UCD8220EVM Digitally Managed Isolated Half-Bridge

Contents

List of Figures

List of Tables

1 Overview

The UCD8220EVM is a digitally managed 100-W power converter that generates a regulated 12-V output from an unregulated input voltage ranging from 36 V to 75 V. The EVM uses a half-bridge topology in voltage-control mode. A microcontroller controls the high-level functions of the power converter. It has the following specifications:

Specifications					
Parameter	**Test Conditions**	**Min**	**Typ**	**Max**	**Unit**
Input voltage		36	48	75	V
Output voltage			12		V
Load current		0		8.33	A
Switching frequency	Out ripple		600		kHz
Output ripple voltage			20		mVpp
Efficiency	V_{IN} = 48; I_O = 5A		90.5%		

The UCD8220 is a digitally managed analog PWM controller with integrated MOSFET drivers. It supports various power-supply topologies, and can be managed by a standard microcontroller.

The MSP430 in this application (the MSP430F155) also provides communication to a system host via the standard PMBus protocol. The system is designed to be able to have multiple power supplies managed by the same system host via the PMBus. A typical system block diagram is shown in Figure 1.

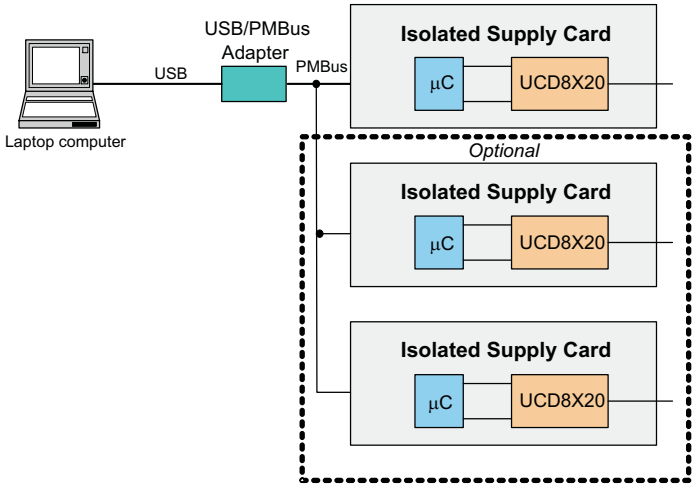


Figure 1. System Block Diagram

In this reference system, a Windows™ PC can be used to configure, manage, and control a number of power supplies based on the UCD8220 and MSP430 devices. The PC uses a USB link to communicate with an external adapter (available from TI) that translates USB transactions from the host or PC to the PMBus. The PMBus connects the master (the adapter) with the slaves (power supplies).

Each power supply in this example supports the following features:

- VIN = 36 V to 75 V, Vout = 12 V
- PMBus Communication
- JTAG Realtime firmware debugging with TI's Code Composer Studio™ Essentials firmware-development environment
- INPUT Voltage UVLO, OVLO with adjustable hysteresis
- Maximum Duty-cycle limiting with Volt*second Clamp
- Output-voltage monitoring
- Pulse-by-pulse current limiting

2 PMBus

The Power-Management Bus (*PMBus*) is an open-standard protocol that manages a power supply via communication over a digital communications bus. The communication bus between the UCD8220 evaluation board and the host was developed to comply with the PMBus standard.

This section describes the physical and electrical interface of the PMBus and the PMBus commands the UCD8220 evaluation board can support.

2.1 Overview

The UCD8220 evaluation board uses the MSP430F155 microcontroller from TI's MSP430 family as the digital controller for the power system.

The PMBus electrical interface is implemented through the general I/O (GPIO) pins and the universal synchronous/asynchronous receive/transmit (USART) peripheral interface on the MSP430F155. It is configured as follows:

Table 1. PMBus Implementation

PMBus Signal	MSP430F155 Signal
Control	GPIO pin configured as input
Write Protect	Not supported
Alert	GPIO pin configured as output
Clock	Serial clock (SCL) pin on a USART peripheral in I^2C mode
Data	Serial data pin (SDA) on a USART peripheral in I^2C mode

If a controller other than MSP430F155 is chosen, changes in both the hardware connection and the firmware must be made. The above configuration is an example of how to set up the PMBus using a microcontroller with a built-in I^2C-control module. The related firmware design is described in Section 4.

For a microcontroller without an I^2C interface, both PMBus clock and data signals can be configured by GPIO pins. However, an I^2C low-level driver must be implemented in the firmware.

The PMBus protocol is intended to cover a wide range of power-system architectures and converters. Not all available PMBus features, functions and commands are supported by the UCD8220 evaluation board and firmware.

The supported features, functions and commands are:

- Read/write word
- Block read process call.
- Packet error checking (PEC) for read commands.
- Use the CONTROL signal to turn the power converter on and off.

For details on the commands supported by the firmware, see Section 2.3.

The unsupported features that can be added by updating the firmware include:

- **Group command protocol.** This is used to send commands to multiple PMBus devices. Because the evaluation board only has one such device, the firmware does not support this command protocol.
- **Read/write-byte and block-write commands.**
- **Procedure to notify the host that a command is rejected.** The MSP430F155 microcontroller does not allow the firmware to send a NACK signal in I^2C slave mode. One workaround is to send an ALERT to the host.
- **PEC for write commands.** Because no procedure is implemented to notify the host if a received command is wrong, the write-with-PEC command is not included.

The unsupported features that can be added by updating both firmware and hardware include:

- **Write protect (WP).**
- **Program physical address through pins.** The current physical address is defined only through firmware.

For more information about PMBus operation, see the PMBus Power Management Protocol Specification.

2.2 Electrical Interface

As described above, the electrical interface of the PMBus uses the GPIO and the USART peripheral interface on MSP430F155. The USART peripheral is configured as an I^2C slave using 7-bit addressing.

In slave mode, both transmit and receive operations are controlled automatically by the I^2C hardware module.

When the host transmits data, the serial data bits received at the SDA are shifted in by the clock from the master device. Each byte received is automatically acknowledged. There is no way to generate a NACK condition for received data.

When data is transmitted from the MSP430F155 to the host, the transmitter shifts the serial data out on SDA using the clock from the master device.

In slave mode, the I^2C module can generate an interrupt when:

- START condition is detected
- STOP condition is detected
- Master requests data (slave-transmit mode)
- New data received (slave-receive mode)

Each event sets a specific value in the interrupt vector.

The PMBus firmware was developed based on the features of the MSP430F155 I^2C module. For more details about the I^2C module, see the *MSP430x1xx Family User's Guide* (SLAU049).

2.3 PMBus Commands

Table 2 lists the commands that the firmware supports. Data is in PMBus DIRECT format unless otherwise noted.

Table 2. PMBus Commands

Function	PMBus Command Name	Command Code	SMBus Transaction Name	Mfr. Default(1)	Notes
Read/Write Commands					
DMAX Limit	MAX_DUTY	0x32	R/W word	0x19	Maximum number of PWM counts. Represents duty cycle.
CLF Counts Limit	MFR_SPECIFIC_01	0xD1	R/W word	0x7D00	Number of CLF count-limit flags from UCD8220 before stopping conversion. If CLF counts > CLF Counts Limit, then stop conversion.
V_{IN} Low Start threshold	VIN_ON	0x35	R/W word	0x5A0 (EVM software: 36)	V_{IN} must be greater than this value to start conversion.
V_{IN} Low Stop threshold	VIN_OFF	0x36	R/W word	0x528 (EVM software: 33)	If V_{IN} falls below this value, then power conversion stops because input voltage is too low.
V_{IN} High Start threshold	MFR_SPECIFIC_02	0xD2	R/W word	0xC30 (EVM software: 78)	When starting up (CONTROL asserted), if V_{IN} > this value, then the input has too much bias and the converter should not start.
V_{IN} High Stop threshold	VIN_OV_FAULT_LIMIT	0x55	R/W word	0xD20 (EVM software: 84)	V_{IN} O V Fault voltage at which power conversion stops.
Temperature Start	MFR_SPECIFIC_03	0xD3	R/W word	0xC5C	During initialization, if the temperature is greater than this value, then do not start power conversion until it falls below this value.

(1) When EVM software needs do some conversion on firmware data for display, the displayed EVM software data is in ( ).

Table 2. PMBus Commands (continued)

Function	PMBus Command Name	Command Code	SMBus Transaction Name	Mfr. Default[1]	Notes
Temperature Stop	OT_FAULT_LIMIT	0x4F	R/W word	0xE5B	When enabled, if the temperature exceeds this value, then stop power conversion until it falls below this value.
Soft-Start Delay	TON_RISE	0x61	R/W word	0x7 (EVM software: 0.7)	After CONTROL is asserted and Startup Delay has passed, this is the amount of time that the output voltage should take to enter the regulation band.
Startup Delay	TON_DELAY	0x60	R/W word	0x00	After CONTROL is asserted, this is the delay before beginning soft-start.
Fault Delay	MFR_SPECIFIC_14	0xDE	R/W word	0x3E8 (0x64)	After current counts reach the limit, this is the amount of time before outputting PWM.
Mid-cap low limit	MFR_SPECIFIC_04	0xD4	R/W word	0x32 (EVM software: 40)	Multiplier of V_{IN} that gives Vmid_cap.
Mid-cap high limit	MFR_SPECIFIC_05	0xD5	R/W word	0x4B (EVM software: 60)	This is a multiplier of V_{IN} that gives Vmid_cap.
Current Limit	MFR_SPECIFIC_00	0xD0	R/W word	0x00 (EVM software: 0.5)	Setting for the I_{LIM} DAC.
V*S margin	MFR_SPECIFIC_06	0xD6	R/W word	0x509B (EVM software: 1.15)	Voltage second clamp. EVM software displays the anticipated duty-cycle margin and calculates a firmware format data followed by V*S-margin equation.
Read-Only (status) Commands					
Temperature	READ_TEMPERATURE	0x8D	Read Word	N/A[2]	Current temperature
V_{IN}	READ_VIN	0x88	Read Word	N/A[2]	Current input voltage
V_{OUT}	READ_VOUT	0x8B	Read Word	N/A[2]	Current output voltage
Mid-cap voltage	MFR_SPECIFIC_07	0xD7	Read Word	N/A[2]	Current voltage from VMID_CAP_SENSE
Current	READ_IOUT	0x8C	Read Word	N/A[2]	Output current
CLF Counts	MFR_SPECIFIC_13	0xDD	Read Word	N/A[2]	current-limit flag counts.
V_{CTRL}	MFR_SPECIFIC_08	0xD8	Read Word	N/A[2]	Current voltage from VCTRL_SENSE
Status Fault Status 1	MFR_SPECIFIC_09	0xD9	Read Word	N/A[2]	Current status and fault status n from system. Lower byte indicates status, higher byte indicates Fault Status 1.
Fault Status 2 Fault Status 3	MFR_SPECIFIC_12	0xDC	Read Word	N/A[2]	Lower byte indicates Fault status 2 and higher byte indicates Fault Status 3 from system.
DMAX	MFR_SPECIFIC_10	0xDA	Read Byte	N/A[2]	Current value for DMAX (scaled). EVM software shows the value it gets from firmware then it is divided by PWM_PERIOD(22).
ID name	MESSAGE_CODE_MFR_ID	0x99	Read Block	TI	Manufacturer ID name.
Model name	MESSAGE_CODE_MFR_MODEL	0x9A	Read Block	UCD8220 EVM	Manufacturer model name.
Revision number	MESSAGE_CODE_MFR_REVISION	0x9B	Read Block	2.0	Firmware revision.

(2) Real-time reading

3 Hardware Design

3.1 Introduction

The UCD8220 is an analog pulse-width modulator device used in digitally managed power supplies using a microcontroller or a device from the TMS320™ DSP family. The UCD8220 is a double-ended PWM controller configured with push-pull drive logic (two outputs, 180° out of phase). This EVM closes the PWM feedback loop with traditional analog methods, but the UCD8220 controllers include circuitry to interpret a time-domain digital pulse train. The pulse train or clock signal (CLK) contains the operating frequency and maximum duty cycle limit which are used to control the power supply operation. This interface eases the implementation of a converter with high-level control features without the added complexity or possible PWM resolution limitations of closing the control loop in the discrete time domain. Figure 2 shows a timing diagram describing the operation of the UCD8220 and its interaction with the microcontroller.

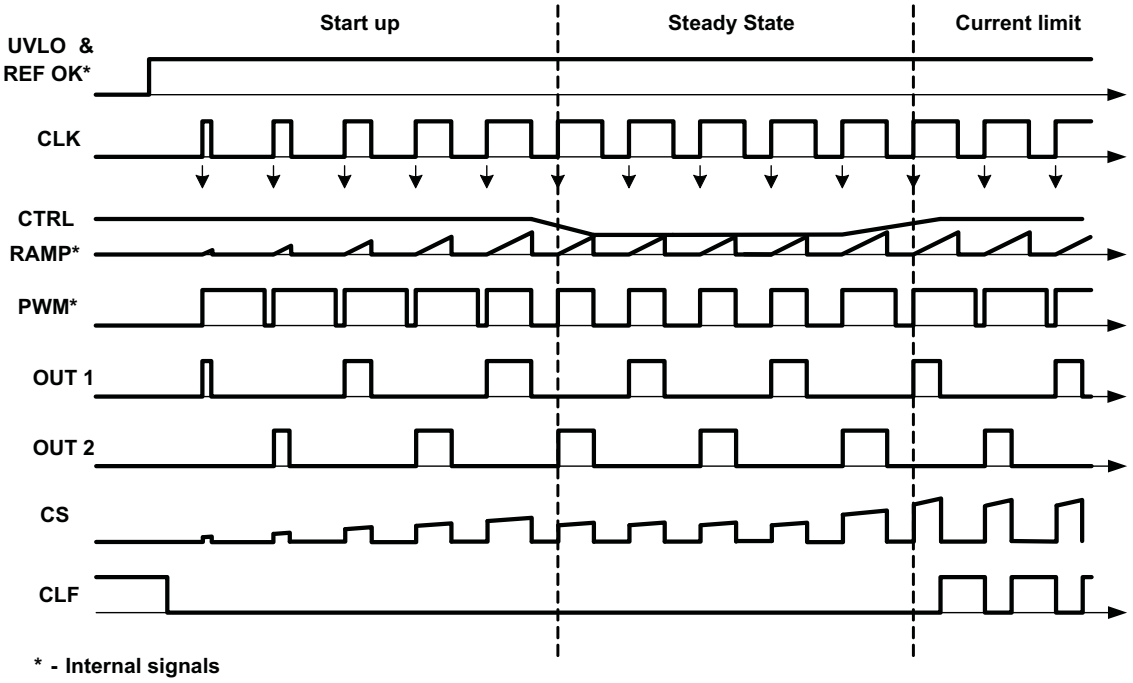


Figure 2. UCD8220 and MSP430F155 Timing Diagram

3.1.1 Hardware Design

The CLK signal from the microcontroller sets the operating frequency and maximum duty cycles of the outputs, OUT1 and OUT2.

3.2 Schematic

3.2.1 Jumper Settings

- **J6:** (VIN) Connect J6 jumper between J6-3 and J6-2.
- **J11:** (PVDD) Install J11
- **J9 and J13:** (VCC) Jumpers J9 and J13 enable different power sources to be connected to the MSP430F155 microcontroller. When programming the device via the JTAG connector, connect J9-3 to J9-2. For normal operation connect J9-1 to J9-2. The MSP430 may be powered from the USB-to-PMBus adapter (SAA) by connecting J13-3 to J13-2. Use only one jumper at a time.

Note: Do not run the Power stage when the MSP430 is powered by the JTAG source. The voltage coming from the JTAG source is lower than the voltage from the UCD8220, which results in a lower switching frequency.

