Dual High Speed Serial Interface Expansion Card

for Acorn RISC OS-Based Computer Systems

User Guide and

Programmer's Reference Manual

Intelligent Interfaces Ltd

May 1996

All information is given by Intelligent Interfaces in good faith. However, it is acknowledged that there may be errors or omissions in this User Guide. Intelligent Interfaces welcomes comments and suggestions relating to the Dual High Speed Serial Interface Expansion Card and this User Guide.

All correspondence should be addressed to:-

Technical Enquiries
Intelligent Interfaces Ltd
P O Box 80
Eastleigh
Hampshire
SO53 2YX

Tel:	023 8026 1514
Fax:	087 0052 1281
Email:	support@intint.demon.co.uk

Contents

Introduction

Each Dual High Speed Serial Interface Expansion Card provides any Acorn Risc OS-based computer, with an expansion backplane and RISC OS 3.1 or later, with two additional high speed RS232 compatible serial ports for communicating with serial devices eg modems, printers, plotters, instruments, etc. The ports can input and output data at up to 230400 baud enabling the use of the latest modems for high speed Internet access. They have robust socketed line drivers and receivers. The connectors have the same pin assignments as the internal port. Modem cables with either Acorn or PC wiring can be used.

Full software support is provided by a DeviceFS device driver module and block drivers to enable the use of popular applications such as the ANT Internet Suite, Termite, Voyager, Internet Starter Pak, ARCfax, Hearsay II, ArcTerm 7, etc.

Other areas of application include the connection of engineering, scientific and medical instruments. The card enables the computer to be used in laboratory and production testing and process monitoring and control.

This document is divided into two parts:-

i) a User Guide which describes how to fit the expansion card, install the software and configure popular applications. This part should be read by all users.

ii) a Programmer's Reference Manual. This part should be read by those users writing applications which use serial communications.

Before fitting the expansion card check that the following items in addition to this document have been received. If any item is missing contact the supplier.

Dual High Speed Serial Interface Expansion Card
Software Distribution Disc

User Guide

1 Fitting the Expansion Card

The card can be fitted in any Acorn Computer with an expansion backplane and RISC OS 3.1 or later. The standard card cannot be fitted in the external expansion card socket of an A3000 computer unless an independent -5V supply is available for the card. However, a modified card is available from Intelligent Interfaces for fitting in the external expansion card socket of an A3000 computer.

To fit the card in an A300 series, A400 series, A540, A5000, Risc PC or A7000 computer:-

1 Switch off the power to the Computer.
2 Disconnect the Computer from the mains supply.
3 The card can be fitted in any unused expansion card slot.
4 Remove the blanking plate from the rear of the Computer and retain the two screws.
5 Fit the card and secure it in position using the two screws retained at stage 4 (if required, fit a joiner and blanking plate).
6 Reconnect the Computer to the mains supply.
7 Switch on the power to the Computer.
8 Confirm that the card has been fitted correctly by pressing F12 and typing

```
*Podules
```

This should list the Dual High Speed Serial Interface as

```
Intelligent Interfaces Dual High Speed Serial Interface
```

together with any other cards fitted. Press <Return> to return to the Desktop.

The maximum number of Dual High Speed Serial Interface Expansion Cards that can be fitted is limited only by the number of unused expansion card slots.

2 Installing the Software

It is recommended that a Comms directory is created with sub-directories for each communications application eg Internet, ArcFax, Hearsay, etc. If a !SerialDev application (which contains block drivers) is already in use it is recommended that this is moved into the Comms directory.

2.1 Installing the IIDual Module

It is essential that the IIDual module is loaded before any serial communications application.

1. Open the DevDriver directory of the distribution disc and drag the !IIDual application from the distribution disc to the Comms directory of the destination filing system. The !IIDual application automatically loads the IIDual module from the Modules directory within the application when a window is opened on the directory containing it.

2. Immediately after installation, as the window is already open, it is necessary to double click on the !IIDual application to load the IIDual module.

2.2 Installing the IIDual and IIDualPC Block Drivers

1. Open the BlkDriver directory of the distribution disc and drag the !SerialDev application from the distribution disc to the Comms directory of the destination filing system. If a !SerialDev application is already in use then it will contain the original II_Dual driver for use with the original IIRS423 Module. It is recommended that this is deleted in order to avoid confusion.

2. Immediately after installation, as the window is already open, it is necessary to double click on !SerialDev in order to initialise the application.

