ENVISIONING • EMPOWERING • EXCELLING

XRP7724, XRP7725 I²C Command Set and Programming Guide

Application Note ANP-38

Revision History

Revision	Release Date	Change Description
1.0.0	10/3/12	Initial release of document.
1.1.0	9/12/13	Fixed syntax errors, added command 0x2A and flash command flow diagrams with delays.
1.2.0	1/16/14	Added support for XRP7725, XRP9710/1 devices.
238ANR00	7/9/19	Updated Program Flash Image, Clear Pages, and Erase flow charts. Removed obsolete XRP9710/1 devices.

Table of Contents

List of Figures

List of Tables

Introduction

Some MaxLinear Programmable Power products include an I2C serial command interface for performing numerous common tasks. This document covers details on the command types and their structure as well as their usage in a system. Unless otherwise noted, this guide applies to the following devices:

- XRP7724
- XRP7725

The different type of commands are:

- System commands
- Power Commands
- IO Commands
- Flash Commands

The Command Structure

Serial communication is performed through an SMBus compliant I2C interface that supports Packet Error Checking (PEC) and clock stretching. Allowing Packet Error Checking means that a PEC (packet error code) byte is appended at the end of each transaction. The PEC byte is calculated with a CRC-8 checksum, and calculated over the entire message including the address and read or write bit. The clock stretching feature allows the slave to hold the clock line (SCL) low after receiving (or sending) a byte, indicating that it is not yet ready to process more data. The master in this case may not finish the transmission of the current bit, but must wait until the clock line actually goes high.

Supported I2C serial commands are 8-bit long (with addresses less than 0x80) with data transmitted in word (16-bit long) lengths. The structure is as follows:

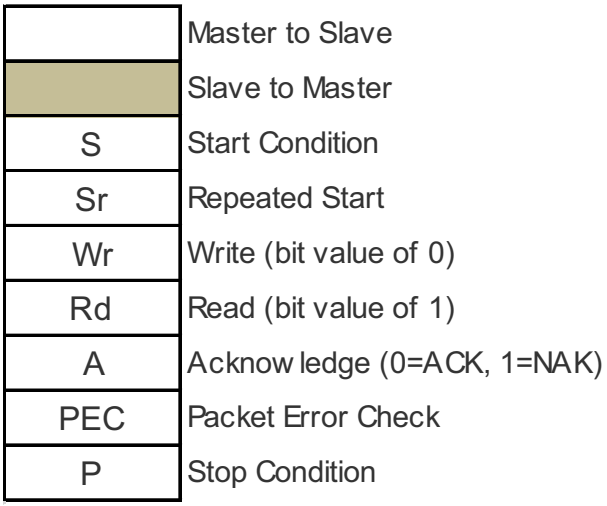


Figure 1: Diagram Key

Standard Write

S	Slave Addr	Wr	A	Std Cmd	A	Wr Byte1	A	Wr Byte2	A	P
1	7	1	1	8	1	8	1	8	1	1

Standard Write with PEC

S	Slave Addr	Wr	A	Std Cmd	A	Wr Byte1	A	Wr Byte2	A	PEC	A	P
1	7	1	1	8	1	8	1	8	1	8	1	1

Standard Read

S	Slave Addr	Wr	A	Std Cmd	A	Sr	Slave Addr	Rd	A	Rd Byte1	A	Rd Byte2	A[1]	P
1	7	1	1	8	1	1	7	1	1	8	1	8	1	1

Standard Read with PEC

S	Slave Addr	Wr	A	Std Cmd	A	Sr	Slave Addr	Rd	A	Rd Byte1	A	Rd Byte2	A	PEC	A[1]	P
1	7	1	1	8	1	1	7	1	1	8	1	8	1	8	1	1

1. Per I2C spec, master NACKs after second read byte or PEC byte (if PEC used) to indicate to slave no more bytes expected and therefore the end of I2C read transaction.

Figure 2: Standard Read and Write with and without PEC

The Command Types

Supported I2C serial commands can be placed into the following categories:

1. **System Commands** — commands to monitor and manage the device, interrupts and faults generated by it.
2. **Power Commands** — commands to control, monitor and adjust power parameters in a running system.
3. **IO Commands** — commands to control direction and polarity of GPIOs and PSIOs.
4. **Flash Commands** — commands to program customer configuration onto NVM.

The System Commands

Table 1: The System Commands

Command	Name	Function	Direction	Byte1	Byte2
0x01	RESERVED				
0x02	GET_HOST_STS	Interrupt status register	R	InterruptStatus[13:8]	InterruptStatus[7:0]
0x03	SET_HOST_INT_MASK	Interrupt status mask	W	InterruptEnable[13:8]	InterruptEnable[7:0]
0x04	CLEAR_HOST_INT	Clear interrupt status	W	InterruptStatus[15:8]	InterruptStatus[7:0]
0x05	GET_FAULT_STS	Get fault status	R	FaultStatus[9:8]	FaultStatus[7:0]
0x06	CLEAR_FAULT_STS	Clear fault status	W	FaultStatus[9:8]	FaultStatus[7:0]
0x08	RESERVED				
0x09	PWR_GET_STATUS	Get supply status	R	SupplyInregMap	SupplyFaultMap
0x0E	PWR_CHIP_READY	Set / read chip ready	R / W	0x00	ChipReady
0x0F	PWR_RESTART	Soft reset	W	0x0F	0x00

GET_HOST_STS (0x02)

Reports the status of the Interrupt Status Register. With this command, the host will determine the cause of the interrupt.

