diskserial.device

Stefan Sichler

COLLABORATORS			
	TITLE : diskserial.device		
ACTION	*NAME*	*DATE*	*SIGNATURE*
WRITTEN BY	Stefan Sichler	June 24, 2022	

REVISION HISTORY			
NUMBER	DATE	DESCRIPTION	NAME

Contents

1 diskserial.device **1**

1.1 diskserial.device - Guide . . . 1
1.2 General Information . . . 1
1.3 Features . . . 2
1.4 Restrictions to the diskserial implementation . . . 2
1.5 Requirements . . . 2
1.6 Cable Assembly Instructions . . . 3
1.7 Hard- and Software Installation . . . 3
1.8 User's manual . . . 4
1.9 Differences to serial.device . . . 4
1.10 Bugs/Known Problems . . . 5
1.11 diskserial.device's evolution history . . . 6
1.12 About the Author... . . . 6

Chapter 1

diskserial.device

1.1 diskserial.device - Guide

diskserial.device V1.2a

==========

by Stefan Sichler

General Info What's it good for?

Features Why is it better than the serial.device?

Restrictions Why doesn't it replace a RS232 port?

Requirements What do you need to use it?

The Cable How to build that cable?

Installation How to put it on your Amiga?

User's Manual How to use it?

Differences What differs from serial.device?

Bugs Where have I been too lazy to correct the errors?

History What has changed recently?

1.2 General Information

"diskserial.device" is FREEWARE.

{Because I don't have any money, I don't expect you having any!)

It is an (almost) serial.device compatible Exec device that uses the disk port instead of the serial port, therefore a special cable is required.

It should help to easily build a fast network between two or more Amigas.

For example, it can be used together with a terminal program, slip or cslip.

!!!! It does NOT emulate a RS-232 interface !!!! - It can ONLY be used between Amigas.

1.3 Features

- very compatible with serial.device
- on accelerated Amigas up to 31KB/sec. (= ~280 000 Baud)
- on non-accelerated Amigas (old A500) up to 6KB/sec. (= ~55 000 Baud) at little CPU usage
- VERY less hardware required (only a cheap easy-to-do cable)

1.4 Restrictions to the diskserial implementation

I have been asked several times (mails, mails, mails,...) to do a RS-232 <-> diskport converter by now. But in my opinion it's almost impossible.

The only way I see is to use a singe-chip computer for the conversion.

For those who don't understand the reasons, here is a >little< ;) list.

The problems are:

- conversion MFM->serial

(problem of syncing - converter must be synced by a syncword and then follow this syncronisation)

- conversion serial->MFM

(converter must be able to handle serial syncing and convert to MFM using a syncword)

- shared access to diskport

(because the diskport must be shared with other tasks, incoming data must be buffered until the diskport is free)

- Paula's disk I/O-behaviour

(Paula is not able to send and receive data at the same point of time, but serial ports must be able to do this. e.g. xON/xOFF handshaking whould not be possible otherwise)

- Paula's fixed I/O length

(When Paula does I/O, you have to set the I/O length before a transmission - it can't be changed during a transmission. This causes two problems:
During sending, data must be buffered in the converter when the (RS232-)receiver interrupts or stops input processing.
During receiving, data must be buffered, because Paula can only read data in fixed-length-packets.)

- RS232 Baud rates

(Paula's "Baud Rate" can only be 250000 or 500000 Baud (mfm), others are not supported (afaik).
But those are unfortunately not supported by RS232, so again buffering must be used.)

Of course, there are many other (minor) problems, so this list is not complete.

1.5 Requirements

To run "diskserial.device", you need ...

- A couple of Amigas
- Kickstart 2.0 or higher on each
- A free disk port on each (i.e. max. three floppies connected to it)
- a special cable

That's all!

1.6 Cable Assembly Instructions

WARNING: You could damage your computer.

I am not responsible for any damage that could be

caused by this hardware.

To build the cable, you will need:

- 2x 23pol. SUB-D connectors, male

- 2x cases for them

- 1x 9pol. data cable (max. 3 meters)

- 1x 74LS365 TTL-Circuit (DIL case)

- 2x (germanium) diodes

- 2x resistors, 1,2 KOhm

"Where to put the TTL?" you'll ask.

This mini-circuit exactly fits into one of the connector cases.

And I mean EXACTLY !! ;-)

Look at the schematic diagram.

If you like to connect more than two Amigas (the software is able to manage it), there is a modification you have to make on the circuit:

Like every external Amiga floppy has a "through"-connector to plug the next external floppy in, you need to add one to the circuit by ...

- using a bigger connector case at one end of the cable (e.g. a so-called "interface" case).

It has two ends where connectors can be fixed.

- buying another 23pol. SUB-D conn., but this time a female one

- putting ALL lines (1:1) through from the first (male) to the second (female) connector,

except the \SELxB signals (pins 9, 20, 21).

- the \SELxB signals must be connected as follows:

male connector female connector

pin 9 ---------- pin 21

pin 20 ---------- pin 9

pin 21 ONLY connected N.C. pin 20

to the TTL and

the resistor

1.7 Hard- and Software Installation

You first have to make the cable , then you can install the software by executing

Copy diskserial.device to devs:

Easy, hmm?