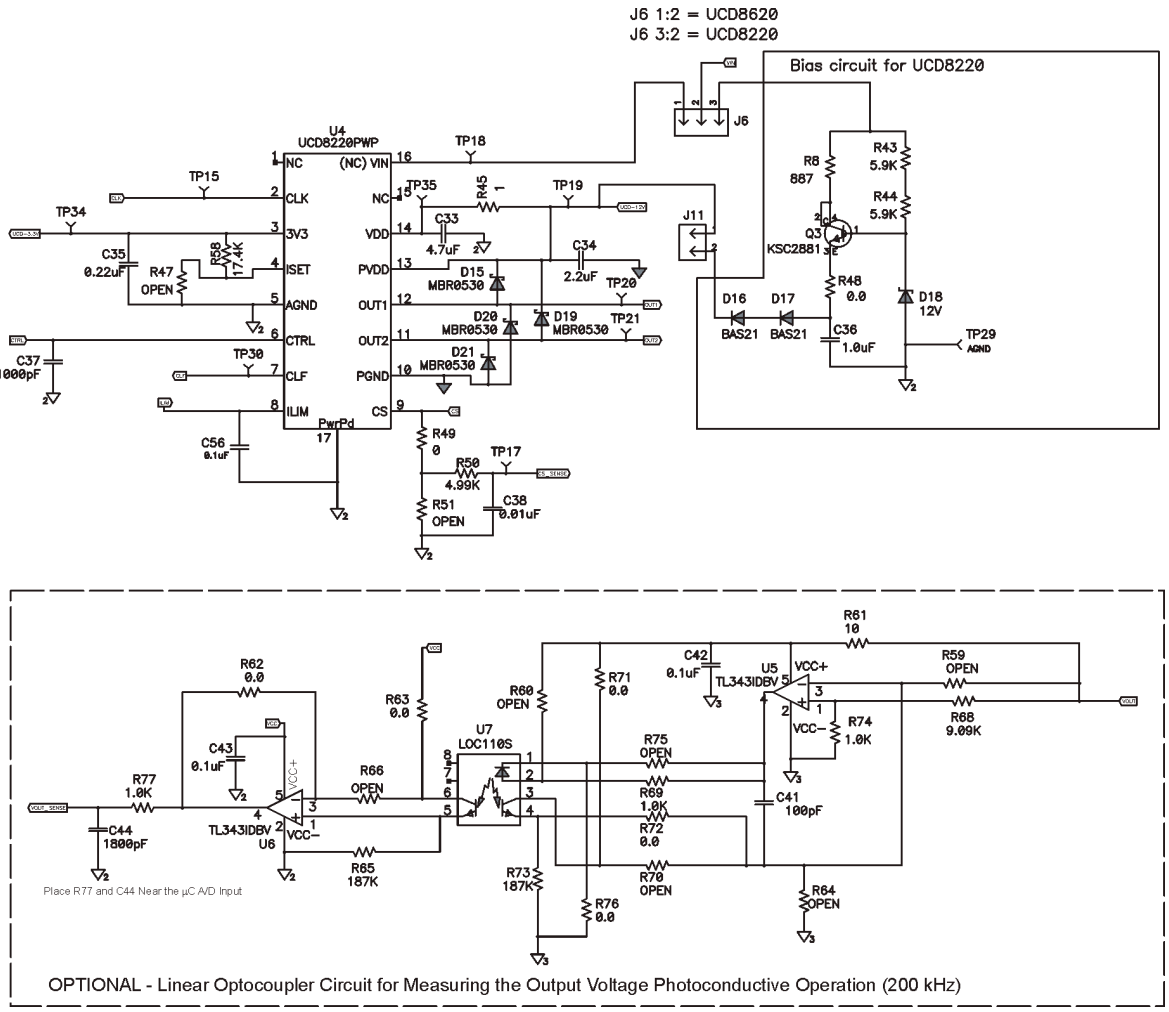


Figure 3. UCD8220EVM Schematic — UCD8220 Circuitry and Output Voltage Sensing

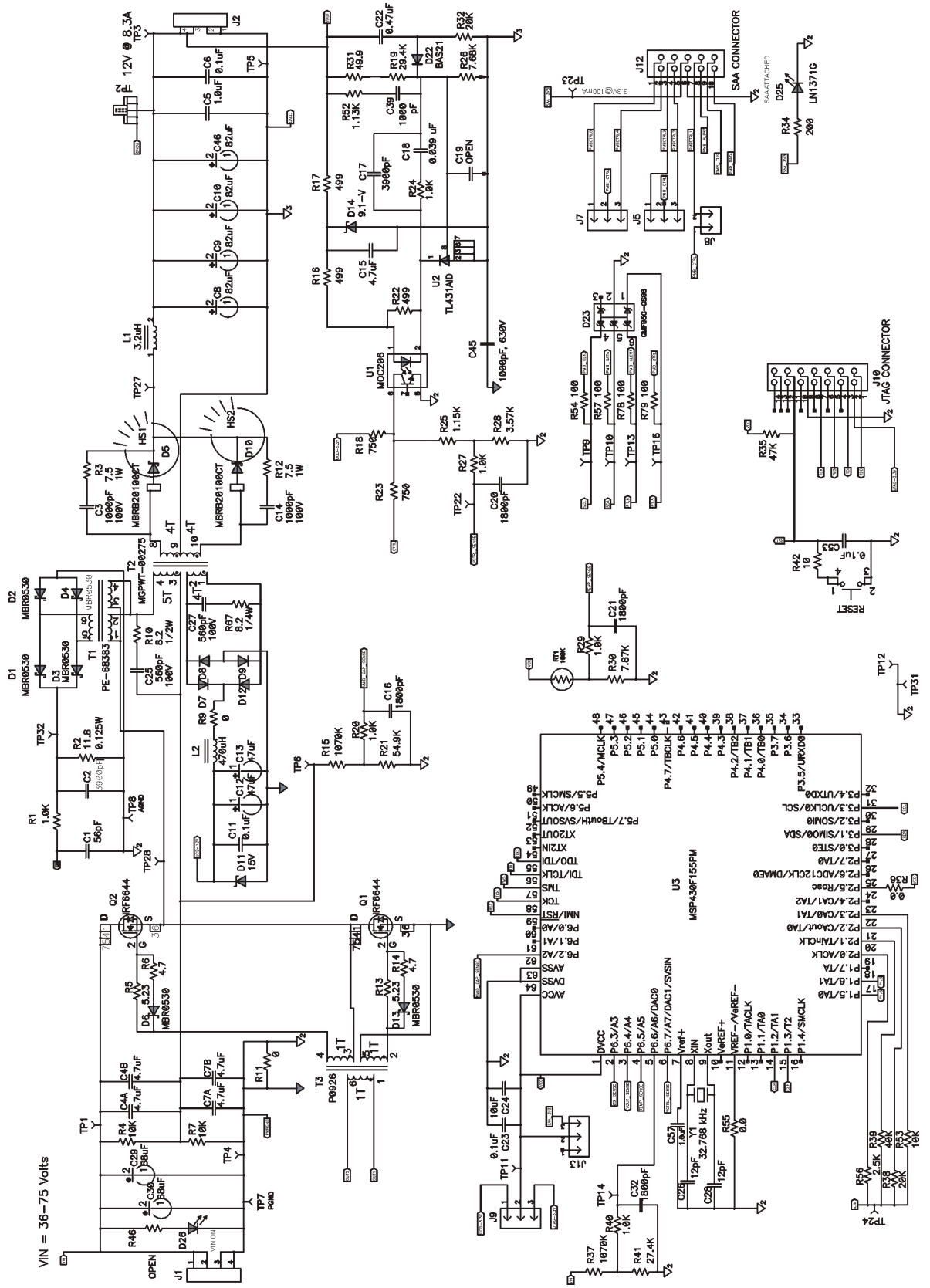


Figure 4. UCD8220EVM Schematic — Power Stage and Microcontroller Circuitry

3.3 Layout

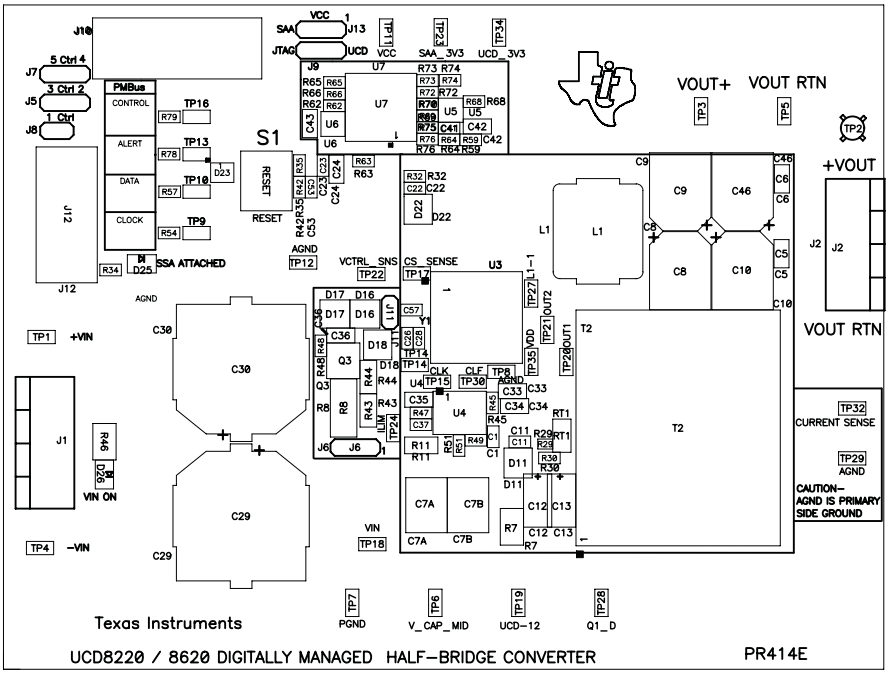


Figure 5. Top View of Board Assembly

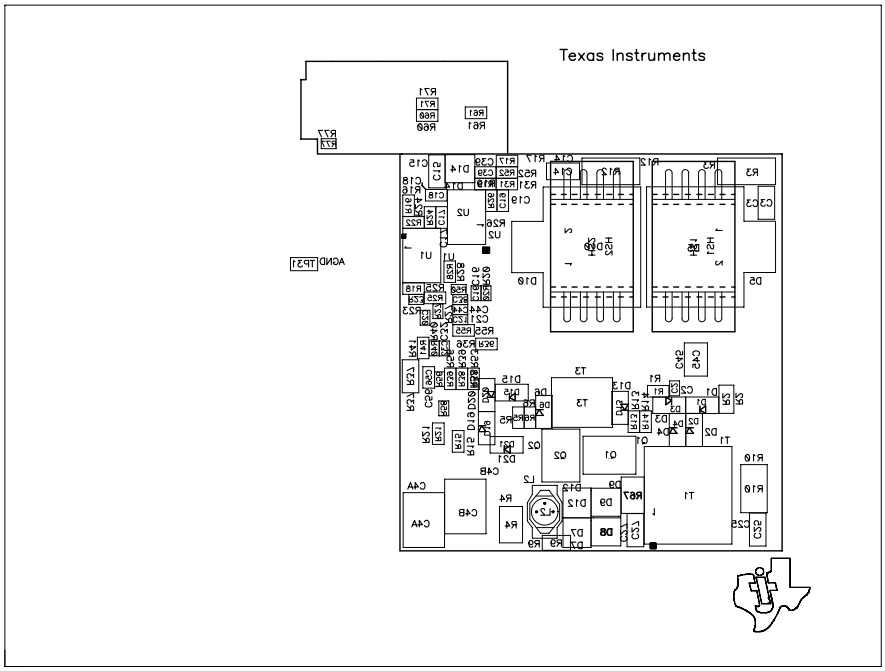


Figure 6. Bottom View of Board Assembly

3.4 Bill of Materials

Count	RefDes	Description	Size	Mfr
1	C1	Capacitor, Ceramic, 56-pF, 50V,NP0, 5%	603	Kemet
1	C11	Capacitor, Ceramic, 0.1-µF, 25V, X7R, 10%	603	Kemet
2	C12, C13	Capacitor, Tantalum Chip, 47-µF, 16V, 10%	0.281 × 0.126	AVX
1	C15	Capacitor, Ceramic, 4.7-µF, 25V, X5R, 10%	1206	Murata
4	C16, C21, C32, C44	Capacitor, Ceramic, 1800-pF, 50V, X7R, 10%	402	Kemet
1	C17	Capacitor, Ceramic, 3900-pF, 50V, COG, 5%	603	TDK
1	C18	Capacitor, Ceramic, 0.039-µF, 25V, X7R, 10%	603	Murata
1	C19	Capacitor, Ceramic, OPEN-µF, 25V, X7R, 10%	603	Std
1	C2	Capacitor, Ceramic, 3900-pF, 50V, X7R, 10%	402	Kemet
1	C20	Capacitor, Ceramic, 1800-pF, 25V, X7R, 10%	402	Kemet
1	C22	Capacitor, Ceramic, 0.47-µF, 16V, X7R, 10%	603	AVX
1	C23	Capacitor, Ceramic, 0.1-µF, 50V, X7R, 10%	603	Murata
1	C24	Capacitor, Ceramic, 10-µF, 6.3V, X7R, 10%	805	Kemet
2	C25, C27	Capacitor, Ceramic, 560-pF, 100V, COG, 5%	1206	AVX
2	C26, C28	Capacitor, Ceramic, 12-pF, 50V, X7R, 10%	603	Murata
2	C29, C30	Capacitor, Aluminum, 68-µF, 160V, EB-type, 750mArms	18mm × 16,5mm	Panasonic
2	C3, C14	Capacitor, Ceramic, 1000-pF, 100V, COG, 5%	1206	Murata
1	C33	Capacitor, Ceramic, 4.7-µF, 10V, X5R, 10%	805	Murata
1	C34	Capacitor, Ceramic, 2.2-µF, 10V, X5R, 10%	805	Murata
1	C35	Capacitor, Ceramic, 0.22-µF, 10V, X7R, 10%	805	Murata
1	C37	Capacitor, Ceramic,1000-pF, 50V, X7R, 10%	603	Kemet
1	C38	Capacitor, Ceramic, 0.01-µF, 16V, X7R, 10%	402	Murata
1	C39	Capacitor, Ceramic, 1000-pF, 25V, X7R, 10%	603	Std
1	C41	Capacitor, Ceramic,100-pF, 50V, NP0, 5%	603	Kemet
2	C42, C43	Capacitor, Ceramic, 0.1-µF, 16V, X7R, 10%	805	Murata
1	C45	Capacitor, Ceramic, 1000-pF, 630V, X7R, 20%	1210	TDK
4	C4A, C4B, C7A, C7B	Capacitor, Ceramic, 4.7-µF, 50V, X7R, 10%	2220	Murata
2	C5, C36	Capacitor, Ceramic, 1.0-µF, 50V, X7R, 10%	805	Murata
1	C53, C56	Capacitor, Ceramic, 0.1-µF, 16V, X7R, 10%	603	Murata
1	C57	Capacitor, Ceramic, 1.0-µF, 6.3V, X5R, 10%	603	Murata
1	C6	Capacitor, Ceramic, 0.1-µF, 50V, X7R, 10%	805	Murata
4	C8–C10, C46	Capacitor, OS-CON, 82-µF, 16V, 25-mΩ, 20%	8,3mm (E7)	Sanyo
10	D1–D4, D6, D13, D15, D19, D20, D21	Diode, Schottky, 500-mA, 30-V	SOD123	ON Semiconductor
1	D11	Diode, Zener, 15-V, 350-mW	SOT-23	Diodes, Inc.
1	D14	Diode, Zener, 9.1-V, 350-mW	SOT23	Diodes, Inc.
1	D18	Diode, Zener, 12V, 200-mW	SOT363	Diodes, Inc.
1	D23	Diode, Protection Unbidirectional/Bidirectional , 12.5V, 12A, 200W	SOT-363	Vishay
2	D25, D26	Diode, LED, Green, x-mA, yy-mcd	0.114 × 0.049	Panasonic
2	D5, D10	Diode, Dual Schottky, 20A, 100-V	D2PAK/HS	International Rectifier
7	D7–D9, D12, D16, D17, D22	Diode, Switching, 200-mA, 200-V, 330-mW	SOT23	Zetex
1	J10	Header, 2×7 pin, 100mil spacing, Straight, 4 Wall	0.338 × 0.988	3M
1	J12	Connector, Male Straight 2×10 pin, 100mil spacing, 4 Wall	0.338 × 0.788	3M

