

UCD30xx

Fusion Digital Power Peripherals

Programmer's Manual

Literature Number: xxxxxx

Date

Table of Contents

UCD30xx	1
Fusion Digital Power Peripherals	1
Programmer's Manual	1
Literature Number: xxxxxx.....	1
Date	1
1 Fusion Power Peripherals	5
2 Error ADC (EADC)	5
2.1 EADC Output.....	6
2.2 AFE_GAIN	7
2.3 EADC DAC	7
2.4 Filter Front End Control Register (FLTRFECTRL)	7
2.5 Continuous EADC Triggering	8
2.5.1 DPWM Sample Trigger Register (DPWMSAMPTRIG)	8
2.5.2 OVERSAMPLE	9
2.6 Single frame triggering	9
2.7 EADC status bits	9
2.8 EADC Enable Bits	10
2.9 EADC Output.....	10
2.10 Summary of EADC Control Registers.....	10
3 Digital Fusion Compensator	11
3.1 Compensator 2P2Z and Input Scaling	12
3.1.1 Filter X Data (FLTRXDATA)	12
3.1.2 Filter Nonlinear Response Gains (FLTRNLR xxx_GAIN).....	13
3.1.3 Pages A and B	13
3.1.4 B Coefficients (FLTRCOEF1 - B01 and B11, FILTERCOEF2 - B21).....	14
3.1.5 Coefficient Scaler (COEF_SCALER)	14
3.1.6 A Coefficients (FILTERCOEF3 - A11 and A21)	15
3.1.7 FLTRCLAMP	15
3.1.8 Y values	16
3.1.9 Scaling.....	16
3.2 Compensator 1P1Z.....	17
3.3 PWM scaling.....	17
3.4 Dynamic changes to the compensator configuration/values.....	19
4 Digital Pulse Width Modulator (DPWM).....	20
4.1 Time resolution of various registers	21
4.2 PWM counter and clocks	22
4.3 Comparisons to PWM counter.....	22
4.4 DPWM modes.....	23
4.5 Normal Mode – open loop	23
4.6 Normal Mode – Closed Loop	25
4.6.1 Normal Mode Cycle Adjust	28
4.6.2 EADC sample timing in Normal Mode	28

4.7	DPWM Multiple Output mode (MULTI_OUT).....	29
4.8	DPWM Phase Shift Mode (PHASE_MODE).....	30
4.9	DPWM Resonant Mode.....	33
4.10	Resonant mode Code Example	34
4.11	Synchronizing Multiple DPWMs.	35
4.12	Other DPWM Bit Fields	36
4.12.1	CLA_CH_SEL.....	36
4.12.2	FAULT_ENA	36
4.12.3	SYNC_SLAVE_ENA and Sync In pin.....	36
4.12.4	SYNC_OUT_DIV_SEL and SYNC_OUT pin.....	36
4.12.5	GPIO with DPWM pins	37
4.12.6	Oversample	38
4.12.7	SFRAME Trigger and SFRAME_ENA.....	39
4.12.8	CHECK_OVERRIDE.....	39
4.12.9	PWM_A_PROT_DIS and PWM_A_PROT_DIS.....	39
4.12.10	HIRES_SCALE, ALL_PHASE_CLOCK_ENABLE, HIRES_DIS.....	39
4.13	Current Limit Flag (CLF) support	40
4.14	DPWM Overflow.....	41
4.15	DPWM Interrupt Register (DPWMINT)	41
5	Loop 1-4 Compensator Registers.....	41
5.1	Filter Status Register (FLTRST).....	41
5.2	Filter Control Register (FLTRCTRL).....	42
5.3	Filter X Data Register 1 (FLTRXDATA1).....	44
5.4	Filter Y Data Register 1 (FLTRYDATA1).....	44
5.5	Filter Y Data Register 2 (FLTRYDATA2).....	45
5.6	Filter Y' Data Register 1 (FLTRYPDATA1)	45
5.7	Filter Nonlinear Response Register 1 (FLTRNLR1).....	46
5.8	Filter Coefficient Register 1 (FLTRCOEF1).....	47
5.9	Filter Coefficient Register 2 (FLTRCOEF2).....	47
5.10	Filter Coefficient Register 3 (FLTRCOEF3).....	48
5.11	Filter Coefficient Register 4 (FLTRCOEF4).....	48
5.12	Filter Clamp Register (FLTRCLAMP).....	50
5.13	Filter Nonlinear Response Register 2 (FLTRNLR2).....	50
5.14	Filter Front End Control Register (FLTRFECTRL).....	51
6	Loop 1-4 DPWM Registers	51
6.1	DPWM Control Register 1 (DPWMCTRL1)	51
6.2	DPWM Control Register 2 (DPWMCTRL2)	54
6.3	DPWM Period Register (DPWMPRD)	55
6.4	DPWM Event 1 Register (DPWMEV1)	55
6.5	DPWM Event 2 Register (DPWMEV2)	55
6.6	DPWM Event 3 Register (DPWMEV3)	56
6.7	DPWM Event 4 Register (DPWMEV4)	56
6.8	DPWM Sample Trigger Register (DPWMSAMPTRIG)	56
6.9	DPWM Phase Trigger Register (DPWMPHASETRIG)	57
6.10	DPWM Cycle Adjust A Register (DPWMCYCADJA)	57
6.11	DPWM Cycle Adjust B Register (DPWMCYCADJB).....	57

6.12	DPWM Current Limit Flag Control Register (DPWMCLFCTRL).....	57
6.13	DPWM Overflow Register (DPWMOVERFLOW)	59
6.14	DPWM Interrupt Register (DPWMINT)	59
6.15	EADC Control Register (EADCCTRL)	60
6.16	EADC Status Register (EADCST)	60
6.17	EADC DAC Value Register (EADC DAC)	61
7	Other Registers mentioned.....	62
7.1	Comparator Control Register (COMPCTRL).....	62
7.2	Sync Control Register (SYNCCTRL)	62
7.3	Comparator Read Register (COMPREAD)	63
7.4	CLF Control Register (CLFCTRL)	64
7.5	Clock Control Register (CLKCNTL)	64
7.6	Clock Control Register2 (CLKCTRL).....	65
8	Appendix 1 – C program which emulates compensator	65

1 Fusion Power Peripherals

The Fusion Power Peripherals are the elements of the UCD30xx family which are specifically designed for digital power supply control. They are designed to replace the analog compensation network and PWM generation system used in analog power supplies, and add enhanced digital features to the system.

This is a simplified block diagram of the Digital Fusion peripherals:



The Error ADC accepts a differential voltage signal as an input. It measures the difference between this input and a digitally controlled set point. It passes this digital error information to the compensator.

The compensator takes the error signal and passes it through a digital filter which compensates for the characteristics of the external loop. This filter can be dynamically reprogrammed for changing power load and source characteristics. It also offers non-linear response capability for better handling of transients.

The output of the compensator is passed to a Digital PWM generator. The DPWM has two outputs, which can be used in many different ways. There are modes for synchronous rectification, multiple phases, phase shifted full bridge, and resonant configurations.

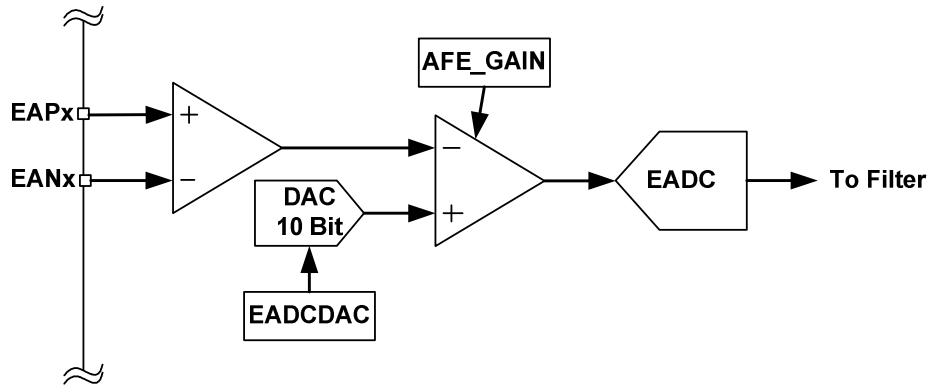
The peripherals can be run tied together as shown, or they can be used in different groupings. On devices with multiple sets of Digital Fusion peripherals, many different connection schemes are possible.

The three modules are interconnected in ways not shown on the simplified block diagram above. For example, the DPWM can be used to trigger the Error ADC, which will trigger the Compensator at the end of its conversion.

The documentation which follows starts with the EADC, goes onto the Compensator, and then to the DPWM, but there will be considerable overlap because of this interconnection.

2 Error ADC (EADC)

Here is a simplified block diagram of the EADC:



2.1 EADC Output

The EADC output is a 6 bit two's complement number. Nominally, the step size is 8 millivolts. But this can be changed by changing the AFE_GAIN control bits.

Error voltage (millivolts)

EADC	Gain = 1	Gain = 2	Gain = 4	Gain = 8
Output				
0x20	-256	-128	-64	-32
0x21	-248	-124	-62	-31
...
0x3e	-16	-8	-4	-2
0x3f	-8	-4	-2	-1
0x00	0	0	0	0
0x01	8	4	2	1
0x02	16	8	4	2
...
0x1e	240	120	60	30
0x1f	248	124	62	31

For systems with rapid, large transients, a gain of 1 is ideal. For systems which require very precise regulation, a gain of 8 is indicated. It is possible to change the gain while the power supply is operating. This can be done to cope with transients, or for system imposed changes, such as a rapid ramp down or ramp up.

2.2 AFE_GAIN

The AFE_GAIN bits are in the Filter Nonlinear Response Register 1 (FLTRNLR1). The C code to write to it for filter 1 is:

```
Filter1Regs.FLTRNLR1.bit.AFE_GAIN = 0;
```

The AFE_GAIN values work as follows:

AFE_GAIN	0	1	2	3
EADC gain	1	2	4	8

2.3 EADCDAC

The EADCDAC has a range of 0 to 1.6 volts nominal. There are 10 bits in the EADCDAC register, with a step size of about 1.5625 millivolts. So the values work as follows:

EADCDAC	Voltage
0	0
1	.00156
2	.00312
...	
0x3ff	1.60

The EADCDAC register is found in the DPWM registers. The C code to write to it is:

```
Dpwm1Regs.EADCDAC.bit.DAC_VALUE = 0x0;
```

The EADCDAC can be changed while the power supply is running. It can be used in voltage mode for ramp up and ramp down, for example.

2.4 Filter Front End Control Register (FLTRFECTRL)

The FLTRFECTRL register has 2 bits – SC_CLK_DIV_2, and SC_GAIN_FILTER_SELECT.

These bits control the switch cap (SC) filter on the input to the EADC.

For proper operation, SC_CLK_DIV_2 needs to be changed from its default value of 0 to a 1 to enable the switch cap filter clock to be divided by 2. If more than one EADC on a chip is being used, this must be done synchronously so that the filters will all be on the same phase. Here is a sample C code to perform this function:

```
#define nop                               asm(" nop");
#define FLTRFECTRL_INIT 9
```

```
// Bits 0:1 Reserved = 01.
// Bit 2 0 = Disable SC noise filter.
// Bit 3 1. Selects the switch cap divider to 2X rate.

// !!!IMPORTANT!!! DO NOT CHANGE THIS SECTION. THE EXACT TIMING IS CRITICAL!
// The SC clocks on all four CLAs must have the same phase.
// To ensure that, the CPU must switch them from 1x mode to 2x mode at the same
// relative time. Since the CPU runs at OSC_H/8, the timing can be ensured by
// adding nops so that the STR assembly instructions are always spaced by a multiple
// of 4 CPU clock ticks. The spacing between STRs is 8 CPU ticks when using 3 nops.
// The last write does not need any nops after it.
Filter1Regs.FLTRFECTRL.all = FLTRFECTRL_INIT;      nop; nop; nop;
Filter2Regs.FLTRFECTRL.all = FLTRFECTRL_INIT;      nop; nop; nop;
Filter3Regs.FLTRFECTRL.all = FLTRFECTRL_INIT;      nop; nop; nop;
Filter4Regs.FLTRFECTRL.all = FLTRFECTRL_INIT;
```

Interrupts must be disabled when the above code is run, since any interrupt would change the timing.

If only one EADC is being used on the chip, and the rest are disabled, synchronization is not required. In that case, this statement is all that is required:

```
Filter1Regs.FLTRFECTRL.bit.SC_CLK_DIV_2 = 1;
```

SC_GAIN_FILTER_SELECT, if it is a 1, enables a switched cap filter for the front end of the EADC. Here is the C code:

```
Filter1Regs.FLTRFECTRL.bit.SC_GAIN_FILTER_SEL = 1;
```

This filter is a low pass filter with a cutoff well above the sampling frequency for the EADC. Depending on system architecture and noise sources it may or may not be helpful to enable the filter.

