

# **SiS 900/7016 Fast Ethernet Device Driver**

**Ollie Lho**

**Lei Chun Chang**

## **SiS 900/7016 Fast Ethernet Device Driver**

by Ollie Lho and Lei Chun Chang

Document Revision: 0.3 for SiS900 driver v1.06 & v1.07 Edition

Published November 16, 2000

Copyright © 1999 Silicon Integrated System Corp.

This document gives some information on installation and usage of SiS 900/7016 device driver under Linux.

This program is free software; you can redistribute it and/or modify it under the terms of the GNU General Public License as published by the Free Software Foundation; either version 2 of the License, or (at your option) any later version.

This program is distributed in the hope that it will be useful, but WITHOUT ANY WARRANTY; without even the implied warranty of MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE. See the GNU General Public License for more details.

You should have received a copy of the GNU General Public License along with this program; if not, write to the Free Software Foundation, Inc., 59 Temple Place, Suite 330, Boston, MA 02111-1307 USA

# Table of Contents

<b>1. Introduction</b> .....	<b>1</b>
<b>2. Changes</b> .....	<b>2</b>
<b>3. Tested Environment</b> .....	<b>3</b>
<b>4. Files in This Package</b> .....	<b>4</b>
<b>5. Installation</b> .....	<b>5</b>
5.1. Building the driver as loadable module.....	5
5.2. Building the driver into kernel .....	7
<b>6. Known Problems and Bugs</b> .....	<b>8</b>
<b>7. Revision History</b> .....	<b>9</b>
<b>8. Acknowledgements</b> .....	<b>10</b>
<b>9. List of Functions</b> .....	<b>11</b>
sis900_get_mac_addr.....	11
sis630e_get_mac_addr .....	11
sis635_get_mac_addr.....	12
sis96x_get_mac_addr.....	13
sis900_probe .....	14
sis900_mii_probe .....	14
sis900_default_phy .....	15
sis900_set_capability .....	16
read_eeeprom.....	17
mdio_read.....	17
mdio_write .....	18
sis900_reset_phy .....	19
sis900_open.....	20
sis900_init_rxfilter .....	20
sis900_init_tx_ring.....	21
sis900_init_rx_ring .....	21
sis630_set_eq .....	22
sis900_timer .....	24
sis900_check_mode .....	24
sis900_set_mode .....	25
sis900_auto_negotiate.....	26
sis900_read_mode.....	27
sis900_tx_timeout .....	27
sis900_start_xmit .....	28
sis900_interrupt.....	29
sis900_rx .....	29
sis900_finish_xmit .....	30
sis900_close .....	31
netdev_ethtool_ioctl.....	32
mii_ioctl .....	32
sis900_get_stats .....	33
sis900_set_config.....	34
sis900_mcast_bitnr.....	34

set_rx_mode .....	35
sis900_reset .....	36
sis900_remove .....	36

# Chapter 1. Introduction

This document describes the revision 1.06 and 1.07 of SiS 900/7016 Fast Ethernet device driver under Linux. The driver is developed by Silicon Integrated System Corp. and distributed freely under the GNU General Public License (GPL). The driver can be compiled as a loadable module and used under Linux kernel version 2.2.x. (rev. 1.06) With minimal changes, the driver can also be used under 2.3.x and 2.4.x kernel (rev. 1.07), please see Chapter 5. If you are intended to use the driver for earlier kernels, you are on your own.

The driver is tested with usual TCP/IP applications including FTP, Telnet, Netscape etc. and is used constantly by the developers.

Please send all comments/fixes/questions to Lei-Chun Chang (<mailto:lcchang@sis.com.tw>).