Count	RefDes	Description	Size	Mfr
5	J5–J7, J9, J13	Header, 3-pin, 100mil spacing, (36-pin strip)	0.100 × 3	Sullins
2	J8, J11	Header, 2-pin, 100mil spacing, (36-pin strip)	0.100 inch × 2	Sullins
1	L1	Inductor, SMT, 3.2-µH, 18-Arms, 6.8-mΩ	0.51 × 0.51	
1	L2	Inductor, SMT, 470-µH, 0.15Arms, 5.06-Ω	0.26 × 0.09	Coilcraft
1	U3	IC, Mixed Signal Microcontroller	QFP-64	TI
1	U7	IC, Linear Optocoupler		Clare
1	Y1	Crystal, 32.768-kHz, 8-pF	0.059 dia	ECS, Inc.
2	R77, R20	Resistor, Chip, 1.0 kΩ, 1/16-W, 1%	1.0K	Std
1	R56	Resistor, Chip, 2.5 kΩ, 1/16-W, 5%	2.5K	Std
1	R50	Resistor, Chip, 4990-Ω, 1/16-W, x%	4.99K	Std
1	R58	Resistor, Chip, 17.4 kΩ, 1/16-W, 1%	17.4K	Std
1	R23	Resistor, Chip, 750 Ω, 1/16-W, 1%	750	Std
6	R48, R62, R63, R71, R72, R76	Resistor, Chip, 0.0 Ω, 1/16-W, 1%	0	Std
1	R45	Resistor, Chip, 1 Ω, 1/16-W, 1%	1	Std
1	R24	Resistor, Chip, 1.0 kΩ, 1/16W, 1%	1.0K	Std
2	R1, R69	Resistor, Chip, 1.0 kΩ, 1/16-W, 1%	1.0K	Std
1	R74	Resistor, Chip, 1.0 kΩ, 1/16-W, 1%	1.0K	Std
1	R52	Resistor, Chip, 1.13 kΩ, 1/16W, 1%	1.13K	Std
1	R26	Resistor, Chip, 7.68 kΩ, 1/16-W, 1%	7.68K	Std
1	R30	Resistor, Chip, 7.87 kΩ, 1/16-W, yy%	7.87K	Std
1	R68	Resistor, Chip, 9.09 kΩ, 1/16-W, 1%	9.09K	Std
1	R61	Resistor, Chip, 10 Ω, 1/16-W, 1%	10	Std
1	R53	Resistor, Chip, 10 kΩ, 1/16-W, 1%	10K	Std
1	R32	Resistor, Chip, 20 kΩ, 1/16-W, 1%	20K	Std
1	R38	Resistor, Chip, 20 kΩ, 1/16-W, 1%	20K	Std
1	R19	Resistor, Chip,29.4 kΩ, 1/16-W, 1%	29.4K	Std
1	R39	Resistor, Chip, 40 kΩ, 1/16-W, 1%	40K	Std
1	R31	Resistor, Chip, 49.9 Ω, 1/16-W, 1%	49.9	Std
4	R54, R57, R78, R79	Resistor, Chip, 100 Ω, 1/16W, yy%	100	Std
2	R65, R73	Resistor, Chip, 187 kΩ, 1/16-W, 1%	187K	Std
1	R34	Resistor, Chip, 200, 1/16-W, 5%	200	Std
3	R16, R17, R22	Resistor, Chip, 499-Ω, 1/16-W, 1%	499	Std
1	R18	Resistor, Chip, 750-Ω, 1/16W, 1%	750	Std
6	R47, R59, R60, R66, R70, R75	Resistor, Chip, OPEN Ω, 1/16-W, 1%	OPEN	Std
1	R64	Resistor, Chip, OPEN Ω, 1/16-W, 1%	OPEN	Std
1	R49	Resistor, Chip, 0-Ω, 1/16-W, 1%	0	Std
1	R25	Resistor, Chip, 1.15 kΩ, 1/16-W, 1%	1.15K	Std
1	R28	Resistor, Chip, 3.57 kΩ, 1/16-W, 1%	3.57K	Std
1	R42	Resistor, Chip, 10-Ω, 1/16-W, 1%	10	Std
1	R41	Resistor, Chip, 27.4K-Ohms, 1/16-W, 1%	27.4K	Std
1	R35	Resistor, Chip, 47 kΩ, 1/16-W, 1%	47K	Std
1	R21	Resistor, Chip, 54.9 kΩ, 1/16-W, 1%	54.9K	Std
1	R15	Resistor, Chip, 1.07 mΩ, 1/16-W, 1%	1070K	Std
1	R51	Resistor, Chip, OPEN-Ω, 1/16-W, 1%	OPEN	Std
2	R36, R55	Resistor, Chip, 0.0-Ω, 1/16W, 5%	0	Std
2	R6, R14	Resistor, Chip, 4.7-Ω, 1/16W, 5%	4.7	Std

Count	RefDes	Description	Size	Mfr
2	R5, R13	Resistor, Chip, 5.23-Ω, 1/16W, 5%	5.23	Std
1	R2	Resistor, Chip, 11.8-Ω, 1/10-W, 1%	11.8	Std
1	R9	Resistor, Chip, 0.0-Ω, 1/10W, 5%	0	Std
1	R11	Resistor, Chip, 0.0 Ω, 1/8-W, 5%	0	Std
1	R43	Resistor, Chip, 5.9 kΩ, 1/8-W, 1%	5.9K	Std
1	R44	Resistor, Chip, 5.91 kΩ, 1/8-W, 1%	5.9K	Std
1	R37	Resistor, Chip, 1070 kΩ, 1/8-W, yy%	1070K	Std
1	R7	Resistor, Chip, 10 kΩ, 1/4-W, yy%	10K	Std
1	R67	Resistor, Chip, 8.2-Ω, 1/4 W, 5%	8.2	Std
1	R4	Resistor, Chip, 10 kΩ, 1/4 W, 5%	10K	Std
1	R46	Resistor, Chip, OPEN, 1/4 W, 5%	OPEN	Std
1	R10	Resistor, Chip, 8.2-Ω, 1/2-W, yy%	8.2	Std
2	R3, R12	Resistor, Chip, 7.5 Ω, 1W, 5%	7.5	Std
1	R8	WSL-2512-xx 1% R86 Resistor, Chip, 1W, 887Ω, 1%	887	
1	SH1	Short jumper		
1	RESET	Switch, SPST, PB Momentary, Sealed Washable	KT11P2JM	KT11P2JM
2	J1, J2	Terminal Block, 4-pin, 15-A, 5,1mm	ED2227	
1	RT1	Thermistor, 100 kΩ, 5%, Beta=4250	NHQ104B425T10	
2	U5, U6	IC, Op Amp	TL343IDBV	TL343IDBV
1	U2	IC, Adj Shunt Regulator, 100-mA, 36-V	TL431AID	TL431AID
1	TP1	Test Point, SMT	5015	#NAME?
1	TP4	Test Point, SMT	5015	#NAME?
2	TP3, TP5	Test Point, SMT	5015	
4	TP9, TP10, TP13, TP16	Test Point, SMT	5015	
4	TP8, TP12, TP29, TP31	AGND, Test Point, SMT	5015	
1	TP30	CLF, Test Point, SMT	5015	
1	TP15	CLK, Test Point, SMT	5015	
1	TP17	CS_SENSE, Test Point, SMT	5015	
1	TP32	CURRENT SENSE, Test Point, SMT	5015	
1	TP24	ILIM, Test Point, SMT	5015	
1	TP27	L1-1, Test Point, SMT	5015	
1	TP20	OUT1, Test Point, SMT	5015	
1	TP21	OUT2, Test Point, SMT	5015	
1	TP7	PGND Test Point, SMT	5015	
1	TP28	Q1_D Test Point, SMT	5015	
1	TP23	SAA_3V3 Test Point, SMT	5015	
1	TP14	TP14 Test Point, SMT	5015	
1	TP19	mcd-12 Test Point, SMT	5015	
1	TP34	UCD_3V3 Test Point, SMT	5015	
1	TP11	VCC Test Point, SMT	5015	
1	TP22	VCTRL_SENSE Test Point, SMT	5015	
1	TP35	PVDD Test Point, SMT	5015	
1	TP18	VIN Test Point, SMT	5015	
1	TP6	V_CAP_MID Test Point, SMT	5015	
1	TP2	Adaptor, 3.5-mm probe clip (or 131-5031-00)	131-4244-00	
2	Q1 Q2	Transistor, MOSFET, 100V, 8.3A, 10.7 mΩ	IRF6644	IRF6644

Count	RefDes	Description	Size	Mfr
1	Q3	Transistor, NPN SOT-89, 120V	KSC2881	KSC2881
1	U4	IC, Digitally Managed Double-Ended Analog PWM Controller	UCD8220PWP	UCD8220PWP
1	T2	XFMR, Half-Bridge Transformer 48V to 12V at 200 Watts	MGPWT-00275	MGPWT-00275
1	T3	Transformer, Driver, 1:1:1, 330uH Ip, 1500V isolation, 2.0 Ω	P0926	P0926
1	T1	1:1:200 Transformer, Current Sense, 15A, 50-500kHz	PE-68383	PE-68383
1	U1	IC, Optocoupler	MOC206	MOC206

3.5 Test Results

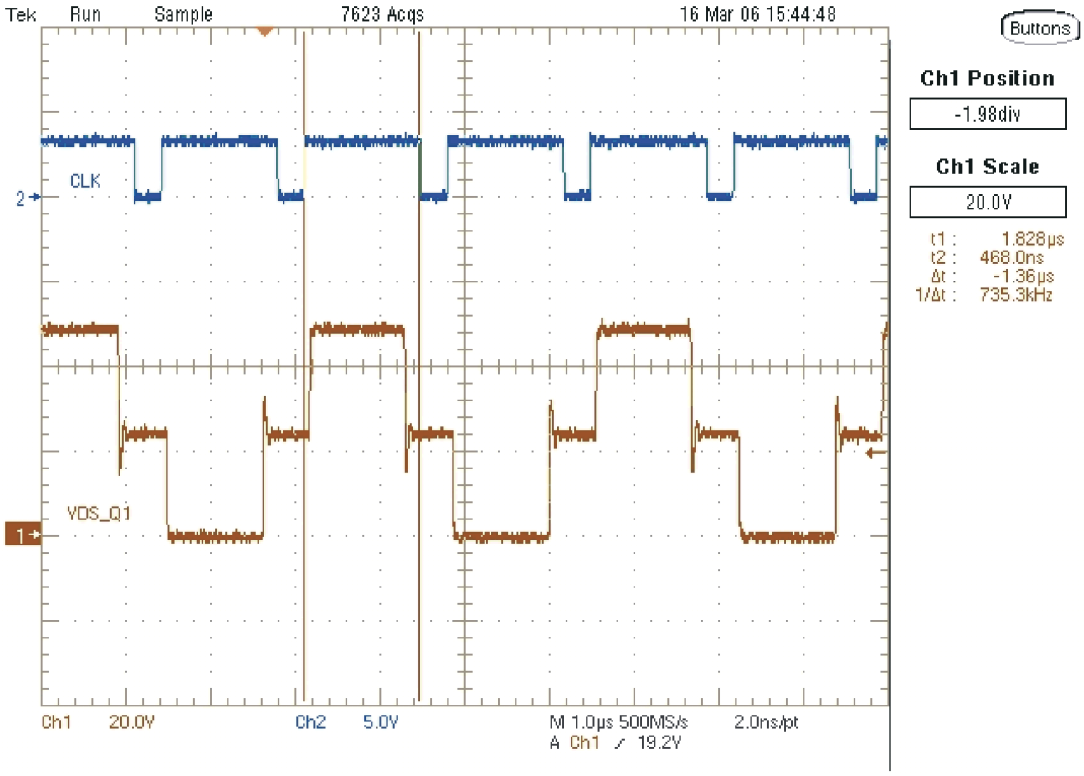


Figure 7. CLK and VDSQ1 in Steady-State With Vin = 48 V , Iout = 5 A, Demonstrating Volt*Second Clamp

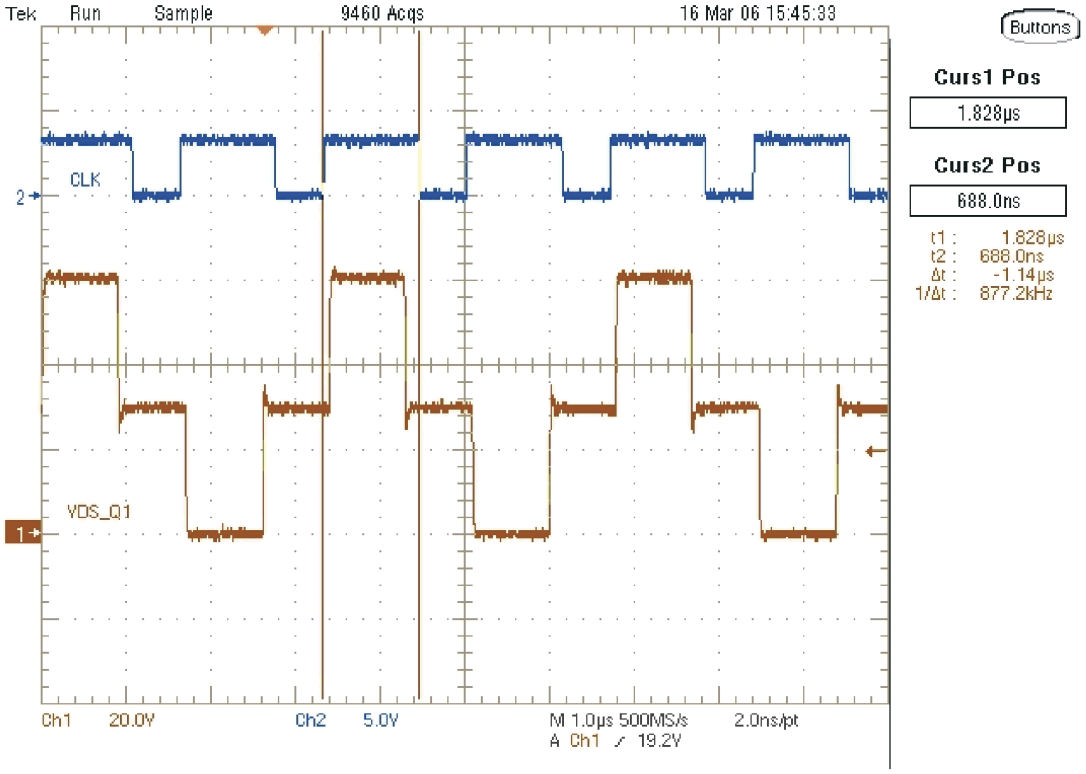


Figure 8. CLK and VDSQ1 in Steady-State With Vin = 60 V , Iout = 5 A, Demonstrating Volt*Second Clamp