2.3 Installing the Diagnostic Applications

1. Drag the Diagnostic directory from the distribution disc to the Comms directory of the destination filing system. Open the Diagnostics directory and double click on the !Version application. This lists the version number of the IIDual module and the version numbers of the block drivers in the text file Version. Check that the IIDual module version and IIDual and IIDualPC versions are correct.

3 Dual High Speed Serial Interface Port Designations

The IIDual module designates the inbuilt serial interface as Port 0.

If only one Intelligent Interfaces Dual High Speed Serial Interface is fitted, the ports are designated 1 and 2 from left to right (viewed from the rear of the computer), irrespective of which expansion slot the card is fitted. If more than one card is fitted, the card in the lowest numbered slot has its ports designated as 1 and 2, the card in the next highest slot has its ports designated as 3 and 4, etc. For example, a computer fitted with cards in slots 1 and 3 will have ports 1 and 2 on the card in slot 1 and 3 and 4 on the card in slot 3.

Important note - when configuring any application that uses Block Drivers eg Internet Starter Pak, ArcFax, Hearsay II, etc., one must be subtracted from the port numbers as designated above ie they start at 0 rather than 1. For example, if only one card is fitted the ports should be specified as port 0 and port 1.

(Note - the original IIRS423 module designated the port numbers in a different way)

4 Configuring Applications

Notes on configuring popular applications, including the ANT Internet Suite, Voyager, Internet Starter Pak, ArcFax, Hearsay II, etc, are contained in the directory $.Docs.InstallApp on the distribution disc.

5 Optimising the Performance of Applications

The following recommendations apply to the older A300 series, A400 series, A540 and A5000 computers.

i) for A300 Series and A400 Series computers still fitted with the original ARM 2 processor, upgrade to an ARM 3 processor.

ii) choose the lowest resolution screen mode with the lowest number of colours during periods of Internet access. A significant increase in performance can be achieved by changing from mode 28 (640 x 480 x 256 colours) to mode 25 (640 x 480 x 2 colours) if a VGA monitor is in use or from mode 15 (640 x 256 x 256 colours) to mode 0 (640 x 256 x 2 colours) if a TV standard monitor is in use.

iii) normally a baud rate of 57600 is recommended for a 14k4 baud modem and a baud rate of 115200 for a 28k8 baud modem. In some circumstances reducing the baud rate to 38400 for a 14k4 baud modem and to 57600 for a 28k8 baud modem can result in a slightly higher overall data transfer rate. This is due to the smoother data transfer which results from the reduction in the amount of handshaking required (fewer stops and starts).

iv) have as few applications installed on the icon bar as possible.

Programmer's Reference Manual

1 Software

1.1 The IIDual Module

The IIDual module provides a RISC OS DeviceFS device driver for each serial port which appear as separate objects IIDual1, IIDual2, etc, within the 'devices:' filing system. Most filing system interface SWI's, OS_Find, OS_BGet, OS_BPut, etc, can be used, ie files can be opened for input or output to each port and data transferred.

The IIDual module also provides a SWI interface. The IIDual SWI's are listed in Appendix IV and are described in the text file $.Docs.IIDualSWIs on the distribution disc. The IIDual commands are listed in Appendix VI. The IIDual SWI's offer a direct interface to the ports and should be used in preference to the filing system interface. Additionally, they provide a means of setting the baud rate, data format, etc, for each port.

The first 10 IIDual SWI's from IIDual_ReadWriteSerialStatus to IIDual_EnumerateBaudRates are equivalent to the 10 OS_SerialOp's provided by the Risc OS SerialDeviceSupport module.

The IIDual module options listed in Appendix V enable

(i) diagnostic information to be read from the module.

(ii) the values of parameters used to optimise the input and output of data to be written to the module. It should not be necessary to change the default values of these parameters.

An IIDual module option enables the buffer insert/remove mode, for both block and single byte buffer operations, to be set.

Mode	Description
0	RISC OS indirect routines ie insV, remV and cnpV used (the same way as the device driver for the internal port)
1	RISC OS direct buffer manager service routines used (for faster operation on ARM6 and ARM7 computers - only possible with RISC OS 3.5 or later)
2	IIDual module direct routines used (not RISC OS DeviceFS device driver compatible - for faster operation on ARM2 and ARM3 computers)

On module initialisation the following defaults are set:-

baud rate set	set 4
input buffer size	1 Kbyte
output buffer size	1 Kbyte
input buffer minimum space	32 bytes
input buffer hysteresis	32 bytes
Rx baud rate	read from CMOS memory as one of 9600 or 19200 or 57600 or 115200
Tx baud rate	read from CMOS memory as one of 9600 or 19200 or 57600 or 115200

data format	read from CMOS memory as one of 8 data, no parity, 2 stop bits or 8 data, no parity, 1stop bit or 8 data, even parity, 1 stop bit or 8 data, odd parity, 1 stop bit
insert/remove mode	mode 0

In its default state the IIDual module sets the DTR output true, requires that both the CTS and DSR inputs must be true before a byte can be transmitted and that the DCD input must be true before a byte can be received. These conditions can be changed using the IIDual_ReadWriteStatus SWI or the IIDual_Handshake command.

