Smart Reflex Class0 Temperature Compensation

User's Guide

Revision 1.1

February 4th, 2015

Revision Record	
Document Title: **User’s Guide**	
Revision	Description of Change
1.0	Initial Release
1.1	Add SRSS VNTL interface support

TABLE OF CONTENTS

TABLE OF FIGURES

1 Scope

This document defines the Smart Reflex Class0 Temperature Compensation feature implemented on TI Keystone II devices (K2L and K2E).

2 Definitions

Acronym	Description
API	Application Programming Interface
BMC	Board Management Controller
DSP	Digital Signal Processor
PMBus	Power Management Bus
SMPS	Switch Mode Power Supply
SOC	System On Chip (Keystone II devices)
SRSS	Smart Reflex Sub-System
SRSS C0 TC	Smart Reflex Sub-System Class 0 Temperature Compensation
VID	TI Voltage ID
VPRM	Voltage Processor Request Merger IP

Table 1. Definitions

3 Board/BMC modification

The older version of the board always connects the SRSS VCNTL bus to TA via LM10011. A 0-ohm resistor that shorts LM10011 EN pin and VCC3V3_AUX needs to be replaced with a 10k-ohm resistor to allow BMC to either enable or disable LM10011. This problem will be fixed in the next version of the K2L and K2E EVM.

3.1 PMBus Interface mode

BMC needs to be changed to connect SoC I2C bus to external power controller (E.g. Top Avatar device) and disconnect SoC VCNTL bus to external power controller in all the boot modes except DSP/ARM no-boot mode. For testing purpose, customers can use the below BMC commands.

For K2L EVM, I2C port 2 can be connected to TA I2C bus:

```
hwdbg cmd show
gpio d xx0x_xxxx
gpio e xx0x_xxxx
```

```
 gpio f xxxx_xxx1
 gpio c x0xx_xxxx
```

For K2E EVM, I2C port 1 can be connected to TA I2C bus:

```
 hwdbg cmd show
 gpio xd xxxx_xxxx_xxxx_1xxx
 gpio c x0xx_xxxx
```

3.1 SRSS VCNTL Interface mode

The existing BMC always connects the SRSS VCNTL bus to TA via LM10011, there is no need to change the board or default BMC configurations.

4 Software Implementation

4.1 SRSS Registers

The base address of the SRSS IP in Keystone II devices is 0x0233_0000, below are the offset of the SRSS registers

```
/* Smart Reflex Sub-System register offset */
#define SRSS_PID             0x0
#define SRSS_INTR_EN_SET     0x4
#define SRSS_EVENT_FLAG      0x8
#define SRSS_EVENT_CLR       0xc
#define SRSS_TEMP_CTL0       0x1c
#define SRSS_TEMP_CTL1       0x20
#define SRSS_TEMP_CTL2       0x24
#define SRSS_TEMP_STAT0      0x28
#define SRSS_TEMP_STAT1      0x2c
#define SRSS_VPRM_CTL0       0x80
#define SRSS_VPRM_CTL1       0x84
#define SRSS_VPRM_STAT0      0x88
#define SRSS_VPRM_STAT1      0x8c
```

The `SRSS_TEMP_CTLx` and `SRSS_TEMP_STATx` registers are very useful to check the temperature and VID information, user can use the below U-boot "md" command to dump the SRSS registers for debugging purpose.

Please refer to the SRSS IP UG (TBD) for detailed description of the registers.

4.2 U-boot

U-boot initializes SRSS IP and enables C0 with TC. U-boot does not do a close-loop to handle TC. The initial temperature threshold points, voltage step, hysteresis and threshold points are preprogrammed to the e-fuse registers by production, if these initial settings are not programmed, U-boot will disable SRSS C0 with TC:

U-boot will do the following configurations to enable SRSS C0 with TC:

- VPRM_EN set to '01' (enables SRSS C0) in VPRM_CTRL0_REG
- SMPS_MODE set to '10' (enable VID 6-bit interface) in VPRM_CTRL0_REG
- srss_c0temp_en set to '1' (enables TC) in SRSS_TEMP_CTL0_REG
- srss_class0_i2c_req_en to '0' (disables SRSS I2C) in SRSS_TEMP_CTL0_REG

4.2.1 SRSS C0 with TC over a generic I2C interface

If SoC's Bit [4:1] of DEVSTAT register are not all zeros (non-sleep mode), BMC needs to connect one of the SoC's generic I2C interfaces to Top Avatar's I2C interface.

