OXFORD SEMICONDUCTOR

OXmPCI954

Integrated High Performance 3.3V Quad UARTs, 8-bit Local Bus/Parallel Port PCI interface.

EVALUATION BOARD

This document describes the OXmPCI954 3.3V evaluation board, how to set its configuration, install drivers and troubleshoot common problems.

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

REVISION HISTORY

REV	DATE	REASON FOR CHANGE / SUMMARY OF CHANGE

Parts List for OXmPCI954 Evaluation Board

On receiving the OXmPCI954 evaluation board the following items should be found within it: -

Component	Quantity
8 or 4 Way Serial cable	1

FEATURES

- Software compatible with OX16PCI954
- Four 16C950 High performance UART channels
- 8-bit Pass-through Local Bus (*PCI Bridge*)
- IEEE1284 Compliant SPP/EPP/ECP parallel port *(with external transceiver)*
- Efficient 32-bit, 33MHz, Multi-function target-only PCI controller, fully compliant to PCI Local Bus specification 3.0, and PCI Power Management Specification 1.1
- PCI and miniPCI Modes *(with CLKRUN# and D3cold support in miniPCI modes)*
- UARTs fully software compatible with 16C550-type devices.
- UART operation up to 60 MHz via external clock source. Up to 20MHz with the crystal oscillator.
- Baud rates up to 60Mbps in external 1x clock mode and 15Mbps in asynchronous mode.
- 128-byte deep FIFO per transmitter and receiver
- Flexible clock prescaler, from 1 to 31.875
- Operation via IO or memory mapping.
- 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
- Global Interrupt Status and readable FIFO levels to facilitate implementation of efficient device drivers.
- Detection of bad data in the receiver FIFO
- 9-bit data framing, as well as 5,6,7 and 8 bits
- Local registers to provide status/control of device functions.
- 12 multi-purpose I/O pins, which can be configured as input interrupt pins or 'wake-up'.
- Auto-detection of a wide range of optional MicrowireTM compatible EEPROMs, to re-configure device parameters.
- Function access, to pre-configure each function prior to handover to generic device drivers.
- 3.3V operation (5v tolerance on selected I/Os)

DESCRIPTION

The OXmPCI954 is a single chip solution for PCI and mini-PCI based serial and parallel expansion add-in cards. **It should be noted that this is a 3.3V card. At present this card slot seems to be only available in server PC and motherboards designed for these high-end machines.**
It is a dual function PCI device, where function 0 offers four ultra-high performance UARTs, and function 1 is configurable either as an 8-bit Local Bus or a bidirectional parallel port.
Each UART channel in the OXmPCI954 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.
To enhance device driver efficiency and reduce interrupt latency, internal UARTs have multi-port features such as shadowed FIFO fill levels, a global interrupt source register and Good-Data Status, readable in four adjacent DWORD registers visible to logical functions in I/O space and memory space.

Expansion of serial cards beyond four channels is possible using the 8-bit pass-through Local Bus function. The addressable space can be increased up to 256 bytes, and divided into four chip-select regions. This flexible expansion scheme caters for cards with up to 20 serial ports using external 16C950, 16C952, 16C954 or compatible devices, or composite applications such as combined serial and parallel port expansion cards. Serial port cards with up to 20 ports (or with 4 serial ports and a parallel port) can be designed without redefining any device or timing parameters.
The parallel port is an IEEE 1284 compliant SPP/EPP/ECP parallel port that fully supports the existing Centronics interface. The parallel port can be enabled in place of the Local Bus. A*n external bus transceiver is required for 5Vparallel port operation*.
For applications that do not require the local bus, or the parallel port, card designers can assign a Subsystem Vendor ID and a Subsystem ID directly using 32 input pins.
For full flexibility, all the default configuration register values can be overwritten using an optional MicrowireTM compatible serial EEPROM. This EEPROM can also be used to provide *function access* to pre-configure each UART into enhanced modes or pre-configure devices on the local bus/parallel port, prior to any PCI configuration accesses and before control is handed to (generic) device drivers.

TABLE OF CONTENTS

1 EVALUATION BOARD DESCRIPTION

This document SHOULD be read before attempting to use the evaluation board in order to prevent simple setup errors occurring. The evaluation board for OXmPCI954 users provides an environment in which the various modes and features of the OXmPCI954 device can be demonstrated. For more details on the features and technical specification of the device consult the datasheet for the device which can be downloaded from the Oxsemi website. A reference schematic for the OXmPCI954 evaluation board is available on request from Oxford Semiconductor. The evaluation board provides a simple means of realising the following products:-

- 8-port serial card (using four internal UARTs and an OX16C954 on a local bus connection). This is operationally backward compatible with our previous OXPCI954 device.
- 4-port serial / 1 parallel port card (using only functions internal to the OXmPCI954). This is operationally backward compatible with our previous OXPCI954 device.
- 4-port serial card with pin-assignable Subsystem ID and subsystem Vendor ID. This is backward compatible with our previous OXPCI954 device.

The specific features available include:-

- Configuration into any of the device modes is available
- Access to the four OX16C950 UARTs (***2xRS232, 2xRS422***) from a standard 37-way D-type interface
- Use of the 8-bit local bus function to drive an external OX16C954 UART (***2xRS232, 2xRS422***) from a similar interface
- Use of the parallel port function via a standard 25-way D-type header
- A serial EEPROM socket to provide maximum device configurability
- Use of the internal crystal oscillator, or any frequency via a TTL oscillator socket (switchable). The external OX16C954 can be driven from the TTL module, or directly from the UART_Clk_Out pin of the PCI device.
- Use of isochronous mode feature of the internal UARTs via simple connections

The drivers that have been provided for this board have been written for Windows 9*, Windows 2000, Windows NT and Windows XP. There are drivers available for Linux but these ***HAVE NOT*** been written by Oxford Semiconductor we cannot therefore be held responsible nor do we provide updates or support for these.

2 BOARD LAYOUT

The layout of the evaluation board is shown in figure 1.

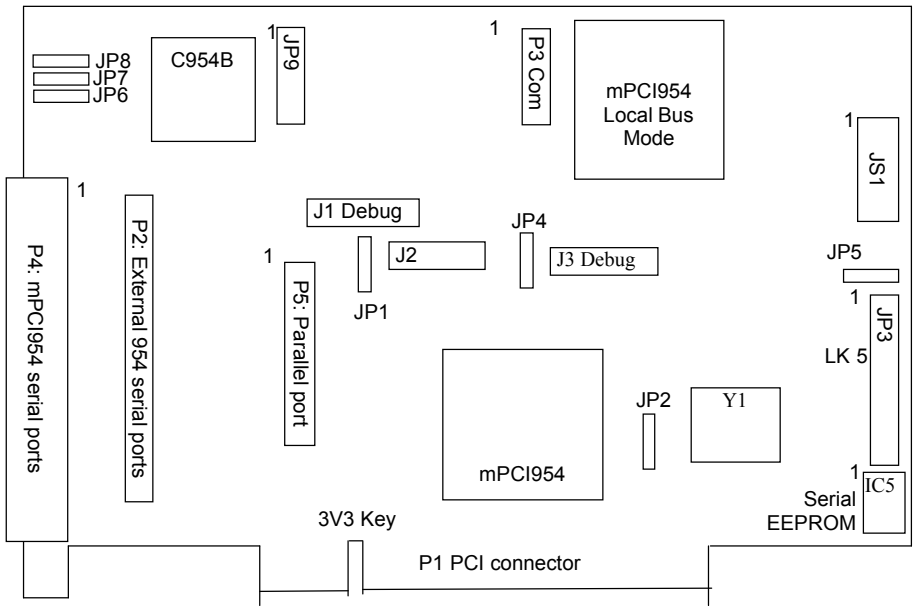


Figure 1: Evaluation board layout

Key	**Function**
J1:	Test pin header - Local Bus pins
J2:	Test pin header - MIO and Serial pins from mPCI954 (Local bus mode)
J3:	Test pin header - All serial pins from mPCI954 (Main)
P1:	3V3 PCI Connector
P2:	Header for access to external UART serial com ports
P3:	Header for mPCI954 (Local bus mode) serial com ports
P4:	37-way 'D' connector for access to PCI UART serial ports (FEMALE D-type)
P5:	Header for parallel port
JP1:	Voltage selection of local bus (5V of 3V3 clamped)
JP2:	Select internal oscillator or TTL module
JP3:	Configuration header
JP4:	MIO11 (mPCI mode) configuration selection
JP5:	Connect RI from PCI UART channel 0 or external UART channel 0 to MIO0
JP6:	Select TTL module or LB_Clk_Out [UART clock out of mPCI954]
JP7 & JP8:	5V / 3V Operation selection for the C954
JP9:	DTR/DSR header for RS422 channels
JS1:	Socket for TTL module
Y1:	Crystal Oscillator
IC5:	Socket for serial EEPROM

All links settings are detailed in the section on CONFIGURATION. In addition to the evaluation board a multi-way serial adaptor is provided which allows 4/8 serial connections to be made.

