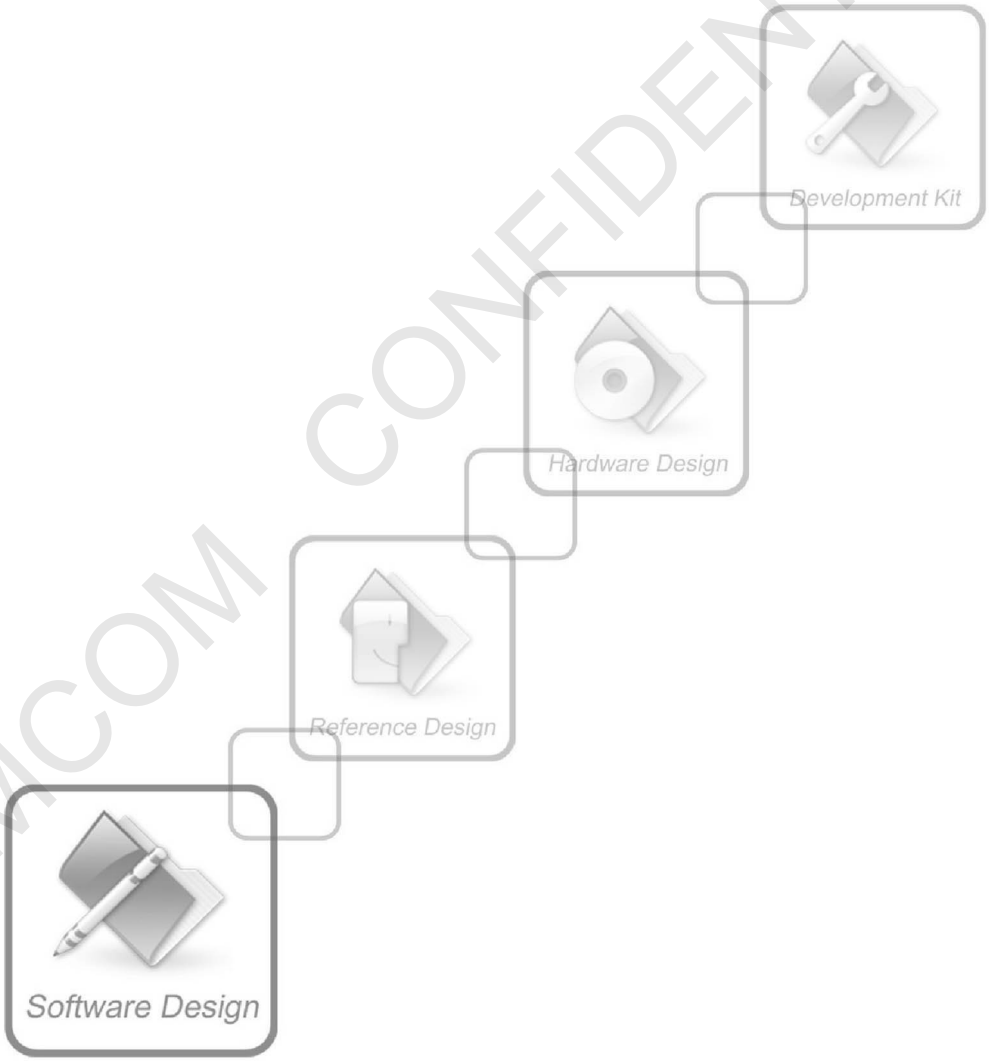
SIM5300E_Linux_Driver_Application Note



Shanghai SIMCom Wireless Solutions Ltd.

Document Title:	SIM5300E_Linux_Driver_Application Note
Version:	1.00
Date:	2016-11-04
Status:	Release
Document Control ID:	SIM5300E_Linux_Driver_Application Note V1.00

General Notes

SIMCom offers this information as a service to its customers, to support application and engineering efforts that use the products designed by SIMCom. The information provided is based upon requirements specifically provided to SIMCom by the customers. SIMCom has not undertaken any independent search for additional relevant information, including any information that may be in the customer's possession. Furthermore, system validation of this product designed by SIMCom within a larger electronic system remains the responsibility of the customer or the customer's system integrator. All specifications supplied herein are subject to change.

Copyright

CONTENTS

1. INTERDUCTION

This document is a brief description on:

How to modify, build and use the driver on Linux issued by Linux kernel in order to use SIMCom 3G modules.

2. DRIVER ISSUED BY LINUX KERNEL

In fact the kernel with version of 2.6.20 and later has a common driver named usbserial which can be used by SIMCom device.

Succeeding sections will use the kernel code of 2.6.35 as an example to depict how to modify, build and use kernel driver for SIMCom device in fail detail.

