PowerXR Configuration and Programming

This article describes the procedure to configure and program EXAR Corporation's PowerXR Digital Power devices via I²C interface. Details shown here apply to XRP7704/08/40 and XRP7713/14 devices and PowerArchitect software version 3.00.

PowerXR is the new generation of digital DC/DC converters from EXAR Corporation. PowerXR products are highly configurable via I²C interface allowing designers to configure parameters such as output voltage, switching frequency, loop compensation, sequencing, etc., via PowerArchitect software. Furthermore, operating parameters such as output current, input voltage, chip temperature, etc. can be read easily from the device. Ability to configure parts using software minimizes component count (see Fig.1 for a typical schematic of design), increases flexibility, reduces time to market and minimizes the risk of a re-design.

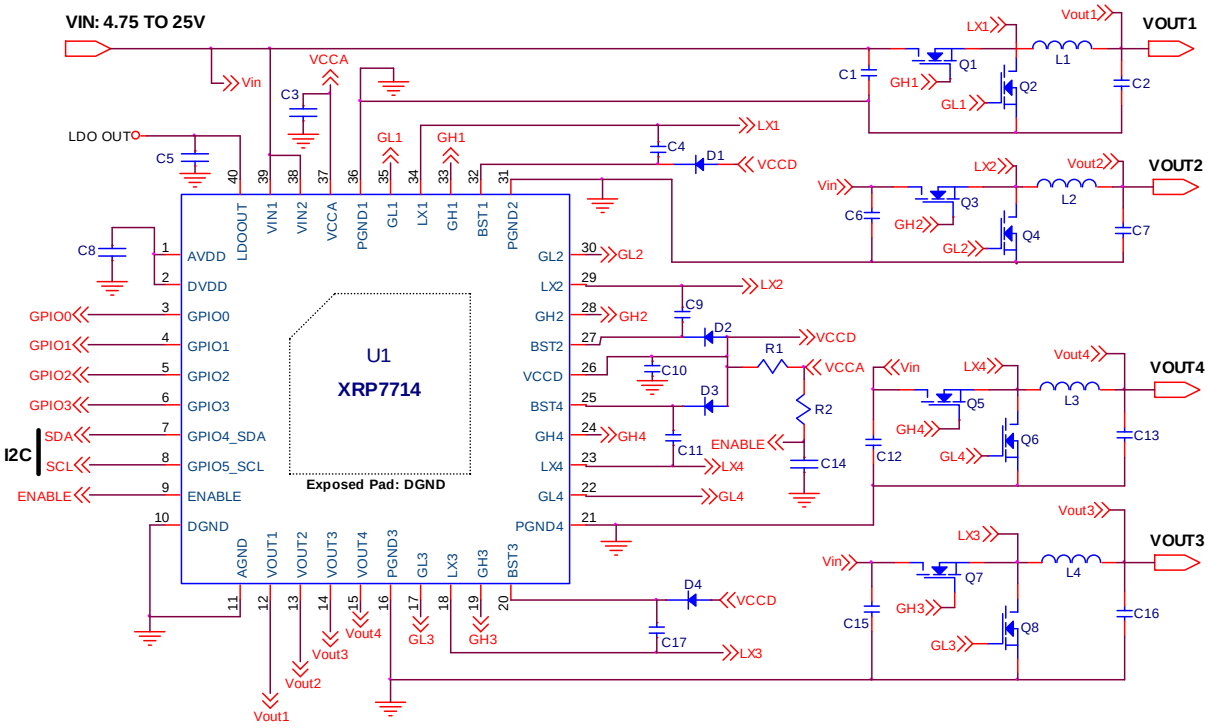


Figure 1 – XRP7714 Application Diagram

I²C Interface

Created by Philips Semiconductors (NXP) and commonly written as 'I²C' stands for Inter-Integrated Circuit and allows communication of data between I²C devices over two wires. I²C interface available in PowerXR products allows:

- Communication with a System Controller or other Power Management devices for optimized system functionality
- Access to modify or read internal registers that control or monitor:
  - Output Current
  - Input and Output Voltage
  - Soft-Start/Soft-Stop Time
  - "Power Good"
  - Part Temperature
  - Enable/Disable Outputs
  - Over Current and Over Voltage
  - Temperature Faults
  - Adjusting fault limits and disabling/enabling faults

Internal memory – Configuration Registers and NVM (OTP bootprom):

In each PowerXR device, there are two types of memory that can be accessed via I²C interface (see Fig. 2):

- Volatile **Configuration Registers** that can be either read/write or read only. These registers hold the effective control and status information.
- **N**on **V**olatile **M**emory (NVM) that can be read and programmed (**O**ne **T**ime **P**rogramming Only). These registers hold initial configuration to be copied at power up or after a soft reset, into Configuration Registers.

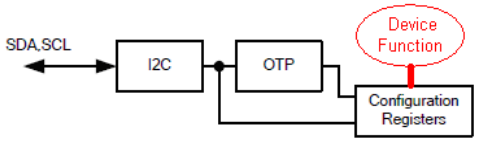


Fig. 2: Internal Memory Structure

Configuration Registers provide effective control and status functions for the device. Control registers are implemented as read/write registers used for setting parameters such as output voltages, loop compensation parameters, channel enable/disable, etc. Status registers are read-only registers that are used for feedback functions such as error/warning flags, output voltage/currents, temperature, etc. Configuration Registers lose their contents when power is removed. This means that at every power-up, Configuration Registers have to be set, to get desired device functionality. This does not necessarily mean that all Configuration Registers can be altered via I²C command. Some of them can be altered only by copying the contents of the NVM (OTP bootprom) into Configuration Registers. At every power-up or after software reset or after disabling/enabling of the device, contents of the NVM (OTP) are automatically transferred into Configuration Registers. Configuration Registers can still be altered via I²C during normal operation of the device (e.g. output voltage). However, some parameters can only be changed when corresponding channels are disabled (e.g. switching frequency).

NVM is partly programmed during manufacturing process. At the least some trimming information is programmed into the NVM (oscillator, voltage reference, etc.). It is also possible to program complete user design during manufacturing process using the programming service provided by Exar. User can program NVM during final board testing via I²C (e.g. nail-bed adapter) or before the device is mounted onto the PCB through a 3rd party device programmer. Devices delivered without a programmed configuration can be safely mounted on the PCB as all power rails are disabled until the device is configured properly via I²C. The NVM can be programmed on a by-bit basis ("0" un-programmed, "1" programmed).