Table 2: Interrupt Status

Bit	InterruptStatus
[0]	GPIO_EVENT
[1]	SUPPLY_FAULT_EVENT
[2]	TEMP_OVER_EVENT
[3]	TEMP_UNDER_EVENT
[4]	TEMP_WARNING_EVENT
[5]	UVLO_FAULT_ACTIVE_EVENT
[6]	UVLO_FAULT_INACTIVE_EVENT
[7]	UVLO_WARNING_EVENT
[8]	FLASH_CLEAR_DONE_EVENT
[9]	FLASH_ERASE_DONE_EVENT
[10]	RESERVED
[11]	V5EXT_RISE_EVENT
[12]	V5EXT_FALL_EVENT
[13]	LDO5OVC_EVENT

SET_HOST_INT_MASK (0x03)

Sets the interrupt mask. Only unmasked interrupts will be reported in the Interrupt Pin. See table above for details.

CLEAR_HOST_INT (0x04)

Clear interrupts. An interrupt will be reported by the Interrupt Pin until the host clears it using this command. See table above for details.

GET_FAULT_STATUS (0x05)

Reports if channels are at OCP or OVP faults, and if LDO3.3 is in regulation or at OVC fault.

Table 3: Get Fault Status

Bit	FaultStatus
[0]	Ch1 OCP
[1]	Ch2 OCP
[2]	Ch3 OCP
[3]	Ch4 OCP
[4]	Ch1 OVP
[5]	Ch2 OVP
[6]	Ch3 OVP
[7]	Ch4 OVP
[8]	LDO3.3 OVC
[9]	LDO3.3 OK

CLEAR_FAULT_STATUS (0x06)

Clears faults. A fault will be reported by the FaultStatus register until the host clears it using this command. See table above for details.

PWR_GET_STATUS (0x09)

Reports supply status, whether it is in regulation or in fault.

Table 4: Pwr Get Status

Bit	SupplyFaultMap
[0]	Ch1 fault
[1]	Ch2 fault
[2]	Ch3 fault
[3]	Ch4 fault
[4]	LDO3.3 fault
Bit	**SupplyInregMap**
[0]	Ch1 inreg
[1]	Ch2 inreg
[2]	Ch3 inreg
[3]	Ch4 inreg
[4]	LDO3.3 inreg

PWR_CHIP_READY (0x0E)

Controls and reports whether the device is operating in configuration or regulation mode. The device must be placed in the configuration mode during in-system programming. This ensures the configuration gets properly loaded into the run time registers, passing all internal checks before channels can be enabled.

Table 5: Pwr Chip Ready

Bit 0	ChipReady
0	Configuration mode
1	Regulation mode

PWR_RESTART (0x0F)

Performs orderly shutdown and restart. This command will not ACK. After the command is sent, the device will not be available for about 50ms.

Interrupts

The following are interrupts that are generated:

- GPIO EVENT — generated from a signal connected to any GPIO or PSIO configured as a general input.
- SUPPLY_FAULT_EVENT — generated by either a channel fault, or LDO3.3 being in regulation or at OVC fault.
- TEMP_OVER_EVENT — generated when the die temperature exceeds the over temperature fault level.
- TEMP_UNDER_EVENT — generated when die temperature is below the temperature warning level. This interrupt will appear as soon as any UVLO or OTP interrupt gets serviced. This interrupt can be asserted at any time the temperature is below the warning level. It should be ignored until after an over-temperature event has been seen.
- TEMP_WARNING_EVENT — generated when the die temperature exceeds the over temperature warning level.
- UVLO_FAULT_ACTIVE_EVENT — generated when the V_{CC} voltage level drops below the UVLO fault level.
- UVLO_FAULT_INACTIVE_EVENT — generated when the V_{CC} voltage is above the UVLO fault level; this could occur while ramping up.
- UVLO_WARNING_EVENT — generated when the V_{CC} voltage drops below the UVLO warning level.
- FLASH_CLEAR_DONE_EVENT — generated when the page clear command is completed.
- FLASH_ERASE_DONE_EVENT — generated when the page erase command is completed.
- V5EXT_RISE_EVENT — generated when a supply connected to the V5EXT pin gets switched in and the chip gets powered from it.
- V5EXT_FALL_EVENT — generated when a supply connected to the V5EXT pin gets switched out and the chip gets powered from LDO5.