2.1. MODIFY THE DRIVER

One needs to add the vendor ID and product ID of SIMCom to kernel driver in order to support SIMCom device.

drivers\usb\serial\option.c:.

```
/*added by simcom for SIM5300 - s*/
#define SIM5300_PRODUCT_ID        0x0020
#define SIM5300_VENDOR_ID        0x1E0E
/*added by simcom for SIM5300 - e*/
/* some devices interfaces need special handling due to a number of reasons */
enum option_blacklist_reason {
        OPTION_BLACKLIST_NONE = 0,
        OPTION_BLACKLIST_SENDSETUP = 1,
        OPTION_BLACKLIST_RESERVED_IF = 2
};

struct option_blacklist_info {
    const u32 infolen;   /* number of interface numbers on blacklist */
    const u8  *ifaceinfo;   /* pointer to the array holding the numbers */
    enum option_blacklist_reason reason;
};

static const u8 four_g_w14_no_sendsetup[] = { 0, 1 };
static const struct option_blacklist_info four_g_w14_blacklist = {
    .infolen = ARRAY_SIZE(four_g_w14_no_sendsetup),
    .ifaceinfo = four_g_w14_no_sendsetup,
    .reason = OPTION_BLACKLIST_SENDSETUP
};

static const u8 alcatel_x200_no_sendsetup[] = { 0, 1 };
static const struct option_blacklist_info alcatel_x200_blacklist = {
    .infolen = ARRAY_SIZE(alcatel_x200_no_sendsetup),
    .ifaceinfo = alcatel_x200_no_sendsetup,
    .reason = OPTION_BLACKLIST_SENDSETUP
};

static const u8 zte_k3765_z_no_sendsetup[] = { 0, 1, 2 };
static const struct option_blacklist_info zte_k3765_z_blacklist = {
    .infolen = ARRAY_SIZE(zte_k3765_z_no_sendsetup),
    .ifaceinfo = zte_k3765_z_no_sendsetup,
    .reason = OPTION_BLACKLIST_SENDSETUP
};

static const struct usb_device_id option_ids[] = {
    { USB_DEVICE(SIM5300_VENDOR_ID, SIM5300_PRODUCT_ID) },/*added by simcom for SIM5300*/
    { USB_DEVICE(OPTION_VENDOR_ID, OPTION_PRODUCT_COLT) },
    { USB_DEVICE(OPTION_VENDOR_ID, OPTION_PRODUCT_RICOLA) },
    { USB_DEVICE(OPTION_VENDOR_ID, OPTION_PRODUCT_RICOLA_LIGHT) },
    { USB_DEVICE(OPTION_VENDOR_ID, OPTION_PRODUCT_RICOLA_QUAD) },
```

2.1.1. Support system suspend/resume

Add .reset_resume call-back function if kernel support, for some USB HOST controller issue a bus reset to USB devices when system resume, USB port will be unloaded, and loaded later, the reset_resume call-back function will avoid the port unloading when system resume, for more detail please refer to kernel USB driver documents.

```
static struct usb_driver option_driver = {
    .name       = "option",
    .probe      = usb_serial_probe,
    .disconnect = usb_serial_disconnect,
#ifdef CONFIG_PM
    .suspend    = usb_serial_suspend,
    .resume     = usb_serial_resume,
    .supports_autosuspend = 1,
#endif
    .id_table   = option_ids,
    .no_dynamic_id =    1,
};
```

```
static struct usb_driver option_driver = {
    .name       = "option",
    .probe      = usb_serial_probe,
    .disconnect = usb_serial_disconnect,
#ifdef CONFIG_PM
    .suspend    = usb_serial_suspend,
    .resume     = usb_serial_resume,
    .reset_resume =     usb_serial_resume,
    .supports_autosuspend = 1,
#endif
    .id_table   = option_ids,
    .no_dynamic_id =    1,
};
```

2.1.2. Support low power mode

For kernel 2.6.36, add the follow highlight code to end of option_probe function:

```
1076
1077    if (serial->dev->descriptor.idVendor == SIMCOM_WCDMA_VENDOR_ID &&
1078        serial->dev->descriptor.idProduct == SIMCOM_WCDMA_PRODUCT_ID)
1079    {
1080  #ifdef CONFIG_PM
1081      serial->interface->needs_remote_wakeup = 1        /* autosuspend (15s delay) */
1082      device_init_wakeup(&serial->interface->dev, 1);
1083      serial->dev->autosuspend_delay = 15 * HZ;   /* for kernel 2.6.36 */
1084      usb_enable_autosuspend(serial->dev);
1085  #endif /* CONFIG_PM */
1086    }
1087
```

For kernel 2.6.38, add the follow highlight code to end of option_probe function:

```
1076
1077    if (serial->dev->descriptor.idVendor == SIMCOM_WCDMA_VENDOR_ID &&
1078        serial->dev->descriptor.idProduct == SIMCOM_WCDMA_PRODUCT_ID)
1079    {
1080  #ifdef CONFIG_PM
1081      pm_runtime_set_autosuspend_delay(&serial->dev, 12 * 1000);    /*  for kernel 2.6.38 and above */
1082      usb_enable_autosuspend(serial->dev);
1083  #endif /* CONFIG_PM */
1084    }
1085
```

2.1.3. Add short packet flag

Since the max packet size of BULK endpoint on SIMCOM module in High USB speed is 512 bytes, in Full USB speed is 64 bytes, in addition the USB protocol says:

> An endpoint must always transmit data payloads with a data field less than or equal to the endpoint's reported *wMaxPacketSize* value. When a bulk IRP involves more data than can fit in one maximum-sized data payload, all data payloads are required to be maximum size except for the last data payload, which will contain the remaining data. A bulk transfer is complete when the endpoint does one of the following:
>
> - Has transferred exactly the amount of data expected
> - Transfers a packet with a payload size less than *wMaxPacketSize* or transfers a zero-length packet
>
> When a bulk transfer is complete, the Host Controller retires the current IRP and advances to the next IRP. If a data payload is received that is larger than expected, all pending bulk IRPs for that endpoint will be aborted/retired.