# Chapter 2. Changes

Changes made in Revision 1.07

1. Separation of sis900.c and sis900.h in order to move most constant definition to sis900.h (many of those constants were corrected)
2. Clean up PCI detection, the pci-scan from Donald Becker were not used, just simple pci\_find\_\*.
3. MII detection is modified to support multiple mii transceiver.
4. Bugs in read\_eeprom, mdio\_\* were removed.
5. Lot of sis900 irrelevant comments were removed/changed and more comments were added to reflect the real situation.
6. Clean up of physical/virtual address space mess in buffer descriptors.
7. Better transmit/receive error handling.
8. The driver now uses zero-copy single buffer management scheme to improve performance.
9. Names of variables were changed to be more consistent.
10. Clean up of auto-negotiation and timer code.
11. Automatic detection and change of PHY on the fly.
12. Bug in mac probing fixed.
13. Fix 630E equalizer problem by modifying the equalizer workaround rule.
14. Support for ICS1893 10/100 Interated PHYceiver.
15. Support for media select by ifconfig.
16. Added kernel-doc extratable documentation.

# Chapter 3. Tested Environment

This driver is developed on the following hardware

- Intel Celeron 500 with SiS 630 (rev 02) chipset
- SiS 900 (rev 01) and SiS 7016/7014 Fast Ethernet Card

and tested with these software environments

- Red Hat Linux version 6.2
- Linux kernel version 2.4.0
- Netscape version 4.6
- NcFTP 3.0.0 beta 18
- Samba version 2.0.3

# Chapter 4. Files in This Package

In the package you can find these files:

sis900.c

Driver source file in C

sis900.h

Header file for sis900.c

sis900.sgml

DocBook SGML source of the document

sis900.txt

Driver document in plain text

# Chapter 5. Installation

Silicon Integrated System Corp. is cooperating closely with core Linux Kernel developers. The revisions of SiS 900 driver are distributed by the usual channels for kernel tar files and patches. Those kernel tar files for official kernel and patches for kernel pre-release can be download at official kernel ftp site (<http://ftp.kernel.org/pub/linux/kernel/>) and its mirrors. The 1.06 revision can be found in kernel version later than 2.3.15 and pre-2.2.14, and 1.07 revision can be found in kernel version 2.4.0. If you have no prior experience in networking under Linux, please read Ethernet HOWTO (<http://www.tldp.org/>) and Networking HOWTO (<http://www.tldp.org/>) available from Linux Documentation Project (LDP).

The driver is bundled in release later than 2.2.11 and 2.3.15 so this is the most easy case. Be sure you have the appropriate packages for compiling kernel source. Those packages are listed in Document/Changes in kernel source distribution. If you have to install the driver other than those bundled in kernel release, you should have your driver file `sis900.c` and `sis900.h` copied into `/usr/src/linux/drivers/net/` first. There are two alternative ways to install the driver

## 5.1. Building the driver as loadable module

To build the driver as a loadable kernel module you have to reconfigure the kernel to activate network support by

```
make menuconfig
```

Choose “Loadable module support --->”, then select “Enable loadable module support”.

Choose “Network Device Support --->”, select “Ethernet (10 or 100Mbit)”. Then select “EISA, VLB, PCI and on board controllers”, and choose “SiS 900/7016 PCI Fast Ethernet Adapter support” to “M”.

After reconfiguring the kernel, you can make the driver module by

```
make modules
```

The driver should be compiled with no errors. After compiling the driver, the driver can be installed to proper place by

```
make modules_install
```

Load the driver into kernel by

```
insmod sis900
```

When loading the driver into memory, some information message can be view by

```
dmesg
```

or

```
cat /var/log/message
```

If the driver is loaded properly you will have messages similar to this:

```
sis900.c: v1.07.06 11/07/2000  
eth0: SiS 900 PCI Fast Ethernet at 0xd000, IRQ 10, 00:00:e8:83:7f:a4.  
eth0: SiS 900 Internal MII PHY transceiver found at address 1.  
eth0: Using SiS 900 Internal MII PHY as default
```

showing the version of the driver and the results of probing routine.