Supply Fault Reporting

If a supply is at fault, the following is a sequence of events in communication between the device and a host:

1. The Get Supply Status, Get Fault Status and Interrupt Status Register (SUPPLY_FAULT_EVENT goes high) get updated indicating the fault.
2. The Interrupt pin may be asserted if the Supply Fault Status event is unmasked and there was not a previously asserted interrupt.
3. The host checks the Interrupt Status Register via the **GET_HOST_STS** command (0x02) and determines that a supply is at fault.
4. The host checks the Get Supply Status Register via the **PWR_GET_STATUS** command (0x09) to determine which supply is at fault. The host reads the Get Fault Status Register via **GET_FAULT_STS** (0x05) to determine if the channel at fault reports OCP or OVP. In the case of the supply at fault that is LDO3.3, this register will report if the regulator is regulating OK or if it is at OVC fault. If channel fault is reported but this register does not indicate either OCP or OVP, the channel is at Start-up Timeout Fault. For more information about the Start-up Timeout Fault refer to the product's data sheet. It is also possible, if the channel is set to auto-restart, this register will show a fault while the **PWR_GET_STATUS** command shows that the supply is Inreg and not faulted. This means that the channel had been faulted previously but has successfully restarted itself.
5. The host takes whatever system actions it deems necessary to react to the fault.
6. The host clears the fault(s) via the **CLEAR_FAULT_STS** command (0x06).
7. The host clears the interrupt via the **CLEAR_HOST_INT** command (0x04).

The Power Commands

Table 6: The Power Commands

Command	Name	Function	Direction	Byte1	Byte2
0x10	PWR_READ_VOLTAGE_CH1	Read voltage Ch1	R	ReadVal[15:8]	ReadVal[7:0]
0x11	PWR_READ_VOLTAGE_CH2	Read voltage Ch2	R	ReadVal[15:8]	ReadVal[7:0]
0x12	PWR_READ_VOLTAGE_CH3	Read voltage Ch3	R	ReadVal[15:8]	ReadVal[7:0]
0x13	PWR_READ_VOLTAGE_CH4	Read voltage Ch4	R	ReadVal[15:8]	ReadVal[7:0]
0x14	PWR_READ_VOLTAGE_VIN	Read voltage Vin	R	ReadVal[15:8]	ReadVal[7:0]
0x15	PWR_READ_TEMP_VTJ	Read voltage VTj	R	ReadVal[15:8]	ReadVal[7:0]
0x16	PWR_READ_CURRENT_CH1	Read ViL Ch1	R	ReadVal[15:8]	ReadVal[7:0]
0x17	PWR_READ_CURRENT_CH2	Read ViL Ch2	R	ReadVal[15:8]	ReadVal[7:0]
0x18	PWR_READ_CURRENT_CH3	Read ViL Ch3	R	ReadVal[15:8]	ReadVal[7:0]
0x19	PWR_READ_CURRENT_CH4	Read ViL Ch4	R	ReadVal[15:8]	ReadVal[7:0]
0x1D	PWR_ENABLE_SUP_GROUP	PWREN groups enable / disable	W	SupplyGroup	Enable
0x1E	PWR_ENABLE_SUP	Supply channel enable / disable	W	Supply	Enable
0x20	PWR_SET_VOLTAGE_CH1	Set voltage Ch1	W	(FineAdj[14:12], SetValue[8])	SetValue[7:0]
0x21	PWR_SET_VOLTAGE_CH2	Set voltage Ch2	W	(FineAdj[14:12], SetValue[8])	SetValue[7:0]
0x22	PWR_SET_VOLTAGE_CH3	Set voltage Ch3	W	(FineAdj[14:12], SetValue[8])	SetValue[7:0]
0x23	PWR_SET_VOLTAGE_CH4	Set voltage Ch4	W	(FineAdj[14:12], SetValue[8])	SetValue[7:0]
0x24	PWR_SET_CURRENT_CH1	Set current Ch1	W	WarningValue	FaultValue
0x25	PWR_SET_CURRENT_CH2	Set current Ch2	W	WarningValue	FaultValue
0x26	PWR_SET_CURRENT_CH3	Set current Ch3	W	WarningValue	FaultValue
0x27	PWR_SET_CURRENT_CH4	Set current Ch4	W	WarningValue	FaultValue
0x28	RESERVED				
0x29	RESERVED				
0x2A	PWR_POWER_OK_CFG	Power OK configuration	W	Supply	Supply

The Power Command Descriptions

PWR_READ_VOLTAGE_CHx (0x10-0x13)

Reports V_{OUT} value read by AUX ADC and adjusted by voltage range prescale factor. Follow the provided equation to calculate corresponding voltage level:

V_{OUT} = ReadVal * 0.015 (PRESCALE is accounted for)

PWR_READ_VOLTAGE_VIN (0x14)

Reports V_{CC} value as read from AUX ADC. Follow the provided equation to calculate corresponding voltage level:

V_{OUT} = ReadVal * 0.0125

PWR_READ_TEMP_VTJ (0x15)

Reports Vtj value as read from AUX ADC. Follow the provided equation to calculate corresponding temperature:

Temp (in Kelvin) = ReadVal * 5

PWR_READ_CURRENT_CHx (0x16-0x19)

Reports value of a voltage across the synchronous FET as read from AUX ADC. Follow the provided equations to calculate corresponding voltage level:

(V_PGND - V_LX) = ((ReadVal * 0.010) / IFE_GAIN - 0.040)

If GAIN 8 is enabled, IFE_GAIN = 8, else IFE_GAIN = 4.