For K2L EVM, SoC I2C bus 2 (0 based) is connected. For K2E EVM, SoC I2C bus 1 (0 based) is connected.

U-boot will also need to initialize Top Avatar by sending the following PMBus commands via a generic I2C interface to turn on the Top Avatar, the existing U-boot API do_i2c_write_block() in common/cmd_i2c.c can be used to send the PMBus commands:

- ON_OFF_CONFIG (command 02h, to turn on Top Avatar)
- VOUT_OV_FAULT_LIMIT (command 40h, to set the value of the output voltage that causes an output overvoltage fault)
- VOUT_OV_WARN_LIMIT (command 42h, to set the value of the output voltage that causes an output over voltage warning condition)
- VOUT_COMMAND (command 21h, to set the initial output voltage in volts)

Below is the software call flow diagram:

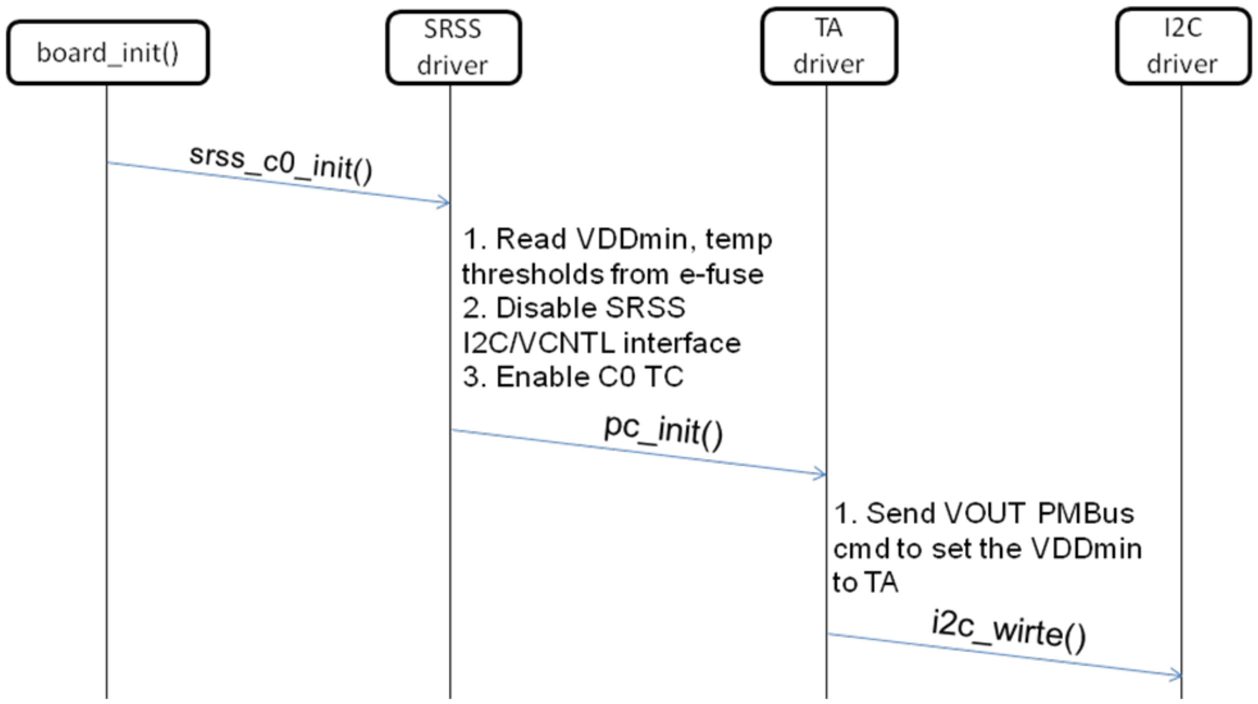
board_init()
SRSS driver
TA driver
I2C driver
srss_c0_init()
1. Read VDDmin, temp thresholds from e-fuse
2. Disable SRSS I2C/VCNTL interface
3. Enable C0 TC
pc_init()
1. Send VOUT PMBus cmd to set the VDDmin to TA
i2c_wirte()

Figure 1 Boot-time software call flow in PMBus mode

4.2.2 SRSS C0 with TC over VCNTL interface

SRSS IP will send the VDDmin and updated VID to external power controller via SRSS VCNTL interface, software only needs to enable VCNTL and TC.

