

Semtech SC493 Software User's Guide

1 Purpose

This document describes the software Application Programming Interface (API) for the Semtech SC493 and describes how to integrate the reference drivers into your application. This document serves to aid the software developer in interfacing to the SC493.

2 Overview

The SC493 contains registers that provide control of output power and status. For additional details, see the Semtech SC493 datasheet.

All names specific to the drivers are shown in the `Courier New` font.

3 Application Programming Interface

The SC493 is provided with reference drivers facilitating complete control of the device. A low-level interface provides the means to directly write registers. A high-level interface allows control of features and is abstracted from the register-level accesses required to enact the features.

3.1 General Interface

3.1.1 WriteRegister

The `WriteRegister` function is the lowest level API provided. All functionality of the chip may be accessed through this single function which writes data to the specified register. The function verifies that the register is within the valid range before attempting the write. In addition, all reserved bits are masked off before the data is transferred.

Prototype

```
Semtech_StatusCode WriteRegister(SC493_Register address,
    unsigned char data)
```

Parameters

<code>address</code>	The register to write to. This must be one of the following values: <code>CONTROL_1</code> <code>CONTROL_2</code> <code>CONTROL_3</code> <code>CONTROL_4</code> <code>CONTROL_6</code>
<code>Data</code>	The data to be written to the register.

Return Value

<code>SEMTECH_SUCCESS</code>	The data has been written to the register of the SC493.
<code>SEMTECH_INVALID_ADDRESS</code>	The register is outside of the valid range and the data will not be sent to the SC493.

3.1.2 ReadRegister

The `ReadRegister` function is the corresponding read function. Any of the available registers can be read using this function. The `STATUS` register may be the most useful and values for bit masking are provided (see the example). To read single status bits, the `GetStatus` function may be more convenient.

Prototype

```
unsigned char ReadRegister(SC493_Register address)
```

Parameters

address	The register to write to. This must be one of the following values: STATUS CONTROL_1 CONTROL_2 CONTROL_3 CONTROL_4 CONTROL_6
---------	--

Return Value

An unsigned char containing the current value of the given register. See the SC493 Datasheet for more details on these registers.

Example

```
unsigned char r = ReadRegister(STATUS);
if( r & STATUS_FLAG_OVER_TEMPERATURE ) {
    // Over Temperature flag is set
}
```

3.1.3 ResetAllRegisters

The `ResetAllRegisters` function sets all of the registers of the SC493 back to the initial power up state. The reset state of the registers is as follows.

Address	Register Name	Reset Value
0x01	Control Register 1	0x01
0x02	Control Register 2	0x20
0x03	Control Register 3	0x2E
0x04	Control Register 4	0x0F
0x06	Control Register 6	0x3B

Prototype

```
void ResetAllRegisters()
```

Parameters

None

Return Value

None

3.1.4 EnableDevice

The `EnableDevice` function pulls the EN pin of the SC493 high.

Prototype

```
void EnableDevice()
```

Parameters

None

Return Value

None

3.1.5 DisableDevice

The `DisableDevice` function pulls the EN pin of the SC493 low. This causes the SC493 to reset all internal registers to their power up values. The ENSW bit should be cleared instead if register values need to be preserved.

Prototype

```
void DisableDevice()
```

Parameters

None

Return Value

None

3.2 Power Control Interface

3.2.1 GetStatus

The `GetStatus` function returns the status or flag at `statusBit`. If multiple `StatusBits` are bitwise ORed together and passed in, this function returns the bitwise OR of each result (see example 2). This feature may be useful to check several or all of the error flags (as in the example) with fewer I²C calls than getting the OR of several `GetStatus()` results. To check multiple bits in a single function call, use the `ReadRegister` function.

Prototype

```
unsigned char GetStatus(unsigned char statusBits)
```

Parameters

<code>statusBit</code>	The status or flag bit to be checked. This parameter must be one of the following: STATUS_POWER_GOOD STATUS_DISCONTINUOUS_MODE STATUS_FLAG_DID_NOT_START STATUS_FLAG_BROWN_OUT STATUS_FLAG_OUTPUT_CURRENT_LIMIT
------------------------	--

	STATUS_FLAG_OVER_TEMPERATURE STATUS_FLAG_OVER_VOLTAGE STATUS_FLAG_UNDER_VOLTAGE
--	---

Return Value

If one bit was checked, the value of that bit is returned (0 or 1). If multiple bits were checked, a 1 is returned if any of those bits had a value of 1, otherwise 0 is returned.

