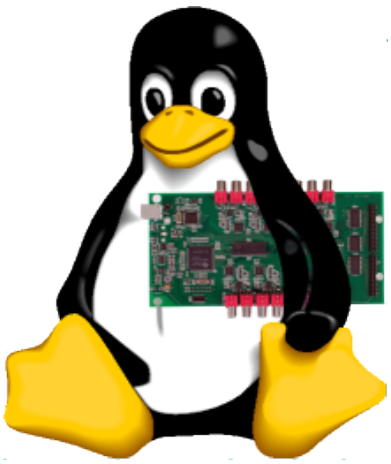
# The Signal Ranger Linux Documentation Project



by Percy Zahl

May 7, 2008

# Contents

# 1 Front Matter

**Signal Ranger Linux Project:**
**drivers, utilities, demos and documentation**

**Copyright (C) 2003 Percy Zahl**
**Email:** zahl@users.sourceforge.net

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; see the file COPYING. If not, write to the Free Software Foundation, Inc., 59 Temple Place - Suite 330, Boston, MA 02111-1307, USA.

See the end of this document for complete license: C

## 1.1 Abstract

The *Signal Ranger*[1] project is focusing on the development of an open source software environment to use the Signal Ranger DSP board (manufactured by *Soft dB*) with Linux and it provides a growing set of open source DSP software projects related to the Signal Ranger DSP board.

- Supported boards: Signal Ranger-SP2 and Signal Ranger-STD (needs the specialilized firmware loader to work)
- Generic Signal Ranger Hardware support for Linux: USB driver module ”usb-sranger”, tested on i386 and PPC platforms with the 2.4.x Linux kernel
- First stage (renumeration) firmwareloader for Signal Ranger-STD is suppied as static binary for i386 and PPC via *SF-filearea* (support for other platforms is possible, please ask the author!)
- Tools: COFF2 loader and DSP memory debugging and watch tool
- DSP applications:
  - SPM DSP software, used by *Gxsm*
  - Linux controls SRanger: Sound field processing demo
  - THX: Dolby (digital) “Pro-Logic” like Surround Decoder DSP software project
  - and more ...

This documentation expects the user to be familiar with the Signal Ranger DSP board itself – the Signal Ranger comes with a documentation covering that part (*Soft dB*). But it covers how to install and use the provided Linux tools and describes the features of the Linux – DSP interface. It also expects the user to be a little familiar with UNIX/Linux device mechanism. But it contains simple examples (written in C and Python) showing how to write and read data from and to the DSP via the `/dev/sranger0`.

[1] The Project can be found in the Internet at **http://sranger.sf.net**

# Part I

# Signal Ranger and Linux

# 2 The Signal Ranger USB Linux kernel module

To support the Signal Ranger DSP board by Linux a kernel module was written. It is based on the "usb-skeleton driver 0.7", which is included in the Linux 2.4.x kernel tree. This is a very good starting point for writing new USB device drivers. Here are some USB related WWW sites:

**http://www.linux-usb.org**
**http://libusb.sourceforge.net/doc/index.html**
**http://www.beyondlogic.org/usbnutshell/usb4.htm**
**http://linux-hotplug.sourceforge.net/**

*The* **Signal Ranger-SP2** *is supported natively. Support of the* **Signal Ranger-STD** *is now available, the extra firmware loader is needed, see chapter 2.1.3. After firmware load/renumeration it is detected by the unified module as described here.* ♪♫

## 2.1 Installation

### 2.1.1 Quick-start – Linux 2.4.x

To build the Signal Ranger[1] module follow this guidelines:

- have a installed and configured Linux kernel 2.4.x in `/usr/src/linux`
- get the `SRanger` module from CVS
- change into the directory `SRanger` and run "./autogen"
- change into the directory `SRanger/modules` and run "make"
- run "make install" as root to create `/dev/sranger0..3` or execute manually:
  "mknod -m 666 /dev/sranger0 c 180 200",
  "mknod -m 666 /dev/sranger1 c 180 201",

[1] I assume a Signal Ranger -SP2 or -STD board with Vendor/Product: 0xa59/0x103 (SP2), 0xa59/x0101 (STD), if your board is found (not accepted) with 0xa59/0x100 (STD, not renumerated) the firmware has not yet been downloaded! See 2.1.3.

...[2]
"mknod -m 666 /dev/sranger15 c 180 215"

- change into the directory `SRanger/loadusb` and run "make"

To load the module run "insmod usb-sranger.o" as root.

### 2.1.2 Quick-start – Linux 2.6.x

Assuming any Linux 2.6.x system with propper Linux kernel headers and gcc development tools installed. Have a look at the kernel module make and install instructions in `SRanger/modules-2.6.x/IN` for details.

To build the moduleSignal Ranger[3] you need to run this from command line, replace the kernel source paths below to match your version! And make sure you are using the same gcc (i.e. gcc-4.0, or such) as used for kernel compilation! Where "sudo" is put in front of the commands below yo need root permissions or your password if you are granted sudo rights.

- get the `SRanger` module from CVS
- change into the directory `SRanger` and run "./autogen" and "make"
- change into the directory `SRanger/modules-2.6.x`
- To build it: "make -C /usr/src/linux-2.6.12 SUBDIRS=$PWD modules"
- To install it: "make -C /usr/src/linux-2.6.12 SUBDIRS=$PWD modules_install"
- To load it: "sudo insmod usb-sranger.ko"
- From this point you are ready. Plug in the SR-SP2, or -STD (this one needs the firmware loaded before proceeding, see next chapter)
- To fix/adjust permissions: "sudo chmod a+rw /dev/sranger0"

♪♫ *For some reason on some systems you may need to fix permission after loading, the device will occurr automatically after insmod! Also check kernel messages "dmesg" and list you usb bus ("lsusb -v") in case of troubles.*

[2] create as many device file system nodes (up to 16) as you need.

[3] I assume a Signal Ranger -SP2 or -STD board with Vendor/Product: 0xa59/0x103 (SP2), 0xa59/x0101 (STD), if your board is found (not accepted) with 0xa59/0x100 (STD, not renumerated) the firmware has not yet been downloaded! See 2.1.3.

### 2.1.3 Signal Ranger-STD firmware loader setup

A little background first, the Signal Ranger-STD needs a "first-stage" firmware (of the USB controller on the SR board) download to get alive and then it "renumerates" itself on the USB bus and finally responds as Vendor/Product: 0xa59/x0101 (STD, firmware loaded). Only in this state it is detected by the Linux driver described here.

