sx.txt -- specialix SX/SI multiport serial driver readme.



Specialix pays for the development and support of this driver.
Please DO contact support@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.

(History)
There used to be an SI driver by Simon Allan. This is a complete
rewrite from scratch. Just a few lines-of-code have been snatched.

(Sources)
Specialix document number 6210028: SX Host Card and Download Code
Software Functional Specification.



(Addendum)
I'd appreciate it that if you have fixes, that you send them
to me first.

## Introduction

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?

## Theory of operation

An important thing to know is that the driver itself doesn't have the

firmware for the card. This means that you need the separate package "sx_firmware". For now you can get the source at

ftp://ftp.bitwizard.nl/specialix/sx_firmware_<version>.tgz

The firmware load needs a "misc" device, so you'll need to enable the "Support for user misc device modules" in your kernel configuration. The misc device needs to be called "/dev/specialix_sxctl". It needs misc major 10, and minor number 167 (assigned by HPA). The section on creating device files below also creates this device.

After loading the sx.o module into your kernel, the driver will report the number of cards detected, but because it doesn't have any firmware, it will not be able to determine the number of ports. Only when you then run "sx_firmware" will the firmware be downloaded and the rest of the driver initialized. At that time the sx_firmware program will report the number of ports installed.

In contrast with many other multi port serial cards, some of the data structures are only allocated when the card knows the number of ports that are connected. This means we won't waste memory for 120 port descriptor structures when you only have 8 ports. If you experience problems due to this, please report them: I haven't seen any.

Interrupts
==========

A multi port serial card, would generate a horrendous amount of interrupts if it would interrupt the CPU for every received character. Even more than 10 years ago, the trick not to use interrupts but to poll the serial cards was invented.

The SX card allow us to do this two ways. First the card limits its own interrupt rate to a rate that won't overwhelm the CPU. Secondly, we could forget about the cards interrupt completely and use the internal timer for this purpose.

Polling the card can take up to a few percent of your CPU. Using the interrupts would be better if you have most of the ports idle. Using timer-based polling is better if your card almost always has work to do. You save the separate interrupt in that case.

In any case, it doesn't really matter all that much.

The most common problem with interrupts is that for ISA cards in a PCI system the BIOS has to be told to configure that interrupt as "legacy ISA". Otherwise the card can pull on the interrupt line all it wants but the CPU won't see this.

If you can't get the interrupt to work, remember that polling mode is more efficient (provided you actually use the card intensively).

Allowed Configurations
======================

Some configurations are disallowed. Even though at a glance they might seem to work, they are known to lockup the bus between the host card and the device concentrators. You should respect the drivers decision not to support certain configurations. It's there for a reason.

Warning: Seriously technical stuff ahead. Executive summary: Don't use SX cards except configured at a 64k boundary. Skip the next paragraph.

The SX cards can theoretically be placed at a 32k boundary. So for instance you can put an SX card at 0xc8000-0xd7fff. This is not a "recommended configuration". ISA cards have to tell the bus controller how they like their timing. Due to timing issues they have to do this based on which 64k window the address falls into. This means that the 32k window below and above the SX card have to use exactly the same timing as the SX card. That reportedly works for other SX cards. But you're still left with two useless 32k windows that should not be used by anybody else.

## Configuring the driver

PCI cards are always detected. The driver auto-probes for ISA cards at some sensible addresses. Please report if the auto-probe causes trouble in your system, or when a card isn't detected.

I'm afraid I haven't implemented "kernel command line parameters" yet. This means that if the default doesn't work for you, you shouldn't use the compiled-into-the-kernel version of the driver. Use a module instead. If you convince me that you need this, I'll make it for you. Deal?

I'm afraid that the module parameters are a bit clumsy. If you have a better idea, please tell me.

You can specify several parameters:

sx_poll: number of jiffies between timer-based polls.

Set this to "0" to disable timer based polls. Initialization of cards without a working interrupt will fail.

Set this to "1" if you want a polling driver. (on Intel: 100 polls per second). If you don't use fast baud rates, you might consider a value like "5". (If you don't know how to do the math, use 1).

sx_slowpoll: Number of jiffies between timer-based polls.
Set this to "100" to poll once a second. This should get the card out of a stall if the driver ever misses an interrupt. I've never seen this happen, and if it does, that's a bug. Tell me.

sx_maxints: Number of interrupts to request from the card.

