TI81XX PSP AVS Driver Guide

Linux PSP AVS Driver Guide

Linux PSP

IMPORTANT

This document applies only to DM816x.
DM816x refers to DM816x/C6A816x/AM389x devices unless specified.

This AVS driver is applicable only for Characterized silicon samples

Introduction

SmartReflex-AVS is a technology that uses adaptive power supply to achieve the goal of reducing active power consumption. DM816x device have Class 2B implementation of smart reflex and this allows dynamic AVS using Software.

Acronyms & definitions

SmartReflex Driver: Acronyms

Acronym	Definition
AVS	Adaptive Voltage Scaling
SR	SmartReflex
HVT	High Voltage Threshold sensor
SVT	Standard Voltage Threshold sensor
GPIO	General Purpose Input Output
PMIC	Power Management Integrated Circuit
VR	Voltage regulator

Driver Usage

On silicon which has been characterized for AVS operation and resulting characterization data has been blown into eFuse registers, the SR driver initialized during system boot-up and starts the voltage scaling operation.

On silicon which does not have the nTarget values blown, then SR driver will print the following error messages

1. During SR driver initialization

```
smartreflex: Driver is not initialized, nTarget values are not found
```

Note: If you saw above message in boot log, then the device does not have nTarget values. If you think the device what you have is AVS enabled then check these register values from u-boot prompt, these registers should have **Non-Zero values**

a. SmartReflex HVT nTarget value, address - 0x481406AC

```
md 0x481406AC
```

b. SmartReflex SVT nTarget value, address - 0x481406A8

```
md 0x481406A8
```

2. While SR driver enabling

```
smartreflex: SR module not enabled, nTarget values are not found
```

Driver Configuration

This section describes about the kernel configurations for Smartreflex driver & its dependencies

Voltage Regulator Driver configuration

The default kernel configuration enables support for GPIO voltage regulator Driver (built into the kernel).

To enable or disable GPIO based voltage regulator driver kernel build, follow these steps:

```
$ make CROSS_COMPILE=arm-none-linux-gnueabi- ARCH=arm menuconfig
```

- Select Device Drivers from the main menu.

```
    Power management options  --->
[*] Networking support  --->
    Device Drivers  --->
    File systems  --->
...
...
```

- Select Voltage and Current Regulator Support from the menu.

```
    Sonics Silicon Backplane  --->
-*- Multifunction device drivers  --->
-*- Voltage and Current Regulator Support  --->
<*> Multimedia support  --->
...
...
```

- Select GPIO voltage regulator from the menu

```
< > National Semiconductors LP3972 PMIC regulator driver
< > TI TPS65023 Power regulators
<*> GPIO voltage regulator
< > TI TPS6507X Power regulators
< > Intersil ISL6271A Power regulator
...
...
```

- After doing driver selection, exit and save the kernel configuration when prompted.

SR Driver configuration

The default kernel configuration enables support for SR Driver (built into the kernel).

To enable or disable SR driver kernel build, follow these steps:

```
$ make CROSS_COMPILE=arm-none-linux-gnueabi- ARCH=arm menuconfig
```

- Select System Type from the main menu.

```
    General setup  --->
[*] Enable loadable module support  --->
-*- Enable the block layer  --->
    System Type  --->
    Bus support  --->
...
...
```

- Select TI OMAP Common Features from the menu.

```
[*] MMU-based Paged Memory Management Support
    ARM system type (TI OMAP)  --->
    TI OMAP Common Features  --->
    TI OMAP2/3/4 Specific Features  --->
...
...
```

- Select SmartReflex class2 support for ti816x from menu

```
OMAP System Type (TI OMAP2/3/4/81XX)  --->
    *** OMAP Feature Selections ***
[ ] Reset unused clocks during boot
[*] SmartReflex class2 support for ti816x
[*] OMAP multiplexing support
...
```

- After doing driver selection, exit and save the kernel configuration when prompted.

Debugfs configuration

The default kernel configuration enables support for debugfs (built into the kernel).

