

PI33xx-xx-EVAL1

ZVS Switching Regulators

I²C™ Digital Interface Guide



Contents	Page
Introduction	1
Program Capability	1
Fault Telemetry	1
I²C Communication Compatibility	1
Interface Software Tool	1
USB Interface Hardware	2
Addressing & Register Mapping	3
Software Installation and Operation	4
Fault Telemetry Command Structure	7
Configuration Programming	10
Programming Initial Conditions	16
Configuration Examples	16
Sync Delay	16
Enable Polarity	18
Setting Kill Bit	19
Error Messages	20

Introduction

The PI33xx-xx I²C digital interface guide provides an overview of how to read and write to the user accessible registers within the PI33xx-xx series. This guide details use of the Vicor I²C digital interface software tool coupled with a USB based I²C hardware interface device. Information is also provided for I²C communication protocol using generic tools and/or embedded hardware.

Program Capability

The PI33xx-xx devices provide an I²C digital interface that enables the user to program the EN pin polarity (from high to low assertion) and input switching frequency synchronization phase/delay. Margining the output is also available via the I²C interface.

All settings are one-time programmable except for the V_{OUT} margining, which settings are stored in volatile memory only and not retained by the device.

Fault Telemetry

The PI33xx-xx can report when a fault is detected via the I²C interface. Reported faults include:

- Input Undervoltage Lockout (UVLO)
- Input Overvoltage Lockout (OVLO)
- Output Voltage Overvoltage (OVP)
- Overtemperature Protection (OTP)
- Slow Current Limit
- Fast Current Limit
- V_{CC} (internal to the SiP) Undervoltage

I²C Communication Compatibility

The PI33xx-xx is hardware-compatible with the NXP I²C bus specification Version 2.1 Standard Mode (100kHz) January 2000 for all bus timing and voltage resistors levels up to 5.5V maximum. The PI33xx-xx is configured as an I²C child device with no internal bus pull up or pull down.

Interface Software Tool

The Buck GUI software tool offered by Vicor allows quick access to the one time programmable parameters and user adjustable parameters. The one time user programmable parameters include enable (EN) polarity, synchronization (SYNCl) delay and edge trigger. User-adjustable parameters include output margin (MRGN) level percentage and the ability to clear the fault (FLT) register. This software also provides the user with the ability to read and decode fault telemetry information.

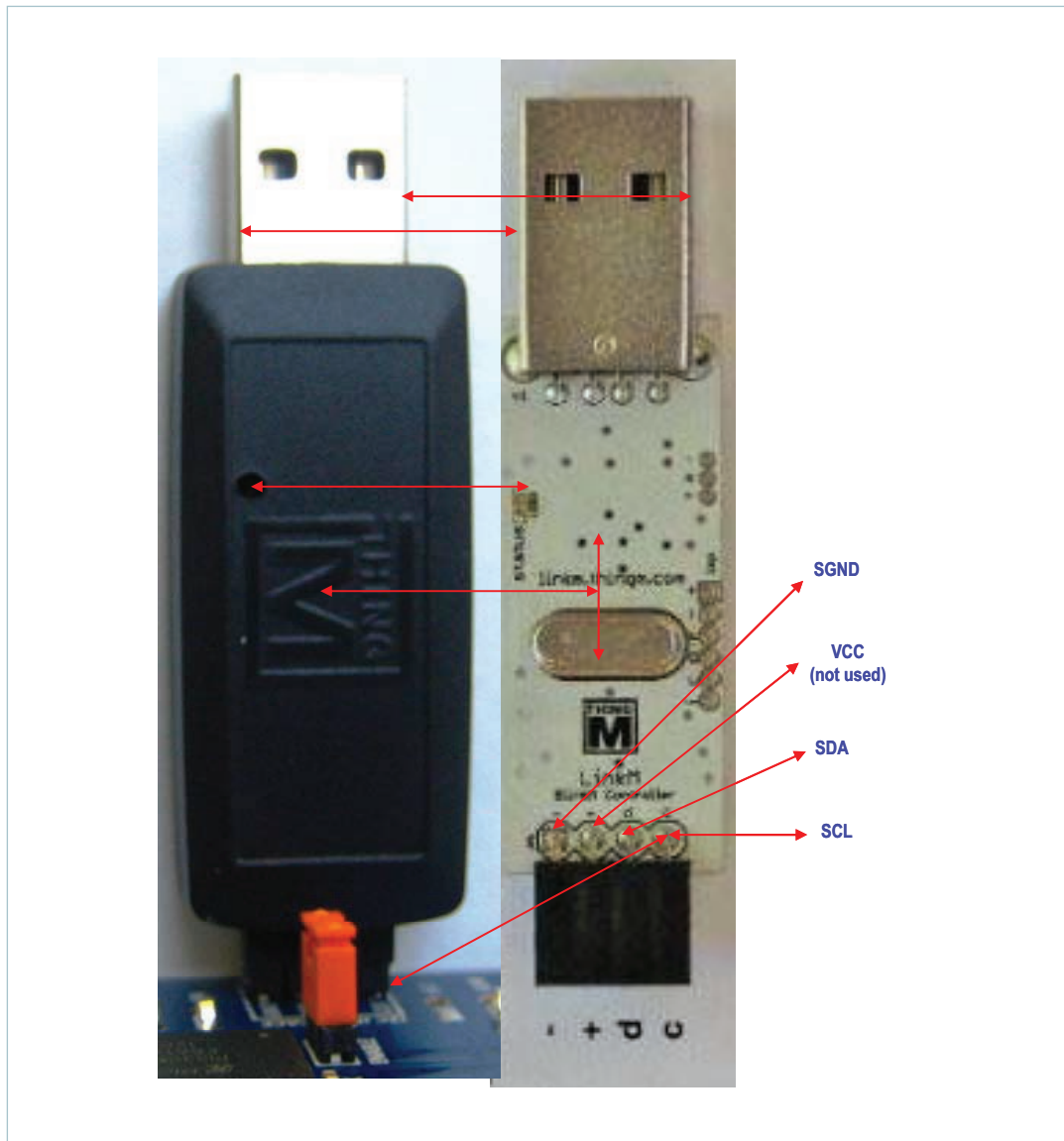
The Buck GUI operates under the Windows™ 7 or XP (Service Pack 3) environment. Java™ version 1.5 or greater 32-bit version is required. This software has been tested using Windows 7, 32 bit and 64 bit and Windows XP with service pack 3.

USB Interface Hardware

The Buck GUI software is designed to work with the LinkM™ USB to I²C™ interface device manufactured by ThingM. Information about the LinkM can be found at the ThingM [website](http://www.thingm.com). A Windows-based PC is required with a dedicated USB port. The use of USB expansion ports, especially those that are heavily loaded, is not recommended.

The LinkM interface can be mounted directly to the PI33xx-xx evaluation board using a straight 4-pin header (see Figure 1). Users who wish to connect the LinkM in this manner will need to use a male-to-female USB extension cable. This method works very well and is the preferred method. Other users may wish to probe the I²C bus or connect to it with ball clips. For this option, the LinkM data sheet describes the recommended connectors to build such a cable.

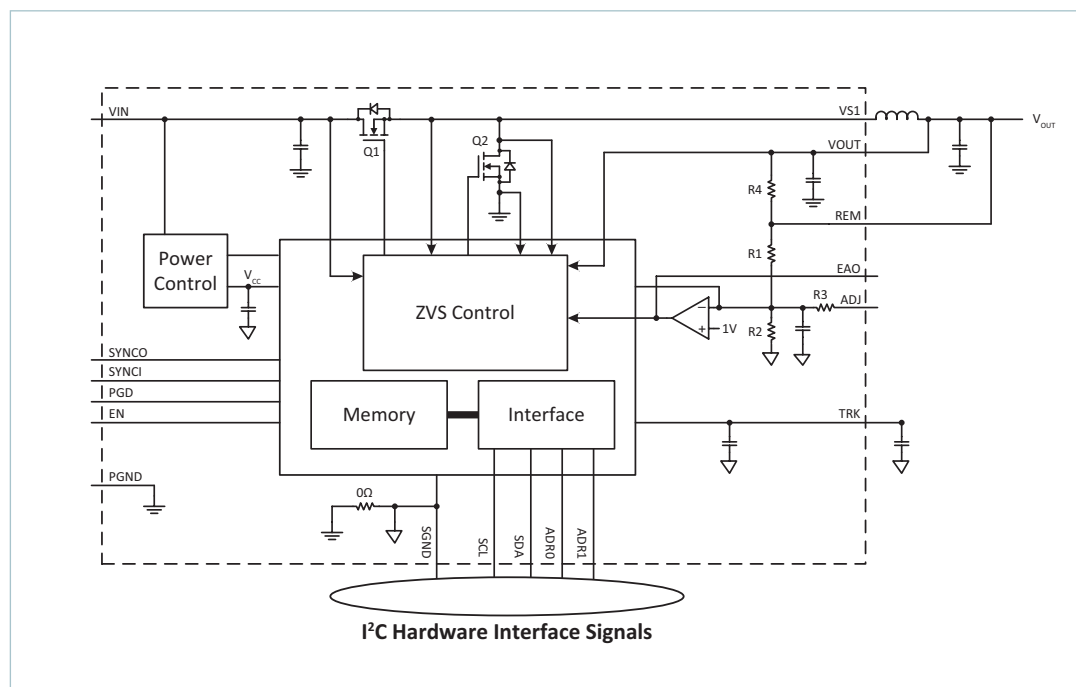
Figure 1
LinkM with case removed



Addressing & Register Mapping

Figure 2 shows the PI33xx-xx I²C™ hardware interface. The PI33xx-xx supports floating addressing so that two address lines allows for up to eight programmable addresses. The address is 7 bit with the read/write bit not included. Table 1 shows the address range that can be achieved using all possible combinations of ADR0 and ADR1. Bits A[6] – A[3] are fixed internally and may not be changed.

Figure 2
PI33xx-xx block diagram showing I²C hardware interface



The least significant 3 bits; A[2] – A[0], will assume the values in the table based on the decoding of ADR1 and ADR0. A zero or one indicates the logic strength of the bit and "NC" indicates that the pin is floating or not connected. The HEX column indicates the final address in hexadecimal, while the DEC column is the decimal address value.