1.2 The !IIDual Application

The !IIDual application loads the !IIDual module. The !IIDual.!Boot obey file sets

the baud rate set
the Rx and Tx baud rate
the data format
the input buffer size
the output buffer size
the input buffer minimum space
the input buffer hysteresis
the handshake method
the private input buffer entries at which input is halted
the maximum number of bytes to be output per transmit interrupt
the buffer insert/remove mode

Normally, it should not be necessary to change the values of these parameters set by the !IIDual.!Boot obey file.

Most popular commercial applications subsequently set the values of the following parameters

the Rx and Tx baud rate
the data format
the input buffer minimum space
the handshake method

1.3 The !SerialDev Application - Block Drivers

The IIDual (for modem cable with Acorn wiring) and IIDualPC (for modem cable with PC wiring) block drivers comply with the Block Driver Specification defined by Hugo Fiennes. They are provided to enable the card to be used with existing popular commercial applications.

1.4 The Serial BASIC Library

The Serial BASIC Library is provided for the convenience of those writing applications in BASIC which use serial communications. The library is located in the !IIDual.BASICLib directory. The !IIDual.!Boot obey file sets the OS variable IIDual$Dir to the directory containing BASICLib. To use the serial functions and procedures provided by the Serial BASIC Library, which are listed in Appendix VII, include the following line at the beginning of the program.

```
LIBRARY "<IIDual$Dir>.BASICLib.Serial"
```

Examples are included on the distribution disc. The !ModemInit and !ModemDiag applications can be used with a US Robotics Sportster 28k8 modem connected to port 1 (left hand port viewed from the rear of the computer) and the !LoopBack application can be used if a loop back connector which connects pin2 to pin 3 and pin 6 to pin 7 is plugged into port 2 (right hand port viewed from the rear of the computer).

There are further examples in the Examples.BASIC directory.

1.5 Writing and Optimising the Performance of Applications that Use Serial Communications

It is recommended that an application sets the baud rate, data format, etc, for each port by calling IIDual SWI' s. The procedure Serial_Init in the Serial BASIC Library is an
example of this.

Every SWI call takes a certain amount of time to enter and exit and therefore reducing the number of SWI calls reduces the total time taken for the same number of bytes transferred. It is important to avoid any unnecessary SWI calls. Using block input and output rather than single byte input and output does this.

The SWI IIDual_GetByte can be used to input data. For example to input 8 bytes

```
FOR i% = 1 TO 8
 REPEAT
  SYS"IIDual_GetByte",port% TO ,byte% ;flags%
 UNTIL (flags% AND %0010) = %0000
 data$ = data$ + CHR$(byte%)
NEXT i%
```

In the above example the C flag is set if there are no bytes in the input buffer. Therefore, there is never a requirement to use the SWI IIDual_ReadEntriesInBuf before calling the SWI IIDual_GetByte.

The SWI IIDual_SendByte can be used to output data. For example to output 8 bytes

```
FOR i% = 1 TO 8
 REPEAT
  SYS"IIDual_SendByte",port%,ASC(MID$(data$, i%, 1)) TO ;flags%
 UNTIL (flags% AND %0010) = %0000
NEXT i%
```

In the above example the C flag is set if there is no space in the output buffer. Therefore, there is never a requirement to use the SWI IIDual_ReadSpaceOutBuf before calling the SWI IIDual_SendByte.

The SWI IIDual_GetBlock can be used to input data. For example to input 8 bytes

```
DIM buffer% 8

adr_ptr% = buffer%
cnt% = 8

REPEAT
 SYS"IIDual_GetBlk", port%, adr_ptr%, cnt% TO , adr_ptr%, cnt%
UNTIL cnt% <= 0

FOR i% = 1 TO 8
 data$ = data$ + CHR$(buffer%(i% - 1))
NEXT i%
```

Note - all the bytes may be removed from the input buffer on the first call. For example

```
 DIM buffer% 1024

 adr_ptr% = buffer%
 max_cnt% = 1024

SYS"IIDual_GetBlk", port%, adr_ptr%, max_cnt% TO , adr_ptr%,not_removed%

 removed% = max_cnt% - not_removed%
```

In the above example removed% can range from 0 (if there were no entries in the input buffer) to 1024 (if there were greater or equal to max_cnt% entries). There is never any need to call the SWI IIDual_ReadEntriesInBuf before calling the SWI IIDual_GetBlock. adr_ptr% is updated to point to the location in buffer% following the last byte inserted.