Below is the software call flow diagram:

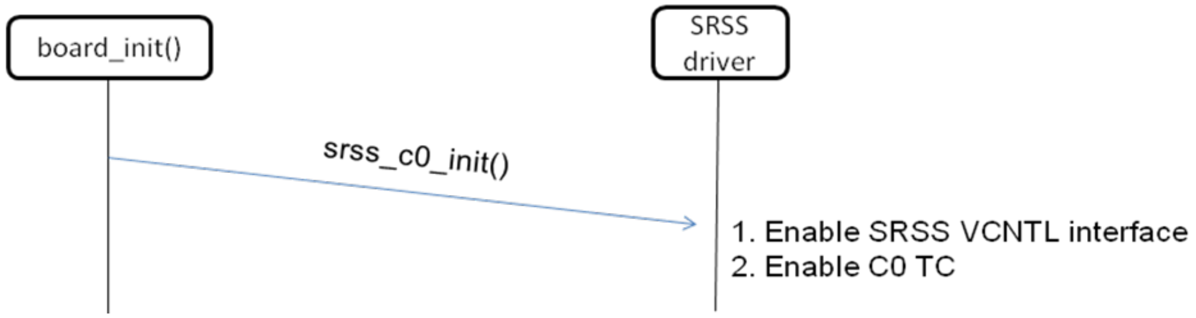

Figure 2 Boot-time software call flow in VCNTL interface mode

4.2.3 Debug commands

The following U-boot command will be used for testing purpose:

- memory display command (md) can be used to display the SRSS registers for initial temperature threshold points, voltage step change size, temperature threshold hysteresis, etc.

4.3 Linux Implementation

A new user space srss_tc application will be created to handle the SRSS C0 temperature threshold point interrupt. It sends Vout PMBus command to Top Avatar via a generic I2C interface.

4.3.1 Linux Device Tree

The user space srssc0 application will receive SRSS interrupt via Linux kernel's UIO framework, the following device tree configuration will be added to arch/arm/boot/dts/<k2l|k2e>-evm.dts to add the uio_srss device tree binding:

```
uio_srss: srss {
      compatible = "ti,uio-module-drv";
      mem  = <0x02330000 0x400>;
      clocks = <&clksr>;
      interrupts = <0 0x173 0xf01>;
      label = "srss";
};
```

4.3.2 User space application

User space application is only required in PMBus interface mode.

4.3.2.1 UIO SRSS device

The user mode application will search through /sys/class/uio and open the uio device for SRSS. It will do the following to handle the SRSS interrupt:

1. Enable the SRSS temperature threshold point interrupt
2. Use read() API to wait for the SRSS temperature threshold point interrupt.
3. Once it receives the interrupt, it will read the new VID value in SRSS_TEMP_STAT1_REG[29:24]
4. Convert the VID to the Absolute voltage value in Vout command format, and send the Vout to the Top Avatar
5. Clear the SRSS temperature threshold point interrupt flag in SRSS_EVENT_CLR_REG and go to step 2.

Note:

1. VID is the 6-bit voltage data from voltage processors. Each of the VID value has a one-to-one mapping to an absolute voltage value (total 64 mappings), below is a table of VID to Vout PMBus format voltage conversion array defined in the U-boot/User space app source:

```
uint8_t vout_pmbus[64][2] = {
     {0x66, 0x1}, {0x69, 0x1}, {0x6d, 0x1}, {0x70, 0x1},
     {0x73, 0x1}, {0x76, 0x1}, {0x79, 0x1}, {0x7d, 0x1},
     {0x80, 0x1}, {0x84, 0x1}, {0x87, 0x1}, {0x8a, 0x1},
     {0x8d, 0x1}, {0x90, 0x1}, {0x94, 0x1}, {0x97, 0x1},
     {0x9a, 0x1}, {0x9e, 0x1}, {0xa1, 0x1}, {0xa4, 0x1},
     {0xa7, 0x1}, {0xab, 0x1}, {0xae, 0x1}, {0xb1, 0x1},
     {0xb5, 0x1}, {0xb8, 0x1}, {0xbb, 0x1}, {0xbe, 0x1},
     {0xc2, 0x1}, {0xc5, 0x1}, {0xc8, 0x1}, {0xcb, 0x1},
     {0xcf, 0x1}, {0xd2, 0x1}, {0xd6, 0x1}, {0xd9, 0x1},
     {0xdc, 0x1}, {0xdf, 0x1}, {0xe2, 0x1}, {0xe6, 0x1},
     {0xe9, 0x1}, {0xec, 0x1}, {0xf0, 0x1}, {0xf3, 0x1},
     {0xf6, 0x1}, {0xf9, 0x1}, {0xfc, 0x1}, {0x00, 0x2},
     {0x03, 0x2}, {0x07, 0x2}, {0x0a, 0x2}, {0x0d, 0x2},
     {0x10, 0x2}, {0x13, 0x2}, {0x17, 0x2}, {0x1a, 0x2},
     {0x1d, 0x2}, {0x21, 0x2}, {0x24, 0x2}, {0x27, 0x2},
     {0x2b, 0x2}, {0x2e, 0x2}, {0x31, 0x2}, {0x34, 0x2}
};
```

Each vout_pmbus[] has two 8-bit data, the absolute voltage is decoded in below formula:

absolute voltage = (vout_pmbus[x][0] + vout_pmbus[x][1] * 256)/512 (Volt), where x is the VID

E.g. for VID = 0, vout_pmbus[0][0] = 0x66, vout_pmbus[0][1] = 0x1,

absolute voltage = (0x66 + 0x1 * 256) / 512 = 0.699 (Volt)

4.3.2.2 I2C device

The user space srssc0 application will use Linux kernel’s SMBus/I2C device interface to communicate with the slave I2C devices. The application needs to open the device file for the I2C adapter and set the Top Avatar slave I2C address (0x09) to the device file.

The application will use i2c_smbus_write_i2c_block_data API to send the VOUT_COMMAND PMBus command to the Top Avatar, below is a list of SMBus API’s:

```
i2c_smbus_write_quick( int file, __u8 value)
i2c_smbus_read_byte(int file)
i2c_smbus_write_byte(int file, __u8 value)
i2c_smbus_read_byte_data(int file, __u8 command)
i2c_smbus_write_byte_data(int file, __u8 command, __u8 value)
i2c_smbus_read_word_data(int file, __u8 command)
i2c_smbus_write_word_data(int file, __u8 command, __u16 value)
i2c_smbus_process_call(int file, __u8 command, __u16 value)
i2c_smbus_read_block_data(int file, __u8 command, __u8 *values)
i2c_smbus_write_block_data(int file, __u8 command, __u8 length, __u8 *values)
i2c_smbus_read_i2c_block_data(int file, __u8 command, __u8 *values)
i2c_smbus_write_i2c_block_data(int file, __u8 command, __u8 length, __u8
*values)
i2c_smbus_block_process_call(int file, __u8 command, __u8 length, __u8
*values)
```

4.3.2.3 Debug options

A debug option will be added to enable the printout on the serial console and logging of SRSS register setting, temperature change, interrupt events and error status. The log file is located in /var/log/srss.log.