3 OPERATION

In any mode the device requires an external clock. Ensure a jumper is fitted to JP2, selecting either the TTL module which must be fitted or the UART_Clk_Out of the OXmPCI954; this must be enabled by writing to the configuration register before it can be used. This requires an EEPROM and the Oxford Semiconductor utility program Oxprom.

To use the device as a **Quad UART + 8-bit local bus**, set the mode pins to '000' via JP3. For both the PCI UARTs and the local bus UARTs, channels 0 and 1 are RS232 connections, and channels 2 and 3 are RS422 connections.

The DTR and DSR signals of the RS422 ports are available via the JP9 header – an extra cable is needed if these lines are required. Basic hardware flow control can be achieved without these; however they will be required for DTR/DSR flow control or isochronous mode operation.

Connect the extra 37-way D connector to P4, and if the RS422 channels are required enable them via JP3.

Jumper JP1 allows the Local bus voltage setting to be selected. Linked the local bus isolators are run from 5V. If not linked the local bus isolators are run from 3.3V

Jumper JP7 determines what voltage the OX16C954 uses as its VDD and allows the RS232 transceivers to also be run at either 3.3V or 5V. Selection is user and application dependent.

Jumper JP8 is the 3.3v or 5.0v I/O buffer selection for the OX16C954B. This pin must be tied according to the voltage supply used to power the TQFP package option.
For 5v supply voltage: Vdetect must be tied low.
For 3.3v supply voltage: Vdetect must be tied high.

The board is now ready for use.

To use the device as a **Quad UART + parallel port**, set the mode pins to '001' via JP3. The UART channels interface is as above. Connect the parallel port header to P5, and if the UART's RS422 channels are required enable them also via JP3.

Connect the extra 37-way D connector to P4, and if the RS422 channels are required enable them via JP3.

The board is now ready for use.

To configure the OXmPCI954 via the **serial EEPROM**, merely insert the EEPROM into its socket IC5. All features of the device can be programmed into the PROM via the OXPROM.EXE utility. If an EEPROM has been fitted it WILL not have been programmed with any data and will therefore read back 0xFF when OPROM is used.

The PCI UART can use the internal crystal oscillator or an external TTL clock source. Select the mode required using the JP2 link. The external UART can also use the TTL source or the UART_Clk_Out line from the OXmPCI954. Select the mode required, and enable/disable the UART_Clk_Out line as appropriate via the local configuration registers. (This can be done in the serial EEPROM).

To use the Local bus in **Motorola mode**, short the Intel/Motorola link on the JP3 header. In this mode the default Local bus timing values should be changed in the local configuration registers as follows:-

Read-not-Write De-assertion = 4
Read Data-strobe assertion = 1
Write data-strobe assertion = 1
Write data-strobe de-assertion = 3

Also, MIO [8] in the MIO configuration registers should be configured to an inverting input, and MIO [9-10] masked off. An EEPROM configuration file, motorola.dat has been provided for this purpose.

4 DRIVER INSTALLATIONS

For detailed instructions on how to install the reference drivers, please refer to the document “Reference Drivers.pdf” document. The document ADVANCED TOPICS.pdf describes the use of advanced features of the Oxford Semiconductor reference drivers for the OX16C95x high performance UART family. If particular problems are being encountered with baud rates and the use of drivers options request Application note 15 ‘Setting UART baud rates under Windows’ from Oxford Semiconductor.

5 CONFIGURATION

The board and device are configured using the configuration header JP3. The pin out is shown in Figure 2. (All links are shown oriented with Figure 1).

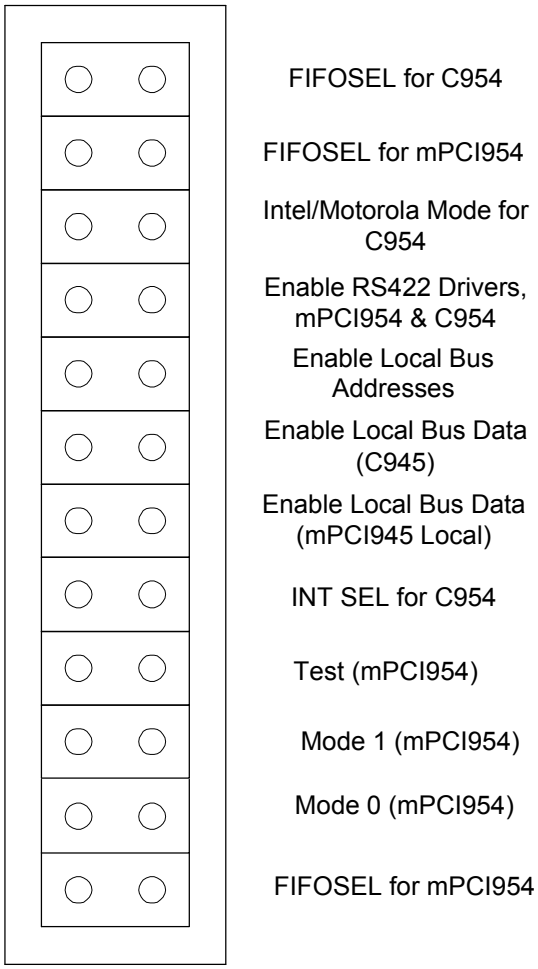


Figure 2: Configuration header JP3

The table below shows the function of each pair of pins for JP3

Pins	Open	Short
External FIFOSEL# for C954	External UART has 16 byte deep FIFO	External UART has 128 byte deep FIFO
FIFOSEL for mPCI954 (Local Mode)	128 byte deep FIFO	16 byte deep FIFO
Intel/Motorola# mode for C954	Intel mode	Motorola mode
Enable RS422 Drivers for mPCI954 & C954	Disable RS422 line drivers	Enable RS422 line drivers
Enable local bus addresses	LBA Disabled	LBA Enabled
Enable local bus Data (C954)	LBD Disabled	LBD Enabled
Enable local bus Data (mPCI954 Local)	LBD Enabled	LBD Disabled
INT SEL	External UART interrupts enabled via MCR[3]	External UART interrupts enabled
TEST (mPCI954)	Depends on device mode required	Depends on device mode required
MODE1 (mPCI954)	Mode[1] = 0	Mode[1] = 1
MODE0 (mPCI954)	Mode[0] = 0	Mode[0] = 1
FIFOSEL for mPCI954	16 byte deep FIFO	128 byte deep FIFO

The OXmPCI954 supports several modes of operation.

Three modes of the device are (software) *backwards compatible* with the OXPCI954 device. There are a further 3 modes that are *enhanced* modes, which offer additional features over those available in the *backwards compatible* modes. Then, there is a *standalone mode* that allows a synchronous local bus access to the internal UARTs, without any form of PCI transactions. These modes are summarized in the following table.

Device Mode	Mode Pin Selection	Functionality	Backwards Compatible / Enhanced Modes
000	*MODE(2:0) = 000*	**Function 0** : *QUAD Uarts* **Function 1** : *8-bit Local Bus*	*Backwards Compatible**
001	*MODE(2:0) = 001*	**Function 0** : *QUAD Uarts* **Function 1** : *Parallel Port*	*Backwards Compatible**
010	*MODE(2:0) = 010*	**Function 0** : *QUAD Uarts* *Subsys ID/Subsys Vendor ID via Device Pins*	*Backwards Compatible**
011	*MODE(2:0) = 011*	**Function 0** : *QUAD Uarts (Unique BARs)* **Function 1** : *8-bit Local Bus*	*Enhanced Mode*
100	*MODE(2:0) = 100*	**Function 0** : *QUAD Uarts* **Function 1** : *8-bit Local Bus*	*Enhanced Mode*
101	*MODE(2:0) = 101*	**Function 0** : *QUAD Uarts* **Function 1** : *Parallel Port*	*Enhanced Mode*
110	*MODE(2:0) = 110*	TestMode (Reserved).	N/A
111	*MODE(2:0) = 111*	***Standalone Mode***	N/A

** The OXmPCI954 is not a direct 'drop-in' replacement part for the OXPCI954 owing to a small pinout change and voltage. The device is only S/W compatible.*

In the *Enhanced Modes*, the following additional features are made available over the underlying functionality of the device.