Table 1
Addressing options
(See Table 3 where
 V_{CC} = PI33xx-xx fixed,
internal 5.1V bias rail)

A[6]	A[5]	A[4]	A[3]	A[2]	A[1]	A[0]	R/W	ADR1	ADR0	HEX	DEC
1	0	0	1	0	0	0	X	0	0	48	72
1	0	0	1	0	0	1	X	0	NC	49	73
1	0	0	1	0	1	0	X	0	1	4A	74
1	0	0	1	0	1	1	X	NC	0	4B	75
1	0	0	1	1	0	0	X	NC	NC	4C	76
1	0	0	1	1	0	1	X	NC	1	4D	77
1	0	0	1	1	1	0	X	1	0	4E	78
1	0	0	1	1	1	1	X	1	1 or NC	4F	79

Table 2
User-accessible registers

Name	Address HEX	# Bits	Description
TSTMDE[2:0]	18	3	Test mode register used for burning bits into non-volatile memory
MRGN[3:0]	19	4	Volatile register for output voltage margining
FLT[7:0]	1A	8	Fault Register read only
FREG_CLR	1B	0	Register for clearing the Fault Register-writing address clears register
ENA_POL	20	1	EN pin polarity programming bit
SYN[3:0]	21	4	SYNC programming. SYN[3] = SYNC polarity
KBIT2	22	1	User kill bit for SYN[3:0] and ENA_POL – can not be reversed; register is write only, can not be read

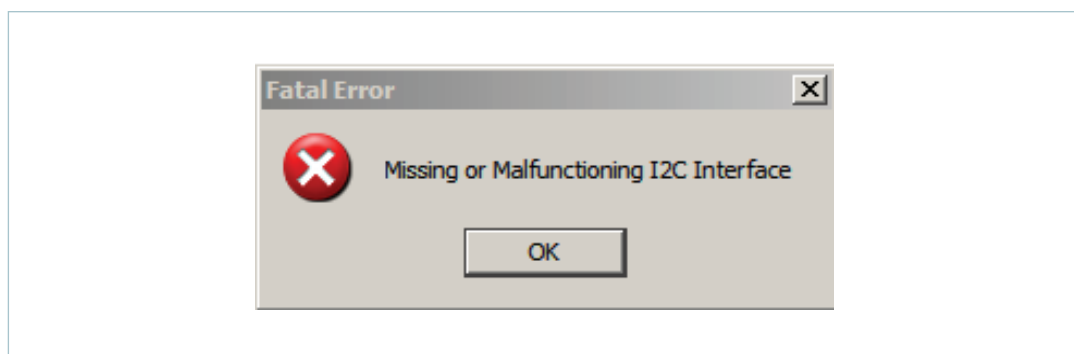
Table 3
I²C™ port specifications

Parameter	Conditions	Min	Typ	Max	Units
SCL, SDA V _{IH}	Rising, 3.3V & 5V bus compatible	2.1			V
SCL, SDA V _{IL}	Falling 3.3V & 5V bus compatible			1.5	V
SCL, SDA input current (sink to hold open pins low)	At 5V	1.8	5	10	μA
SDA V _{OL}	3mA			0.4	V
ADDR0, 1V _{MID}	V _{CC} = PI33xx-xx fixed, internal 5.1V	0.4V _{CC}		0.6V _{CC}	V
ADDR0, 1V _{IH}	V _{CC} = PI33xx-xx fixed, internal 5.1V	0.8V _{CC}			V
ADDR0, 1V _{IL}	V _{CC} = PI33xx-xx fixed, internal 5.1V		0.1V _{CC}	0.2V _{CC}	V
ADDR0, 1 hysteresis			0		mV
ADDR0, 1 Output resistance from ½ V _{CC}	0.3V _{CC} < ADDR _x < 0.7V _{CC} V _{CC} = PI33xx-xx fixed, internal 5.1V		6k		kΩ
ADDR0, 1 Output resistance from ½ V _{CC}	ADDR _x < 0.2V _{CC} , ADDR _x > 0.8V _{CC} V _{CC} = PI33xx-xx fixed, internal 5.1V		70k		kΩ

Software Installation and Operation

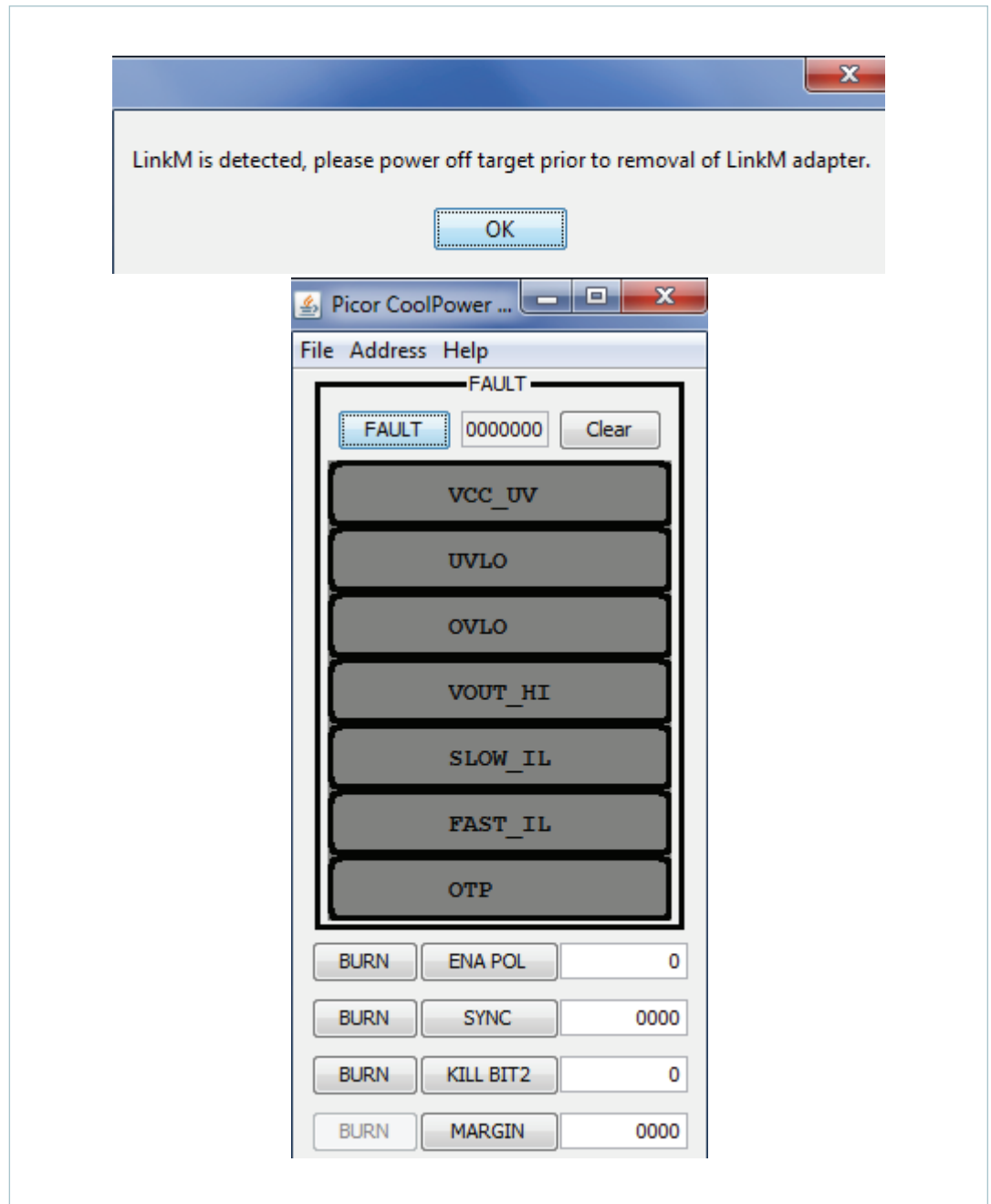
Users downloading the [Buck GUI](#) should save the compressed file in a desired directory. The user should unzip the compressed file and run the executable installation file "PicorBuckGuiFull_setup.exe". The installer will then install the Buck GUI in either the default Programs directory or to a directory specified by the user. Once the software is installed, the user should next plug in the LinkM™ into a dedicated USB port. The LinkM will detect and use a valid driver without the need of installing one. If the LinkM device is not detected, the user should consult the LinkM [homepage](#). If the LinkM is not installed, the Buck GUI will not start and an error message will be generated like the one shown in Figure 3.

Figure 3
Error message due to missing LinkM upon software start



Successful installation of the LinkM Interface should result in a successful start up screen of the Buck GUI as shown in Figure 4. First a dialog box will pop up indicating proper detection of the LinkM. When the user clicks the "OK" button, that box will disappear. The Buck GUI program screen will appear as shown in the bottom half of Figure 4.

Figure 4
LinkM™ detection followed by
Buck GUI main screen



After the user observes a successful main screen like the one shown in Figure 4 and the LinkM is installed to the target board as shown in Figure 1, it is possible to use the Buck GUI to monitor and configure the PI33xx-xx. The target PI33xx-xx unit should be powered on using the proper input voltage value and enabled by the user. Failure to apply input voltage will result in the error message shown in Figure 28 (see Buck GUI error messages section) upon pressing any of the “soft” buttons on the screen. The LinkM should not be removed or installed with the power already applied to the PI33xx-xx. Always power down the PI33xx-xx prior to installing or removing the LinkM.

The following information is intended to provide the user with a description of the various “soft” buttons and user menus available in the Buck GUI.

The “File” menu provides “Exit” as the only option available when left clicking on it. If the user chooses “Exit”, a dialog box will pop up to allow the user to confirm their selection. If “Yes” is chosen, the GUI will close and Buck GUI will terminate. If “No” is selected, the user will be returned to the main program without exiting the GUI.

The “Help” menu provides the current revision of the Buck GUI. There is no interactive user help or search utility provided at this time, although there may be expanded help facilities in the future build releases of the software.

The “Address” menu will produce a drop down dialog box for the user to be able to select the decimal address of the PI33xx-xx determined by the ADR0 and ADR1 decodes in Table 1. For example, if the PI33xx-xx unit has both ADR0 and ADR1 floating or not connected, the child address will be 76 decimal or 4C. Configuring ADR0 and ADR1 allow for eight addressable locations from 72d to 79d or 48h to 4Fh.

