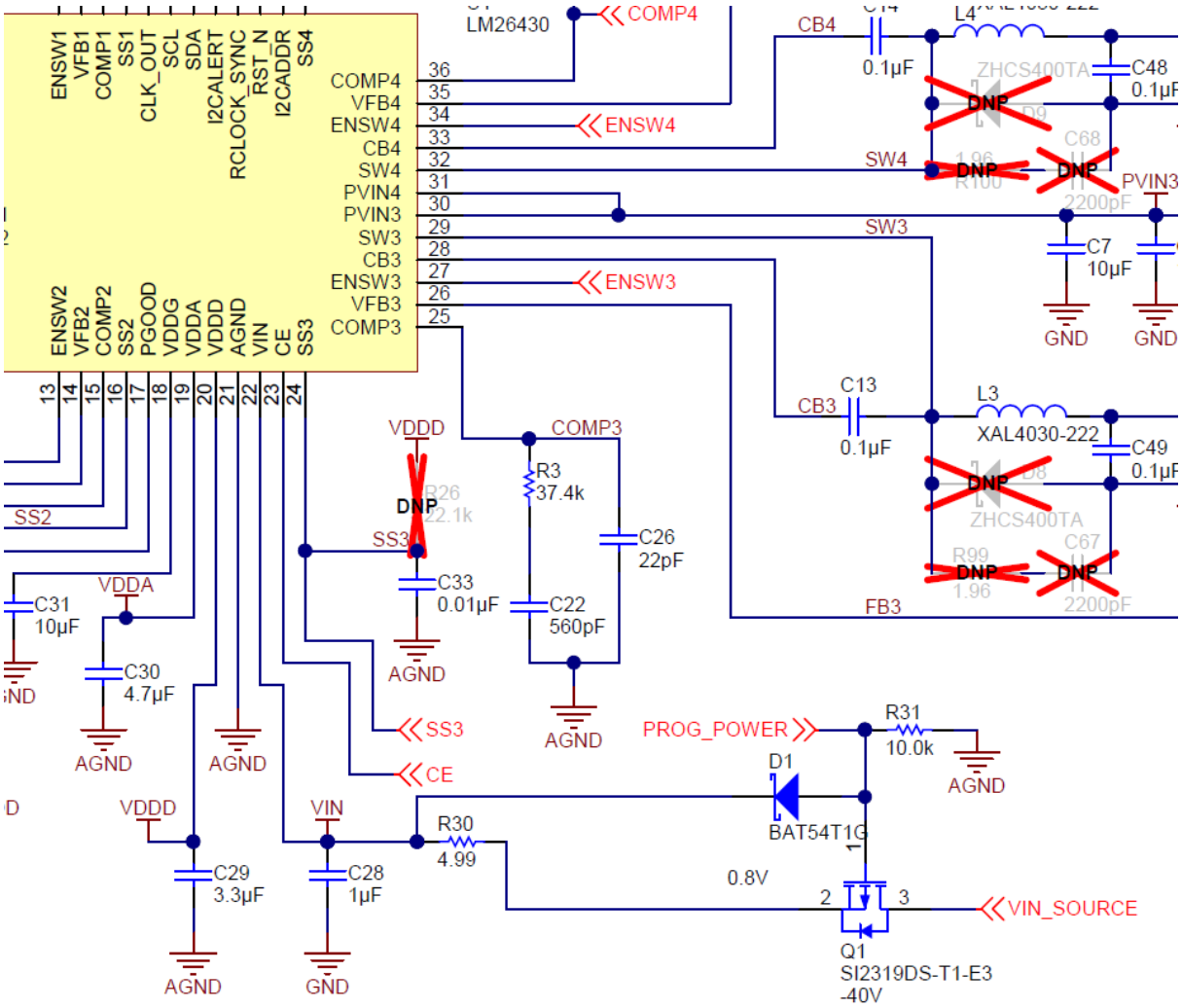
Programming the TPS65400

Hardware Setup:

The TPS65400 can be programmed as a standalone part in a socket or after assembly on the system PCB. In either case, to program the TPS65400, the following connections must be made:

Pin #	Pin name	Connection
18	VDDG	10uF ceramic cap
19	VDDA	4.7uF ceramic cap
20	VDDD	3.3uF ceramic cap
21	VIN	Apply supply voltage between 4.5V and 18V
23	CE	Tied to logic high or floating
38	I2CADDR	Resistor to ground to set I2C address, or if floating, this will default to address 7-bit I2C 0b1101111
39	RST_N	Tied to logic high (> 1.2V). This is often tied to VDDD.
42	SDA	I2C SDA line. Needs pull-up resistor according to I2C spec.
43	SCL	I2C SCL line. Needs pull-up resistor according to I2C spec.