2.5 Continuous EADC Triggering

The trigger for the EADC comes from the DPWM, so the trigger control is in those registers. The Compensator can be set up to be triggered by the EADC.

The EADC can be triggered automatically, or by software. Normally automatic triggering is used, and it is controlled by DPWMSAMPTRIG and OVERSAMPLE

2.5.1 DPWM Sample Trigger Register (DPWMSAMPTRIG)

The DPWMSAMPTRIG register determines where in the DPWM period the EADC will sample the error. It can be changed while the power supply is operating. It can be changed to keep the sample time at a specific point in the DPWM pulse, or to keep the sample time away from edges which could cause noise on the sample.

It is a 12 bit register. Its timebase is equal to the DPWM low resolution clock divided by 4.

For more information, see the DPWM section of this guide.

To write to it:


```
#define PERIOD 2500
```

```
#define HIGH_SAMPLE_TRIGGER ((PERIOD/4) * .75) //put trigger at 75% of period
```

```
Dpwm1Regs.DPWMSAMPTRIG.all = HIGH_SAMPLE_TRIGGER; //trigger late  
//in period;
```

2.5.2 OVERSAMPLE

The OVERSAMPLE bits in the DPWMCTRL1 register make it possible to run the EADC multiple times in one PWM period. This is intended to increase the bandwidth of the filter by giving it information at a higher rate.

There are two OVERSAMPLE bits. They work like this:

- 00 – Default – 1 sample time controlled by DPWMSAMPTRIG
- 01 – 2X Oversampling – 1 sample at $\frac{1}{2}$ of DPWMSAMPTRIG value, one at total value
- 10 – 4X Oversampling – samples at $\frac{1}{4}$, $\frac{1}{2}$, $\frac{3}{4}$ of DPWMSAMPTRIG, and at total value
- 11 – 8X Oversampling – 8 samples total, starting at $\frac{1}{8}$ of DPWMSAMPTRIG, and ending at total value.

Here is the C code:

```
Dpwm1Regs.DPWMCTRL1.bit.OVERSAMPLE = 0; //No oversampling
```

2.6 Single frame triggering

It is also possible to trigger the EADC in software. To do this, the SFRAME_ENA and SFRAME_TRIGGER bits in DPWMCTRL1 are used.

If the SFRAME_ENA is set to a 1, then the SAMPTRIG triggering is disabled, and SFRAME_TRIGGER is used instead. When SFRAME_TRIGGER is set, the EADC is triggered. SFRAME_TRIGGER is self clearing, so there is no need to clear it before setting it again to trigger the next frame.

Here is the C code:

```
Dpwm1Regs.DPWMCTRL1.bit.SFRAME_ENA = 1; //enable single  
//frame trigger  
  
Dpwm1Regs.DPWMCTRL1.bit.SFRAME_TRIGGER = 1; //trigger a  
//frame
```

2.7 EADC status bits.

There are 3 EADC status bits. The EADC Status register (EADCST) contains a single bit called EOC, which indicates that the conversion is complete.

```
result = Dpwm1Regs.EADCST.bit.EOC;
```

The other two bits are in the Filter Status Register (FLTRST). They are EADC_RAIL_HIGH and EADC_RAIL_LOW.

EADC_RAIL_HIGH when set indicates that the last EADC reading was at the maximum value of 31 (0x1f)

EADC_RAIL_LOW when set indicates that the last EADC reading was at the minimum value of -32 (0x20)

```
result = Filter1Regs.FLTRST.bit.EADC_RAIL_HIGH;
result = Filter1Regs.FLTRST.bit.EADC_RAIL_LOW;
```

2.8 EADC Enable Bits

Finally, there are two bits which enable the EADC. They are in the EADCCTRL register, EADC_ENA and SCFE_ENA. They default to the on setting (1). They should be disabled if that particular EAD is not being used to reduce noise and power consumption.

```
Dpwm1Regs.EADCCTRL.bit.EADC_ENA = 0; //disable ead
Dpwm1Regs.EADCCTRL.bit.SCFE_ENA = 0; //and switched cap front end.
```

2.9 EADC Output

Normally the EADC output goes to the compensator, and that is where it can be read. If the CLA is enabled to take input from the EADC, the latest EADC output can be found in FLTRXDATA1.XN. This is the 6 bit EADC value sign extended to 8 bits.

```
result = Filter1Regs.FLTRXDATA1.bit.XN;
```

XN1 and XN2 in the same register will contain the previous two samples from the EADC.

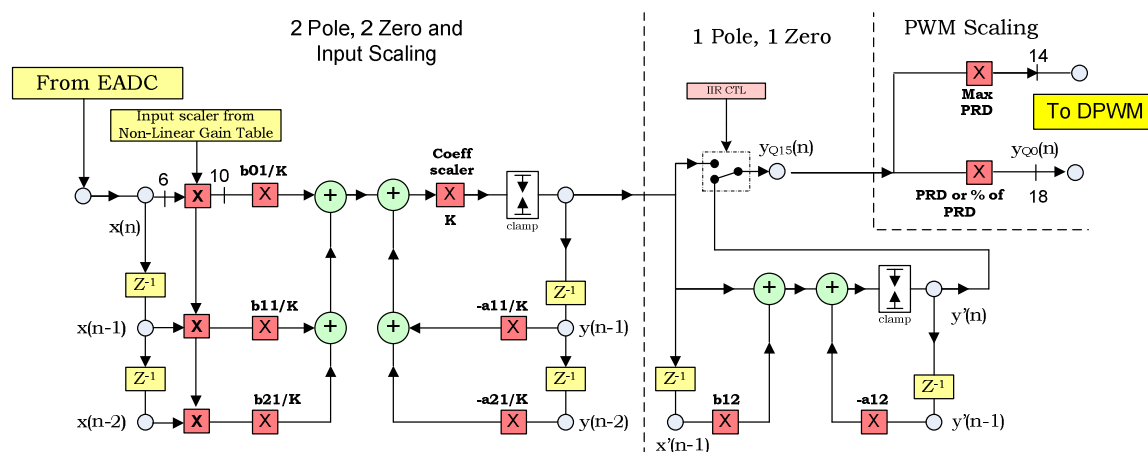
2.10 Summary of EADC Control Registers

Register	Field	Bits	Function
FLTRST	EADC_RAIL_HIGH	1	Set if EADC at high end of range
	EADC_RAIL_LOW	1	Set if EADC at low end of range
FLTRXDATA1	XN	8	Most recent EADC sample
	XN1	8	Second most recent EADC sample
	XN2	8	Third most recent EADC sample
FLTRNLR1	AFE_GAIN	2	Gain of EADC front end
FLTRFECTRL	SC_CLK_DIV_2	1	Must be set synchronously for all modules
	SC_GAIN_FILTER_SEL	1	Enables a prefilter for EADC
DPWMCTRL1	OVERSAMPLE	2	Oversampling rate for EADC

	SFRAME_TRIGGER	1	Software controlled EADC trigger
	SFRAME_ENA	1	Enable software trigger for EADC
			Disable DPWM based trigger
DPWMSAMPTRIG	All	12	Timing for EADC sampling
EADCCTRL	SCFE_ENA	1	Enable front end for EADC
	EADC_ENA	1	Enable EADC
EADCST	EOC	1	End of conversion
EADC_DAC	DAC_VALUE	10	Reference voltage for error calculation

3 Digital Fusion Compensator

Here is an overall block diagram of the Digital Fusion Compensator:



It incorporates 3 major sections. The first section, on the left, is a 2 pole, 2 zero digital filter (2P2Z), which also includes scaling for the input from the EAD and clamping of the output with both high and low limits.

The second section is an optional 1 pole, 1 zero filter, which can be used in conjunction with the 2P2Z to give a 3P3Z filter if desired. It also incorporates clamping.

The output of these filters goes to the third section, an output scaling section. This section automatically normalizes the filter output to the DPWM period, so that the period length can be changed and the DPWM percentage will stay the same.

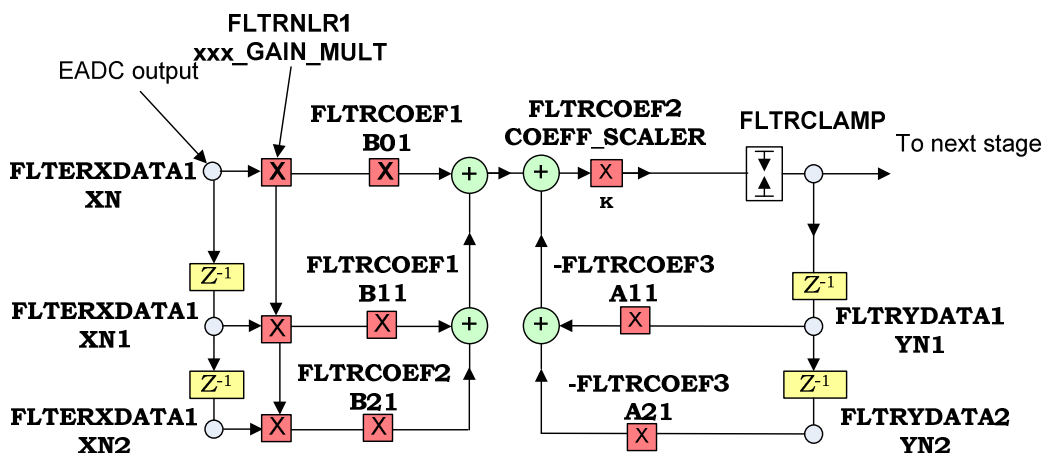
For a C program which describes the exact numerical effect of the compensator, see the Appendix.

These sections will be analyzed one at a time.

The compensator is often referred to as a filter, since that is its primary function. Also in some of the register names it is referred to as CLA, which stands for Control Law Accelerator, a former term for an earlier version.

3.1 Compensator 2P2Z and Input Scaling

This is the main section of the compensator. Most power supply algorithms call for a 2P2Z compensation network, so all the filtering takes place in this section. The drawing below shows the register names associated with each filter term.



The basis for the 2P2Z is a standard biquad structure, but it has several unique enhancements that make it much better for digital power. In addition, the scaling of values at all stages is carefully defined to reduce system cost but still provide plenty of precision.

3.1.1 Filter X Data (FLTRXDATA)

The Filter X data is used for the feed forward part of the filter, the zeroes. This data comes directly from the EADC, or it can be written to directly if the USE_CPU_SAMPLE bit in the Filter Control Register (FLTRCTRL) is set. This mode is useful for testing filter coefficients with a known input pattern.

The C code for the USE_CPU_SAMPLE bit is:

```
Filter1Regs.FLTRCTRL.bit.USE_CPU_SAMPLE = 1;
```

For the X data, it is:

```
Filter1Regs.FLTRXDATA1.bit.XN = 0;
Filter1Regs.FLTRXDATA1.bit.XN1 = 0;
Filter1Regs.FLTRXDATA1.bit.XN2 = 0;
```

The X data is a 6 bit signed number, just like the output of the EADC. The values are stored on 8 bit boundaries, and will return an 8 bit sign extended number on a read. This is done so that a byte read assembly language instruction will preserve the correct sign. The actual values are still 6 bits wide.

Every time the filter is run, after the calculation, XN1 is copied to XN2, and XN is copied to XN1. This way the filter is ready for the next sample to be put into XN from the EADC.

3.1.2 Filter Nonlinear Response Gains (FLTRNLR xxx_GAIN)

This is one of the features of the Digital Fusion Compensator that adapts it for digital power. With small values from the EADC this value is taken from the Filter Nonlinear Response Register 1 (FLTRNLR1 NOM_GAIN_MULT). It scales the X register values up from 6 bits to a maximum of 10 bits. It is a 6 bit value. The C code works like this:

```
Filter1Regs.FLTRNLR1.bit.NOM_GAIN_MULT = 4;
```

But if the EADC has a value far from zero, this causes several other values to be used for the gain multiplier, depending on the input to the EADC. Here is the C code for all of these values:

```
Filter1Regs.FLTRNLR1.bit.NEG_LRG_GAIN_MULT = 8;  
Filter1Regs.FLTRNLR1.bit.NEG_MID_GAIN_MULT = 6;  
Filter1Regs.FLTRNLR1.bit.NOM_GAIN_MULT = 4;  
Filter1Regs.FLTRNLR1.bit.POS_MID_GAIN_MULT = 6;  
Filter1Regs.FLTRNLR1.bit.POS_LRG_GAIN_MULT = 8;
```

The example above shows a hypothetical situation where the gain – and therefore the filter response speed – increases as the EADC error becomes farther from zero.

