specialix.txt -- specialix IO8+ multiport serial driver readme.

Specialix pays for the development and support of this driver.
Please DO contact io8-linux@specialix.co.uk if you require
support.

This driver was developed in the BitWizard linux device
driver service. If you require a linux device driver for your
product, please contact devices@BitWizard.nl for a quote.

This code is firmly based on the riscom/8 serial driver,
written by Dmitry Gorodchanin. The specialix IO8+ card
programming information was obtained from the CL-CD1865 Data
Book, and Specialix document number 6200059: IO8+ Hardware
Functional Specification, augmented by document number 6200088:
Merak Hardware Functional Specification. (IO8+/PCI is also
called Merak)

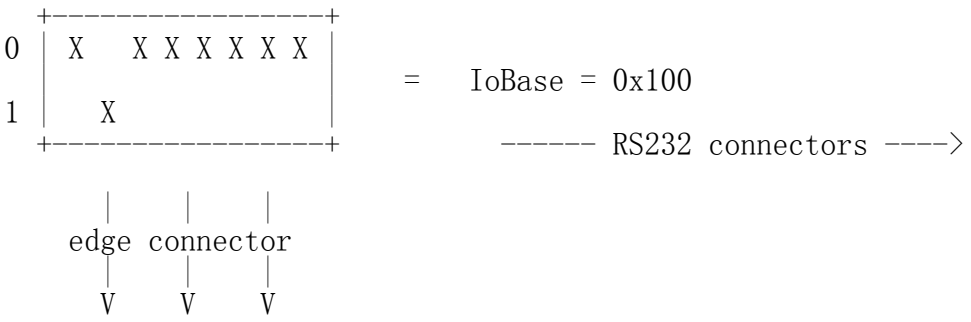
Intro

This file contains some random information, that I like to have online
instead of in a manual that can get lost. Ever misplace your Linux
kernel sources? And the manual of one of the boards in your computer?

Addresses and interrupts

Address dip switch settings:
The dip switch sets bits 2-9 of the IO address.

9 8 7 6 5 4 3 2



Base address 0x100 caused a conflict in one of my computers once. I haven't the foggiest why. My Specialix card is now at 0x180. My other computer runs just fine with the Specialix card at 0x100.... The card occupies 4 addresses, but actually only two are really used.

The PCI version doesn't have any dip switches. The BIOS assigns an IO address.

The driver now still autoprobes at 0x100, 0x180, 0x250 and 0x260. If that causes trouble for you, please report that. I'll remove autoprobing then.

The driver will tell the card what IRQ to use, so you don't have to change any jumpers to change the IRQ. Just use a command line argument (irq=xx) to the insmod program to set the interrupt.

The BIOS assigns the IRQ on the PCI version. You have no say in what IRQ to use in that case.

If your specialix cards are not at the default locations, you can use the kernel command line argument "specialix=io0,irq0,io1,irq1...". Here "io0" is the io address for the first card, and "irq0" is the irq line that the first card should use. And so on.

Examples.

You use the driver as a module and have three cards at 0x100, 0x250 and 0x180. And some way or another you want them detected in that order. Moreover irq 12 is taken (e.g. by your PS/2 mouse).

```
insmod specialix.o iobase=0x100,0x250,0x180 irq=9,11,15
```

The same three cards, but now in the kernel would require you to add

```
specialix=0x100,9,0x250,11,0x180,15
```

to the command line. This would become

```
append="specialix=0x100,9,0x250,11,0x180,15"
```

in your /etc/lilo.conf file if you use lilo.

The Specialix driver is slightly odd: It allows you to have the second or third card detected without having a first card. This has

advantages and disadvantages. A slot that isn't filled by an ISA card, might be filled if a PCI card is detected. Thus if you have an ISA card at 0x250 and a PCI card, you would get:

```
sx0: specialix IO8+ Board at 0x100 not found.
sx1: specialix IO8+ Board at 0x180 not found.
sx2: specialix IO8+ board detected at 0x250, IRQ 12, CD1865 Rev. B.
sx3: specialix IO8+ Board at 0x260 not found.
sx0: specialix IO8+ board detected at 0xd800, IRQ 9, CD1865 Rev. B.
```

This would happen if you don't give any probe hints to the driver. If you would specify:

```
   specialix=0x250,11
```

you'd get the following messages:

```
sx0: specialix IO8+ board detected at 0x250, IRQ 11, CD1865 Rev. B.
sx1: specialix IO8+ board detected at 0xd800, IRQ 9, CD1865 Rev. B.
```

ISA probing is aborted after the IO address you gave is exhausted, and the PCI card is now detected as the second card. The ISA card is now also forced to IRQ11....