Note: IFE_GAIN value can be obtained by reading register 0xD016 via the PowerArchitect™ Peek Poke function. It is a four bit register with following bit description:

Bit 0 — Isense gain 8 setting for channel 1

Bit 1 — Isense gain 8 setting for channel 2

Bit 2 — Isense gain 8 setting for channel 3

Bit 3 — Isense gain 8 setting for channel 4

Value 1 indicates gain 8 setting while value 0 indicates gain 4.

Reading this register from customer software requires the implementation of the I2C register read command structure described in ANP-39.

PWR_ENABLE_SUP_GROUP (0x1D)

Turns supply groups on or off. The first byte specifies the group; the second whether to enable or disable it. There needs to be one command per group. If a group is turned on, its members can be turn off individually using the **PWR_ENABLE_SUP** command. However, the group must be turned off first before a new group turn on command is serviced.

Table 7: Power Enable Supply Group

SupplyGroup	
0	PWREN0
1	PWREN0
2	PWREN0
Enable	
0	Disable
1	Enable

PWR_ENABLE_SUP (0x1E)

Turns channels and LDO3.3 on or off individually. The first byte specifies the supply, the second whether to enable or disable it. There needs to be one command per supply.

Table 8: Power Enable Supply

Supply	
0	Ch1
1	Ch2
2	Ch3
3	Ch4
4	LDO3.3
Enable	
0	Disable
1	Enable

PWR_SET_VOLTAGE_CHx (0x20-0x23)

Changes output voltage settings dynamically. The set voltage level maintains the same prescale voltage factor as in the original configuration, and therefore the same set point resolution. Follow the provided equation to calculate a desired set voltage level. Refer to Appendix A for sample code. As it can be seen from the example code (Appendix A), it is also possible to change the output voltage in a fine set point resolution if desired using this command.

SetValue = V_{OUT} / 0.0125 (PRESCALE is accounted for)

PWR_SET_CURRENT_CHx (0x24-0x27)

Changes channel OCP fault and warning levels dynamically. Follow the provided equation to calculate desired set OCP fault and warning levels:

SetValue = (IFE_GAIN * (40mV + (V_PGND - V_LX))) / 10mV

If GAIN 8 is enabled, IFE_GAIN = 8, else IFE_GAIN = 4.

WarningValue (optional, Warning thresh if non-zero)

Note: The IFE_GAIN value can be obtained by reading register 0xD016 via the PowerArchitect™ Peek Poke function. It is a four bit register with following bit description:

Bit 0 — Isense gain 8 setting for channel 1

Bit 1 — Isense gain 8 setting for channel 2

Bit 2 — Isense gain 8 setting for channel 3

Bit 3 — Isense gain 8 setting for channel 4

Value 1 indicates gain 8 setting while value 0 indicates gain 4.

Reading this register from customer software requires implementation of the I2C register read command structure described in ANP-39.

PWR_POWER_OK_CFG (0x2A)

Include or exclude selected channel(s) or 3.3 LDO to or from Power OK pool dynamically. The byte 1 selects channels:

Bit 0 — LDO 3.3

Bit 8 — Channel 1

Bit 9 — Channel 2

Bit 10 — Channel 3

Bit 11 — Channel 4

Example:

Value 0x0800 will place Ch4 in PowerOK pool by itself.

Value 0x0700 will place all channels except Ch4 in the PowerOK pool.

Value – 0x0001 will place LDO3.3 in the PowerOK pool by itself.

The IO Commands

Table 9: The IO Commands

Command	Name	Function	Direction	Byte1	Byte2
0x30	GPIO_READ_GPIO	Read GPIO pins	R	0x00	IO pin value
0x31	GPIO_SET_GPIO	Set GPIO pins	W	IO number	value
0x32	GPIO_POL_GPIO	Set GPIO polarity	W	IO number	value

The IO Command Descriptions

GPIO_READ_GPIO (0x30)

Reads GPIO or PSIO pin value.

Table 10: Read GPIO Pins

IO Pin Value	
[0]	GPIO0
[1]	GPIO1
[2]	PSIO0
[3]	PSIO1
[4]	PSIO2

GPIO_SET_GPIO (0x31)

Sets GPIO or PSIO pin value when configured as output. The first byte specifies GPIO or PSIO, the second its value.

Table 11: Set GPIO Pins

IO Number	
[8]	GPIO0
[9]	GPIO1
[10]	PSIO0
[11]	PSIO1
[12]	PSIO2
Value	
0	Set to 0
1	Set to 1

GPIO_POL_GPIO (0x32)

Sets GPIO or PSIO polarity. The first byte specifies GPIO or PSIO, the second its polarity. Sending a 0 will clear the polarity bit; 1 will set the polarity bit, inverting the pin function.

