# Device Tab

## Display the ID registers

Read and display the DIE\_REV register (address = FDh).

Read and display the CHIP\_ID register (address = FEh).

Read and display the PSEMI\_ID register (address = FFh).



Figure 1. Device Tab



Figure 2. Event and Status indicator registers.

# I2C Registers Tab

## Display I2C register map upon GUI initialization and refresh.

## Refer to Section 5 in Vader Digital DFT Guide

## 

Figure 3. Registers Tab

# Protocols Tab

## Simular to previous GUIs

Add OTP Access Tab

## 

Figure 4. Protocols Tab – I2C



Figure . Protocols Tab – OTP Access

OTP Access

Refer to 6.1 Reload Shadow Registers procedure in Vader Digital DFT Guide.

Refer to 6.2 Read OTP Memory procedure in Vader Digital DFT Guide.

Refer to 6.3 Write OTP Memory procedure in Vader Digital DFT Guide.

Page Status (Refer to 5.1 Page Loading Algorithm in Vader Digital DFT Guide)

Read I2C register 59h to determine page status. Green indicates Valid (1010b); red indicates Invalid (1111b).

Notes:

1. The MODE register shows the current, active setting for boost-mode and ratio[1:0].  It is read-only and shows the internal d\_boost\_mode and d\_ratio{1:0] signals.  These settings can be controlled in a few ways:
   1. Strap the boost\_mode, ratio0 and ratio1 pins.  They normally get sampled once on POR.
   2. These pins can be resampled on demand.  When the d\_dig\_read signal is force high it will re-sample the pins again.  The d\_dig\_read signal can be forced high with the TM\_OVER3 register.
   3. The individual d\_boost\_mode, d\_ratio[0] and d\_ratio[1] signals can also be directly forced via the TM\_OVER0 register.
   4. Strapping both of the ratio pins HIGH will freeze the normal power-up sequence to allow time for manually overriding of these boost\_mode and ratio settings.
2. The EVENT0 and EVENT1 registers are sticky and the STATUS0 and STATUS1 registers are dynamic, live.  Writing 1s will clear an EVENT bit (eg, write FFh to clear all bits).  An individual bit can also be cleared by writing a 1 in just that bit.  The bit won’t clear if the active equivalent STATUS\* bit is high.
3. The MAIN\_STATE and PWR\_STATE registers could have state names instead of just the raw state variable.  This might make it more usable.  The state variable encoding are included below.
4. The PAGE\_VALID register (at 59h) is a shadow register.  It could come for any of the 3 pages in OTP.  For the Page Status display in the GUI, you would need to manually read the OTP at locations Page-0:00h, Page-1:28h and Page-2:50h.
5. For the OTP Access area in the GUI more information is needed:  NVM\_CTRL, NVM\_ADDR, NVM\_DATA\*, …
6. The NVM\_CMD bits are not self-clearing.  There is a handshake between NVM\_CMD and NVM\_STATUS.  For example, to issue an OTP write.
   1. Set NVM\_CMD.WRITE\_CMD to 1
   2. wait for NVM\_STATUS.WRITE\_DONE to be 1
   3. clear NVM\_CMD.WRITE\_CMD to 0
   4. NVM\_STATUS.WRITE\_DONE will then become 0
7. It might be useful to have GUI support to automate committing the current shadow registers to the OTP.  It should have a way to indicating which page to burn (0, 1 or 2).  For example, if we wanted to reprogram a device with page-0 programmed, we would:
   1. Write 0Fh to the PAGE shadow register
   2. Commit the shadow registers to PAGE-0
   3. Configure all of the shadow registers to the new values
   4. Commit the shadow registers to PAGE-1
8. 1010b indicates a valid page and anything else is invalid. 1111b is a good value to over-write to mark it as invalid.

Here are the names for each state.

MAIN\_STATE:

// startup states

typedef enum logic [1:0] {

   MAIN\_RESET         = 2'd0,

   MAIN\_LOAD\_OTP      = 2'd1,

   MAIN\_MEASURE\_CPCLK = 2'd2,

   MAIN\_IDLE          = 2'd3

} main\_state\_t;

PWR\_STATE:

// power control states

typedef enum logic [4:0] {

   PWR\_RESET          = 5'd0,    // wait for shadow register load to complete

   PWR\_INIT\_SETTLE    = 5'd1,    // allow time for deglitching logic to settle

   PWR\_CPCLKM\_WAIT    = 5'd2,    // wait until the CP clock period is known

   PWR\_PGATE          = 5'd3,    // wait for PGATE pulldown

   PWR\_FAULT\_CHECK    = 5'd4,    // wait for all fault condition to clear

   PWR\_WAIT\_CHECK     = 5'd5,    // wait for all fault condition to clear

   PWR\_CP\_START       = 5'd6,    // enable CP

   PWR\_DISCHG\_PHASE   = 5'd7,    // discharge P1/P2 phase caps

   PWR\_DISCHG\_FLYCAP  = 5'd8,    // discharge flying capacitor

   PWR\_UP\_SOFTSTART   = 5'd9,    // step-up softstart

   PWR\_UP\_ACTIVE      = 5'd10,   // step-up active

   PWR\_DOWN\_BSTCHARGE = 5'd11,   // step-down bootstrap charge

   PWR\_DOWN\_SOFTSTART = 5'd12,   // step-down softstart

   PWR\_DOWN\_ACTIVE    = 5'd13,   // step-down active

   PWR\_HICCUP         = 5'd14,   // cooldown

   PWR\_SHUTDOWN       = 5'd15,   // controlled shutdown

   PWR\_OFF            = 5'd16,   // off

   PWR\_LATCH\_FAULT    = 5'd17    // critical fault, stay in cooldown

} pwr\_state\_t;