4.3.2.4 Run-time software call flow

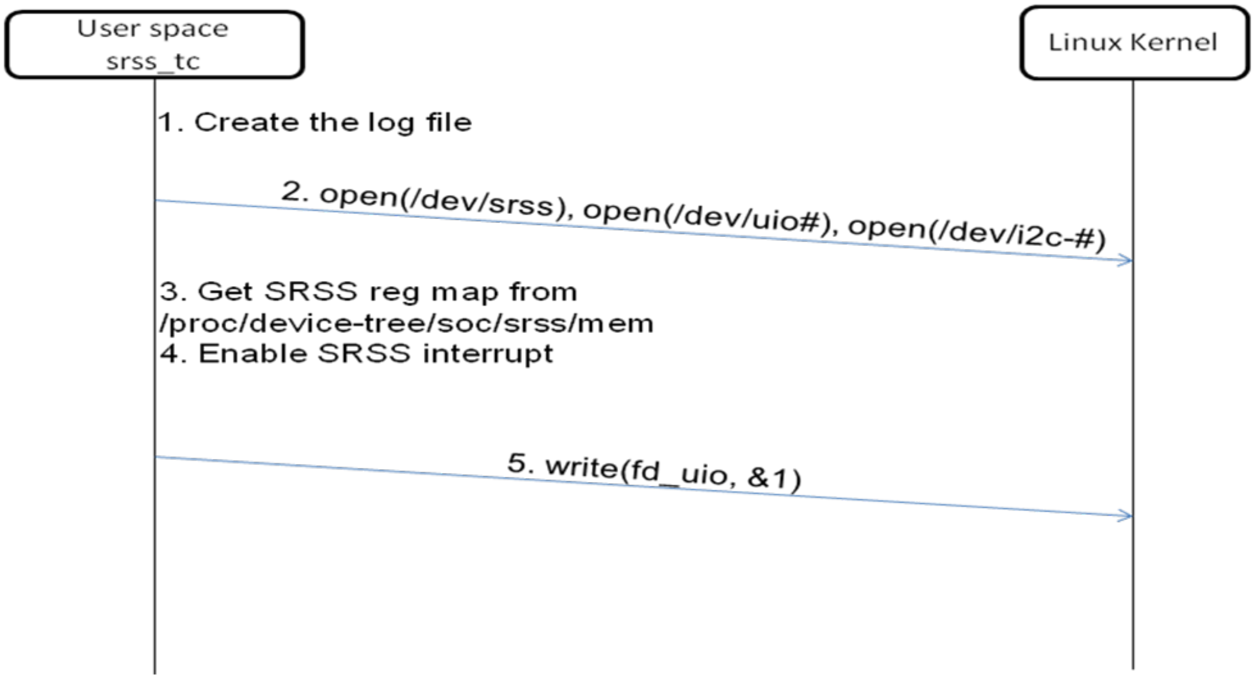

Figure 3 User space run-time software setup

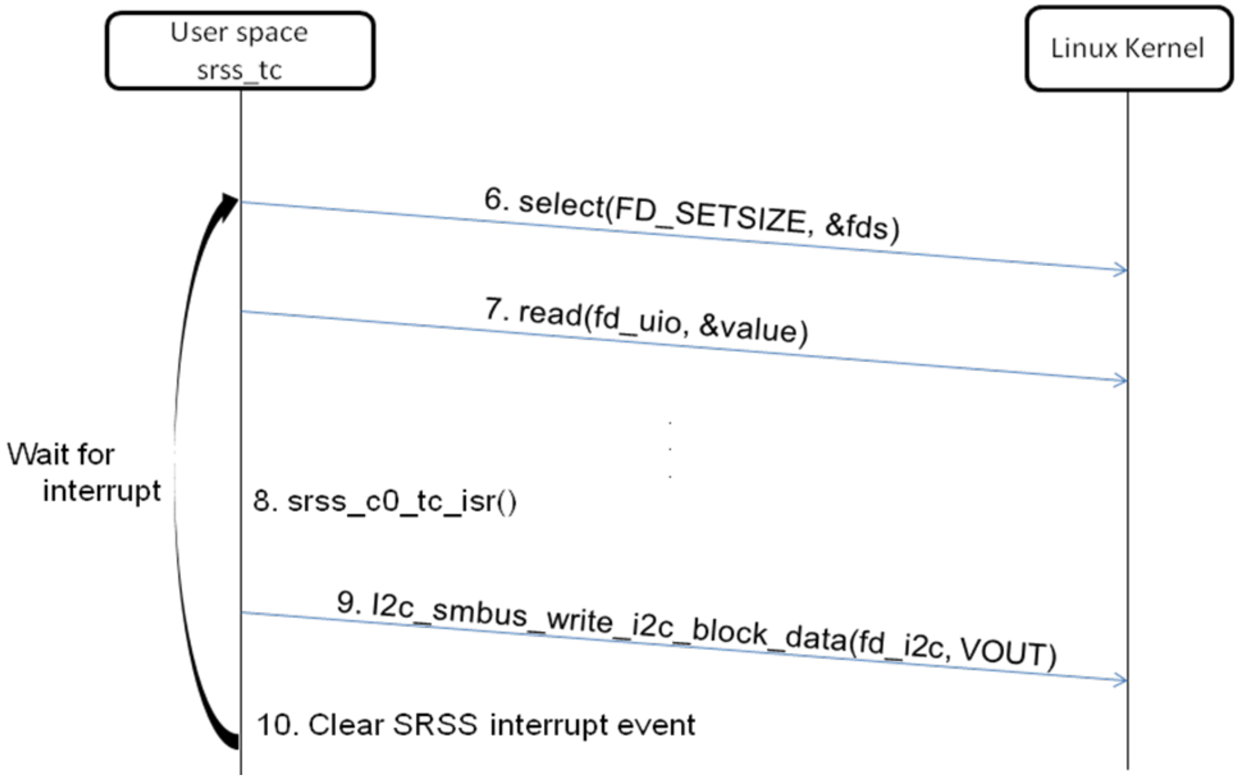
User space
srss_tc
Linux Kernel
6. select(FD_SETSIZE, &fds)
7. read(fd_uio, &value)
Wait for interrupt
8. srss_c0_tc_isr()
9. I2c_smbus_write_i2c_block_data(fd_i2c, VOUT)
10. Clear SRSS interrupt event

