```
<?xml version="1.0" encoding="UTF-8"?>
<!DOCTYPE book PUBLIC "-//OASIS//DTD DocBook XML V4.1.2//EN"
        "http://www.oasis-open.org/docbook/xml/4.1.2/docbookx.dtd" []>

<book id="Reed-Solomon-Library-Guide">
 <bookinfo>
  <title>Reed-Solomon Library Programming Interface</title>

  <authorgroup>
   <author>
    <firstname>Thomas</firstname>
    <surname>Gleixner</surname>
    <affiliation>
     <address>
      <email>tglx@linutronix.de</email>
     </address>
    </affiliation>
   </author>
  </authorgroup>

  <copyright>
   <year>2004</year>
   <holder>Thomas Gleixner</holder>
  </copyright>

  <legalnotice>
   <para>
     This documentation is free software; you can redistribute
     it and/or modify it under the terms of the GNU General Public
     License version 2 as published by the Free Software Foundation.
   </para>

   <para>
     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.
   </para>

   <para>
     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
   </para>

   <para>
     For more details see the file COPYING in the source
     distribution of Linux.
   </para>
  </legalnotice>
 </bookinfo>

<toc></toc>

  <chapter id="intro">
```

```
      <title>Introduction</title>
  <para>
        The generic Reed-Solomon Library provides encoding, decoding
        and error correction functions.
  </para>
  <para>
        Reed-Solomon codes are used in communication and storage
        applications to ensure data integrity.
  </para>
  <para>
        This documentation is provided for developers who want to utilize
        the functions provided by the library.
  </para>
  </chapter>

  <chapter id="bugs">
     <title>Known Bugs And Assumptions</title>
  <para>
        None.
  </para>
  </chapter>

  <chapter id="usage">
        <title>Usage</title>
        <para>
                This chapter provides examples of how to use the library.
        </para>
        <sect1>
                <title>Initializing</title>
                <para>
                        The init function init_rs returns a pointer to an
                        rs decoder structure, which holds the necessary
                        information for encoding, decoding and error correction
                        with the given polynomial. It either uses an existing
                        matching decoder or creates a new one. On creation all
                        the lookup tables for fast en/decoding are created.
                        The function may take a while, so make sure not to
                        call it in critical code paths.
                </para>
                <programlisting>
/* the Reed Solomon control structure */
static struct rs_control *rs_decoder;

/* Symbolsize is 10 (bits)
 * Primitive polynomial is x^10+x^3+1
 * first consecutive root is 0
 * primitive element to generate roots = 1
 * generator polynomial degree (number of roots) = 6
 */
rs_decoder = init_rs (10, 0x409, 0, 1, 6);
                </programlisting>
        </sect1>
        <sect1>
                <title>Encoding</title>
                <para>
                        The encoder calculates the Reed-Solomon code over
```

the given data length and stores the result in
the parity buffer. Note that the parity buffer must
be initialized before calling the encoder.
</para>
<para>
The expanded data can be inverted on the fly by
providing a non-zero inversion mask. The expanded data
is
XOR'ed with the mask. This is used e.g. for FLASH
ECC, where the all 0xFF is inverted to an all 0x00.
The Reed-Solomon code for all 0x00 is all 0x00. The
code is inverted before storing to FLASH so it is 0xFF
too. This prevents that reading from an erased FLASH
results in ECC errors.
</para>
<para>
The databytes are expanded to the given symbol size
on the fly. There is no support for encoding continuous
bitstreams with a symbol size != 8 at the moment. If
it is necessary it should be not a big deal to implement
such functionality.
</para>
<programlisting>
/* Parity buffer. Size = number of roots */
uint16_t par[6];
/* Initialize the parity buffer */
memset(par, 0, sizeof(par));
/* Encode 512 byte in data8. Store parity in buffer par */
encode_rs8 (rs_decoder, data8, 512, par, 0);
</programlisting>
</sect1>
<sect1>
<title>Decoding</title>
<para>
The decoder calculates the syndrome over
the given data length and the received parity symbols
and corrects errors in the data.