To enable or disable debugfs kernel build, follow these steps:

```
$ make CROSS_COMPILE=arm-none-linux-gnueabi- ARCH=arm menuconfig
```

- Select Device Drivers from the main menu.

```
    Device Drivers  --->
    File systems  --->
    Kernel hacking  --->
    Security options  --->
...
...
```

- Select Debug Filesystem Support from the menu.

```
[ ] Strip assembler-generated symbols during link
[ ] Enable unused/obsolete exported symbols
[*] Debug Filesystem
[ ] Run 'make headers_check' when building vmlinux
...
...
```

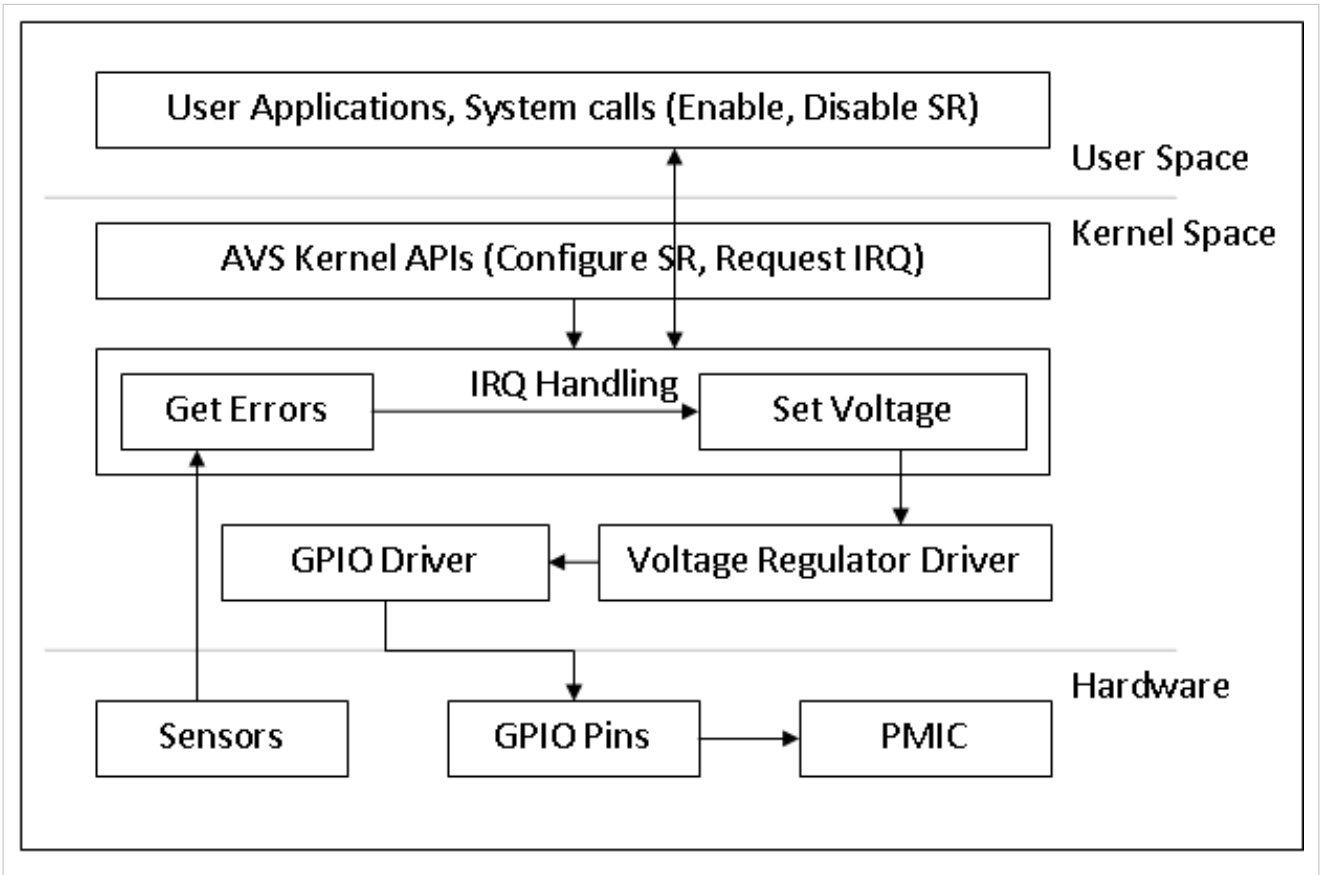
- After doing driver selection, exit and save the kernel configuration when prompted.