All other pins can be unconnected or connected as described in the datasheet for a normal application scenario. Please note that when conducting programming after assembly onto the PCB, care should be taken to ensure that when applying power to VIN that if power flows into the PVIN's there is not and unexpected startup of the output rails. In some cases it may be desirable to connect VIN and PVIN with a diode or PFET circuit to prevent power flowing from VIN to PVIN. And example of a PFET circuit is shown on the TPS65400 EVM (consisting of components Q1, D1, and R31). In this case PROG_POWER goes to a programming connector which supplies +5V, and VIN_SOURCE is powered from the PVIN. The circuit below prevents PROG_POWER from flowing into PVIN but has less drop than a Schottky diode based solution under normal operating circumstances when PROG_POWER is not present.



Configuration:

The basic programming sequence is summarized as:

1. Apply power to VIN. Wait 20ms for part to startup and configure itself.
2. Enable writes by writing to “address 0x10 value 0x00”
3. Write to all the registers in address 0xD0 through 0xF1 (a total of 49 registers)
4. Burn to EEPROM by writing to “address 0x11 value 0x00”
5. Perform a reset (pull RST_N low for 100us and then high) or power-cycle (pull VIN below UVLO of 3.8V for 100us and then re-apply)
6. Read-back all registers to confirm.

TPS65400 Programming Files

The TPS65400 GUI can save the TPS65400 EEPROM configuration in a .xml format. This allows users to configure a TPS65400 EVM to the desired settings and store them into EEPROM using the GUI, and then test the configuration. The configuration can then be saved in the .xml format and used to configure other parts using the GUI.

In order to streamline the programming of the TPS65400 in a production environment, these .xml formats can be converted into a .hex format.

The format of the .hex file follows the Intel .hex format (http://en.wikipedia.org/wiki/Intel_HEX) and in this case is a series of lines, each with 2 bytes of data. The first byte is the register address, and the second is the data to be written. Since the registers in the TPS65400 are "paged", the .hex file embeds the necessary commands to change the PMBus page when needed.

This is an example of one line in the .hex file:

:02001200D30019

Byte count = 02
Address=0012 (this is not really relevant to the TPS65400, but is used to be consistent with the .hex format)
Record type = 00
Data = D300 (TPS65400 register address D3, write data 00)
Checksum = 19

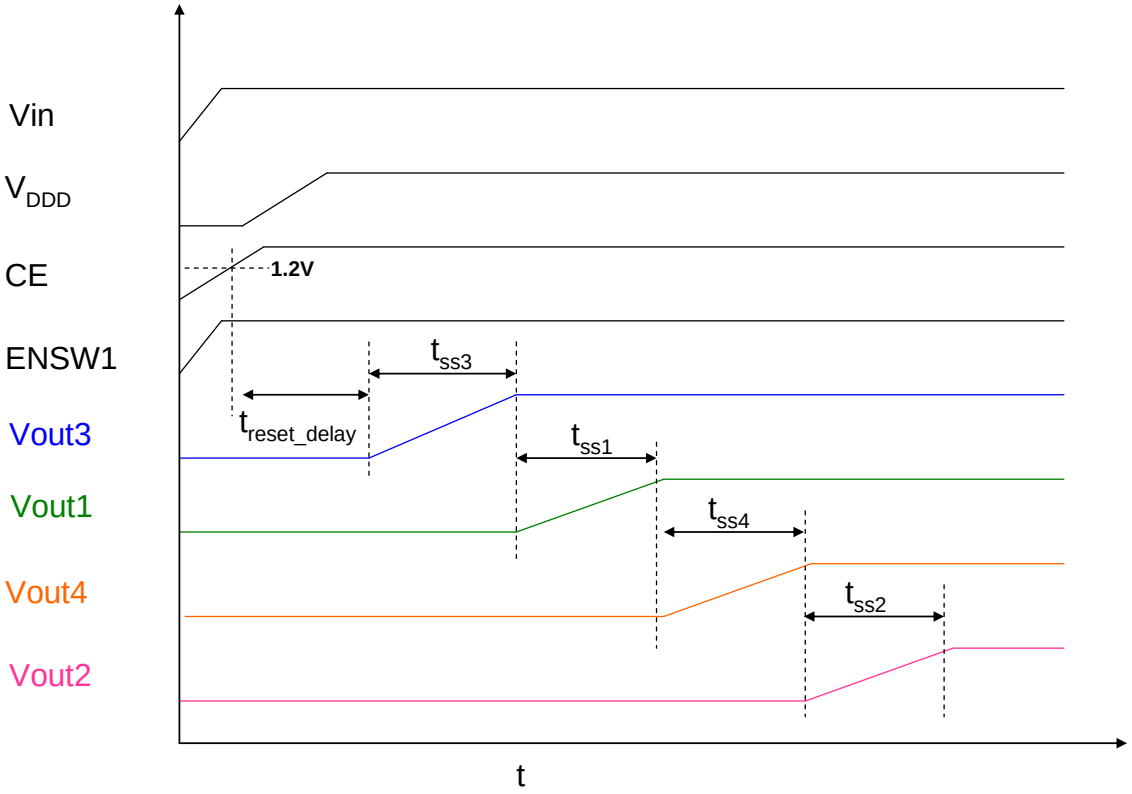
The EOF line for the .hex file is:
:00000001FF