The SWI IIDual_SendBlock can be used to output data. For example to output 8 bytes

```
 DIM buffer% 8

 FOR i% = 1 TO 8
  buffer%(i% - 1) = ASC(MID$(data$, i%, 1))
 NEXT i%

 adr_ptr% = buffer%
 cnt% = 8

 REPEAT
  SYS"IIDual_SendBlk", port%, adr_ptr%, cnt% TO , adr_ptr%, cnt%
 UNTIL cnt% <= 0
```

Note - all the bytes may be inserted in the output buffer on the first call. For example

```
 DIM buffer% 1024

 adr_ptr% = buffer%
 max_cnt% = 1024

 SYS"IIDual_SendBlk", port%, adr_ptr%, max_cnt% TO , adr_ptr%,
not_inserted%

 inserted% = max_cnt% - not_inserted%
```

In the above example inserted% can range from 0 (if there was no space in the output buffer) to 1024 (if there was greater or equal to max_cnt% space). There is never a requirement to call the SWI IIDual_ReadSpaceOutBuf before calling the SWI IIDual_SendBlock. adr_ptr% is updated to point to the location in buffer% following the last byte removed.

To summarise, it should be possible to input and output data using only the SWI' s IIDual_GetBlock and IIDual_SendBlock. This significantly reduces the number of SWI calls and the resulting overhead, thereby increasing performance.

Appendix I Connector Pin Designations

Pin			
1	DCD	Data Carrier Detect	Input
2	RxD	Receive Data	Input
3	TxD	Transmit Data	Output
4	DTR	Data Terminal Ready	Output
5	0V	Signal Ground	
6	DSR	Data Set Ready	Input
7	RTS	Request to Send	Output
8	CTS	Clear to Send	Input
9	RI	Ring Indicator	Input

The above pin designations are identical to those of the inbuilt serial port. The Dual High Speed Serial Interface uses RS423 line drivers and receivers which are RS232 compatible.

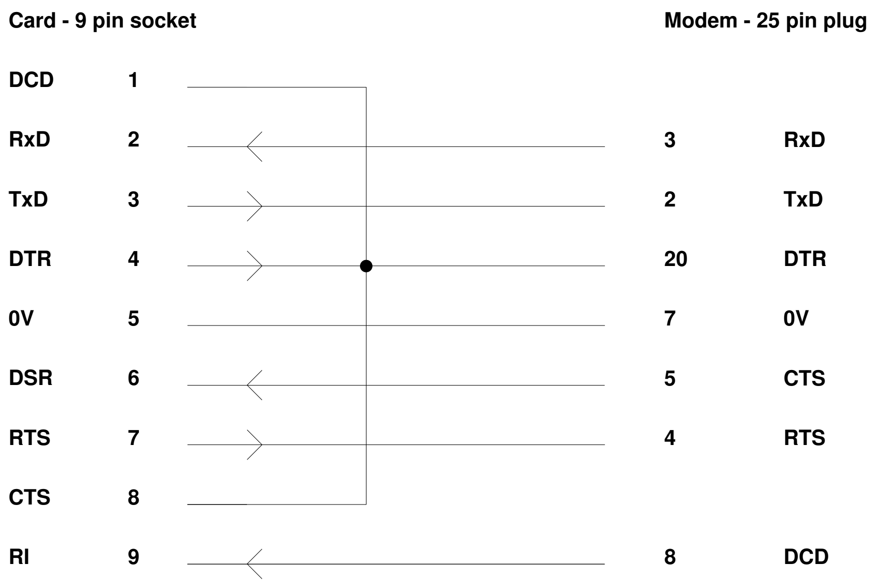
Appendix II Serial Cables

Cable Wiring for Modems

Intelligent Interfaces recommend the use of a modem cable with PC wiring, as these are more readily available, unless the cable is also to be used with the inbuilt port of early Acorn RISC OS-based computers, ie A300 series, A400 series, A3000 and A540, in which case a modem cable with Acorn wiring must be used.