So one needs to send an zero-length packet additional if one wants to transmit the data stream with length exactly multiple of wMaxPacketSize.

Fortunately one needs not to send a zero packet manually; one only needs to modify a little driver code:

drivers\usb\serial\usb_wwan.c:

```
/* Setup urbs */
static void usb_wwan_setup_urbs(struct usb_serial *serial)
{
        int i, j;
        struct usb_serial_port *port;
        struct usb_wwan_port_private *portdata;

        dbg("%s", __func__);

        for (i = 0; i < serial->num_ports; i++) {
                port = serial->port[i];
                portdata = usb_get_serial_port_data(port);

                /* Do indat endpoints first */
                for (j = 0; j < N_IN_URB; ++j) {
                        portdata->in_urbs[j] = usb_wwan_setup_urb(serial,
                                                port->
                                                bulk_in_endpointAddress,
                                                USB_DIR_IN,
                                                port,
                                                portdata->
                                                in_buffer[j],
                                                IN_BUFLEN,
                                                usb_wwan_indat_callback);
                }

                /* outdat endpoints */
                for (j = 0; j < N_OUT_URB; ++j) {
                        portdata->out_urbs[j] = usb_wwan_setup_urb(serial,
                                                port->
                                                bulk_out_endpointAddress,
                                                USB_DIR_OUT,
                                                port,
                                                portdata->
                                                out_buffer
                                                [j],
                                                OUT_BUFLEN,
                                                usb_wwan_outdat_callback);

                        portdata->out_urbs[j]->transfer_flags |= URB_ZERO_PACKET;  //add by simcom
                }
        } ?  end for i=0;i<serial->num_por... ?
} ?  end usb_wwan_setup_urbs ?
```

NOTE: This modification is only for the driver option.ko

2.2. BUILD THE DRIVER

One needs to setup the kernel development environment first which include kernel source code and cross compiler environment.

Following is a step-by-step instruction on how to build the driver into kernel.

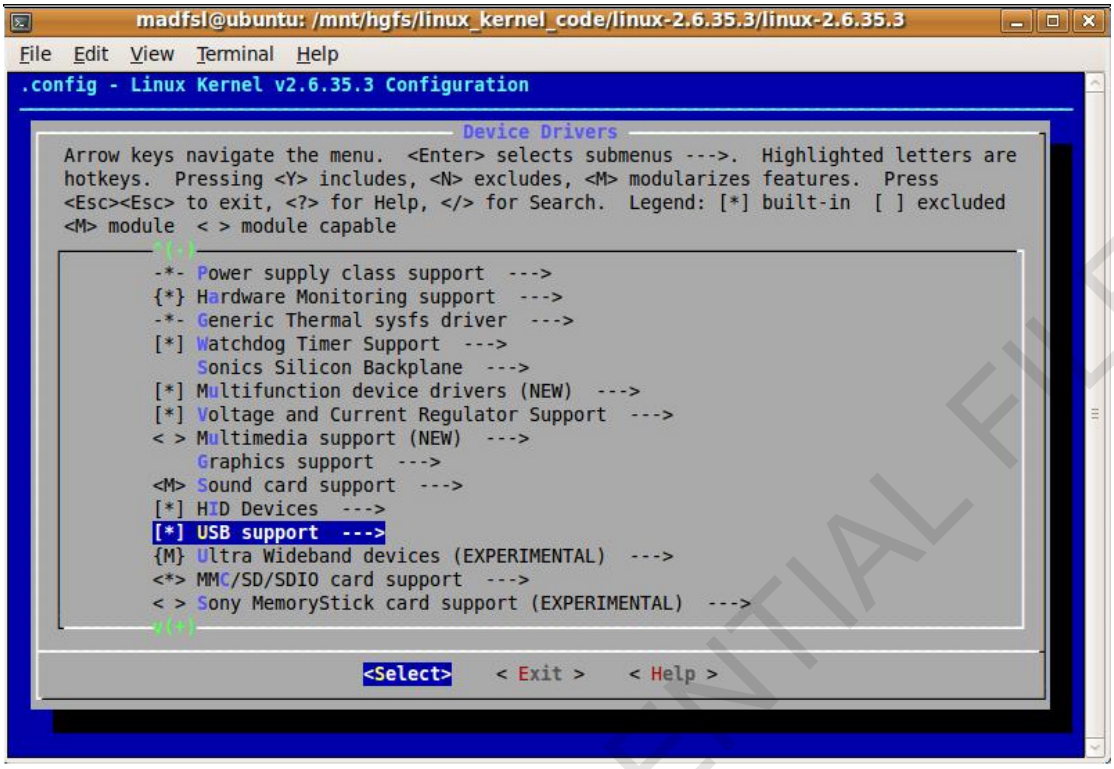
1) Use “sudo make menuconfig” to configure the kernel.

```
madfsl@ubuntu:/mnt/hgfs/linux_kernel_code/linux-2.6.35.3/linux-2.6.35.3$ ls
arch      crypto          fs        Kbuild       Makefile  REPORTING-BUGS  sound
block     Documentation   include   kernel       mm        samples         tools
COPYING   drivers         init      lib          net       scripts         usr
CREDITS   firmware        ipc       MAINTAINERS  README    security        virt
madfsl@ubuntu:/mnt/hgfs/linux_kernel_code/linux-2.6.35.3/linux-2.6.35.3$
madfsl@ubuntu:/mnt/hgfs/linux_kernel_code/linux-2.6.35.3/linux-2.6.35.3$
madfsl@ubuntu:/mnt/hgfs/linux_kernel_code/linux-2.6.35.3/linux-2.6.35.3$
madfsl@ubuntu:/mnt/hgfs/linux_kernel_code/linux-2.6.35.3/linux-2.6.35.3$ sudo make menuconfig
```

