```
<?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="regulator-api">
 <bookinfo>
  <title>Voltage and current regulator API</title>

  <authorgroup>
   <author>
    <firstname>Liam</firstname>
    <surname>Girdwood</surname>
    <affiliation>
     <address>
      <email>lrg@slimlogic.co.uk</email>
     </address>
    </affiliation>
   </author>
   <author>
    <firstname>Mark</firstname>
    <surname>Brown</surname>
    <affiliation>
     <orgname>Wolfson Microelectronics</orgname>
     <address>
      <email>broonie@opensource.wolfsonmicro.com</email>
     </address>
    </affiliation>
   </author>
  </authorgroup>

  <copyright>
   <year>2007-2008</year>
   <holder>Wolfson Microelectronics</holder>
  </copyright>
  <copyright>
   <year>2008</year>
   <holder>Liam Girdwood</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>
        This framework is designed to provide a standard kernel
        interface to control voltage and current regulators.
    </para>
    <para>
        The intention is to allow systems to dynamically control
        regulator power output in order to save power and prolong
        battery life.  This applies to both voltage regulators (where
        voltage output is controllable) and current sinks (where current
        limit is controllable).
    </para>
    <para>
        Note that additional (and currently more complete) documentation
        is available in the Linux kernel source under
        <filename>Documentation/power/regulator</filename>.
    </para>

    <sect1 id="glossary">
       <title>Glossary</title>
       <para>
        The regulator API uses a number of terms which may not be
        familiar:
       </para>
       <glossary>

         <glossentry>
           <glossterm>Regulator</glossterm>
           <glossdef>
             <para>
        Electronic device that supplies power to other devices.  Most
        regulators can enable and disable their output and some can also
        control their output voltage or current.
             </para>
           </glossdef>
         </glossentry>

         <glossentry>
           <glossterm>Consumer</glossterm>
           <glossdef>
             <para>
        Electronic device which consumes power provided by a regulator.
```

```
      These may either be static, requiring only a fixed supply, or
      dynamic, requiring active management of the regulator at
      runtime.
           </para>
         </glossdef>
       </glossentry>

       <glossentry>
         <glossterm>Power Domain</glossterm>
         <glossdef>
           <para>
      The electronic circuit supplied by a given regulator, including
      the regulator and all consumer devices.  The configuration of
      the regulator is shared between all the components in the
      circuit.
           </para>
         </glossdef>
       </glossentry>

       <glossentry>
         <glossterm>Power Management Integrated Circuit</glossterm>
         <acronym>PMIC</acronym>
         <glossdef>
           <para>
      An IC which contains numerous regulators and often also other
      subsystems.  In an embedded system the primary PMIC is often
      equivalent to a combination of the PSU and southbridge in a
      desktop system.
           </para>
         </glossdef>
       </glossentry>
      </glossary>
   </sect1>
</chapter>

<chapter id="consumer">
   <title>Consumer driver interface</title>
   <para>
     This offers a similar API to the kernel clock framework.
     Consumer drivers use <link
     linkend='API-regulator-get'>get</link> and <link
     linkend='API-regulator-put'>put</link> operations to acquire and
     release regulators.  Functions are
     provided to <link linkend='API-regulator-enable'>enable</link>
     and <link linkend='API-regulator-disable'>disable</link> the
     reguator and to get and set the runtime parameters of the
     regulator.
   </para>
   <para>
     When requesting regulators consumers use symbolic names for their
     supplies, such as "Vcc", which are mapped into actual regulator
     devices by the machine interface.
   </para>
   <para>
      A stub version of this API is provided when the regulator
      framework is not in use in order to minimise the need to use
   ifdefs.
</para>

<sect1 id="consumer-enable">
  <title>Enabling and disabling</title>
  <para>
    The regulator API provides reference counted enabling and
    disabling of regulators. Consumer devices use the <function><link
    linkend='API-regulator-enable'>regulator_enable</link></function>
    and <function><link
    linkend='API-regulator-disable'>regulator_disable</link>
    </function> functions to enable and disable regulators.  Calls
    to the two functions must be balanced.