To get the firmware on the board please download the `fxload-sr-xxx` binary for your platform (xxx="i386" or "ppc") from here: *SF-filearea* (support for other platforms is possible, please ask the author!) The binaries are zipped using bzip2 (`.bz2` extension), use bunzip2 to unpack!

This program is compatible to the standard Linux tool called `fxload`, except it incorporates the Signal Ranger firmware in binary form[4] and you are allowed to use it as it is.

This version of `fxload-sr-xxx` is compatible to the Linux fxload, but it knows the Signal Ranger-STD firmware if it is called like:

```
fxload-sr-xxx -I SRangerFx
```

Command line options of fxload (`fxload-sr-xxx --help`) are:

```
usage: ./fxload [-vV] [-t type] [-D devpath]
                [-I firmware_hexfile | 'SRangerFx']
                [-s loader] [-c config_byte]
                [-L link] [-m mode]
... [-D devpath] overrides DEVICE= in env
... device types:  one of an21, fx, fx2
... at least one of -I, -L, -m is required
```

The firmware can be loaded manually like
`fxload-sr-xxx -I SRangerFx -D /proc/bus/usb/002/003`,
but please figure out the correct device path in your case instead of "/proc/bus/usb/002/003" and use the appropriate `fxload-sr-ppc` or `fxload-sr-i386`.

Unpack (bunzip2) it and place it into `/usr/local/sbin` or `/usr/local/bin`. Also make sure it is execuatble.

For automating this procedure after plugging in the USB cable of the Signal Ranger-STD the Linux hotplug mechanism can be used, here is how to set it up. I'm sorry, but it's to be done manually. So first make sure you have the hotplug stuff installed and working.

#### For 2.4.x systems:

Second, get the script `getdevpath` from here: *SF filearea, get getdevpath.bz2*

[4] The firmware contained herein is Copyright (C) 2003 by B.Paillard and SoftdB, http://www.softdb.com/ as an unpublished work. This notice does not imply unrestricted or public access to the source code from which this firmware image is derived. Except as noted below this firmware image may not be reproduced, used, sold or transferred to any third party without SoftDb's prior written consent. All Rights Reserved.

### For 2.6.x systems:

Use the script `SRanger/scripte/fxfirm.sh` to do the firmware load. Make sure the `fxload-386` is in you path, i.e. put it in `/usr/local/sbin`, make sure it is executable.

The `fxfirm.sh` script locates the current usb bus path and runs the fxload-sr-i386.

This is what is does, the grep/awk/... construct figures out the bus parameters, i.e. where is it currently plugged into:

```
#!/bin/sh
fxload-sr-i386 -I SRangerFx -D /proc/bus/usb/`cat /proc/bus/usb/devices
         | grep "T:\|P:"
         | grep -B1 0a59 | tac | tail -n1 | awk -F "=" '{print $2 $7}'
         | awk '{ printf "%03d/%03d",$1,$3 }'`
```

*If `/proc/bus/usb/...` dose not exist, the usbdevfs may be at `/dev/bus/usb/...`, please verify and adapt if needed.*

### For debian 2.4.x and some newer systems

Then create the following script (or use the `fxfirm.sh` script from above (rename it!), sorry, this depens on your distribution and used hot-plug subsystem, etc.).

**/etc/hotplug/usb/sranger_script**:

```
#!/bin/sh
# sranger_script: do not forget to replace the
#                 -xxx below by -ppc or -i386 below!

SR_DEV=$(/usr/local/sbin/getdevpath -va59 -p100)

/usr/local/sbin/fxload-sr-xxx -I SRangerFx -D $SR_DEV
```

and table file (note: do not break the line at "\" and do not put the "\" in the line!).

**/etc/hotplug/usb.usermap**:

```
sranger_script       0x0003 0x0a59 0x0100 0x0001 0x0000 0x0000 \
                      0x00 0x00 0x00 0x00 0x00 0x00 0x00000000
```

If the file `usb.usermap` exists and contains any entries, just add this line shown above!

## 2.1.4 Troubleshooting

Just in case something did not work as expected, do not worry, figure out the trouble by checking messages in your system log file. The kernel module places some messages and notifications

into the system log file (`/var/log/messages`). Before you start, make sure the kernel USB subsystem is working. Then start with no module loaded and no sranger USB connection.[5]

1. monitor messages in a separate xterm via "tail -f /var/log/messages" to see what happens at each step
2. load module "insmod usb-sranger.o" (there should be a message in messages)
3. connect SRanger via USB cable and power it on (order does not matter) (this should show some messages also)
4. now try "loadusb -m" (in mem debug mode for testing)

See appendix A for sample messages.

## 2.2 Module description

The Linux Signal Ranger module source is in `SRanger/modules/usb-sranger.c`. SRanger "IO-Control" definitions are defined in `SRanger/modules/sranger_ioctl.h`.

The module provides basic access to the DSP board(s) via the standard UNIX device file system mechanism. It supports up to 16[6] Signal Ranger devices plugged in at once (only limited by "#define MAX_DEVICES 16" in `usb-sranger.c`), starting with Minor 200 (set by "#define USB_SRANGER_MINOR_BASE 200" in `usb-sranger.c`).

The following standard file operations are implemented[7]:

**open** Open the device. Multiple openings are allowed and handled correctly in an multitasking environment. Each requested operation is blocking the device for all other operations (they are waiting), possibly requested by other tasks, until finished. This allows using the memory debugger (loadusb -m) simultaneous to other DSP using tasks or even multiple active file handles within on application to avoid frequent repositioning of the read/write address or DSP memory space via lseek.

Definition:
static int sranger_open (struct inode *inode, struct file *file)
Arguments:
**file**: Path to device (e.g. `/dev/sranger0`)
Return Value (type int):

[5] BTW: The Signal Ranger could be connected, but try this order first.
[6] Multiple board support not yet tested. – Please tell us if you can test it!
[7] Some arguments (like "struct inode *inode") are not described, because they are only used internally and not of interest from users point of view. Refer to a good Unix/Linux kernel documentation. All file operations are well defined for Unix/Linux and only the SRanger specific meanings are described here.

**int**: file handle id, a negative return value indicates an error code.

Errors:

**-ENODEV**: Device not found, no free minor

**-ENOMEM**: Memory allocation error

☺ Get Unix/Linux help about open: “man 2 open”

Example:

```
gint sranger = open (“/dev/sranger0”, O_RDWR);
if (sranger < 0) printf(“Error !”);
```

**llseek** This system call positions the start address and selects the DSP memory space via the orig parameter for all following read and write operations on the given file handle and selects the DSP memory space via the orig parameter.