Example 1

```
if ( GetStatus(STATUS_POWER_GOOD) ) {
    // Power is good
}
```

Example 2

```
if ( GetStatus(STATUS_FLAG_OVER_VOLTAGE | STATUS_FLAG_UNDER_VOLTAGE) )
{
    // Power is either under or over voltage
}
```

3.2.2 ClearFlags

The ClearFlags function clears all indicator flags.

Prototype

```
void ClearFlags()
```

Parameters

None

Return Value

None

3.2.3 SetEnableSwitch

The SetEnableSwitch function sets the ENSW bit. If set to low, the output is disabled. When set to high, the output is enabled.

Prototype

```
Semtech_StatusCode SetEnableSwitch(unsigned char state)
```

Parameters

state	The new state for the ENSW bit. 1 turns the output on. 0 turns the output off.
-------	--

Return Value

SEMTECH_SUCCESS	The state was changed.
SEMTECH_NO_CHANGE	The state was already set at the state.
SEMTECH_INVALID_PARAMETER	The specified value for state was out of range.

3.2.4 GetEnableSwitch

The `GetEnableSwitch` function returns the current state of the ENSW pin. A return value of 1 means the device has the enable switch on. A return value of 0 means the device has the enable switch off.

Prototype

```
unsigned char GetEnableSwitch()
```

Parameters

None

Return Value

Returns 1 if the enable switch is high, 0 if it is low.

3.2.5 SetEnableInternalLoad

The `SetEnableInternalLoad` function sets the `EnIntLd` bit. If high, the internal pull down on VOUT is enabled when slewing down.

Prototype

```
Semtech_ResultCode SetEnableInternalLoad(unsigned char state)
```

Parameters

<code>state</code>	The new state for the <code>EnIntLd</code> bit. 1 enables the <code>EnIntLd</code> bit. 0 disables the <code>EnIntLd</code> bit.
--------------------	--

Return Value

<code>SEMTECH_SUCCESS</code>	The state was changed.
<code>SEMTECH_NO_CHANGE</code>	The state was already set at the <code>state</code> .
<code>SEMTECH_INVALID_PARAMETER</code>	The specified value for <code>state</code> was out of range.

3.2.6 GetEnableInternalLoad

The `GetEnableInternalLoad` function returns the current state of the `EnIntLd` pin. A return value of 1 means the internal load is enabled. A return value of 0 means the internal load is disabled.

Prototype

```
unsigned char GetEnableInternalLoad()
```

Parameters

None

Return Value

Returns 1 if the internal load is enabled, 0 if not.

3.2.7 SetFrequency

The `SetFrequency` function sets the registers controlling the main frequency.

Prototype

Semtech_ResultCode SetFrequency(Frequency frequency)

Parameters

frequency	The output frequency, which must be one of the following: FREQUENCY_250_KHZ FREQUENCY_500_KHZ FREQUENCY_750_KHZ FREQUENCY_1_MHZ
-----------	---

Return Value

SEMTECH_SUCCESS	The frequency was set to the specified value.
SEMTECH_INVALID_PARAMETER	The specified value for frequency is outside of the allowed range. Thus, no action has taken place.

3.2.8 GetFrequency

The GetFrequency function returns the unit's main frequency setting.

Prototype

Frequency GetFrequency()

Parameters

None

Return Value

The output frequency, which will be one of the following:

FREQUENCY_250_KHZ
 FREQUENCY_500_KHZ
 FREQUENCY_750_KHZ
 FREQUENCY_1_MHZ

3.2.9 SetPowerSaveFrequency

The SetPowerSaveFrequency function sets a new frequency for the ultrasonic power save.