The exact EADC values at which the gain factors apply is dictated by these register bits:

```
//below this value, uses NEG_LRG_GAIN_MULT  
Filter1Regs.FLTRNLR2.bit.LIMIT0 = -30;  
//above this value, uses NEG_MID_GAIN_MULT  
Filter1Regs.FLTRNLR2.bit.LIMIT1 = -16;  
//above this value, uses NOM_GAIN_MULT  
Filter1Regs.FLTRNLR2.bit.LIMIT2 = 16;  
//above this value, uses POS_MID_GAIN_MULT  
Filter1Regs.FLTRNLR2.bit.LIMIT3 = 30;  
//above this value, uses POS_LRG_GAIN_MULT
```

The comments are pretty self-explanatory. If XN is below LIMIT0, NEG_LRG_GAIN_MULT is used. Above that point, but below LIMIT1, NEG_MID_GAIN_MULT is used. And so on. This makes it possible to have faster response to voltages that are further away from the desired value. It is also possible to decrease the gain, and slow down response to transients if desired.

To avoid nonlinear calculations, simply load all of the GAIN_MULT registers with the same value.

3.1.3 Pages A and B

All of the filter coefficients and many of the filter controls are actually duplicated in two pages, A and B. This is provided so that dynamic switching between two filter responses can be done while the system is running. This can be used for transients, for low power/high power efficiency, or for any other purpose.

In addition, the page not currently in use can be written to, making it possible to switch between an infinite number of possible filter responses without shutting down the system.

There are three bits which select which page is active for different sets of coefficients:

```
Filter1Regs.FLTRCTRL.bit.NL_PG_CONTROL = 0;  
//page A active for nonlinear gain  
Filter1Regs.FLTRCTRL.bit.CLAMP_PG_CONTROL = 0;
```

```
//page A active for clamp
Filter1Regs.FLTRCTRL.bit.BNA_ACTIVE = 0;
//coefficient page A active
```

When page A is active, writes automatically go to page B, and vice versa.

There is one bit which selects which page is read from for all cases:

```
Filter1Regs.FLTRCTRL.bit.BNA_READ = 1; //coefficient page B read
```

C statements of the type used in this document actually have a read included in them. They read the entire register and then set or clear the appropriate bits and write the entire value back to the register. If the write and read pages are out of phase, errors are very likely to occur. So it is necessary, in general to work as the above sample shows.

Enable page A for the filter so that writes will occur to page B. Then enable page B for read as well. And do the opposite if A requires modification.

There are also 3 bits to indicate which page is active:

```
status = Filter1Regs.FLTRST.bit.CLAMP_ACTIVE_PG;
status = Filter1Regs.FLTRST.bit.COEF_PAGE;
status = Filter1Regs.FLTRST.bit.NL_ACTIVE_PG;
```

These bits are only needed when the filter is active. The filter will not switch pages until the end of the filter cycle, so before writing to the page, verify that the proper page is active.

3.1.4 B Coefficients (FLTRCOEF1 - B01 and B11, FILTERCOEF2 – B21)

The B coefficients are signed numbers 12 bits in length. . For more information, see the scaling section.

Their C representation is:

```
Filter1Regs.FLTRCOEF1.bit.B01 = 400;
Filter1Regs.FLTRCOEF1.bit.B11 = 200;
Filter1Regs.FLTRCOEF2.bit.B21 = 100;
```

3.1.5 Coefficient Scaler (COEF_SCALER)

COEF_SCALER is used to permit the A coefficients to have values greater than 2. In effect, it acts like a exponent in a floating point number. The C code for it is:

```
Filter1Regs.FLTRCOEF2.bit.COEF_SCALER = 0;
```

This code will permit values of up to almost 2 for the a coefficients. COEF_SCALER can go all the way to 15, which will permit a maximum value of 65535 for the a coefficients. For more information on this, see the scaling section. It is not advised to use values of COEF_SCALER above 8, however, as filter performance may degrade.

COEF_SCALER is a 4 bit unsigned number.

As shown in the diagram COEF_SCALER is a multiply. It is actually used as a shift, so a COEF_SCALER value of 0 is equivalent to a multiply by 1, COEF_SCALER = 1 is equivalent to a multiply by 2, 2 is 4, and so on.

To put it another way, the multiply is by 2 to the power of COEF_SCALER.

The page for COEF_SCALER is also controlled by:

```
Filter1Regs.FLTRCTRL.bit.BNA_ACTIVE = 0;  
//coefficient page A active
```

3.1.6 A Coefficients (FILTERCOEF3 – A11 and A21)

Filter Coefficients A11 and A21 are 12 bit signed numbers. For more information, see the scaling section. Note that to simplify the chip design, the coefficients must be two's complement of the desired coefficients. This removes the need for subtraction hardware in the filter.

Their C code is:

```
Filter1Regs.FLTRCOEF3.bit.A11 = 400;  
Filter1Regs.FLTRCOEF3.bit.A21 = 200;
```

3.1.7 FLTRCLAMP

There are two signed 16 bit values in FLTRCLAMP:

```
Filter1Regs.FLTRCLAMP.bit.CLAMPH = 16384; \\50%  
Filter1Regs.FLTRCLAMP.bit.CLAMPL = -16384; \\-50%
```

If the value passing through the clamp is higher than CLAMPH, it will be adjusted down to the CLAMPH value. If it is lower than CLAMPL, it will be adjusted to the CLAMPL value.

CLAMPH is the way to impose a maximum pulse width on the PWM output. The CLAMPH value has a maximum of $2^{15} - 1$. To set a maximum PWM percentage, make CLAMPH that percentage of 2^{15} .

CLAMPL can be used to impose a minimum percentage in the same way.

The clamp values are actually imposed inside the filter loop. If they were outside the filter, the filter could, at the limit, saturate its integrating component. In this case when the input moved away from the limits, it would take a long time for the integrating component to respond.

Putting the clamp inside the loop prevents this, but it has another effect. Since the Y values are clamped, they do not correspond properly to the X values for the same sample time. This causes the filter to be noisy at the limit. In practice this is not a serious problem, except if 0 pulse width is desired.

If 0 pulse width is desired, it is wise to set the CLAMPL value to a negative value, say -30%. This way the filter will never reach the clamp, and there will be no noise on the output. Or even if there is noise because of a negative offset on the EADC input, the noise will stay negative.

The page for CLAMPL and CLAMPH is controlled by:
Filter1Regs.FLTRCTRL.bit.CLAMP_PG_CONTROL = 0;

//page A active for clamp

3.1.8 Y values

The Y values are signed 16 bit values. They are the values used for the feedback portion of the filter. Reading YN1 after the filter has executed gives the latest value given to the DPWM. It can be used to calculate the PWM percentage by the equation $YN1 / 2^{15}$.

The C code to read these values is:

```
status = Filter1Regs.FLTRYDATA1.bit.YN1;
status = Filter1Regs.FLTRYDATA2.bit.YN2;
```

These registers can also be written to, if it is desired to initialize the filter to a certain state. This is best done with the filter disabled.

3.1.9 Scaling

The hardware expects the coefficients to be scaled down to fractions and then saved in 2's complement form. This is done by dividing all of the 2p/2z coefficients by a 2ⁿ integer that is larger than the largest coefficient. The following example illustrates this point.

Example:

2p/2z

1p/1z(optional)

$$\frac{14.35 - 24.635z^{-1} + 10.418z^{-2}}{1 - 1.521z^{-1} + 0.521z^{-2}}$$

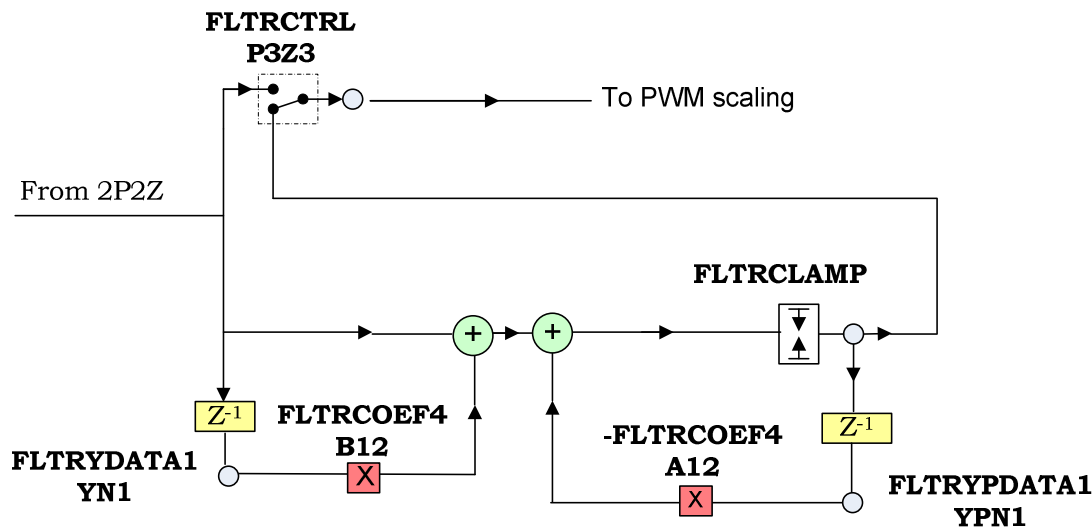
$$\frac{1 - 0.612z^{-1}}{1 - 0.128z^{-1}}$$

B01 = 14.350 =>(14.350/2⁵)*2¹¹), Convert to Hex and 2's complement = 0x0396
 B11 = 24.635 =>(24.635/2⁵)*2¹¹), Convert to Hex and 2's complement = 0xF9D7
 B21 = 10.418 =>(10.418/2⁵)*2¹¹), Convert to Hex and 2's complement = 0x029B
 A11 = -1.521 =>-(-1.521/2⁵)*2¹¹), Convert to Hex and 2's complement = 0x0061
 A21 = 0.521 =>-(0.521/2⁵)*2¹¹), Convert to Hex and 2's complement = 0xFFDF
 B12 = -0.612 =>-(-0.612)*2¹¹), Convert to Hex and 2's complement = 0xFB1B
 A12 = -0.128 =>-(-0.128)*2¹¹), Convert to Hex and 2's complement = 0x0106

Notice that the scaling factor in the above example is 2⁵ = 32, which is the smallest 2ⁿ that is larger than the largest coefficient (24.635 in this example). The scaling factor exponent (5 in this ex) is programmed into the device for use in the hardware. This scaling factor is also stored in banks so that each independent coefficient set is scaled separately. The scaling factor is stored in the B21 coefficient register as bits [3:0]. The optional 3rd pole and 3rd zero coefficients are always required to be fractional. All the fractional coefficients are then multiplied by 2¹¹ to convert to a 12-bit Q11 format and then programmed into the device as a hex number. As shown in this example, the "A" coefficients are sign changed to account for the negative sign in the difference equation. This allows hardware to be implemented only with adders.

3.2 Compensator 1P1Z

There is also an optional 1 pole, 1 zero filter in the Digital Fusion Compensator peripheral. Here is a diagram of this portion:



The 1P1Z is much simpler than the 2P2Z. There is no coefficient scaler. The filter clamp is the same as in the 2P2Z. There are only two coefficients.

The feedforward "X" value is the same as the YN1 value for the 2P2Z. The coefficients and the feedback value have this C code:

```
Filter1Regs.FLTRCOEF4.bit.A12 = 0;
Filter1Regs.FLTRCOEF4.bit.B12 = 0;

status = Filter1Regs.FLTRYPDATA1.bit.YPN1;
```

To turn on this filter, use this C code and set the P3Z3 bit:

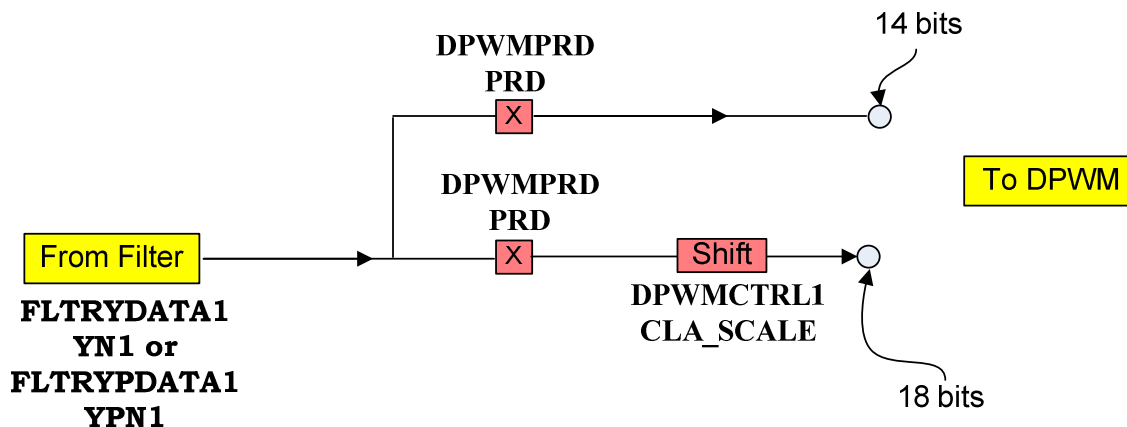
```
Filter1Regs.FLTRCTRL.bit.P3Z3 = 1;
```

If this bit is cleared, the 1P1Z filter is not executed. It takes 7 clock cycles for the filter to run with only 2P2Z, but 9 cycles are required for a 3P3Z

Since there is no coefficient scaling, the coefficients must be within the range of 1 to -1 before scaling.

