Gunzets 1.4

Support software for Gunze touch screens
March 2002

by **Alessandro Rubini** (`rubini@linux.it`)

1 General Information

The package supports Gunze touch panels under Linux and XFree86. Supported panels are the serial, PS/2, and USB devices. The serial and PS/2 devices are supported by means of an X module, xf86Gunze. Support for USB comes as a kernel module that emulates the PS/2 protocol, so you still need the xf86Gunze module.

The latest version of this driver can be downloaded from:

`ftp://ar.linux.it/pub/gunzets`

`http://ar.linux.it/pub/gunzets`

The latest source tree can be accessed from my CVS server, instructions are available at `http://ar.linux.it/software/#cvs` . The package name to use is `gunzets`. The latest CVS snapshot can also be downloaded from `ftp://ar.linux.it/pub/gunzets` .

2 Bugs

The following bugs are currently known:

- Debouncing is only supported for XFree 4.0 (not 3.3.6).
- A bounce event breaks smoothing processing
- Support for multi-screen operation is not supported.

3 Device Support

To run the serial or PS/2 devices you only need the information in this section. To run the USB device you also need to load the kernel module, described under See Chapter 5 [Usb Driver], page 7, later in this file.

Support for all three device flavors is implemented by means of '`xf86Gunze.c`', a module for XFree86.

The driver has been used under XFree86 version 3.3.6 and version 4.0.1.

In order to use the touch screen in X, you should install in your system the "xf86Gunze" module, part of this distribution.

The compiled module ('`xf86Gunze.so`' for XFree86 3.3 or '`gunze_drv.o`' for XFree86 4.0) should go in the module directory of your X server, usually '`/usr/X11R6/lib/modules`' for XFree86 3.3 or '`/usr/X11R6/lib/modules/input`' for XFree86 4.0. When the file is in place, a proper '`XF86Config`' will arrange for its loading.

In order to compile the module for different version of the XFree86 systems you need access the source tree and compile using there commands:

```
make XFREE33SOURCE=<location-of-source> xf86Gunze.so
make XFREE40SOURCE=<location-of-source> gunze_drv.o
```

The "location of source" above should be the full path name to the directory called "xc" after you uncompress the source tar file.

Both `XFREE33SOURCE` and `XFREE40SOURCE` can be defined in your environment. Also, calling `make` without arguments will compile both modules.

To compile *gunzets* you'll most likely need to compile X first (by issuing `make World` and waiting a huge lot of time). A freshly uncompressed source tree lacks the proper header files for compilation to succeed; since one of the errors happens inside an X header, you can only fix it by compiling X first).

4 XFree Configuration

4.1 XFree 3.3

In order to use a Gunze device with XFree 3.3 you need to add the following two sections to your '`XF86Config`' (the example below shows all the options available for the module, but most of them are optional, see below for details):

```
Section "module"
Load "xf86Gunze.so"
EndSection

Section "Xinput"
Subsection "GunzeTS"
DeviceName "touchscreen"
Port            "/dev/ttyS0"
DeviceType      Serial
Speed           9600
CalibrationFile "/etc/gunzets.calib"
Smoothness      9
TappingDelay    0
DebugLevel      0
#Res12Bit
AlwaysCore
EndSubSection
EndSection
```

The meaning of the individual options is the following:

`DeviceName "touchscreen"`

The device name is an identifier string chosen by the user. It is mandatory.

`Port "/dev/ttyS0"`

The hardware channel used to read input data. The line is mandatory. It can be a serial port, a PS/2 port ('`/dev/psaux`') or the USB entry point ('`/dev/gunzets`').

`DeviceType Serial`

Optional specification of the device type. It is used to select the protocol used by the port. Possible values are (case-insensitive): "serial", "ps2" (or "ps/2") and "usb". If the information is not present in the configuration file, the file name is used to guess the device type. Such guess will work under Linux but may fail under other operating systems.

`Speed 9600`
: Speed of the serial connection. Only used for serial devices, the value defaults to 9600. Only valid values are 9600 and 19200.

`CalibrationFile "/etc/gunzets.calib"`
: Optional specification of the calibration file. The default is '`/etc/gunzets.calib`' as shown. The format of the calibration file is specified later in this documents.

`Smoothness 9`
: The smoothness value is used to smooth out movements. It is only meaningful for motion and drag events. Allowed values are in the range 0 to 50. If you want to do handwriting, you'll need to disable smoothness processing (i.e., set it to 0).

`TappingDelay 0`
: Optional selection of tapping mode. By default (tapping delay set to 0), any touch event is reported as a button press. In tapping mode the first touch event is used to move the pointer, and button press is only reported when the user taps on the device. If the delay, measured in milliseconds, between pen-up and pen-down is greater than the selected value, the pen-down even is considered motion. I personally prefer tapping mode when testing the device, as it allows me to use the common point-and-click semantics instead of click-only; this is especially true if your touchscreen is not placed over the image but in side of it (as you guess, I do that).