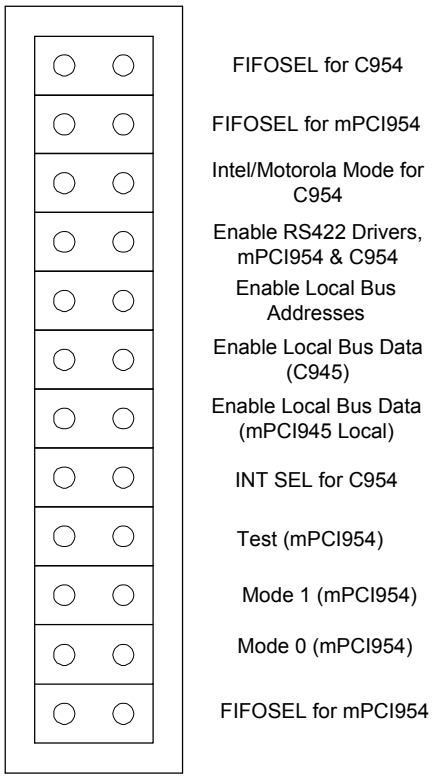
- Pin 88 (MIO[11]) is re-defined as a *PCI/miniPCI Mode Selection Pin*.
  Functionality normally associated with the pin MIO[11] is no longer available. This results in the pins MIO[10:0] serving as Multi-purpose I/O pins.
- All Function 0 and Function 1 interrupts assert on the INTA# pin (*by default).*
- Local Registers provide additional Controls and Status Indication.
- Function 0 option to allow each UART to be separately addressable via its own Base Address Register (in I/O Space)
  This option can be exercised by the device pins (MODE 011) or by using the external EEPROM to set a specified field in the local registers.
- Function 0 and Function 1 Power Management Registers indicate Compatibility to Power Management Specification 1.1
- Each function implements Power Management Data/Data Scale fields, *returning user defined data*.
- Availability of 2 Additional EEPROM Zones: The *Power Management Data* Zone and the *Function Access* Zone.

Specifically for *MiniPCI Selection (Pin 88 = '1' in Enhanced Modes)*

- Device pin INTB#' is re-defined as a CLKRUN# pin. Compliant to PCI Mobile Design Guide revision 1.1
  For PCI modes (Pin 88 = '0' in Enhanced Modes) INTB# is an unused PCI interrupt line.
- Device supports PME# generation from the D3cold state and preserves PME# context. This is compliant to Mini PCI Specification, revision 1.0

5.1 How to set-up the evaluation board for 8 serial ports

Set the pins on JP3 as follows:



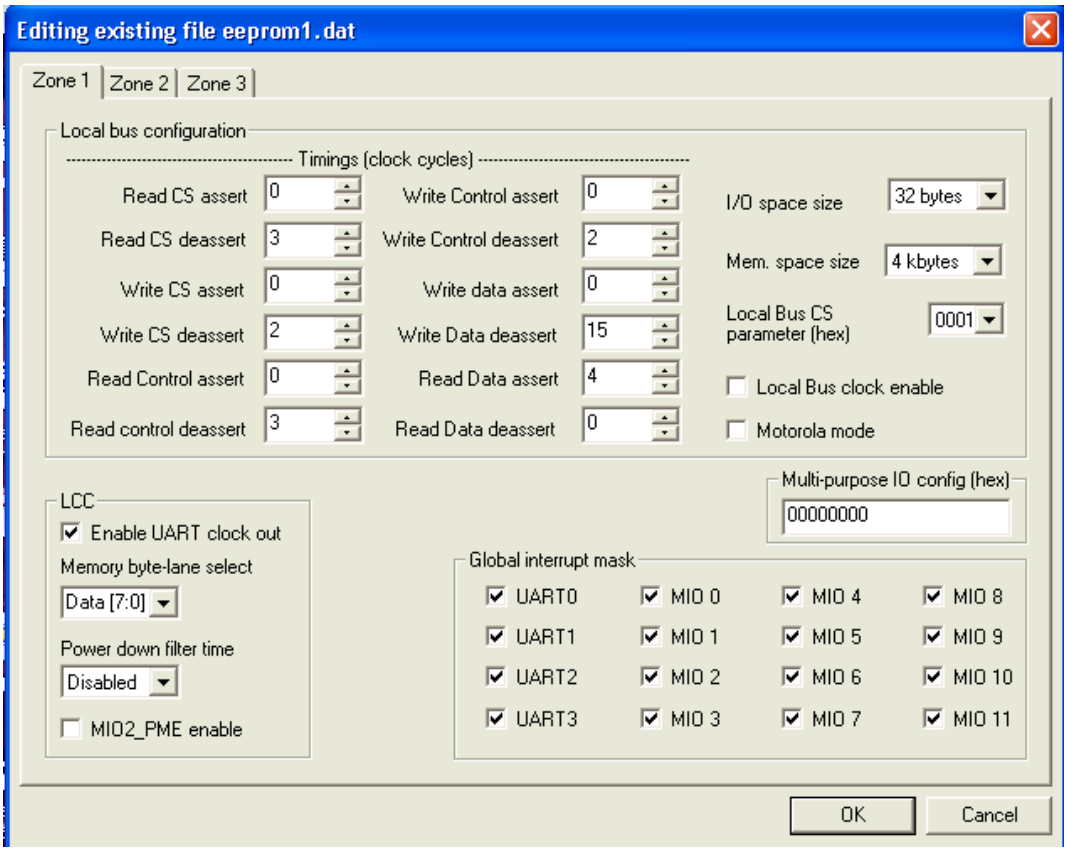
Pins	Open	Short
External FIFOSEL# for C954	External UART has 16 byte deep FIFO	External UART has 128 byte deep FIFO
FIFOSEL for mPCI954 (Local Mode)	128 byte deep FIFO	16 byte deep FIFO
Intel/Motorola# mode for C954	Intel mode	
Enable RS422 Drivers for mPCI954 & C954		Enable RS422 line drivers
Enable local bus addresses		LBA Enabled
Enable local bus Data (C954)		LBD Enabled
Enable local bus Data (mPCI954 Local)		LBD Disabled
INT SEL	External UART interrupts enabled via MCR[3]	External UART interrupts enabled
TEST (mPCI954)	Open connection	
MODE1 (mPCI954)	Mode[1] = 0	
MODE0 (mPCI954)	Mode[0] = 0	
FIFOSEL for mPCI954	16 byte deep FIFO	128 byte deep FIFO

Where both columns have been left filled in it means that it is an either or situation.

Device Mode	Mode Pin Selection	Functionality	Backwards Compatible / Enhanced Modes
000	*MODE(2:0) = 000*	**Function 0** : *QUAD Uarts* **Function 1** : *8-bit Local Bus*	*Backwards Compatible**

5.1.1 Points to Note about Local bus Operation and clock operation

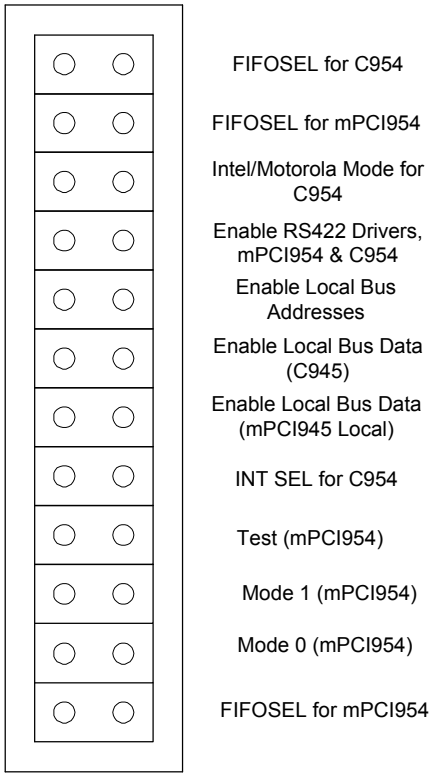
Bit 2 of the Local Configuration and Control register 'LCC' (Offset 0x00) enables the UART clock output. When this bit is set, the buffered UART clock output pin (UART_CLK_Out) is active. When low the UART_CLK_Out is permanently low. **By DEFAULT it is LOW** and therefore needs to be enabled. In order to do this an EEPROM would be required and used in conjunction with OXPROM as shown below:-



Check – Enable UART clock out. This means that a single oscillator can be used to drive serial ports on the local bus as well as the internal UARTs.