PowerXR devices have more than 200 Configuration Registers. These registers have a width of 8-bit even though not all bits are used in every register. Some registers are read only (e.g. VOUT1_RDBACK), some are write only (e.g. SOFT_RESET), and some are read/write registers (e.g. VOUT_TARGET_CH1). See Table.1 for an example list of registers and their function. NVM (OTP bootprom) can be accessed through writing to a certain Configuration Register - 0xAE for reading, 0xAF for writing to NVM.

Register Name	Register Address	Effective Bits	Function
SLAVE_ADDR	0x0A	<6:1>	I²C Slave Address
VOUT_TARGET_CH1	0x20	<6:0>	Set Output Voltage Channel 1 (50mV/LSB)
VOUT1_RDBACK	0xB0	<7:0>	Read Output Voltage Ch 1 (20mV/LSB)
OTP_READ	0xAE	<7:0>	Location to read from NVM
OTP_WRITE	0xAF	<7:0>	Location to write to NVM
SOFT_RESET	0xFF	<0>	Software Reset

Table 1: Configuration Register Map

From the user perspective there are 2 ways to configure the device:

- "Manually" via I²C interface (e.g. from a micro controller) at every power-up. Internal stand-by LDO can provide power for the micro controller.
- Automatically via internal NVM (OTP bootprom)

Accessing PowerXR device via I²C

Device addressing: PowerXR device acts as a standard I²C slave. The **7-bit slave address** is stored in NVM (OTP bootprom) and the default slave address is 0x00 but can be altered by the user. Fig. 3 shows I²C electrical interface specification; Fig. 4 shows reserved I²C addresses.

I²C SPECIFICATION

PARAMETER	MIN	TYP	MAX	UNITS		CONDITIONS
I2C Speed			400	KHz		Based upon I2C Master Clock
Input Pin Low Level, V_{IL}			0.3 VI0	V		VIO = 3.3 V ±10%
Input Pin High Level, V_{IH}	0.7 VIO			V		VIO = 3.3 V ±10%
Hysteresis of Schmitt Trigger inputs, V_{hys}	0.05 VIO			V		VIO = 3.3 V±10%
Output Pin Low Level (open drain or collector), V_{OL}			0.4	V		I_{SINK} = 3mA
Input leakage current	-10		10	µA		Input is between 0.1 VIO and 0.9 VIO
Output fall time from V_{IHmin} to V_{ILmax}	20 + 0.1 C_b		250	ns		With a bus capacitance from 10 pF to 400 pF
Capacitance for each I/O Pin			10	pF		

Note
1. C_b is the capacitance of one bus in pF

Fig. 3: XRP7704/08/40 I²C specs

As the default slave address of 0x00 is a reserved address (see Fig. 4), user has to change I²C slave address of the PowerXR device when used in conjunction with other I²C devices. However, if PowerXR device is the only part connected to I²C bus, there is no need to change the slave address.

Slave address	R/W bit	Description
0000 000	0	general call address[1]
0000 000	1	START byte[2]
0000 001	X	CBUS address[3]
0000 010	X	reserved for different bus format[4]
0000 011	X	reserved for future purposes
0000 1XX	X	Hs-mode master code
1111 1XX	X	reserved for future purposes
1111 0XX	X	10-bit slave addressing

[1] The general call address is used for several functions including software reset.

[2] No device is allowed to acknowledge at the reception of the START byte.

[3] The CBUS address has been reserved to enable the inter mixing of CBUS compatible and I²C bus compatible devices in the same system. I²C-bus compatible devices are not allowed to respond on reception of this address.

[4] The address reserved for a different bus format is included to enable I²C and other protocols to be mixed. Only I²C-bus compatible devices that can work with such formats and protocols are allowed to respond to this address.

Fig. 4: I²C reserved addresses

Note: I²C slave address simply cannot be altered by writing into volatile configuration registers. To alter I²C slave address, NVM (OTP bootprom) address 0x0A has to be programmed. For the new address to take effect, it is necessary to cycle power or toggle enable pin or initiate software reset i.e. write to configuration register 0xFF.

The SLAVE_ADDR location in NVM (address 0x0A) uses 6 bits (<6:1>). This means that the LSB of the 7-bit I²C address is always 0. The LSB can only be changed through GPIO3, when configured accordingly. Doing so allows two PowerXR devices with the same slave address programmed in a chain by connecting GPIO3 to 0 on one device and to 1 on the other. The two devices then have different 7-bit slave addresses: xxxxxx0 & xxxxxx1 (Fig. 5).

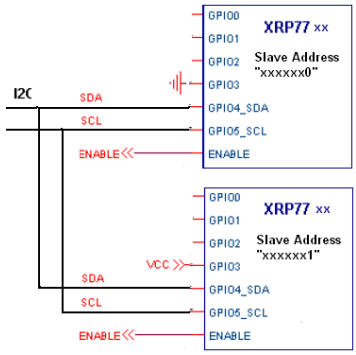
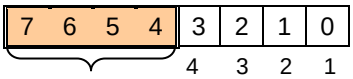


Fig. 5: I²C chain with 2 devices – LSB slave address is determined by GPIO3

There are four GPIO Configuration Registers (GPIOx_CONFIG). The upper 4 bits determine functionality of the GPIO, lower 4 bits determine corresponding channel (See Fig. 6). For example, when writing "1010 0011" to Configuration Register, 0x90, then GPIO0 acts as an enable signal for channels 1 and 2. To configure GPIO3 for LSB slave address functionality, 0x94 has to be written into Configuration Register 0x93 i.e. NVM location 0x93 has to be programmed with a value "0x94". The same functionality can be achieved by activating the checkbox, "Use GPIO3 to control LSB of I²C address" under the "Digital Design" tab.



GPIO	Address
0	0x90
1	0x91
2	0x92
3	0x93

GPIOx_CONFIG

- "0001" SoftStart in Progress for Channel X
- "0010" OVC Fault occurred on Channel X
- "0011" OVC Warning occurred on Channel X
- "0100" OVP Fault occurred on Channel X
- "0101" PWRGD flag on Channel X
- "0110" UVLO Fault/OverTemp Fault or Warning
- "**1001**" CLK_IN (GPIO1)/SYNC_IN (GPIO2)/ **I²C LSB Select (GPIO3)**
- "1010" ENABLE pin for Channel X
- "1011" ENABLE pin for StandbyLDO and Channel X

Fig. 6: GPIO Configuration

Read/Write Configuration Register & Program/Verify NVM (OTP bootprom):

A Configuration Register read is initiated by a START condition followed by the 7-bit slave address and the R/W bit set to write direction. The following data byte written to the slave contains the address of the register that will be transferred during the following read cycle. Current write transfer is terminated by an ACK condition followed by a repeated Start condition to begin the read transfer. Read transfer again transmits the slave address followed by R/W set to red direction. The following byte transferred from the slave to the master contains the content of the addressed register. The transfer has to be concluded by a NAK and a stop condition. Configuration Register writes and NVM read/writes require a similar procedure. Fig. 7 to Fig. 10 show how to read/write the Configuration Register and how to read (verify) and program the NVM.