Baud rates

The rev 1.2 and below boards use a CL-CD1864. These chips can only do 64kbit. The rev 1.3 and newer boards use a CL-CD1865. These chips are officially capable of 115k2.

The Specialix card uses a 25MHz crystal (in times two mode, which in fact is a divided by two mode). This is not enough to reach the rated 115k2 on all ports at the same time. With this clock rate you can only do 37% of this rate. This means that at 115k2 on all ports you are going to lose characters (The chip cannot handle that many incoming bits at this clock rate.) (Yes, you read that correctly: there is a limit to the number of -=bits=- per second that the chip can handle.)

If you near the "limit" you will first start to see a graceful degradation in that the chip cannot keep the transmitter busy at all times. However with a central clock this slow, you can also get it to miss incoming characters. The driver will print a warning message when you are outside the official specs. The messages usually show up in the file /var/log/messages .

The specialix card cannot reliably do 115k2. If you use it, you have to do "extensive testing" (*) to verify if it actually works.

When "mgetty" communicates with my modem at 115k2 it reports:

```
got: +++[0d]ATQ0V1H0[0d][0d][8a]0[cb][0d][8a]
                             ^^^^ ^^^^    ^^^^
```

The three characters that have the "^^^^" under them have suffered a bit error in the highest bit. In conclusion: I've tested it, and found

that it simply DOESN'T work for me. I also suspect that this is also caused by the baud rate being just a little bit out of tune.

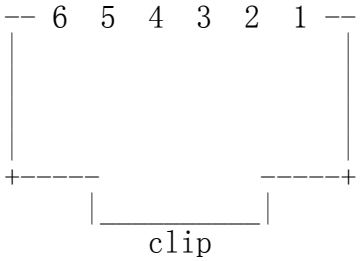
I upgraded the crystal to 66Mhz on one of my Specialix cards. Works great! Contact me for details. (Voids warranty, requires a steady hand and more such restrictions....)

(*) Cirrus logic CD1864 databook, page 40.

Cables for the Specialix IO8+
==========

The pinout of the connectors on the IO8+ is:

pin	short name	direction	long name
Pin 1	DCD	input	Data Carrier Detect
Pin 2	RXD	input	Receive
Pin 3	DTR/RTS	output	Data Terminal Ready/Ready To Send
Pin 4	GND	-	Ground
Pin 5	TXD	output	Transmit
Pin 6	CTS	input	Clear To Send



Front view of an RJ12 connector. Cable moves "into" the paper. (the plug is ready to plug into your mouth this way...)

NULL cable. I don't know who is going to use these except for testing purposes, but I tested the cards with this cable. (It took quite a while to figure out, so I'm not going to delete it. So there! :-)

This end goes straight into the RJ12 plug. IO8+	RJ12		This end needs some twists in the wiring. IO8+	RJ12
1	DCD	white	-	
-		-	1	DCD
2	RXD	black	5	TXD
3	DTR/RTS	red	6	CTS
4	GND	green	4	GND
5	TXD	yellow	2	RXD
6	CTS	blue	3	DTR/RTS

Same NULL cable, but now sorted on the second column.

1	DCD	white	-
-		-	1 DCD
5	TXD	yellow	2 RXD
6	CTS	blue	3 DTR/RTS
4	GND	green	4 GND
2	RXD	black	5 TXD
3	DTR/RTS	red	6 CTS

This is a modem cable usable for hardware handshaking:

RJ12				DB25	DB9
1	DCD	white		8 DCD	1 DCD
2	RXD	black		3 RXD	2 RXD
3	DTR/RTS	red		4 RTS	7 RTS
4	GND	green		7 GND	5 GND
5	TXD	yellow		2 TXD	3 TXD
6	CTS	blue		5 CTS	8 CTS
			+----	6 DSR	6 DSR
			+----	20 DTR	4 DTR

This is a modem cable usable for software handshaking:
It allows you to reset the modem using the DTR ioctls.
I (REW) have never tested this, "but xxxxxxxxxxxx
says that it works." If you test this, please
tell me and I'll fill in your name on the xxx's.