5.2 How to set-up the evaluation board for 4 serial ports and a parallel port

Set the pins on JP3 as follows:



Pins	Open	Short
External FIFOSEL# for C954	External UART has 16 byte deep FIFO	External UART has 128 byte deep FIFO
FIFOSEL for mPCI954 (Local Mode)	128 byte deep FIFO	16 byte deep FIFO
Intel/Motorola# mode for C954	Intel mode	
Enable RS422 Drivers for mPCI954 & C954		Enable RS422 line drivers
Enable local bus addresses	LBA Disabled	
Enable local bus Data (C954)	LBD Disabled	
Enable local bus Data (mPCI954 Local)		LBD Disabled
INT SEL	External UART interrupts enabled via MCR[3]	External UART interrupts enabled
TEST (mPCI954)	Open connection	
MODE1 (mPCI954)	Mode[1] = 0	
MODE0 (mPCI954)		Mode[0] = 1
FIFOSEL for mPCI954	16 byte deep FIFO	128 byte deep FIFO

Where both columns have been left filled in it means that it is an either or situation.

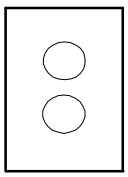
Device Mode	Mode Pin Selection	Functionality	Backwards Compatible / Enhanced Modes
001	*MODE(2:0) = 001*	**Function 0** : *QUAD Uarts* **Function 1** : *Parallel Port*	*Backwards Compatible**

6 MISCELLANEOUS LINK SETTINGS

The following sections detail the additional miscellaneous links available on the OXmPCI954. These can be used to observe the various operational features of the OXmPCI954. The function of each jumper/link has been described in this section and the user of the evaluation board is free to use these.

6.1 JP1 Local bus voltage setting

Jumper JP1 allows the Local bus voltage setting to be selected. Linked the local bus isolators are run from 5V. If not linked the local bus isolators are run from 3.3V

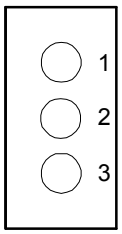


Jumpered-Allow 5V signals

Non-Jumpered-Voltages 3v3 Clamped

6.2 JP2 Clock select for OXmPCI954

The JP2 link selects the oscillator type for the OXmPCI954.



TTL Oscillator

Crystal

Select desired oscillator by linking from centre to the pin required.

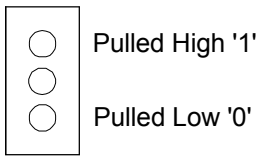
6.3 JP4 MIO11, mPCI (Enhanced mode) selection

The JP4 link ties MIO11 either ‘high’ or ‘low’. MIO11 has been re-defined as a PCI/miniPCI Mode Selection Pin and is specifically for MiniPCI Selection (Pin 88 = ‘1’ in Enhanced Modes)

> *When in this mode the device pin INTB#’ is re-defined as a CLKRUN# pin. Compliant to PCI Mobile Design Guide revision 1.1*

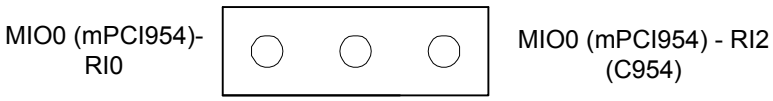
For PCI modes (Pin 88 = ‘0’ in Enhanced Modes) INTB# is an unused PCI interrupt line.

Device supports PME# generation from the D3cold state and preserves PME# context. This is compliant to Mini PCI Specification, revision 1.0



6.4 JP5 Ring Indicator Input Selection

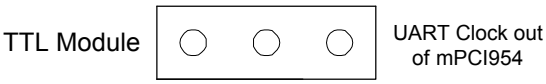
The JP5 link connects MIO0 to RI of PCI UART channel 0 or external UART channel 0.



Select RI desired by linking centre to that desired. Default = no link. If this is used, MIO [0] should be reconfigured in the local configuration registers otherwise an interrupt on function 0 will be constantly present.

6.5 JP6 Clock select for C954 and OXmPCI954 (Local mode)

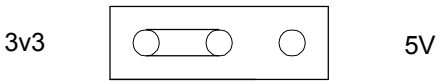
The JP6 clock select link selects the oscillator source for the external UART



Select desired oscillator by linking from centre to the pin required

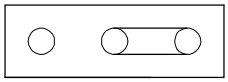
6.6 JP7 and JP8 Voltage selection for the C954

Jumper JP7 determines what voltage the OX16C954B uses as its VDD and allows the RS232 transceivers to also be run at either 3.3V or 5V. Selection is user and application dependent. The diagram below shows JP7 linked for a voltage selection of 3v3.



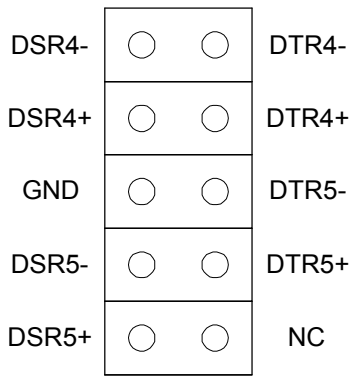
Jumper JP8 is the 3.3v or 5.0v I/O buffer selection for the C954B. This pin must be tied according to the voltage supply used to power the TQFP package option.
For 5v supply voltage : Vdetect must be tied low.
For 3.3v supply voltage : Vdetect must be tied high.

The diagram below shows JP8 linked for 3v3.



6.7 JP9 RS422 flow control links

The lines provided on the RS422 connectors only allow for CTS/RTS flow control. If DTR/DSR flow control is needed, JP9 provides access to the extra hardware lines. The pin out is as follows:-



Pinout of RS422 ports:
RS422 supports much higher data transfer rates due to its differential signalling. The RS422 ports are connected as follows on the 9-way D-connectors:-

Pin	Signal
1	TXD-
2	TXD+
3	RTS-
4	RTS+
5	GND
6	6 RXD-
7	RXD+
8	CTS-
9	CTS+

6.8 Pinout of P4 [37-way D-type connector]

The following pin description is the same for the PCI internal UARTs and the four ports from the OX16C954 local bus device

37-Way D-Type pin number	Port Number / Type	9-Way D-Type pin number	Description
1	Port 1 / RS232	1	DCD
20	Port 1 / RS232	6	DSR
2	Port 1 / RS232	2	RxD
21	Port 1 / RS232	7	RTS
3	Port 1 / RS232	3	TxD
22	Port 1 / RS232	8	CTS
4	Port 1 / RS232	4	DTR
23	Port 1 / RS232	9	RI
5	Port 1 / RS232	5	GND
24	Port 2 / RS232	1	DCD
6	Port 2 / RS232	6	DSR
25	Port 2 / RS232	2	RxD
7	Port 2 / RS232	7	RTS
26	Port 2 / RS232	3	TxD
8	Port 2 / RS232	8	CTS
27	Port 2 / RS232	4	DTR
9	Port 2 / RS232	9	RI
28	Port 2 / RS232	5	GND
10	Port 3 / RS422	1	DCD
29	Port 3 / RS422	6	DSR
11	Port 3 / RS422	2	RxD
30	Port 3 / RS422	7	RTS
12	Port 3 / RS422	3	TxD
31	Port 3 / RS422	8	CTS
13	Port 3 / RS422	4	DTR
32	Port 3 / RS422	9	RI
14	Port 3 / RS422	5	GND
33	Port 4 / RS422	1	DCD
15	Port 4 / RS422	6	DSR
34	Port 4 / RS422	2	RxD
16	Port 4 / RS422	7	RTS
35	Port 4 / RS422	3	TxD
17	Port 4 / RS422	8	CTS
36	Port 4 / RS422	4	DTR
18	Port 4 / RS422	9	RI
37	Port 4 / RS422	5	GND
19	NC	NC	NC

7 TROUBLESHOOTING COMMON PROBLEMS

Communication Issues

Problem: The system detects the ports but a terminal application will not recognise them.