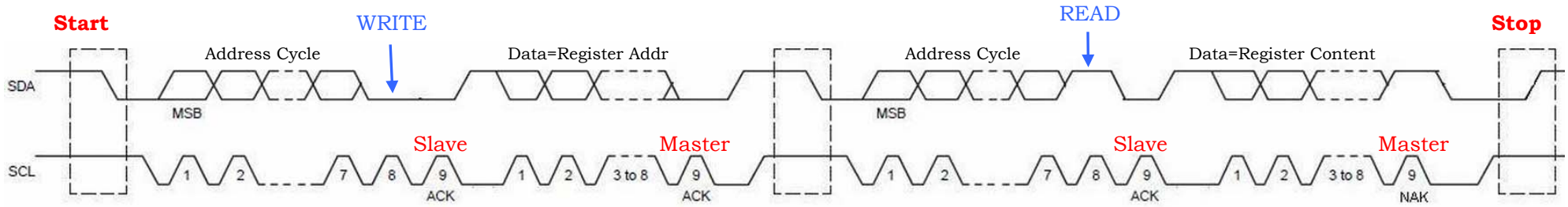


Fig. 7 - Configuration Register Read

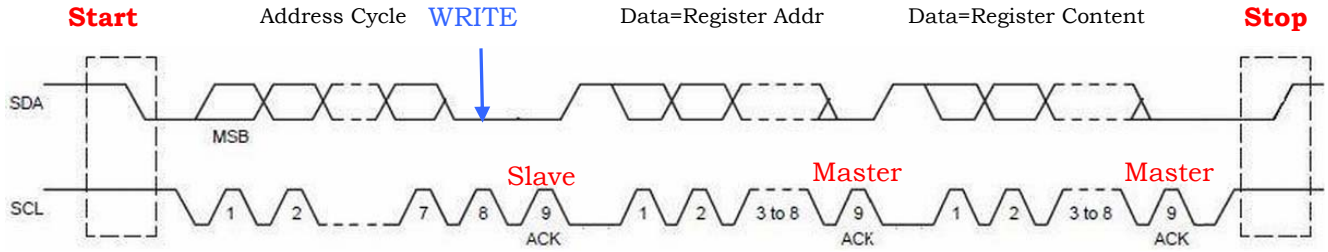


Fig. 8 -Configuration Register Write

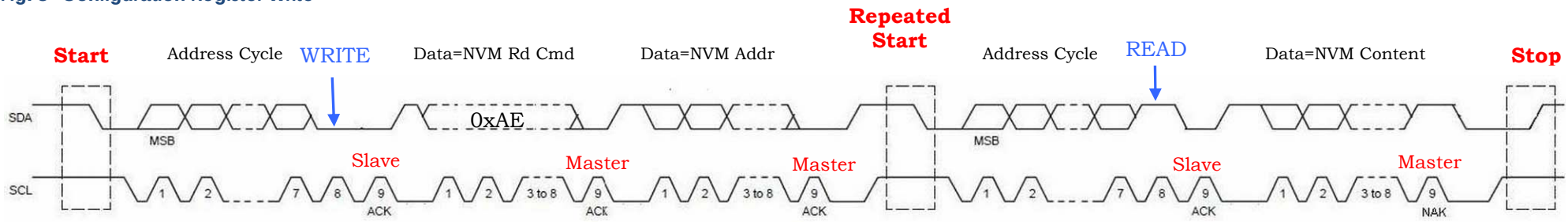


Fig. 9 - NVM Read (Verify)

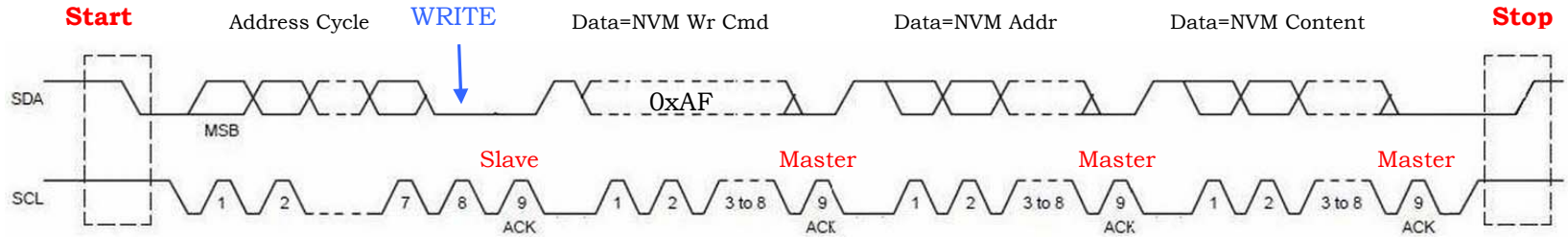


Fig. 10 - NVM Write (Program)

Using PowerArchitect Design Software:

PowerArchitect is the design software for all EXAR Power^{XR} Digital Power products. PowerArchitect supports complete design process and allows the designer to configure, program and debug the circuit. Following section describes how to modify the contents of the NVM to change I^{2}C slave address and how to setup PowerArchitect software to select the desired device. Fig. 11 shows how to make necessary steps in the design software.

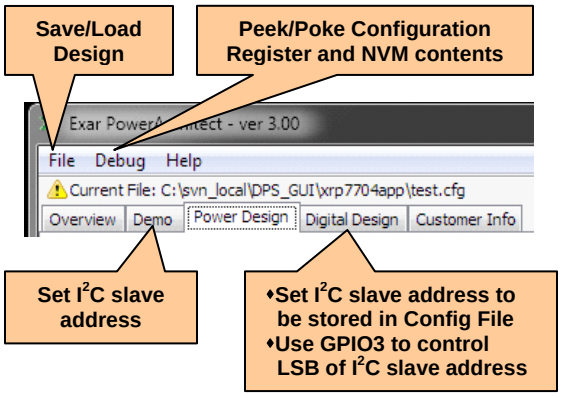


Fig. 11: PowerArchitect Software Selection Tabs

Selecting desired slave device: Power^{XR} design software, "PowerArchitect," needs to know I^{2}C slave address in order to talk to the desired device. The address can be set under the "Demo" tab by selecting "Set I^{2}C Address …" (Fig. 12). The address has to be set as a decimal value (0-127). This value is NOT stored in the configuration file (.cfg) when saving the design.