The “FAULT” button allows the user to read the PI33xx-xx fault telemetry information. The fault register is 8 bits wide with the most significant bit set to logic 0 always. The gray indicator panels for each fault on the Buck GUI display will illuminate bright red to indicate the decoded fault(s) for the user, so they don’t have to refer to Table 4 for the decoded value. In addition, the binary value of the register will be displayed in the “FAULT” window.

Table 4
Fault register assignments

FLT[7]	FLT[6]	FLT[5]	FLT[4]	FLT[3]	FLT[2]	FLT[1]	FLT[0]
0	OTP	FAST_IL	SLOW_IL	VOUT_HI	OVLO	UVLO	VCC_UV

The PI33xx-xx fault logging only occurs when the controller is in normal operation mode. After a fault is detected and assessed, the fault is latched into the register so long as the controller V_{CC} is active and above the minimum threshold. The controller will take the appropriate action to protect the PI33xx-xx and system based on the type of fault. If for example, the input voltage is high enough to power the controller but below the minimum under voltage lock out threshold, the controller will prevent the unit from enabling and remain in a low-power state. Since it has not entered operate mode, the UVLO fault will not be logged. If the input voltage is higher than the undervoltage lockout threshold but drops to zero, the fault will remain latched as long as there is V_{CC} to the controller. Once the controller V_{CC} dips below the minimum value, the fault data will not remain valid. The fault register will be cleared upon power on reset. If V_{CC} remains after a logged fault, the fault register must be cleared in order to log any new events. A description of the logged faults is as follows:

FLT[0] – VCC_UV: if this bit is set, it indicates that the internal power supply for the PI33xx-xx has gone into undervoltage.

FLT[1] – UVLO: if this bit is set, the indication is that the input voltage decreased below the undervoltage lock outthreshold (UVLO) at some point while the unit was in operate mode. The UVLO threshold is defined as the minimum value required for a PI33xx-xx to be able to meet all specified parameters of operation.

FLT[2] – OVLO: if set indicates that the input voltage exceeded the over voltage lockout threshold (OVLO) at some point while the unit was in operate mode. The OVLO threshold is that value where the input voltage is too high for a PI33xx-xx to be able to meet all specified parameters of operation.

FLT[3] – VOUT_HI: if this bit is set, it indicates that the error amplifier input was higher than it should be for the programmed output voltage during operate mode, indicating that the output voltage may be too high.

FLT[4] – SLOW_IL: if this bit is set, it indicates that the error amplifier output was at the positive rail for more than 1 ms during operate mode. This means that the load current demand was higher than the maximum output current available from the PI33xx-xx.

FLT[5] – FAST_IL: if this bit is set, it indicates that the peak current in the output inductor was higher than the maximum peak current allowed during operate mode. It is an indicator of output short circuit or inductor failure.

FLT[6] – OTP: if this bit is set, it indicates that the PI33xx-xx internal temperature exceeded the maximum temperature for safe operation during operate mode and that the PI33xx-xx shut down to prevent damage.

The “Clear” button allows the user to clear the PI33xx-xx fault telemetry information.

The “ENA POL” button allows the user to read the EN polarity by left clicking on it. Changing the polarity can be accomplished by entering the desired value in the dialog box and then left clicking the “BURN” button. The Buck GUI will then instruct the user for the remaining steps. The burn function is irreversible and requires careful consideration. For that reason, further details can be found in the section “Configuration Programming”.

The “SYNC” button allows the user to read the SYNC polarity and delay settings by left clicking on it. Changing the polarity can be accomplished by entering the desired value in the dialog box and then left clicking the “BURN” button. The Buck GUI will then instruct the user for the remaining steps. The burn function is irreversible and requires careful consideration. For that reason, further details can be found in the section “Configuration Programming”.

The “KILL BIT2” button allows the user to prevent further changes to any programmed register value when used in conjunction with the “BURN” button. The burn function is irreversible and requires careful consideration. For that reason, further details can be found in the section “Configuration Programming”.

The “MARGIN” button operates using volatile memory. Any changes made to this register are dynamic and will change as soon as the command is sent. The user simply enters the value they wish to margin and then clicks margin. Any value sent to this register will be lost as soon as power is removed. Note that margining down 20% or more in one step may cause a VOUT_HI fault, which is a normal condition.

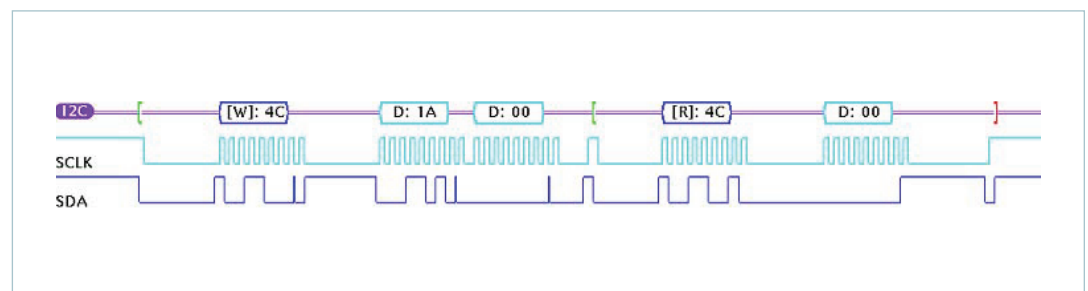
Table 5
PI33xx-xx margin
register assignments

MARGN[3:0]	% V _{OUT}
1100	-20
1101	-15
1110	-10
1111	-5
0000	0
1000	5
1001	10
1010	15
1011	20

Fault Telemetry Command Structure

The PI33xx-xx command structure is always a 2-byte write and a 1-byte read following the start condition and the base address when using Buck GUI and the LinkM interface. The second data byte is either 00h or additional data based on the command being sent. Figure 5 is an example of an I²C™ bus command to read the fault register and it can be implemented using the Buck GUI by left-clicking the “FAULT” button. In a similar fashion, the actual fault register can be read from a generic I²C™ device as shown in Figure 6 by sending the commands shown followed by stop bits. The I²C device used for sending the command required a R/W bit for addressing so the address of 4Ch was sent as 98h.

Figure 5
Fault register read no fault



In Figure 5, the address of 4Ch was sent followed by the first data byte. “4Ch” represents the base address of the PI33xx-xx as determined by ADRO and ADR1. The first data byte is the address decode of the internal register to be read (or written); in this case, read. “1Ah” is address of the fault register as shown in Table 2.

Figure 8
*I²C™ bus capture input
 undervoltage fault register
 read using Buck GUI
 and LinkM™*

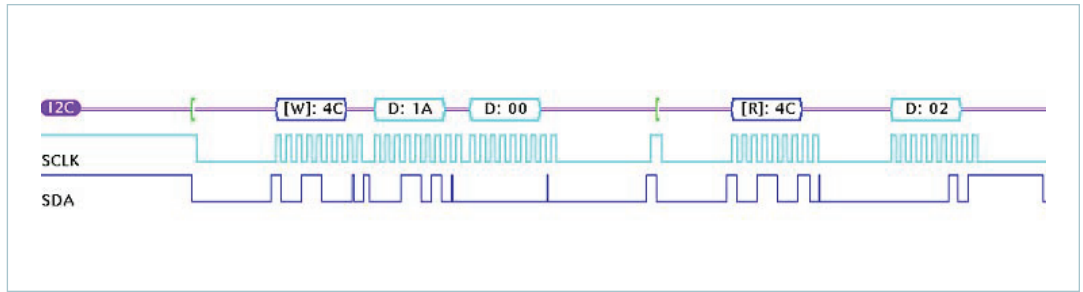


Figure 9
*Buck GUI capture of input
 undervoltage fault*

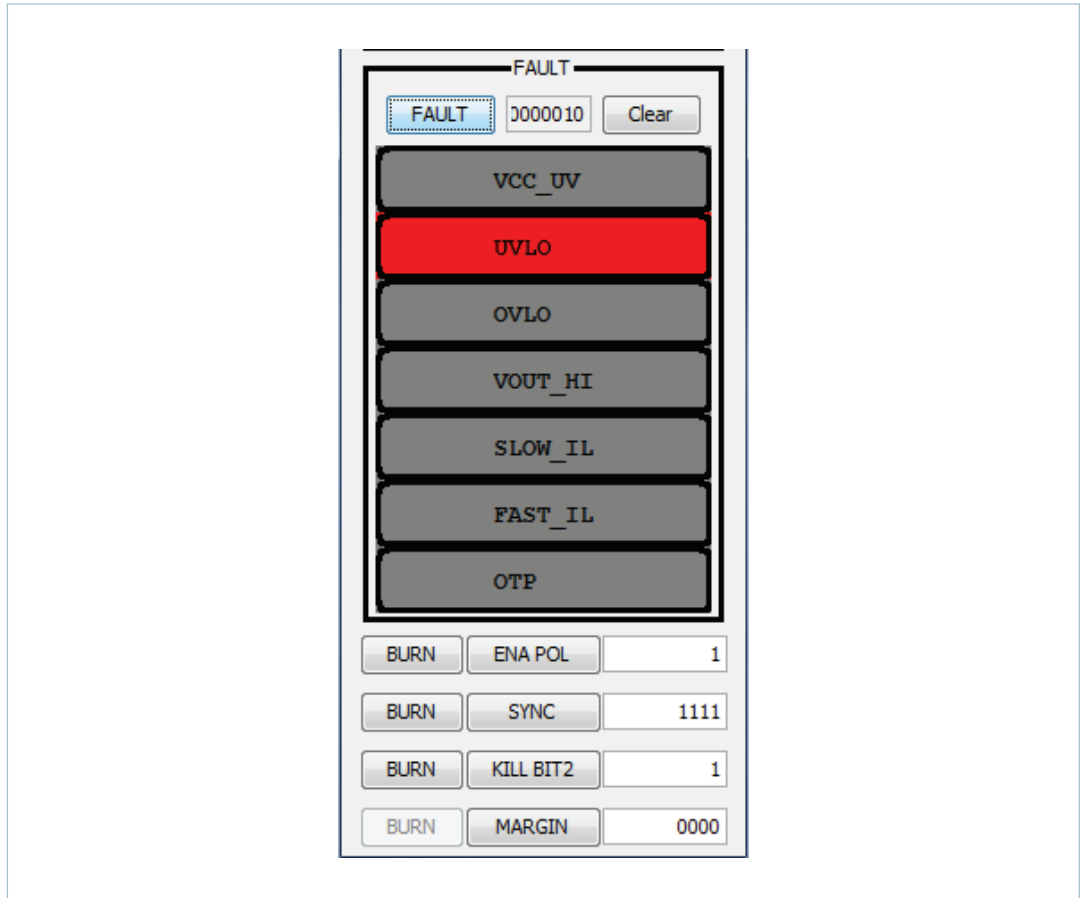


Figure 10 shows several more common faults displayed by the Buck GUI along with the corresponding I²C bus capture of each event. The first fault SLOW_IL, the second fault is VOUT_HI and the third fault is OVLO.