RJ12				DB25	DB9
1	DCD	white		8 DCD	1 DCD
2	RXD	black		3 RXD	2 RXD
3	DTR/RTS	red		20 DTR	4 DTR
4	GND	green		7 GND	5 GND
5	TXD	yellow		2 TXD	3 TXD
6	CTS	blue		5 CTS	8 CTS
			+----	6 DSR	6 DSR
			+----	4 RTS	7 RTS

I bought a 6 wire flat cable. It was colored as indicated.
Check that yours is the same before you trust me on this.

Hardware handshaking issues.
==========

The driver can be told to operate in two different ways. The default behaviour is specialix.sx_rtscts = 0 where the pin behaves as DTR when hardware handshaking is off. It behaves as the RTS hardware handshaking signal when hardware handshaking is selected.

When you use this, you have to use the appropriate cable. The cable will either be compatible with hardware handshaking or with software handshaking. So switching on the fly is not really an

option.

I actually prefer to use the "specialix.sx_rtscts=1" option.
This makes the DTR/RTS pin always an RTS pin, and ioctls to
change DTR are always ignored. I have a cable that is configured
for this.

Ports and devices
=================

Port 0 is the one furthest from the card-edge connector.

Devices:

You should make the devices as follows:

```
bash
cd /dev
for i in  0  1  2  3  4  5  6  7  8  9 10 11 12 13 14 15 \
         16 17 18 19 20 21 22 23 24 25 26 27 28 29 30 31
do
  echo -n "$i "
  mknod /dev/ttyW$i c 75 $i
  mknod /dev/cuw$i c 76 $i
done
echo ""
```

If your system doesn't come with these devices preinstalled, bug your
linux-vendor about this. They have had ample time to get this
implemented by now.

You cannot have more than 4 boards in one computer. The card only
supports 4 different interrupts. If you really want this, contact me
about this and I'll give you a few tips (requires soldering iron)....

If you have enough PCI slots, you can probably use more than 4 PCI
versions of the card though....

The PCI version of the card cannot adhere to the mechanical part of
the PCI spec because the 8 serial connectors are simply too large. If
it doesn't fit in your computer, bring back the card.

Fixed bugs and restrictions:

- During initialization, interrupts are blindly turned on.
  Having a shadow variable would cause an extra memory
  access on every IO instruction.
- The interrupt (on the card) should be disabled when we
  don't allocate the Linux end of the interrupt. This allows
  a different driver/card to use it while all ports are not in
  use..... (a la standard serial port)

== An extra _off variant of the sx_in and sx_out macros are

now available. They don't set the interrupt enable bit.
These are used during initialization. Normal operation uses
the old variant which enables the interrupt line.
- RTS/DTR issue needs to be implemented according to
specialix' spec.
I kind of like the "determinism" of the current
implementation. Compile time flag?
== Ok. Compile time flag! Default is how Specialix likes it.
== Now a config time flag! Gets saved in your config file. Neat!
- Can you set the IO address from the lilo command line?
If you need this, bug me about it, I'll make it.
== Hah! No bugging needed. Fixed! :-)
- Cirrus logic hasn't gotten back to me yet why the CD1865 can
and the CD1864 can't do 115k2. I suspect that this is
because the CD1864 is not rated for 33MHz operation.
Therefore the CD1864 versions of the card can't do 115k2 on
all ports just like the CD1865 versions. The driver does
not block 115k2 on CD1864 cards.
== I called the Cirrus Logic representative here in Holland.
The CD1864 databook is identical to the CD1865 databook,
except for an extra warning at the end. Similar Bit errors
have been observed in testing at 115k2 on both an 1865 and
a 1864 chip. I see no reason why I would prohibit 115k2 on
1864 chips and not do it on 1865 chips. Actually there is
reason to prohibit it on BOTH chips. I print a warning.
If you use 115k2, you're on your own.
- A spiky CD may send spurious HUPs. Also in CLOCAL???
-- A fix for this turned out to be counter productive.
Different fix? Current behaviour is acceptable?
-- Maybe the current implementation is correct. If anybody
gets bitten by this, please report, and it will get fixed.

-- Testing revealed that when in CLOCAL, the problem doesn't
occur. As warned for in the CD1865 manual, the chip may
send modem intr's on a spike. We could filter those out,
but that would be a cludge anyway (You'd still risk getting
a spurious HUP when two spikes occur.).....

Bugs & restrictions:
- This is a difficult card to autoprobe.
You have to WRITE to the address register to even
read-probe a CD186x register. Disable autodetection?
-- Specialix: any suggestions?