Table 12: Set GPIO Polarity

IO Number	
0	GPIO0
1	GPIO1
2	PSIO0
3	PSIO1
4	PSIO2
Value	
0	No inversion
1	Inversion

The Flash Commands

Table 13: The Flash Commands

Command	Name	Function	Direction	Byte1	Byte2
0x40	FLASH_PROGRAM_ADDRESS	Flash address pointer	RW	FlashAddress[15:8]	FlashAddress[7:0]
0x41	FLASH_PROGRAM_DATA	Flash Program / Read	RW	Data0	Data1
0x42	FLASH_PROGRAM_DATA_INC_ADDRESS	Flash Program / Read (Addr++)	RW	Data0	Data1
0x4D	FLASH_INIT	Flash init	W	0x00	Command
0x4E	FLASH_PAGE_CLEAR	Flash pre-erase	RW	Write: 0x00	Write: Page
				Read: FlashPageClearStatus	Read: FlashPageClearStart
0x4F	FLASH_PAGE_ERASE	Flash erase	RW	Write: 0x00	Write: Page
				Read: FlashPageEraseStatus	Read: FlashPageEraseStart

The Flash Command Descriptions

FLASH_PROGRAM_ADDRESS (0x40)

Flash address pointer.

FLASH_PROGRAM_DATA (0x41)

Flash data to be programmed or read. Operates on 2 bytes of data at a time. The first byte is data from the address written into the pointer (above); the second byte is data from the address +1. If programming a bit is not needed, keep it high.

Data0 = [*FlashAddress]

Data1 = [*(FlashAddress+1)]

FLASH_PROGRAM_DATA_INC_ADDRESS (0x42)

Flash data to be programmed or read to or from incrementing address location. Operates on 2 bytes of data at a time, then increments Flash address by 2. The first byte is data from the address; the second byte is data from the address +1. If programming a bit is not needed, keep it high.

Data0 = [*FlashAddress]

Data1 = [*(FlashAddress+1)]

FLASH_INIT (0x4D)

Performs power Flash down, enable Flash / Clear / Program, and initialize page erase tasks.

Table 14: Flash Init

Command	
0	Powerdown flash
1	Enable flash / clear / program
5	Initialize page erase

FLASH_PAGE_CLEAR (0x4E)

Performs page clear task on pages 0 to 6. Refer to the table below for page identifiers. As a write command, the first byte is 0x00; the second byte indicates a page. As a read command, the first byte represents the clear task status; the second byte represents start indicator (ready or busy).

Table 15: Flash Pre-Erase

Write	
Page	
0x00	Page 0 (0x000 - 0x03F)
0x01	Page 1 (0x040 - 0x07F)
...	
0x06	Page 6 (0x180 - 0x1BF)
Read	
FlashPageClearStatus	
0x00 - 0xFE	Number of retries
0xFF	Clear error
FlashPageClearStart	
0	Ready
1	Busy

FLASH_PAGE_ERASE (0x4F)

Performs page erase task on pages 0 to 6. Refer to the table below for page identifiers. As a write command, the first byte is 0x00; the second byte indicates a page. As a read command, the first byte represents the erase task status; the second byte represents start indicator (ready or busy).

Table 16: Flash Erase

Write	
Page	
0x00	Page 0 (0x000 - 0x03F)
0x01	Page 1 (0x040 - 0x07F)
...	
0x06	Page 6 (0x180 - 0x1BF)
Read	
FlashPageEraseStatus	
0x00 - 0xFE	Number of retries
0xFF	Erase error
FlashPageEraseStart	
0	Ready
1	Busy