Fig. 12: Setting I^{2}C address in the GUI

Note: Software releases before version 2.69 do not support slave addresses other than the default value of 0x00.

Using GPIO3 to control LSB of I^{2}C slave address: In order to use GPIO3 to control LSB of the slave address, it is necessary to select "Use GPIO3 to control LSB of I^{2}C address" option under the Digital Design tab (Fig. 13). When this option is selected, software writes 0x94 into Configuration Register 0x93 to enable this feature. As the Configuration Registers are volatile, setting is lost when power is removed or a software reset occurs. To permanently enable this feature, NVM has to be programmed accordingly.

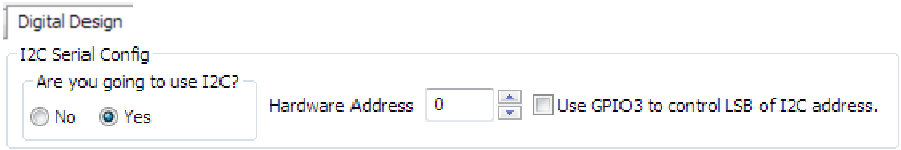


Fig. 13: Using GPIO3 to control the LSB of the slave address

Storing I²C slave address in the Configuration File (.cfg): Value specified under "Hardware Address" field (Fig. 13) will be stored in the Configuration File. Whenever a Configuration File is loaded with a non-zero value for "Hardware Address," I²C address (Fig. 12) is set accordingly. The value is also taken into account when programming NVM (OTP bootprom) with the current settings.

Changing default I²C slave address of the PowerXR device: To change default I²C slave address of the PowerXR device, the NVM (OTP bootprom) location - 0x0A (SLAVE_ADDR) has to be programmed. Since, NVM is one-time programmable, one needs to make sure that the right value has been chosen. (see Fig. 4 for reserved addresses)

- When changing I²C slave address, Peek/Poke functionality can be used to write to the NVM (Fig. 14). Value programmed into NVM only takes effect when a soft reset is issued or power is cycled. Peek/Poke function asks for 8-bit hex value as data to be programmed, whereas the slave address is only 7-bit. Therefore, MSB is not used, the LSB has to be 0 (even addresses only) For example, to program a slave address of 64 (decimal), a hex value of 0x40 has to be written into NVM location 0x0A (Fig. 14). For complete user modifiable 7705 register map, see Appendix – A.
- When programming complete NVM with the current settings the "Hardware Address" field (Fig. 13) is used.

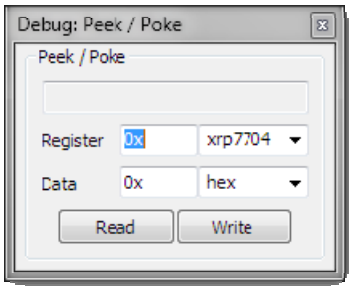


Fig. 14: Programming SLAVE_ADDR Register

- To program NVM, select "Program NVM" under the File tab (Fig. 15)

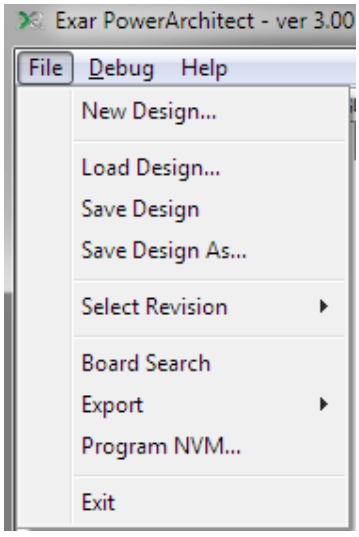


Fig. 15: Programming NVM

Appendix – A – 7704 Register Map

XRP7704 Register Map

Address	Register Name	Bit Length	Description	RunTime
	I²C			No
0x0A	SLAVE_ADDRESS	<7:0>	• I²C address in hexadecimal and should be greater than 80H. • Bit [7] needs to be a "1." • Bit [6:1] is the configurable portion of the address. ◦ NOTE: Bit [1] can be set through GPIO4, depending upon the OTP GPIO configuration. • Bit [0] is reserved based upon I²C protocol and is used for read or write operations.	

CH 1	CH 2	CH 3	CH 4	Output Voltage Set Point			Yes
0x20	0x40	0x60	0x80	VOUT_TARGET_CHx	<7:0>	• 7-bit data [6:0] sets output voltage target for each channel. MSB [7] must be "0." • The target VOUT value ranges from 600mv to 5.1v (linearly equivalent to 7'h0C to 7'h66, with a resolution of 50mv/lsb.) Any inputs with 7-bit value above 7'h66 will produce 5.1v target. • The VOUT_TARGET_CHx is used as a final value for the internal ramped version VOUTx_TARG. VOUTx_TARG is incremented from 0 to VOUTx_TARGET over the soft-start-ramp period. VOUTx_TARG is used as a reference for error generator block. Bit [7] is unused.	

Vout					
5	3.3	2.5	1.8	1.2	0.9
64h	42h	32h	24h	18h	12h

CH 1	CH 2	CH 3	CH 4	**Start up and Shut down**			Yes
0x21	0x41	0x61	0x81	PD _FALL_LB_CHx (Power Down Configuration)	<7:0>	• This register along with PD_FALL_HB_CHx register bits are used to set delay time and ramp-down of VOUT for each channel over a specified time when the channel is turned off. • The channel is turned off by ENABLE which is done either by I^2C register, or GPIO. • The 2 x 8-bit data bytes set delay and ramp-down time. High Byte [7:2] of PD_FALL_HB_CHx register sets delay time at 250us per step. High Byte [1:0] (of PD_FALL_HB_CHx register), Low Byte [7:0] (of PD _FALL_LB_CHx register) sets the time to decrement 50mv. The time it takes to decrement 50mv is the decimal value of the 10-bit data *1us. Ex: If 10-bit data set is 0x002, 1v VOUT fall time would be (1/.05)*1us*2= 40us; if 10-bit data is 0x0FF, it would be (1/0.05)*1us*255 = 5.1ms; if 10-bit data is 0x3FF, it would be (1/0.05)*1us*1023=20.46ms	
0x22	0x42	0x62	0x82	PD_FALL_HB_CHx	<7:0>	• This register along with PD_FALL_LB_CHx register bits are used to set delay and ramp-down time of VOUT signal for each channel when the channel is turned off. • Bits [7:2] set delay time at 250us per step. These bits set delay time before ramp-down of VOUT starts. • Bits [1:0] along with bits [7:0] of PD_FALL_LB_CHx register are used to set ramp-down time.	