1.8 User's manual

In general, the "diskserial.device" can be used just like the standard "serial.device", but there are few differences you should know about.

Especially, the unit number has a special meaning when you OpenDevice() it. It depends on where the cable is plugged into your Amiga.

E.g. when you plug the cable into the disk port of an Amiga1200 with no external drive, then you have to open with unit number 1.

Then, the diskserial.device knows, that the cable is plugged there, where disk drive df1: would be normally plugged.

If you have an Amiga4000 with two internal and one external drive, and you plug the cable into the connector of your external drive, then you have to open with unit number 3.

Easy, isn't it?

You see that it IS possible to plug into your computer up to three diskserial cables (unit 0 is always occupied by the standard internal floppy). But therefore, you have to modify the cable as described here .

When you do so, it is also possible to "mix up" diskserial cables and floppies, i.e. for example, df0:, and df2: are floppies, and diskserial cables are connected to the units 1 and 3.

You should know that diskserial.device makes excessively use of the Blitter when transmissions take place.

Disk operations are not disturbed by transmissions, but delayed, because diskserial.device then "owns" the disk port.

1.9 Differences to serial.device

This is a - as I hope complete - list of the differences between diskserial.device and serial.device.

If there is no difference, it is not listed here, so please read the serial.device documentation first!

OpenDevice()

- The unit number has a special meaning .

- 7WIRE handshaking is not supported, i.e. DSR, CTS, RTS, DTR are always

active, CD is always inactive.

- Currently, the device always opens up in XDISABLED mode, so if you

want to enable the xON/xOFF protocol, you have to do it by sending

a SDCMD_SETPARAMS.

- no PARITY feature is implemented

- io_RBufLen, io_StopBits are ignored

- io_Baud is ignored, but you should set it to 250000

CloseDevice()

- data that has not been sended so far, will be lost.

To avoid that, you should DoIO() a CMD_FLUSH before CloseDevice().

BeginIO()

- read and write requests are always queued, they will never return

immediately.

CMD_READ

- is always queued

CMD_RESET
- has actually no effect and returns IOERR_NOCMD
CMD_START
CMD_STOP
- xON/xOFF protocol is emulated (I hope so), but takes no
effect on the transmission protocol, for diskserial.device
has its own protocol.
CMD_WRITE
- is always queued
- the request returns after the data was transferred into diskserial's
output buffer. This doesn't mean that it was already sended when the
request returns.
To be sure that it was sended to the other side before a certain
point of time, use CMD_FLUSH.
SDCMD_QUERY
- because diskserial.device uses some kind of double-buffering method,
it could happen that the io_Actual field does not return the total
amount of unread bytes in the input buffers.
Safe is to call SDCMD_QUERY and afterwards CMD_READ until SDCMD_QUERY
returns 0 in the io_Actual field.
- DSR, CTS, RTS, DTR are always active (low)
- CD is always inactive (high)
SDCMD_SETPARAMS
- the read buffer length (io_RBufLen) is fix and can't be changed
- io_Baud is ignored, but you should set it to 250000
- io_StopBits is ignored
- PARITY flags are ignored
- 7WIRE handshaking is ignored

1.10 Bugs/Known Problems

Though I could not find any real bugs, you should take this program as very
buggy.
This is because I had few time to spend on testing it.
But there is one VERY IMPORTANT (hardware) bug:
If you use a "Real HD Floppy" from Amtrade, the diskserial.device will
probably not work.
In this case you have to correct one small hardware bug in Amtrades shuttle
platine:

The GAL on the shuttle platine generates the \INDEX (pin 8 on disk conn.) signal
instead of the floppy. This should be an open collector output, but it isn't.
The easiest way to correct this is to insert a diode (best germanium) into the
wire between the GAL and the disk connector that directs to the computer.
I also had problems to get it working together with Envoy1.3, but
when I heard that Envoy1.3 is so buggy, I didn't believe that this is
because of the diskserial.device.
Can anybody test it together with Envoy2.0? I had no chance to do so.

1.11 diskserial.device's evolution history

V1.2a ----------
Timing protocol little changed. "Short-packet" transmissions
should be a bit faster now.
V1.2 ----------
I fixed the bug of V1.1/V1.1a, so the transmission should be
more secure now.
V1.1a ----------
MAXIMA MEA CULPA MAXIMA MEA CULPA MAXIMA MEA CULPA
When I removed all debugging code from diskserial.device V1.1
for the Aminet, I removed a part of the transmission protocol
(sender) by mistake, so that V1.1 DID NOT WORK.
In this version, I removed the appropriate parts on receiver
side, so that V1.1a should work now, but the transmission
will be a bit insecure.

1.12 About the Author...

I am very pleased you want to know something about me! :-))))
To contact me, write to....
eMail: sichler@rumms.uni-mannheim.de
or stefan@ds.domino.de
Postal address: Stefan Sichler
Messkircher Str. 49
68239 Mannheim
Germany