Prototype

Semtech_ResultCode SetPowerSaveFrequency(PowerSaveFrequency frequency)

Parameters

frequency	The output frequency, which must be one of the following: FREQUENCY_6250_HZ FREQUENCY_12500_HZ FREQUENCY_18750_HZ FREQUENCY_25000_HZ
-----------	--

Return Value

SEMTECH_SUCCESS	The power save frequency was set to the specified value.
SEMTECH_INVALID_PARAMETER	The specified value for frequency is outside of the allowed range. Thus, no action has taken place.

3.2.10 GetPowerSaveFrequency

The GetPowerSaveFrequency function returns the frequency setting for ultrasonic power save.

Prototype

PowerSaveFrequency GetPowerSaveFrequency()

Parameters

None

Return Value

The output frequency, which will be one of the following:

FREQUENCY_6250_HZ
FREQUENCY_12500_HZ
FREQUENCY_18750_HZ
FREQUENCY_25000_HZ

3.2.11 SetMargining

The SetMargining function controls whether margining is disabled, set to below the current output value, or set to above the current output value, and by how much.

Prototype

```
Semtech_StatusCode SetMargining(Margin margin)
```

Parameters

margin	The new state for margin control. Must be one of the following: MARGIN_10_PERCENT_ABOVE MARGIN_5_PERCENT_ABOVE MARGIN_DISABLED MARGIN_5_PERCENT_BELOW MARGIN_10_PERCENT_BELOW
--------	--

Return Value

SEMTECH_SUCCESS	The margining was set to the specified value.
SEMTECH_INVALID_PARAMETER	The specified value for margin is outside of the allowed range. Thus, no action has taken place.

3.2.12 GetMargining

The GetMargining function returns the current margining setting.

Prototype

```
Margin GetMargining()
```

Parameters

None

Return Value

The current state for margin control. Will be one of the following:

```
MARGIN_10_PERCENT_ABOVE  
MARGIN_5_PERCENT_ABOVE  
MARGIN_DISABLED  
MARGIN_5_PERCENT_BELOW  
MARGIN_10_PERCENT_BELOW
```

3.2.13 SetSoftStartTime

The SetSoftStartTime function sets the duration of soft start.

Prototype

```
Semtech_StatusCode SetSoftStartTime(SemtechTime time)
```

Parameters

time	The duration of soft start. Must be one of the following: TIME_250_MICROSECONDS TIME_500_MICROSECONDS TIME_1_MILLISECOND TIME_2_MILLISECONDS TIME_4_MILLISECONDS TIME_8_MILLISECONDS TIME_16_MILLISECONDS
------	--

Return Value

SEMTECH_SUCCESS	The soft start time was set to the specified value.
SEMTECH_INVALID_PARAMETER	The specified value for time is outside of the allowed range. Thus, no action has taken place.

3.2.14 GetSoftStartTime

The GetSoftStartTime function returns the current duration of soft start.

Prototype

```
SemtechTime GetSoftStartTime()
```

Parameters

None

Return Value

The duration of soft start. Will be one of the following:

```
TIME_250_MICROSECONDS  

TIME_500_MICROSECONDS  

TIME_1_MILLISECOND  

TIME_2_MILLISECONDS  

TIME_4_MILLISECONDS  

TIME_8_MILLISECONDS  

TIME_16_MILLISECONDS
```

3.2.15 SetPowerOnDelay

The SetPowerOnDelay function sets the time before power is turned on after the enable is switched on.

Prototype

```
Semtech_StatusCode SetPowerOnDelay(SemtechTime time)
```

Parameters

time	<p>The amount of time to delay before powering on. Must be one of the following:</p> <p>TIME_0_SECONDS TIME_250_MICROSECONDS TIME_500_MICROSECONDS TIME_1_MILLISECOND TIME_2_MILLISECONDS TIME_4_MILLISECONDS TIME_8_MILLISECONDS TIME_16_MILLISECONDS</p>
------	---

Return Value

SEMTECH_SUCCESS	The power on delay was set to the specified value.
SEMTECH_INVALID_PARAMETER	The specified value for time is outside of the allowed range. Thus, no action has taken place.

3.2.16 GetPowerOnDelay

The GetPowerOnDelay function returns the time before power is turned on after the enable is switched on.