Once the driver is loaded, network can be brought up by

```
/sbin/ifconfig eth0 IPADDR broadcast BROADCAST netmask NETMASK media TYPE
```

where IPADDR, BROADCAST, NETMASK are your IP address, broadcast address and netmask respectively. TYPE is used to set medium type used by the device. Typical values are "10baseT"(twisted-pair 10Mbps Ethernet) or "100baseT" (twisted-pair 100Mbps Ethernet). For more information on how to configure network interface, please refer to Networking HOWTO (<http://www.tldp.org/>).

The link status is also shown by kernel messages. For example, after the network interface is activated, you may have the message:

```
eth0: Media Link On 100mbps full-duplex
```

If you try to unplug the twist pair (TP) cable you will get

```
eth0: Media Link Off
```

indicating that the link is failed.

## 5.2. Building the driver into kernel

If you want to make the driver into kernel, choose “Y” rather than “M” on “SiS 900/7016 PCI Fast Ethernet Adapter support” when configuring the kernel. Build the kernel image in the usual way

```
make dep  
make clean  
make bzlilo
```

Next time the system reboot, you have the driver in memory.

# Chapter 6. Known Problems and Bugs

There are some known problems and bugs. If you find any other bugs please mail to [lcchang@sis.com.tw](mailto:lcchang@sis.com.tw) (<mailto:lcchang@sis.com.tw>)

1. AM79C901 HomePNA PHY is not thoroughly tested, there may be some bugs in the “on the fly” change of transceiver.
2. A bug is hidden somewhere in the receive buffer management code, the bug causes NULL pointer reference in the kernel. This fault is caught before bad things happen and reported with the message:  
`eth0: NULL pointer encountered in Rx ring, skipping` which can be viewed with `dmesg` or `cat /var/log/message`.
3. The media type change from 10Mbps to 100Mbps twisted-pair ethernet by `ifconfig` causes the media link down.

# Chapter 7. Revision History

- November 13, 2000, Revision 1.07, seventh release, 630E problem fixed and further clean up.
- November 4, 1999, Revision 1.06, Second release, lots of clean up and optimization.
- August 8, 1999, Revision 1.05, Initial Public Release

## Chapter 8. Acknowledgements

This driver was originally derived from Donald Becker (<mailto:becker@cesdis1.gsfc.nasa.gov>)'s pci-skeleton (<ftp://cesdis.gsfc.nasa.gov/pub/linux/drivers/kern-2.3/pci-skeleton.c>) and rtl8139 (<ftp://cesdis.gsfc.nasa.gov/pub/linux/drivers/kern-2.3/rtl8139.c>) drivers. Donald also provided various suggestion regarded with improvements made in revision 1.06.

The 1.05 revision was created by Jim Huang (<mailto:cmhuang@sis.com.tw>), AMD 79c901 support was added by Chin-Shan Li (<mailto:lcs@sis.com.tw>).

# Chapter 9. List of Functions

## sis900\_get\_mac\_addr

### Name

`sis900_get_mac_addr` — Get MAC address for stand alone SiS900 model

### Synopsis

```
int __devinit sis900_get_mac_addr (struct pci_dev * pci_dev, struct  
net_device * net_dev);
```

### Arguments

*pci\_dev*

the sis900 pci device

*net\_dev*

the net device to get address for

### Description

Older SiS900 and friends, use EEPROM to store MAC address. MAC address is read from `read_eeeprom` into `net_dev->dev_addr`.

## sis630e\_get\_mac\_addr

### Name

`sis630e_get_mac_addr` — Get MAC address for SiS630E model

## Synopsis

```
int __devinit sis630e_get_mac_addr (struct pci_dev * pci_dev, struct
net_device * net_dev);
```

## Arguments

*pci\_dev*

the sis900 pci device

*net\_dev*

the net device to get address for

## Description

SiS630E model, use APC CMOS RAM to store MAC address. APC CMOS RAM is accessed through ISA bridge. MAC address is read into *net\_dev->dev\_addr*.