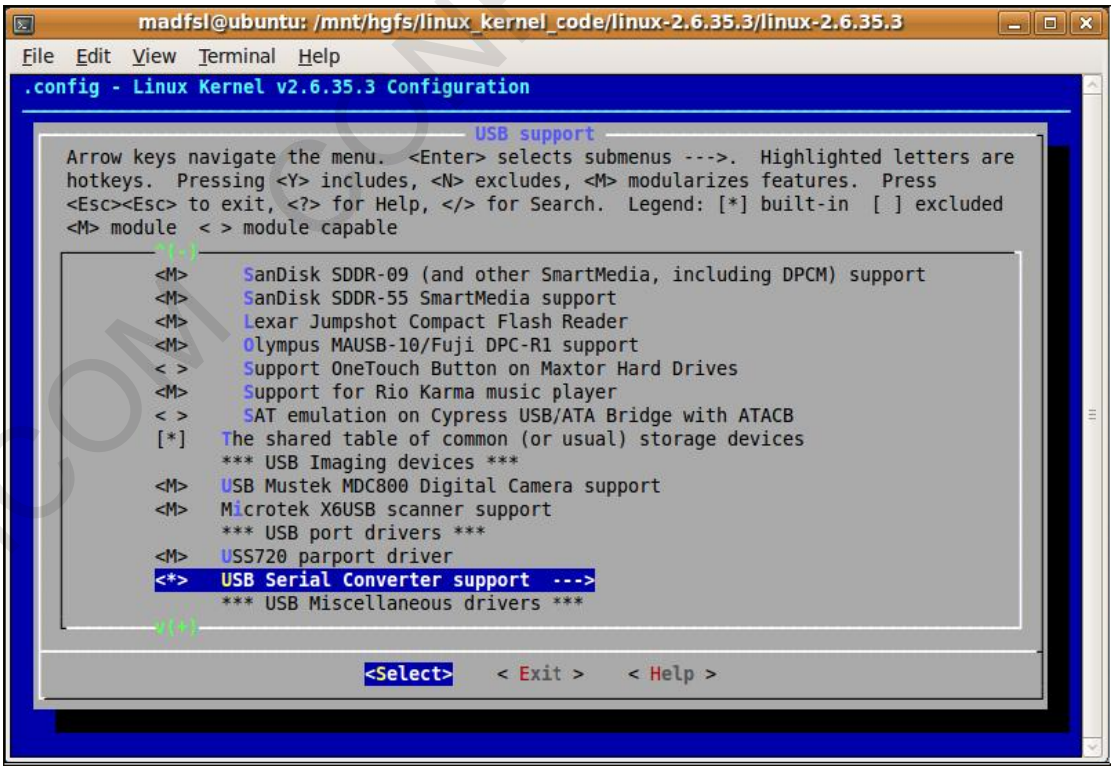
2) Enter into menu “Device Drivers”

```
madfsl@ubuntu: /mnt/hgfs/linux_kernel_code/linux-2.6.35.3/linux-2.6.35.3
File  Edit  View  Terminal  Help
.config - Linux Kernel v2.6.35.3 Configuration

                    Linux Kernel Configuration
  Arrow keys navigate the menu.  <Enter> selects submenus --->.  Highlighted letters are
  hotkeys.  Pressing <Y> includes, <N> excludes, <M> modularizes features.  Press
  <Esc><Esc> to exit, <?> for Help, </> for Search.  Legend: [*] built-in  [ ] excluded
  <M> module  < > module capable

        General setup  --->
    [*] Enable loadable module support  --->
    -*- Enable the block layer  --->
        Processor type and features  --->
        Power management and ACPI options  --->
        Bus options (PCI etc.)  --->
        Executable file formats / Emulations  --->
    -*- Networking support  --->
        Device Drivers  --->
        Firmware Drivers  --->
        File systems  --->
        Kernel hacking  --->
        Security options  --->
    -*- Cryptographic API  --->
    [*] Virtualization  --->

                <Select>    < Exit >    < Help >
```