Prototype

```
SemtechTime GetPowerOnDelay()
```

Parameters

None

Return Value

The amount of time to delay before powering on. Will be one of the following:

TIME_0_SECONDS
 TIME_250_MICROSECONDS
 TIME_500_MICROSECONDS
 TIME_1_MILLISECOND
 TIME_2_MILLISECONDS
 TIME_4_MILLISECONDS
 TIME_8_MILLISECONDS
 TIME_16_MILLISECONDS

3.2.17 SetPowerSaveMode

The SetPowerSaveMode function is used to set what power save mode is in use.

Prototype

```
Semtech_StatusCode SetPowerSaveMode(PowerSaveMode mode)
```

Parameters

mode	The power save mode to use. Must be one of the following: POWER_SAVE_DISABLED POWER_SAVE_ENABLED POWER_SAVE_ULTRASONIC_ENABLED
------	---

Return Value

SEMTECH_SUCCESS	The power save mode was set to the specified value.
SEMTECH_INVALID_PARAMETER	The specified value for mode is outside of the allowed range. Thus, no action has taken place.

3.2.18 GetPowerSaveMode

The GetPowerSaveMode function returns the power save mode currently in use.

Prototype

```
PowerSaveMode GetPowerSaveMode()
```

Parameters

None

Return Value

The power save mode being used. Will be one of the following:

POWER_SAVE_DISABLED
POWER_SAVE_ENABLED
POWER_SAVE_ULTRASONIC_ENABLED

3.2.19 SetOutputVoltageAdjustment

The SetOutputVoltageAdjustment function adjusts output voltage by between -9% and +9%.

Prototype

```
Semtech_StatusCode SetOutputVoltageAdjustment(VoltageAdjustment adjustment)
```

Parameters

adjustment	The amount to adjust the output voltage. Must be one of the following VOLTAGE_ADJUSTMENT_MINUS_9_00_PERCENT VOLTAGE_ADJUSTMENT_MINUS_8_25_PERCENT VOLTAGE_ADJUSTMENT_MINUS_7_50_PERCENT VOLTAGE_ADJUSTMENT_MINUS_6_75_PERCENT VOLTAGE_ADJUSTMENT_MINUS_6_00_PERCENT VOLTAGE_ADJUSTMENT_MINUS_5_25_PERCENT VOLTAGE_ADJUSTMENT_MINUS_4_50_PERCENT VOLTAGE_ADJUSTMENT_MINUS_3_75_PERCENT
------------	---

	VOLTAGE_ADJUSTMENT_MINUS_3_00_PERCENT VOLTAGE_ADJUSTMENT_MINUS_2_25_PERCENT VOLTAGE_ADJUSTMENT_MINUS_1_50_PERCENT VOLTAGE_ADJUSTMENT_MINUS_0_75_PERCENT VOLTAGE_ADJUSTMENT_NONE VOLTAGE_ADJUSTMENT_PLUS_0_75_PERCENT VOLTAGE_ADJUSTMENT_PLUS_1_50_PERCENT VOLTAGE_ADJUSTMENT_PLUS_2_25_PERCENT VOLTAGE_ADJUSTMENT_PLUS_3_00_PERCENT VOLTAGE_ADJUSTMENT_PLUS_3_75_PERCENT VOLTAGE_ADJUSTMENT_PLUS_4_50_PERCENT VOLTAGE_ADJUSTMENT_PLUS_5_25_PERCENT VOLTAGE_ADJUSTMENT_PLUS_6_00_PERCENT VOLTAGE_ADJUSTMENT_PLUS_6_75_PERCENT VOLTAGE_ADJUSTMENT_PLUS_7_50_PERCENT VOLTAGE_ADJUSTMENT_PLUS_8_25_PERCENT VOLTAGE_ADJUSTMENT_PLUS_9_00_PERCENT
--	---

Return Value

SEMTECH_SUCCESS	The voltage was set to the specified value.
SEMTECH_INVALID_PARAMETER	The specified value for voltage is outside of the allowed range. Thus, no action has taken place.

3.2.20 GetOutputVoltageAdjustment

The GetOutputVoltageAdjustment function returns the current output voltage adjustment.