3.3 PWM scaling

The PWM scaling section could really be included in either the DPWM or compensator section, as it is the interface where the filter output is translated to a form usable by the DPWM. Here is a diagram:



The filter output, which can come from two different registers depending on whether the filter is 2P2Z or 3P3Z, is multiplied by the DPWM period

```
Dpwm1Regs.DPWMCTRL1.bit.CLA_SCALE = 0;
```

```
Dpwm1Regs.DPWMPRD.bit.PRD = 250;
```

The PWM Scaler is designed so that a full scale output from the Compensator will provide a full scale input to the DPWM. For example, if the Compensator output is 0x4000 (16384), equivalent to 50%, and the DPWM period is 0x100 (256), the multiplication is as follows:

$0x4000 \cdot 0x100 = 0x400000$ or

$$16384 \cdot 256 = 4194304$$

To get the correct alignment with the DPWM, this value has to be shifted right by 11 bits.

So we get:

$$0x400000 \gg 11 = 800 \text{ or}$$

$$4194304 / 2048 = 2048$$

This is the correct result even though 50% of 0x100 is actually 0x80. There are an additional 4 bits of high resolution output available for the DPWM pulse width that are not available for the period register. This is explained in the next section. This is why the output of the PWM Scaler is 18 bits.

The 14 bit output which does not offer a shift capability is used for resonant mode only.

CLA_SCALE offers several shifts of the output of the multiply operation:

000 = no shift
001 = multiplied by 2
010 = divided by 2
011 = multiplied by 4
100 = divided by 4
101 = multiplied by 8
110 = divided by 8
111 = no shift

“CLA” is a historical remnant from when the Filter was called “Control Law Accelerator”.

The divide functions may be useful for cases when the maximum pulse width is less than 50%, because they make it possible to use the full dynamic range of the Compensator without exceeding 50% duty cycle.

The filter always takes the period from its associated DPWM. So, for example, Filter1 gets its period from DPWM1. So if Filter1 is used also to drive DPWM2, and DPWM2 has a different period, the value given to DPWM2 will still be the same. The PWM percentage will therefore be different.

However, the CLA_SCALE is done in each DPWM. So, for instance, if DPWM2 had a period twice that of DPWM1, and was being driven by DPWM1, setting the CLA_SCALE value in DPWM2 to multiply by 2 would result in the same PWM percentage in both DPWM1 and DPWM2.

3.4 Dynamic changes to the compensator configuration/values

It is possible to change the compensator values while it is running, but the following sequence is recommended to avoid asynchronous events causing issues with the output.

The asynchronous issues occur when the compensator is actually running a calculation when the change is attempted. It is very difficult to synchronize the firmware with the calculation time, so a different approach is used.

The firmware loads the sample trigger with a very large value. This value is well outside the period, so the sample trigger will never be reached, so the compensator will not ever be triggered for a calculation.

The sample trigger load will take effect in the next DPWM period, so after the start of the next period, it is safe to change the compensator configuration.

Then the sample trigger is restored to its normal value, and CLA operation is resumed.

The following sequence is designed for changing the Yn values while the power supply is running. This is done, for example, to decrease or increase the output quickly after a transient. In addition to the sequence above, 2 more DPWM periods are used to avoid spikes on the output of the comparator.

This code is used for cases like OVP or OCP, so the DPWM output is first turned off, and then the operations are performed.

Pseudo code for CLA dynamic clamping:

Step 1 (comparator interrupt triggers this)

```
{
    turn off DPWM outputs
    sample trigger = too big to trigger
    clear DPWM period interrupt
    enable DPWM period interrupt
    state = 0
}
DPWM period interrupt state machine
{
    switch ( state)
    {
        case 0 //put in clamp values, start CPU sample mode
            disable CLA
```

```

        YN1 = desired value
        YN2 = desired value
        cpu sample = 1
        enable cla
        Xn = 0
        state = 1
        sample trigger = normal value
        break;
    case 1 // just let clamp values percolate through CLA
        state = 2
        break;
    case 2 //back to running on next cycle
        if current/voltage back down below limits
        {
            turn DPWM outputs back on
            cpu sample = 0
            disable DPWM period interrupt
        }
    }
}

```

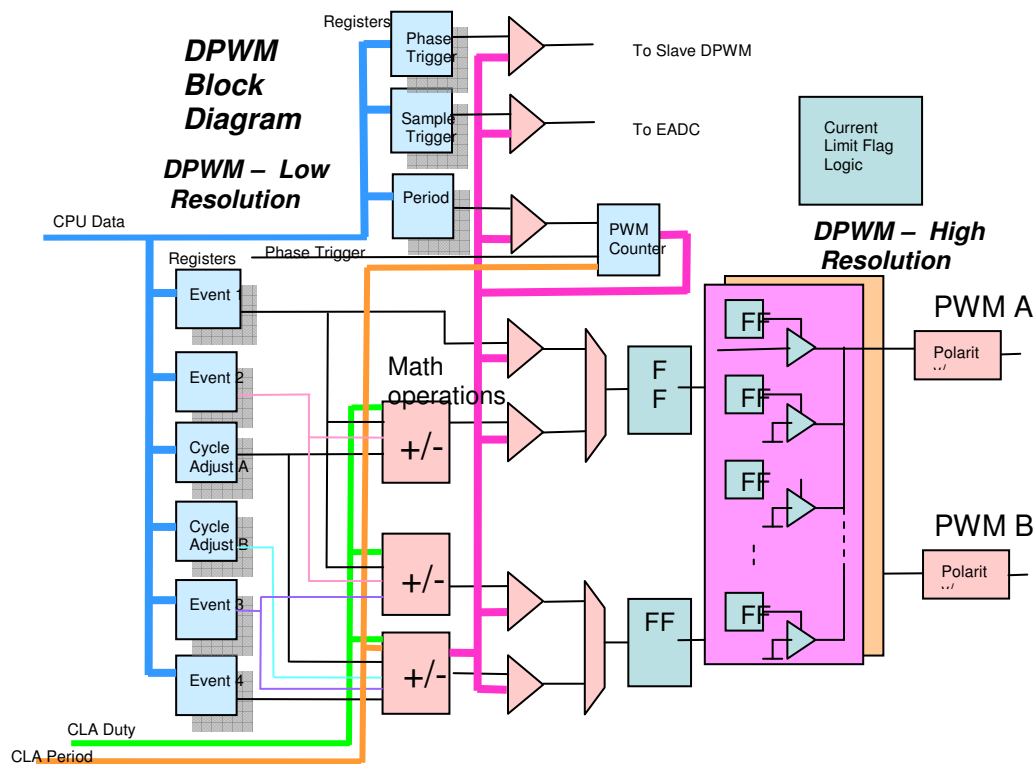
Note: - this is the high speed version for dealing with OVP and OCP.

Where more time is available, as in prebias startup, there is no problem with having more than 1 DPWM period between states. It is permissible to use the slower timer interrupt instead of the DPWM period interrupt if speed is not required.

4 Digital Pulse Width Modulator (DPWM)

The DPWM logic is probably the most complex of the Digital Fusion Peripherals. It takes the output of the Compensator and converts into the correct PWM output for several power supply topologies. It provides for programmable dead times, and for cycle adjustments for current balancing between phases. It controls the triggering of the EADC. It can synchronize to other DPWMs or to external sources. It can provide synchronization information to other DPWMs or to external recipients. In addition, it interfaces to several fault handling circuits. Some of the control for these fault handling circuits is in the DPWM registers.

Here is a block diagram of much of the DPWM logic:



4.1 Time resolution of various registers

Different registers in the DPWM block have different time resolutions. Pulse widths are generally adjustable in nominal 250 picosecond steps, while period and phase shift are adjustable in 4 nanosecond steps. The sample trigger is adjustable in 16 nanosecond steps.

Register	Resolution	Number of bits
Phase Trigger	4 ns.	14
Sample Trigger	16 ns.	12
Period	4 ns.	14
Event1	4 ns.	14
Event2	250 ps.	18
Event3	250 ps.	18
Event4	250 ps.	18
Cycle Adjust A	250 ps.	18
Cycle Adjust B	250 ps.	18
(Phase Shift)	4 ns.	14

The Phase Shift register mentioned in the table above is a hidden register used in Phase Shifted mode. See 4.8 DPWM Phase Shift Mode (PHASE_MODE) below for more information.

4.2 PWM counter and clocks

The PWM counter is the center of the DPWM logic. There is no register that can be read to give the value of the PWM counter, but most events are triggered by it. In all modes except for resonant mode, it is allowed to count up to the period value, and then restarted.

The PWM counter is also restarted by the receipt of a sync event. If the sync occurs before the end of a period, the effect is to shorten the period. If the sync occurs at the beginning of a period, the effect is to stretch out the beginning of the period somewhat.

Even though the period register has only 14 bits, the PWM counter effectively has 18 bits. The extra 4 bits are called “high resolution bits”.

The nominal clock frequency for the period register is 250 MHz. The nominal clock frequency for the high resolution bits is 4 GHz., giving a resolution of 250 picoseconds to the DPWM control circuitry.

4.3 Comparisons to PWM counter

Many events are triggered by comparisons to the PWM counter value. The rising and falling edges on the DPWM are the most obvious ones. Other events include the phase trigger and the sample trigger.

All of these comparisons are caused by the PWM counter value becoming exactly equal to the target value for the event. This simple comparison strategy causes some interesting effects in certain dynamic situations, which are described where appropriate.

4.4 DPWM modes

There are several DPWM modes. Most of them correspond roughly to one or more power supply topologies.

They are:

Normal Mode – open loop
Normal Mode – closed loop
Multi-mode
Phase shift mode
Resonant mode.

4.5 Normal Mode – open loop

Normal mode – open loop is the simplest mode.

To start Normal mode, make sure that none of the special mode bits in DPWMCTRL1 are set. Also make sure that the Compensator is not enabled as a source of data for the DPWM.

Then load the period register with the desired period. For example, for a 100 KHz. PWM rate, the value would be 250,000 KHz./100 KHz., or 2500. To get a perfectly correct value, 1 should really be subtracted before putting the value into the register, so it should be 2499. But for simplicity, and because the error is so small, this is left out of the examples.

For convenience, define a constant of PERIOD.

```
#define PERIOD (2500)
```

```
Dpwm1Regs.DPWMPRD.all = PERIOD;
```

Or this is also acceptable:

```
Dpwm1Regs.DPWMPRD.bit.PRD = PERIOD;
```

Next, initialize EVENT1, which will set the time at which the DPWM1A signal will rise:

```
Dpwm1Regs.DPWMEV1.all = 0; // A starts at beginning
```

This will cause DPWM1A to rise at the very beginning of the period.

DPWMEV1 is, like the DPWMPRD register, only 14 bits, so it has no high resolution capability.