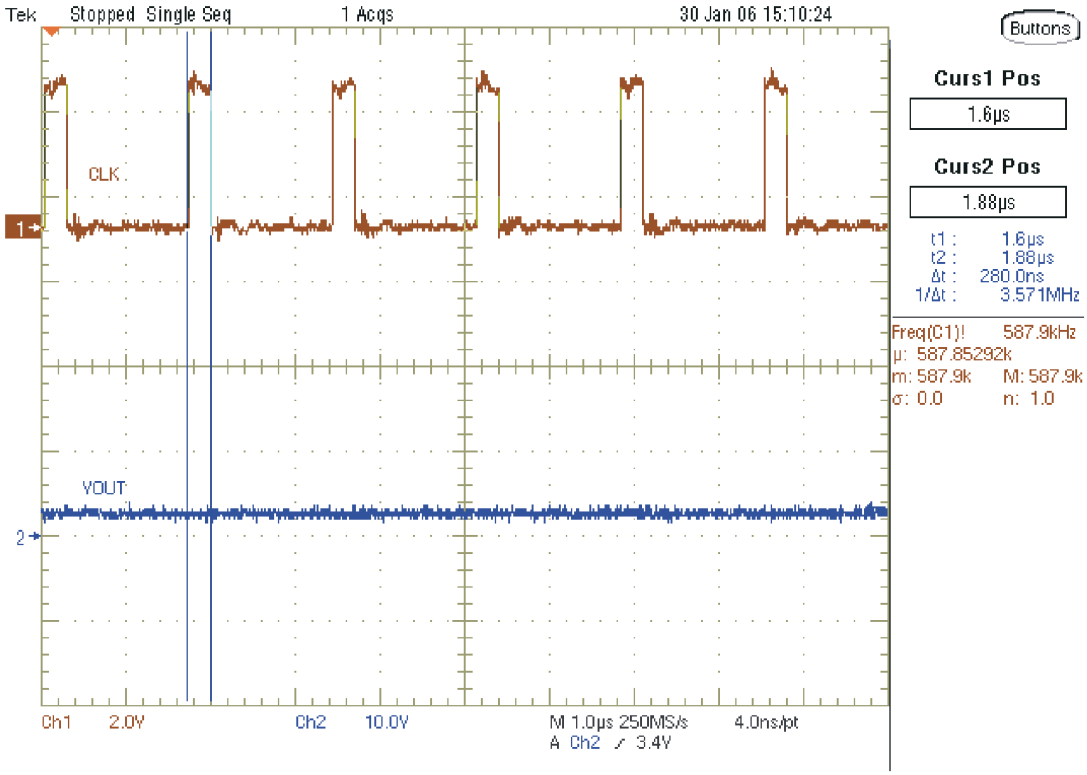


Figure 9. CLK and Vout During Soft-Start

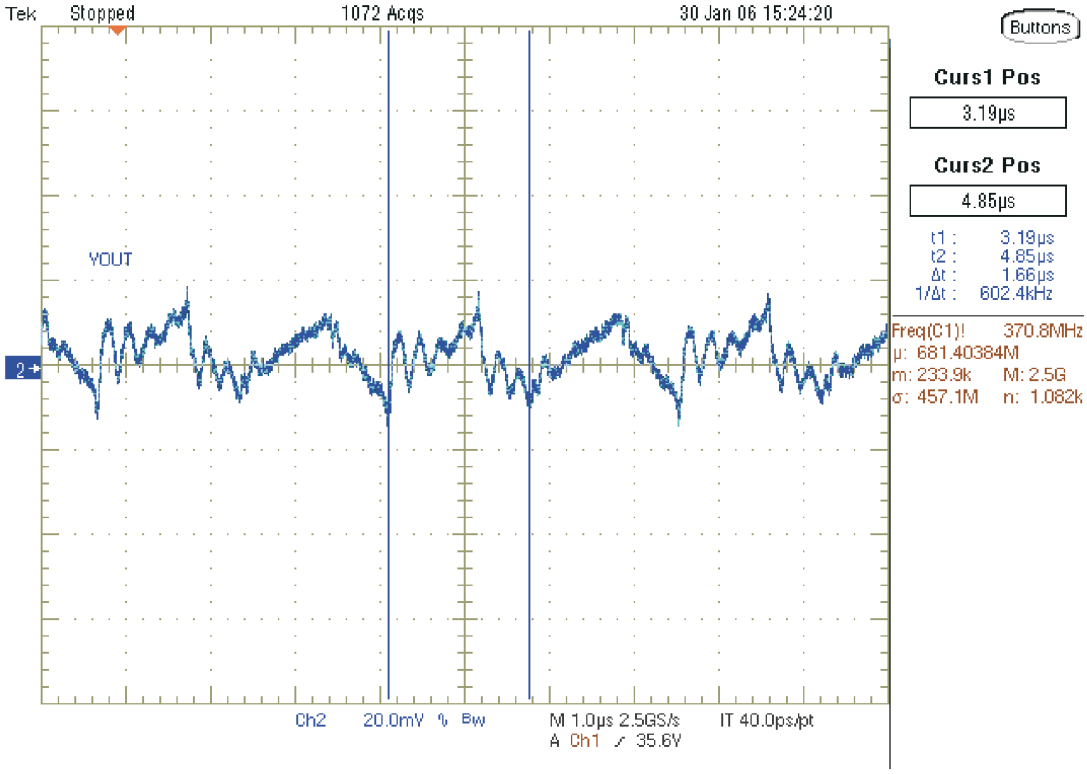


Figure 10. Vout Ripple, Vin = 48 V, Iout = 5 A

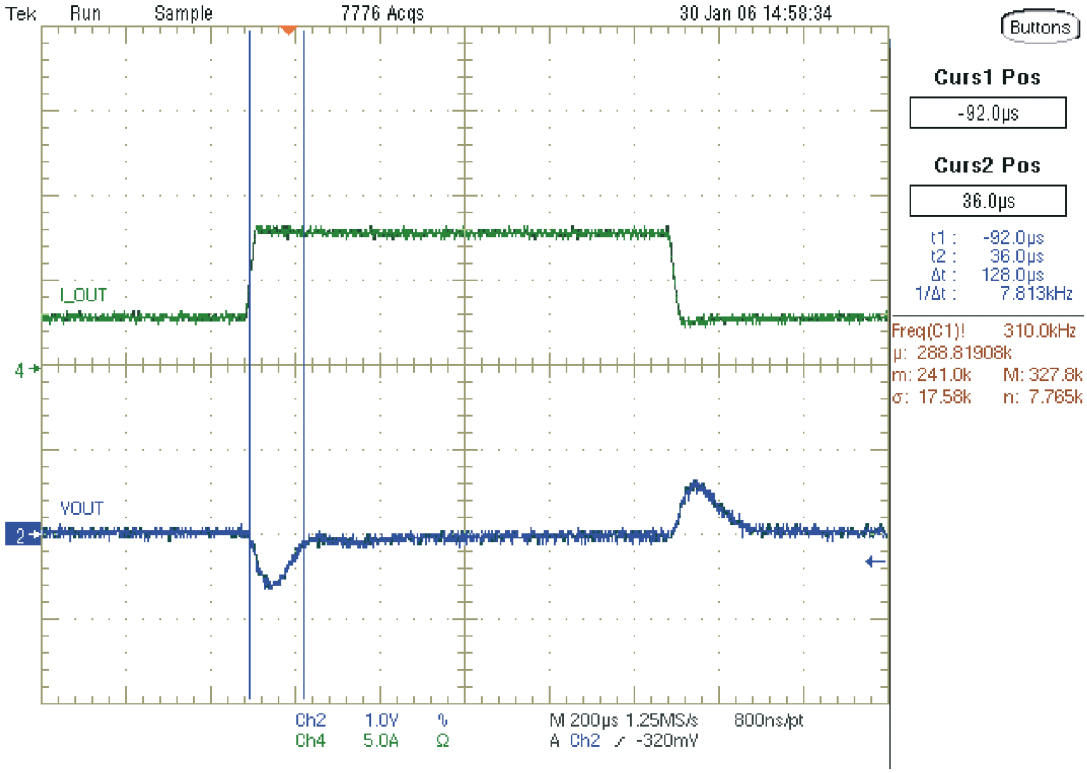


Figure 11. Load Transient Response Vin = 48 V

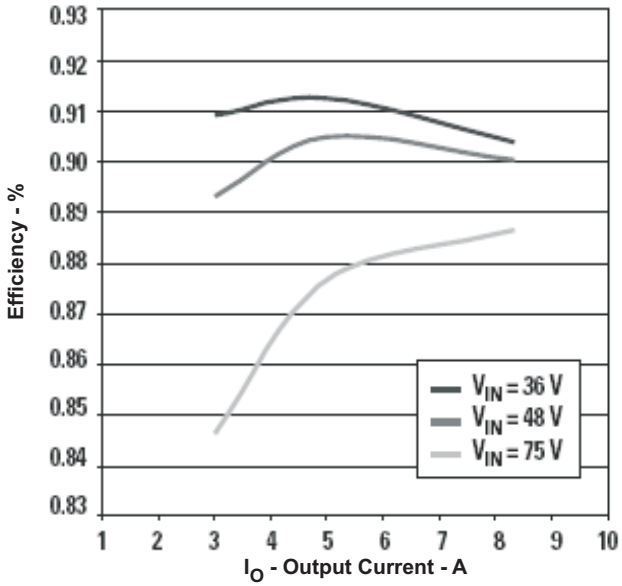


Figure 12. Efficiency vs Output Current

3.6 Examples of How the EVM Software Translates the MSP430 A/D Converter Counts to Real-World Power Supply Values (See Also Q-Math Section)

The EVM software reads the firmware values at startup, and displays the values in a convenient format for power-supply engineers. For example, the input voltage that is read from the hardware is an integer value between 0 and 4095. The EVM software translates this A/D value into the actual input voltage in standard units (V, A, etc.). Alternatively, the EVM software is used to set parameters in the program. Again, the EVM software converts user inputs in standard power-supply units to the value required by the hardware to redefine that parameter.

3.6.1 Initial Firmware Status Display Values

Constant Definitions:

Vref = 2.5
R30 = 7870
β = 4250
Max_count = 4095
RT1 = 100000

VIN:

$$Vin = Vad\_Vin \times \frac{Vref}{Max\_count} \times \frac{R37 + R41}{R41} \quad (1)$$

Where:

Vad_Vin = The decimal count that comes from the A/D for the A6 Input
R37 = 1.07×10^6 Ohms
R41 = 27.410^3 Ohms

VMID_CAP_ACTUAL:

$$Vmid\_cap\_actual = \frac{(Vad\_Vmid\_cap) \times \frac{R15+R21}{R21}}{Vad\_Vin \times \frac{R37+R41}{R41}} \quad (2)$$

Where:

Vad_Vmid_cap = The decimal count that comes from the A/D for the A2 Input

R15 = $1.070 \times 10^6 \Omega$

R21 = $549 \times 10^3 \Omega$

TEMP:

$$T\_celsius = \left(Vad\_temp \times \frac{Vref}{Max\_count} + 6.3064\right) \times \frac{100}{2.196} - 273.15 \quad (3)$$

Vad_temp = The decimal count that comes from the A/D for the A5 Input

I_in_avg:

$$I\_in\_avg = Vad\_I\_in \times \frac{Vref}{Max\_count} \times \frac{200}{RCS} \times \left(\frac{1.2}{2}\right) \quad (4)$$

Vad_I_in = The decimal count that comes from the A/D for the A3 Input (Pin 19)

RCS = 11.8 Ω

Vcntrl DISPLAY VALUE:

$$Vcntrl = Vad\_Vcntrl \times \frac{Vref}{Max\_count} \times \frac{R25 + R28}{R28} \quad (5)$$

Where

Vad_Vcntrl = The decimal count that comes from the A/D for the A7 Input

R25 = $1.15 \times 10^3 \Omega$

R28 = $3.57 \times 10^3 \Omega$

PWM DISPLAY VALUE:

$$PWM = \frac{PWM\_count}{PWM\_period} \times 100 \quad (6)$$

Where

PWM_count = The decimal count that is calculated to determine the pulse width of the clock signal.

PWM_period = The count that is used to determine the period of the Clock signal

PWM_period = 27

3.7 Setpoint Display Values

On power up, the EVM software displays the default values for these parameters, and allows the user to modify those values during operation of the unit. The new values are only used for the remainder of that operating session. When the unit it turned off and then back on, the starting value is the default value.

3.7.1 Current Limit Flag (CLF) Counts Limit

The Current Limit Flag Counts Limit defines the number of consecutive current-limit events that must occur before the firmware responds by initiating the current-limit response routine. This stops the switching action of the MOSFETS and waits a period of time defined as the fault delay (100ms), then invokes the soft-start routine to determine if the fault is cleared.

3.7.2 LOW Vin Stop Limit (VIN_UVLO_OFF)

Initial Setpoint Display Value:

$$UVLO\_stop = UVLO\_stop\_software \times \frac{Vref}{Max\_count} \times \frac{R37 + R41}{R41} \text{ Volts} \quad (7)$$

Where:

UVLO_stop_software = The decimal count that comes from the firmware code.

User Setpoint Procedure:
The user may modify this value for the remainder of the operating session by entering a new value, in volts. The EVM software then converts that voltage to the appropriate A/D value for use in the hardware.

$$\text{UVLO\_stop\_count} = \text{UVLO\_stop\_User} \times \frac{\text{Max\_count}}{\text{Vref}} \times \frac{\text{R41}}{\text{R37} + \text{R41}} \text{ Count} \quad (8)$$

UVLO_stop_count = The decimal count that goes to the firmware – [A/D Count]
UVLO_stop_User = The Voltage Setpoint entered by the User – [Volts]

The same procedure applies to **LOW Vin Start Limit (VIN_UVLO_ON), High Vin Start Limit (VIN_OVLO_ON), and High Vin Stop Limit (VIN_OVLO_OFF)**

3.7.3 VMID_CAP_LIMIT

This parameter demonstrates that with a digital controller, the designer can add protection features rather easily. The intention is to limit the imbalance on the input capacitors to some reasonable value. For ideal operating conditions, the voltage at VMID_CAP equals exactly half that of the input voltage. During start up and through transients, the voltage at VMID_CAP may vary from its ideal value. This protection feature limits the amount of mid-cap voltage variation. For example: For transients, a 20% variation from the ideal value is acceptable. During soft-start this limit is ignored.

VMID_CAP_LIMIT_LOW = 0.4
VMID_CAP_LIMIT_HIGH = 0.6

The EVM software converts the displayed VMID_CAP_ACTUAL value to a ratio of VMID_CAP / Vin. See the Q Math Appendix for the best way to implement this in real-time coding.

VMID_CAP_LIMIT_LOW = 0.4
VMID_CAP_ACTUAL = VMID_CAP / Vin
VMID_CAP_LIMIT_HIGH = 0.6

The user can then set the **VMID_CAP_LIMIT_LOW** or the **VMID_CAP_LIMIT_HIGH** value during operation to cause the supply to shut down and run through its fault delay and soft-start routine.

3.7.4 Temperature Stop Limit

Initial Setpoint Display Value:

$$\text{TEMP\_stop} = \left(\text{TEMP\_stop\_software} \times \frac{\text{Vref}}{\text{Max\_count}} + 6.3064\right) \times \frac{100}{2.196} - 273.15^{\circ}\text{C} \quad (9)$$

Where:

TEMP_stop_software = The decimal count that comes from the firmware.

User Setpoint Procedure:
The user may modify this value for the remainder of the operating session by entering a new value in degrees Celsius. The EVM software converts the temperature to the appropriate A/D value for use in the hardware.

$$\text{TEMP\_stop\_count} = \left[(\text{TEMP\_stop\_User} + 273.15) \times \frac{2.196}{100} - 6.3064\right] \times \frac{\text{Max\_count}}{\text{Vref}} \text{ Count} \quad (10)$$

TEMP_stop_count = The decimal count that goes to the firmware – [A/D Count]
TEMP_stop_User = The Temperature Setpoint that comes from the User – [°C]

The same procedure applies to the **Temperature Start Limit**.

3.7.5 Dmax Limit

Initial Setpoint Display Value:

$$\text{DMax\_Limit} = \frac{\text{Dmax\_limit\_software}}{\text{Periiod\_count}} \times 100\% \quad (11)$$

Where:

Dmax_limit_software = The constant that comes from the firmware code.

User Setpoint Procedure:
The user may reset this value for the remainder of the operating session by entering in new value. The EVM software then converts the new value of VS margin to the appropriate constant value for use in the hardware.

$$DMax\_Limit\_run = \frac{Dmax\_Limit\_user}{100} \times Period\_count \quad (12)$$

DMax_Limit_run = The new constant that is used in the hardware for the remainder of the operating session. – [Count]
Dmax_Limit_user = The Dmax Limit input that comes from the User. – [%]

3.7.6 V*S Margin (For Half-Bridge)

The Volt*Second clamp value is based on calculating the required regulating duty cycle, then applying a certain amount of margin to that required duty cycle so that the analog control loop has the ability to regulate the output voltage while protecting the power train from saturation.

The transfer function for the Half-Bridge Topology is:

$$Vout + Vd = \left(\frac{Vin - 1}{2}\right) \times \frac{Ns}{Np} D \quad (13)$$

Therefore:

$$D_{required} = 2\frac{(Vout + Vd)}{(Vin - 1)} \times \frac{Np}{Ns} \quad (14)$$

The UCD8220 clock frequency and duty cycle are the same as the frequency and duty cycle of the output waveform. The KD_margin constant represents the duty-cycle margin allowed for the design.

KD_margin = 1.15

$D_{voltsecond} = D_{required} \times$ KD_margin

The program calculates the actual duty cycle of the clock by taking the lower of the Dvoltsecond and the Dmax_limit.

The displayed value, VS_Margin is the anticipated duty-cycle margin, given by:

VS_Margin = (KD_margin) $\times$ 100

3.7.7 ILim DAC

The user can change this value at any time.

Initial Setpoint Display Value:
ILim_DAC = The voltage at the ILIM pin selected from the list of possible values in the following table.

The following table shows the port settings and the expected current-limit setpoint values. ILIM0 is the default startup state (all ports in high impedance). The EVM software has a pulldown menu with the possible ILIM SETPOINT values to choose from. The current-sense resistor was selected for a default value of 0.5 V, so at that value the supply provides full power.