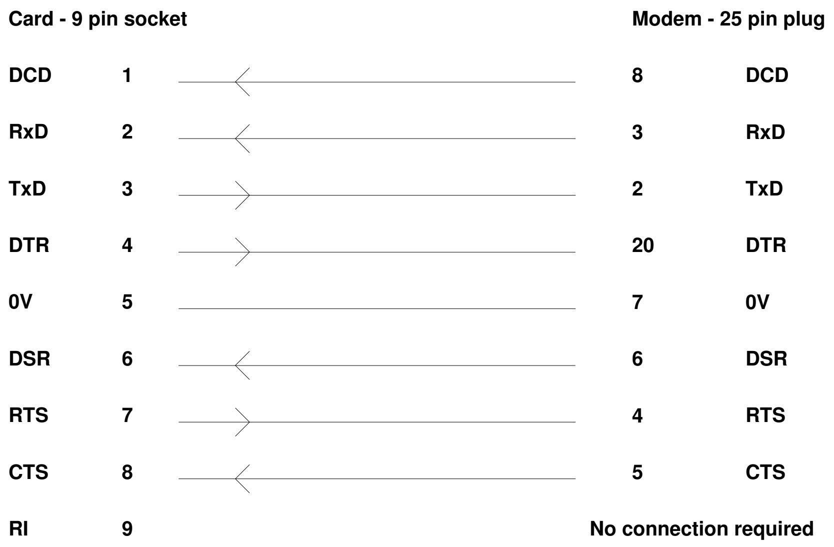
Modem Cable with Acorn Wiring

This cable wiring can also be used with the inbuilt port of any Acorn RISC OS-based computer.



Modem Cable with PC Wiring

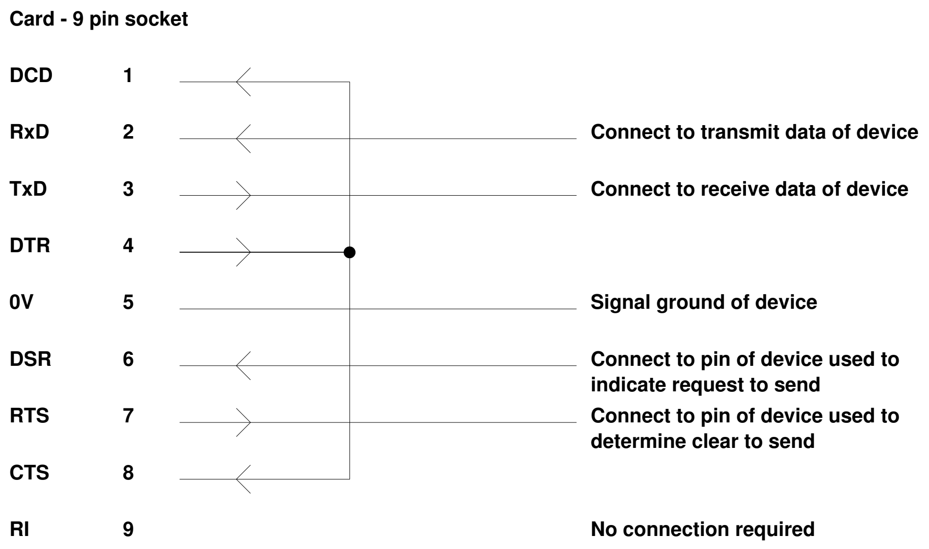
This cable wiring should not be used with the inbuilt port of early Acorn RISC OS-based computers, ie A300 series, A400 series, A3000 and A540.



Cable Wiring for Other Devices

Intelligent Interfaces recommend the use of a cable with hardware handshaking (RTS/CTS) wiring unless the cable is also to be used with the inbuilt port of early Acorn RISC OS-based computers, ie A300 series, A400 series, A3000 and A540, in which case a cable with Hardware Handshaking (RTS/DSR) wiring must be used.