## sis635\_get\_mac\_addr

### Name

*sis635\_get\_mac\_addr* — Get MAC address for SIS635 model

### Synopsis

```
int __devinit sis635_get_mac_addr (struct pci_dev * pci_dev, struct
net_device * net_dev);
```

### Arguments

*pci\_dev*

the sis900 pci device

*net\_dev*

the net device to get address for

## Description

SiS635 model, set MAC Reload Bit to load Mac address from APC to rfd. rfd is accessed through rfc. MAC address is read into *net\_dev->dev\_addr*.

## sis96x\_get\_mac\_addr

### Name

*sis96x\_get\_mac\_addr* — Get MAC address for SiS962 or SiS963 model

### Synopsis

```
int __devinit sis96x_get_mac_addr (struct pci_dev * pci_dev, struct
net_device * net_dev);
```

### Arguments

*pci\_dev*

the sis900 pci device

*net\_dev*

the net device to get address for

### Description

SiS962 or SiS963 model, use EEPROM to store MAC address. And EEPROM is shared by LAN and 1394. When access EEPROM, send EEREQ signal to hardware first and wait for EEGNT. If EEGNT is ON, EEPROM is permitted to be access by LAN, otherwise is not. After MAC address is read from EEPROM, send EEDONE signal to refuse EEPROM access by LAN. The EEPROM map of SiS962 or SiS963 is different to SiS900. The signature field in SiS962 or SiS963 spec is meaningless. MAC address is read into *net\_dev->dev\_addr*.

## sis900\_probe

### Name

sis900\_probe — Probe for sis900 device

### Synopsis

```
int __devinit sis900_probe (struct pci_dev * pci_dev, const struct  
pci_device_id * pci_id);
```

### Arguments

*pci\_dev*

the sis900 pci device

*pci\_id*

the pci device ID

### Description

Check and probe sis900 net device for *pci\_dev*. Get mac address according to the chip revision, and assign SiS900-specific entries in the device structure.

### ie

sis900\_open, sis900\_start\_xmit, sis900\_close, etc.

## sis900\_mii\_probe

### Name

`sis900_mii_probe` — Probe MII PHY for sis900

### Synopsis

```
int __init sis900_mii_probe (struct net_device * net_dev);
```

### Arguments

*net\_dev*

the net device to probe for

### Description

Search for total of 32 possible mii phy addresses. Identify and set current phy if found one, return error if it failed to found.

## sis900\_default\_phy

### Name

`sis900_default_phy` — Select default PHY for sis900 mac.

### Synopsis

```
u16 sis900_default_phy (struct net_device * net_dev);
```

## Arguments

*net\_dev*

the net device to probe for

## Description

Select first detected PHY with link as default. If no one is link on, select PHY whose types is HOME as default. If HOME doesn't exist, select LAN.

# sis900\_set\_capability

## Name

`sis900_set_capability` — set the media capability of network adapter.

## Synopsis

```
void sis900_set_capability (struct net_device * net_dev, struct mii_phy *
phy);
```

## Arguments

*net\_dev*

the net device to probe for

*phy*

default PHY

## Description

Set the media capability of network adapter according to mii status register. It's necessary before auto-negotiate.

## read\_eeprom

### Name

`read_eeprom` — Read Serial EEPROM

### Synopsis

```
u16 __devinit read_eeprom (long ioaddr, int location);
```

### Arguments

*ioaddr*

base i/o address

*location*

the EEPROM location to read

### Description

Read Serial EEPROM through EEPROM Access Register. Note that location is in word (16 bits) unit

## mdio\_read

### Name

`mdio_read` — read MII PHY register

### Synopsis

```
u16 mdio_read (struct net_device * net_dev, int phy_id, int location);
```

## Arguments

*net\_dev*

the net device to read

*phy\_id*

the phy address to read

*location*

the phy register id to read

## Description

Read MII registers through MDIO and MDC using MDIO management frame structure and protocol(defined by ISO/IEC). Please see SiS7014 or ICS spec