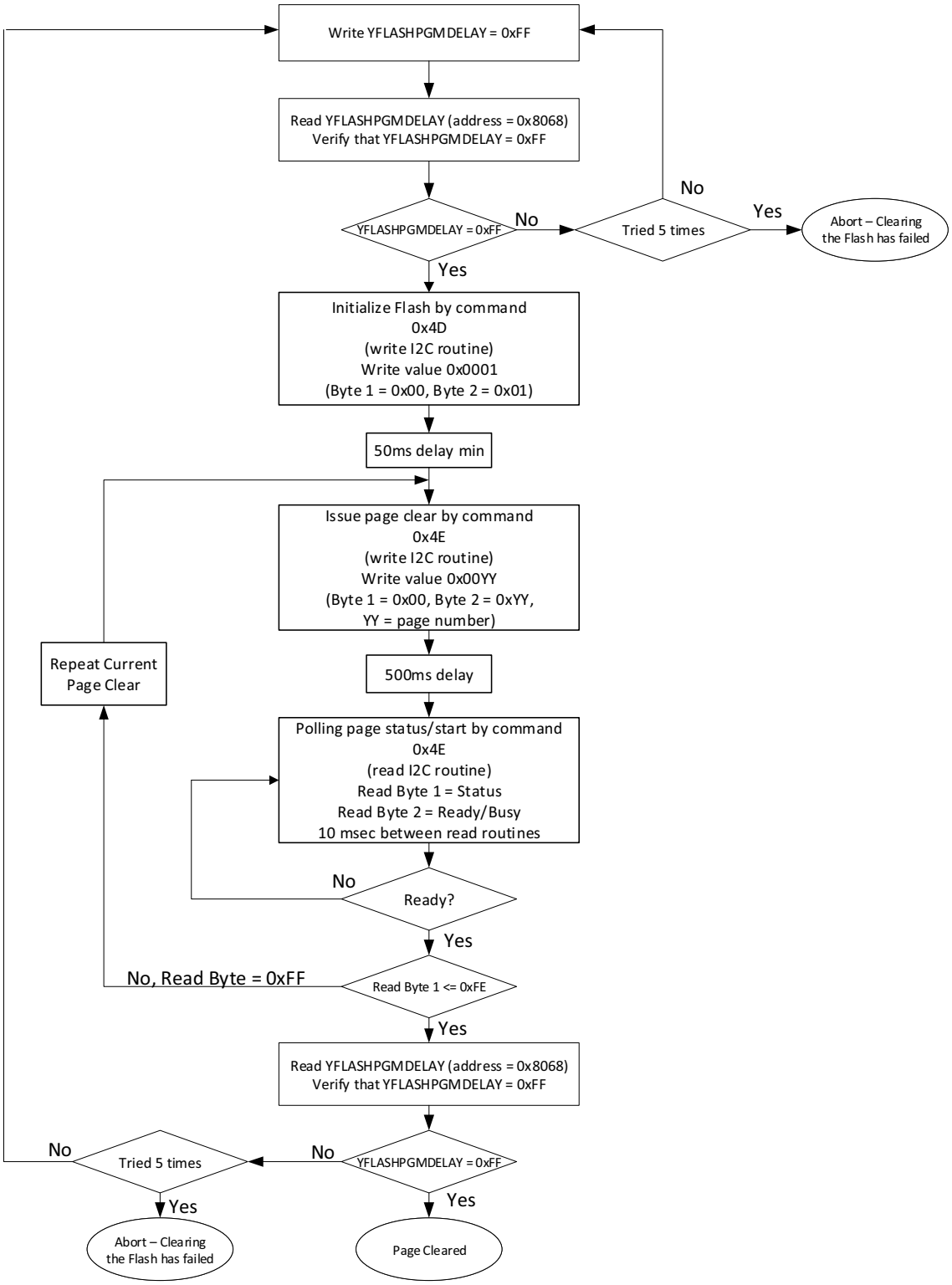
Programming can be done while regulating, however flash data is only read at power-on or reset.

Flash Page Programming Procedure

Before programming of a flash HEX image can take place, flash pages need to be properly cleared and erased. See the flow diagrams starting on the next page.

Clear Pages

Follow the diagram below:

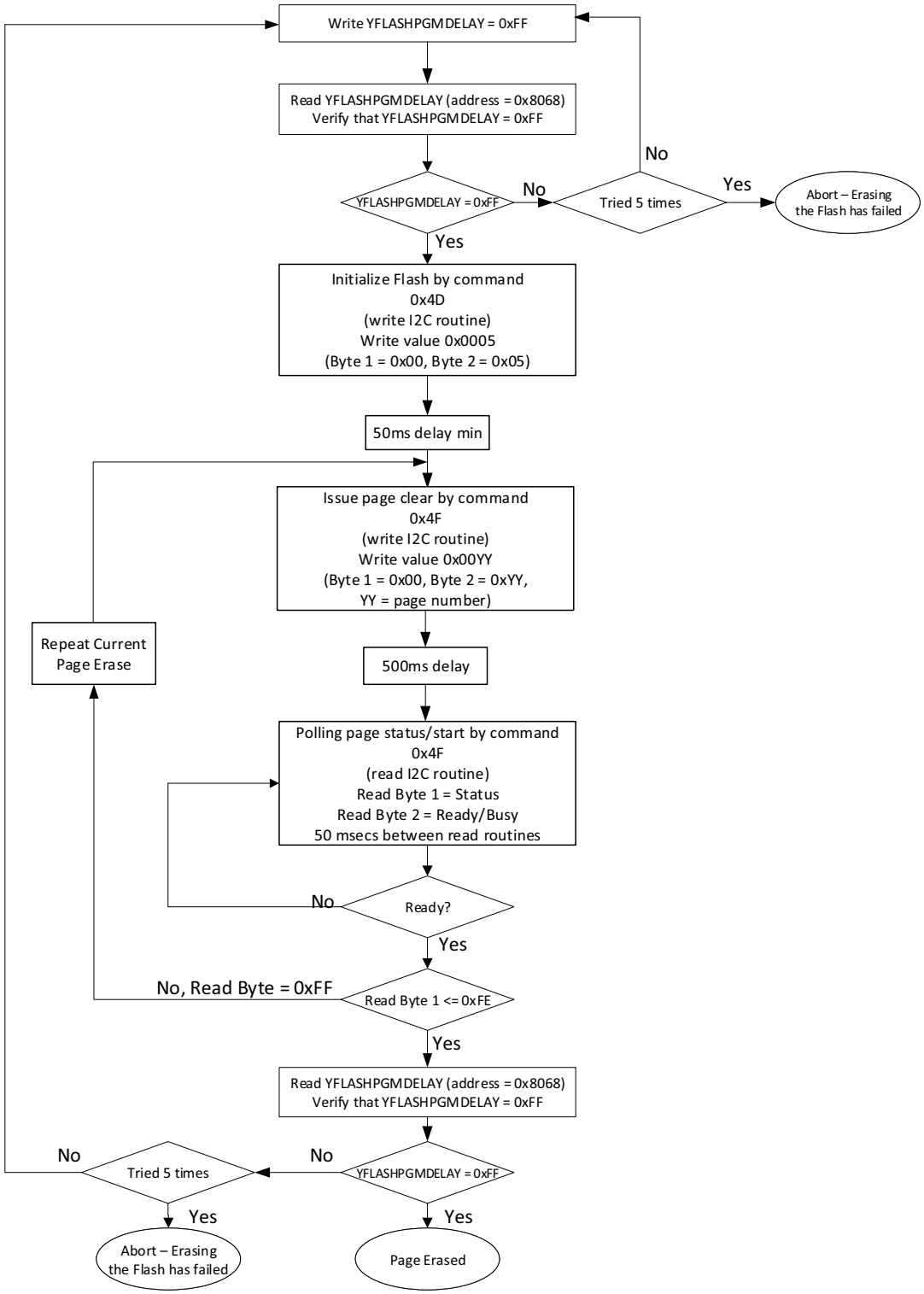


Note: The device will generate an interrupt once the page clear command has completed (refer to system command section of this document). Therefore, if a customer configuration forwards an interrupt to an IO, the host can be informed about this event without polling the status of page clear as explained in the flow chart above.

Figure 3: Clear Pages

Erase Pages

Follow the Diagram below:



Note: The device will generate an interrupt once the page erase command has completed (refer to system command section of this document). Therefore, if a customer configuration forwards an interrupt to an IO, the host can be informed about this event without polling the status of page erase as explained in the flow chart above.

Figure 4: Erase Pages

Program Flash Image

Follow the diagram below:

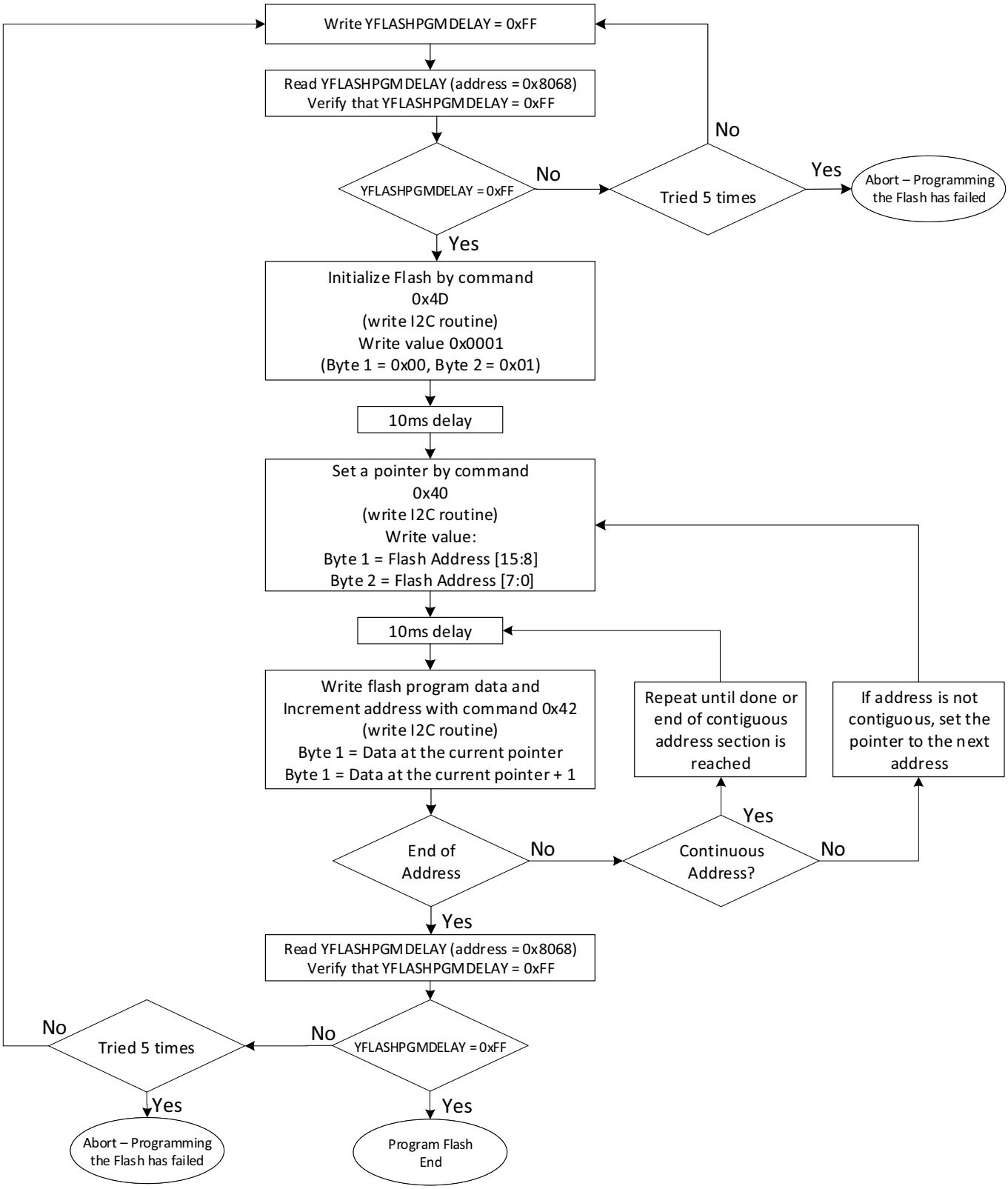
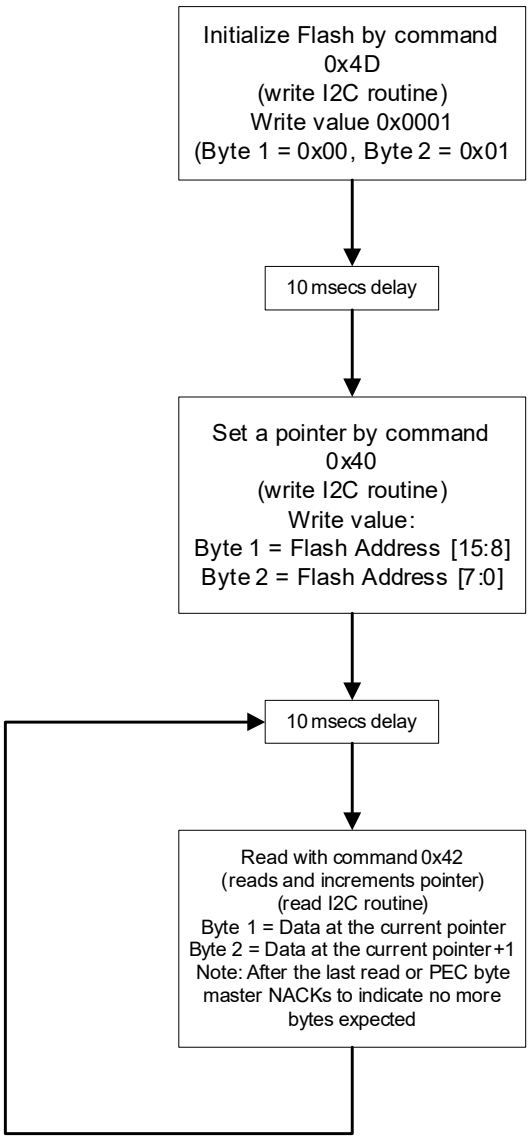