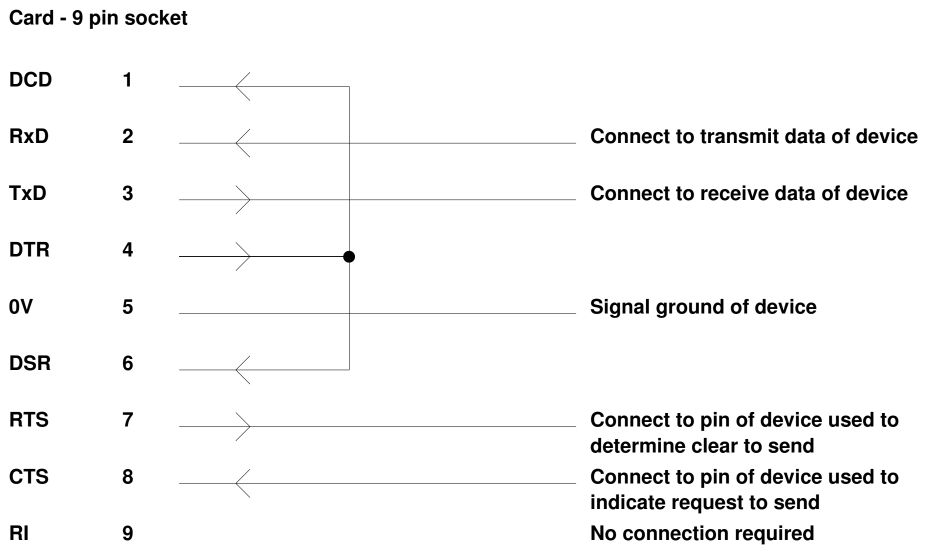
Hardware Handshaking (RTS/DSR)


Card - 9 pin socket
DCD 1
RxD 2
TxD 3
DTR 4
0V 5
DSR 6
RTS 7
CTS 8
RI 9
Connect to transmit data of device
Connect to receive data of device
Signal ground of device
Connect to pin of device used to indicate request to send
Connect to pin of device used to determine clear to send
No connection required

This cable wiring can also be used with the inbuilt port of any Acorn RISC OS-based computer.

Hardware Handshaking (RTS/CTS)

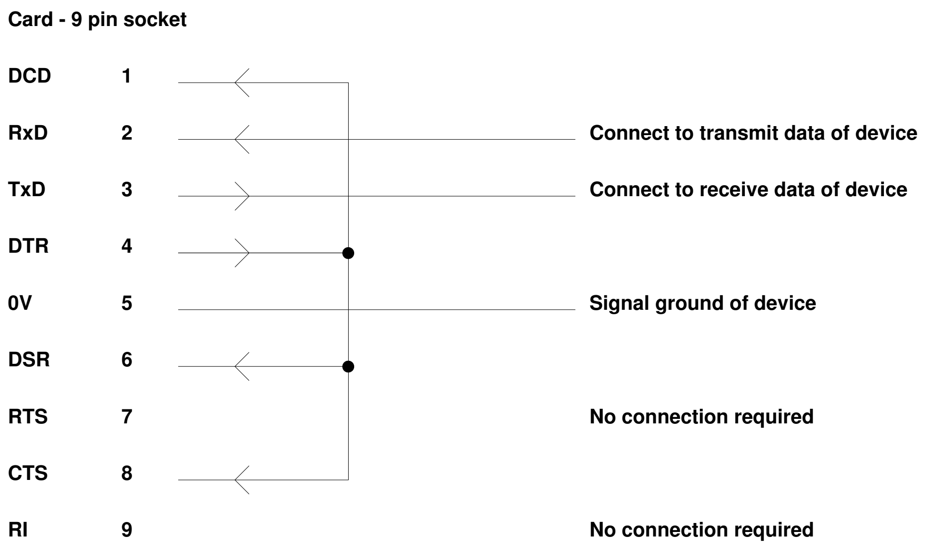
This cable wiring should not be used with the inbuilt port of early Acorn RISC OS-based


Card - 9 pin socket
DCD 1
RxD 2
TxD 3
DTR 4
0V 5
DSR 6
RTS 7
CTS 8
RI 9
Connect to transmit data of device
Connect to receive data of device
Signal ground of device
Connect to pin of device used to determine clear to send
Connect to pin of device used to indicate request to send
No connection required

computers, ie A300 series, A400 series, A3000 and A540.

Cable Wiring for Software Handshaking (Xon/Xoff)

Software handshaking (Xon/Xoff) should only be used when hardware handshaking is not available.



This cable wiring can also be used with the inbuilt port of any Acorn RISC OS-based computer.

Appendix III Baud Rates and Data Formats

Baud Rates

The default baud rate set is Set 4. This set provides the same range of baud rates (from 50 to 115200) as the Risc PC and A7000 computers. The only reason to change from Set 4 (default) to Set 2 is if a baud rate of 230400 is required.

The baud rate can be set using the SWIs IIDual_ReadWriteRxBaudRate and IIDual_ReadWriteTxBaudRate. These SWIs are documented in the text file $.Docs.IIDualSWIs on the distribution disc.

The baud rates available on odd numbered ports are slightly more accurate than those on even numbered ports. The table below shows the baud rates available. Those baud rates denoted by an asterisk are selected from the table of standard baud rates whilst the others are generated for odd numbered ports by programming the 26C92 timer and for even numbered ports by programming the 65C22 timer.