Answer: Ensure that the application supports COM ports other than the four legacy addresses (3f8, 2f8, 3e8, 2e8). If not, use an application that supports all COM devices, such as Quarterdeck Procomm Plus, or Ericom PowerTerm.

Problem: Some or all of the ports will not communicate with other devices

Answer: Ensure that the baud rates are correct. The OX16C95x family of UARTs has a very flexible baud rate generator, which can accept differing crystal frequencies, prescalar values etc. The serial port configuration utility in the Windows 9x driver can set baud rate multipliers etc. or completely override an application's baud rate setting.

Answer: Two of the ports on each UART chip have RS422 line drivers (note only on evaluation board). These support higher data transfer rates, but will not interface to RS232 ports. Ensure that the correct ports are being used.

Answer: Ensure that the port has a clock signal, either from the OXmPCI954 oscillator, or from the TTL clock module. This is selected with XTALsel (internal UARTs) and CLKsel (local bus UARTs)

Problem: Doing a loop back test with a terminal program there is no response.

Answer: On the evaluation board port 1 and 2 are RS232 while ports 3 and 4 are RS422 therefore ensure the correct loop back set up is used.

Answer: Ensure the loop back connector has been wired up in correctly.

Problem: The parallel port is not recognised.

Answer: Ensure that the correct mode (*MODE(2:0) = 001)* has been selected.

Problem: When the board is installed in a system, it complains of a resource clash.

Answer: PCI devices should be able to share interrupts; however, not all other devices have drivers written to handle this. If another device is using one of the interrupts which is being assigned to the OXmPCI954, there are various things that can be done to work around this problem.

1) Move the OXmPCI954 board to a different PCI slot

2) Change the settings in your system BIOS to allocate different resources (not all BIOS's have this feature)

3) Use the serial EEPROM to only allocate one interrupt to the two OXmPCI954 functions

4) Disable any non-critical device that is using the same interrupt – for instance USB controllers can often not share resources adequately.

Problem: Are errors are being detected at baud rates of 300 baud and less?

Answer: The transmit FIFO length (=4) and receive FIFO length (=100) are set to these values by default. For the lower baud rates (300 or 75), the difference in

length between the transmit and receive FIFOs should be greater than (127-4) =123. Making this alteration should result in it working correctly.

Problem: How do I support flow control?

Answer: If you need to do this, set MCR (1) and ECR (7:6) then the h/w will take care of it. On top of this, if you set IER (7:6), then the device will assert a level 6 into whenever CTS and RTS toggle.

Driver Issues

Problem: If the EEPROM is programmed with an invalid image it may not load the driver or a message from the OS may appear saying that there is an error of some description

Answer: In order to get the OS to accept the card the following steps are necessary.

1. Unplug the board from the system.
2. Locate the EEPROM on the card unsolder pin 4 [DO] (This will not allow the EEPROM to be interrogated by the OS as the data out line is effectively disconnected)
3. Plug the board back in. The system will now pick the default settings in the driver and not be overwritten by the contents of the EEPROM.
4. Use OXPROM to erase the EEPROM
5. Remove board from the system
6. Re-solder pin 4 [DO]
7. Once the system has recognised the card and installed all the appropriate driver files the board can be used.
8. The EEPROM can then be reprogrammed using one of the .dat files provided for the OXmPCI954.

Problem: I have downloaded the drivers and get the following message on the event viewer – Device not found.

Answer: This may happen if either the vendor ID or subsystem ID has been changed using OXPROM. Our device driver looks for these fields and will not work if they have been changed. For example the line below shows a vendor id of 1415 and no subsystem id being used.

```
[ControlFlags]
ExcludeFromSelect=PCI\VEN_1415&DEV_9501
ExcludeFromSelect=PCI\VEN_1415&DEV_9511
ExcludeFromSelect=PCI\VEN_1415&DEV_9521
ExcludeFromSelect=PCI\VEN_1415&DEV_950A
ExcludeFromSelect=PCI\VEN_1415&DEV_950B
ExcludeFromSelect=PCI\VEN_1415&DEV_8401
ExcludeFromSelect=PCI\VEN_1415&DEV_9512
ExcludeFromSelect=PCI\VEN_1415&DEV_9510
ExcludeFromSelect=PCMCIA\PC_CARD-GENERIC-1748

; Drivers
;-----------------------------------------------------------

[Manufacturer]
%mfg%=mfg

; List of available devices
[mfg]
```

```
%PCI\VEN_1415&DEV_9501.DeviceDesc% = PCI_9501,
  PCI\VEN_1415&DEV_9501
%PCI\VEN_1415&DEV_9511.DeviceDesc% = PCI_9511,
  PCI\VEN_1415&DEV_9511
%PCI\VEN_1415&DEV_950A.DeviceDesc% = PCI_950A,
  PCI\VEN_1415&DEV_950A
%PCI\VEN_1415&DEV_950B.DeviceDesc% = PCI_950B,
  PCI\VEN_1415&DEV_950B
%PCI\VEN_1415&DEV_8401.DeviceDesc% = PCI_8401,
  PCI\VEN_1415&DEV_8401
%PCI\VEN_1415&DEV_9521.DeviceDesc% = PCI_9521,
  PCI\VEN_1415&DEV_9521

%PCI\VEN_1415&DEV_9512.DeviceDesc% = PCI_NULL,
  PCI\VEN_1415&DEV_9512
%PCI\VEN_1415&DEV_9500.DeviceDesc% = PCI_NULL,
  PCI\VEN_1415&DEV_9500
%PCI\VEN_1415&DEV_9510.DeviceDesc% = PCI_NULL,
  PCI\VEN_1415&DEV_9510

%Card0.DeviceDesc% = CF_950, PCMCIA\PC_CARD-GENERIC-1748
```

If changes are made to the vendor and subsystem id then these will have to be changed in the appropriate .inf files for a driver to be loaded.

Problem: I get the following error message “The clock for the COM9 was not detected, default to 1843200 Hz”.

Answer: As Windows 98 uses 16 bit DLLs, if the COM port is >8 it cannot do auto detect. You have to select the device speed manually in Device Manager under Data Rates.

Problem: Will the drivers support 4 x PCI cards?

Answer: Yes.

Problem: The ports are not being installed correctly in Windows 2000.

Answer:

1. Check all the driver binaries are present in \winnt\system32\drivers
2. Check one of the settings, FIFO or data rate tabs - if it can see the UART it should say in the top right, something like 16C950B
3. If our development board is not being used then the UARTS may have mapped to different memory addresses, then (only if it’s not an Oxsemi development board) recompile oxmfuf.sys to use I/O mapping - it will be a bit slower but doesn't have to work out the memory base address.

Hardware Issues

Problem: How do I reduce the quiescent current during standby?

Answer: To reduce the quiescent current tie all pins as the internal pull ups and downs as given in the data sheet. We do not have any leakage figures from the oscillator, but this should be tied to a rail and not left floating.

Problem: I cannot set the RS485 driver enables using DTR.

Answer: 2 things are needed to enable RS485, the MCr and ACR registers. Set ACR first and then MCR. Then the chip automatically asserts DTR to transmit.

Problem: At high baud rates the data throughput is very slow.

Answer: The application requests a port at a certain speed. The driver works out which divider to use based on the crystal speed (amongst other things). If the port speed cannot be achieved within 2%, then it goes slowly. The crystal needs to be a multiple of 1.8432MHz

Problem: With my board set-up for 8 serial ports my PC either resets or blue screens as a consequence.
Answer: Bit 2 of the Local Configuration and Control register 'LCC' (Offset 0x00) enables the UART clock output. When this bit is set, the buffered UART clock output pin (UART_CLK_Out) is active. When low the UART_CLK_Out is permanently low. **By DEFAULT it is LOW** and therefore needs to be enabled. In order to do this an EEPROM would be required and used in conjunction with OXPROM (see section 5.1.1).