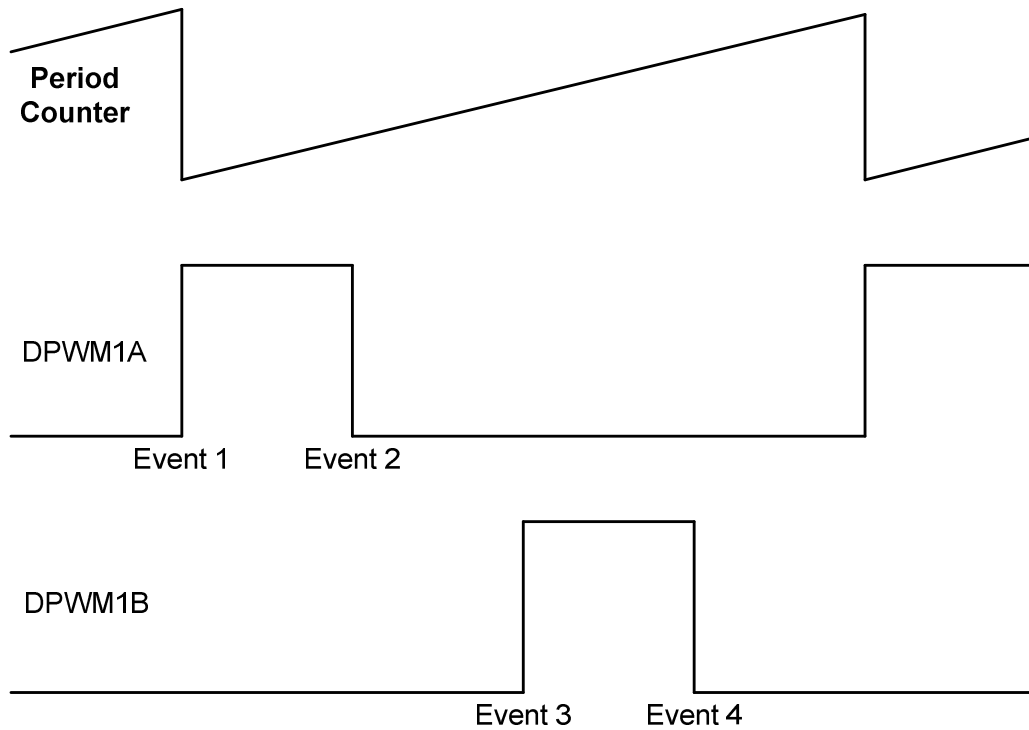
To set the fall time for DPWM1A, initialize Event 2:

```
Dpwm1Regs.DPWMEV2.all = PERIOD*4; // A stop at 25%
```

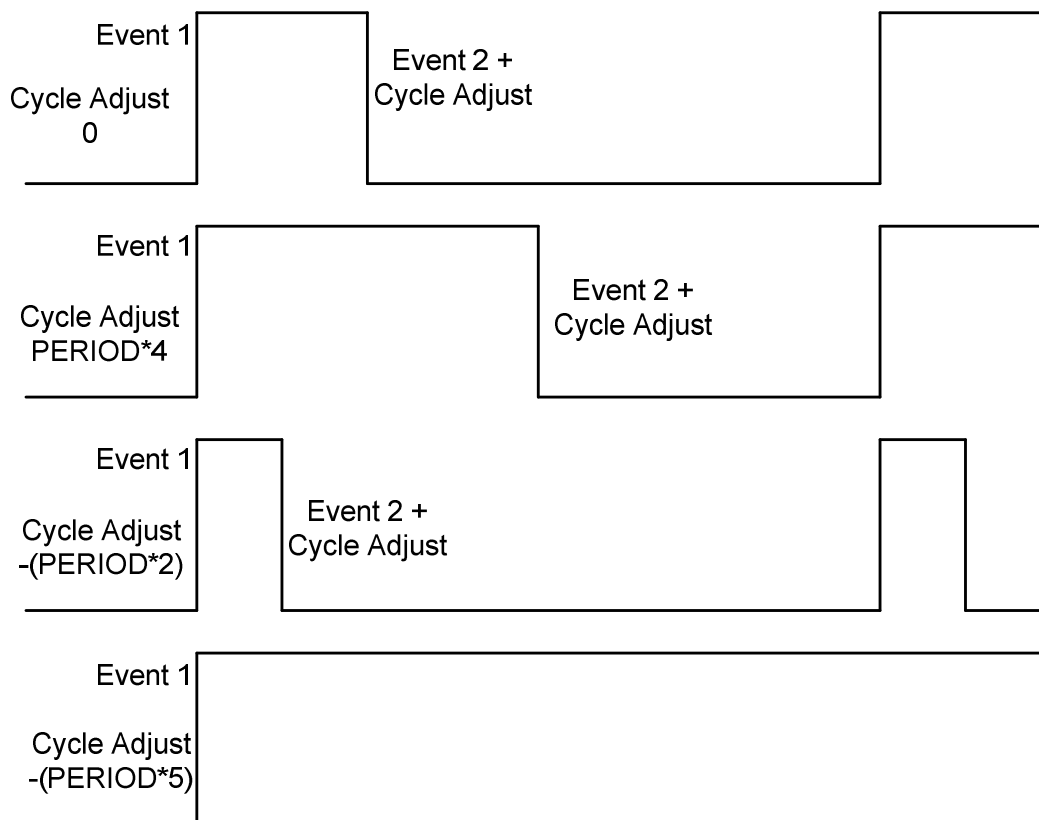
This C statement will, as it says, turn off the DPWM1A pin at 25%. This happens because Event 2 has the extra 4 bits of resolution for a total of 18. So 100 percent would be $\text{PERIOD} * 16$. Dividing by 4 (or multiplying by .25) gives $\text{PERIOD} * 4$. The rise and fall times for DPWM1B are set by Events 3 and 4 respectively:

```
Dpwm1Regs.DPWMEV3.all = PERIOD*8; // B start at 50%
Dpwm1Regs.DPWMEV4.all = PERIOD*12; // B stop at 75%
```

This will give a waveform something like this:



The cycle adjust A registers will affect the pulse width of the A channel. For instance, with the above event register values, here are some variations on cycle adjust:



With Cycle adjust:

```
Dpwm1Regs.DPWMCYCADJA.all = -(PERIOD*5);
```

It is possible to change the pulse width in almost all DPWM modes. The diagram above shows the effect. Note the bottom waveform, where the sum of cycle adjust and Event 2 is negative. This is to be avoided, as it will cause the DPWM pin to turn on all the time.

There is no dedicated dead band handling in this mode, as there is no data coming from the compensator. So in this mode, dead band must be provided by the event register settings.

There is no benefit to using cycle adjust in this mode, it is just described for purposes of introduction.

Cycle adjust B has no effect in Normal mode.

4.6 Normal Mode – Closed Loop

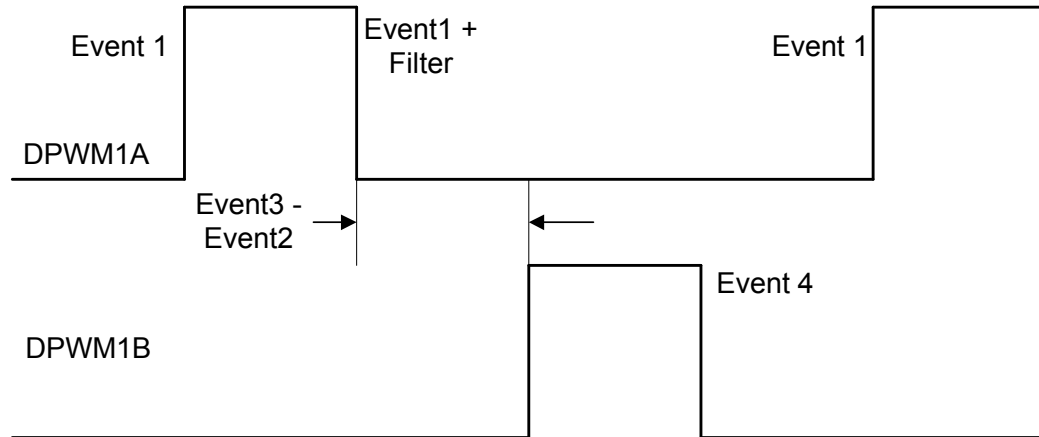
Normal Mode Closed Loop is similar to Open Loop, except the Compensator (CLA) input is enabled:

```
Dpwm1Regs.DPWMCTRL1.bit.CLA_ENABLE = 1; //use filter now
```

This causes the pulse width on DPWMA to be set by the Compensator output, rather than by Events 1 and 2. Event 1 is still used as the starting point, but Events 2, 3, and 4 are now used for dead time calculation.

Normal Mode with closed loop is intended to be used where both DPWMA and DPWMB are being used for the same phase, but only side is on at a time, for instance in a buck converter with a synchronous rectifier.

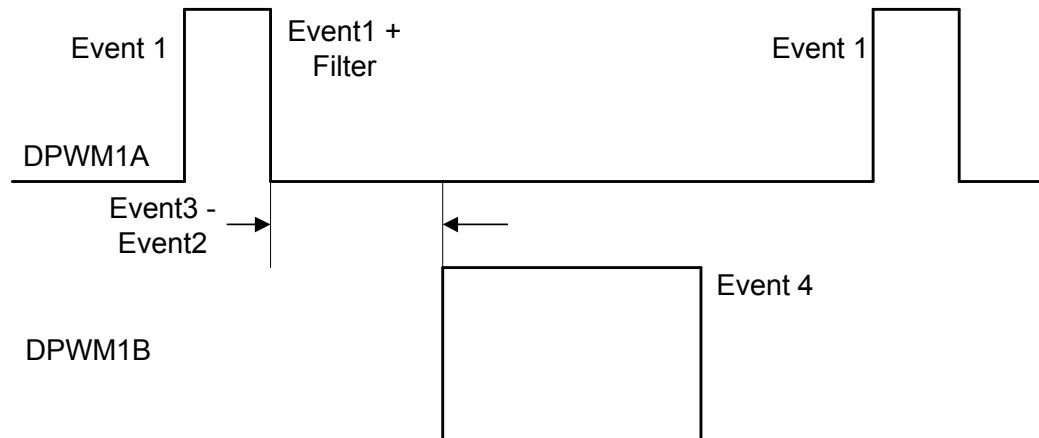
If the Compensator output is 0x2000, which is equivalent to 25% of full scale, then the output signals will look the same as the open loop example:



This is because the Compensator is at the same 25% as Event 2 was. The delay between the falling edge on DPWMA and the rising edge on DPWMB is set by the difference between Event 3 and Event 2. This is best illustrated by showing other Compensator output values, and their effect on the waveform.

Note that in this discussion, the dead times are much larger than typical actual dead times. This is done to make them more visible. All dead time calculations have at least 1 high resolution edge, so they can be defined with a resolution of 250 picoseconds.

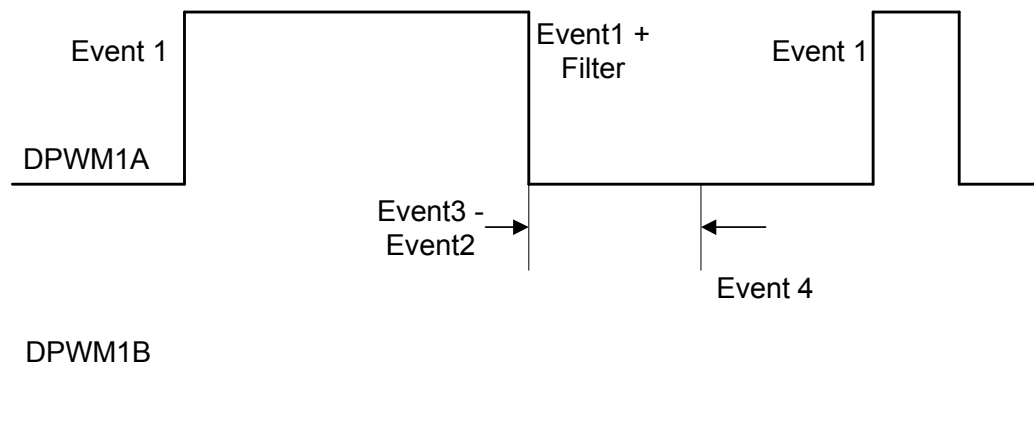
For example, a Compensator value of 0x1000, or 1/8 of full scale, will produce this waveform:



As the compensator output value continues to shrink, so will the pulse width on DPWMA, and the pulse width on DPWMB will continue to grow. When the compensator output value reaches zero, however, both DPWMA and DPWMB will turn off completely. If there is a requirement for transitions on DPWM pins under all conditions, the LCLAMP value on the compensator must be set to greater than zero. If the Compensator output is zero or negative, even adding a positive cycle adjust will not turn on the DPWM pins.

As the Compensator output increases, DPWMA positive pulse width increases, and DPWMB pulse width decreases. Finally, at 50% (in this case), DPWMB pulse width will be zero, so that pin will never go high.

This occurs when the end of the first dead time interval hits the start of the second dead time interval:



The first dead time interval is Event3 – Event2, as shown. The second dead time interval is represented by the time between Event 4 and Event 1. The end of the period is not shown in the figure above, but the second dead time interval can be represented as:

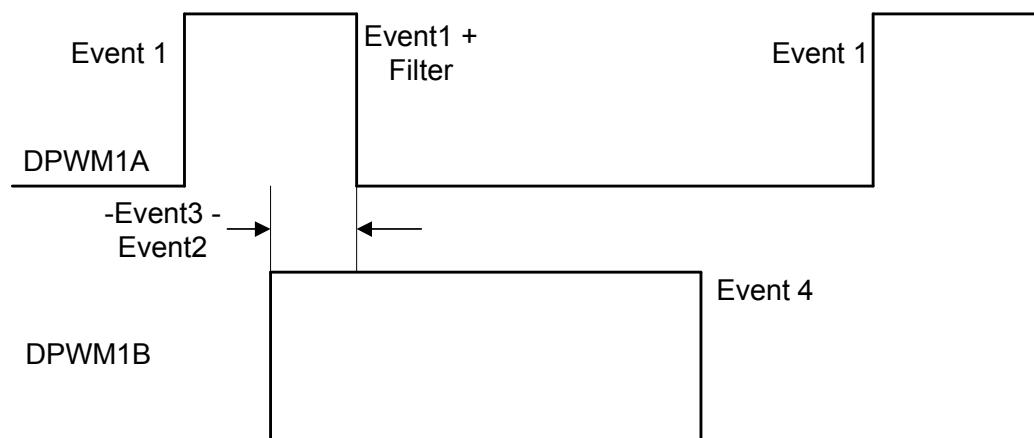
(Period – Event4) + Event 1.