</para>
<para>
If a syndrome is available from a hardware decoder
then the syndrome calculation is skipped.
</para>
<para>
The correction of the data buffer can be suppressed
by providing a correction pattern buffer and an error
location buffer to the decoder. The decoder stores the
calculated error location and the correction bitmask
in the given buffers. This is useful for hardware
decoders which use a weird bit ordering scheme.
</para>
<para>
The databytes are expanded to the given symbol size
on the fly. There is no support for decoding continuous
bitstreams with a symbolsize != 8 at the moment. If
it is necessary it should be not a big deal to implement
such functionality.

```
		</para>

		<sect2>
		<title>
			Decoding with syndrome calculation, direct data
correction
		</title>
		<programlisting>
/* Parity buffer. Size = number of roots */
uint16_t par[6];
uint8_t  data[512];
int numerr;
/* Receive data */
.....
/* Receive parity */
.....
/* Decode 512 byte in data8.*/
numerr = decode_rs8 (rs_decoder, data8, par, 512, NULL, 0, NULL, 0, NULL);
		</programlisting>
		</sect2>

		<sect2>
		<title>
			Decoding with syndrome given by hardware decoder, direct
data correction
		</title>
		<programlisting>
/* Parity buffer. Size = number of roots */
uint16_t par[6], syn[6];
uint8_t  data[512];
int numerr;
/* Receive data */
.....
/* Receive parity */
.....
/* Get syndrome from hardware decoder */
.....
/* Decode 512 byte in data8.*/
numerr = decode_rs8 (rs_decoder, data8, par, 512, syn, 0, NULL, 0, NULL);
		</programlisting>
		</sect2>

		<sect2>
		<title>
			Decoding with syndrome given by hardware decoder, no
direct data correction.
		</title>
		<para>
			Note: It's not necessary to give data and received
parity to the decoder.
		</para>
		<programlisting>
/* Parity buffer. Size = number of roots */
uint16_t par[6], syn[6], corr[8];
uint8_t  data[512];
int numerr, errpos[8];
```

```
/* Receive data */
.....
/* Receive parity */
.....
/* Get syndrome from hardware decoder */
.....
/* Decode 512 byte in data8.*/
numerr = decode_rs8 (rs_decoder, NULL, NULL, 512, syn, 0, errpos, 0, corr);
for (i = 0; i &lt; numerr; i++) {
        do_error_correction_in_your_buffer(errpos[i], corr[i]);
}
                </programlisting>
                </sect2>
        </sect1>
        <sect1>
                <title>Cleanup</title>
                <para>
                        The function free_rs frees the allocated resources,
                        if the caller is the last user of the decoder.
                </para>
                <programlisting>
/* Release resources */
free_rs(rs_decoder);
                </programlisting>
        </sect1>

  </chapter>

  <chapter id="structs">
     <title>Structures</title>
     <para>
     This chapter contains the autogenerated documentation of the structures
which are
     used in the Reed-Solomon Library and are relevant for a developer.
     </para>
!Iinclude/linux/rslib.h
  </chapter>

  <chapter id="pubfunctions">
     <title>Public Functions Provided</title>
     <para>
     This chapter contains the autogenerated documentation of the Reed-Solomon
functions
     which are exported.
     </para>
!Elib/reed_solomon/reed_solomon.c
  </chapter>

  <chapter id="credits">
     <title>Credits</title>
        <para>
                The library code for encoding and decoding was written by Phil
Karn.
        </para>
        <programlisting>
                Copyright 2002, Phil Karn, KA9Q
```

```
                May be used under the terms of the GNU General Public License
(GPL)
        </programlisting>
        <para>
                The wrapper functions and interfaces are written by Thomas
Gleixner.
        </para>
        <para>
                Many users have provided bugfixes, improvements and helping
hands for testing.
                Thanks a lot.
        </para>
        <para>
                The following people have contributed to this document:
        </para>
        <para>
                Thomas Gleixner<email>tglx@linutronix.de</email>
        </para>
  </chapter>
</book>
```