3) Continue enter into menu “USB support”



4) Continue enter into menu “USB Serial Converter support”



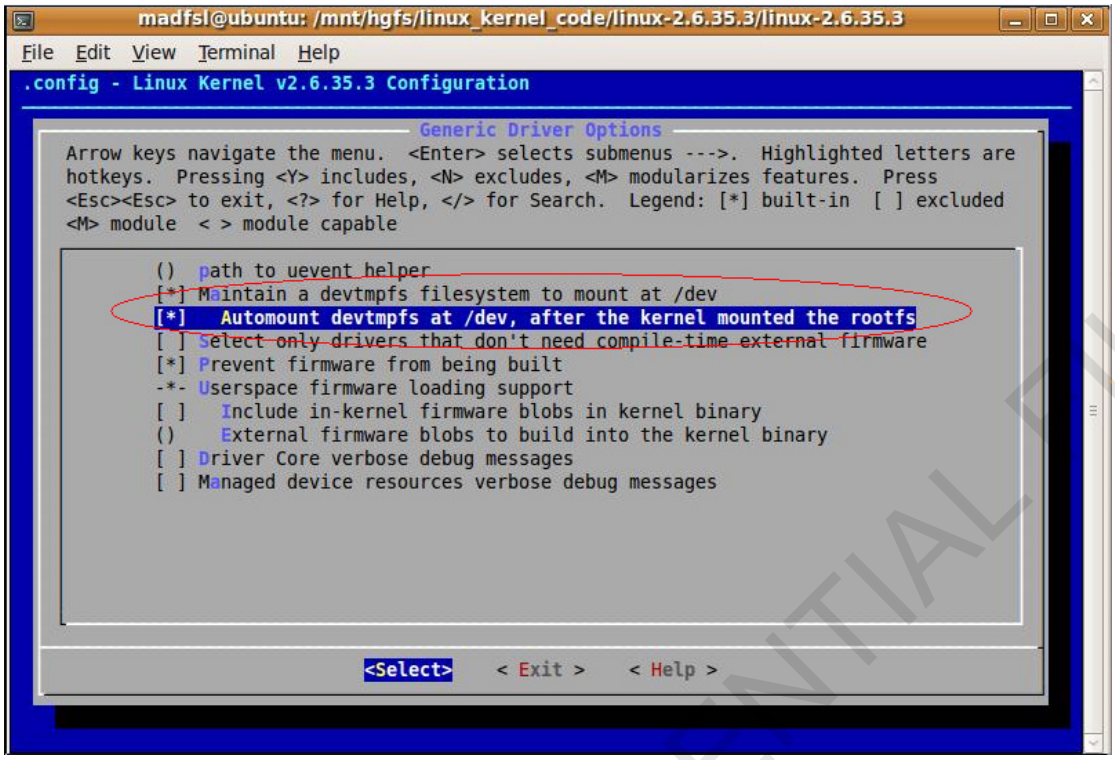
5) Type “y” to select menu “USB driver for GSM and CDMA modems”, of course one can type “m” to compile the driver as a module.

```
madfsl@ubuntu: /mnt/hgfs/linux_kernel_code/linux-2.6.35.3/linux-2.6.35.3
File  Edit  View  Terminal  Help
.config - Linux Kernel v2.6.35.3 Configuration

                    USB Serial Converter support
Arrow keys navigate the menu.  <Enter> selects submenus --->.  Highlighted letters are
hotkeys.  Pressing <Y> includes, <N> excludes, <M> modularizes features.  Press
<Esc><Esc> to exit, <?> for Help, </> for Search.  Legend: [*] built-in  [ ] excluded
<M> module  < > module capable
        < >   USB Qualcomm Auxiliary Serial Port Driver (NEW)
        < >   USB Qualcomm Serial modem (NEW)
        <M>   USB SPCP8x5 USB To Serial Driver
        <M>   USB HP4x Calculators support
        <M>   USB Safe Serial (Encapsulated) Driver
        [ ]     USB Secure Encapsulated Driver - Padded
        < >   USB Siemens MPI driver (NEW)
        <M>   USB Sierra Wireless Driver
        < >   USB Symbol Barcode driver (serial mode) (NEW)
        <M>   USB TI 3410/5052 Serial Driver
        <M>   USB REINER SCT cyberJack pinpad/e-com chipcard reader
        <M>   USB Xircom / Entregra Single Port Serial Driver
        <*>   USB driver for GSM and CDMA modems
        <M>   USB ZyXEL omni.net LCD Plus Driver
        < >   USB Opticon Barcode driver (serial mode) (NEW)

                    <Select>    < Exit >    < Help >
```