## mdio\_write

### Name

`mdio_write` — write MII PHY register

### Synopsis

```
void mdio_write (struct net_device * net_dev, int phy_id, int location, int value);
```

## Arguments

*net\_dev*

the net device to write

*phy\_id*

the phy address to write

*location*

the phy register id to write

*value*

the register value to write with

## Description

Write MII registers with *value* through MDIO and MDC using MDIO management frame structure and protocol(defined by ISO/IEC) please see SiS7014 or ICS spec

## sis900\_reset\_phy

### Name

`sis900_reset_phy` — reset sis900 mii phy.

### Synopsis

```
u16 sis900_reset_phy (struct net_device * net_dev, int phy_addr);
```

### Arguments

*net\_dev*

the net device to write

*phy\_addr*

default phy address

### Description

Some specific phy can't work properly without reset. This function will be called during initialization and link status change from ON to DOWN.

## sis900\_open

### Name

sis900\_open — open sis900 device

### Synopsis

```
int sis900_open (struct net_device * net_dev);
```

### Arguments

*net\_dev*

the net device to open

### Description

Do some initialization and start net interface. enable interrupts and set sis900 timer.

## sis900\_init\_rxfilter

### Name

sis900\_init\_rxfilter — Initialize the Rx filter

### Synopsis

```
void sis900_init_rxfilter (struct net_device * net_dev);
```

## Arguments

*net\_dev*

the net device to initialize for

## Description

Set receive filter address to our MAC address and enable packet filtering.

# sis900\_init\_tx\_ring

## Name

`sis900_init_tx_ring` — Initialize the Tx descriptor ring

## Synopsis

```
void sis900_init_tx_ring (struct net_device * net_dev);
```

## Arguments

*net\_dev*

the net device to initialize for

## Description

Initialize the Tx descriptor ring,

## sis900\_init\_rx\_ring

### Name

`sis900_init_rx_ring` — Initialize the Rx descriptor ring

### Synopsis

```
void sis900_init_rx_ring (struct net_device * net_dev);
```

### Arguments

*net\_dev*

the net device to initialize for

### Description

Initialize the Rx descriptor ring, and pre-allocate receive buffers (socket buffer)

## sis630\_set\_eq

### Name

`sis630_set_eq` — set phy equalizer value for 630 LAN

### Synopsis

```
void sis630_set_eq (struct net_device * net_dev, u8 revision);
```

## Arguments

*net\_dev*

the net device to set equalizer value

*revision*

630 LAN revision number

## Description

630E equalizer workaround rule(Cyrus Huang 08/15) PHY register 14h(Test)

### Bit 14

0 -- Automatically detect (default) 1 -- Manually set Equalizer filter

### Bit 13

0 -- (Default) 1 -- Speed up convergence of equalizer setting

### Bit 9

0 -- (Default) 1 -- Disable Baseline Wander Bit 3~7 -- Equalizer filter setting

## Link ON

Set Bit 9, 13 to 1, Bit 14 to 0 Then calculate equalizer value Then set equalizer value, and set Bit 14 to 1, Bit 9 to 0

## Link Off

Set Bit 13 to 1, Bit 14 to 0

## Calculate Equalizer value

When Link is ON and Bit 14 is 0, SIS900PHY will auto-detect proper equalizer value. When the equalizer is stable, this value is not a fixed value. It will be within a small range(eg. 7~9). Then we get a minimum and a maximum value(eg. min=7, max=9)  $0 \leq \max \leq 4 \rightarrow$  set equalizer to max  $5 \leq \max \leq 14 \rightarrow$  set equalizer to max+1 or set equalizer to max+2 if max == min  $\max \geq 15 \rightarrow$  set equalizer to max+5 or set equalizer to max+6 if max == min

