DM8168 AVS PMBus Driver User Guide

- **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.

PMBus specific driver support is required for PMIC’s such as TP40400. PMBus is an open standard protocol that defines a means of communicating with power conversion and other devices. It is a communications protocol based on I2C. Hence it is just a specification or a wrapper over I2C.

TPS40400 is a buck controller and allows programming and monitoring via the PMBus interface. The TPS40400 supports a subset of the commands in the PMBus 1.1 specification.

Acronyms & definitions

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
PMBus	Power Management Bus

Driver Configuration

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

Voltage Regulator Driver configuration

The default kernel configuration enables support for TPS40400 PMBus voltage regulator Driver (built into the kernel).

To enable or disable TPS40400 PMBus 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 TI TPS40400 PMBus PMIC from the menu

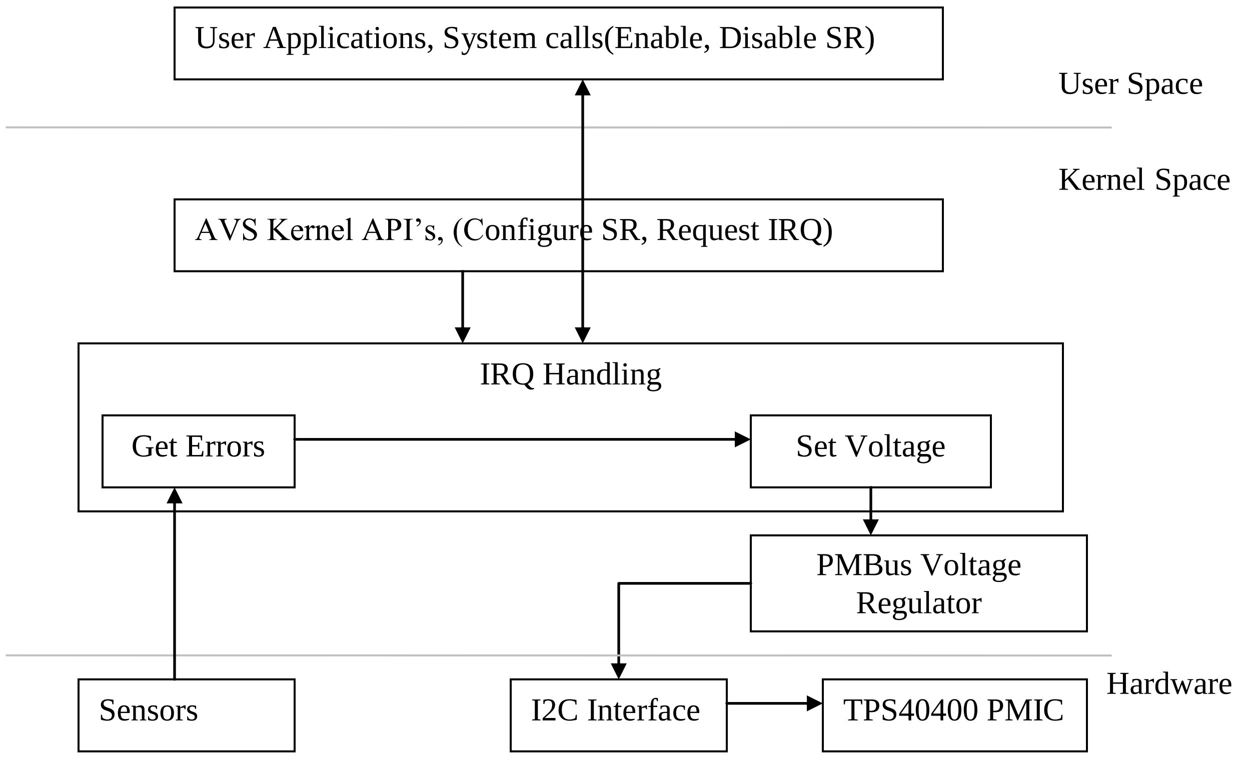
```
< > National Semiconductors LP3972 PMIC regulator driver
<*> TI TPS40400 PMBus PMIC
<*> GPIO voltage regulator
< > TI TPS6507X Power regulators
< > Intersil ISL6271A Power regulator
...
...
```

- 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-PMBus Driver Architecture



Features

The AVS driver supports following features

- Supports the PMIC TPS40400
- Supports both HVT and SVT sensors
- Voltage control over PMBus-I2C lines

Supported PMBus Commands

The voltage regulator driver which has been developed for AVS supports the following PMBus commands only required for the this purpose:

- **PMBUS_OPERATION**

  The OPERATION command is used to turn the device output on or off. It is also used to set the output voltage to the upper or lower MARGIN voltages.

- **PMBUS_CLEAR_FAULTS**

  The CLEAR_FAULTS command is used to clear any fault bits that have been set.

- **PMBUS_VOUT_MODE**

  It is a byte that consists of a 3-bit Mode and 5-bit exponent parameter, as shown below. The 3-bit Mode sets whether the device uses the Linear or Direct modes for output voltage related commands. The 5-bit parameter sets the exponent value for the linear data mode. It is a read only based command.

- **PMBUS_VOUT_TRIM**

  The VOUT_TRIM command is used to apply a fixed offset voltage to the output voltage command value.

- **PMBUS_VOUT_CAL_OFFSET**

  This command applies an offset to the READ_VOUT command results to calibrate out offset errors in the on board measurement system.