Figure 11 shows the I²C bus capture of dynamically margining the PI33xx-xx output voltage down -20% by entering "1100" in the "MARGIN" dialog box and clicking the "MARGIN" button. Note that "0Ch" is sent in the second data byte, which is the margin value.

Figure 10
Buck GUI capture of
common faults with
corresponding I²C™ Data

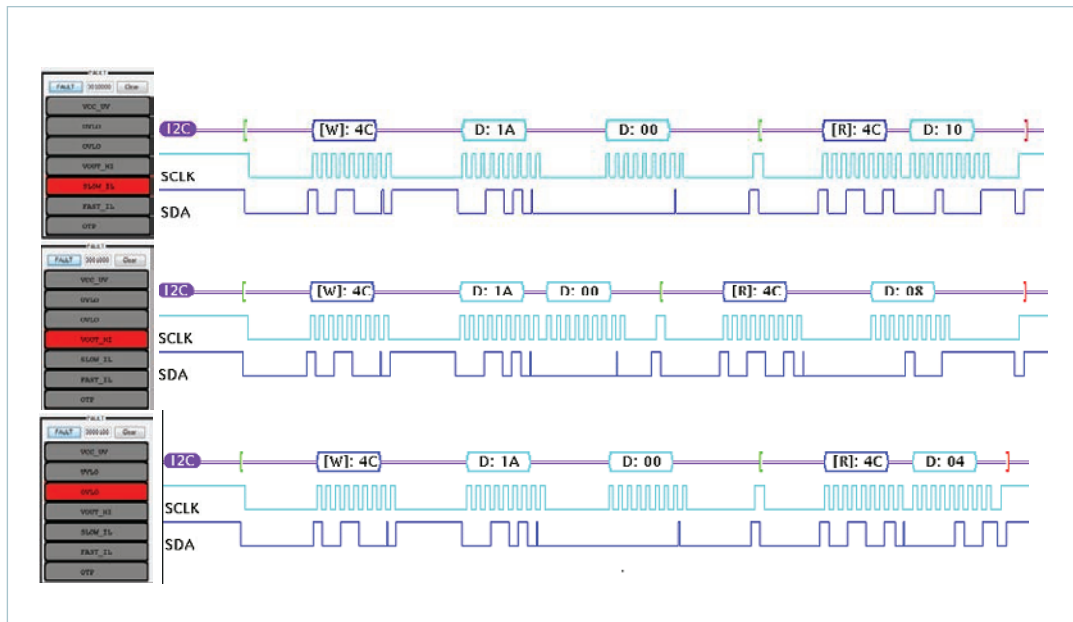
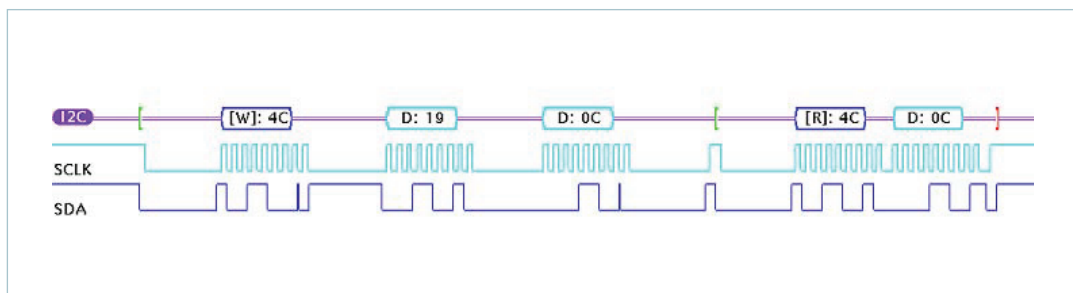


Figure 11
I²C bus capture



Configuration Programming

The PI33xx-xx has three programming registers that can be changed by the user as a one-time change only. Of each register, any bit that is not set may be set. The programming requires burning internal fuse links as permanent memory. Once they are burned there is no way to change or reset the settings back to default. For this reason, **it is critical to be sure the settings are correct prior to clicking on the burn button**. The three registers are ENA_POL, SYN[3:0] and KBIT2. ENA_POL is programmed by using the “ENA POL” soft key and dialog box. SYN[3:0] is programmed by the “SYNC” soft key and dialog box. KBIT2 is programmed using the “KILL BIT2” soft key and dialog box.

The programmable values for the SYN[3:0] register are shown in Table 6. Bits 0 through 2 define the delay setting between a synchronizing signal (rising or falling edge as selected) applied to the SYNCI input and the SYNCO output rising edge, applied as a fraction of the main system clock period (MP). The most significant bit SYN[3] determines which edge trigger occurs. A “1” indicates rising edge and a “0” indicates falling edge. As an example, if the MP value is 1μs, and the SYNC dialog box reads “1101”, the programmed delay is 500ns from the rising edge of the SYNCI input. The actual programmed value does not take effect until the bits are actually burned so there is no way to measure the delay without actually burning the value in. For this reason, the user should review the waveform plots in Figures 12 – 19 to understand the timing relationship prior to attempting to burn the values into the register. The value of main system clock can be found in the PI33xx-xx data sheet for each part number.

To understand the relationship between the synchronization timing and the PI33xx-xx operation, an overview of the PI33xx-xx power train timing is shown in Figure 12. There are three main timing states in the ZVS buck topology, T1, T3 and T4 as defined below:

T1: T1 defines the start of a power cycle when the clamp switch has opened and the zero voltage switching resonant action has started, followed by the turn on of Q1 and continuing until Q1 turns off. During T1, current ramps up to a positive peak value, charge is delivered to the output capacitor and energy is stored in the output inductor.

T3: Q1 has turned off, Q2 has turned on and energy stored in the inductor is delivered to the load. As the current in the inductor passes through zero, energy is stored in the inductor to provide zero-voltage switching for the next time Q1 is required to turn on.

T4: After Q2 turns off, the clamp switch turns on to preserve the energy stored in the inductor to be used for the next T1, while clamping V_S to V_{OUT} . At the end of T4 and beginning of T1, the clamp switch opens and the parasitic capacitance of Q1 and Q2 resonates with the output inductor to provide zero-voltage switching.

The rising edge of SYNCO defines the beginning of T1 and can be observed in Figure 13. The rising edge of SYNCO is synchronized to SYNCI rising or falling edge as programmed plus any delay desired. Figure 13 shows the timing relationship of SYNCO, SYNCI and the phase node V_S with default timing and phase delay. Note that the node V_S in the schematic of Figure 12 is called V_{S1} in Figure 13 and can be observed on Channel 3 of the plot.

Figure 12
ZVS buck topology
timing diagram

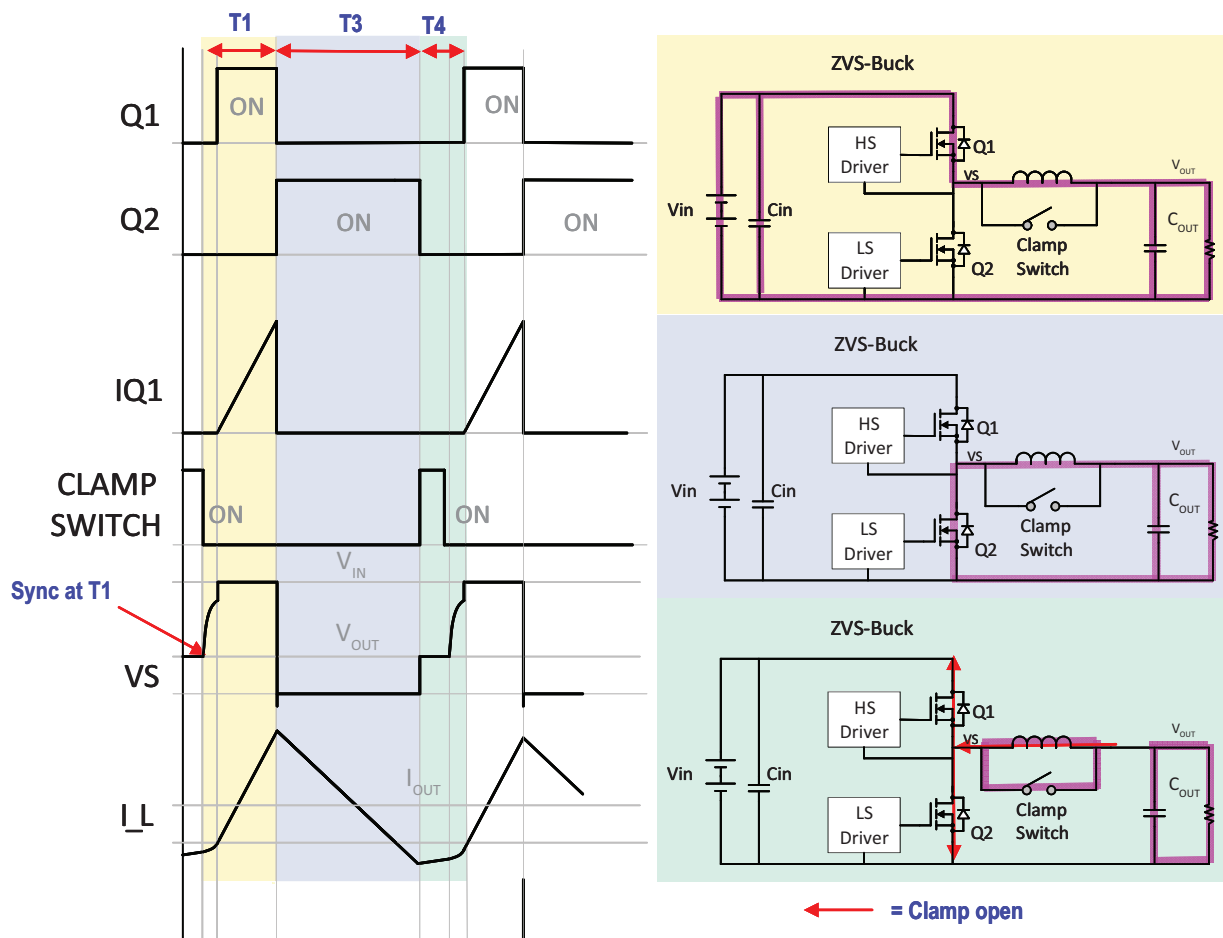
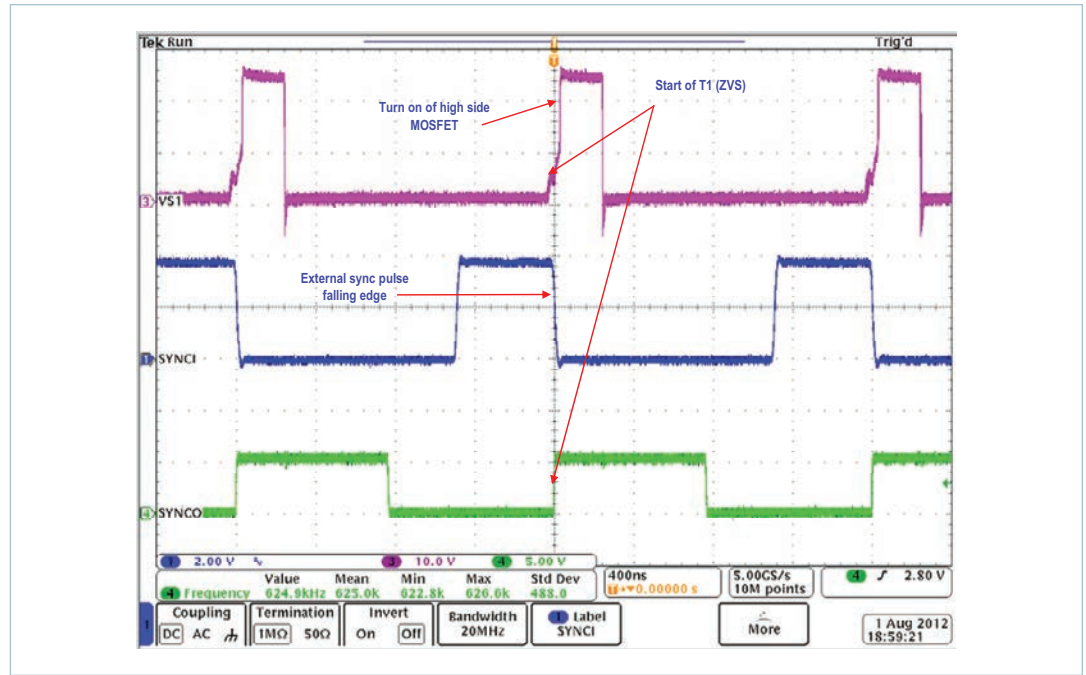