	ILIM SETPOINT	P2.3	P2.2	P2.1	P2.0
ILIM1	0.143	0	0	1	0
ILIM2	0.287	0	1	0	0
ILIM3	0.43	0	1	1	0
ILIM0	0.5	OPEN	OPEN	OPEN	OPEN
ILIM4	0.574	1	0	0	0
ILIM5	0.717	1	0	1	0
ILIM6	0.861	1	1	0	0
ILIM7	1.004	1	1	1	0

User Setpoint Procedure:
The user may select the new ILIM voltage from a pulldown menu. This reverts to the default value when the unit is power cycled. For evaluation purposes, the user may select a value that would immediately trigger a current-limit condition, resulting in a shutdown and restart.

4 Firmware Design

This section is intended to help designers understand the current firmware design so that they can easily modify it to expand the features or accommodate hardware changes.

First, an overview of the firmware design requirements is given. Then the firmware-development process is described from the following perspectives: the peripheral configuration, interrupts, state machine, and source-code review.

4.1 Overview

The firmware for MSP430F155, the digital controller on UCD8220 evaluation board, includes the following functions:

- Monitor the power system status:
  - Monitor the following six A/D-channels:
    - Current
    - Temperature sensor
    - Input voltage
    - Vctrl voltage
    - Output voltage
    - Mid capacitor voltage
  - Monitor current, count the number of times the current-limit flag is set.
  - Check if PMBus CONTROL is enabled or disabled.
- Control the following output signals to the UCD8220 chip:
  - PWM envelope signal. Adjust its duty cycle according to system-status and system-timing considerations.
  - Current-limit threshold ILIM setting.
- Provide a PMBus communication interface between the power system and the host for real-time status reporting and dynamic power-supply control.

The hardware interface is shown in Table 3.

Table 3. Pin Descriptions of MSP430F155

Pin Label	Port	Description
Input		
CS_sense (A/D)	P6.3	Power system current
Temp_sense (A/D)	P6.5	Temperature sensor
Vin (TP 14) (A/D)	P6.6	Input voltage (one of the factors to calculate PWM duty cycle)
Vctrl_sense (A/D)	P6.7	Error voltage between the desired output voltage and the real output
Vout_sense (A/D)	P6.4	Output voltage of the power supply
Vmid_cap_sense (A/D)	P6.2	Middle capacitor between Vin and ground
CLF	P1.3	Timer A2 input, current-limit flag. If UCD8220 detects overcurrent, this pin is set high until the overcurrent condition is cleared.
SDA	P3.1	I^2C data bus for PMBus
SCL	P3.3	I^2C clock for PMBus
P1.5	P1.5	GPIO Input. PMBus control line

Table 3. Pin Descriptions of MSP430F155 (continued)

Pin Label	Port	Description
Output		
Clk	P1.2	PWM envelope control signal to UCD8220 chip
P1.6	P1.6	GPIO Output. PMBus alert. Set pin high to alert host
TD0	55	For JTAG debugging
TD1	56	For JTAG debugging
TMS	57	For JTAG debugging
TCK	58	For JTAG debugging
RST	59	For JTAG debugging
Input/Output		
ILIM	P2.0~P2.3	GPIO current-limit setting. 4-bit binary value that controls the current-limit value of the UCD8220 device.

4.2 Peripheral Settings

Some of the peripheral configurations were shown in Table 3. The six A/D channels are set in sequence-of-channel mode. When conversion of the last channel completes, it triggers a conversion-done interrupt.

The MSP430F155 requires that the A/D clock frequency must be between 0.45 MHz and 6.3 MHz. Because the A/D clock is sourced from SMCLK (approximately 13.5 MHz), a divide-by-3 factor is used. As a result, the A/D clock is calculated as 13.5 MHz/3, or approximately 4.5 MHz. Each A/D channel is configured to take 21 cycles to complete one accurate reading. It takes approximately 28 µs (21×6/4.5 MHz) to complete all six A/D readings.

I^2C is configured in slave mode. It serves as the physical layer of the PMBus. For more information about its configuration, see Section 2.2.

Besides the *external* hardware pin configurations, some *internal* peripherals are configured:

- **Timer A1:** set up in PWM output mode. The output is P1.2, which is the PWM envelope-control signal to the UCD8220 chip. The PWM period is approximately 600kHz. Its duty cycle is adjusted based on system status and system timing.
- **Timer A2:** set up in capture mode to detect current-limit flag (CLF). See Table 3 for P1.3 description.
- **Timer B0:** set up as an upcounter to establish a system tick with a 50-µs period. The falling edge, which happens every 100 µs, is set to trigger the A/D readings of all the six A/D channels.

4.3 Interrupts

The overall firmware structure is a small event-driven system. There are only two types of events: A/D conversion-done and I^2C interrupts.

- A/D conversion-done interrupt; happens periodically, can also be treated as a system timer. The responses include:
  - Update all A/D readings
  - Invoke the system-control state machine to monitor system status and calculate the desired PWM duty cycle. See section Section 4.4.1 for more information.
- I^2C interrupt:
  - Run a state machine to interpret the incoming bytes. Insert the valid ones into a receive buffer. Shift the data out from a transmit buffer and calculate packet-error checking (PEC) when necessary. See section 4.4.2 for more information.
  - Parse the received data according to the PMBus protocol. Update the system setting when a write command is issued and put the requested data into the transmit buffer when it is a PMBus read command.

4.4 State Machines

4.4.1 System Control State Machine

As previously mentioned, the system-control state machine is called when A/D conversion is completed, which happens periodically. The period is determined by Timer B0. In this case, it is 100 µsec.

There are four states:

- **System idle state:** In this state, the firmware clears PWM output by setting the duty cycle to zero. Invoke a delay in response to a system fault status to give the power supply time to recover. Otherwise check the routine status. If it passes routine check, then go to the next state. Otherwise, remain in this state. The UCD8220 power converter does not operate during this state.

Note: The routine check is to check whether the PMBus CONTROL is enabled, input voltage is within range, and the temperature is normal. As long as all the conditions are met, it passes the routine check. The routine check is done in every state.

- **Startup delay state:** Repeat the routine check until the startup delay period expires. PWM output is stopped to allow the system to stabilize before the output PWM is activated. During this state, the UCD8220 power converter does not operate. If the routine check fails, a fault status is generated and the system goes back to the system idle state.
- **Soft-start delay state:**. Check routine status and current-limit flag. Firmware increases PWM duty cycle step by step if the status check is OK. The ramp-up step is calculated by evenly dividing the maximum PWM duty cycle by the number of clock cycles of delay time. In this mode, the UCD8220 power converter starts its soft-start sequence.
- **Run state:**. Perform routine check. Besides that, check whether mid capacitor reading within range and the current-limit flag counter below the maximum value. If any one of those checkings fails, the fault status is generated and the system goes back to the system idle state. If all pass, then adjust PWM duty cycle according to the V*S margin formula given in section 3.

Figure 13 illustrates the preceding system states.

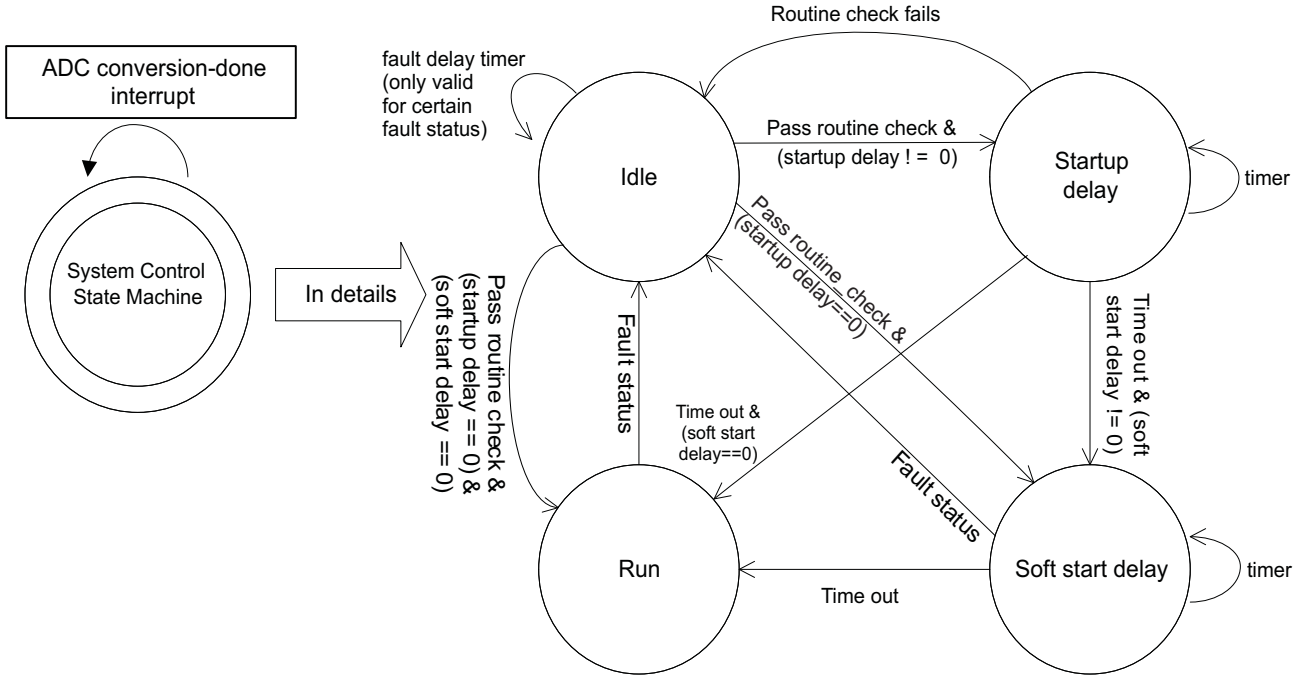


Figure 13. System Control Status Machine

4.4.2 I²C State Machine

This state machine is invoked by the I²C interrupt service routine. As described in Section 4.3, it is used to

- Prepare data for the PMBus command parser
- Output requested PMBus data
- Calculate the PEC for PMBus read commands

Figure 14, Figure 15, and Figure 16 show the format of the PMBus command packets.

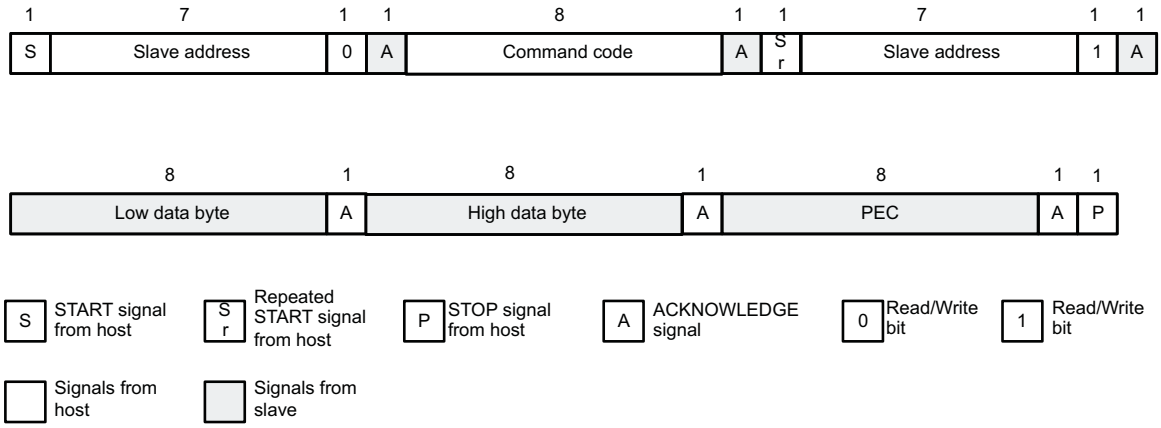


Figure 14. Read Word Packet Protocol With PEC

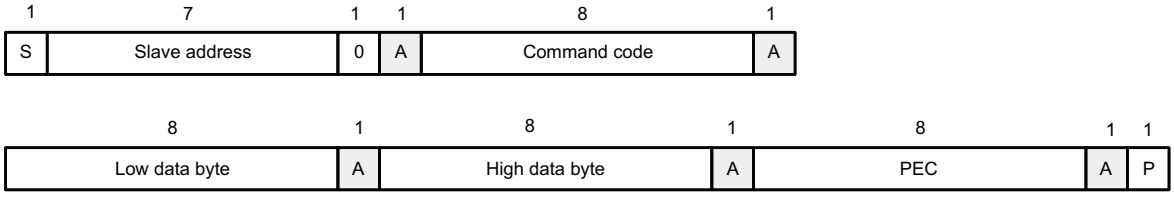


Figure 15. Write Word Packet Protocol With PEC

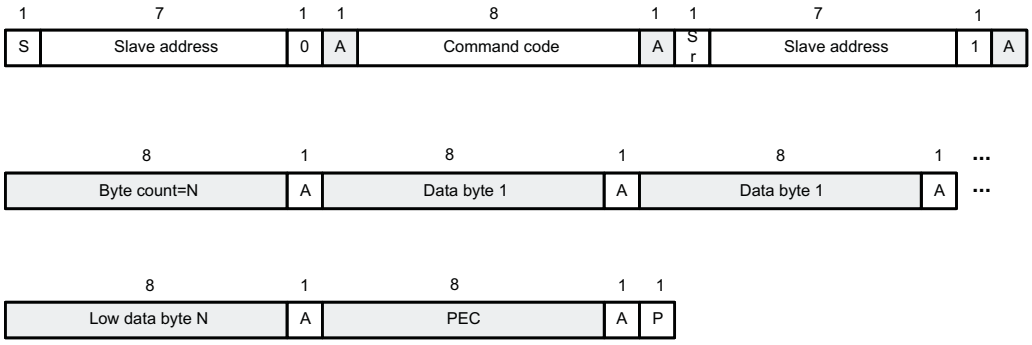


Figure 16. Read Block Packet Protocol With PEC

The MSP430F155 I²C peripheral hardware shifts data in and out using the clock from the I²C master, and automatically sends an ACK to the host.

When a START/STOP signal is detected, an interrupt is triggered.

Following the START signal, the hardware module interprets the next received byte as the combination of slave address and the read/write bit. It then automatically checks for a slave-address match. If it matches, depending on whether the read/write bit is 0 or 1, it generates an interrupt. The interrupt vector indicates master writing data and master requesting data events, respectively. Otherwise, it idles until it detects the next START signal.

Unique interrupt vectors are used for

- Start
- Stop
- Write data
- Read data

The I²C state machine is shown in Figure 17.

