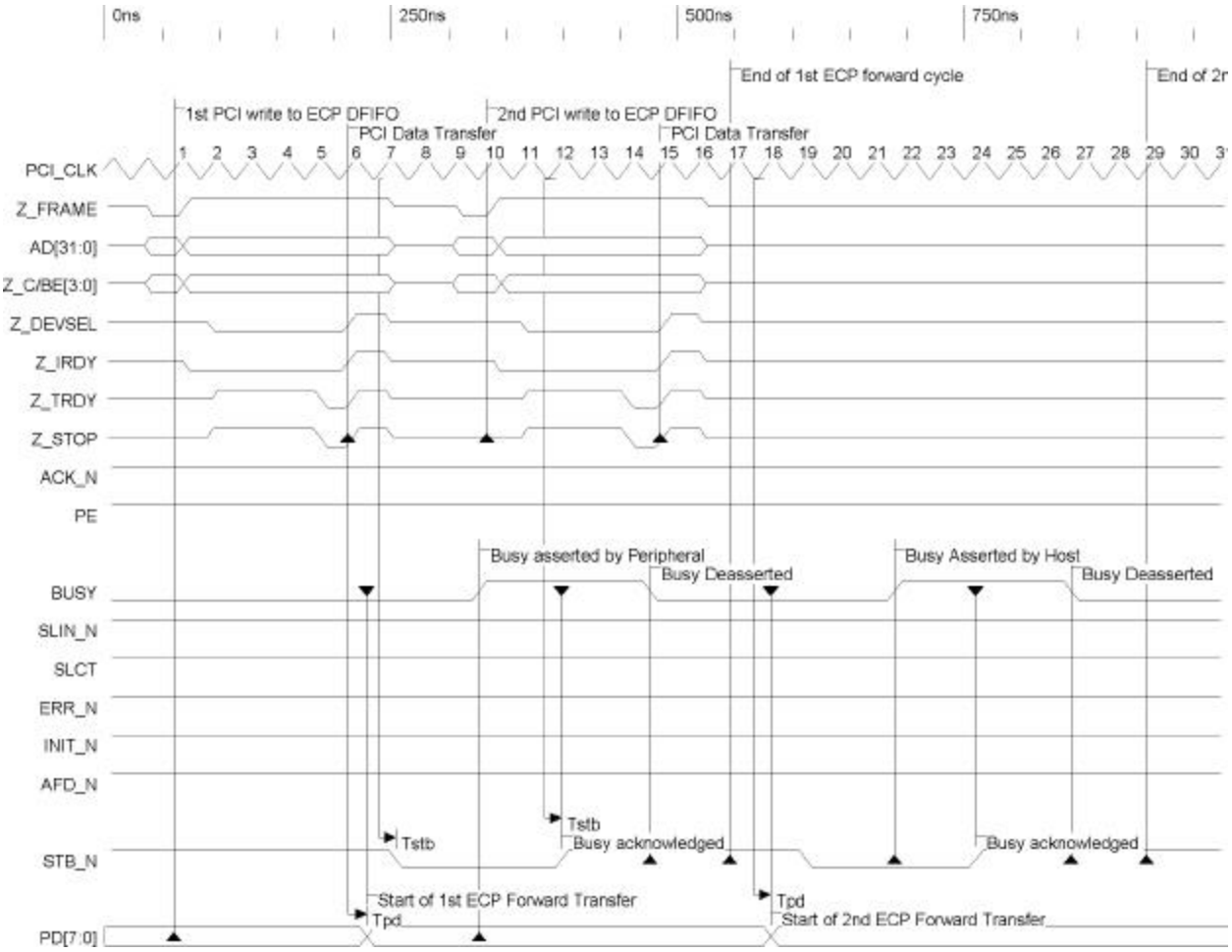
PCI CLK no (Retry Transaction)

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) - 16ns max*
Tpd (PCI clk to valid Port Address) - 16ns max*

EPP Read Address Cycle duration is dependant 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 will be incurred in the response of the host to the peripheral's BUSY line when the filters are enabled.