0x23	0x43	0x63	0x83	SS_RISE_LB_CHx (Power Up Configuration)	<7:0>	• This register along with SS_RISE_HB_CHx register bits are used to set delay time and ramp-up of VOUT for each channel over a specified time when the channel is enabled. • 2 x 8-bit data bytes set delay and the ramp-up time • High Byte [7:2] of SS_RISE_HB_CHx register, sets delay time, with 250us per step. If set to 0x00, there is no delay. • High Byte [1:0] (of SS_RISE_LB_CHx register), Low Byte [7:0] (of SS_RISE_LB_CHx register) make up 10-bit data that sets the incremental time for each 50mv. The time it takes to increment each 50mv is the decimal value of the 10-bit data *1us. ◦ Ex: If 10-bit data set is 0x002, 1v VOUT total rise time would be (1/.05)*1us*2= 40us; if 10-bit data is 0x0FF, it would be (1/0.05/)*1us*255 = 5.1ms; if 10-bit data is 0x3FF, it would be (1/0.05)*1us*1023=20.46ms
0x24	0x44	0x64	0x84	SS_RISE_HB_CHx	<7:0>	• This register along with SS_RISE_LB_CHx register bits are used to set delay and ramp-up time of VOUT signal for each channel when the channel is enabled. • Bits [7:2] set delay time at 250us per step. These bits set delay time before ramp-up of VOUT starts. • Bits [1:0] along with bits [7:0] of SS_RISE_LB_CHx register are used to set ramp-up time.
0x2E	0x4E	0x6E	0x8E	VOUT_SHUTD_THRESH_CHx (Shutdown Threshold Voltage)	<7:0>	• This register sets output voltage threshold at which the channel stops outputting pulses. • 8-bit byte. The value can be programmed from 0 to SET_VOUT_TARGET voltage output, with the same resolution as the set target output voltage i.e. a resolution of 50mv/lsb. • To perform a “Passive Shutdown” of the channel, i.e. stop sending pulses and allow any residual load to pull down the output, set this register to the same value as the target voltage. • To perform an “Active Shutdown” of the channel, i.e. have the chip regulate the output voltage down to a “stop threshold,” set this register value to the preferred “stop voltage.”

CH 1	CH 2	CH 3	CH 4	**OCP - Over Current Set Point**			Yes
0x25	0x45	0x65	0x85	VIOUT_MAX_CHx	<7:0>	• The maximum current output for each channel is set through this register. The load current is represented in the form of voltage across Rds-on of the low side FET. For example, if the voltage is 300mv, and Rds-on is 20 m ohms, then the load current would be 15A. • The Voltage across the FET is referred to as VIOUT. The 8-bit register VIOUT_MAX_CHx has 6 bits [5:0] that set the maximum VIOUT allowed, and 2 bits [7:6] that set warning levels. • The maximum VIOUT allowed is 320mv. The 6-bit data [5:0] gives a full range of 320mv with a resolution of 5mv/lsb. • If the output current *Rds-on exceeds this setting for 2 consecutive switching cycles, OCP fault will be set. This OCP flag can be read out via READ_FAULT register or through one of the GPIO pins (given that this GPIO pin had been configured to be an output for OCP flag). • The OCP voltage is measured 4 clock cycles after the measured GL external signal goes high. It i.e. OCP Voltage, is not based on the internal GL signal. ◦ Ex: If 8 bit data is set to 0x14, maximum VIOUT allowed would be = 20*5 = 100mv. • The 2 MSB [7:6] of VIOUT_MAX_CHx register are to set OCP warning limit. VIOUT_MAX_CHx [7:6] Voltage level below VIOUT - Max allowed 00 10mV 01 20mV 10 30mV 11 40mV	
0x26	0x46	0x66	0x86	ISENSE_PARAM_CHx	<7:0>	• This register sets 2 parameters in current load measurement. • Bit [2:0] sets the zero-crossing offset. Bit [3] is not used and is ignored. • Bit [7:4] sets the delay from when GL goes high to when track-n-hold starts. This delay includes “blanking time.” ◦ *NOTE:* On readback, Bit [7:4] are shifted right by one bit. So while reading the data, user needs to keep in mind that Bit [7] is unused, Bits[6:3] represent delay, and Bits[2:0] represent zero-crossing offset.	

CH 1	CH 2	CH 3	CH 4	**Dead Time Adjustment**			Yes
0x27	0x47	0x67	0x87	DT_RISE_CHx	<5:0>	• This register sets the non-overlap or overlap between GLx fall and GHx rise. • Parameter Range is 1-20 decimal units. Deadtime can be programmed to be between 1 and 20 units of time, where each unit of time = (1/(fswitching*256)) ◦ Note: that the value "00" is automatically remapped to "1F" by the digital logic. ◦ Ex: For switching frequency of 500MHz, 1 unit of time = (1/(500K *256)) ~ 7.8125 ns. So 10 (0x0A) units of time would result in 78.125 ns of non-overlap or overlap between GLx fall and GHx rise	
0x28	0x48	0x68	0x88	DT_FALL_CHx	<5:0>	• This register sets the non-overlap or overlap between GLx fall and GHx rise. • Parameter Range is 1-20 decimal units. Deadtime can be programmed to be between 1 and 20 units of time, where each unit of time = (1/(fswitching*256)). • Note: that the value "00" is automatically remapped to "1F" by the digital logic. ◦ Ex: For switching frequency of 500MHz, 1 unit of time = (1/(500K *256)) ~ 7.8125 ns. So 20 (0x14) units of time would result in 156.25 ns of non-overlap or overlap between GLx fall and GHx rise.	
0xA8	0xA8	0xA8	0xA8	DEAD_TIME_CONTROL (One register for all channels)	<3:0>	• The bit position sets the appropriate channel to use auto or user controlled deadtime. ◦ Ex: “0000” Auto Dead time all channels "0001” Non-overlap time of Channel 1 is user controlled “0010” Non-overlap time of Channel 2 is user controlled “0100” Non-overlap time of Channel 3 is user controlled “1000” Non-overlap time of Channel 4 is user controlled “0011” Non-overlap time of Channel 1, 2 is user controlled “1111” Non-overlap time of all 4 Channels is user controlled	