Figure 13
 PI33xx-xx ZVS buck sync to
 T1 timing relationship
 where VS = Ch3 VS1



SYN[3]	SYN[3:0]	SYN[2:0]	SYNC Delay
Polarity Bit	Control Bits		
0 = Falling	000		NONE
1 = Rising	001		3/4MP
	100		2/3MP
	101		1/2MP
	110		1/3MP
	111		1/4MP

The ENA_POL and KBIT2 registers are single bit registers. A “0” in the ENA_POL register is the default value. This means the PI33xx-xx will enable when the ENA pin is floating or logic high. Pulling down ENA pin will disable the PI33xx-xx. If a “1” is programmed into the ENA_POL register, the ENA polarity reverses. If the ENA pin is floating or logic low, the PI33xx-xx is enabled. Pulling the ENA pin high will disable the PI33xx-xx.

The KBIT2 register may only be written with a “1” entered into the dialog box. This register can not be read, it is write only. Burning this register will prevent making any more changes to any register, even if there are unused bits available to burn.

Figure 14
 PI33xx-xx
 SYNC = "1000" timing

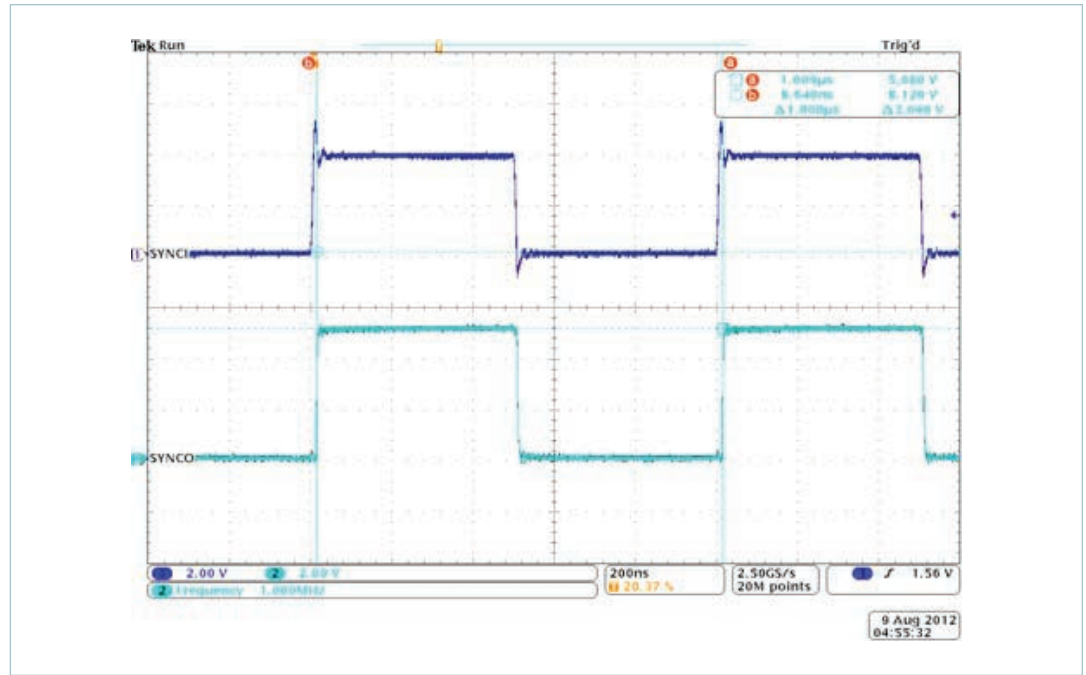


Figure 15
 PI33xx-xx
 SYNC = "1111" 1/4MP

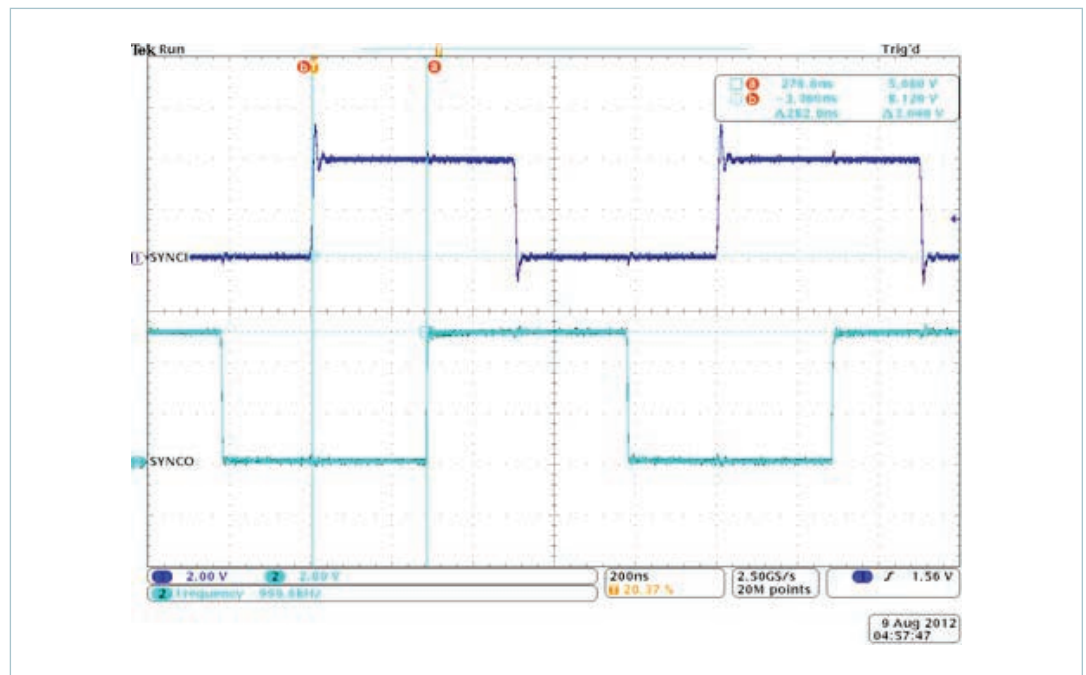


Figure 16
 PI33xx-xx
 SYNC = "1110" 1/3MP

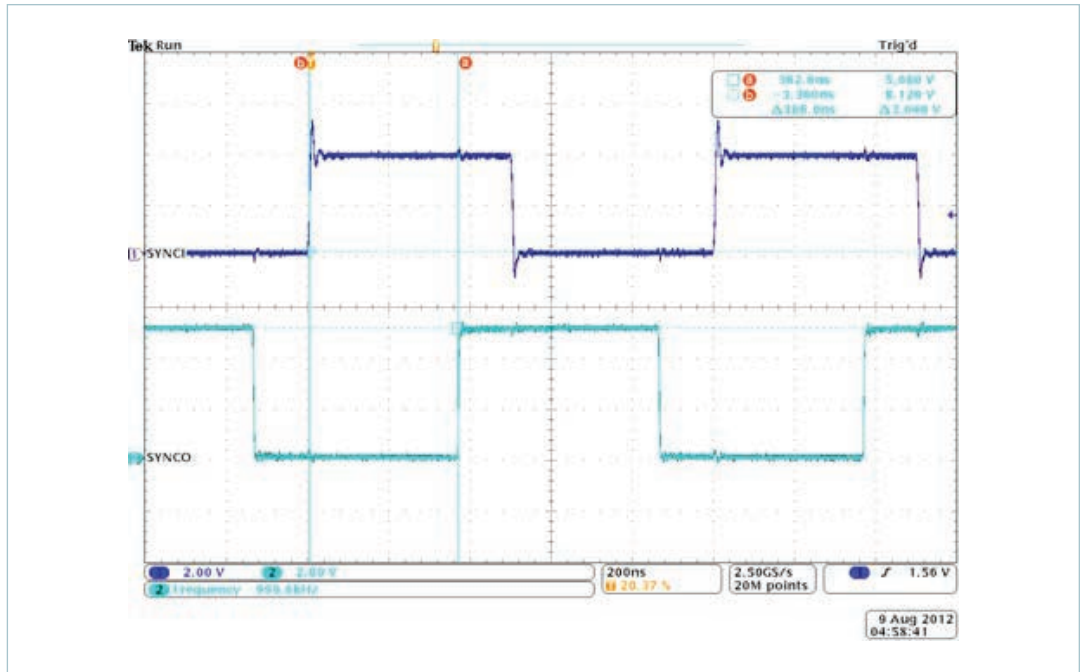


Figure 17
 PI33xx-xx
 SYNC = "1101" 1/2MP



Figure 18
 PI33xx-xx
 SYNC = "1100" 2/3MP

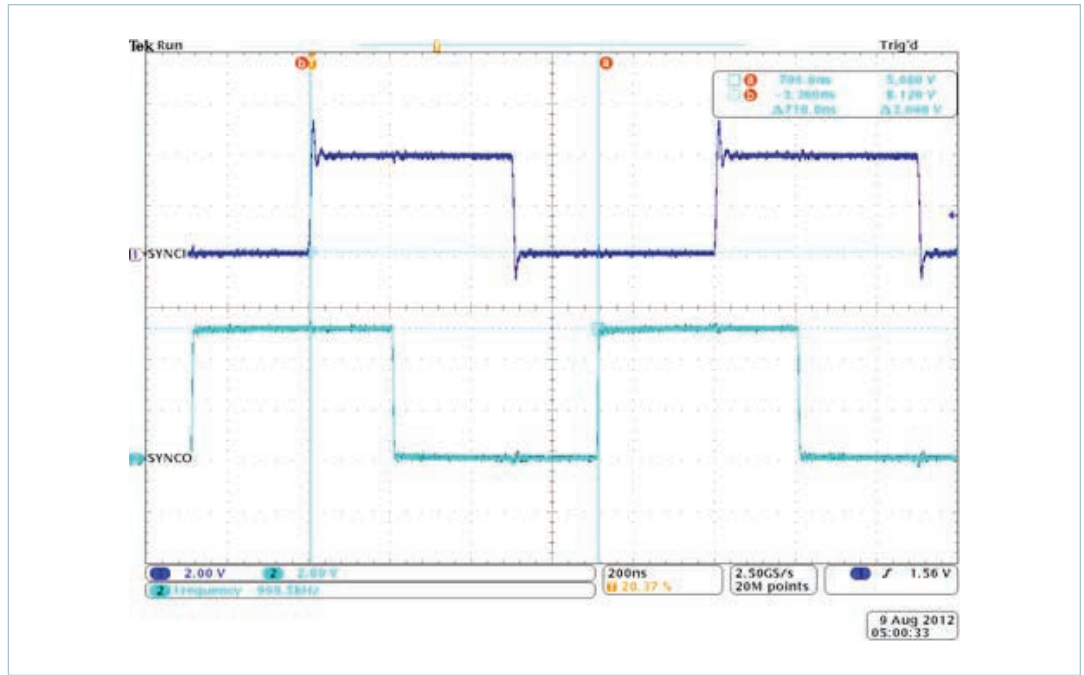
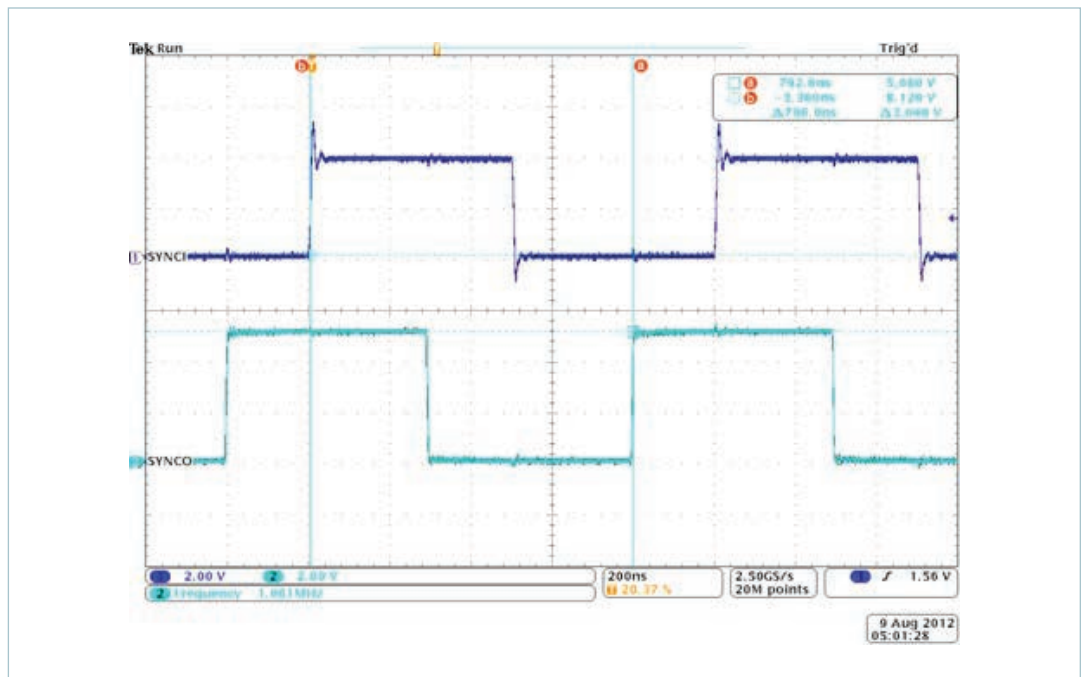


Figure 19
 PI33xx-xx
 SYNC = "1001" 3/4MP



Programming Initial Conditions

Once the user has selected the configuration for the timing register, the burn process can be initiated. To begin this process, the user must be ready to be able to enable and disable the PI33xx-xx, since this is required to enter the burn mode and actually burn the bits. The **initial conditions** to begin the burn process are as follows:

1. Proper input voltage should be applied to the target PI33xx-xx.
2. The unit can be either enabled and producing output voltage or disabled with the output voltage at zero. The preferred method is disabled since in some cases multiple units may depend on proper phase delay for best performance. **The user must adhere to the instructions from the GUI to ensure proper burn results.**