Definition:

static loff_t sranger_llseek(struct file *file, loff_t offset, int orig)

Arguments:

**file**: File handle

**offset**: address in DSP memory space

**orig**: select one of the DSP memoryspaces, defined as:

Defines:

**SRANGER_SEEK_DATA_SPACE**: DSP data address space

**SRANGER_SEEK_PROG_SPACE**: DSP program address space

**SRANGER_SEEK_IO_SPACE**: DSP IO address space

Return Value (type loff_t):

**loff_t**: new address

Errors:

**-ENODEV**: Device not found or unplugged

**-ENOLINK**: Device link invalid

**-EINVAL**: Address (offset) invalid

♪♫ *The upper address limit (offset) is not checked, but is limited to 0xffff.*

☺ Get Unix/Linux help about lseek: “man 2 lseek”

Example:

```
lseek (sranger, 0x4000, SRANGER_SEEK_DATA_SPACE);
```

**read** Read data from DSP memory, starting at the address and memory space set by llseek. The read method uses the SR-kernel for data transfer, and therefore needs, the SR-kernel running. HPI data transfer is only possible via the **ioctl** method.

Definition:
static ssize_t sranger_read (struct file *file, char *buffer, size_t count, loff_t *ppos)
Arguments:
**file**: File handle
**buffer**: Pointer to buffer to accept data
**count**: Number of bytes to transfer.
Return Value (type ssize_t):
**ssize_t**: read count, a negative return value indicates an error code.
Errors:
**-ENODEV**: Device not found or unplugged
**-EFAULT**: Error while copying data to user space

*The size of the transfered block "count" is in bytes and not in words and must be even!* ♪♫

*The maximum transfer block size is 0x8000 bytes* ♪♫
*The user is responsible for correct endianess handling, especially on i386! On PPC systems the DSP and host are both big-endian.* ♪♫
Get Unix/Linux help about read: "man 2 read" ☺

Example:
```
short data[10];
read (sranger, data, sizeof (data));
```

**write** Write data to DSP memory, starting at the address and memory space set by llseek. The write method uses the SR-kernel for data transfer, and therefore needs, the SR-kernel running. HPI data transfer is only possible via the **ioctl** method.

Definition:
static ssize_t sranger_write (struct file *file, const char *buffer, size_t count, loff_t *ppos)
Arguments:
**file**: File handle
**buffer**: Pointer to buffer to accept data
**count**: Number of bytes to transfer
Return Value (type ssize_t):

**ssize_t**: write count, a negative return value indicates an error code.

Errors:

**-ENODEV**: Device not found or unplugged

**-EFAULT**: Error while copying data to user space

♪♫ *The size of the transfered block "count" is in bytes and not in words and must be even!*

♪♫ *The maximum transfer block size is 0x8000 bytes*

♪♫ *The user is responsible for correct endianess handling, especially on i386! On PPC systems the DSP and host are both big-endian.*

☺ Get Unix/Linux help about write: "man 2 write"

Example:

```
short data[10];
write (sranger, data, sizeof (data));
```

**ioctl** System call used to perform special "IO-controls". Usually not needed by user-applications. It is used in "loadusb" to reset the DSP and to upload the DSP kernel via HPI or for memory debugging.

Definition:

static int sranger_ioctl (struct inode *inode, struct file *file, unsigned int cmd, unsigned long arg)

Arguments:

**file**: File handle

**cmd**: Command Id, see "SRANGER_IOCTL_..." list below

**arg**: Value/Pointer for command.

Defines:

**SRANGER_IOCTL_VENDOR**: request USB vendor Id

Return value: Vendor Id

**SRANGER_IOCTL_PRODUCT**: request USB product Id

Return value: Product Id

**SRANGER_IOCTL_ASSERT_DSP_RESET**: activate DSP reset line

**SRANGER_IOCTL_RELEASE_DSP_RESET**: release DSP reset line

**SRANGER_IOCTL_INTERRUPT_DSP_FROM_HPI**: interrupt DSP from HPI

**SRANGER_IOCTL_W_LEDS**: set SRanger multi color LED

arg: LED color: 0: off, 1: red, 2: green, 3: orange

**SRANGER_IOCTL_HPI_MOVE_OUTREQUEST**: HPI data move out request, host to DSP

arg: pointer to args struct:

```
        struct{
            unsigned short index; // word address
            unsigned short length;// length in bytes, even!
            void *buffer;         // pointer to buffer
        } args;
```

**SRANGER_IOCTL_HPI_MOVE_INREQUEST**: HPI data move in request, DSP to host

arg: pointer to args struct:

```
        struct{
            unsigned short index; // word address
            unsigned short length;// length in bytes, even!
            void *buffer;         // pointer to buffer
        } args;
```

**SRANGER_IOCTL_HPI_CONTROL_OUTREQUEST**: HPI control out request

arg: pointer to args struct

**SRANGER_IOCTL_HPI_CONTROL_INREQUEST**: HPI control in request

arg: pointer to args struct

**SRANGER_IOCTL_KERNEL_EXEC**: DSP kernel exec request

arg: kernel execute address

**SRANGER_IOCTL_KERNEL_TIMEOUT**: set USB HPI timeout

Return Value (type int):

result of request, depending on command

**NULL**: OK, no error

Errors:

**-ENODEV**: Device not found or unplugged

**-EIO**: IO Error

**-ENOTTY**: did not understand this ioctl call

Get Unix/Linux help about ioctl: “man 2 ioctl” ☺

Examples:

```
vendorid = ioctl(sranger, SRANGER_IOCTL_VENDOR, (unsigned long)&vendor);
ioctl(sranger, SRANGER_IOCTL_W_LEDS, 1); // LED red
ioctl(sranger, SRANGER_IOCTL_W_LEDS, 2); // LED green
```

More samples how to use iotcl, read, write and seek can be found in the source code of `SRanger/loadusb/loadusb.c`. Especially look at the function “void mini_mem_debugger()”. ☺

☺ All requests are similar to the original SRanger hardware documentation!

**release** Close device.

Definition:
static int sranger_release (struct inode *inode, struct file *file)
Arguments:
**file**: File handle

☺ Get Unix/Linux help about close: “man 2 close”

Example:
close (sranger);

# 3 Example user program

This example demonstrates how to move data in between the DSP and host PC. The "DSP kernel data mode" is used. This will reduce the needed system calls to open, lseek, read, write and close.

Actually the user is not limited to C, any language which includes the mentioned basic system calls can be used. The second example is written in Python.