The card normally limits interrupts to about 100 per second to offload the host CPU. You can increase this number to reduce latency on the card a little.
Note that if you give a very high number you can overload your CPU as well as the CPU on the host card. This setting is inaccurate and not recommended for SI cards (But it works).

sx_irqmask: The mask of allowable IRQs to use. I suggest you set this to 0 (disable IRQs all together) and use polling if the assignment of IRQs becomes problematic. This is defined as the sum of (1 << irq) 's that you want to allow. So sx_irqmask of 8 (1 << 3) specifies that only irq 3 may be used by the SX driver. If you want to specify to the driver: "Either irq 11 or 12 is ok for you to use", then specify (1 << 11) | (1 << 12) = 0x1800 .

sx_debug: You can enable different sorts of debug traces with this. At "-1" all debugging traces are active. You'll get several times more debugging output than you'll get characters transmitted.

## Baud rates

Theoretically new SXDCs should be capable of more than 460k baud. However the line drivers usually give up before that. Also the CPU on the card may not be able to handle 8 channels going at full blast at that speed. Moreover, the buffers are not large enough to allow operation with 100 interrupts per second. You'll have to realize that the card has a 256 byte buffer, so you'll have to increase the number of interrupts per second if you have more than 256*100 bytes per second to transmit. If you do any performance testing in this area, I'd be glad to hear from you...

(Psst Linux users..... I think the Linux driver is more efficient than the driver for other OSes. If you can and want to benchmark them against each other, be my guest, and report your findings..... :-)

## Ports and devices

Port 0 is the top connector on the module closest to the host card. Oh, the ports on the SXDCs and TAs are labelled from 1 to 8 instead of from 0 to 7, as they are numbered by linux. I'm stubborn in this: I know for sure that I wouldn't be able to calculate which port is which anymore if I would change that....

Devices:

You should make the device files as follows:

```
#!/bin/sh
```

```
# (I recommend that you cut-and-paste this into a file and run that)
cd /dev
t=0
mknod specialix_sxctl c 10 167
while [ $t -lt 64 ]
  do
  echo -n "$t "
  mknod ttyX$t c 32 $t
  mknod cux$t  c 33 $t
  t=`expr $t + 1`
done
echo ""
rm /etc/psdevtab
ps > /dev/null
```

This creates 64 devices. If you have more, increase the constant on the line with "while". The devices start at 0, as is customary on Linux. Specialix seems to like starting the numbering at 1.

If your system doesn't come with these devices pre-installed, bug your linux-vendor about this. They should have these devices "pre-installed" before the new millennium. The "ps" stuff at the end is to "tell" ps that the new devices exist.

Officially the maximum number of cards per computer is 4. This driver however supports as many cards in one machine as you want. You'll run out of interrupts after a few, but you can switch to polled operation then. At about 256 ports (More than 8 cards), we run out of minor device numbers. Sorry. I suggest you buy a second computer.... (Or switch to RIO).

---

Fixed bugs and restrictions:

- Hangup processing.
  -- Done.

- the write path in generic_serial (lockup / oops).
  -- Done (Ugly: not the way I want it. Copied from serial.c).

- write buffer isn't flushed at close.
  -- Done. I still seem to lose a few chars at close.
     Sorry. I think that this is a firmware issue. (-> Specialix)

- drain hardware before  changing termios
- Change debug on the fly.
- ISA free irq -1. (no firmware loaded).
- adding c8000 as a probe address. Added warning.
- Add a RAMtest for the RAM on the card.c
- Crash when opening a port "way" of the number of allowed ports.
  (for example opening port 60 when there are only 24 ports attached)
- Sometimes the use-count strays a bit. After a few hours of
  testing the use count is sometimes "3". If you are not like
  me and can remember what you did to get it that way, I'd

appreciate an Email. Possibly fixed. Tell me if anyone still
sees this.
- TAs don't work right if you don't connect all the modem control
signals. SXDCs do. T225 firmware problem -> Specialix.
(Mostly fixed now, I think. Tell me if you encounter this!)

Bugs & restrictions:

- Arbitrary baud rates. Requires firmware update. (-> Specialix)

- Low latency (mostly firmware, -> Specialix)