As the Compensator output continues to increase, the pulse width can be extended all the way out until there is nothing but a tiny glitch low on DPWM1A, and no activity at all on DPWM1B.

It is also possible to have negative dead times, but only for events 2 and 3. For instance, if the Event3 setting is changed to:

```
Dpwm1Regs.DPWMEV3.all = PERIOD*2; // B start at 12.5%
```

When the Compensator output is 0x4000, the signals will look like this:



Negative dead times are not supported for Events 1 and 4.

4.6.1 Normal Mode Cycle Adjust

Cycle adjust A adjusts the length of the positive pulse on DPWM1A. It has no effect on DPWM1B. The dead times continue to be determined as if there was no cycle adjust. So to maintain the same dead time, it is necessary to adjust Event2 or Event 3.

When increasing cycle adjust, first increase the dead time, then increase the cycle adjust.

When decreasing the cycle adjust, first decrease cycle adjust, then decrease the dead time.

Cycle Adjust B has no effect in Normal mode.

As in open loop mode, avoid situations where the sum of the lowest possible Compensator output and the value of cycle adjust is negative. In this case, the Compensator output is clamped at a minimum of zero by logic between the Compensator and the DPWM, so it cannot be negative. But a negative cycle adjust with a zero compensator output will result in a negative value, and will cause DPWM1A to be on all the time.

Cycle adjust is primarily intended for current balancing between phases. It is best to use only positive Cycle Adjust for this function. If negative Cycle Adjust is used, it is best to set the LCLAMP value on the Compensator to a value large enough to make sure that the sum of the Compensator output and the cycle adjust is always positive.

To translate between the LCLAMP value and the Cycle adjust, see the PWM Scaling topic of the Compensator section earlier in this document. Cycle Adjust is covered in the Normal Mode Open Loop topic earlier in this section.

4.6.2 EADC sample timing in Normal Mode

The Sample Trigger register has already been mentioned in passing in the Error ADC section. Here is its C code:

```
Dpwm1Regs.DPWMSAMPTRIG.all = HIGH_SAMPLE_TRIGGER; //trigger late
//in period;
```

Note that the sample trigger is only a 12 bit value. Its clock frequency is only $\frac{1}{4}$ of the low resolution Period Counter resolution. The nominal Period Counter frequency is 250 MHz., so the sample trigger has only 62.5 MHz.

There are different strategies to be used for the sample trigger. In general, to avoid noise, it is wise to have the sample trigger as far from switching edges as possible.

If output voltage is being measured, that is generally the primary consideration, as there is little ripple. Sometimes when current is being measured, it is best to try to measure in the middle of the on time of the FETs, so as to get an average current value.

In either case, it is necessary to read the YN1 output of the Filter to determine the PWM pulse width, and then to multiply this by the correct value to get the correct sample trigger value. This should be done frequently enough so that the PWM value will not change too much in the interval. The exact details are left to an application note or to example code.

Since the comparisons for events are done on a basis of simple equality, rather than greater than or less than, it is necessary to keep the end of Compensator calculations away from switching times, as a change of the event value at the wrong time could lead to an event being missed, and a DPWM pin not being turned on or off when it should.

There is a bit which provides some protection from this:

```
Dpwm1Regs.DPWMCTRL1.bit.UPDATE_END_PRD_ENA = 1;
```

Setting this bit causes the transfer of data from the Compensator to the DPWM to occur only at the end of the period, which should prevent any problem from occurring with compares.

4.7 DPWM Multiple Output mode (MULTI_OUT)

Multiple output is intended primarily for use with the Compensator. It is used for systems where each phase has only one driver signal. It enables each DPWM peripheral to drive two phases with the same pulse width, but with a time offset between the phases, and with different cycle adjusts for each phase.

If MULTI_OUT mode is used open loop, it is the same as normal mode, except that cycle adjust B affects the pulse width on DPWMB

Event 2 and Event 4 are not relevant in MULTI_OUT mode. The pulse on DPWMA is started by Event1 and the width is set by the sum of the Compensator output and cycle adjust A.

On DPWMB, the start is set by Event 3 and the width is the sum of the Compensator output and cycle adjust B.

Since the signals are intended to drive different phases, there is no dead time relationship between DPWMA and DPWMB.

DPWMB can cross over the period boundary safely, and still have the proper pulse width, so full 100% pulse width operation is possible. DPWMA cannot cross over the period boundary.

To start MULTI_OUT mode, set the MULTI_OUT bit:

```
Dpwm1Regs.DPWMCTRL1.bit.MULTI_OUT_MODE = 1 ;//make it multimode.
```

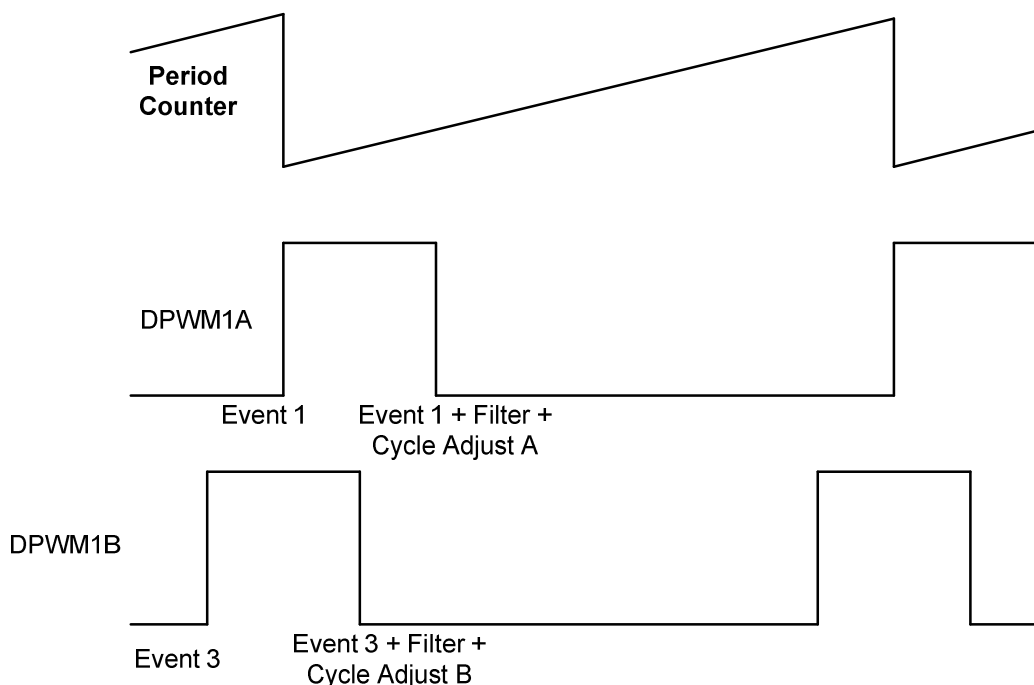
It is important to keep the SAMPTRIG point away from the switching point of the DPWM, as the comparison can be missed if the Compensator output updates close to the switching point. Setting the UPDATE_END_PRD_ENA bit can also be a problem if the switching edge of DPWMB is close to the end of the period.

A compensator output of zero is corrected to be non-negative, but if the sum of cycle adjust and compensator output is negative, it will result in the relevant DPWM channel turning on all the time. The best way to avoid this is to avoid negative cycle adjust values.

If the LCLAMP value of the compensator is allowed to be negative, do not set the UPDATE_END_PRD_ENA (see above for definition). Setting this bit may cause DPWMB to stay on for 100% during the transition between negative and positive Compensator output value.

Note that as mentioned above, if UPDATE_END_PRD_ENA is cleared, care must be taken to keep the end of CLA calculations away from the switching times for the PWM outputs.

Here is an example of Multi Mode:



Note that DPWM1B extends past the start of the next period.

4.8 DPWM Phase Shift Mode (PHASE_MODE)

Phase Shift Mode is intended for use in phase shifted full bridge topologies. To enable it, set this bit:

```
Dpwm1Regs.DPWMCTRL1.bit.PHASE_MODE = 1;
```

Using PHASE_MODE requires at least 2 DPWMs. The first DPWM must be the master, and the control Compensator should be associated with that DPWM. The master DPWM should be in PHASE_MODE. In PHASE MODE, the DPWMA and DPWMB pins function exactly as they do in Normal mode, in open loop. Instead of being used to control DPWM pins, the Compensator output is used to control the SYNC signal from the master DPWM. If the Compensator output is 0, the SYNC signal from the Master DPWM has a delay of 0. If the Compensator output is full range, the SYNC signal is equal to the entire period.

The calculated phase shift is low resolution, meaning that the phase shift will adjust in 4 nanosecond steps.

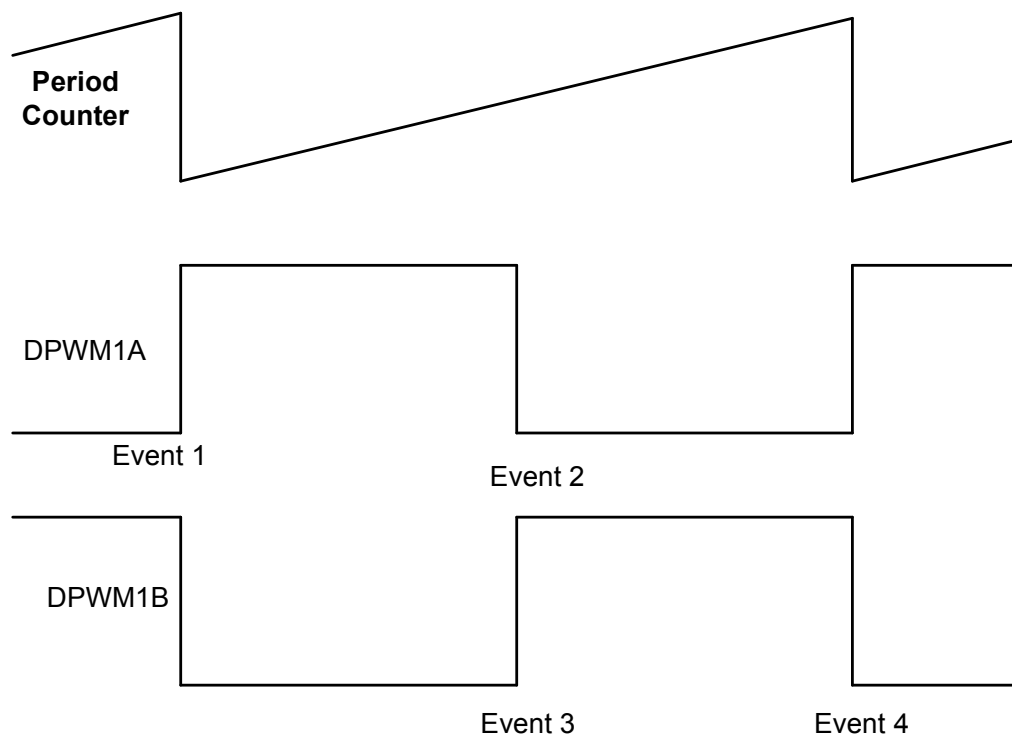
The Sync signal resets the period counter on the slave DPWM. This fact combined with the compare logic of the DPWM requires a special configuration of the DPWM for PHASE_MODE.

The slave DPWM is configured in Normal Mode open loop. Its position relative to the master DPWM is changed by the changing SYNC timing. This changing SYNC timing is what drives the special configuration.