** These values exclude the effects of external parasitic (board) capacitances.*



2 Consecutive Write Transactions to ECP DFIFO

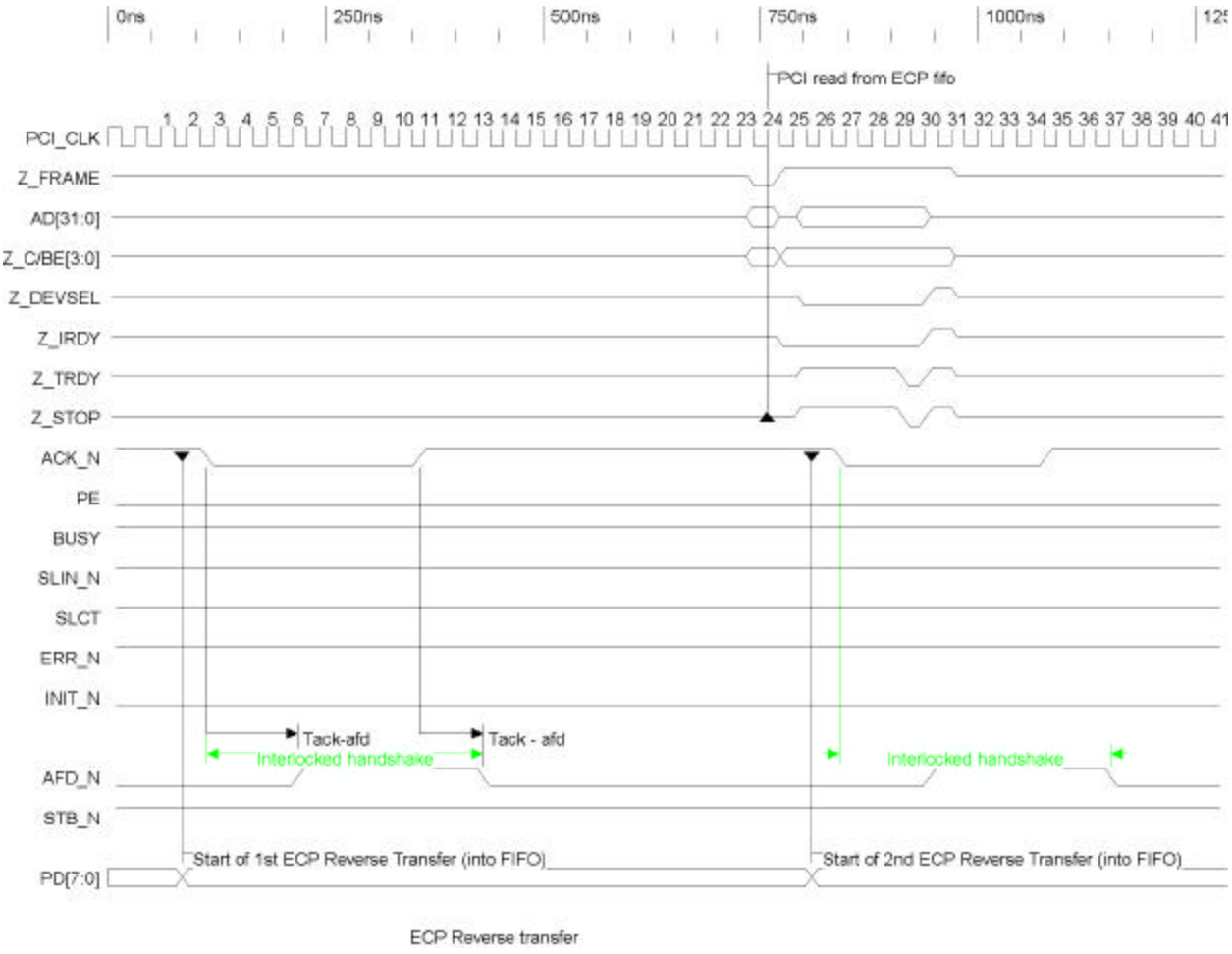
PCI CLK no

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"
9-10	- Peripheral Asserts BUSY in response to the host driving STB_N low
10	- Start of 2nd PCI write to ECP DFIFO register. Current ECP forward transfer remains unaffected.
11	- Host responds to BUSY by de-asserting STB_N (2 clock cycles after sampling BUSY) – 1st ECP forward transfer.
14-15	- 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) - 16ns max*
Tpd (PCI clk to valid port Address) - 16ns max*

ECP Forward Transfer Cycle duration is dependant 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 will be incurred in the response of the host to the peripheral's BUSY line when the filters are enabled.

** These values exclude the effects of external parasitic (board) capacitances.*



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)
10-11 - 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)