`DebugLevel 0`
: Any debug level greater than 0 tells the driver to spit informative messages. The greater the number, the higher the message rate. Messages go to the usual X error channel (stderr, normally connected to the text console used to fire X).

`Res12Bit`
: The boolean option states whether your panel reports data with 12 bits of resolution (i.e., as numbers in the 0-4095 range). The default, when no option is specified, is 10 bits (0-1023). 12-bit support is implemented by discarding the lowest two bits, and has been contributed by Atsushi Nemoto.

`AlwaysCore`
: The input device is a "core" input device (like the main pointer device – the mouse or equivalent). If the directive is not specified, the touch screen acts as an auxiliary input device, and applications may individually request to use it.

4.2 XFree 4.0

In order to use a Gunze device with XFree 4.0 you need to add an `InputDevice` section to your '`XF86Config`'. The following example shows how it looks like. The options that are shown commented are not implemented in this version of the module, and are ignored if specified.

```
Section "InputDevice"
Identifier "Touchscreen0"
```

```
    Driver "gunze"
    Option "Device"             "/dev/ttyS0"
    #Option "DeviceType"        "serial"
    Option "BaudRate"           "9600"
    #Option "CalibrationFile"   "/etc/gunzets.calib"
    Option "Smoothness"         "9"
    Option "TappingDelay"       "0"
    Option "JitterDelay"        "50"
    Option "DebugLevel"         "0"
    Option "Res12Bit"           "False"
    Option "SendCoreEvents"

    EndSection
```

Moreover, you need to add an `InputDevice` line in the `ServerLayout` section. After the addition, the section will most likely look like this:

```
Section "ServerLayout"
Identifier  "Simple Layout"
Screen      "Screen 1"
InputDevice "Mouse1" "CorePointer"
InputDevice "Keyboard1" "CoreKeyboard"
InputDevice "Touchscreen0"
EndSection
```

Note that if your system has no mouse device, you can remove the `InputDevice "Mouse1"` line and add `"CorePointer"` to the touch screen line.

The meaning of individual options is as follows:

`Identifier "Touchscreen0"`
: The identifier string is mandatory, and appears, literally, in the `InputDevice` directive within the `ServerLayout` section that is usually found at the end of the configuration file.

`Driver "gunze"`
: The name of the driver is mandatory. It is used to load the associated module. In this case the file '`gunze_drv.o`'.

`Option "Device" "/dev/ttyS0"`
: The device name is mandatory. It states where input data is collected from. It can be a serial port, a ps2 port or the USB device created by the `gunzets` device driver.

`Option "DeviceType" "serial"`
: The device type specification is not currently supported. If the option is present it is ignored. The actual device type is retrieved from the device name: any device name that includes the string "tty" is assumed to be a serial port, other devices are assumed to speak the PS/2 protocol.

`Option "BaudRate" "9600"`
: The transmission rate for serial ports. It defaults to 9600 and is ignored for non-serial input devices.

`Option "CalibrationFile" "/etc/gunzets.calib"`
: The option is not currently implemented.

`Option "Smoothness" "9"`
: The smoothness of the pointer. The default value is 9. The greater the value the smoother the motion. If you need to do handwriting or similar "fast" interaction, you should set the smoothness to 0.

`Option "TappingDelay" "0"`
: Optional selection of tapping mode. By default (tapping delay set to 0), any touch event is reported as a button press. In tapping mode the first touch event is used to move the pointer, and button press is only reported when the user taps on the device. If the delay, measured in milliseconds, between pen-up and pen-down is greater than the selected value, the pen-down even is considered motion. I personally prefer tapping mode when testing the device, as it allows me to use the common point-and-click semantics instead of click-only; this is especially true if your touchscreen is not placed over the image but in side of it (as you guess, I do that).

`Option "JitterDelay" "50"`
: The debouncing time lapse, measured in milliseconds. If a pen-down event occurs immediately after pen-up (within this time lapse), then both events are discarded. Debouncing has been contributed by Chris Howe <chris@howeville.com>, and defaults to 50ms.

`Option "DebugLevel" "0"`
: The level of messages spit out by the driver. The directive is optional and it defaults to 0.

`Option "Res12Bit" "False"`
: The boolean option states whether your panel reports data with 12 bits of resolution (i.e., as numbers in the 0-4095 range). The default is 10 bits (0-1023). 12-bit support is implemented by discarding the lowest two bits, and has been contributed by Atsushi Nemoto.

`Option "SendCoreEvents"`
: The directive instructs X to use the touch screen as a core input device (like the main mouse). This module can currently only work as a core device.