CH 1	CH 2	CH 3	CH 4	Power Good			Yes
0x29	0x49	0x69	0x89	PWRGD_TARG_MAX_CHx	<7:0>	• This register sets the lower bound value of VOUT that triggers PWRGOOD flag. • The 8-bit register content sets the value, at 20mv per LSB. ◦ Ex: If 8 bit data is set to 0x0A, lower bound value of VOUT that triggers PWRGOOD flag would be = 20*10 = 200mv.	
0x2A	0x4A	0x6A	0x8A	PWRGD_TARG_MIN_CHx	<7:0>	• This register sets the upper bound value of VOUT that triggers PWRGOOD flag. • The 8-bit register content sets the value, at 20mv per LSB. • Ex: If 8 bit data is set to 0x20, upper bound value of VOUT that triggers PWRGOOD flag would be = 20*32 = 640mv.	
0x2B	0x4B	0x6B	0x8B	PWRGD_DLY_CHx	<2:0>	• This register sets the delay(ing) time before output the PWRGOOD Flag when VOUT is within range set by PWRGD_TARG_MIN and PWRGD_TARG_MAX. • If any of the GPIO pins are configured to be output, carrying PWRGOOD flag information, it will either go high/low (depending on active polarity of the pin). • In the 8 bit register, only 3 bits are used. The most 5 MSB are ignored. Bit [2:0] 000 No delay. 001 50ms 010 100ms 011 150ms 100 200ms 101 300ms 110 400ms 111 600ms • Each channel is set independently. If a fault occurs (either due to Over – voltage or over-current), this delay counter will be reset. • The counter used for this purpose is the same one that times the delay of soft-start and soft-off.	

					<3>	• OverVoltageProtection (OVP) Shutdown Selection. This bit defines the operation that the logic will take when there is an OverVoltage Protection Fault for a Channel. • If this bit is set to “1,” then the channel does an Active (Brute Force) shutdown, where GHx is OFF and GLx is ON. • If this bit is set to “0,” then the channel does a Passive shutdown, where GHx is OFF and GLx is OFF.

	GPIOs			Yes
0x90 0x91 0x92 0x93 0x94 0x95	GPIO0_CONFIG GPIO1_CONFIG GPIO2_CONFIG GPIO3_CONFIG GPIO4_CONFIG GPIO5_CONFIG	<7:0>	• The upper nibble sets the functionality on the pin, while the lower nibble chooses the channel. • The upper nibbles are defined as follows: [7:4] "0001" SoftStart in Progress for Channel X [7:4] "0010" OCP Fault occurred on Channel X [7:4] "0011" OCP Warning occurred on Channel X [7:4] "0100" OVP Fault occurred on Channel X [7:4] "0101" PWRGD flag on Channel X [7:4] "0110" UVLO Fault/OverTemp Fault or Warning [7:4] "1001" CLK_IN (GPIO1)/SYNC_IN (GPIO2)/ I^2C LSB Select (GPIO3) [7:4] "1010" ENABLE pin for Channel X [7:4] "1011" ENABLE pin for StandbyLDO and Channel X [7:4] "1100" UVLO WARNING on VIN1 or VIN2 (lower bits 1 and 0, define VIN2 and VIN1 respectively) • **Note: For 7713/7714 an added configuration was added:** ◦ [7:4] “1111” Controlled by the SET_GPIO_ACTIVE (0xA9) Register. The corresponding bit in the GPIO_ACTIVE Register will either raise or lower the pin output. • **Note:** Only for UVLO WARNING bits 1 and 0 represent Vin2 and Vin1 respectively. ◦ Bits [3:0]:	

			Bit 0: Channel 1 (1 - Enable, 0 - Disable) Bit 1: Channel 2 (1 - Enable, 0 - Disable) Bit 2: Channel 3 (1 - Enable, 0 - Disable) Bit 3: Channel 4 (1 - Enable, 0 - Disable) ◦ Ex: Bits [3:0] “1010” chooses channel 4 and channel 2.
0x96	GPIO_ACT_POL	<5:0>	• This register sets active polarity of GPIO. Under GPIO_CONFIG table, all exerted values are high active. ◦ Ex: After GPIO 3 pin has been configured to be an ENABLE pin for both CH1 and Standby LDO, then when GPIO3 is asserted high, these two, Standby LDO and Channel 1, are enabled. If GPIO_ACT_POL[3] is set, then GPIO3 pin needs to be asserted low to enable Standby LDO and channel 1. • The default is active high. • Bit [5:0] are set to reverse polarity of GPIO5 to GPIO0. If chip operates with I^2C, Bit [4:5] are ignored since GPIO [4:5] are used for SDA/SCL.
0xBB	GPIO_STATE (Read the present status of a GPIO Pin)	<3:0>	• Bit <0>: READ_GPIO_STATE[0]: state of GPIO[0] • If GPIO_0 is configured as input, then this is the current input state of the corresponding GPIO Pin. • If GPIO_0 is configured as output, then this is the current state (HI/LOW) that logic is attempting to put on this pin (depending on how GPIO is configured; fault state / polarity, etc…). • Bits <3:1>: Read state for other GPIOs.

0x9D	SYNC_MODE_CONFIG	<7:0>	• This register sets the chip to either Master or Slave during Sync mode where 2 chip's outputs are synchronized. Clock of Master is output and used as clock for Slave. • Description: The slave chip would power up with its internal VCO, then switch to external clock when "Slave mode" is detected. The output clock from Master connects to Slave's external clock. The Master can also be taking external clock instead of its own VCO. When switching over to external clock, frequency detector would detect whether external clock is within range of the internal clock. If it's not, switching will not occur. Bit [3:0] : 0000- Default Stand-alone: Use internal VCO; No CLK_OUT & SYNC_OUT 0001- Stand-alone: Use external CLK via CLK_IN via configured GPIO pin; Auto switchback. 1001- Stand-alone: Use external CLK via CLK_IN via a configured GPIO pin; Auto switch back is disabled 0010- Master: Use internal VCO; output CLK_OUT and SYNC_OUT 0011- Master: Use external CLK via CLK_IN, via a configured GPIO pin; Output CLK_OUT and SYNC_OUT; Auto switchback. 1011- Master: Use external CLK via CLK_IN, via a configured GPIO pin; Output CLK_OUT and SYNC_OUT; Auto switchback disabled x1xx- Slave: Use external CLK (from Master) and SYNC_IN; both via 2 GPIO pins; No CLK_OUT, nor SYNC_OUT; No auto switchback Bit [4:5]: Set % range External CLK can be with respect to internal VCO. 00- 20%; 01- 16%; 10- 12%; 11- 8% Bit [6]: Enable High Frequency IO. Bit [7]: Force master to do a 50% SYNC_OUT duty cycle.