8 REFERENCE DRIVERS

The all the drivers are available from the Oxsemi web site at http://www.oxsemi.com under **Product information** UART / parallel port and PCI. These areas will contain a reference driver for Windows 9x , Windows 2000, and Windows NT4. It also contains related utilities and documentation. The contents are listed below:-

\	oxpar.inf - driver .inf file oxpar2.inf - driver .inf file (2) oxpci.inf - driver .inf file oxpci2.inf - driver .inf file (2) oxser.inf - driver .inf file oxmep.sys - PCI support driver oxmf.vxd/sys - bus enumerator oxmfuf.sys - Upper filter driver oxpar.sys - port driver oxpp.vxd - parallel port enumerator oxser.vxd/sys - port driver oxppui.dll - parallel port configuration utility oxserui.dll - driver configuration utility oxui.dll - driver configuration utility
\ oxprom	oxprom.exe - Serial EEPROM programming utility oxeeprom.sys - needed by oxprom.exe .dat files - example EEPROM configuration files
\utils	readuart.exe - dump UART registers from within Win9x. poke.exe - memory access utility .vxd files - needed by above utilities
\doc	OX16PCI954 Eval Board.pdf - this file advancedtopics.pdf - advanced baud rate configuration in Windows 2000 customdrivers.pdf - details on customising drivers referencedrivers.pdf - driver installation and use
\NT4_Parallel	Install_Parallel.exe - install program for Windows NT4 parallel port driver oxpar.inf - installation script oxpar.sys - port driver licence.txt - licence agreement
\NT4_Serial	Install_Serial.exe - install program for Windows NT4 serial port driver oxser.inf - installation script oxser.sys - port driver licence.txt - licence agreement

These existing drivers are fine with everything except with unique BAR mode a new release of the drivers is due to be released and can be obtained from our web-site.

9 NOTES

This page has been intentionally left blank.

APPENDIX A: OXPROM - EEPROM PROGRAMMING UTILITY

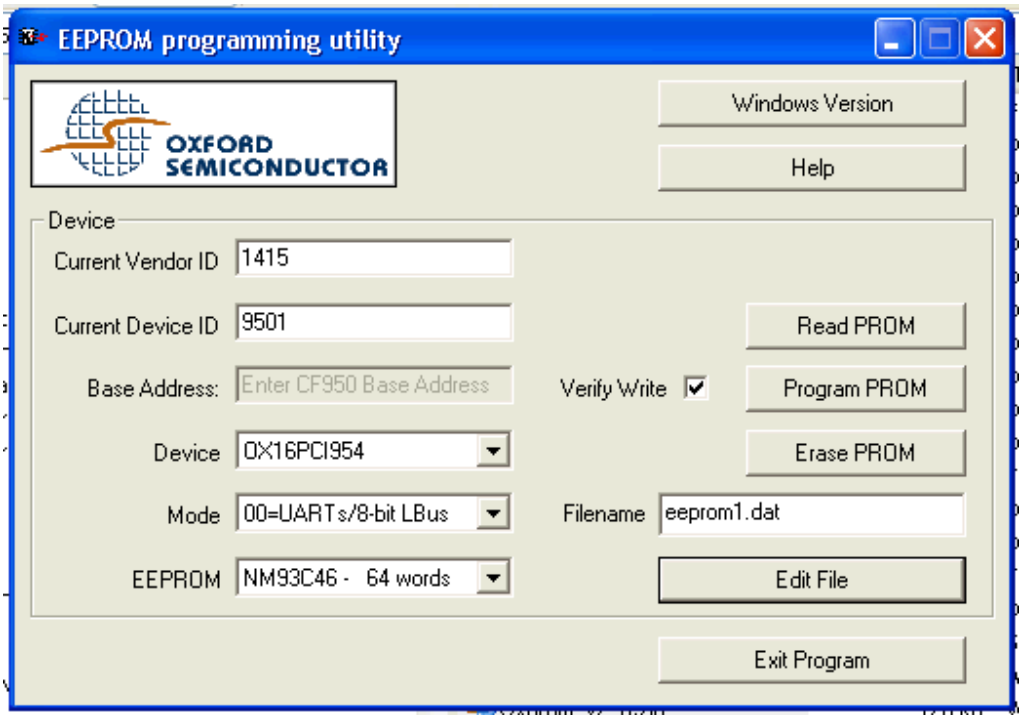
This program enables card designers to configure Oxford Semiconductor PCI devices by programming the attached EEPROM. It provides a full GUI interface which is used to create a configuration file according to the exact requirements of the vendor. This file can then be used to program the card while it is still in the slot. Then (after a reboot) the device will take on the attributes set in the newly-programmed EEPROM. If an EEPROM has been fitted it WILL not have been programmed with any data and will therefore read back 0xFF when OPROM is used.

A newer version of OxProm is due to be released which will enable the newer features of the device to be implemented. For details on how to use this, consult the OxProm User guide v1.0 which will also be released at the same time.

Instructions for use

The figure below shows the old version of “OxProm” that can be used to access all the older features associated with the OXPCI954 that the OXmPCI954 has inherited.

Select the device being used in your new design, and the mode in which it will be used (dependent on the device used).



Enter a filename which will be used to save the configuration data. Then click "Edit File". If the file does not already exist, the default settings will be loaded into the application interface; otherwise the settings already stored in the file will be loaded for re-editing.

APPENDIX B - CUSTOMISING VENDOR ID AND SUB-SYSTEM ID'S

This describes the recommended procedures to reconfigure the OXmPCI954 hardware and reference drivers in order to identify custom add-in card configurations to device drivers.

OVERVIEW

If a designer wishes to use the OXmPCI954 chip for another purpose, it maybe necessary to change the identification fields in PCI Configuration space. These are the VENDOR ID, DEVICE ID, SUBSYSTEM VENDOR ID and SUBSYSTEM ID fields. In this way, it is possible for the Plug 'n' Play system to identify different boards (even though they use the same chip) and load the correct drivers.
There are three recommended approaches to uniquely identify an add-in card using the OXmPCI954.

1. The best solution is for the board manufacturer to obtain their own VENDOR ID from the PCI special interest group (PCISIG). Then they should use the OXProm to reprogram the VENDOR ID and SUBSYSTEM VENDOR ID fields with this value, and they choose any DEVICE ID and SUBSYSTEM ID value.
2. Another recommended approach is to leave the VENDOR ID and DEVICE ID as the default values set by Oxford Semiconductor. The SUBSYSTEM VENDOR ID should be set to board manufacturer's unique value assigned by the PCISIG, and then they can choose the SUBSYSTEM ID.
3. For small vendors, it is sometimes possible to use Oxford Semiconductor's identification fields. In these cases, Oxford Semiconductor will assign a subsystem ID to the board vendor, which is the only value that needs to be changed using the EEPROM. To request a subsystem ID value, please contact Oxford Semiconductor with a full functional description of the add-in card being designed.

The first solution is the best because Windows 95 and Windows NT do not recognise subsystem ID values. Therefore they will not be able to distinguish between different add-in cards. Windows 98 and Windows 2000 will be able to identify different cards, provided that the driver .inf file specifies the subsystem ID values, in the form of:

```
PCI\VEN_1415&DEV_9501&SUBSYS_00011415
```

Board vendors should NEVER assign a new DEVICE ID or SUBSYSTEM ID unless they have obtained their own VENDOR ID from the PCISIG. Oxford Semiconductor reserves the right to refuse to allocate a subsystem ID to a board vendor.

Examples

4-serial, 2 parallel port card

To do this it is necessary to use the internal UARTs, and have two external parallel port chips on the local bus. In this case the board vendor can use the default ID for the serial ports, but must have a different ID for the local bus. They can contact Oxford Semiconductor for a subsystem ID for the local bus, or obtain their own VENDOR ID and define their own drivers for the entire card. The three possible approaches are therefore:-

Use the board vendor's own PCI VENDOR ID (in this example case, 0x1234)

	4 serial ports (internal UARTs) FUNCTION 0	**2 parallel ports (Local bus) FUNCTION 1**
VENDOR ID	0x1234	0x1234
DEVICE ID	Chosen by vendor	Chosen by vendor
SUBSYSTEM VENDOR ID	0x1234	0x1234
SUBSYSTEM ID	Chosen by vendor	Chosen by vendor

In this case, the .inf file needs to identify the devices as:-

```
PCI\VEN_1234&DEV_XXXX
PCI\VEN_1234&DEV_YYYY
```

Use the board vendor's own PCI VENDOR ID as the SUBSYSTEM VENDOR ID (in this case, 0x1234)

	4 serial ports (internal UARTs) FUNCTION 0	**2 parallel ports (Local bus) FUNCTION 1**
VENDOR ID	0x1415 (default)	0x1415 (default)
DEVICE ID	0x9501 (default)	0x9511 (default)
SUBSYSTEM VENDOR ID	0x1234	0x1234
SUBSYSTEM ID	Chosen by vendor	Chosen by vendor

In this case, the .inf file needs to identify the devices as:-

```
PCI\VEN_1415&DEV_9501&SUBSYS_XXXX1234
PCI\VEN_1415&DEV_9511&SUBSYS_YYYY1234
```