Kernel Building

- Once the configuration done according to your requirement then build the kernel by referring *Compiling Linux Kernel* part of PSP User Guide

SmartReflex Driver Architecture

SmartReflex sensors output some nCounts depends upon the process, voltage and temperature of the device. SmartReflex module compute errors based on the nCounts from two sensors, these errors being used for changing the voltage of the device. As shown in the below figure, Linux SmartReflex driver architecture is mainly divided into three layers, user space, Kernel space and hardware.



User Space

User space mainly provides the debug-fs entries of the SmartReflex driver, these can be used for debugging the SmartReflex driver.

Kernel Space

This layer mainly consisting of the core SR driver, voltage regulator driver and gpio driver. SmartReflex driver generally do the module configuration, enable IRQs for both the sensors (HVT and SVT), reading the errors from two sensors, requesting/changing the voltage

During SmartReflex driver initialization SR module get the voltage regulator by passing the voltage domain name (should be same as supply_name member of regulator init data). Then SmartReflex driver uses regulator calls for requesting the current operating voltage and for setting the new operating voltage.

Voltage regulator driver intern uses the GPIO driver API's for changing/requesting the gpio value.

Files

```
        SR driver core file: arch/arm/mach-omap2/smartreflex-ti816x.c
  SR Platform data addition: arch/arm/mach-omap2/devices.c
        VR driver core file: drivers/regulator/gpio-regulator.c
  VR Platform data addition: arch/arm/mach-omap2/board-ti8168evm.c
```

Hardware

This layer mainly consisting of HVT & SVT sensors, GPIO pins and PMIC. Sensors are responsible for monitoring the device characteristics. For voltage control DM816X uses Bank-1 GPIO pins 0, 1, 2 and 3. Power Management IC is feeding the required voltage to the device based on the GPIO output.

Features

The DM816X AVS driver supports following features

- Supports the PMIC present on the DM816X EVM(TPS40041)
- Supports both HVT and SVT sensors
- Voltage control over GPIO lines

Debugging AVS driver

SmartReflex driver provides some debugfs entries to aid debugging. Prior to that mount the debug file system into a directory

```
$ mount -t debugfs debugfs /sys/kernel/debug/
```

To know the current nTarget values for HVT and SVT sensors use:

```
$ cat /sys/kernel/debug/smartreflex/sr_hvt/nvalue
$ cat /sys/kernel/debug/smartreflex/sr_svt/nvalue
```

For enabling the SR driver from debugfs use:

```
$ echo 1 > /sys/kernel/debug/smartreflex/autocomp
```

For disabling the SR driver from debugfs use:

```
$ echo 0 > /sys/kernel/debug/smartreflex/autocomp
```

User can also configure the period for which SR interrupts are kept disabled. This is achieved by:

```
$ echo <delay in msec> > /sys/kernel/debug/smartreflex/interrupt_delay
```

By default delay is set to 2000msec.

Note: Please be aware that writing too small value to this can cause high interrupt load on the system.

This value tells us the initial voltage of the AVS voltage domain.

```
$ cat /sys/kernel/debug/smartreflex/initial_voltage
```

To know the current voltage feeding to AVS voltage domain, use:

```
$ cat /sys/kernel/debug/smartreflex/current_voltage
```

Error to voltage gain helps in converting the percentage error (between the blown nTarget value and the current nTarget value reading by the SR sensor) to voltage delta. Err2VoltGain value is specific to each sensor

```
$ cat /sys/kernel/debug/smartreflex/sr_hvt/err2voltgain
$ cat /sys/kernel/debug/smartreflex/sr_svt/err2voltgain
```

Supporting AVS with a different PMIC

Note: This section is only meant for controlling the vdd_avs voltage from SmartReflex driver

1. Implement voltage regulator for chosen PMIC and provide the generic calls to the SmartReflex driver

```
Like, regulator_get(), regulator_get_voltage(), regulator_set_voltage(). If needed provide the enable and disable hook ups.
```

For reference go through GPIO based voltage regulator driver at "drivers/regulator/gpio-regulator.c". Current implementation of SmartReflex driver uses this GPIO based voltage regulator

2. Add Voltage Regulator platform specific data to the board file at "arch/arm/mach-omap2/board-ti8168evm.c". Regulator platform device need to be registered at board initialization, so that other modules can use this driver utilities.

```
supply name is used while SmartReflex driver is requesting the regulator
```