  </para>
  <para>
    Note that since multiple consumers may be using a regulator and
    machine constraints may not allow the regulator to be disabled
    there is no guarantee that calling
    <function>regulator_disable</function> will actually cause the
    supply provided by the regulator to be disabled. Consumer
    drivers should assume that the regulator may be enabled at all
    times.
  </para>
</sect1>

<sect1 id="consumer-config">
  <title>Configuration</title>
  <para>
    Some consumer devices may need to be able to dynamically
    configure their supplies.  For example, MMC drivers may need to
    select the correct operating voltage for their cards.  This may
    be done while the regulator is enabled or disabled.
  </para>
  <para>
    The <function><link
    linkend='API-regulator-set-voltage'>regulator_set_voltage</link>
    </function> and <function><link
    linkend='API-regulator-set-current-limit'
    >regulator_set_current_limit</link>
    </function> functions provide the primary interface for this.
    Both take ranges of voltages and currents, supporting drivers
    that do not require a specific value (eg, CPU frequency scaling
    normally permits the CPU to use a wider range of supply
    voltages at lower frequencies but does not require that the
    supply voltage be lowered).  Where an exact value is required
    both minimum and maximum values should be identical.
  </para>
</sect1>

<sect1 id="consumer-callback">
  <title>Callbacks</title>
  <para>
     Callbacks may also be <link
     linkend='API-regulator-register-notifier'>registered</link>
     for events such as regulation failures.
  </para>
  </sect1>
</chapter>

<chapter id="driver">
  <title>Regulator driver interface</title>
  <para>
    Drivers for regulator chips <link
    linkend='API-regulator-register'>register</link> the regulators
    with the regulator core, providing operations structures to the
    core.  A <link
    linkend='API-regulator-notifier-call-chain'>notifier</link> interface
    allows error conditions to be reported to the core.
  </para>
  <para>
    Registration should be triggered by explicit setup done by the
    platform, supplying a <link
    linkend='API-struct-regulator-init-data'>struct
    regulator_init_data</link> for the regulator containing
    <link linkend='machine-constraint'>constraint</link> and
    <link linkend='machine-supply'>supply</link> information.
  </para>
</chapter>

<chapter id="machine">
  <title>Machine interface</title>
  <para>
    This interface provides a way to define how regulators are
    connected to consumers on a given system and what the valid
    operating parameters are for the system.
  </para>

  <sect1 id="machine-supply">
    <title>Supplies</title>
    <para>
      Regulator supplies are specified using <link
      linkend='API-struct-regulator-consumer-supply'>struct
      regulator_consumer_supply</link>.  This is done at
      <link linkend='driver'>driver registration
      time</link> as part of the machine constraints.
    </para>
  </sect1>

  <sect1 id="machine-constraint">
    <title>Constraints</title>
    <para>
      As well as definining the connections the machine interface
      also provides constraints definining the operations that
      clients are allowed to perform and the parameters that may be
      set.  This is required since generally regulator devices will
      offer more flexibility than it is safe to use on a given
      system, for example supporting higher supply voltages than the
      consumers are rated for.
    </para>
    <para>
      This is done at <link linkend='driver'>driver
      registration time</link> by providing a <link
```

```
          linkend='API-struct-regulation-constraints'>struct
          regulation_constraints</link>.
        </para>
        <para>
          The constraints may also specify an initial configuration for the
          regulator in the constraints, which is particularly useful for
          use with static consumers.
        </para>
      </sect1>
   </chapter>

   <chapter id="api">
     <title>API reference</title>
     <para>
       Due to limitations of the kernel documentation framework and the
       existing layout of the source code the entire regulator API is
       documented here.
     </para>
!Iinclude/linux/regulator/consumer.h
!Iinclude/linux/regulator/machine.h
!Iinclude/linux/regulator/driver.h
!Edrivers/regulator/core.c
   </chapter>
</book>
```