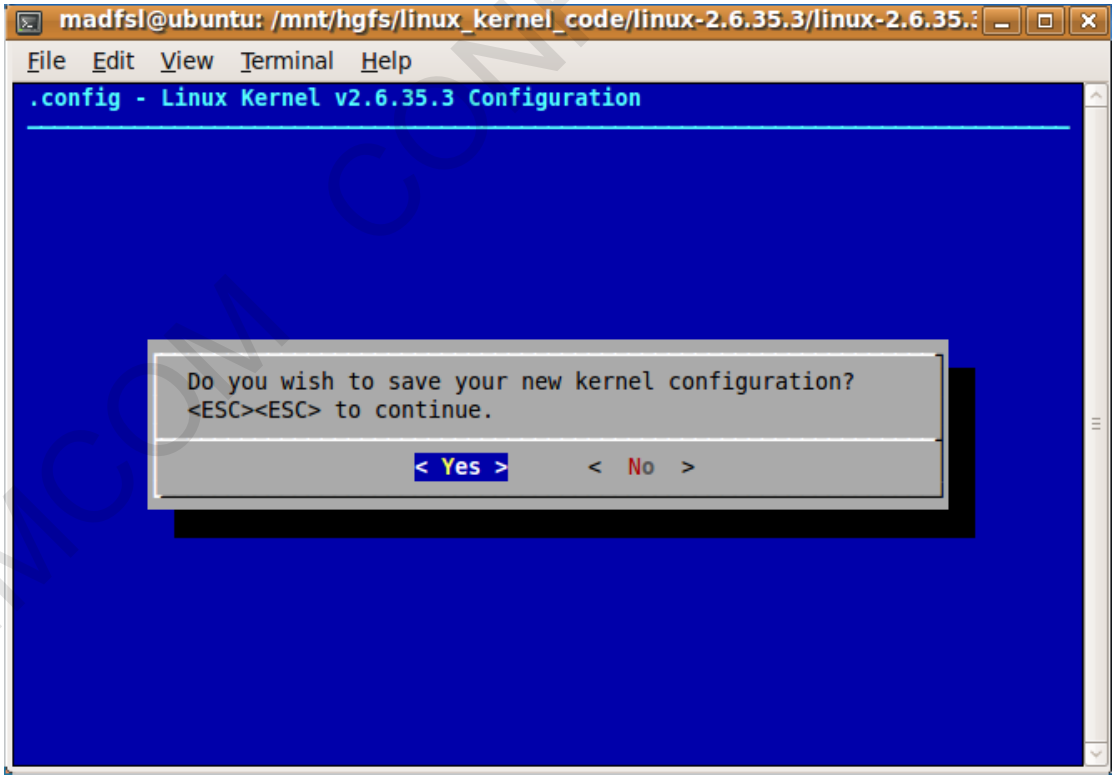
6) Some other options need to be configured, so please enter into menu “Device Drivers -> Generic Driver Options”

```
madfsl@ubuntu: /mnt/hgfs/linux_kernel_code/linux-2.6.35.3/linux-2.6.35.3
File  Edit  View  Terminal  Help
.config - Linux Kernel v2.6.35.3 Configuration

                    Device Drivers
Arrow keys navigate the menu.  <Enter> selects submenus --->.  Highlighted letters are
hotkeys.  Pressing <Y> includes, <N> excludes, <M> modularizes features.  Press
<Esc><Esc> to exit, <?> for Help, </> for Search.  Legend: [*] built-in  [ ] excluded
<M> module  < > module capable
            Generic Driver Options  --->
        <*> Connector - unified userspace <-> kernelspace linker  --->
        <M> Memory Technology Device (MTD) support  --->
        <M> Parallel port support  --->
        -*- Plug and Play support  --->
        [*] Block devices  --->
        [*] Misc devices  --->
        < > ATA/ATAPI/MFM/RLL support (DEPRECATED)  --->
            SCSI device support  --->
        <*> Serial ATA and Parallel ATA drivers  --->
        [*] Multiple devices driver support (RAID and LVM)  --->
        [*] Fusion MPT device support  --->
            IEEE 1394 (FireWire) support  --->
        <M> I2O device support  --->
        [*] Macintosh device drivers  --->

                    <Select>    < Exit >    < Help >
```

7) Type “y” to select the following two options.



8) Exit and save the configuration.



After configuration, these items will be configured:

CONFIG_USB = y

CONFIG_USB_SERIAL=y

CONFIG_USB_SERIAL_OPTION=y

CONFIG_DEVTMPFS=y

CONFIG_DEVTMPFS_MOUNT=y

1) Use “sudo make” to compile the kernel or use “sudo make modules” to compile the driver as a module

```
madfsl@ubuntu:/mnt/hgfs/linux_kernel_code/linux-2.6.35.3/linux-2.6.35.3$ sudo ma
ke
[sudo] password for madfsl:
  HOSTLD  scripts/kconfig/conf
scripts/kconfig/conf -s arch/x86/Kconfig
```

2.3. USE THE DRIVER

As you move through this chapter new kernel firmware or new driver: option.ko(compiled as module) is ready.

2.3.1. Install the driver (driver as module only)

If one compiles the driver as a module one needs to install it first.