3. Add SR platform specific data based on the chosen PMIC to the devices file at "arch/arm/mach-omap2/devices.c", so that SR driver request the same regulator

These fields **must** change according to the PMIC

```
a. voltage domain name, which is registered as a supply name in voltage regulator driver
.vd_name = "vdd_avs",
```

```
b. Step size of the PMIC, which is used to calculate the voltage delta
.vstep_size_uv = 15000, Ex: 15mV, which is a default one for GPIO voltage regulator.
```

Based on voltage step size, modify err_minlimit & e2v_gain fields in **sr_sensor_data** structure. Below table list all the changing parameters with step size

HVT Sensor Parameters: SR0

Step Size (uV)	1000	2500	3750	4000	5000	6500	7500	10000	12500	15000	16667	50000
e2v_gain	0xC3	0x4E	0x34	0x30	0x27	0x1E	0x1A	0x13	0xF	0xD	0xB	0x3
err_minlimit	0xFD	0xFD	0xFD	0xFD	0xFB	0xFB	0xFB	0xF9	0xF7	0xF6	0xF4	0xD5

SVT Sensor Parameters: SR1

Step Size (uV)	1000	2500	3750	4000	5000	6500	7500	10000	12500	15000	16667	50000
e2v_gain	0x117	0x6F	0x4A	0x45	0x37	0x2A	0x25	0x1B	0x16	0x12	0x10	0x5
err_minlimit	0xFD	0xFD	0xFD	0xFD	0xFC	0xFC	0xFC	0xFA	0xFA	0xF8	0xF7	0xE6

References

[1] http://creativecommons.org/licenses/by-sa/3.0/

Article Sources and Contributors

TI81XX PSP AVS Driver Guide *Source*: http://processors.wiki.ti.com/index.php?oldid=96278 *Contributors*: Parth.saxena, RK

Image Sources, Licenses and Contributors

Image:TIBanner.png *Source*: http://processors.wiki.ti.com/index.php?title=File:TIBanner.png *License*: unknown *Contributors*: Nsnehaprabha

Image:Avs Driver.png *Source*: http://processors.wiki.ti.com/index.php?title=File:Avs_Driver.png *License*: unknown *Contributors*: Chanilkumar

License

b. Subject to the above terms and conditions, the license granted here is perpetual (for the duration of the applicable copyright in the Work). Notwithstanding the above, Licensor reserves the right to release the Work under different license terms or to stop distributing the Work at any time; provided, however that any such election will not serve to withdraw this License (or any other license that has been, or is required to be, granted under the terms of this License), and this License will continue in full force and effect unless terminated as stated above.

8. Miscellaneous

a. Each time You Distribute or Publicly Perform the Work or a Collection, the Licensor offers to the recipient a license to the Work on the same terms and conditions as the license granted to You under this License.
b. Each time You Distribute or Publicly Perform an Adaptation, Licensor offers to the recipient a license to the original Work on the same terms and conditions as the license granted to You under this License.
c. If any provision of this License is invalid or unenforceable under applicable law, it shall not affect the validity or enforceability of the remainder of the terms of this License, and without further action by the parties to this agreement, such provision shall be reformed to the minimum extent necessary to make such provision valid and enforceable.
d. No term or provision of this License shall be deemed waived and no breach consented to unless such waiver or consent shall be in writing and signed by the party to be charged with such waiver or consent.
e. This License constitutes the entire agreement between the parties with respect to the Work licensed here. There are no understandings, agreements or representations with respect to the Work not specified here. Licensor shall not be bound by any additional provisions that may appear in any communication from You. This License may not be modified without the mutual written agreement of the Licensor and You.
f. The rights granted under, and the subject matter referenced, in this License were drafted utilizing the terminology of the Berne Convention for the Protection of Literary and Artistic Works (as amended on September 28, 1979), the Rome Convention of 1961, the WIPO Copyright Treaty of 1996, the WIPO Performances and Phonograms Treaty of 1996 and the Universal Copyright Convention (as revised on July 24, 1971). These rights and subject matter take effect in the relevant jurisdiction in which the License terms are sought to be enforced according to the corresponding provisions of the implementation of those treaty provisions in the applicable national law. If the standard suite of rights granted under applicable copyright law includes additional rights not granted under this License, such additional rights are deemed to be included in the License; this License is not intended to restrict the license of any rights under applicable law.