Obtain subsystem ID from Oxford Semiconductor (example 0x00FF), and use other values as default

	4 serial ports (internal UARTs) FUNCTION 0	**2 parallel ports (Local bus) FUNCTION 1**
VENDOR ID	0x1415 (default)	0x1415 (default)
DEVICE ID	0x9501 (default)	0x9511 (default)
SUBSYSTEM VENDOR ID	0x1415 (default)	0x1415 (default)
SUBSYSTEM ID	0x0001 (default)	0x00FF

In this case, the .inf file needs to identify the devices as:-

```
PCI\VEN_1415&DEV_9501&SUBSYS_00011415
PCI\VEN_1415&DEV_9511&SUBSYS_00FF1415
```

2-serial, 1 parallel port card

It is not possible to limit the hardware to only enable two serial ports, so this solution merely requires a driver that enumerates two ports. Oxford Semiconductor has assigned a DEVICE ID of 0x950A for this, so the recommended solution is:-

	2 serial ports (internal UARTs) FUNCTION 0	**1 parallel ports (Local bus) FUNCTION 1**
VENDOR ID	0x1415	0x1415
DEVICE ID	0x950A	0x9513
SUBSYSTEM VENDOR ID	0x1415	0x1415
SUBSYSTEM ID	Default	Default

Driver versions 2.51 and greater include identification of the 0x950A device ID as 2-serial ports. The card designer should use the first two ports as these as the ones which will be used by the driver.

For Windows 9x, setting the device ID using the EEPROM will be the only modification required. For Windows NT4, the oxser.inf file needs to be modified (all occurrences of 9501 need to be changed to 950A, BEFORE INSTALLATION OF THE DRIVERS.

Please contact

support@oxsemi.com for further details.

1 parallel port only

It is not possible to disable the UARTs on the device, so this solution will require the user to install a null driver for the UARTs. Again Oxford Semiconductor already has assigned a DEVICE ID for this (value 0x9500), so the recommended solution is:-

	NULL function FUNCTION 0	**1 parallel ports (Local bus) FUNCTION 1**
VENDOR ID	0x1415	0x1415
DEVICE ID	0x9500	0x9513
SUBSYSTEM VENDOR ID	0x1415	0x1415
SUBSYSTEM ID	Default	Default

The driver will recognise the device ID 0x9500 as a null function, and load a null driver for this device.

APPENDIX C - TO USE THE WINDOWS NT4 DRIVER FOR A 2-SERIAL PORT SOLUTION

In order to do this, perform the following steps:-

1) Use the EEPROM to program the deviceID for function 0 to 0x950A
2) Replace the oxser.inf file in the WinNT4 directory with the version located in shown below. (This can also be found in doc\misc)
3) Start up the system
4) Run the Install_Serial program from the WinNT4 directory

```
; OXSER.INF
; Used to install TWO OXmPCI954 Serial ports
; Device ID = 0x9501, 9511, 0x8401 assumed for this reference.
;
; If using custom Vendor ID / Device ID, this file must be changed to
; reflect the new settings


[Version]
Signature="$CHICAGO$"
Provider=%ProviderName%
ClassGUID={4d36e978L-e325-11ce-bfc1-08002be10318}
Class=Ports

[DestinationDirs]
DriverCopy        = 12
InfCopy           = 17

[Manufacturer]
%ProviderName% = oxsemi

[oxsemi]
%PCISerial.DeviceDesc%        = Install, PCI\VEN_1415&DEV_950A
%PCISerial.DeviceDesc%        = Install, PCI\VEN_1415&DEV_9511
%PCISerial.DeviceDesc%        = Install, PCI\VEN_1415&DEV_8401

[Install]
AddService=OxSer,, OxSer_Service_Install, OxSer_EventLog_Install
CopyFiles=InfCopy, DriverCopy
AddReg=OxSer_AddReg

[Uninstall]
DelService=OxSer
DelFiles=InfCopy,DriverCopy

[OxSer_Service_Install]
DisplayName  = %ServiceName%
ServiceType  = 1                  ; SERVICE_KERNEL_DRIVER
StartType    = 2                  ; SERVICE_AUTO_START
ErrorControl = 0                  ; SERVICE_ERROR_NONE
ServiceBinary= %12%\OxSer.sys
```

```
[OxSer_AddReg]
HKR,%ParametersSubKey%, EventLogFlags,    0x00010001, 0x00
HKR,%ParametersSubKey%, PCIVendorID,      0x00010001, 0x1415
HKR,%ParametersSubKey%, PCIDeviceID0,     0x00010001, 0x950A
HKR,%ParametersSubKey%, PCIDeviceID1,     0x00010001, 0x9511
HKR,%ParametersSubKey%, PCIDeviceID2,     0x00010001, 0x8401
HKR,%ParametersSubKey%, TxFIFO,           0x00010001, 0x80
HKR,%ParametersSubKey%, RxFIFO,           0x00010001, 0x64
HKR,%ParametersSubKey%, OxsemiPortCount,0x00010001,   2

[OxSer_EventLog_Install]
AddReg=OxSer_EventLog_AddReg

[OxSer_EventLog_AddReg]
HKR,,EventMessageFile,0x00020000,"%%SystemRoot%%\System32\IoLogMsg.dll;
%%SystemRoot%%\System32\drivers\OxSer.sys"
HKR,,TypesSupported,0x00010001,7

[DriverCopy]
OxSer.sys

[InfCopy]
OxSer.inf

[Strings]
PCISerial.DeviceDesc    = "PCI Serial Controller"
ProviderName            = "OEM"
ParametersSubKey        = "Parameters"
ServiceName       = "PCI Serial Driver"
```

APPENDIX D - APPLICATION NOTES AVAILABLE

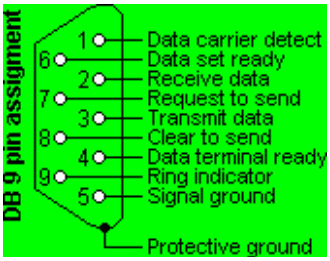
App note number	Title	Product
AN2	Using the OX16C950B in an ISA environment	OX16C950B
AN3	Using the OX16C95x in isochronous clock mode	OX16C950B
AN3	Using the OX16C95x in isochronous clock mode	OX16C954
AN4	Interfaces and line drivers for standard serial and parallel applications	OX16C950B
AN4	Interfaces and line drivers for standard serial and parallel applications	OX16C954
AN4	Interfaces and line drivers for standard serial and parallel applications	OX16PCI954
AN4	Interfaces and line drivers for standard serial and parallel applications	OX16PCI952
AN4	Interfaces and line drivers for standard serial and parallel applications	OX16CB950
AN4	Interfaces and line drivers for standard serial and parallel applications	OX12PCI840
AN4	Interfaces and line drivers for standard serial and parallel applications	OX16CF950
AN5	Software examples for the OX16C95x family of products	OX16C950B
AN5	Software examples for the OX16C95x family of products	OX16C954
AN5	Software examples for the OX16C95x family of products	OX16PCI954
AN7	Common problems with the OX16PCI954 and NT4	OX16PCI954
AN8	Using the OX16PCI954 in a low-cost four-port serial card	OX16PCI954
AN9	Using the OX16PCI954 in a combo four-port serial, one-port parallel card	OX16PCI954
AN 10	8-port serial card design	OX16PCI954
AN 10	8-port serial card design	OX16C954
AN 11	12,16,20-port card design	OX16PCI954
AN 11	12,16,20-port card design	OX16C954
AN12	Common Connectivity Examples for the OX16PCI954	OX16PCI954
AN13	Using the OXCF950 with legacy drivers	OXCF950
AN14	Using the OXCF950 under Windows	OXCF950
AN15	Setting UART baud rates under Windows	OX16PCI954
AN15	Setting UART baud rates under Windows	OX16PCI952
AN15	Setting UART baud rates under Windows	OX16CB950
AN15	Setting UART baud rates under Windows	OXCF950
AN16	Using OXPROM with the OXCF950	OXCF950
AN17	How to write a CIS for the OXCF950	OXCF950
AN18	Physical details of PC and CF cards	OXCF950

APPENDIX E - RS232 CONNECTOR PIN ASSIGNMENT

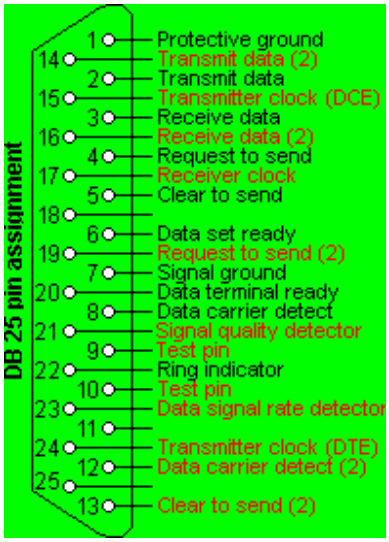
The diagrams below show the signals common to both connector types in black. The signals only present on the larger connector are shown in red. Note, that the protective ground is assigned to a pin at the large connector where the connector outside is used for that purpose with the DB9 version.