One can use the following command to install the driver:

modprobe option.ko

This command will install all the needed drivers.

```
USB-Serial-COM4
root@freescale /lib/modules/2.6.35.3-571-gcca29a0/kernel/drivers/usb/serial$ ls
option.ko     usb_wwan.ko   usbserial.ko
root@freescale /lib/modules/2.6.35.3-571-gcca29a0/kernel/drivers/usb/serial$ mod
probe option.ko
usbcore: registered new interface driver usbserial
usbserial: USB Serial Driver core
USB Serial support registered for GSM modem (1-port)
usbcore: registered new interface driver option
option: v0.7.2:USB Driver for GSM modems
root@freescale /lib/modules/2.6.35.3-571-gcca29a0/kernel/drivers/usb/serial$
```

If all right the driver will be installed to the system, one can use the following command to query the result:

```
lsmod |grep option
```

```
root@freescale /lib/modules/2.6.35.3-571-gcca29a0/kernel/drivers/usb/serial$ lsm
od |grep option
option                  12548  0
usb_wwan                 7381  1 option
usbserial               23430  2 option,usb_wwan
root@freescale /lib/modules/2.6.35.3-571-gcca29a0/kernel/drivers/usb/serial$
```

Note: this installation procedure is invalid when rebooting the system, so if one wants to install the driver automatically when starting the system, one should better put the installation instruction to the startup script.

2.3.2. Use the driver

After the driver installed one can use SIMCom device via the driver, now plug the SIMCom device to the host device via USB connector, and if the device is identified by the driver, 5 device files named ttyUSB0, ttyUSB1, ttyUSB2, ttyUSB3, ttyUSB4, ttyUSB5, ttyUSB6, ttyUSB7, ttyUSB8, ttyUSB9, ttyUSB10, ttyUSB11, ttyUSB12, ttyUSB13 and ttyUSB14 will be created in directory /dev

The relationship between the device files and SIMCom composite device is like this:

Device file	SIMCom composite device
ttyUSB7	ATCOM interface
ttyUSB1	MODEM interface

SIMCom device is plugged in:

```
[  105.186704] option 1-1.2:1.0: GSM modem (1-port) converter detected
[  105.196327] usb 1-1.2: GSM modem (1-port) converter now attached to ttyUSB0
[  105.201505] option 1-1.2:1.1: GSM modem (1-port) converter detected
[  105.210942] usb 1-1.2: GSM modem (1-port) converter now attached to ttyUSB1
[  105.216292] option 1-1.2:1.2: GSM modem (1-port) converter detected
[  105.226091] usb 1-1.2: GSM modem (1-port) converter now attached to ttyUSB2
[  105.231275] option 1-1.2:1.3: GSM modem (1-port) converter detected
[  105.238526] usb 1-1.2: GSM modem (1-port) converter now attached to ttyUSB3
[  105.243725] option 1-1.2:1.4: GSM modem (1-port) converter detected
[  105.253580] usb 1-1.2: GSM modem (1-port) converter now attached to ttyUSB4
[  105.260730] option 1-1.2:1.5: GSM modem (1-port) converter detected
[  105.270951] usb 1-1.2: GSM modem (1-port) converter now attached to ttyUSB5
[  105.276290] option 1-1.2:1.6: GSM modem (1-port) converter detected
[  105.286737] usb 1-1.2: GSM modem (1-port) converter now attached to ttyUSB6
[  105.291957] option 1-1.2:1.7: GSM modem (1-port) converter detected
[  105.301373] usb 1-1.2: GSM modem (1-port) converter now attached to ttyUSB7
[  105.306620] option 1-1.2:1.8: GSM modem (1-port) converter detected
[  105.317543] usb 1-1.2: GSM modem (1-port) converter now attached to ttyUSB8
[  105.322769] option 1-1.2:1.9: GSM modem (1-port) converter detected
[  105.332154] usb 1-1.2: GSM modem (1-port) converter now attached to ttyUSB9
[  105.337390] option 1-1.2:1.10: GSM modem (1-port) converter detected
[  105.344091] usb 1-1.2: GSM modem (1-port) converter now attached to ttyUSB10
[  105.352984] option 1-1.2:1.11: GSM modem (1-port) converter detected
[  105.363232] usb 1-1.2: GSM modem (1-port) converter now attached to ttyUSB11
[  105.368661] option 1-1.2:1.12: GSM modem (1-port) converter detected
[  105.379927] usb 1-1.2: GSM modem (1-port) converter now attached to ttyUSB12
[  105.383737] option 1-1.2:1.13: GSM modem (1-port) converter detected
[  105.393550] usb 1-1.2: GSM modem (1-port) converter now attached to ttyUSB13
[  105.399945] option 1-1.2:1.14: GSM modem (1-port) converter detected
[  105.422185] usb 1-1.2: GSM modem (1-port) converter now attached to ttyUSB14
```

Device files are created:

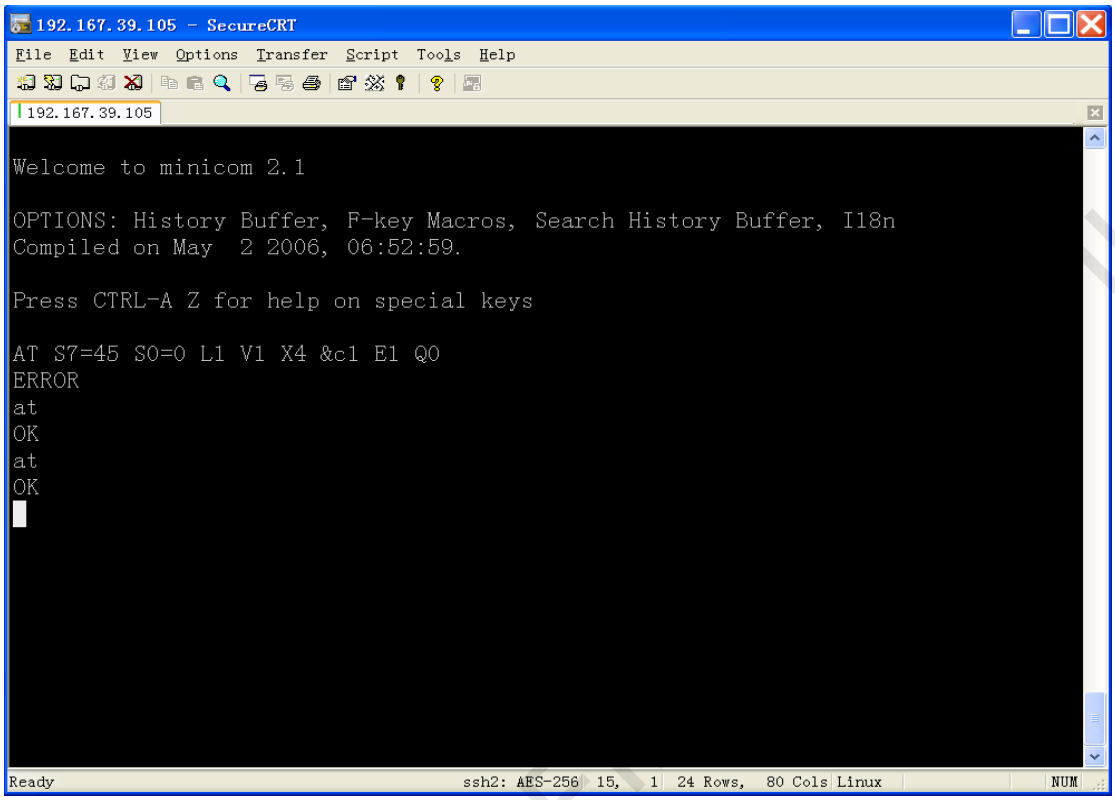
```
[root@FriendlyARM /]# ls /dev | grep USB
ttyUSB0
ttyUSB1
ttyUSB10
ttyUSB11
ttyUSB12
ttyUSB13
ttyUSB14
ttyUSB2
ttyUSB3
ttyUSB4
ttyUSB5
ttyUSB6
ttyUSB7
ttyUSB8
ttyUSB9
[root@FriendlyARM /]#
```

NOTE:

1. *In some composite devices of SIMCom not all of the interfaces are existed, so the relationship is dynamic.*

2. *Only the ATCOM and MODEM interface can be worked correctly with this driver.*

If one gets the device files ready one can use tools such as minicom, wvdial etc to use the device。



ATCOM interface

2.3.3. Remove the driver

One can use the following command to uninstall the driver:

rmmod option

```
root@freescale /lib/modules/2.6.35.3-571-gcca29a0/kernel/drivers/usb/serial$ rmm
od option.ko
usbcore: deregistering interface driver option
option: option_instat_callback: error -108
option1 ttyUSB4: GSM modem (1-port) converter now disconnected from ttyUSB4
option 2-1:1.4: device disconnected
option: option_instat_callback: error -108
option1 ttyUSB3: GSM modem (1-port) converter now disconnected from ttyUSB3
option 2-1:1.3: device disconnected
option1 ttyUSB2: GSM modem (1-port) converter now disconnected from ttyUSB2
option 2-1:1.2: device disconnected
option1 ttyUSB1: GSM modem (1-port) converter now disconnected from ttyUSB1
option 2-1:1.1: device disconnected
option1 ttyUSB0: GSM modem (1-port) converter now disconnected from ttyUSB0
option 2-1:1.0: device disconnected
USB Serial deregistering driver GSM modem (1-port)
root@freescale /lib/modules/2.6.35.3-571-gcca29a0/kernel/drivers/usb/serial$
```

After the command executed one can use “lsmod |grep option” to check if the driver has been removed successfully.

Note: when removing the driver one must disconnect the device and close all the tools using the device first.

CONTACT US:

Shanghai SIMCom Wireless Solutions Ltd.
Add: Bldg A, SIM Technology Bldg., No.633, Jinzhong Road, Changning Dist., Shanghai P.R. China 200335
Tel: +86 21 32523424
Fax: +86 21 32523020
URL: www.simcomm2m.com