Figure 5: Program Flash Image

Read Flash

Follow the diagram below:



Note: the flash is organized in 16 bit words so make sure you set the pointer to even addresses only.

We recommend verify (read back) after each write. If verify fails, set the pointer back to the failing location and try programming it again.

After programming is complete and verified, reset the device to see it take effect. If the new configuration changed the I2C address, start using the new address now.

Only pages 0 to 6 are for customer use. Clearing, erasing or programming of other Flash pages will cause the chip to be permanently disabled.

Figure 6: Read Flash

Appendix A

The following is a sample code demonstration on how to use the **PWR_SET_VOLTAGE_CHx** commands to dynamically change output voltage levels.

Assuming I2C commands are sent and received with PEC:

```
i2c_write(SlaveAddress, StdCmd, WrByte1, WrByte2)
(RdWord1, RdWord2) = i2c_read(SlaveAddress, StdCmd)
```

Changing output voltage level on the native step size:

```
Ch = 0         # 0=CH1, 1=CH2, 2=CH3, 3=CH4
Vout = 3.3
SetValue = Vout / 0.0125 #(PRESCALE is accounted for, Vout is in Volts but make sure it is
not outside the selected range)

i2c_write(chip_addr, 0x20 + Ch, (SetValue >> 8), (SetValue & 0xFF))
```

Changing output voltage level if in PWM mode of operation and fine step size adjustment is desired (knowing the prescale value is assumed):

```
#prescale = 0.0025    # (0.6V - 1.6V)
#prescale = 0.0050    # (0.6V - 3.2V)
prescale = 0.0100    # (0.6V - 5.5V)

Ch = 0         # 0=CH1, 1=CH2, 2=CH3, 3=CH4
Vout = 3.3

#determine what the quantized base voltage will be in the prescale resolution
vref = int( float(Vout) / (prescale * 5) ) * (prescale * 5)

#figure out the fine adjustment difference from the base voltage in the prescale resolution
FineAdj = min(int(round((float(Vout) - float(vref)) / prescale)), 0xF)
VrefValue = min(int(float(Vout)/0.0125), 0x1B8) #5.5v or 0x1B8 is max value
#put the values in the correct spots
SetValue = (FineAdj << 12) + VrefValue

i2c_write(chip_addr, 0x20 + Ch, (SetValue >> 8), (SetValue & 0xFF))
```

When using this command, the FineAdj value will be executed immediately and then the output will ramp to the new value based on the user defined ramp rate. This may be undesired depending on the previous values of the FineAdj and the VrefValue. It would be then possible to do two commands: first setting the VrefValue keeping the same FineAdj value and then another command just changing the FineAdj.

MaxLinear, Inc.
5966 La Place Court, Suite 100
Carlsbad, CA 92008
760.692.0711 p.
760.444.8598 f.

www.maxlinear.com