Although this interface is differential (the receive and transmit have their own floating ground level) it is possible to connect RS232 compatible devices with this interface.

RS232 DB 9 pin assignment



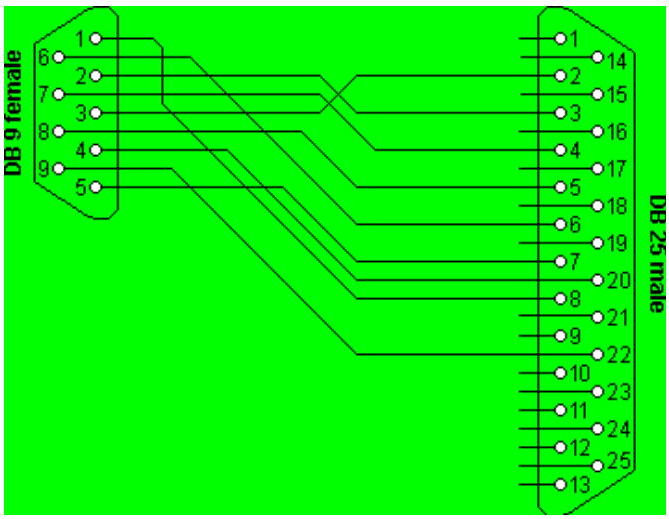
RS232 DB 25 pin assignment



DB9 to DB25 converter

The original pin layout for RS232 was developed for a 25 pins sub D connector. Since the IBM-AT, 9 pins connectors are commonly used. In mixed applications, a 9 to 25 pins converter can be used to connect connectors of different sizes.

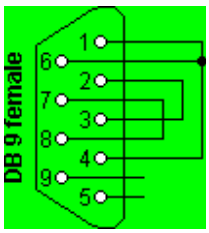
RS232 DB9 to DB25 converter

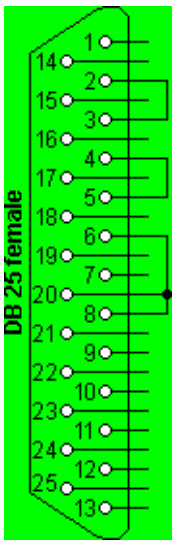


RS232 loop back test plug

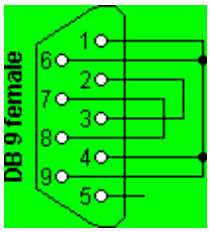
The following connectors can be used to test a serial port on your computer. The data and handshake lines have been linked. In this way all data will be sent back immediately. The PC controls its own handshaking. The first one can be used to check the function of the serial port with standard terminal software. The second version can be used to test the full functionality of the serial port with Norton Diagnostics or CheckIt.

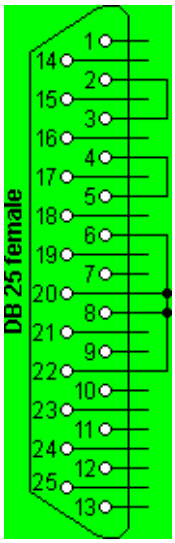
RS232 loop back test plug for terminal emulation software





RS232 loopback test plug for Norton Diagnostics and CheckIt



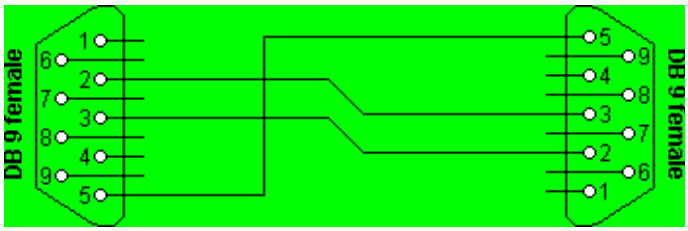


RS232 null modem cables

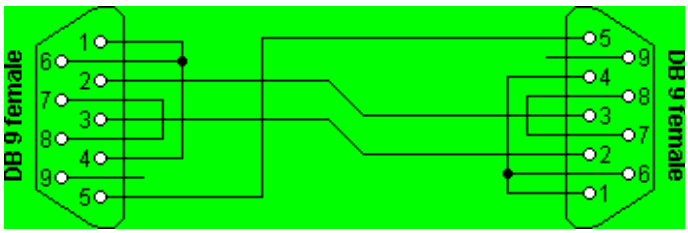
The easiest way to connect two PC's is using a null modem cable. The only problem is the large variety of null-modem cables available. For simple connections, a three line cable connecting the signal ground and receive and transmit lines is sufficient. Depending of the software used, some sort of handshaking may however be necessary. For a Windows 95/98 Direct Cable Connection, the null modem cable with loop back handshaking is a good choice.

Null modem cables with handshaking can be defined in numerous ways, with loop back handshaking to each PC, or complete handshaking between the two systems. The most common cable types are shown here.

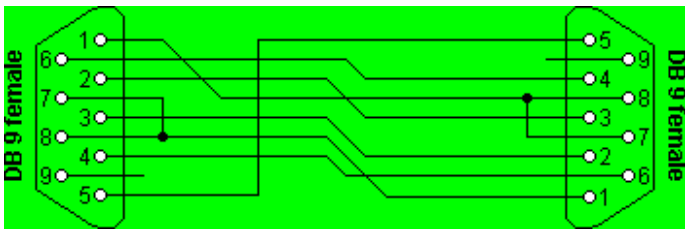
Simple null modem without handshaking



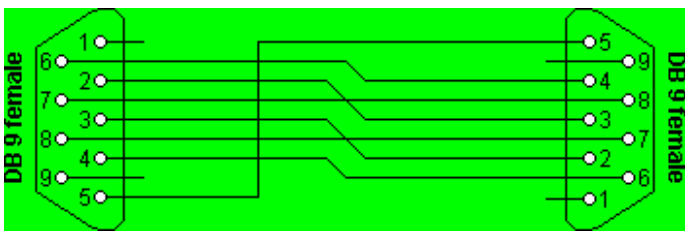
Null modem with loop back handshaking



Null modem with partial handshaking



Null modem with full handshaking

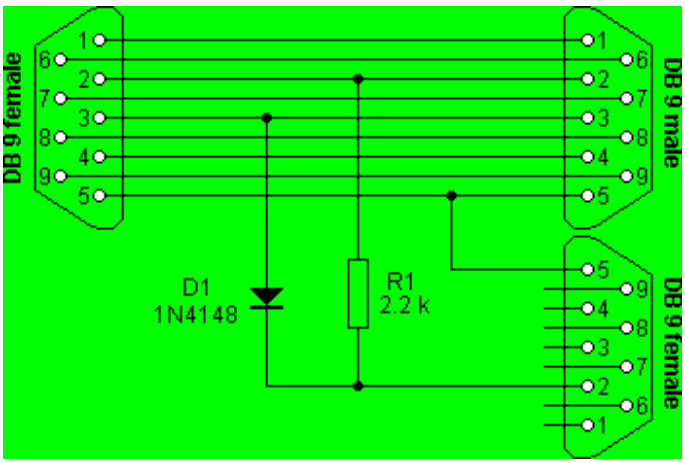


RS232 monitor cable

It is not difficult to monitor the serial communication between two devices with a PC. To do this you need the monitor cable which is displayed in the next picture. Two sockets are connected straight through. The PC is connected to the third one.

This monitor cable taps communication from both sides. This means that if the two devices happen to talk simultaneously, the monitored information will be garbage. In most circumstances communication software works half duplex, in which case this problem does not exist.

RS232 monitor cable



CONTACT INFORMATION

For further information please contact:

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
Technical support: Support@oxsemi.com
Web site: http://www.oxsemi.com

DISCLAIMER