The burn process consumes power from the internal bias for the controller, so only a single bit can be burned at a time. Buck GUI takes care of this for the user and will generate the appropriate number of bus commands to ensure each bit is burned correctly. Users that write their own software or operate from a different GUI will need to consider this. In addition, power should not be removed from the PI33xx-xx during the burn process. The Buck GUI will prompt the user when to enable and disable the target. Enabling and disabling is done using the EN pin. One important note to consider is that if the enable polarity has been changed already due to a separate operation, the new enable polarity must be used when enabling and disabling. The procedure along with the Buck GUI commands, for changing the SYNC[3:0] register so that the synchronization polarity is falling edge and the delay is 3/4MP is as follows.

Configuration Example – Sync Delay

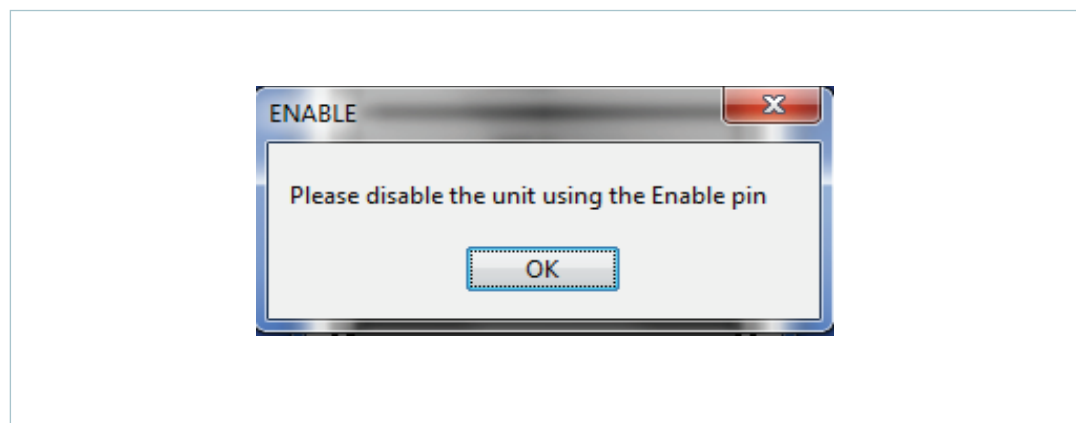
Type “0001” into the SYNC dialog box. Left-click on the “BURN” button.

Figure 20
SYNC “0001” burn phase 1
while unit enabled or disabled

Time	Repeat	Start	Address	Data	Missing Ack
-250.4µs			[W]4C	2000	
256.0µs	X		[R] 4C	00	
38.33ms			[W]4C	2100	
38.77ms	X		[R] 4C	00	
69.39ms			[W]4C	2200	
69.69ms	X		[R] 4C	FF	
100.3ms			[W]4C	2200	
100.7ms	X		[R] 4C	FF	
132.3ms			[W]4C	1805	
132.8ms	X		[R] 4C	05	

The first four writes and reads ensure that the three user registers are cleared. Then the test mode register is selected at address 18h and the test mode 05h is entered. Next, the user is prompted to disable the unit using the enable pin. The target can be enabled or disabled prior to being prompted to disable. See Figures 20 and 21.

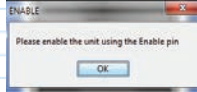
Figure 21



When the user disables the target, another prompt will provide notification to enable the target again as shown in Figure 22. When the unit is enabled, the output voltage will remain low as if the PI33xx-xx was still disabled. This is normal. The Buck GUI will next write "01h" into the SYN[3:0] register.