## 3.1 C program

This demo C program is part of the SRanger CVS module: `SRanger/demos/demo1.c`

Compile it manually using "gcc demo1.c -o demo1". You should have run the `autogen.sh` script before, because this generates the `config.h`. The only reason for including `config.h` is the definition if **WORDS_BIGENDIAN**. This is used to figure out if a correction of the byte order is needed or not. The function "swapshort()" is used for that.

```
/*
 * DSP tools for Linux -- demo program
 *
 * Compile with: "gcc demo1.c -o demo1"
 *
 * Copyright (C) 2003 Percy Zahl
 *
 * Authors: Percy Zahl <zahl@users.sf.net>
 * WWW Home: http://sranger.sf.net
 *
 * 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.
 */

#include <stdio.h>
#include <stdlib.h>
#include <unistd.h>
#include <sys/types.h>
#include <sys/stat.h>
#include <fcntl.h>
#include "../config.h"

/*
 * #include "../modules/sranger_ioctl.h"
 */

/* define yourself or include from ../modules/sranger_ioctl.h */
#define SRANGER_SEEK_DATA_SPACE  1
#define SRANGER_SEEK_PROG_SPACE  2
#define SRANGER_SEEK_IO_SPACE    3

/* Define to 1 if your processor stores words with the most significant byte
   first (like Motorola and SPARC, unlike Intel and VAX). */

#ifndef WORDS_BIGENDIAN
# define WORDS_BIGENDIAN 0
#endif

/* swap short */
void swapshort(short *addr)
{
        unsigned short temp1,temp2;

        temp1 = temp2 = *addr;
        *addr = ((temp2 & 0xFF) << 8) | ((temp1 >> 8) & 0xFF);
}

/* open SRanger device and return file handle */
int open_sr (){
        int sranger;
        sranger = open ("/dev/sranger0", O_RDWR);
        if (sranger < 0){
                fprintf (stderr,
                         "Error: Cannot connect to sranger.\n"
                         "Please plugin SR / load module.\n");
                exit(1);
        }
```

```
        return sranger;
}

int main (int argc, char *argv[]){
        int i, ret;
        short data[20];
        int sranger;

        if (WORDS_BIGENDIAN)
                printf ("Your host CPU is BIG ENDIAN (Motorola: PPC, Sparc).\n");
        else
                printf ("Your host CPU is LITTLE ENDIAN (Intel: i386, Vax).\n");

        sranger = open_sr ();

        /* set address to 0x1500, DSP data space */
        ret = lseek (sranger, 0x1500, SRANGER_SEEK_DATA_SPACE);
        printf ("lseek: 0x%04x\n", ret);

        /* create some new data */
        for (i=0; i<20; ++i)
                data[i] = i*i;

        if (!WORDS_BIGENDIAN)
                for (i=0; i<20; ++i)
                        swapshort (&data[i]);

        /* write data, 20 words */
        ret = write (sranger, data, sizeof (data));
        printf ("write: %d bytes\n", ret);

        /* read data, 20 words -- still at address 0x1500 */
        ret = read (sranger, data, sizeof (data));
        printf ("read: %d bytes\n", ret);

        if (!WORDS_BIGENDIAN)
                for (i=0; i<20; ++i)
                        swapshort (&data[i]);

        for (i=0; i<20; ++i)
                printf ("data[%3d] = %6d\n", i, data[i]);

        /* close sranger connection */
        ret = close (sranger);
        printf ("close: %d\n", ret);
```

```
        return 0;
}
```

## 3.2 Python program

The same program implemented in Python (tested with Python 2.1.3). This demo Python program is part of the SRanger CVS module: `SRanger/demos/demo1.py`

*Please note the different behavior of read and write here, the current seek position is moved with the number of bytes read or written.* ♪♫

*Endianess correction is performed automatically by Python, using the pack/unpack string functions and the format ">h". Please refer to the Python documentation.* ♪♫

```
#!/usr/bin/env python

## * Python demo program
## *
## * Copyright (C) 2003 Percy Zahl
## *
## * Author: Percy Zahl <zahl@users.sf.net>
## * WWW Home: http://sranger.sf.net
## *
## * 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.

import struct
import array

SRANGER_SEEK_DATA_SPACE = 1
SRANGER_SEEK_PROG_SPACE = 2
SRANGER_SEEK_IO_SPACE   = 3

def square(x): return x*x

def main():
    print ("Demo1: Python talks with Signal Ranger")
```

```
    # Data Format: short, automatic conversion system to BIG ENDIAN
    fmt    = ">hhhhhhhhhhhhhhhhhhhh"
    device = "/dev/sranger0"

    # create data
    data = map(square, range(0, 20))
    print ("Created Data      :", data)

    # open SRanger
    sr = open (device, "wb")
    sr.seek (0x1500, SRANGER_SEEK_DATA_SPACE)

    print ("writing...")
    # convert data to bin and write
    for value in data:
        sr.write (struct.pack (">h", value))
    sr.close ()

    sr = open (device, "rb")
    sr.seek (0x1500, SRANGER_SEEK_DATA_SPACE)

    print ("reading...")
    # read data and unpack from bin format
    dspdata = struct.unpack (fmt, sr.read (struct.calcsize (fmt)))
    sr.close ()

    print ("Data read from DSP:", dspdata)

    print ("Byby.")

# ----- program start -----
main()

# ----- end of python program -----
```

# Part II

# Utilities

# 4 The Signal Ranger Linux DSP COFF loader and memory debugger

## 4.1 Description

The program "loadusb" in `SRanger/loadusb` is the Linux version of the "Mini-Debugger" which comes with the Signal Ranger. It did not offer a GUI but it works as command line utility in a text terminal (xterm) based user interface to explore the DSP memory and allows some additional actions like memory dumping to screen or file, memory read, write, move, test, set or clear.

Refer to the end of this chapter for release notes for newer versions (`SRanger/loadusb, SRanger/loadusb64, SRanger/loadusb-mk2`).

In addition it has a special fast refreshing "memory watch" mode, which can be customized by editing some code.

Command line options, list this help via "loadusb –help":

```
pzahl@charon:~/SRanger/loadusb$ ./loadusb --help
COFF Loader/Terminal Emulator for Signal Ranger SP2/USB
 Version 0.01 by P.Zahl
======================================================================
loadusb -[dqvVbclEsrmLth] file.out (COFF2)
 default action: COFF2 load and exec at symbol address _c_int00
Device Options:
 -d: path to device (default:/dev/sranger0)
COFF load/dump options:
 -x: load only, no reset, no SRkernel, no exec
 -b: clear_bss
 -q: quite, -v: verbose, -V: more verbose
 -c: Dump, -C C Dump (dump only, no load)
Debug/Testing Options:
 -s: simulate only, no real load or reset
 -r: reset DSP and load SRkernel, exit
 -e address: Exec DSP prog at address, exit
 -m: run SRanger memory debugger, no load
 -L #no:  Set SRanger LED (#no: off=0, red=1, green=2, orange=3)
 -t: run terminal emulator/watch process after loading,
     may be used with -s to avoid loading [no terminal, but watch mode]
```

```
-w watchfile: use watchfile                [not yet available]
-h: this help, exit
```

## 4.2 Memory debugger mode

The memory debugger mode is entered via “loadusb -m”. Using this option no reset and not loading is performed. This allows to debug any running DSP program, even in parallel using multiple instances of the loadusb program itself!