Figure 4 User space run-time temperature interrupt handling

5 Testing Considerations

5.1 U-boot

5.1.1 Temperature with Compensation

U-boot always enables Smart Reflex Class 0 with temperature compensation via PMBus interface by default. U-boot will print out the below log on serial console when TC is enabled:

```
Reseting entire DDR3 memory to 0 ...
DRAM:  2 GiB
NAND:  2048 MiB
Smart Reflex Class 0 temperature compensation enabled
Net:   K2L_EMAC0
```

Customers can use the below U-boot "md" command to dump the SRSS registers for debugging purpose:

```
K2L EVM # md 0x2330000
02330000: 43004000 00000000 00020000 00000000    .@.C............
02330010: 00000000 00000000 00000000 f8005547    ............GU..
```

```
02330020: 5c504030 00000000 312d312a 3431f728    0@P\....*1-1(.14
02330030: 00003210 00000000 00000000 00000000    .2..............
02330040: 0000040f 00000000 000000ee 000000ee    ................
02330050: 000000ee 000000ee 000000ee 000000ee    ................
02330060: 000000ee 000000ee 000000ee 000000ee    ................
02330070: 000000ee 000000ee 000000ee 000000ee    ................
02330080: fff00009 00000000 00000001 00000000    ................
02330090: 00000000 00000000 00000000 00000000    ................
023300a0: 00000000 00000000 00000000 00000000    ................
023300b0: 00000000 00000000 00000000 00000000    ................
023300c0: 00000000 00000000 00000000 00000000    ................
023300d0: 00000000 00000000 00000000 00000000    ................
023300e0: 00000000 00000000 00000000 00000000    ................
023300f0: 00000000 00000000 00000000 00000000    ................
```

5.1.1.1 PMBus interface mode

The temperature threshold point values, VID change step size, temperature monitor/TC enable bits are set in `SRSS_TEMP_CTLx` registers. The SoC temperatures captured by SRSS temperature monitors are stored in `SRSS_TEMP_STAT0` register (Bits[7:0] and Bits[23:16]), and the initial VID value set to the external power controller is store in `SRSS_TEMP_STAT1` register (Bits[29:24]). Customers can also use the below U-boot command to read back Vout value from the external power controller to check if it matches with the Vout written:

```
i2c dev 2 (for K2L EVM)
i2c dev 1 (for K2E EVM)
i2c md 09 8b.1 2
```

Note: There is always a difference between the actual Vout read and Vout write values, if the difference is less than 3, it is in the acceptable range.