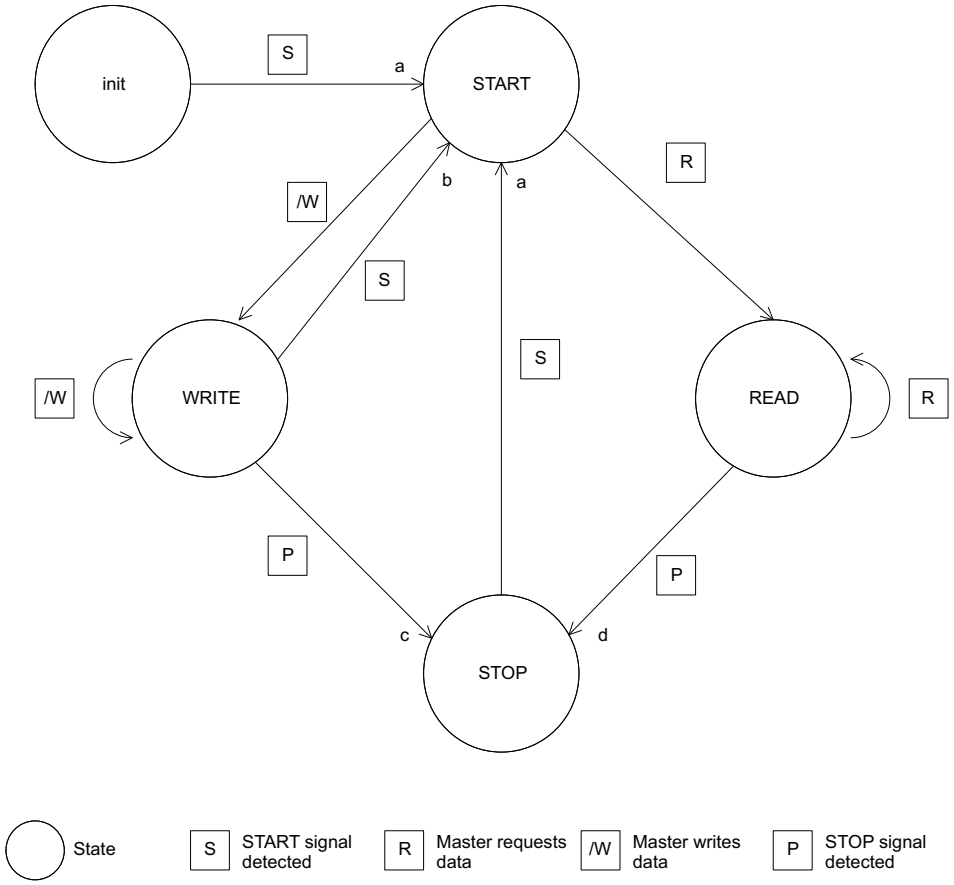


Figure 17. I²C State Machine

More explanation on each of those states is as following:

- **Init state:** When system first power on, it enters into this state.
- **Start state:** A START signal is detected, it can mean different things according to the previous state.
  - When the previous state is init or stop, as shown branch *a* in Figure 17, it indicates a new start-of-packet.
  - When the previous state is write, as shown branch *b* in Figure 17, it means that a PMBus-read command code was just issued. The master (host), expects data shifting out of the slave, i.e., MSP430F155, with the next clock pulses. In this state, the firmware calculates the PEC combined with the slave address and the write bit (1) in advance, then calls the PMBus parser to process the received read-command code.
- **Read state:** The host requests data from the slave. The slave outputs the data from the transmit buffer which is written by the PMBus parser when it processes the read-command code.
- **Write state:** The host writes data to the slave. The slave puts the received data into the receive buffer, ans calculates the PEC accumulatively if PEC is enabled.
- **Stop state:** The actions to be taken in this state depends on the previous state.
  - Write state (branch *c* in Figure 17): The master finishes sending its write command, then calls PMBus parser to process the received write-command code.

– Read state (branch *d* in Figure 17): The master stops requesting data, no action need to be taken.

4.5 Source Code

The source code is written under the Code Composer Studio™ Essentials IDE for MSP430. Some calculations use Q-math in consideration of real-time execution speed. Chapter A together with this section provides information to help in understanding the firmware design.

The source code files are listed as below:

- I^2C low level driver and PMBus parser: i2c.h, i2c.c, pec.h, pec.c, pmbus.h, pmbus.c, msgCode.h.
- A/D low level driver: adc.h, adc.c.
- System time configurations: sysTime.h, sysTime.c.
- System control state machines: control_state_idle.c, control_state_run.c, control_state_soft_start.c, control_state_start_up_delay.c, read_control.c, status_store_and_stop.c.
- Global variables and definitions: include.h, variables.c.
- Others: main.c.

5 EVM software

The Isolated Half-Bridge application runs on a reference board equipped with a programmable microcontroller (MSP430) which configures the digitally enabled PWM controller and driver (UCD8X20).

The microcontroller communicates with a host through the industry-standard power supply bus (PMBus) to configure parameters as well as read status from the microcontroller. The Windows PC runs EVM software that communicates via USB to an adapter to the PMBus that communicates with the controller.

This section explains how to install and use the EVM software for management of the Isolated Half-Bridge application.

5.1 Installation

Because the system design of the USB adapter supports USB HID, no USB driver is necessary on the PC. The EVM software application and supporting PMBus libraries are all part of the PMBus application. No custom or third-party driver is required to run the EVM software. The only requirement is that the Microsoft Windows .NET framework version 2.0 is installed on the system. This version has been installed on new PCs for several months now and can be downloaded from the Microsoft™ Web site.

To install the EVM software, extract the contents the zip file that TI provides with the EVM software and run the *setup.exe* file to install the EVM software. Users can run it from the directory that they specified during installation.

5.2 Using the EVM software

The main window for the EVM software is as follows.

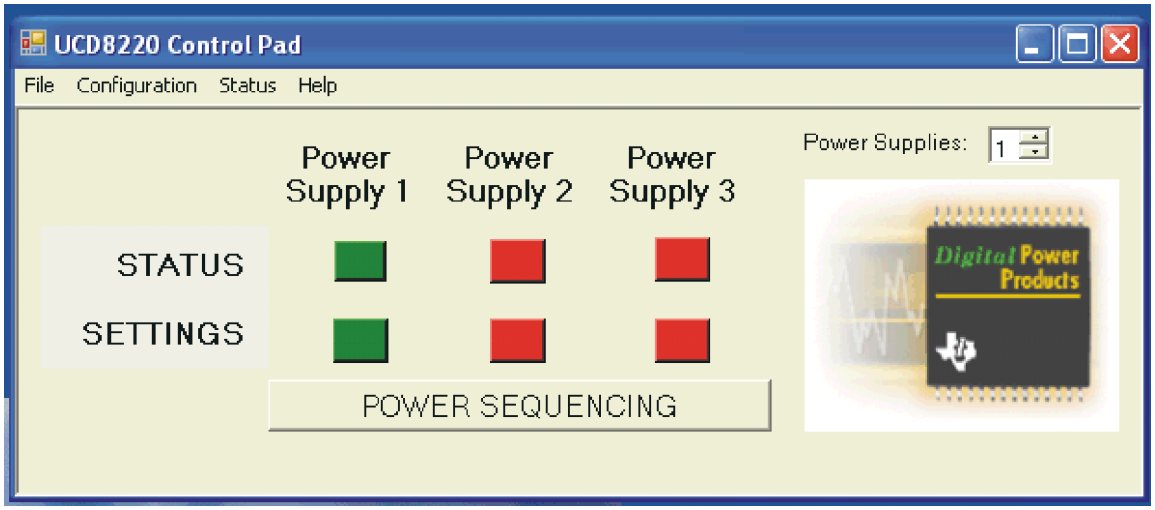


Figure 18. EVM software Main Form

Each main window button launches a new window for the given supply for its settings or status. The user can

- configure the number of power supplies (the EVM software supports between 1 and 3, on the same PMBus)
- display power supply status (read-only parameters)
- configure power supply settings (read/write parameters) for each power supply
- configure power supply sequencing (if there are more than one connected to the same PMBus) for soft-start delay and rise time
- debug the PMBus commands being used for the application and the collection of data for those commands (for debugging the EVM software itself—not needed for normal configuration and control)

5.3 PMBus Debug

The EVM software provides a window that allows low-level access to PMBus commands that the application supports. This allows the user to read and write the low-level PMBus commands to the system, for basic data-communications debugging.

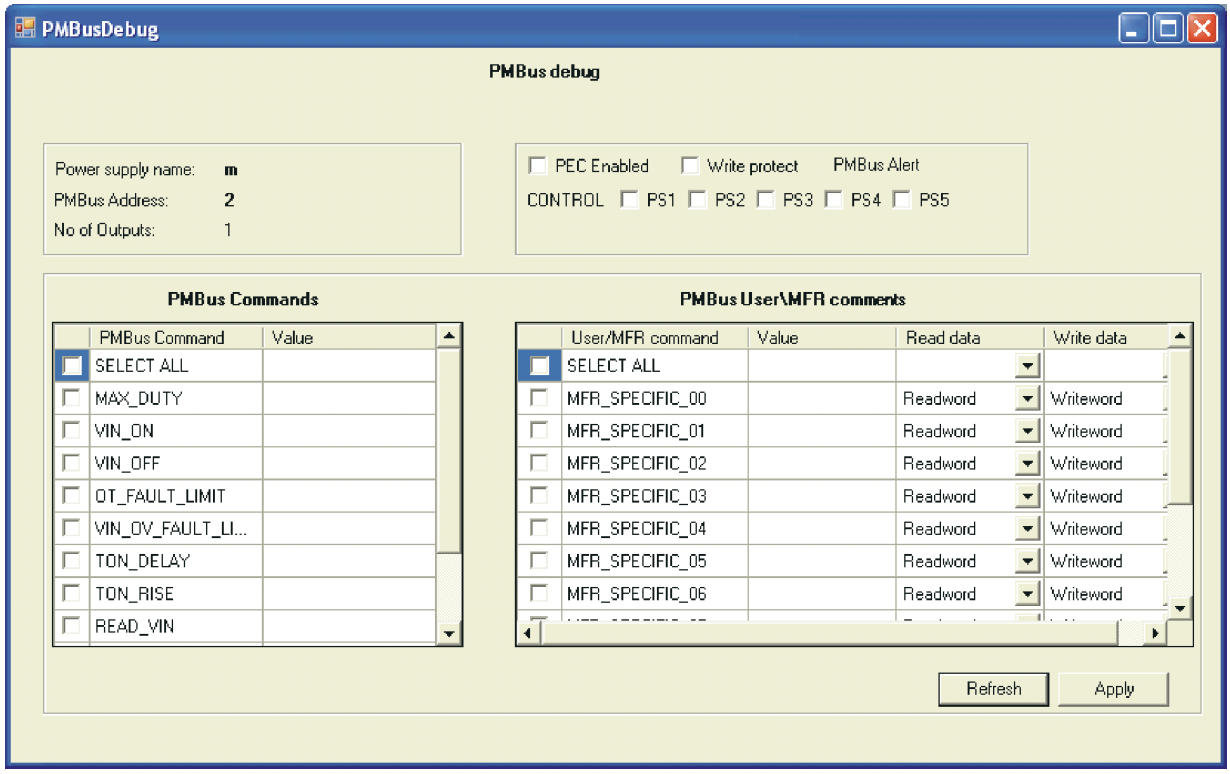


Figure 19. PMBus Debug window

In this window, the power supply name, address and output values are shown in the upper-left section of the window.

The PMBus out-of-band signals are shown on in the group box in the upper right quadrant. From that group box, the user can also set the read/write signals such as PMBus WRITE_PROTECT and CONTROL, and enable or disable PEC support.

The lower-left part of the window has the core PMBus commands that are supported by the application. To read them, click the appropriate check-boxes for each command you wish to read, then click the *Refresh* button in the lower-right. To write a new value for a command, simply enter it in the *Value* column for the desired PMBus command, then type Enter.

The lower-right part of this window houses the Manufacturer and User-Specific PMBus commands. This part of the window behaves the same as the core commands, and the user can specify the SMBus transaction type of each of these commands.

5.4 Power Supply Status

The Status window in the EVM software displays the set of read-only values for each power supply in the system.

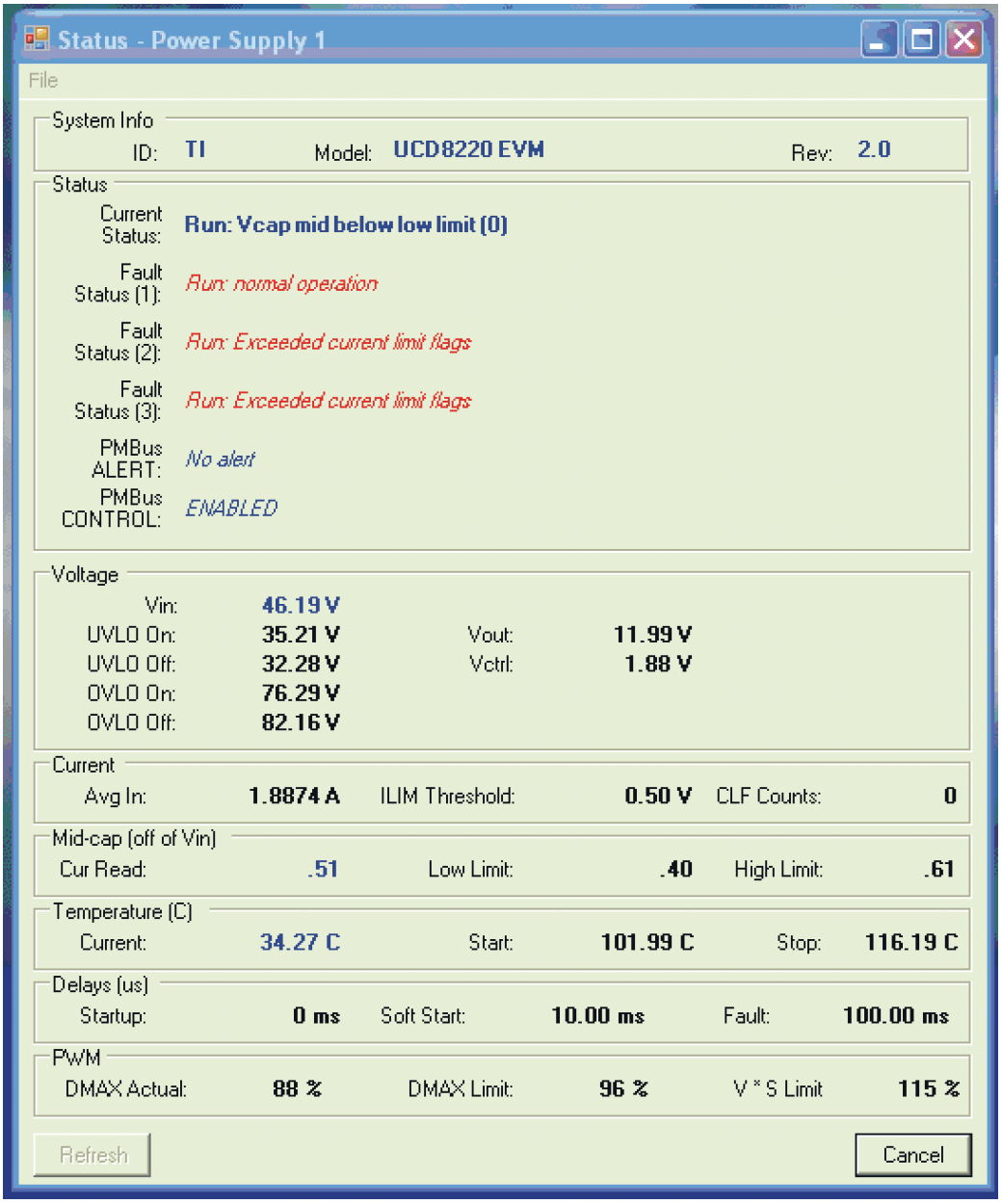


Figure 20. Power Supply Status window

This window automatically refresh its contents every few seconds. The refresh interval as well as the auto-refresh feature can be enabled or disabled from the File→Configure menu on this window. You can also change the constants that the application uses for calculation of the values on this window through the File→Constants menu (such as vRef).

When this window has been auto-refresh disabled, the user can manually refresh its contents by clicking the *refresh* button in the lower left corner.

5.5 Power Supply Settings

The Settings window in the EVM software displays the set of read-write values for each power supply in the system and provides a means for changing those values.

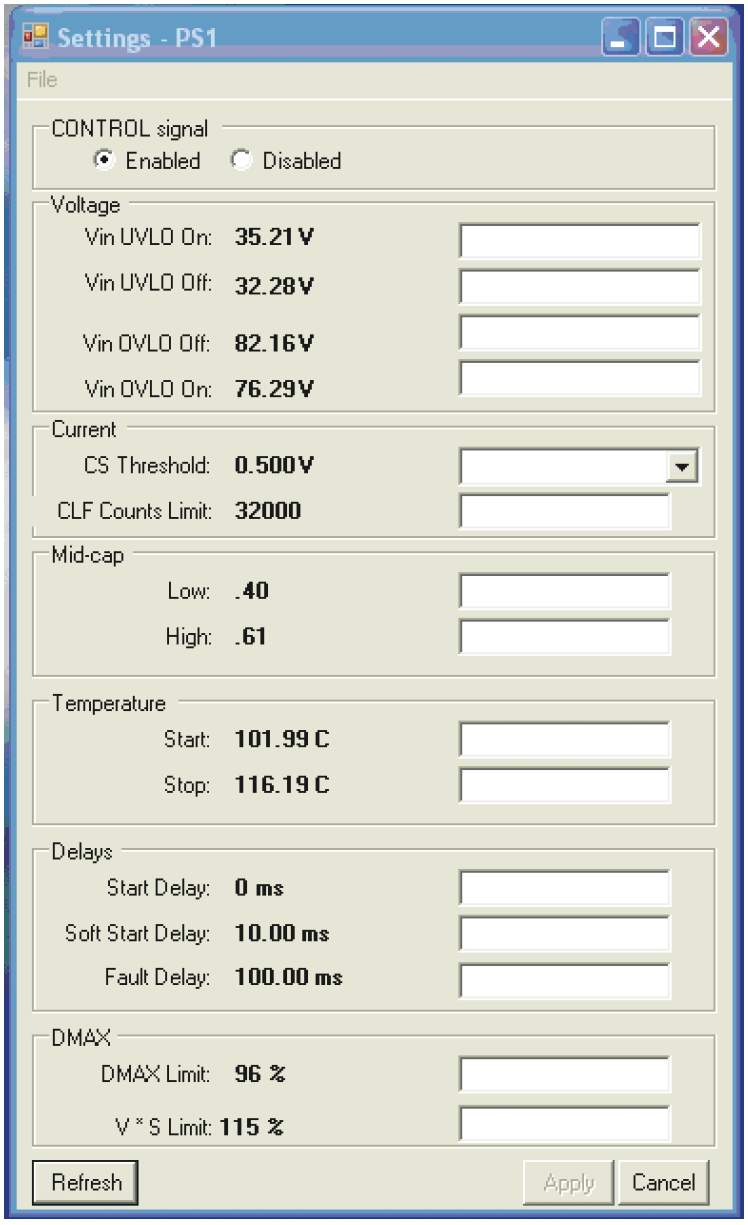


Figure 21. Power Supply Settings window

Like the status window, this window has the capability to modify the constants (under File→Constants) and configure the auto-refresh feature (File→Configure). Unlike the status window, the auto-refresh feature is off by default, so that the user can enter new values for the parameters on this window and refresh them when they wish.