	Channel and LDO Enable Configuration			Yes
0x97	CH_EN_CONFIG	<7:0>	• This command sets the configuration of an “ENABLE.” A channel is enabled when an internal condition (VCCOK/VDDOK/VBGOK) and the “ENABLE” are met. • An “ENABLE” is triggered by different ways. There are 2 bits that control the “ENABLE.” One is the enable_by_i2c, and the other is enable_by_gpio register. These 2 bits must be set to “1” in order to trigger the “ENABLE.” • Upon power-up, the Non-volatile Memory is loaded into the register. The register can also be changed in a running system. The content of enable_by_i2c register can also be changed by Enable/Disable command. • The 2 bit (by_GPIO and by_i2c) configuration is: 00 - I^{2}C Controlled Channel [writing to Register 0xA6 will enable the channel] 01 - Channel is Always ON 10 - Requires BOTH GPIO and I^{2}C to Enable 11 - GPIO Controlled Channel • There is one bit per channel 1 in each nibble: Bit 0: ch1_enable_by_i2c Bit 1: ch2_enable_by_i2c Bit 2: ch3_enable_by_i2c Bit 3: ch4_enable_by_i2c Bit 4: ch1_enable_by_gpio Bit 5: ch2_enable_by_gpio Bit 6: ch3_enable_by_gpio Bit 7: ch4_enable_by_gpio	
0xA6	ENABLE_CH	<3:0>	• This register enables a channel if CH_EN_CONFIG is set to be enabled by I^{2}C interface. • Bit [3:0] enable channel 4 to 1 respectively (at start-up or through I^{2}C Write).	

0x98	STBLDO_EN_CONFIG	<2:0>	• Similar to CH_EN_CONFIG, this register will set the external ENABLE condition of Standby LDO. • Bit [0] if set, LDO is enabled by I^2C. • Bit [1] if set, LDO is enabled by a configured GPIO pin. • If both bits are set, both GPIO pin and a register through I^2C must be set to enable the LDO. • Bit [2] if set, LDOOUT will be at 5v, else 3.3v
0xAD	READ_CH_EN	<3:0>	• This register is a READ-ONLY Register, whose current value lets user know whether or not a channel is enabled [depending upon the channel configuration and transient settings in the system I^2C & GPIO]. Each bit corresponds to a channel. Bit 0: Channel 1 Enable/Disable (1 - Enable, 0 - Disable) Bit 1: Channel 2 Enable/Disable (1 - Enable, 0 - Disable) Bit 2: Channel 3 Enable/Disable (1 - Enable, 0 - Disable) Bit 3: Channel 4 Enable/Disable (1 - Enable, 0 - Disable)

	Faults			Yes
0x99	FAULT_RESP_CONFIG_LB	<7:0>	• This register allows the user to configure interaction among channels when Over Current or Over Voltage occurs. • When OVC occurs in 1 channel, that channel will undergo shut-down and back-up sequence and other channels can also be configured to undergo similar sequence. • Bit [2:0] is to set channel 4, 3, 2 respectively to follow channel 1 when fault occurs in channel 1. • Bit [5:3] is to set channel 4, 3, 1 respectively to follow channel 2 when fault occurs in channel 2. • Bit [7: 6] + HB[0] (of FAULT_RESP_CONFIG_HB register) is to set channel 4, 2, 1 respectively to follow channel 3 when fault occurs in channel 3. • Bit HB[3:1] (of FAULT_RESP_CONFIG_HB register) is to set channel 3, 2, 1 respectively to follow channel 4 when fault occurs in channel 4.	
0x9A	FAULT_RESP_CONFIG_HB	<3:0>		

0xA1	THERMAL_SHDN	<7:0>	• This register sets the value for thermal shut down for OTP (Over Temperature Protection). • Bit [6:0] defines the temperature value with resolution of 5K/lsb. ◦ Ex: To set thermal shut down value at 157 Celsius (i.e. 157 + 273 = 430 Kelvin), 6 bit data should be set to 0x56. Decimal (0x56) = 86, 86 * 5 Kelvin = 430 Kelvin. ◦ NOTE: If the Threshold is set to “0” then the ThermalShutdown is disabled. • Bit [7] defines the warning level. [7] = “0,” the warning level is 5K below the thermal shutdown level. [7] = “1,” the warning level is 10K below the shutdown level.
0xA2	OVP_REG	<7:0>	• This register sets Over Voltage Protection level for a channel. • Bit [1:0] sets the value above target for channel1, bits [3:2] for channel2, bits [5:4] for channel3, and bits [7:6] for channel4. • [1:0], or [3:2], or [5:4], or [7:6] Values above the target are: ≤2.5V >2.5V [00] 300mV 600mV [01] 250mV 500mV [10] 200mV 400mV [11] 150mV 300mV • Blank chip will have default value at 300mv. • The Over-voltage has to be for 4 consecutive switching cycle to set (initiate?) OVP. ◦ Note: When setting ramp-down time, user must set the ramp-down to be slower than the system PID and system response. If ramp-down time is too fast, VOUTx appears to be above the target, since the ramp-down dictates the target values, the OVP detection will trip and system responds with following configured response: i Passive Shutdown with auto 200ms restart ii “Brute force” shutdown with auto 200ms restart iii Raise fault flag only