### 4.2.1 Debugger command line interface

The loadusb “Mini SRanger Debugger mode” is a command line tool and accepts a small set of single letter commands. The command ’h’ for help lists all. After successful debugger start a prompt ’HPI:PROG:[1500]:0040 > ’ is presented. It is preceded by a four element vector, separated by ’:’:

The first element, here ’HPI’, represents the current data transfer mode used. It can be switched to the ’Kernel’ mode using the command “m:K” and back via “m:H’.

The second element, here ’PROG’, represents the current DSP memory space. It can be switched in between ’PROG’ (Program memory space) via “s:p”, ’DATA’ (Data memory space) via “s:d” and ’IO’ (IO space) via “s:i”.

The third element, here ’[1500]’, represents the current (word) address in hexadecimal notation. It can be changed via “a,address”, address is interpreted as hexadecimal word address. Enter “a,1000” to set the address to 0x1000.

The last element, here ’0040’, represents the current data buffer length (words) in hexadecimal. Change the length via “l,100” for example.

To read DSP data just type “r”, this will start reading length words at the current address from the DSP into the debuggers buffer. Type “d” to display the buffers content. The command “w” will write the currently stored data of the debuggers buffer into the DSP memory. Note, the buffer can be read at any valid address set by “a,address” and written later anywhere else (moved!) by changing the address after the read command was issued!

The buffer can be cleared “c” or set to any value “c,13” or set to a repeating value range “c,10,20”.

There are some more commands for you to figure out – ”a,1000,1100” –, enjoy!

☺ The command syntax is not case sensitive, a ’,’ can be used instead of ’:’. “m,k” is equivalent to “m:K”!

♪♫ *The command “t” (memory test) is temporary destructive, so do not overwrite vital words!*

Here is a sample session:

```
pzahl@charon:~/SRanger/loadusb$ ./loadusb -m
COFF Loader/Terminal Emulator for Signal Ranger SP2/USB
 Version 0.01 by P.Zahl
========================================================================
Product: 0x0a59
Product: 0x0103
--> B.Paillard, Signal Ranger SP2 -- OK.

--- Welcome to the Mini SRanger Debugger, enter 'h' for help ---

HPI:PROG:[1500]:0040 > h
*** Mini SRanger Debugger Help ***
---
Syntax        : mode:space:[address]:length > cmd[:arg1[:arg2]];
r             : read length words into buffer from SR address
w             : write length words from buffer to SR address
w,0xXXXX      : write (hex) word to SR address
w,XXXX        : write (dec) word to SR address
c[:XX[:YY]]   : clear [set buffer to XX [...YY modulo]]
f:w:file      : write buffer dump to file
f:r:file      : read buffer dump from file
f:y:file      : write words as data table [i y ys] to file
d             : dump length words starting of buffer
a:XXXX[:YYYY] : address (hex) [range]
l:XXXX        : length (hex)
m:[H|K]       : mode H:HPI,K:Kernel
s:[P|D|I]     : address spaces are Prog,Data,IO
b:adr         : branch to address (Kernel Exec)
o:file        : load COFF file
t             : run memtest at address/length
0/1/2/3       : Set LED off/red/green/orange
x             : reset SRanger and load SRkernel
q             : quit
---
Avoid writing to if not for some special reason:
0x80...:vectors, 0x100-0x2ff:kernel, 0x1000-0x1022:mbox
---
HPI:PROG:[1500]:0040 > m,k
Kernel:PROG:[1500]:0040 > s,d
Kernel:DATA:[1500]:0040 > a,500
Kernel:DATA:[0500]:0040 > l,20
Kernel:DATA:[0500]:0020 > r
lseek: 500
read: 40
Kernel:DATA:[0500]:0020 > d
0500: 00 02 29 f8 03 15 4f f8 03 84 10 f8 03 0f f6 b9   ..)...O.........
```

```
0508: 00 f8 03 91 f0 30 07 ff 88 11 f4 95 f4 95 30 e1   .....0........0.
0510: 18 00 f7 b9 57 f8 03 84 29 f8 03 16 4f f8 03 84   ....W...)...O...
0518: 10 f8 03 10 f6 b9 00 f8 03 91 f0 30 07 ff 88 11   ............0....
Kernel:DATA:[0500]:0020 > c
Clearing/setting/filling buffer...
Kernel:DATA:[0500]:0020 > d
0500: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00   ................
0508: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00   ................
0510: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00   ................
0518: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00   ................
Kernel:DATA:[0500]:0020 > c,11
Clearing/setting/filling buffer...
Kernel:DATA:[0500]:0020 > d
0500: 11 11 11 11 11 11 11 11 11 11 11 11 11 11 11 11   ................
0508: 11 11 11 11 11 11 11 11 11 11 11 11 11 11 11 11   ................
0510: 11 11 11 11 11 11 11 11 11 11 11 11 11 11 11 11   ................
0518: 11 11 11 11 11 11 11 11 11 11 11 11 11 11 11 11   ................
Kernel:DATA:[0500]:0020 > c,10,20
Clearing/setting/filling buffer...
Kernel:DATA:[0500]:0020 > d
0500: 10 11 12 13 14 15 16 17 18 19 1a 1b 1c 1d 1e 1f   ................
0508: 10 11 12 13 14 15 16 17 18 19 1a 1b 1c 1d 1e 1f   ................
0510: 10 11 12 13 14 15 16 17 18 19 1a 1b 1c 1d 1e 1f   ................
0518: 10 11 12 13 14 15 16 17 18 19 1a 1b 1c 1d 1e 1f   ................
Kernel:DATA:[0500]:0020 > r
lseek: 500
read: 40
Kernel:DATA:[0500]:0020 > d
0500: 00 02 29 f8 03 15 4f f8 03 84 10 f8 03 0f f6 b9   ..)...O.........
0508: 00 f8 03 91 f0 30 07 ff 88 11 f4 95 f4 95 30 e1   .....0........0.
0510: 18 00 f7 b9 57 f8 03 84 29 f8 03 16 4f f8 03 84   ....W...)...O...
0518: 10 f8 03 10 f6 b9 00 f8 03 91 f0 30 07 ff 88 11   ............0....
Kernel:DATA:[0500]:0020 > c,10,20
Clearing/setting/filling buffer...
Kernel:DATA:[0500]:0020 > w
lseek: 500
write: 40
Kernel:DATA:[0500]:0020 > c
Clearing/setting/filling buffer...
Kernel:DATA:[0500]:0020 > d
0500: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00   ................
0508: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00   ................
0510: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00   ................
0518: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00   ................
Kernel:DATA:[0500]:0020 > r
lseek: 500
```

```
read: 40
Kernel:DATA:[0500]:0020 > d
0500: 10 11 12 13 14 15 16 17 18 19 1a 1b 1c 1d 1e 1f   ................
0508: 10 11 12 13 14 15 16 17 18 19 1a 1b 1c 1d 1e 1f   ................
0510: 10 11 12 13 14 15 16 17 18 19 1a 1b 1c 1d 1e 1f   ................
0518: 10 11 12 13 14 15 16 17 18 19 1a 1b 1c 1d 1e 1f   ................
Kernel:DATA:[0500]:0020 > q
byby.
```