Prototype

```
VoltageAdjustment GetOutputVoltageAdjustment()
```

Parameters

None

Return Value

The amount that the output voltage is being adjusted. Will be one of the following:

```

VOLTAGE_ADJUSTMENT_MINUS_9_00_PERCENT
VOLTAGE_ADJUSTMENT_MINUS_8_25_PERCENT
VOLTAGE_ADJUSTMENT_MINUS_7_50_PERCENT
VOLTAGE_ADJUSTMENT_MINUS_6_75_PERCENT
VOLTAGE_ADJUSTMENT_MINUS_6_00_PERCENT
VOLTAGE_ADJUSTMENT_MINUS_5_25_PERCENT
VOLTAGE_ADJUSTMENT_MINUS_4_50_PERCENT
VOLTAGE_ADJUSTMENT_MINUS_3_75_PERCENT
VOLTAGE_ADJUSTMENT_MINUS_3_00_PERCENT
VOLTAGE_ADJUSTMENT_MINUS_2_25_PERCENT
VOLTAGE_ADJUSTMENT_MINUS_1_50_PERCENT
VOLTAGE_ADJUSTMENT_MINUS_0_75_PERCENT
VOLTAGE_ADJUSTMENT_NONE
VOLTAGE_ADJUSTMENT_PLUS_0_75_PERCENT

```

VOLTAGE_ADJUSTMENT_PLUS_1_50_PERCENT
VOLTAGE_ADJUSTMENT_PLUS_2_25_PERCENT
VOLTAGE_ADJUSTMENT_PLUS_3_00_PERCENT
VOLTAGE_ADJUSTMENT_PLUS_3_75_PERCENT
VOLTAGE_ADJUSTMENT_PLUS_4_50_PERCENT
VOLTAGE_ADJUSTMENT_PLUS_5_25_PERCENT
VOLTAGE_ADJUSTMENT_PLUS_6_00_PERCENT
VOLTAGE_ADJUSTMENT_PLUS_6_75_PERCENT
VOLTAGE_ADJUSTMENT_PLUS_7_50_PERCENT
VOLTAGE_ADJUSTMENT_PLUS_8_25_PERCENT
VOLTAGE_ADJUSTMENT_PLUS_9_00_PERCENT

4 Integration

4.1 System Requirements

- All API functions must be called from within a single thread. Although the SC493 is intended for multithreaded systems, the APIs are not designed to be simultaneously called from multiple threads.
- The SC493 APIs are written with the assumption that 8-bit data accesses are atomic for the processor the code is executing on. This means that 8-bit values are written to or read from system memory within a single CPU instruction.

4.2 Files

The drivers are contained within three files: `SC493.h`, `SC493.c`, and `semtech_integration.h`. The uses of these files are described in the following table.

File	Usage
<code>SC493.h</code>	Contains all prototypes and data type required to use the SC493 drivers. This file is intended to be included within the files that access the SC493 APIs.
<code>SC493.c</code>	Contains the implementation of the SC493 APIs and drivers. This file does not need to be modified for typical usage models.
<code>semtech_integration.h</code>	Contains macros that must be defined in order to allow the SC493 drivers to access the system hardware and resources. This file must be modified during the system integration process. This file should not be included within any file other than <code>SC493.c</code> .

4.3 General Integration

The macros in this section must be implemented for successful operation.

4.3.1 SetSemHigh() and SetSemLow()

The `SetSemHigh()` and `SetSemLow()` macros are used to drive the EN pin of the Semtech chip high and low respectively. This enables or disables the entire chip.

4.3.2 SC493_SLAVE_ADDRESS

The `SC493_SLAVE_ADDRESS` define determines how the SC493 will be accessed over I²C. The last three bits of the seven bit address are programmable by pins A2-A0. The address in binary should be 0001xxx where x is a programmable bit.

4.3.3 Semtech_ReadI2C(slaveAddress, data, length)

`Semtech_ReadI2C` is a macro that reads from the current I²C address into `data`. `slaveAddress` and `length` should be of type `unsigned char` and `data` should be an `unsigned char *`. A call to `Semtech_WriteI2C` is used to set the address before calling `Semtech_ReadI2C`.

4.3.4 Semtech_WriteI2C(slaveAddress, data, length)

`Semtech_WriteI2C` is a macro that writes `length` bytes of data over I²C. `slaveAddress` and `length` should be of type `unsigned char` and `data` should be an `unsigned char *`.