Index	Set 0	Set 1	Set 2	Set 3	Set 4 (default)	Set 5
0	9600*	9600*	7200*	7200*	9600*	9600*
1	75	75*	75	75	75	75
2	150	150*	150	150	150	150
3	300*	300*	300	300	300	300
4	1200*	1200*	1200	1200	1200	1200
5	2400*	2400*	2400	2400	2400	2400
6	4800*	4800*	4800	4800	4800*	4800*
7	9600*	9600*			9600*	9600*
8		19200*			19200*	19200*
9	50*	50	50*	50	50	50
10	110*	110*	110*	110*	110	110
11	134.5*	134.5*	134.5*	134.5	134.5	134.5
12	600*	600*	600	600	600	600
13	1800	1800*	1800*	1800*	1800	1800
14	3600	3600	3600*	3600*	3600	3600
15	7200	7200	7200*	7200*	7200	7200*
16	38400*				38400*	
17				57600*	57600*	57600*
18				115200*	115200*	115200*
19			230400*			
20			14400*	14400*		14400*
21			28800*	28800*	28800*	28800*
22	200	200	200*	200	200	200
23	450	450	450	450*	450	450
24	880	880	880	880	880*	880*
25	900	900	900	900*	900	900
26	1050	1050	1050*	1050	1050*	1050
27	1076	1076	1076	1076	1076*	1076
28	2000	2000*	2000	2000*	2000	2000

An alternative way of selecting the baud rate is to use the IIDual_Baud command which sets the RX and TX baud rates.

Syntax IIDual_Baud <port> <index (0 - 28) or baud rate (50 - 230400)>

The following command selects for port 1 the receive and transmit baud rate as 19200.

```
IIDual_Baud 1 19200
```

The following command selects for port 2 the receive and transmit baud rate as 19200.

```
IIDual_Baud 2 19200
```

Data Formats

A wide range of data formats can be set using the SWI IIDual_ReadWriteDataFormat. This SWI is documented in the text file $.Docs.IIDualSWIs on the distribution disc.

Bit Name	Read/Write or Read Only	Value	Meaning
0,1	R/W	0	8 bit word
		1	7 bit word
		2	6 bit word
		3	5 bit word
2	R/W	0	1 stop bit
		1	2 stop bits in most cases. 1 stop bit if 8 bit word with parity. 1.5 stop bits if 5 bit word without parity.
3	R/W	0	Parity disabled.
		1	Parity enabled.
4,5	R/W	0	Odd parity.
		1	Even parity
		2	Parity always 1 on TX and ignored on RX.
		3	Parity always 0 on TX and ignored on RX.
6-31			Reserved. Must be set to 0.

An alternative way of selecting the data format is to use the IIDual_Data command. The following data formats are available:-

<n>	Word Length bits	Parity	Stop bits
0	7	even	2
1	7	odd	2
2	7	even	1
3	7	odd	1
4	8	none	2
5	8	none	1
6	8	even	1
7	8	odd	1

Syntax IIDual_Data <port> <data format (0 - 7)>

The following commands selects for port 1 a data format of 8 bits, no parity, 1 stop bits

```
IIDual_Data 1 5
```

The following commands selects for port 2 a data format of 8 bits, no parity, 1 stop bits

```
IIDual_Data 2 5
```

Appendix IV IIDual SWI's

The documentation for the SWI' s can be found in the text file $.Docs.IIDualSWIs on the distribution disc.