5.1.1.2 SRSS VCNTL interface mode

SRSS VCNTL interface by default is disabled. To enable SRSS VCNTL interface, customers can set the below environment variable:

```
 setenv srss_vcntl 1
 saveenv
 reset
```

Note: Temperature compensation cannot be verified on existing version of TI's EVM due to board related problem, and some devices on TI's EVM does not have SRSS e-fuse registers programmed. But Customers can measure the Vout on the board to check if it matches with the initial VID value.

5.1.2 Temperature without Compensation

To disable temperature compensation, customers can set the below environment variable:

```
 setenv no_src0_tc 1
 saveenv
 reset
```

If temperature compensation is disabled, U-boot will still send the initial VID voltage to the external power controller via I2C/PMBus.

5.1.3 Temperature without Compensation

To disable temperature compensation, customers can set the below environment variable:

```
 setenv no_src0_tc 1
 saveenv
 reset
```

5.2 Linux user space app

The user space application srss_tc.out is located under /usr/bin directory in the tisdk rootfs. The init.d script will load/run srss_tc.out automatically when rootfs boots up. To enable debug information printout and logging, customers can use the below command:

```
root@k2hk-evm:~# srss_tc.out --help
`srss_tc.out' handles Smart Reflex class 0 temperature
compensation

 Usage: /srss_tc.out [OPTION [ARG]] ...
 -?, --help         show this help statement
     --version      show version statement
 -v, --verbose      be verbose
 -i, --i2c N        SoC I2C bus number
 -p, --pc_addr N    external power controller I2C slave device
address
 Example: /srss_tc.out --i2c 2 --pc_addr 9 –verbose
```

Below is the log when temperature threshold point interrupt is handled by srss_tc.out:

```
root@k2l-evm:~# srss_tc.out -v
```

```
SoC I2C bus num: 2
External power controller I2C slave address: 9
 Opened device /dev/srss, fd: 4

 Class name: uio8

 Opened device /dev/uio8, fd: 5

 Opened device /dev/i2c-2, fd: 6

 SRSS Event flag reg 0x0

 waiting for interrupt...


SRSS Event flag reg 0x20000
SRSS temp stat0 0x31343132, stat1 0x32412f32.

 Vout write: a 2, vid: 0x32

 interrupt processing time 18348127 ns
 waiting for interrupt...

 Vout read: b 2

 PMBus Vout write value matches with Vout read value!
```

5.3 Performance

5.3.1 Run-time Interrupt latency

The interrupt response time (time between SRSS detects a temperature threshold point interrupt and the user mode application sets a new voltage to the external power supply controller) should be less than **100 msec** at all time regardless the CPU load.

The below delay was measured during the test:

- With normal CPU load the response time is < **30 usec**
- With 100% CPU load the response time is < **2 msec**
    - Used an open source stress test utility to generate the workload (http://freecode.com/projects/stress)
    - Ran 4 stress test threads simultaneously spinning on memory malloc and free
- measured with printf debug option enabled, time between srss uio interrupt received and when the new VID value is confirmed to be written to the TA (read back after write)

5.3.2 Boot-time interrupt latency

There is a new requirement that when TC is enabled during boot time, software may need to handle back-to-back temperature change interrupts (<100usec). To work around this problem, U-boot needs to send an updated VID instead of VDDmin before booting the Linux kernel.

6 Open Issues

- Currently SRSS C0 with TC is only supported on K2L and K2E EVM’s, customers can port the U-boot, Linux device tree and user space code to support on K2HK EVM as needed:
  - U-boot: add SRSS C0 configurations in `include/configs/k2hk_evm.h`

```
/* SRSS configuration */
#define CONFIG_SRSS_C0
#define CONFIG_SRSS_I2C_BUS       ##    /* SoC I2C bus used to
                                           communicate with TA */
#define CONFIG_PC_I2C_SLAVE       ##    /* External power controller
                                           I2C slave address */
```

  - Linux device tree: add uio_srss device tree binding in `arch/arm/boot/dts/k2hk-evm.dts`:

```
uio_srss: srss {
       compatible = "ti,uio-module-drv";
       mem  = <0x02330000 0x0000400>;
       clocks=<&clksr>;
       interrupts = <0 0x173 0xf01>;
       label = "srss";
};
```

  - User space srss_tc script: modify k2hk configurations to set the SoC I2C bus # and external power controller I2C slave address in `scripts/run_srss_tc.sh`