Clicking the *refresh* button only reads the values and fills them in the boldface labels. To modify a variable, update the value in the text box on the right-hand side of the window without the units (e.g., 3.3 V should be entered in as 3.3), then click the *Apply* button.

5.6 Power Supply Sequencing

The last window in the EVM software configures power supply sequencing if you have more than one power supply on the same PMBus in your system.

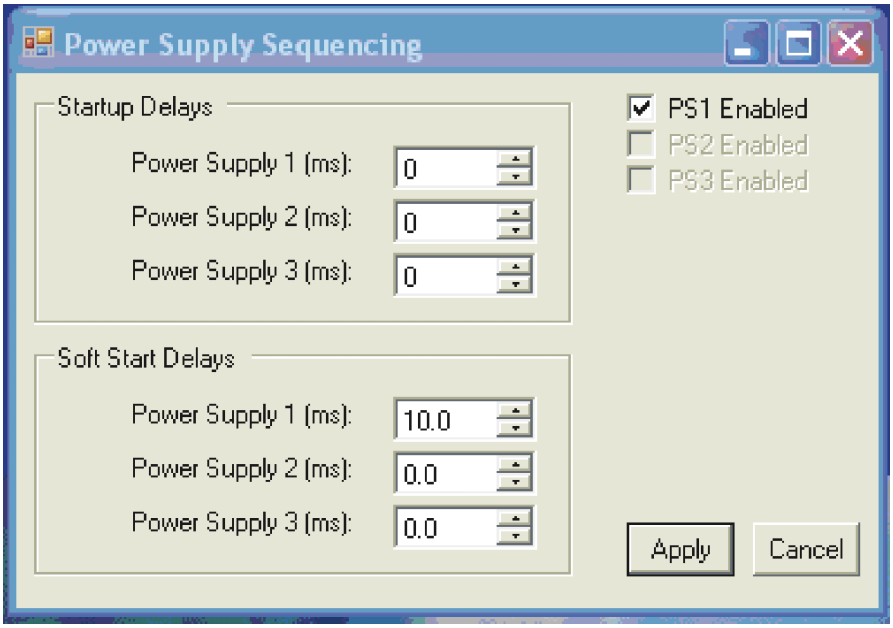


Figure 22. Power Supply Sequencing window

From this window, each power supply's startup delay and soft-start rise time can be configured. The modified values take effect when the *Apply* button is clicked.

Appendix A Q-Math and Scaling in the UCD8220EVM

A.1 Q-Math Introduction

The firmware on the UCD8220EVM uses integer math exclusively. In addition, division is avoided as much as possible. This is done to save time and memory, both RAM and ROM. With care, it is possible to retain sufficient resolution with integer math.

The technique used for doing so is often referred to as Q-math, or Q notation. It is commonly used on fixed-point DSPs for the same reasons as on this EVM. It is a simple concept, but not always simple to understand.

To understand Q-math, start with simple number base theory. For a base-10 number, place values represent powers of 10. For example, consider 143:

$143 = (1 \bullet 10^2)+(4 \bullet 10^1)+(3 \bullet 10^0)$

$143 = 100 + 40 + 3$

To represent 143 in binary, use powers of 2. The decimal number 143 becomes 0b10001111. This looks like:

$10001111 = (1 \bullet 2^7)+(0 \bullet 2^6)+(0 \bullet 2^5)+(0 \bullet 2^4)+(1 \bullet 2^3)+(1 \bullet 2^2)+(1 \bullet 2^1)+(1 \bullet 2^0)$

$10001111 = 128 + 0 + 0 + 0 + 8 + 4 + 2 + 1 = 143$

Now, consider non-integers in base 10 – numbers with a decimal point. Start with an easy-to-represent example in both binary and decimal, 1.25.

$1.25 = (1 \bullet 10^0) + (2 \bullet 10^{-1})+(5 \bullet 10^{-2})$

$1.25 = 1 + 0.2 + 0.05$

For binary, with a binary point. 1.25 becomes 1.01.

$1.01 = (1 \bullet 2^0)+(0 \bullet 2^{-1})+(1 \bullet 2^{-2})$

$1.01 = 1 + 0 + 0.25$

Try a harder one: 1.3. Skip the decimal version. The binary one presents a problem. Try 7 bits to the right of the binary point:

1.0100110 is somewhat too small:

$1.0100110 = (1\bullet 2^0)+(0 \bullet 2^1)+(1 \bullet 2^{-2})+(0 \bullet 2^{-3})+(0 \bullet 2^{-4})+(1 \bullet 2^{-5})+(1 \bullet 2^{-6})+(0 \bullet 2^{-7})$

$1.0100110 = 1 + 0 + 0.25 + 0 + 0 + 0.03125 + 0.015625 + 0 = 1.296875$

1.0100111 is somewhat too big:

$1.0100111 = (1\bullet 2^0)+(0 \bullet 2^{-1})+(1 \bullet 2^{-2})+(0 \bullet 2^{-3})+(0 \bullet 2^{-4})+(1 \bullet 2^{-5})+(1 \bullet 2^{-6})+(1 \bullet 2^{-7})$

$1.0100110 = 1 + 0 + 0.25 + 0 + 0.03125 + 0.015625 +0.0078125 = 1.3046875$

In fact, 1.3 cannot be precisely represented in binary. No matter how many places there are to the right of the binary point, an exact 1.3 is not possible. This highlights one of the goals of Q-math: retain as much precision as possible.

In Q-math, the preceding number would be described as a Q7 number, because there are 7 bits to the right of the binary point.

A.2 Q-Math in C

Because the C language does not explicitly support Q-math, users must provide the translation themselves. Here is an example:

In integer decimal math:

$166 \bullet 160 = 26560$

The binary equivalent is:

$10100111 \bullet 10100000 = 110011111000000$

In decimal math, the decimal points can be arbitrarily changed:

1.66 • 1.60 = 2.6560

Even though the actual values have changed, the digits are the same.

In binary, the same is possible:

1.0100111 • 1.0100000 = 1.10011111000000

Looking closely, one sees 1.3 (actually 1.296875) and 1.25. These are the numbers on the left. The result, on the right, is equal to 1.62109375. This is close to the result of 1.25 • 1.3 = 1.625.

Two Q7 numbers have just been multiplied together to get a Q14 number. Disregard the fact that the C compiler and the casual observer think that 166 times 160 have been multiplied.

To store the result in a single byte, just store the most-significant 8 bits, 1.1001111. This has a decimal value of 1.6171875. However, some precision has been lost.

A.2.1 Q-Math in C – Translating to Integers

It is simple to translate a Q-math number to an integer acceptable to C. Consider 1.25 and 1.3 again. The binary point is being moved to the left by 7 digits. This is equivalent to dividing by 2^7. To go the other way, multiply by 2^7, or 128. Then, 1.25 becomes 160, and 1.3 becomes 166.4, which is represented as 166.

A.2.2 Q-Math in C – An Example

Here is a quick C code example of how to do Q7 math. It is not intended to be usable code, only to show how this can be done.

```
#define CONSTANT_1 1.3
#define CONSTANT_2 1.25

#define CONSTANT_1_Q7 ((unsigned char) (CONSTANT_1 * 128))
#define CONSTANT_2_Q7 ((unsigned char) (CONSTANT_2 * 128))

unsigned char a = CONSTANT_1_Q7;
unsigned char b = CONSTANT_2_Q7;
unsigned char result;
unsigned int temporary_result;

temporary_result = ((int) a) * ((int)b); //2 Q7 numbers give a Q14 number
result = (unsigned char)(temporary_result >> 7); //change Q14 number to Q7
```

Notice that, just like decimal multiplication, the number of digits to the right of the decimal point is additive. In other words, if a Q7 number is multiplied by a Q7 number, a Q14 number is obtained.

So far, the discussion has dealt with Q7 and Q14 numbers. A few of those are in the UCD8220 firmware, but not many. For reasons of precision, most of the numbers are closer to Q15 and Q30.

A.3 Vmid_cap

Now, consider the first actual C code Q-math in the firmware. One of the power-supply voltages which the firmware monitors is called V_mid_cap. It is on the center-tap of the primary side of the main transformer, and it also is tied to capacitors. If the power supply is functioning correctly, it should be equal to ½ of Vin. Because Vin can change, the best way to describe it is as a ratio with Vin, as follows:

$$\text{Vmid\_cap\_ratio} = \frac{\text{Vad\_Vmid\_cap} \times \left(\frac{R15 + R21}{R21}\right)}{\text{Vad\_Vin} \times \left(\frac{R37 + R41}{R41}\right)} \quad \text{(A-1)}$$

Where:

Vmid_cap_ratio = Ratio of Vmid_cap to Vin
Vad_Vmid_cap = Direct A/D reading of Vmid_cap
Vad_Vin = Direct A/D reading of Vin
R15 = 1070 kΩ
R21 = 54.9 kΩ
R37 = 1070 kΩ
R41 = 27.4 kΩ

Using these values in the equation, the result is:

$$\text{Vmid\_cap\_ratio} = \frac{\text{Vad\_vmid\_cap} \times 0.5116}{\text{Vad\_Vin}} \quad \text{(A-2)}$$

It is certainly possible to solve this equation each time it is desired to check the ratio, and then to compare the ratio to the minimum and maximum limits. But division is time consuming on most microcontrollers; therefore, a different approach is selected.

Move the multiplication by 0.5116 out to the host processor. Instead of a single division, substitute two multiplications. Use Vin times the high and low limits to calculate the limits on Vmid_cap, and then compare Vmid_cap to those limits.

Here's how to do it:

First, solve the preceding equation for Vad_Vmid_cap:

$$\text{Vad\_Vmid\_cap} = \frac{\text{Vmid\_cap\_ratio}}{0.5116} \times \text{Vad\_Vin} \quad \text{(A-3)}$$

If the host processor does the divide by 0.5116, the Vad_Vmid_cap limits can be calculated by doing a simple multiply and compare. But what about the Q-math?

This particular calculation does not require high precision. It is only being used to detect gross failures, like a capacitor shorting or opening. So, the multiplication result can be fit into an integer without any real penalty.

The next step is to look at the numbers of bits. The MSP430F155 A/D has 12 bits of resolution, and all of the bits are used in this application. To get 12 bits out of the equation, the easiest way to do this is to assume that both A/Ds are Q12 numbers. Because the ratio between the two voltages is being considered, the actual voltage is not a concern, just its representation.

The *Vmid_cap_ratio* values are two values relatively close to 0.5. The power-up defaults are 0.4 for the low limit, and 0.6 for the high limit. The value is somewhere below 1. However, *Vmid_cap_ratio/0.5116* is wanted. The range for this value is less than 2. So, to get 7 bits of resolution, this number should actually be represented as a Q6 number. It is always less than 2^1. In fact, it could be called a Q1.6 number, signifying that it has 1 bit of information to the left of the binary point, and 6 to the right.

To get the same 7 bits of resolution from the Q12 A/D value (*Vad_Vin*), it needs to be shifted to the right by 5. Generally, it is good to balance the accuracy between the two variables that are to be multiplied.

The resolution is restricted to a total of 14 bits of data so that a multiply can be done with an integer result and not have an overflow. Theoretically, up to 16 bits could be used because the number is always positive, and the user could use an unsigned integer. But this is a developmental product, so 2 bits are reserved for potential unforeseen overflows. For values over 16 bits, use a **long** data type.

To convert *Vmid_cap_ratio* to *Vmid_cap_ratio/0.5116* in Q1.6 format, the host processor calculates:

$$\text{Translated\_Vmid\_cap\_ratio} = \frac{\text{Vmid\_cap\_ratio}}{0.5116} \times 2^6 \quad \text{(A-4)}$$

Or:

Translated_Vmid_cap_ratio = *Vmid_cap_ratio* x 125.1

With this value from the host processor, all the MSP430 has to do is:

```
mid_cap_low_test = ((voltage_in >> 5) * mid_cap_low_limit) >> 1 ;
  /*Q7 * Q6 = Q13 bits - shift to make it a Q12 like A to D*/

  mid_cap_high_test = ((voltage_in >> 5) * mid_cap_high_limit) >> 1 ;
  /*Q7 * Q6 = Q13 - shift to make it a Q12 like A to D*/
```

Then, all it has to do is to compare the results to the Vad_Vmid_cap.

A.4 Volt/Second Clamp

Sometimes, it is impossible to avoid division. This is the case with the Volt/Second Clamp.

The Volt/Second Clamp is a useful feature of the UCD8220 firmware. It limits the power supply's response to a transient, in effect limiting the slew rate. This is done by adjusting the pulse width of the PWM pulse fed into the CLK pin on the UCD8220. This sets the maximum pulse width that the UCD8220 can output to the power supply.

The theoretical pulse width is given by:

$$\text{Drequired} = 2\frac{(\text{Vout} + \text{Vd})}{(\text{Vin} \times \text{Ns})} \times \text{Np} \tag{A-5}$$

Where:

Drequired = pulse width (1 is whole pulse period, 0 is none)
Vout = 12
Vd = 1 (voltage drop in output side)
Vin = variable (voltage in –36 to 85 volts)
Ns = 4 (number of turns in secondary)
Np = 5 (number of turns in primary)

Putting in all the constants gives:

$$\text{Drequired} = \frac{32.5}{\text{Vin}} \tag{A-6}$$

Over the range of acceptable input voltages, Drequired varies from 0.9028 to 0.3823. This value is the theoretical steady-state power-supply output value. With Volt/Second Clamp, the firmware provides a margin above this value to allow the UCD8220 to deal with transients, but still within limits. The default margin is 15%. The Volt/Second Clamp is stated as 1.15 for this value. Therefore, the equation becomes:

$$\text{Drequired\_with\_margin} = \frac{32.5 \times \text{Volt\_second\_clamp}}{\text{Vin}} \tag{A-7}$$

But there is more. This is still an idealized value. It needs to be translated into the value used by the MSP430. Two more steps are required.

In the UCD8220EVM, the PWM pulse width is measured in counts, where 23 is equal to a D of 1. So, the equation becomes:

$$\text{PWM\_count} = 23 \times \frac{32.5 \times \text{Volt\_second\_clamp}}{\text{Vin}} = \frac{747.5 \times \text{Volt\_second\_clamp}}{\text{Vin}} \tag{A-8}$$

Second, the calculation is done with the A/D reading, Vad_Vin, not with Vin. The translation equation is:

$$\text{Vin} = \text{Vad\_Vin} \times \frac{\text{Vref}}{\text{Max\_count}} \times \frac{\text{R37} + \text{R41}}{\text{R41}} \tag{A-9}$$

Where:

Vref = 2.5 volts (A/D reference voltage)
Max_count = 4095 (A/D maximum value)
R37 = 1070 kΩ
R41 = 27.4 kΩ

So, filling in the numbers:

Vin = Vad_Vin • 0.02445

And it can be substituted back into the previous equation:

$$\text{PWM\_count} = \frac{747.5 \times \text{Volt\_second\_clamp}}{\text{Vad\_Vin} \times 0.02445} = \frac{30573 \times \text{Volt\_second\_clamp}}{\text{Vad\_Vin}} \tag{A-10}$$

So, if the host processor does the scaling, the value for the default 1.15, would be:

Scaled_volt_second_clamp = 30573 • 1.15 = 35158

This does not fit into a signed integer. To make it fit, divide by 2. This makes the multiplication constant for the host processor 15286.