&048680	IIDual_ReadWriteSerialStatus
&048681	IIDual_ReadWriteDataFormat
&048682	IIDual_SendBreak
&048683	IIDual_SendByte
&048684	IIDual_GetByte
&048685	IIDual_ReadWriteRxBaudRate
&048686	IIDual_ReadWriteTxBaudRate
&048687	IIDual_Reserved0
&048688	IIDual_ReadWriteInBufMinimumSpace
&048689	IIDual_EnumerateBaudRates
&04868A	IIDual_EnableDisableInput
&04868B	IIDual_EnableDisableOutput
&04868C	IIDual_StartStopBreak
&04868D	IIDual_SendBlk
&04868E	IIDual_GetBlk
&04868F	IIDual_GetBlkUntilMatch
&048690	IIDual_SendByteInTime
&048691	IIDual_GetByteInTime
&048692	IIDual_SendBlkInTime
&048693	IIDual_GetBlkInTime
&048694	IIDual_GetBlkUntilMatchInTime
&048695	IIDual_InsertByteInBuf
&048696	IIDual_RemoveByteOutBuf
&048697	IIDual_InsertBlkInBuf
&048698	IIDual_RemoveBlkOutBuf
&048699	IIDual_ReadSpaceInBuf
&04869A	IIDual_ReadSpaceOutBuf
&04869B	IIDual_ReadEntriesInBuf
&04869C	IIDual_ReadEntriesOutBuf
&04869D	IIDual_ExamineNextByteInBuf
&04869E	IIDual_ExamineNextByteOutBuf
&04869F	IIDual_ExamineBlkInBuf
&0486A0	IIDual_ExamineBlkOutBuf
&0486A1	IIDual_ExamineNextByteInBufInTime
&0486A2	IIDual_ExamineBlkInBufInTime
&0486A3	IIDual_ExamineBlkInBufUntilMatch
&0486A4	IIDual_ExamineBlkInBufUntilMatchInTime
&0486A5	IIDual_FlushInBuf
&0486A6	IIDual_FlushOutBuf
&0486A7	IIDual_ReadWriteInBufSize
&0486A8	IIDual_ReadWriteOutBufSize
&0486A9	IIDual_ReadInBufInfo
&0486AA	IIDual_ReadOutBufInfo
&0486AB	IIDual_ReadWriteInBuf
&0486AC	IIDual_ReadWriteOutBuf
&0486AD	IIDual_ReadWriteOptions
&0486AE	IIDual_ReadNumberPorts

Appendix V IIDual Options

The documentation for the IIDual Options can be found in the text file $.Docs.IIDualSWIs on the distribution disc in the section for the SWI IIDual_ReadWriteOptions.

0	to read the version number
1	to read the count of RxRdy interrupts since last read
2	to read the max number of bytes received per RxRdy interrupt since last read
3	to read the distribution of the number of bytes received per RxRdy interrupt since last read
4	to read the number of TxRdy interrupts received since last read
5	to read the max number of bytes transmitted per TxRdy interrupt since last read
6	to read the distribution of the number of bytes transmitted per TxRdy interrupt since last read
7	to read the number of times the private input buffer minimum has been exceeded since last read
8	to read the maximum number of bytes in the private input buffer since last read
9	to read/write the buffer insert/remove mode (default 0)
10	to read/write the number of bytes for which interrupts are disabled during block insert/remove operations (default 64)
11	to read/write the private input buffer entries (default 4) at which RTS is set in the interrupt routine
12	to read/write the maximum number of bytes to be received per RxRdy interrupt (default 16)
13	to read/write the maximum number of bytes to be transmitted per TxRdy interrupt (default 8)
14	to read the receive errors that have occurred since last read.
15	to read/write the baud rate mode (default 0 common)
16	to read/write the baud rate set
17	to read/write the input buffer hysteresis hysteresis = on_space - off_space
18	to read/write the card default 6522 T2 period
19	to read/write the card short 6522 T2 period
20	to read/write the card short 6522 T2 count
21	to enable and disable the port' s local loopback
22	to read and write the card' s UserIO
253	to read the port DeviceDriverExternal_entry array
254	to read the port DeviceDriver_entry array
255	to read the port swi_cnt array

Appendix VI IIDual Commands and Configuration Keywords

IIDual Commands

IIDual_Baud
IIDual_RxBaud
IIDual_TxBaud
IIDual_Data
IIDual_HandShake
IIDual_InBuf
IIDual_OutBuf
IIDual_Options
IIDual_InBufMinSpace
IIDual_InBufHysteresis
IIDual_BaudRateSet

Configuration Keywords

IIDual_CMOSBaud
IIDual_CMOSData

Appendix VII Serial BASIC Library Procedures and Functions

```
PROCSerial_Init(port%, baud%, data_format%, flow_control%)
PROCSerial_Final(port%)
FNSerial_Status(port%)
PROCSerial_EnableDTR(port%)
PROCSerial_DisableDTR(port%)
PROCSerial_SendBreak(port%, time%)
PROCSerial_EnableInput(port%)
PROCSerial_DisableInput(port%)
PROCSerial_FlushInBuf(port%)
PROCSerial_FlushOutBuf(port%)
FNSerial_ReadEntriesInBuf(port%)
FNSerial_ReadSpaceOutBuf(port%)
PROCSerial_SendByte(port%, byte%)
FNSerial_GetByte(port%)
PROCSerial_SendString(port%,data$)
FNSerial_GetString(port%,match$)
PROCSerial_SendBlock(port%, adr%, cnt%)
PROCSerial_GetBlock(port%, adr%, cnt%)
```