## 4.3 Loading and starting a DSP program

This is the main purpose and default action of the loadusb program. It issues a DSP reset, then loads the DSP kernel and waits until the kernel is up and running (LED turns green). Then the user program (the COFF2 executable with extension ".out", specified at command line) is downloaded section by section and verified at each step. Finally the address of the "C Entry-point" symbol "_c_int00" is looked up and a DSP-Kernel-Exec at this address is performed to start the user DSP program.

The loadusb program needs a DSP kernel (`kernel.out`) file in the current working directory. This kernel – whatever it is – is loaded after reset.

This way a different than the default `SRkernel.out` can be downloaded first, i.e. the Flash-boot kernel `SRKernel_FB.out` can be copied/symlinked to the file with the name `kernel.out` – this allows to reboot from Flash via "loadusb -r". ☺

Here is a sample session:

```
pzahl@charon:~/SRanger/loadusb$ ./loadusb  ../TiCC-project-files
/DolbyThx/Debug/FB_thx.out

COFF Loader/Terminal Emulator for Signal Ranger SP2/USB
 Version 0.01 by P.Zahl
========================================================================
Product: 0x0a59
Product: 0x0103
--> B.Paillard, Signal Ranger SP2 -- OK.

Resetting Signal Ranger loading SRkernel
SRkernel is up and running - LED green
Downloading 00f0bytes
Downloading 0ff8bytes
Downloading 05dabytes
Downloading 0548bytes
Downloading 0144bytes
Downloading 0020bytes
```

```
C Entrypoint _c_int00: 0x0e8d
Kernel Exec at address 0x0e8d
Branch initiated.
```

## 4.4 Load, start and watch

Sample session:

```
pzahl@charon:~/SRanger/loadusb$ ./loadusb -t  ../TiCC-project-files
/FB_spmcontrol/Debug/FB_spmcontrol.out

COFF Loader/Terminal Emulator for Signal Ranger SP2/USB
 Version 0.01 by P.Zahl
======================================================================
Product: 0x0a59
Product: 0x0103
--> B.Paillard, Signal Ranger SP2 -- OK.

Resetting Signal Ranger loading SRkernel
SRkernel is up and running - LED green
Downloading 00f0bytes
Downloading 0642bytes
Downloading 04f4bytes
Downloading 0470bytes
Downloading 0342bytes
Downloading 01a2bytes
Downloading 00a8bytes
Downloading 000ebytes
Downloading 07e2bytes
Downloading 048ebytes
Downloading 02bebytes
Downloading 0034bytes
C Entrypoint _c_int00: 0x26f0
Kernel Exec at address 0x26f0
Branch initiated.

---- sceenshot of quickly updating data view ----

 Running Terminal Emulator in DSP Memory Watch Mode.

 MagVerDatID:  FFFFEE01 0015 2003 0513 1001
      State.:  0000 0000 001F 001D 0000 04A3 0000 07D0 0000 0032 0000 012A 10C9
      AIC-in:       1      3      2      4      2     -9      0      0
```

```
   AIC-out:       0      0      0      0      0  32766  26212      0
 AIC-regGn:   26624  18432  10240   2048  26624  18602  10240   2218      1
                                          -1      0      2      2     -7
      Fifo:       0      0      0      0   2048      1      1      2      3      4
    PrbFifo:      0      0      0      0   2048      1     10     20     30     40
  Feedback:     327    327      0  -6136 -17998 -11862 -32768      0  32767
    Probe-S:      0      0      0      0
 Probe-NIF:       0      0      0      0      0      0
  Probe-DF:        262144       262144       262144       262144       262144       2621
  Probe-NX:       0      0      1      0
  Probe-AC:       0      0      0      0      0
  Probe-ST:       0      0      0      0      0      0


 SignalRanger TermEmu/Watch v1.1 (C) PZ *             Thu May 15 13:51:40 2003
```

*Have a look at the end of the loadusb.c source file to find out how to customize the data shown!* ♪♫

*A Flash write/program option is not yet available. Are you a volunteer?* ♪♫

## 4.5 Release Notes

*The original DSP code loader and mini memory debugger (loadusb) found in* `SRanger/loadusb` ♪♫ *works only on 32bit systems and got obsoleted by now. Use the newer loadusb64 version for all 32- and 64-bit systems.*

*In* `SRanger/loadusb64` *the 64-bit compatible and upgraded version of the original loadusb is located. Same binary name. You should now only use this version, as it work on all 32- and 64-bit systems.* ♪♫

*In* `SRanger/loadusb-mk2`*) the next generation loader for the SignalRanger-MK2 only, 32- and 64-bit host system compatible.* ♪♫

# Part III

# Appendix

# A Example module messages

Here is a typical logfile `/var/log/messages` output:

```
[insmod usb-sranger.o]
usb.c: registered new driver sranger
usb-sranger.c: USB SignalRanger SP2 Driver v0.1

[attach/power on SRanger SP2 to USB bus]
hub.c: new USB device 10:19.0-1, assigned address 18
usb-sranger.c: sranger_probe - probing for B.Paillard Signal Ranger SP2 DSP card
usb-sranger.c: sranger_probe - USB SignalRanger SP2 device now attached to sranger0
usb-sranger.c: sranger_probe - USB Devive Information:
usb-sranger.c: sranger_probe - Vendor :    0xa59
usb-sranger.c: sranger_probe - Product:    0x103
usb-sranger.c: sranger_probe - Device Ver: 1.21

[disconnect/power off]
usb.c: USB disconnect on device 10:19.0-1 address 18
usb-sranger.c: USB SignalRanger #0 now disconnected
```

# B Signal Ranger and Gxsm

## B.1 Startup

1. “pzahl@charon: /SRanger/loadusb$ ./loadusb -t ../TiCC-project-files/FB_spmcontrol/Debug/FB_spmcontrol.out”
2. start gxsm2
3. go to preference or through the first time druids and select the hardware “sranger-spm”
   and set other settings as needed... (only first time)
4. restart gxsm2

# C Terms and conditions

**GNU GENERAL PUBLIC LICENSE**
Version 2, June 1991

Copyright (C) 1989, 1991 Free Software Foundation, Inc.
59 Temple Place - Suite 330
Boston, MA 02111-1307, USA.