- **PMBUS_VOUT_CAL_GAIN**

  This command applies a gain correction to the READ_VOUT command results to calibrate out gain errors in the on board measurement system.

- **PMBUS_VOUT_MARGIN_HIGH**

  The VOUT_MARGIN_HIGH command sets the target voltage which the output changes to when the OPERATION command is set to "Margin High".

- **PMBUS_VOUT_MARGIN_LOW**

The VOUT_MARGIN_LOW command sets the target voltage which the output changes to when the OPERATION command is set to "Margin Low"

- **PMBUS_VOUT_SCALE_LOOP**

VOUT_SCALE_LOOP is equal to the feedback resistor ratio. The nominal output voltage is set by a resistor divider and the internal 600mV reference voltage.

- **PMBUS_READ_VOUT**

The READ_VOUT commands returns data that represents the output voltage of the controller.

- **PMBUS_REVISION**

The PMBUS_REVISION command returns data that indicates that the TPS40400 is compliant with the 1.1 revision of the PMBus specification.

Using this Driver

Type the below commands to enable/disable the driver:

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

Enable: `$ echo 1 > /sys/kernel/debug/smartreflex/autocomp`
Disable: `$ echo 0 > /sys/kernel/debug/smartreflex/autocomp`

Additions made to the Kernel to Support AVS with a PMBus based PMIC

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

1. Implement voltage regulator for chosen PMBus based 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 voltage regulator driver at "drivers/regulator/tps40400-regulator.c". Current implementation of SmartReflex driver uses this PMBus based voltage regulator

2. Add Voltage Regulator platform specific data to the board file at "arch/arm/mach-omap2/board-ti8168evm.c".

Change the pmbus_pmic_init_data for the max and min voltage defaults specific to the PMIC.

Change the PMBus-I2C address which is specific to the PMIC. Here

“I2C_BOARD_INFO("pmbus", 0x00)”

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

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 = 15000, Ex: 15mV, which is a default one for voltage regulator.

4. Based on voltage step size, modify err_minlimit & e2v_gain fields in ti816x_sr_sdata structure.

Table based on step size

5. Based on the platform change the nominal voltage value & the exponent for PMIC here in this file "drivers/regulator/tps40400-regulator.c"

```
pmic->exponent = -10; /*Set this as per default value for PMIC(TPS40400)*/

pmic->nominal_uV = 1000000; /* For DM8168 - nominal voltage is 1V)*/
```

Voltage Control Logic and Calculations

- **Setting the VOUT_SCALE_LOOP(command code 29h)**

This setting is necessary for the correct calculation of the VOUT_TRIM, VOUT_MARGIN_HIGH and VOUT_MARGIN_LOW.

i.) The feedback resistors in the schematic of the DVR shared are 15k and 10k. The nominal reference (VFB) is 600 mV. The expected output voltage is 10k*0.6/15k = 1.0V.
ii.) VOUT_SCALE_LOOP is defined as VFB/VOUT(nom) = 0.6
iii.) Converting 0.6 to the linear format (Mantissa * 2^ Exponent)

- the exponent is fixed by design at -9 decimal or 10111 binary.
- the mantissa is calculated as 0.6/(2^-9) = 0.6 *2^9 = 0.6 * 512 = 307.2 or 00100110011 as an unsigned binary integer

iv.) The resultant conjugation is 1011 1001 0011 0011 b or B933 h.

- **Checking and setting the offset errors for RD_VOUT_CAL_OFFSET**

READ_VOUT calibration example: On initial system startup, the output is precisely trimmed to be 1.0 volts. Issuing a PMBus read on the READ_VOUT command returns 488h indicating that the output voltage reading is 1.132 volts so you decide that calibration is in order. (This is exaggerated for illustration)

i) Using the VOUT_TRIM command, you set the output voltage to multiple values and get the following table:

Actual Vout	READ_VOUT mantissa	READ_VOUT
0.80 volts	3A8h	0.914 volts
1.00 volts	430h	1.132 volts
1.20 volts	450h	1.352 volts

ii) Using a least squares method, you can calculate a gain error of 9.4% and an offset of 39 mV so you set RD_VOUT_CAL_GAIN to C7E8h = -24*2^-8 = -9.375% and RD_VOUT_CAL_OFFSET to FFD9h = -39*2^-10.

- **Understanding the READ_VOUT Command**

The accuracy of this output is greatly influenced by the quality of the PCB layout and the precision of the calibration using RD_VOUT_CAL_GAIN and RD_VOUT_CAL_OFFSET).

i) For example, if issuing a READ_VOUT command returns 480h, this would indicate that the output voltage is 480h * 2^-10 = 1152/1024 = 1.125 volts

- **Understanding the VOUT_TRIM Command** `(command code 22h)`

This command allows the PMBus host to adjust or trim the output voltage. If you use this command to correct for an offset within your system for absolute accuracy AND use this command to implement DVS, the host will need to account for the fixed offset.

i.) If Vout(nominal) is 1 volt, the range of acceptable values for VOUT_TRIM would be -250 mV to 250 mV.

ii.) Suppose that you choose to increase the output voltage by 100 mV:

1.) First you would read the VOUT_TRIM register to see its current setting. Suppose the system has been calibrated and an increase of 8 mV was needed to have a true output of 1.0 V. You would read 8h back from the VOUT_TRIM register.

2.) To get the 100 mV increase, you would write 6Fh to VOUT_TRIM. (0.108/2^-10 = 6Fh)