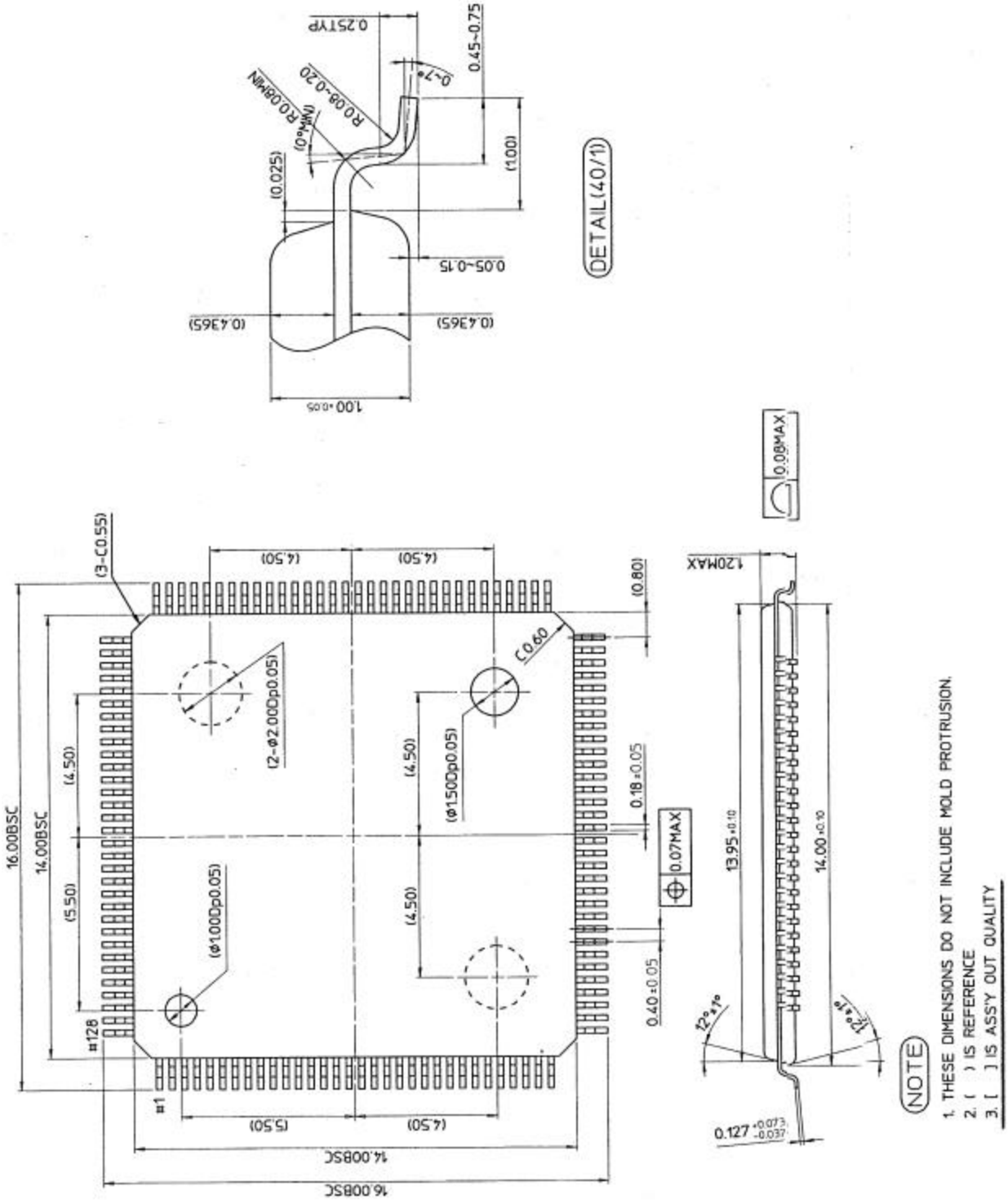
24 - Start of PCI read from ECP DFIFO (does not initiate or affect any ECP reverse transfers that may be taking place)

26 - Start of 2nd ECP reverse transfer by peripheral.

Tafd (PCI clk to AFD_N valid) : 16ns max*
Tack-afd : 3 clock cycles (see below) + Tafd

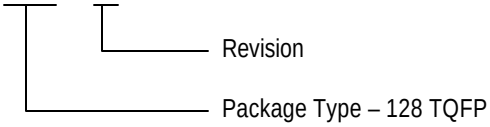
ECP Reverse Transfer Cycle duration is dependant upon the timing response of the peripheral's ACK_N line and the parallel port filters.Example waveform has the parallel port filters disabled. An extra 2 PCI CLK cycles will be incurred in the response of the host to the peripheral's ACK_N line when the filters are enabled.

14 PACKAGE INFORMATION



15 ORDERING INFORMATION

OX16PCI952-TQFP - A



16 CONTACT DETAILS

Oxford Semiconductor Ltd.
25 Milton Park
Abingdon
Oxfordshire
OX14 4SH
United Kingdom

Telephone: +44 (0)1235 824900
Fax: +44 (0)1235 821141
Sales e-mail: sales@oxsemi.com
Web site: http://www.oxsemi.com

DISCLAIMER

Oxford Semiconductor believes the information contained in this document to be accurate and reliable. However, it is subject to change without notice. No responsibility is assumed by Oxford Semiconductor for its use, nor for infringement of patents or other rights of third parties. No part of this publication may be reproduced, or transmitted in any form or by any means without the prior consent of Oxford Semiconductor Ltd. Oxford Semiconductor's terms and conditions of sale apply at all times.