Everyone is permitted to copy and distribute verbatim copies of this license document, but changing it is not allowed.

**Preamble**

The licenses for most software are designed to take away your freedom to share and change it. By contrast, the GNU General Public License is intended to guarantee your freedom to share and change free software–to make sure the software is free for all its users. This General Public License applies to most of the Free Software Foundation's software and to any other program whose authors commit to using it. (Some other Free Software Foundation software is covered by the GNU Library General Public License instead.) You can apply it to your programs, too.

When we speak of free software, we are referring to freedom, not price. Our General Public Licenses are designed to make sure that you have the freedom to distribute copies of free software (and charge for this service if you wish), that you receive source code or can get it if you want it, that you can change the software or use pieces of it in new free programs; and that you know you can do these things.

To protect your rights, we need to make restrictions that forbid anyone to deny you these rights or to ask you to surrender the rights. These restrictions translate to certain responsibilities for you if you distribute copies of the software, or if you modify it.

For example, if you distribute copies of such a program, whether gratis or for a fee, you must give the recipients all the rights that you have. You must make sure that they, too, receive or can get the source code. And you must show them these terms so they know their rights.

We protect your rights with two steps: (1) copyright the software, and (2) offer you this license which gives you legal permission to copy, distribute and/or modify the software.

Also, for each author's protection and ours, we want to make certain that everyone understands that there is no warranty for this free software. If the software is modified by someone else and passed on, we want its recipients to know that what they have is not the original, so that any problems introduced by others will not reflect on the original authors' reputations.

Finally, any free program is threatened constantly by software patents. We wish to avoid the danger that redistributors of a free program will individually obtain patent licenses, in effect making the program proprietary. To prevent this, we have made it clear that any patent must be licensed for everyone's free use or not licensed at all.

The precise terms and conditions for copying, distribution and modification follow.

**GNU GENERAL PUBLIC LICENSE**
**TERMS AND CONDITIONS FOR COPYING, DISTRIBUTION AND MODIFICATION**

0. This License applies to any program or other work which contains a notice placed by the copyright holder saying it may be distributed under the terms of this General Public License. The "Program", below, refers to any such program or work, and a "work based on the Program" means either the Program or any derivative work under copyright law: that is to say, a work containing the Program or a portion of it, either verbatim or with modifications and/or translated into another language. (Hereinafter, translation is included without limitation in the term "modification".) Each licensee is addressed as "you".

Activities other than copying, distribution and modification are not covered by this License; they are outside its scope. The act of running the Program is not restricted, and the output from the Program is covered only if its contents constitute a work based on the Program (independent of having been made by running the Program). Whether that is true depends on what the Program does.

1. You may copy and distribute verbatim copies of the Program's source code as you receive it, in any medium, provided that you conspicuously and appropriately publish on each copy an appropriate copyright notice and disclaimer of warranty; keep intact all the notices that refer to this License and to the absence of any warranty; and give any other recipients of the Program a copy of this License along with the Program.

You may charge a fee for the physical act of transferring a copy, and you may at your option offer warranty protection in exchange for a fee.

2. You may modify your copy or copies of the Program or any portion of it, thus forming a work based on the Program, and copy and distribute such modifications or work under the terms of Section 1 above, provided that you also meet all of these conditions:

a) You must cause the modified files to carry prominent notices stating that you changed the files and the date of any change.

b) You must cause any work that you distribute or publish, that in whole or in part contains or is derived from the Program or any part thereof, to be licensed as a whole at no charge to all third parties under the terms of this License.

c) If the modified program normally reads commands interactively when run, you must cause it, when started running for such interactive use in the most ordinary way, to print or display an announcement including an appropriate copyright notice and a notice that there is no warranty (or else, saying that you provide a warranty) and that users may redistribute

the program under these conditions, and telling the user how to view a copy of this License. (Exception: if the Program itself is interactive but does not normally print such an announcement, your work based on the Program is not required to print an announcement.)

These requirements apply to the modified work as a whole. If identifiable sections of that work are not derived from the Program, and can be reasonably considered independent and separate works in themselves, then this License, and its terms, do not apply to those sections when you distribute them as separate works. But when you distribute the same sections as part of a whole which is a work based on the Program, the distribution of the whole must be on the terms of this License, whose permissions for other licensees extend to the entire whole, and thus to each and every part regardless of who wrote it.

Thus, it is not the intent of this section to claim rights or contest your rights to work written entirely by you; rather, the intent is to exercise the right to control the distribution of derivative or collective works based on the Program.

In addition, mere aggregation of another work not based on the Program with the Program (or with a work based on the Program) on a volume of a storage or distribution medium does not bring the other work under the scope of this License.

3. You may copy and distribute the Program (or a work based on it, under Section 2) in object code or executable form under the terms of Sections 1 and 2 above provided that you also do one of the following:

a) Accompany it with the complete corresponding machine-readable source code, which must be distributed under the terms of Sections 1 and 2 above on a medium customarily used for software interchange; or,

b) Accompany it with a written offer, valid for at least three years, to give any third party, for a charge no more than your cost of physically performing source distribution, a complete machine-readable copy of the corresponding source code, to be distributed under the terms of Sections 1 and 2 above on a medium customarily used for software interchange; or,

c) Accompany it with the information you received as to the offer to distribute corresponding source code. (This alternative is allowed only for noncommercial distribution and only if you received the program in object code or executable form with such an offer, in accord with Subsection b above.)

The source code for a work means the preferred form of the work for making modifications to it. For an executable work, complete source code means all the source code for all modules it contains, plus any associated interface definition files, plus the scripts used to control compilation and installation of the executable. However, as a special exception, the source code distributed need not include anything that is normally distributed (in either source or binary form) with the major components (compiler, kernel, and so on) of the operating system on which the executable runs, unless that component itself accompanies the executable.