## sis900\_timer

### Name

`sis900_timer` — sis900 timer routine

### Synopsis

```
void sis900_timer (unsigned long data);
```

### Arguments

*data*

pointer to sis900 net device

### Description

On each timer ticks we check two things, link status (ON/OFF) and link mode (10/100/Full/Half)

## sis900\_check\_mode

### Name

`sis900_check_mode` — check the media mode for sis900

## Synopsis

```
void sis900_check_mode (struct net_device * net_dev, struct mii_phy *
mii_phy);
```

## Arguments

*net\_dev*

the net device to be checked

*mii\_phy*

the mii phy

## Description

Older driver gets the media mode from mii status output register. Now we set our media capability and auto-negotiate to get the upper bound of speed and duplex between two ends. If the types of mii phy is HOME, it doesn't need to auto-negotiate and `autong_complete` should be set to 1.

## sis900\_set\_mode

### Name

`sis900_set_mode` — Set the media mode of mac register.

### Synopsis

```
void sis900_set_mode (long ioaddr, int speed, int duplex);
```

### Arguments

*ioaddr*

the address of the device

*speed*

the transmit speed to be determined

*duplex*

the duplex mode to be determined

## Description

Set the media mode of mac register `txcfg/rxcfg` according to speed and duplex of phy. Bit `EDB_MASTER_EN` indicates the EDB bus is used instead of PCI bus. When this bit is set 1, the Max DMA Burst Size for TX/RX DMA should be no larger than 16 double words.

## sis900\_auto\_negotiate

### Name

`sis900_auto_negotiate` — Set the Auto-Negotiation Enable/Reset bit.

### Synopsis

```
void sis900_auto_negotiate (struct net_device * net_dev, int phy_addr);
```

### Arguments

*net\_dev*

the net device to read mode for

*phy\_addr*

mii phy address

### Description

If the adapter is link-on, set the auto-negotiate enable/reset bit. `autong_complete` should be set to 0 when starting auto-negotiation. `autong_complete` should be set to 1 if we didn't start auto-negotiation. `sis900_timer` will wait for link on again if `autong_complete = 0`.

## sis900\_read\_mode

### Name

`sis900_read_mode` — read media mode for sis900 internal phy

### Synopsis

```
void sis900_read_mode (struct net_device * net_dev, int * speed, int *  
duplex);
```

### Arguments

*net\_dev*

the net device to read mode for

*speed*

the transmit speed to be determined

*duplex*

the duplex mode to be determined

### Description

The capability of remote end will be put in mii register autorec after auto-negotiation. Use AND operation to get the upper bound of speed and duplex between two ends.

## sis900\_tx\_timeout

### Name

`sis900_tx_timeout` — sis900 transmit timeout routine

## Synopsis

```
void sis900_tx_timeout (struct net_device * net_dev);
```

## Arguments

*net\_dev*

the net device to transmit

## Description

print transmit timeout status disable interrupts and do some tasks

## sis900\_start\_xmit

### Name

`sis900_start_xmit` — sis900 start transmit routine

## Synopsis

```
int sis900_start_xmit (struct sk_buff * skb, struct net_device * net_dev);
```

## Arguments

*skb*

socket buffer pointer to put the data being transmitted

*net\_dev*

the net device to transmit with

## Description

Set the transmit buffer descriptor, and write TxENA to enable transmit state machine. tell upper layer if the buffer is full

## sis900\_interrupt

### Name

`sis900_interrupt` — sis900 interrupt handler

### Synopsis

```
void sis900_interrupt (int irq, void * dev_instance, struct pt_regs * regs);
```

### Arguments

*irq*

the irq number

*dev\_instance*

the client data object

*regs*

snapshot of processor context

### Description

The interrupt handler does all of the Rx thread work, and cleans up after the Tx thread

## sis900\_rx

### Name

sis900\_rx — sis900 receive routine

### Synopsis

```
int sis900_rx (struct net_device * net_dev);
```

### Arguments

*net\_dev*

the net device which receives data

### Description

Process receive interrupt events, put buffer to higher layer and refill buffer pool