Because the numerator was divided by 2, also divide the dividend by 2 to balance the effect on the equation. This can be done with a simple shift right by one. So, the equation from the C code is:

```
temp = (volt_second_margin/( (voltage_in) >> 1)) + 1;
```

The +1 on the end of the equation is added to ensure that the minimum margin is 0.15 (or whatever the margin is set to). If nothing is added, only truncated, the margin would be a maximum of 0.15. Other techniques can be used to round numbers, so that the average margin would be 0.15. However, they are outside the scope of this appendix.

This equation does not use Q notation. It is only an exercise in scaling. It does illustrate the primary principle of real-time scaling and Q-notation for real-time calculation, which is to minimize the complexity of the real-time calculations. It lays the foundation for the next section, which demonstrates addition in Q-math. This is necessary for the soft-start function.

A.5 Soft-Start Calculation

The soft-start function involves translating from time to PWM steps. Soft-start is achieved by ramping the PWM pulse width up from 0 to the maximum permitted. The maximum is the lesser of two values – the Volt Second Clamp value previously described, or the dmax_limit.

At the lower end of the voltage range, it is not possible to maintain the full Volt Second Clamp margin – it would involve a duty cycle over 100%, which is impossible. So the dmax_limit is used. In this case it is 22, because 100% is 23. It is the highest number that provides a less than 100% duty cycle.

The math for the Volt Second Clamp was covered in the previous section, so all that remains is the soft-start math.

The soft-start delay value is stated in 100-µs steps, because this is the period of the clock interrupt for the firmware. The value which must be calculated is the step size for the PWM period. The formula for the step size is simple:

$$\text{PWM\_Step\_size} = \frac{\text{Dmax}}{\text{Number\_of\_steps}} \tag{A-11}$$

Where:

Dmax = lesser of volt second clamp PWM value or Dmax_limit
Number_of_steps = soft-start delay

Division takes from 200 to 300 instruction cycles, and there are only 650 per 100-µs step. So, it is best to do only one division per step. A division is associated with the step size, and also a division is associated with the Dmax calculation. Vin is much more likely to change than soft-start delay. Therefore, the Dmax is recalculated each step, whereas the step size part of the equation is calculated only once. This is done by calculating a constant:

$$\text{Soft\_start\_multiplier} = \frac{1}{\text{Number\_of\_steps}} \tag{A-12}$$

So, the step size equation becomes

PWM_step_size = Dmax • Soft_start_multiplier

In fact, Soft_start_multiplier needs to be a factional number as well. Any value 1 is fractional. In the existing code, Q14 was chosen. This is done to have some room for expansion in a signed integer. It is possible to put in up to a Q16 number into an unsigned integer, but most soft-start delays are short, on the order of about 4 ms (or 40 counts); therefore, this is plenty of resolution. 2^{14} is 16384, so the equation for this is:

$$\text{Q14\_Soft\_start\_multiplier} = \frac{16384}{\text{Number\_of\_steps}} \tag{A-13}$$

This Q14 number is multiplied by Dmax, which is an integer that can be as high as 22. No fractional part is retained, so this number is a Q5.0 number. The two multiplied together yields a Q5.14 number. This is 19 bits. It is possible to truncate the Q14 number down to Q11 to make the result fit into a 16-bit integer, but this might adversely affect resolution in a way that would matter in the application.

The approach used in the existing code is to use a long (32-bit) variable. This does involve long multiplication and long addition, but those are not unacceptably time-consuming on this processor, because it has a hardware multiplier. If software multiplication were the only choice, it would probably be necessary to use an integer for this calculation, and/or lengthen the step-timing to fit slower calculations.

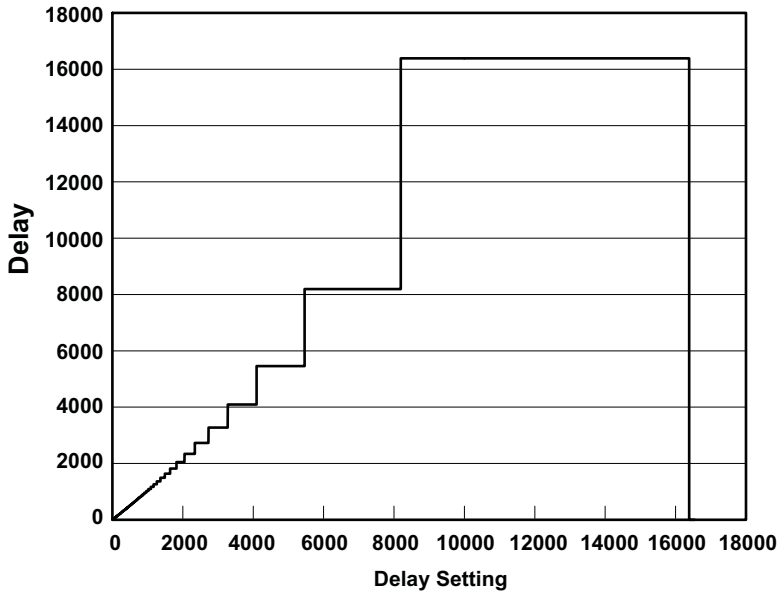
The only remaining step is to take the PWM accumulator, shift it right by 14 to make it a Q5.0 number, and load it into the PWM controller. Here is the code:

```
soft_start_accumulator = soft_start_accumulator +
                            ((long)dmax * (long)soft_start_multiplier) ;
      //and we need a Q0 (integer) for our pulse width, so we need to shift by 14
if((soft_start_accumulator >> 14 ) < dmax) //if still below dmax
    {
        CCR1 = soft_start_accumulator >> 14;
    }
    else //here if soft-start delay is over
    {
        CCR1 = dmax;
        control_state_function = control_state_run;
        status = OK_TO_RUN;
    }
```

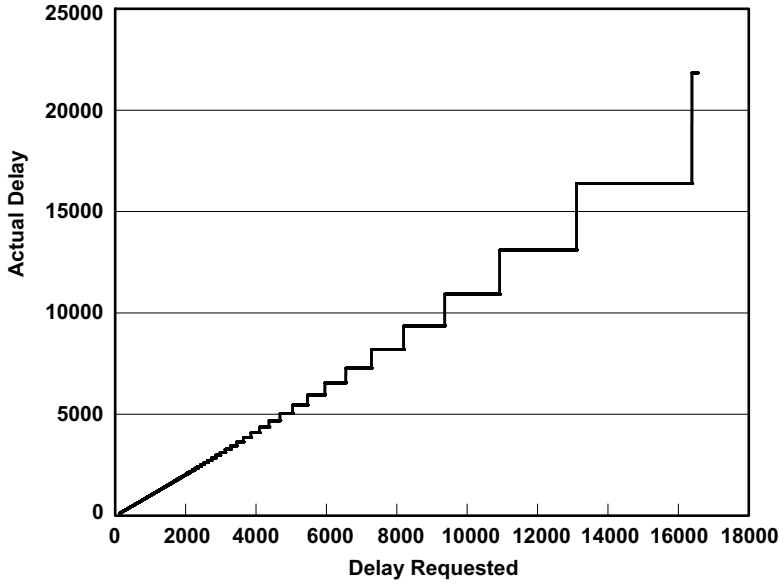
A check determines when soft-start is complete, and the run must begin.

This approach works with delays up to 16384, corresponding to 1.638 seconds. This yields a *Q14_soft_start_multiplier* of 00.00000000000001, which causes dmax to be added to the soft_start_accumulator every time. Because the soft_start_accumulator is a Q14 number, it takes quite some time for the PWM to start changing. But it does. For any delay time above 16384, soft_start_multiplier is 0, so the power supply never starts.

In this demonstration program, error-checking is not provided for this issue. It could be included in a production program. This shows one of the effects of resolution. Another effect is that all numbers between 16384 and 8193 give the same result. Then, the multiplier is 00.00000000000010. The following chart shows the effect:



The timing is usable up to about 200 ms. This should be adequate. For comparison, the following chart shows Q16.


25000
20000
15000
10000
5000
0
Actual Delay
0
2000
4000
6000
8000
10000
12000
14000
16000
18000
Delay Requested

EVALUATION BOARD/KIT IMPORTANT NOTICE

Texas Instruments (TI) provides the enclosed product(s) under the following conditions:

This evaluation board/kit is intended for use for **ENGINEERING DEVELOPMENT, DEMONSTRATION, OR EVALUATION PURPOSES ONLY** and is not considered by TI to be a finished end-product fit for general consumer use. Persons handling the product(s) must have electronics training and observe good engineering practice standards. As such, the goods being provided are not intended to be complete in terms of required design-, marketing-, and/or manufacturing-related protective considerations, including product safety and environmental measures typically found in end products that incorporate such semiconductor components or circuit boards. This evaluation board/kit does not fall within the scope of the European Union directives regarding electromagnetic compatibility, restricted substances (RoHS), recycling (WEEE), FCC, CE or UL, and therefore may not meet the technical requirements of these directives or other related directives.

Should this evaluation board/kit not meet the specifications indicated in the User's Guide, the board/kit may be returned within 30 days from the date of delivery for a full refund. THE FOREGOING WARRANTY IS THE EXCLUSIVE WARRANTY MADE BY SELLER TO BUYER AND IS IN LIEU OF ALL OTHER WARRANTIES, EXPRESSED, IMPLIED, OR STATUTORY, INCLUDING ANY WARRANTY OF MERCHANTABILITY OR FITNESS FOR ANY PARTICULAR PURPOSE.

The user assumes all responsibility and liability for proper and safe handling of the goods. Further, the user indemnifies TI from all claims arising from the handling or use of the goods. Due to the open construction of the product, it is the user's responsibility to take any and all appropriate precautions with regard to electrostatic discharge.

EXCEPT TO THE EXTENT OF THE INDEMNITY SET FORTH ABOVE, NEITHER PARTY SHALL BE LIABLE TO THE OTHER FOR ANY INDIRECT, SPECIAL, INCIDENTAL, OR CONSEQUENTIAL DAMAGES.

TI currently deals with a variety of customers for products, and therefore our arrangement with the user **is not exclusive.**

TI assumes **no liability for applications assistance, customer product design, software performance, or infringement of patents or services described herein.**

Please read the User's Guide and, specifically, the Warnings and Restrictions notice in the User's Guide prior to handling the product. This notice contains important safety information about temperatures and voltages. For additional information on TI's environmental and/or safety programs, please contact the TI application engineer or visit www.ti.com/esh.

No license is granted under any patent right or other intellectual property right of TI covering or relating to any machine, process, or combination in which such TI products or services might be or are used.

FCC Warning

This evaluation board/kit is intended for use for **ENGINEERING DEVELOPMENT, DEMONSTRATION, OR EVALUATION PURPOSES ONLY** and is not considered by TI to be a finished end-product fit for general consumer use. It generates, uses, and can radiate radio frequency energy and has not been tested for compliance with the limits of computing devices pursuant to part 15 of FCC rules, which are designed to provide reasonable protection against radio frequency interference. Operation of this equipment in other environments may cause interference with radio communications, in which case the user at his own expense will be required to take whatever measures may be required to correct this interference.

EVM WARNINGS AND RESTRICTIONS

It is important to operate this EVM within the input voltage range of 36 V to 75 V.

Exceeding the specified input range may cause unexpected operation and/or irreversible damage to the EVM. If there are questions concerning the input range, please contact a TI field representative prior to connecting the input power.

Applying loads outside of the specified output range may result in unintended operation and/or possible permanent damage to the EVM. Please consult the EVM User's Guide prior to connecting any load to the EVM output. If there is uncertainty as to the load specification, please contact a TI field representative.

During normal operation, some circuit components may have case temperatures greater than 60°C. The EVM is designed to operate properly with certain components above 60°C as long as the input and output ranges are maintained. These components include but are not limited to linear regulators, switching transistors, pass transistors, and current sense resistors. These types of devices can be identified using the EVM schematic located in the EVM User's Guide. When placing measurement probes near these devices during operation, please be aware that these devices may be very warm to the touch.

Mailing Address: Texas Instruments, Post Office Box 655303, Dallas, Texas 75265

IMPORTANT NOTICE

Texas Instruments Incorporated and its subsidiaries (TI) reserve the right to make corrections, modifications, enhancements, improvements, and other changes to its products and services at any time and to discontinue any product or service without notice. Customers should obtain the latest relevant information before placing orders and should verify that such information is current and complete. All products are sold subject to TI's terms and conditions of sale supplied at the time of order acknowledgment.

TI warrants performance of its hardware products to the specifications applicable at the time of sale in accordance with TI's standard warranty. Testing and other quality control techniques are used to the extent TI deems necessary to support this warranty. Except where mandated by government requirements, testing of all parameters of each product is not necessarily performed.

TI assumes no liability for applications assistance or customer product design. Customers are responsible for their products and applications using TI components. To minimize the risks associated with customer products and applications, customers should provide adequate design and operating safeguards.

TI does not warrant or represent that any license, either express or implied, is granted under any TI patent right, copyright, mask work right, or other TI intellectual property right relating to any combination, machine, or process in which TI products or services are used. Information published by TI regarding third-party products or services does not constitute a license from TI to use such products or services or a warranty or endorsement thereof. Use of such information may require a license from a third party under the patents or other intellectual property of the third party, or a license from TI under the patents or other intellectual property of TI.

Reproduction of information in TI data books or data sheets is permissible only if reproduction is without alteration and is accompanied by all associated warranties, conditions, limitations, and notices. Reproduction of this information with alteration is an unfair and deceptive business practice. TI is not responsible or liable for such altered documentation.

Resale of TI products or services with statements different from or beyond the parameters stated by TI for that product or service voids all express and any implied warranties for the associated TI product or service and is an unfair and deceptive business practice. TI is not responsible or liable for any such statements.

TI products are not authorized for use in safety-critical applications (such as life support) where a failure of the TI product would reasonably be expected to cause severe personal injury or death, unless officers of the parties have executed an agreement specifically governing such use. Buyers represent that they have all necessary expertise in the safety and regulatory ramifications of their applications, and acknowledge and agree that they are solely responsible for all legal, regulatory and safety-related requirements concerning their products and any use of TI products in such safety-critical applications, notwithstanding any applications-related information or support that may be provided by TI. Further, Buyers must fully indemnify TI and its representatives against any damages arising out of the use of TI products in such safety-critical applications.

TI products are neither designed nor intended for use in military/aerospace applications or environments unless the TI products are specifically designated by TI as military-grade or "enhanced plastic." Only products designated by TI as military-grade meet military specifications. Buyers acknowledge and agree that any such use of TI products which TI has not designated as military-grade is solely at the Buyer's risk, and that they are solely responsible for compliance with all legal and regulatory requirements in connection with such use.

TI products are neither designed nor intended for use in automotive applications or environments unless the specific TI products are designated by TI as compliant with ISO/TS 16949 requirements. Buyers acknowledge and agree that, if they use any non-designated products in automotive applications, TI will not be responsible for any failure to meet such requirements.

Following are URLs where you can obtain information on other Texas Instruments products and application solutions:

Products		**Applications**	
Amplifiers	amplifier.ti.com	Audio	www.ti.com/audio
Data Converters	dataconverter.ti.com	Automotive	www.ti.com/automotive
DSP	dsp.ti.com	Broadband	www.ti.com/broadband
Interface	interface.ti.com	Digital Control	www.ti.com/digitalcontrol
Logic	logic.ti.com	Military	www.ti.com/military
Power Mgmt	power.ti.com	Optical Networking	www.ti.com/opticalnetwork
Microcontrollers	microcontroller.ti.com	Security	www.ti.com/security
RFID	www.ti-rfid.com	Telephony	www.ti.com/telephony
Low Power Wireless	www.ti.com/lpw	Video & Imaging	www.ti.com/video
		Wireless	www.ti.com/wireless

Mailing Address: Texas Instruments, Post Office Box 655303, Dallas, Texas 75265