0xA3	UVLO_TARG_VIN1	<7:0>	• This register sets the under-voltage lock-out (UVLO) Fault level for VIN1. • Each bit represents 100mV. If VIN1 drops below this level, then the channels are shutdown even though the chip is still active. ◦ Ex: To set under-voltage lock-out (UVLO) fault for VIN1 at 5.1v, the data should be set to 0x33. Decimal (0x33) = 51, 51* 100mv = 5.1V
0xA4	UVLO_TARG_VIN2	<7:0>	• This register sets the under-voltage lock-out (UVLO) Fault level for VIN2. • Each bit represents 100mV. If VIN1 drops below this level, then the channels are shutdown even though the chip is still active. ◦ Ex: To set under-voltage lock-out (UVLO) fault for VIN2 at 5.1v, the data should be set to 0x33. Decimal (0x33) = 51, 51* 100mv = 5.1V
0xA5	THERMAL_RESTART	<6:0>	• This register sets the value for Restart after an OverTemperature Fault has been reached. When the temperature returns to this level, the chip automatically goes through a reset. • Bit [6:0] defines the temperature value with resolution of 5K/lsb. ◦ Ex: To set over temperature fault at 127 Celsius (i.e. 127 + 273 = 400 Kelvin), 6 bit data should be set to 0x50. Decimal (0x50) = 80, 80 * 5 Kelvin = 400 Kelvin.
0xAB	UVLO_WARN_VIN1	<7:0>	• This register sets the under-voltage lock-out (UVLO) warning level for VIN1. • Each bit represents 100mV. If VIN1 drops below this level, then the warning bits are set (and GPIO is raised, if configured) to inform the Host. ◦ Ex: To set under-voltage lock-out (UVLO) warning for VIN1 at 5.5v, the data should be set to 0x37. Decimal (0x37) = 55, 55* 100mv = 5.5V
0xAC	UVLO_WARN_VIN2	<7:0>	• This register sets the under-voltage lock-out (UVLO) Warning level for VIN2. • Each bit represents 100mV. If VIN1 drops below this level, then the warning bits are set (and is GPIO raised, if configured) to inform the Host. ◦ Ex: To set under-voltage lock-out (UVLO) warning for VIN2 at 5.5v, the data should be set to 0x37. Decimal (0x37) = 55, 55* 100mv = 5.5V

0xAA	TEST_IGNORE_FAULT	<4:0>	• Following disable bits only disable handling of the fault (restart of supply/chip) and the logic still monitors for the fault and sets appropriate bits/GPIOs. ◦ Set Bit [0] to ignore OVT ◦ Set Bit [1] to ignore OCP ◦ Set Bit [2] to ignore OVP ◦ Set Bit [3] to ignore UVLO VIN1 ◦ Set Bit [4] to ignore UVLO VIN2
0xBC	OCP_FLAG	<7:0>	• This register contains the value of OCP flags. ◦ Bits [3:0] indicate OCP warning for CH4-1 ◦ Bits [7:4] indicate OCP fault for CH4-1
0xBD	OVP_UVLO_OVT_Flag	<7:0>	• This register contains the values of other faults (what other faults, describe, provide examples) flag ◦ Bit[7:6]: OVT_FLAG, OVT_WARN, ◦ Bit[5:4]: UVLO_FLAG[1], UVLO_FLAG[0], ◦ Bit[3:0] OVP_FLAG_CH
0xBF	UVLO_WARNING_FLAG	<1:0>	• Bit [0] - UVLO_WARN_VIN1: ◦ If VIN1 voltage drops below UVLO VIN1 warning threshold (UVLO_WARN_VIN1), then this bit is set. ◦ If VIN1 rises above the UVLO VIN1 warning threshold, then this bit is cleared. • Bit [1] - UVLO_WARN_VIN2: ◦ If VIN2 voltage drops below UVLO VIN2 warning threshold (UVLO_WARN_VIN2), then this bit is set. ◦ If VIN2 rises above UVLO VIN2 warning threshold, then this bit is cleared.

	Channel Status			Yes
0xB0	VOUT1_RDBACK	<7:0>	• This register contains the digital value of measured VOUT for each channel. • 8-bit data, 20mv/lsb. ◦ Ex: If digital value of VOUT on channel 1 is 3.3 V, one would read a value of 0xA5 in register 0xB0. Digital (0xA5) = 165, 165*20mV = 3.3V	
0xB1	VOUT2_RDBACK	<7:0>		
0xB2	VOUT3_RDBACK	<7:0>		
0xB3	VOUT4_RDBACK	<7:0>		

0xB4	VIOUT1_RDBACK	<7:0>	• This register contains digital value of I-Load*Rds for each channel. • Bits [7:6] = "10" – invalid, load current exceeds measurement capability i.e. VRds > 320mv or load current is not measurable due to light load condition (no GL, no V across Rds-on of low FET). • Bits [7:6] = "00" and Bits [5:0] represent VRds at 5mv/lsb. User must know the value Rds of his lower FET to determine load current. ◦ Ex: When Bits [7:6] = "00," if digital value of VOUT1_RDBACK is 0.5V, one would read a value of 0x64. Digital (0x64)=100, 100*5mv = 0.5V
0xB5	VIOUT2_RDBACK	<7:0>	
0xB6	VIOUT3_RDBACK	<7:0>	
0xB7	VIOUT4_RDBACK	<7:0>	
0xB8	VIN1_RDBACK	<7:0>	• This register contains the digital value of measured VIN1 or VIN2. • 8-bit data, 100mv/lsb. ◦ Ex: If digital value of VIN1is 12V, one would read a value of 0x78 in register 0xB8. Digital (0x78) = 120, 120*100mV = 12V
0xB9	VIN2_RDBACK	<7:0>	
0xBA	VTJ_RDBACK	<7:0>	• This register contains the value of VTJ (Junction Temperature). • 8-bit value at 5K/lsb. ◦ Ex: If digital value of VTJ is 42 celcius (i.e. 42 + 273 = 315), one would read a value of 0x3F in register 0xBA. Digital (0x3F) = 63, 63*5 = 315
0xBE	POWER_GOOD_SOFT_START_FLAG	<7:0>	• Bit[7:4] PWRGD flag Bit 7: Channel 4 PWRGD Flag Bit 6: Channel 3 PWRGD Flag Bit 5: Channel 2 PWRGD Flag Bit 4: Channel 1 PWRGD Flag • Bit[3:0] SS flag Bit 3: Channel 4 SS Flag Bit 2: Channel 3 SS Flag Bit 1: Channel 2 SS Flag Bit 0: Channel 1 SS Flag

Technical Support

Technical questions about this application note should be e-mailed to powerxr@exar.com.

Notice

EXAR Corporation reserves the right to make changes to the products contained in this publication in order to improve design, performance or reliability. EXAR Corporation assumes no responsibility for the use of any circuits described herein, conveys no license under any patent or other right, and makes no representation that the circuits are free of patent infringement. Charts and schedules contained here in are only for illustration purposes and may vary depending upon a user's specific application. While the information in this publication has been carefully checked; no responsibility, however, is assumed for inaccuracies.

EXAR Corporation does not recommend the use of any of its products in life support applications where the failure or malfunction of the product can reasonably be expected to cause failure of the life support system or to significantly affect its safety or effectiveness. Products are not authorized for use in such applications unless EXAR Corporation receives, in writing, assurances to its satisfaction that: (a) the risk of injury or damage has been minimized; (b) the user assumes all such risks; (c) potential liability of EXAR Corporation is adequately protected under the circumstances.

Application Note: August 2010