Here is a normal configuration of the slave (and for that matter the master) DPWM for 50% duty cycle:

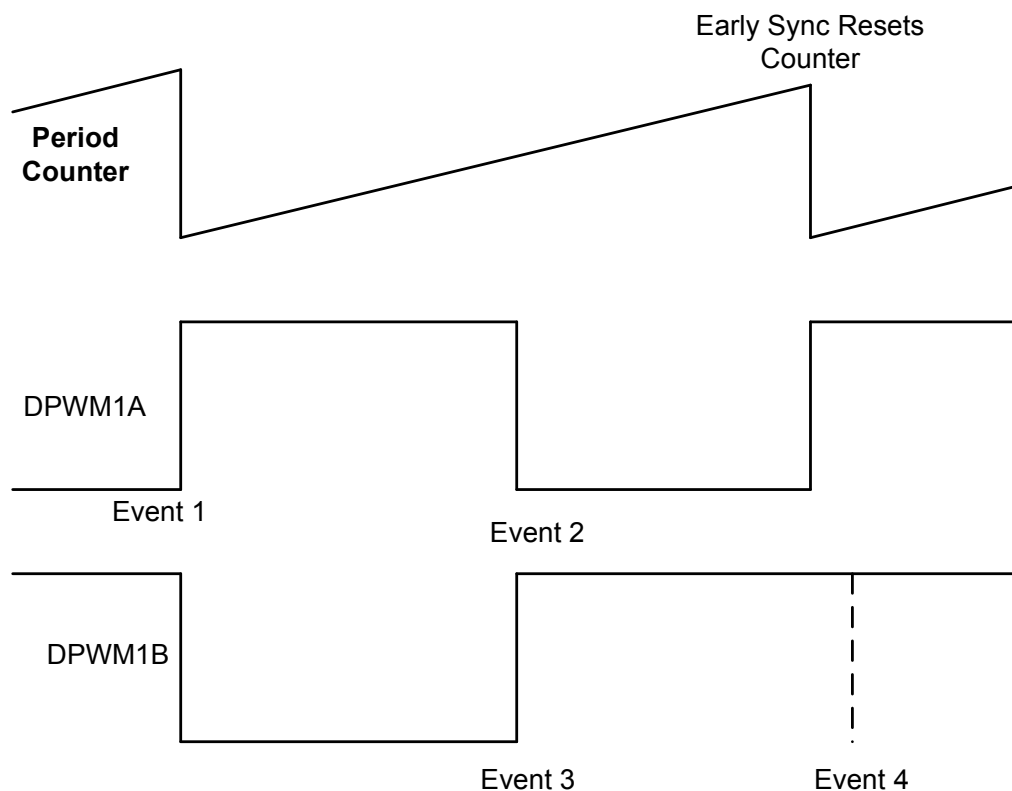
```
Event 1 = 0
Event 2 = Period * 8 50% with high resolution (4 added bits)
Event 3 = Period * 8
Event 4 = Period * 16 100% with high resolution
```

The output looks like this:



For DPWMB to shut off the period counter must count up to the full length of the period so that it is equal to Event 4. But if the Compensator output decreases during the period, which it may well do, the SYNC pulse will come before the full length of the period. This will reset the period counter before the end of the period. In this case, DPWMB will not be turned off, which will cause two transistors of a complimentary pair to be turned on, thus letting the smoke out.

This diagram illustrates this event. Note that the early SYNC pulse resets the counter before it reaches Event 4, causing DPWMB to remain on.



To avoid this, it is necessary to make use of another feature of the DPWM that has not been mentioned yet. This is the capability of inverting outputs. In this case, DPWMB inversion is appropriate. Both output on times are moved to the center of the period, away from the period reset, and then DPWMB is inverted.

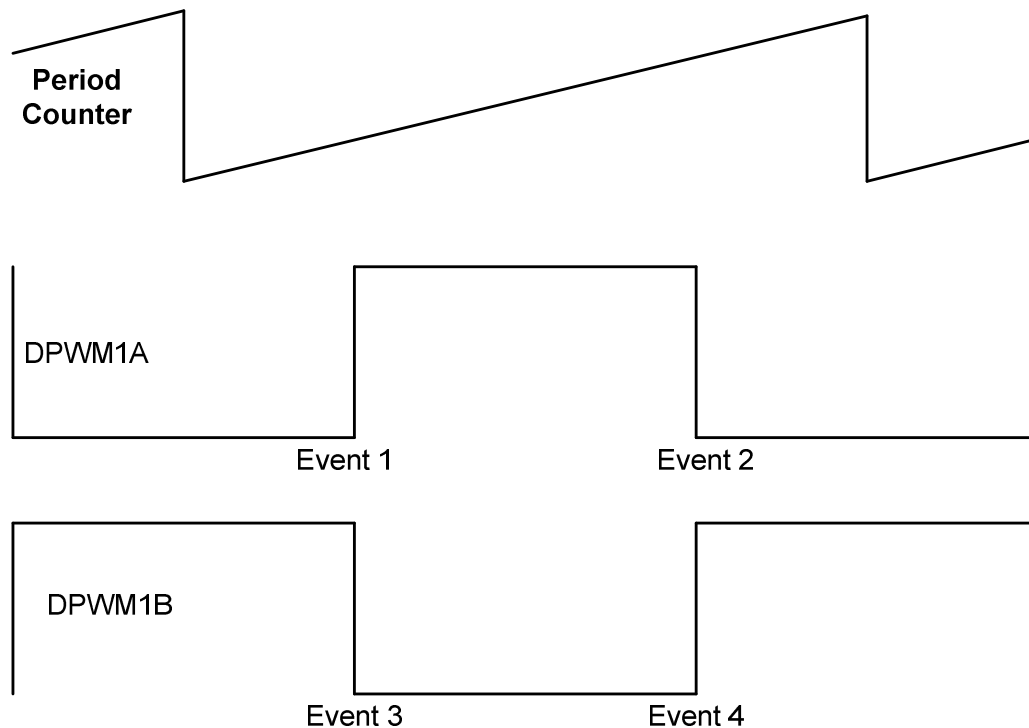
The DPWMB invert bit is set in this way:

```
Dpwm1Regs.DPWMCTRL2.bit.PWM_B_INV = 1;
```

For 50% duty cycle, the Event registers must be set this way:

Event 1 = Period/4 – Low resolution, 25%
 Event 2 = Period * 12 – High resolution, 75%
 Event 3 = Period * 4 – High resolution 25%
 Event 4 = Period * 12 – High resolution 75%

This gives a waveform that looks like this:



Since the period counter is invisible, this waveform looks identical to the previous waveform. But all events are 25% away from the edges of the period. So the compensator output would have to change by 25% in a single cycle to cause problems. This will not happen with a reasonable Compensator response. Changes in Compensator output will be reflected as changes in the width of DPWMB on time, and DPWMA off time. But the integrated offset of these changes cannot be more than one normal pulse width.

The event settings can, of course, be changed for dead time adjustment, and for ZVT, but that is not a topic for this document. Contact Texas Instruments for application notes and sample code. The basic approach outlined here can be used for all phase shifted applications. It is likely that the final code will be somewhat more complex, of course.

4.9 DPWM Resonant Mode

Resonant mode is designed to control resonant power supplies. Basic operation is very simple.

Load the period register with a value which provides the lowest frequency desired. Load the event registers with values that establish the relative width and dead times for the control signals.

The filter output will then adjust the period, and the relative width and dead times of the signals will be preserved. The period will be proportional to the output of the filter.

For example, the maximum filter output of 0x7fff will deliver almost the same period as put into the period register. 0x4000 out of the filter will provide half as much (double the period) and so on.

It is very important to set the LCLAMP value in the Compensator to a value which will prevent an excessively high frequency. For example, if the lowest frequency desired is 100 KHz., and the highest frequency is 500 KHz., set up the Period register for 100 KHz., and set the LCLAMP for 0x8000 (32768) divided by 5 = 0x1999 (6553).

The filter starts running with an output of zero, which will cause the system to lock up. Even if the clamp values are set to numbers above zero, one cycle through the filter is necessary for the value to take effect.

For this reason, it is necessary to run the filter before starting resonant mode.

The code below shows an example of how this is done.

4.10 Resonant mode Code Example

First, initialize the filter as described above in the filter section. This initialization needs to include enabling the filter to run, by setting the CLA_EN bit in FLTRCTRL:

```
Filter1Regs.FLTRCTRL.bit.CLA_EN = 1 ; //enable filter
```

It is very important to also have the LCLAMP initialized at this point to force the filter away from a 0 output.

Then initialize the Event Registers:

```
Dpwm1Regs.DPWMCTRL1.all = 0; //no CLA, leave PWM off for the moment
```

```
Dpwm1Regs.DPWMPRD.all = 2500 ; //out to max period length
Dpwm1Regs.DPWMEV1.all = 0; // A start at beginning
Dpwm1Regs.DPWMEV2.all = 2500 * 4;
Dpwm1Regs.DPWMEV3.all = 2500 * 8; // B start at halfway point - *8 is because it is high res
Dpwm1Regs.DPWMEV4.all = 2500 * 12;
```

And initialize for resonant mode:

```
Dpwm1Regs.DPWMCTRL1.bit.RESONANCE_MODE = 1;
```

Even though the filter is initialized for resonant mode, this does not go into effect until the CLA_ENABLE bit is also set. The filter will run under normal mode, controlled by the event registers until that bit is also set.

So that the filter will run very quickly, initialize the sample trigger to zero:

```
Dpwm1Regs.DPWMSAMPTRIG.all = 0;
```

Next, to make sure that the PWM does not actually put out a signal while the filter is being initialized, enable the PWM pins as GPIO instead.

```
Dpwm1Regs.DPWMCTRL1.bit.GPIO_A_ENA = 1;
Dpwm1Regs.DPWMCTRL1.bit.GPIO_B_ENA = 1;
```

And now enable the PWM. This will trigger the filter to run, even though the PWM is ignoring the filter output

```
Dpwm1Regs.DPWMCTRL1.bit.PWM_ENA = 1;
```

Next, a short delay to allow the filter to run. The NOP is put in to discourage the optimizer from removing the delay loop

```
for(i=1;i<20;i++)  
{  
    asm(" NOP");  
}
```

Then move the sample trigger away from the edge for normal operation with less noise

```
Dpwm1Regs.DPWMSAMPTRIG.all = 20;
```

Finally, enable the DPWM to take its direction from the now non-zeroed filter

```
Dpwm1Regs.DPWMCTRL1.bit.CLA_ENABLE = 1; //use cla now
```

And then permit the DPWM to find its way out to the pins by undoing the GPIO mode.

```
Dpwm1Regs.DPWMCTRL1.bit.GPIO_A_ENA = 0;  
Dpwm1Regs.DPWMCTRL1.bit.GPIO_B_ENA = 0;
```

4.11 Synchronizing Multiple DPWMs.

The special case of synchronizing 2 DPWMs for a phase shifted full bridge has already been covered in that section. This section is more general, primarily for use with multiple phases in either normal or multi-mode. This section also covers using a single Compensator to drive multiple DPWMs.

Synchronization timing is dictated by the DPWMPHASETRIG register. Here is the C code:

```
Dpwm1Regs.DPWMPHASETRIG.bit.PHASE_TRIGGER = PERIOD/2;
```

This example, PERIOD/2 is typical for a system where 2 phases are desired 180 degrees out of phase. Phase Trigger is a low resolution register, with the same resolution as Event1 and Period – a nominal 4 ns. per count.

Note that all sync pulses out of a given DPWM go through the phase trigger delay.

Any DPWM can be synchronized to any other DPWM.

This function works with two bit fields in the DPWMCTRL1 Register.

They are

```
Dpwm2Regs.DPWMCTRL1.bit.MSYNC_SLAVE_ENA = 1 //enable slave sync.  
Dpwm2Regs.DPWMCTRL1.bit.MSYNC_CH_SEL = 0; //slave it to dpwm1
```

The MSYNC_SLAVE_ENA register obviously enables the synchronization, while the MSYNC_CH_SEL register selects the DPWM channel to synchronize to.

The phase trigger can be changed during operation. When changing the phase trigger, care must be taken to avoid resetting the DPWM before all events are complete, or the event will be missed. This is described in more detail in the Phase Shift section above.

4.12 Other DPWM Bit Fields

4.12.1 CLA_CH_SEL

Dpwm1Regs.DPWMCTRL1.bit.CLA_CH_SEL is used to select which CLA is used as an input for the DPWM. It only effects which CLA is used for input. Sample Trigger is always used to trigger the EADC and the CLA for the DPWM with the same number.

4.12.2 FAULT_ENA

This bit enables the turn off of the DPWM channel by the fault pins. See the UCD30xx Faults and External Interrupts (GIO) Programmer's Manual for more information.

4.12.3 SYNC_SLAVE_ENA and Sync In pin

This bit enables the DPWM to be synchronized to an external SYNC pin input. To use the Sync in pin as a general purpose I/O pin, use the SYNCCTRL register in the Miscellaneous Analog Control address range. All control bits for GPIO are in this register. To use the Sync in pin for Sync, leave all these bits in their default (power up) state.

4.12.4 SYNC_OUT_DIV_SEL and SYNC_OUT pin

These 4 bits set the divider for the sync out pin. The value put into the bits = Divider – 1.

So:

Divider	Value
1	0000
2	0001
....
16	1111

There is no way to divide down the sync pulses going between the DPWMs internal to the device. They are always at full speed. So if different DPWMs are to run at multiples of each other's speeds, use the slowest one at the head of the sync chain. Otherwise the slower DPWMs will be reset before they complete their cycles.

Check the datasheet, as the Sync out pulse may be very short.

To select which DPWM is used for the output, put a 0 to 3 into this bit field in the same register:

```
MacRegs.SYNCCTRL.bit.SYNC_MUX_SEL = 0;
```

The Sync out pin is also used for general purpose I/O and as a clock output pin. So its control is distributed between several registers.

Here is a partial truth table, covering its use as a GPIO and as a sync pin. For its use as a clock out pin, see the System Module documentation.