It is assumed that the programmer reading in this file knows the 7 bit hex address of the TPS65400 and can properly construct the I2C commands for each line of data.

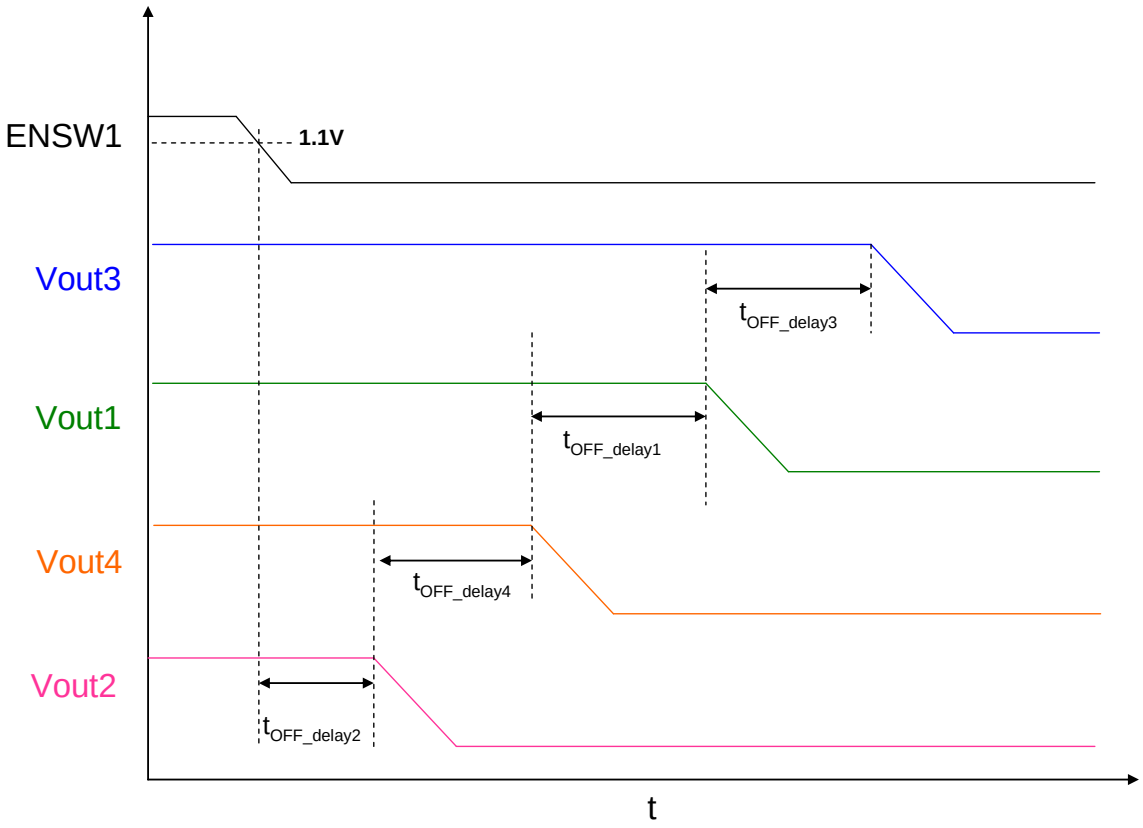
TPS65400 Configuration Example

Start-Up Sequence

This is the desired start-up sequence:



Here is the proposed power-down sequence:



To achieve the desired start-up and power down sequence, the following hardware and PMBus configuration must be done.

Hardware Configuration

CE left floating (internal pull-up current will rise CE voltage to enable level when Vin comes up).

ENSW1 tied to Vin through resistor divider.