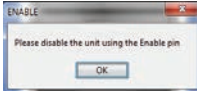
Figure 22
SYNC "0001" burn phase 2
unit must be enabled

Time	Repeat Start	Address	Data	Missing Ack
-251.2µs		[W]4C	2101	
126.8µs	X	[R] 4C	01	

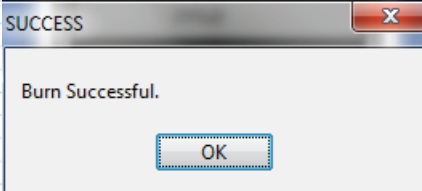


The next step prompts the user to disable the target again. After clicking "OK", Buck GUI will reset from test mode 5 and then clear the other user registers as shown in Figure 23. Buck GUI will display a dialog box to inform the user of the successful burn completion. When the user enables the PI33xx-xx, it should power up normally and the burned in changes shall take effect.

Figure 23
SYNC "0001" burn phase 3
unit must be disabled



Time	Repeat Start	Address	Data	Missing Ack
-312.8µs		[W]4C	1800	
251.6µs	X	[R] 4C	00	
197.7ms		[W]4C	2000	
198.2ms	X	[R] 4C	00	
228.7ms		[W]4C	2101	
229.3ms	X	[R] 4C	01	
260.2ms		[W]4C	2200	
260.7ms	X	[R] 4C	FF	



The user may read the register that was just programmed by clicking on the "SYNC" button. Buck GUI should show the new burned in values read back from the SYN[3:0] register as shown in Figure 24. If an error message occurs or the incorrect results are obtained, refer to the section titled "PI33xx-xx Error Messages" for more information.

Figure 24
Read SYN[3:0]
after burning "0001"

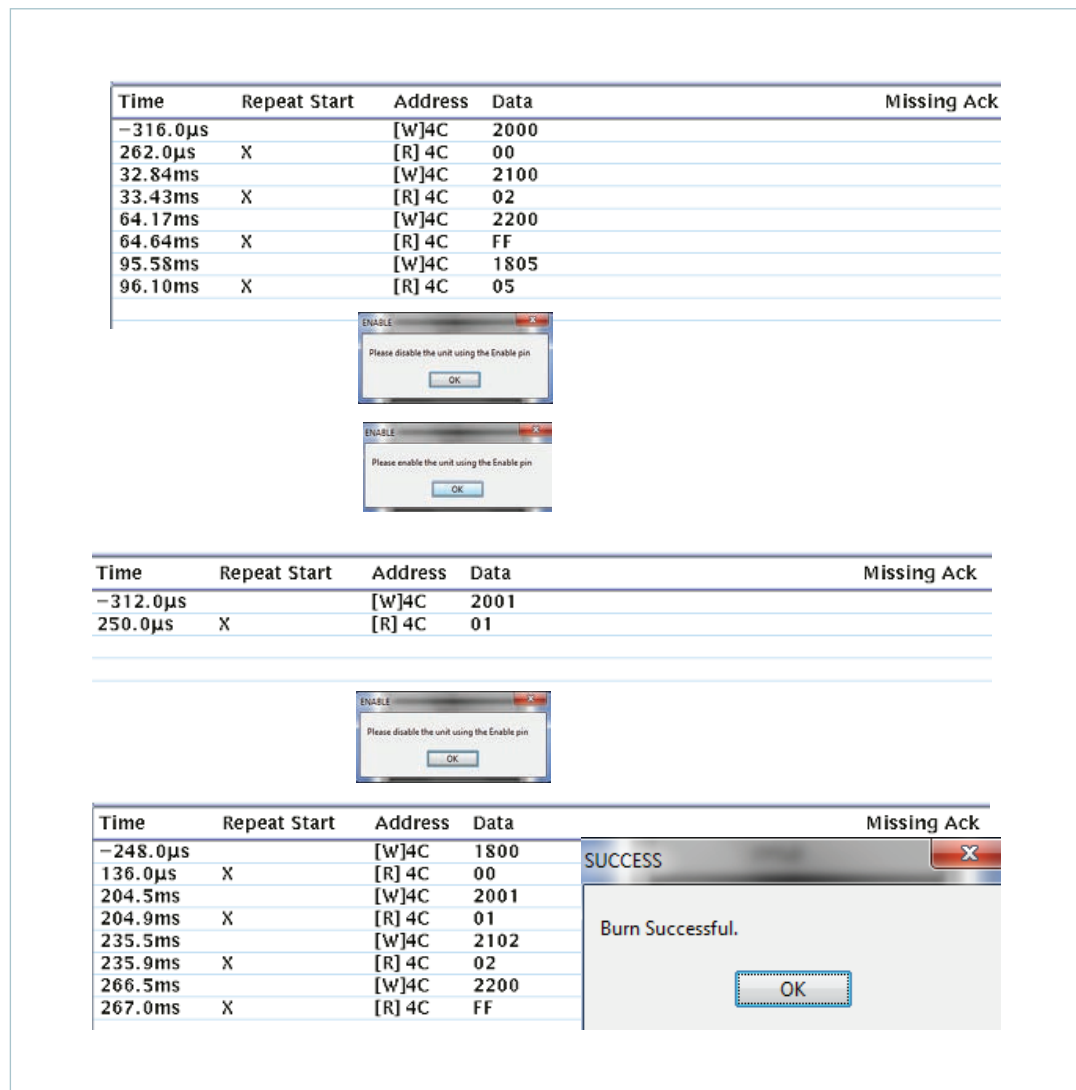
Time	Repeat Start	Address	Data	Missing Ack
-245.2µs		[W]4C	2101	
269.6µs	X	[R] 4C	01	

It is important to note that the data sent here is the value shown in the dialog window. The actual read is the burned in value of the register. Sending "4Ch 21h FFh" or "4Ch 21h 00h" would also work properly.

Configuration Example – Enable Polarity

Figure 25 shows the complete procedure for changing the enable polarity along with the I²C™ bus capture of all of the commands required to make the changes using Buck GUI or some other generic software. Since the enable polarity is not changed until AFTER the burn is completed, always use the default polarity of the signal before the burn process to enable and disable the target PI33xx-xx. It should be pointed out that the SYN[3:0] register had already been programmed to “0010” on the PI33xx-xx target prior to the enable polarity change. Note that the first two bus transactions occur when the target PI33xx-xx is enabled and the final transaction occurs when it is disabled. Failure to follow this exact sequence will prevent successful storage of the desired configuration settings. If an error message occurs or the incorrect results are obtained, refer to the section titled “Error Messages” for more information.

Figure 25
Changing EN_POL register to
“1” with SYN[3:0]
programmed to “0010”

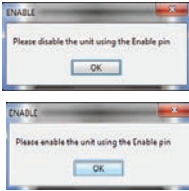


Configuration Example – Setting Kill Bit


The procedure for setting the KBIT2 register to prevent any further programming is outlined in sequence in Figure 26. The SYN[3:0] register has been programmed to “1111” and the ENA_POL register has been programmed to “1” in advance. Like the previous procedures, the first two bus transactions occur while the PI33xx-xx is enabled and the final transaction occurs while the PI33xx-xx is disabled. After completing this step, no further changes can be made to the PI33xx-xx. It is very important to double check all settings before clicking the “BURN” button. If the user makes a mistake in a setting after clicking on the “BURN” button, the instructions for enabling and disabling the unit that are prompted by Buck GUI should be IGNORED by leaving the target enabled and clicking “OK” to each pop-up dialog box. This will prevent the burn from occurring and give the user another chance to correct the mistake. Buck GUI will report the failed burn with a pop up dialog box.

Figure 26
Setting KBIT2 register after setting ENA_POL Register to “1” with SYN[3:0] programmed to “1111”

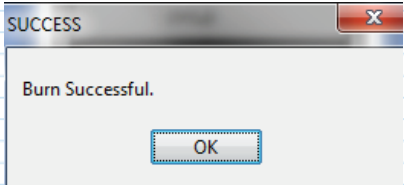
Time	Repeat Start	Address	Data	Missing Ack
-320.0µs		[W]4C	2000	
266.0µs	X	[R] 4C	01	
32.67ms		[W]4C	2100	
33.26ms	X	[R] 4C	0F	
63.68ms		[W]4C	2200	
64.19ms	X	[R] 4C	FF	
95.08ms		[W]4C	1805	
95.59ms	X	[R] 4C	05	



Time	Repeat Start	Address	Data	Missing Ack
-316.0µs		[W]4C	2201	
376.0µs	X	[R] 4C	FF	
30.10ms		[W]4C	2201	
30.54ms	X	[R] 4C	FF	



Time	Repeat Start	Address	Data	Missing Ack
-250.0µs		[W]4C	1800	
188.0µs	X	[R] 4C	00	
198.3ms		[W]4C	2001	
198.9ms	X	[R] 4C	01	
229.9ms		[W]4C	210F	
230.3ms	X	[R] 4C	0F	
260.7ms		[W]4C	2201	
261.2ms	X	[R] 4C	FF	

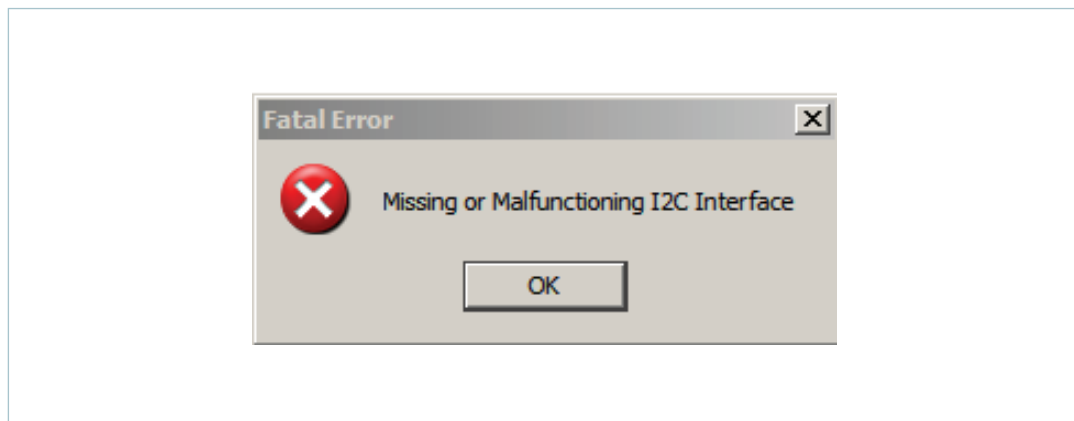


Error Messages

Missing or Malfunctioning I²C™ Interface Error

The error message shown in Figure 27 occurs only at the first boot of the Buck GUI each time the program is executed. During program initialization, the software determines the existence of the LinkM™ USB interface. If the hardware or software interface from the computer USB port to the LinkM is either not connected, missing or malfunctioning, Buck GUI will display this error message. If some error occurs with this interface during program execution, a different error message will be displayed. This error **ONLY** applies to the USB portion of the LinkM and **does not indicate any problem with the PI33xx-xx or the LinkM I²C bus interface.**