### Note

This function is called by interrupt handler, don't do "too much" work here

## sis900\_finish\_xmit

### Name

sis900\_finish\_xmit — finish up transmission of packets

### Synopsis

```
void sis900_finish_xmit (struct net_device * net_dev);
```

## Arguments

*net\_dev*

the net device to be transmitted on

## Description

Check for error condition and free socket buffer etc schedule for more transmission as needed

## Note

This function is called by interrupt handler, don't do "too much" work here

## sis900\_close

### Name

`sis900_close` — close sis900 device

### Synopsis

```
int sis900_close (struct net_device * net_dev);
```

## Arguments

*net\_dev*

the net device to be closed

## Description

Disable interrupts, stop the Tx and Rx Status Machine free Tx and RX socket buffer

## netdev\_ethtool\_ioctl

### Name

`netdev_ethtool_ioctl` — For the basic support of ethtool

### Synopsis

```
int netdev_ethtool_ioctl (struct net_device * net_dev, void * useraddr);
```

### Arguments

*net\_dev*

the net device to command for

*useraddr*

start address of interface request

### Description

Process ethtool command such as “`ethtool -i`” to show information

## mii\_ioctl

### Name

`mii_ioctl` — process MII i/o control command

### Synopsis

```
int mii_ioctl (struct net_device * net_dev, struct ifreq * rq, int cmd);
```

## Arguments

*net\_dev*

the net device to command for

*rq*

parameter for command

*cmd*

the i/o command

## Description

Process MII command like read/write MII register

## sis900\_get\_stats

### Name

`sis900_get_stats` — Get sis900 read/write statistics

### Synopsis

```
struct net_device_stats * sis900_get_stats (struct net_device * net_dev);
```

## Arguments

*net\_dev*

the net device to get statistics for

## Description

get tx/rx statistics for sis900

## sis900\_set\_config

### Name

`sis900_set_config` — Set media type by `net_device.set_config`

### Synopsis

```
int sis900_set_config (struct net_device * dev, struct ifmap * map);
```

### Arguments

*dev*

the net device for media type change

*map*

ifmap passed by ifconfig

### Description

Set media type to 10baseT, 100baseT or 0(for auto) by ifconfig we support only port changes. All other runtime configuration changes will be ignored

## sis900\_mcast\_bitnr

### Name

`sis900_mcast_bitnr` — compute hashtable index

### Synopsis

```
u16 sis900_mcast_bitnr (u8 * addr, u8 revision);
```

## Arguments

*addr*

multicast address

*revision*

revision id of chip

## Description

SiS 900 uses the most significant 7 bits to index a 128 bits multicast hash table, which makes this function a little bit different from other drivers SiS 900 B0 & 635 M/B uses the most significant 8 bits to index 256 bits multicast hash table.

## set\_rx\_mode

### Name

`set_rx_mode` — Set SiS900 receive mode

### Synopsis

```
void set_rx_mode (struct net_device * net_dev);
```

### Arguments

*net\_dev*

the net device to be set

### Description

Set SiS900 receive mode for promiscuous, multicast, or broadcast mode. And set the appropriate multicast filter. Multicast hash table changes from 128 to 256 bits for 635M/B & 900B0.

## sis900\_reset

### Name

`sis900_reset` — Reset sis900 MAC

### Synopsis

```
void sis900_reset (struct net_device * net_dev);
```

### Arguments

*net\_dev*

the net device to reset

### Description

reset sis900 MAC and wait until finished reset through command register change backoff algorithm for 900B0 & 635 M/B

## sis900\_remove

### Name

`sis900_remove` — Remove sis900 device

### Synopsis

```
void __devexit sis900_remove (struct pci_dev * pci_dev);
```

## Arguments

*pci\_dev*

the pci device to be removed

## Description

remove and release SiS900 net device