PMBus configuration

The following registers need to be programmed in the EEPROM of the IC using I2C:

PAGE Support	REGISTER (command code in hex)	NAME	VALUE binary (hex)
0x00	D5h	SEQUENCE_ORDER	0000 1001 (0x09)
0x01	D5h	SEQUENCE_ORDER	0000 0011 (0x03)
0x02	D5h	SEQUENCE_ORDER	0000 1100 (0x0C)
0x03	D5h	SEQUENCE_ORDER	0000 0110 (0x06)
--	DCh	RESET_DELAY	see *Setting RESET_DELAY*
0x00	DDh	TON_TOFF_DELAY	0000 0XXX (see *Setting TOFF_DELAY*)
0x01	DDh	TON_TOFF_DELAY	0000 0XXX (see *Setting TOFF_DELAY*)
0x02	DDh	TON_TOFF_DELAY	0000 0XXX (see *Setting TOFF_DELAY*)
0x03	DDh	TON_TOFF_DELAY	0000 0XXX (see *Setting TOFF_DELAY*)

The command procedure detailed below will configures the TPS65400 to the table above:

1. Write Byte **0x00** to WRITE_PROTECT (10h)
2. Write Byte **0b00000XXX** to RESET_DELAY (DCh) *
3. Write Byte **0b00000010** to PIN_CONFIG_00 (D2h)
   // This sets to ENSW1 to single-pin enable mode

4. Write Byte **0x00** to PAGE (00h)
5. Write Byte **0x09** to SEQUENCE_ORDER (D5h)
6. Write Byte **0b00000XXX** to TON_TOFF_DELAY (DDh) *

7. Write Byte **0x01** to PAGE (00h)
8. Write Byte **0x03** to SEQUENCE_ORDER (D5h)
9. Write Byte **0b00000XXX** to TON_TOFF_DELAY (DDh) *

10. Write Byte **0x02** to PAGE (00h)
11. Write Byte **0x0C** to SEQUENCE_ORDER (D5h)
12. Write Byte **0b00000XXX** to TON_TOFF_DELAY (DDh) *

13. Write Byte **0x03** to PAGE (00h)
14. Write Byte **0x06** to SEQUENCE_ORDER (D5h)
15. Write Byte **0b00000XXX** to TON_TOFF_DELAY (DDh) *

16. Send Byte to STORE_DEFAULT_ALL (11h)

* see sections *Setting RESET_DELAY* and *Setting TOFF_DELAY* for details

After STORE_DEFAULT_ALL is issued, the settings are stored in non-volatile memory (EEPROM). The TPS65400 will remember the settings for all future startups (even after the device loses power) until STORE_DEFAULT_ALL is issued again with new settings.

Register details

The following three sections were drawn from the datasheet and give more detail on the registers that were configured.

Setting RESET_DELAY

The following paragraph of the datasheet shows the available options for the reset delay.

8.10. (DCh) RESET_DELAY

The RESET_DELAY command sets the delay time before any switcher can begin its soft-start after CE is asserted. Thus, if the turn-on sequence or an individual switcher is enabled before this delay is over, there will be no action until the delay is completed. After this delay period is passed, enabling the turn-on sequence or an individual switcher would have an immediate effect.

Table ##. RESET_DELAY Data Byte Contents

Bit [7:3]	Bits [2:0]	Delay Time
XXXXX	000	320 µs (see note below)
XXXXX	001	50ms
XXXXX	010	100 ms (default)
XXXXX	011	250 ms
XXXXX	100	500 ms
XXXXX	101	1000 ms
XXXXX	110	1500 ms
XXXXX	111	2000 ms

Note: All the delay times will be subject to the delay between the rising edge of CE and the stabilizing of the VDDD supply. The RESET_DELAY time in the table is in addition to this power-up delay and has an accuracy of +/- 62.5µs.

There is no PAGE support for this command.

Setting TOFF_DELAY bits

The TON_TOFF_DELAY command sets the delay times after receiving an ON or OFF command for the selected output to *begin* turning ON or OFF.

TON_DELAY of this command are lexically equivalent to TON_DELAY as specified in the PMBus specification Part II. If TON_DELAY is set to 0ms, the device would begin turning ON immediately.

TOFF_DELAY of this command are lexically equivalent to TOFF_DELAY as specified in the PMBus specification Part II. If TOFF_DELAY is set to 0ms, the device would begin turning OFF immediately.

Table ##. TON_TOFF_DELAY Data Byte Contents

Bits [7:6]	Bits [5:3]	Bits [2:0]
XX	TON_DELAY	TOFF_DELAY

This table shows the options for register DDh for setting the TOFF_DELAY, bits [2:0] of register TON_TOFF_DELAY.

Table ##. TON_DELAY, TOFF_DELAY Data Byte Values

Value	Delay Time
000	0 ms (default)
001	1 ms
010	5 ms
011	25 ms
100	100 ms
101	500 ms
110	1000 ms
111	2000 ms