4.3 Generic X Configuration

Please note that with both version 3.3 and 4.0 of XFree you can still use a normal mouse together with the touch panel. While XFree 4.0 can use the touch screen as its only pointer device, version 3.3 of the X server can't do that. If you have no other pointer and don't want X to choke on the configuration file, you can use a trick like this one:

```
Section "Pointer"
Protocol    "MouseSystems"
Device      "/dev/fifo"
EndSection
```

Where '`/dev/fifo`' is just a plain unused fifo ("mknod /dev/fifo p")

If, when calling `startx`, it fails with a message of `Invalid Subsection Name`, please check the previous error lines, as they explain what is wrong. They usually look like:

```
(--) no ModulePath specified using default: /usr/X11R6/lib/modules
xf86Gunze.so: Unknown error loading module
```

The messages are pretty clear, if your are careful in reading them (I am not that careful, and lose half an hour in trying to figure out what was wrong).

4.4 Text Console

Use of the touch screen on the text console is not supported, and I don't think it ever will (although I made it work under gpm some time ago, I no longer maintain gpm).

4.5 Calibration

To calibrate the touch panel, run the '`gunzets_calib`' script, under X. It has been tested with both XFree86 version 3.3.6 and version 4.0. To calibrate pass the device name on the command line of the program:

```
gunzets_calib /dev/ttyS0 (serial device)
gunzets_calib /dev/psaux (PS/2 device)
gunzets_calib /dev/gunzets (USB device)
```

The program needs to find gunzets_control in the command search path (it uses the control program to turn off the touchscreen in X, so the calibrator can read input data).

In order to calibrate you'll need to move the mouse pointer first. You can do that with the '`warp`' command (see below) if no other pointer is available on the system. While I could use the touch panel to this aim, and delay turning it off, I didn't think of it and don't want to re-test it all. The next release will be better in this respect (too).

4.6 XFree Tools

The package includes two tools that work by connecting with the X server:

`gunzets_control`

The program accepts `on`, `off` or `info` as argument. It turns the touchscreen off, turns it on, or reports information about active input devices. The program needs to connect to an X server.

`warp`

The program moves the mouse pointer. It can be used to move the mouse when there is no active mouse. It accepts a series of letters on the command line: each uppercase N, S, W, E moves the pointer by 100 pixels to the north, south, west, east. Each lowercase n, s, w, e moves the pointer by 10 pixels.

5 Usb Driver

This driver has been developed and tested with Linux-2.4.0 and works with Linux-2.2.18 as well.

Input data is made available through an entry point in /dev (using devfs if available) (FIXME: devfs support is not yet implemented) or through the input engine (the input.o module) (FIXME: input support is not yet implemented). By default it uses its own /dev entry point, dynamically allocated from the misc device driver.

5.1 Quick Start

If all of your modules are in place, you can just invoke

```
make
./gunzets_load
```

and read the section about X support, above.

If you want to use the input mechanism (currently unimplemented), add `use_input=1` to the gunzets_load command line.

5.2 Compiling and loading

In order to successfully load the module you need to have the following facilities compiled in your kernel or loaded as modules:

usb generic support (‘`usbcore.o`’)

a usb host controller driver (‘`usb-ohci.o`’ o ‘`usb-uhci.o`’)

support for misc devices like the ps2 mouse (misc.o)

the input mechanism (input.o)

To compile the driver just ‘`make`’. If your 2.4 or 2.2 kernel headers are not available from ‘`/usr/src/linux/include`’ then specify KERNELDIR to point to your kernel source directory, either on the make command line or in the environment. For example:

```
make KERNELDIR=/usr/src/linux-2.4
```

To load the driver, use insmod:

```
insmod ./gunzets.o
```

To use the input mechanism, specify it on the command line:

```
insmod ./gunzets.o use_input=1
```

To automatically create the entry point in ‘`/dev`’ (needed if you are not using the input mechanism) run the gunzets_load script instead:

```
./gunzets_load
```

5.3 Unloading

You can unload the module only after disconnecting the touch panel, otherwise the rmmod program will return "Device Busy" even if no application is using the pane, because the USB mechanism is using it.

5.4 Installing

To install the module to the proper places you can run `make install`. You may also wish to copy '`gunzets_load`' from '`/usr/local/bin`' to '`/etc/rc.d/init.d`' or '`/etc/init.d`' and create a proper symbolic link for automatic load at boot.

5.5 Accessing data

Data read from the touch screen is available as raw data returned in PS/2 format in the file '`/dev/gunzets`'. You can use the same XFree driver described above for serial and PS/2 devices with this driver.

5.6 Usb Calibration

To calibrate the touch panel, run the "gunzets_calib" script, under X. See Section 4.5 [Calibration], page 6.

Table of Contents