If distribution of executable or object code is made by offering access to copy from a designated place, then offering equivalent access to copy the source code from the same place counts as distribution of the source code, even though third parties are not compelled to copy the source

along with the object code.

4. You may not copy, modify, sublicense, or distribute the Program except as expressly provided under this License. Any attempt otherwise to copy, modify, sublicense or distribute the Program is void, and will automatically terminate your rights under this License. However, parties who have received copies, or rights, from you under this License will not have their licenses terminated so long as such parties remain in full compliance.

5. You are not required to accept this License, since you have not signed it. However, nothing else grants you permission to modify or distribute the Program or its derivative works. These actions are prohibited by law if you do not accept this License. Therefore, by modifying or distributing the Program (or any work based on the Program), you indicate your acceptance of this License to do so, and all its terms and conditions for copying, distributing or modifying the Program or works based on it.

6. Each time you redistribute the Program (or any work based on the Program), the recipient automatically receives a license from the original licensor to copy, distribute or modify the Program subject to these terms and conditions. You may not impose any further restrictions on the recipients' exercise of the rights granted herein. You are not responsible for enforcing compliance by third parties to this License.

7. If, as a consequence of a court judgment or allegation of patent infringement or for any other reason (not limited to patent issues), conditions are imposed on you (whether by court order, agreement or otherwise) that contradict the conditions of this License, they do not excuse you from the conditions of this License. If you cannot distribute so as to satisfy simultaneously your obligations under this License and any other pertinent obligations, then as a consequence you may not distribute the Program at all. For example, if a patent license would not permit royalty-free redistribution of the Program by all those who receive copies directly or indirectly through you, then the only way you could satisfy both it and this License would be to refrain entirely from distribution of the Program.

If any portion of this section is held invalid or unenforceable under any particular circumstance, the balance of the section is intended to apply and the section as a whole is intended to apply in other circumstances.

It is not the purpose of this section to induce you to infringe any patents or other property right claims or to contest validity of any such claims; this section has the sole purpose of protecting the integrity of the free software distribution system, which is implemented by public license practices. Many people have made generous contributions to the wide range of software distributed through that system in reliance on consistent application of that system; it is up to the author/donor to decide if he or she is willing to distribute software through any other system and a licensee cannot impose that choice.

This section is intended to make thoroughly clear what is believed to be a consequence of the rest of this License.

8. If the distribution and/or use of the Program is restricted in certain countries either by patents or by copyrighted interfaces, the original copyright holder who places the Program under this License may add an explicit geographical distribution limitation excluding those countries,

so that distribution is permitted only in or among countries not thus excluded. In such case, this License incorporates the limitation as if written in the body of this License.

9. The Free Software Foundation may publish revised and/or new versions of the General Public License from time to time. Such new versions will be similar in spirit to the present version, but may differ in detail to address new problems or concerns.

Each version is given a distinguishing version number. If the Program specifies a version number of this License which applies to it and "any later version", you have the option of following the terms and conditions either of that version or of any later version published by the Free Software Foundation. If the Program does not specify a version number of this License, you may choose any version ever published by the Free Software Foundation.

10. If you wish to incorporate parts of the Program into other free programs whose distribution conditions are different, write to the author to ask for permission. For software which is copyrighted by the Free Software Foundation, write to the Free Software Foundation; we sometimes make exceptions for this. Our decision will be guided by the two goals of preserving the free status of all derivatives of our free software and of promoting the sharing and reuse of software generally.

**NO WARRANTY**

11. BECAUSE THE PROGRAM IS LICENSED FREE OF CHARGE, THERE IS NO WARRANTY FOR THE PROGRAM, TO THE EXTENT PERMITTED BY APPLICABLE LAW. EXCEPT WHEN OTHERWISE STATED IN WRITING THE COPYRIGHT HOLDERS AND/OR OTHER PARTIES PROVIDE THE PROGRAM "AS IS" WITHOUT WARRANTY OF ANY KIND, EITHER EXPRESSED OR IMPLIED, INCLUDING, BUT NOT LIMITED TO, THE IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR PURPOSE. THE ENTIRE RISK AS TO THE QUALITY AND PERFORMANCE OF THE PROGRAM IS WITH YOU. SHOULD THE PROGRAM PROVE DEFECTIVE, YOU ASSUME THE COST OF ALL NECESSARY SERVICING, REPAIR OR CORRECTION.

12. IN NO EVENT UNLESS REQUIRED BY APPLICABLE LAW OR AGREED TO IN WRITING WILL ANY COPYRIGHT HOLDER, OR ANY OTHER PARTY WHO MAY MODIFY AND/OR REDISTRIBUTE THE PROGRAM AS PERMITTED ABOVE, BE LIABLE TO YOU FOR DAMAGES, INCLUDING ANY GENERAL, SPECIAL, INCIDENTAL OR CONSEQUENTIAL DAMAGES ARISING OUT OF THE USE OR INABILITY TO USE THE PROGRAM (INCLUDING BUT NOT LIMITED TO LOSS OF DATA OR DATA BEING RENDERED INACCURATE OR LOSSES SUSTAINED BY YOU OR THIRD PARTIES OR A FAILURE OF THE PROGRAM TO OPERATE WITH ANY OTHER PROGRAMS), EVEN IF SUCH HOLDER OR OTHER PARTY HAS BEEN ADVISED OF THE POSSIBILITY OF SUCH DAMAGES.

**END OF TERMS AND CONDITIONS**

# About this project

All started as I heard first about the Signal Ranger DSP board on a Thursday, Nov. 8th – I remember, because we, the Gxsm team, had an discussion about it and possible future DSP solutions the next day at the AVS conference in Denver – via an E-mail from Alastair McLean, who is interested into the Gxsm project:

*"... There is a Canadian company (SoftdB) that sells a very inexpensive DSP board called the Signal Ranger DSP Board. ..."*

The trouble was getting a board, because there was no funding and no need for our recent science projects for any upgrade, but finally I decided to buy a board for myself to find out more about it – this was in December 2002. I used Christmas and new year holidays to develop the Signal Ranger USB Linux module and succeeded, I appreciate the friendly help of Bruno Paillard with several questions.

The SPM DSP software development started...

Golden, Colorado USA, February 2, 2003 Percy Zahl

. . . and started working end February and is almost complete today.
Negenborn, Germany, May 2003 Percy Zahl

# D Comments and Questions

General comments and questions regarding this document should be sent by email to zahl@users.sourceforge.net. If you find specific errors in this document, please report the bug at the SRanger Bug Tracker at SourceForge.

Questions regarding how to use the information in this document should be sent to the appropriate SRanger help or discussion forum at SourceForge.

For any of these channels, please be sure not to send HTML email. Thanks.