Figure 27
Fatal error message at
Buck GUI boot

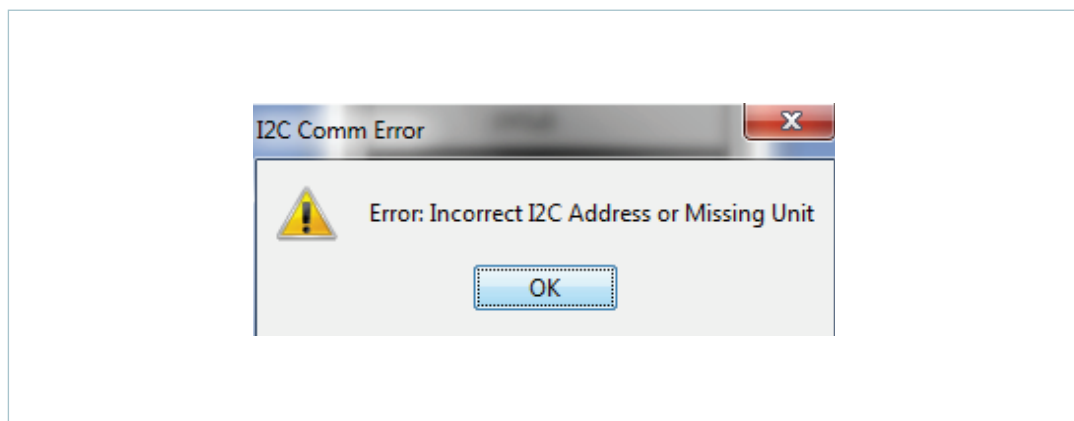


Incorrect I²C Address or Missing Unit Error

The error message shown in Figure 28 occurs during Buck GUI program execution if either the PI33xx-xx or LinkM I²C interface is malfunctioning. It will also occur during program execution if the PC to USB or LinkM USB hardware or software interface is malfunctioning. If the USB portion of the interface is at fault, Buck GUI can not recover and will continue to display this error even if the USB interface problem is corrected. Buck GUI will need to be closed and restarted to reestablish USB connectivity with Buck GUI. If the I²C interface is the problem, the error will clear without restarting Buck GUI once the problem is corrected. **It is very important to note that a failure of the critical interface to either the LinkM or the PI33xx-xx during a programming operation can result in the incorrect or unintentional programming of parameters.** If the user encounters this error during a programming operation:

1. Remove input power to the PI33xx
2. Exit Buck GUI
3. Restore power to the PI33xx-xx
4. Restart Buck GUI

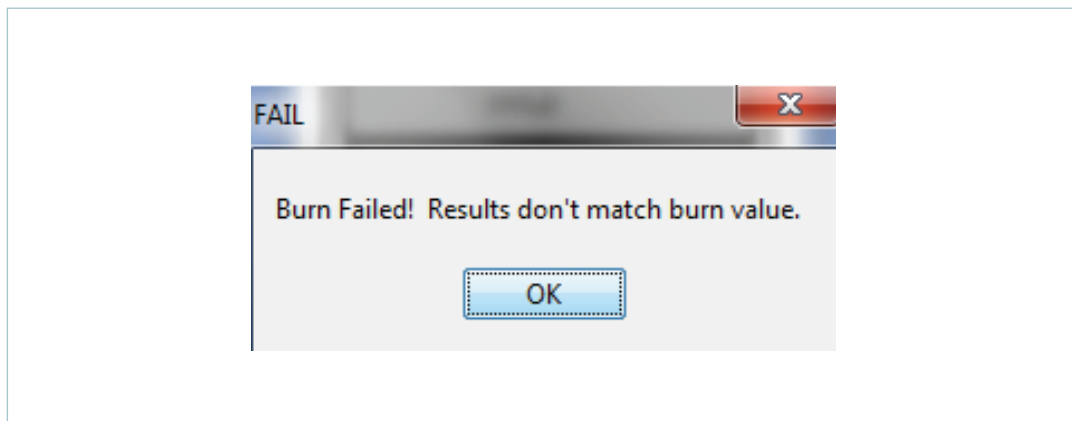
Figure 28
I²C bus or USB error message
during Buck GUI execution



Burn Failed Error

The error message shown in Figure 29 occurs only during a failed program attempt. Here, the value that is programmed into the associated register does not match the value that was entered into that registers dialog box prior to executing the programming operation. The most likely cause of this error is a missed enable or disable step during the programming procedure. Other causes could be a communication error that would also prompt Buck GUI to report the appropriate error. **If this error occurs after receiving the “Missing or Malfunctioning I²C™ Interface Error”, it is critically important to ensure that you shut down power to the PI33xx-xx and exit the Buck GUI prior to attempting to reprogram again. Adhering to this step can possibly avoid programming the wrong register values.**

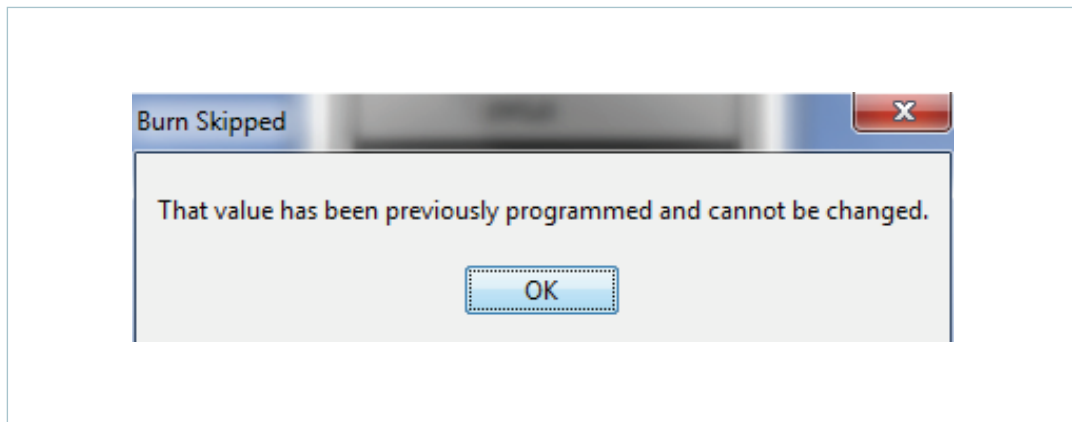
Figure 29
Burn failed error message



Burn Skipped Error

The error message shown in Figure 30 occurs only when the user has attempted to program a value into an associated register that was previously programmed. The burn process will be terminated and the user will be returned to the main menu.

Figure 30
Burn skipped error message



Limitation of Warranties

Information in this document is believed to be accurate and reliable. HOWEVER, THIS INFORMATION IS PROVIDED "AS IS" AND WITHOUT ANY WARRANTIES, EXPRESSED OR IMPLIED, AS TO THE ACCURACY OR COMPLETENESS OF SUCH INFORMATION. VICOR SHALL HAVE NO LIABILITY FOR THE CONSEQUENCES OF USE OF SUCH INFORMATION. IN NO EVENT SHALL VICOR BE LIABLE FOR ANY INDIRECT, INCIDENTAL, PUNITIVE, SPECIAL OR CONSEQUENTIAL DAMAGES (INCLUDING, WITHOUT LIMITATION, LOST PROFITS OR SAVINGS, BUSINESS INTERRUPTION, COSTS RELATED TO THE REMOVAL OR REPLACEMENT OF ANY PRODUCTS OR REWORK CHARGES).

Vicor reserves the right to make changes to information published in this document, at any time and without notice. You should verify that this document and information is current. This document supersedes and replaces all prior versions of this publication.

All guidance and content herein are for illustrative purposes only. Vicor makes no representation or warranty that the products and/or services described herein will be suitable for the specified use without further testing or modification. You are responsible for the design and operation of your applications and products using Vicor products, and Vicor accepts no liability for any assistance with applications or customer product design. It is your sole responsibility to determine whether the Vicor product is suitable and fit for your applications and products, and to implement adequate design, testing and operating safeguards for your planned application(s) and use(s).

VICOR PRODUCTS ARE NOT DESIGNED, AUTHORIZED OR WARRANTED FOR USE IN LIFE SUPPORT, LIFE-CRITICAL OR SAFETY-CRITICAL SYSTEMS OR EQUIPMENT. VICOR PRODUCTS ARE NOT CERTIFIED TO MEET ISO 13485 FOR USE IN MEDICAL EQUIPMENT NOR ISO/TS16949 FOR USE IN AUTOMOTIVE APPLICATIONS OR OTHER SIMILAR MEDICAL AND AUTOMOTIVE STANDARDS. VICOR DISCLAIMS ANY AND ALL LIABILITY FOR INCLUSION AND/OR USE OF VICOR PRODUCTS IN SUCH EQUIPMENT OR APPLICATIONS AND THEREFORE SUCH INCLUSION AND/OR USE IS AT YOUR OWN RISK.

Terms of Sale

The purchase and sale of Vicor products is subject to the Vicor Corporation Terms and Conditions of Sale which are available at: (<http://www.vicorpower.com/termsconditionswarranty>)

Export Control

This document as well as the item(s) described herein may be subject to export control regulations. Export may require a prior authorization from U.S. export authorities.

Contact Us: <http://www.vicorpower.com/contact-us>

Vicor Corporation

25 Frontage Road
Andover, MA, USA 01810
Tel: 800-735-6200
Fax: 978-475-6715
www.vicorpower.com

email

Customer Service: custserv@vicorpower.com
Technical Support: apps@vicorpower.com

©2018 – 2020 Vicor Corporation. All rights reserved. The Vicor name is a registered trademark of Vicor Corporation.
I²C™ is a trademark of NXP Semiconductors.
LinkM™ is a trademark of ThingM Corporation.
Windows™ is a trademark of Microsoft Corporation.
Java™ is a trademark of Oracle Corporation.

All other trademarks, product names, logos and brands are property of their respective owners.