SYNC_OUTPUT_DIR	CLKDIR	Pin function
0	0	Sync_Out
0	1	Output (CLKDOUT)
1	0	Input (SYNC_OUT_IN)
1	1	Input (SYNC_OUT_IN)

The C code for the two bits above is as follows:

```
MacRegs.SYNCCTRL.bit.SYNC_OUTPUT_DIR
SysRegs.CLKCNTL.bit.CLKDIR
```

As an input, the input level can be read from:

```
MacRegs.SYNCCTRL.bit.SYNC_OUT_IN
```

To set the value when it is an output, use:

```
SysRegs.CLKCNTL.bit.CLKDOUT = 0;
```

4.12.5 GPIO with DPWM pins

There are several fields in DPWMCTRL1 that affect the use of the DPWM output pins as general purpose digital I/O. They are:

PWM_A_OE, PWM_B_OE – 0 = output, 1 = input

GPIO_A_VAL, GPIO_B_VAL, - Value to output in GPIO output mode

GPIO_A_ENA, GPIO_B_ENA – 1 = GPIO, 0 = DPWM controlled

Note that two bits in DPWMCTRL2 also effect the output pins in GPIO mode.

PWM_A_INV, PWM_B_INV – a 1 causes output inversion in both DPWM and GPIO modes.

It is not necessary to enable the PWM (with the PWM_ENA bit in DPWMCTRL1) to use the DPWM pins in GPIO mode.

To read the values returned when the DPWM pins are in GPIO mode, look in DPWMOVERFLOW. There are two bits there, GPIO_A_IN and GPIO_B_IN, which will reflect the value on the pin in input mode.

Note that the CLF circuitry overrides the GPIO settings. This means that even if GPIO_B_ENA is a 1, if the CLF logic turns off DPWMA, DPWMB will also be turned off.

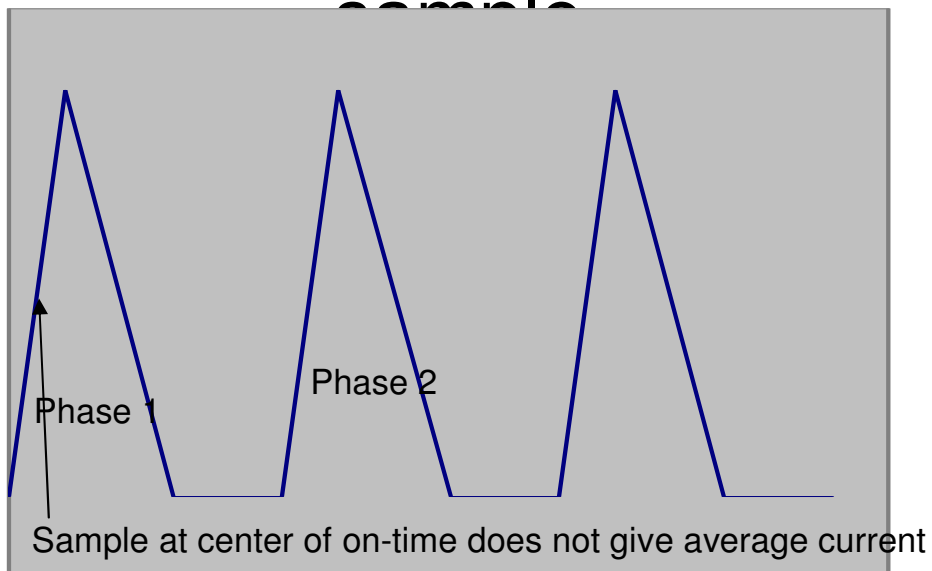
4.12.6 Oversample

The Oversample bits allow the EADC and Compensator to run 1, 2, 4, or 8 times per PWM period. Oversampling is very useful in discontinuous mode. Taking up to 8 samples per switching period gives a much better average than only taking one sample.

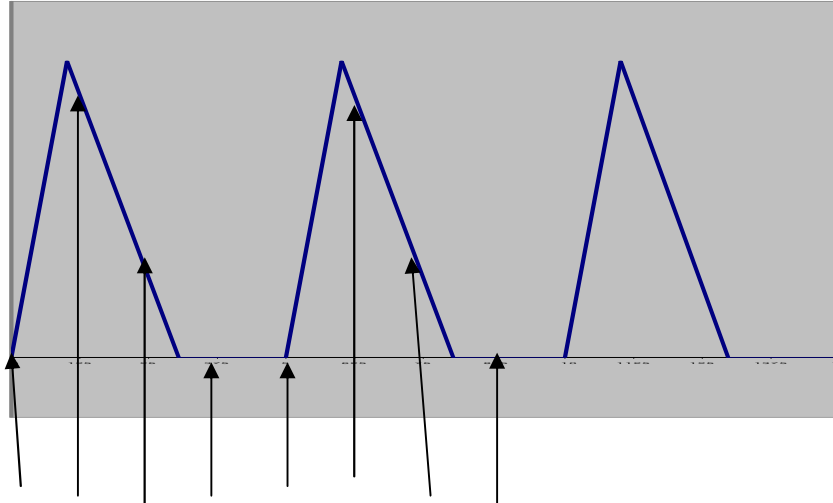
In continuous mode, it is not as essential, but it can still be used.

Oversample does not change the pulse width of the output during the switching cycle, so it does not have a beneficial effect on the phase delay or response time of the filter.

Discontinuous mode, one sample



Discontinuous mode, 8



8 samples measure average better in discontinuous mode,
also average out noise

4.12.7 SFRAME Trigger and SFRAME_ENA

Putting a 1 in SFRAME_ENA enables SFRAME_TRIGGER to become the EADC trigger source. This is useful when running the compensator under controlled conditions to watch its performance with software and/or with a JTAG debugger.

Note that there is no register which indicates that the compensator has run. The Period Interrupt flag is actually asserted at the beginning of the period, not at the end.

The end of the compensator running is best calculated using a timer or a firmware delay. The compensator takes 7 clock cycles to run in 2P2Z mode, and 9 cycles to run in 3P3Z mode. There is also an X clock cycle delay for the EADC. Even in CPU_SAMPLE mode, the EADC delay occurs.

4.12.8 CHECK_OVERRIDE

The Check Override disables some protection functions in the DPWM logic

4.12.9 PWM_A_PROT_DIS and PWM_A_PROT_DIS

These bits in the DPWMCTRL2 register disable the asynchronous protection logic when they are set to a 1.

4.12.10 HIRES_SCALE, ALL_PHASE_CLOCK_ENABLE, HIRES_DIS

These registers, also in the DPWMCTRL2 register control the number of phases in the high resolution clock that are used. They are best left in their default states.

4.13 Current Limit Flag (CLF) support

The CLF hardware is designed to handle current limit events, but in fact can also be used for any input to the comparators that requires a shutdown of the DPWM when a threshold is exceeded, either once or several times.

The comparator output is sampled at the DPWM's Event 2 time. In many DPWM modes, this corresponds to the end of the on pulse, so it would correspond to maximum current.

The input to the CLF logic is the analog comparators built into the device. Each DPWM can be configured to be turned off by a specific comparator. This is done with the CLFCTRL register in the Miscellaneous Analog Control address space.

To configure Comparator D to act as the input to the CLF logic DPWM 2, this is the C code:

```
MacRegs.CLFCTRL.bit.ANALOG_L2_CLF_CTRL = 3;
```

A single comparator can be programmed to drive multiple CLF inputs.

The Comparator threshold is programmable in 64 steps, see the description in the register descriptions below (7.1 Comparator Control Register (COMPCTRL)). This C code sets the threshold for Comparator A to 2 volts:

```
MacRegs.COMPCTRL.bit.COMP_ADJ_A = 0x3f;
```

See (Section 7.1 Comparator Control Register (COMPCTRL));

The comparator output can also be read from the COMPREAD register (Section 7.3 Comparator Read Register (COMPREAD)).

The CLF logic provides some noise immunity and delayed response by making it possible to wait for several consecutive comparator events before shutting down the DPWM channel.

This is made possible by the COUNT_MAX bits in the DPWMCLFCTRL register. This C code sets the maximum count to 12:

```
Dpwm3Regs.DPWMCLFCTRL.bit.COUNT_MAX = 12;
```

Normally the CLF is configured for consecutive limit triggers. In this case, an Event2 where the comparator is not triggered will cause the count to be cleared.

To count limit events that are not consecutive, set the COUNT_CONT bit:

```
Dpwm3Regs.DPWMCLFCTRL.bit.COUNT_CONT = 1;
```

The count of events can be read from the COUNT field of the same register. There is a bit to clear the count, and a bit to enable the CLF logic.

The logic will only disable the DPWM with which it is associated. It is possible to poll, or to generate an interrupt, and use a single CLF module to trigger the shutdown of an entire system. This is obviously a bit slower than the direct hardware shutdown provided by the CLF logic.

There is also a bit which can clear the counter – COUNT_CLR.

4.14 DPWM Overflow

Waiting on Design input

4.15 DPWM Interrupt Register (DPWMINT)

The DPWM can generate interrupts from two sources – the CLF logic and the DPWM Period.

The CLF interrupt will occur when the count register exceeds the value in the COUNT_MAX field. This will cause the CLF bit in the DPWMINT register to be set. If the CLF_INT_ENA bit is also set, this bit will be passed to the CIM module to interrupt the processor if enabled. To clear this bit, it is necessary to clear the CLF counter, as described above.

There are also a PRD bit and a PRD_INT_ENA bit, which work in a similar way. The PRD bit is actually set at the beginning of a DPWM period. The PRD_INT_SCALE bits select the number of periods between interrupts. There are 4 bits, and the number of periods ranges from 1 to 256. See Section 6.14 DPWM Interrupt Register (DPWMINT) for more information

The PRD bit is a clear on read bit. It must be cleared by the interrupt routine, or it will remain set and cause another interrupt.

5 Loop 1-4 Compensator Registers

Registers for Compensator modules 1 – 4 are identical in their bit definitions. The addresses given, and the number of Compensator modules are from the UCD3040. For other devices, consult their respective data sheets.

5.1 Filter Status Register (FLTRST)

Address FFF7E000 – Loop 4 Filter Status Register

Address FFF7E400 – Loop 3 Filter Status Register

Address FFF7E800 – Loop 2 Filter Status Register

Address FFF7EC00 – Loop 1 Filter Status Register

Bit Number	4	3	2
Bit Name	EADC_RAIL_HIGH	EADC_RAIL_LOW	COEF_PAGE
Access	R	R	R
Default	-	-	-

Bit Number	1	0
Bit Name	CLAMP_ACTIVE_PG	NL_ACTIVE_PG
Access	R	R
Default	-	-

Bit 4: EADC_RAIL_HIGH - EADC Input High Rail Indicator

0 = EADC Input not equal to high rail

1 = EADC Input equal to high rail

Bit 3: EADC_RAIL_LOW - EADC Input Low Rail Indicator

0 = EADC Input not equal to low rail

1 = EADC Input equal to low rail

Bit 2: COEF_PAGE - Current active coefficient page

0 = Page A

1 = Page B

Bit 1: CLAMP_ACTIVE_PG – Current active clamp page

0 = Page A

1 = Page B

Bit 0: NL_ACTIVE_PG – Current active non-linear table page

0 = Page A

1 = Page B

5.2 Filter Control Register (FLTRCTRL)

Address FFF7E008– Loop 4 Filter Control Register

Address FFF7E408 – Loop 3 Filter Control Register

Address FFF7E808 – Loop 2 Filter Control Register

Address FFF7EC08 – Loop 1 Filter Control Register

Bit Number	7	6	5	4
Bit Name	NL_PG_CONTROL	CLAMP_PG_CONTROL	USE_CPU_SAMPLE	RSVD
Access	R/W	R/W	R/W	R/W
Default	0	0	0	0

Bit Number	3	2	1	0
Bit Name	P3Z3	BNA_READ	BNA_ACTIVE	CLA_EN
Access	R/W	R/W	R/W	R/W
Default	0	0	0	0

Bit 7: NL_PG_CONTROL – Sets which bank of Gain Table and Error Limits are active

0 = Selects Bank A as active, Bank B is writeable (Default)

1 = Selects Bank B as active, Bank A is writeable

Bit 6: CLAMP_PG_CONTROL – Sets which bank of clamps are active

0 = Selects Bank A as active, Bank B is writeable (Default)

1 = Selects Bank B as active, Bank A is writeable

Bit 5: USE_CPU_SAMPLE – Forces CLA to use CPU programmed error samples

0 = CLA Mode, input data received from EADC (Default)

1 = CPU Mode, input data based on Filter X/Y Data Registers

Bit 4: RSVD – Reserved

Has no effect. Leave at 0.

Bit 3: P3Z3 – Filter Configuration

0 = 2-pole/2-zero filter application (Default)

1 = 3-pole/3-zero filter application

Bit 2: BNA_READ – Coefficient Bank Read Select

0 = Selects Bank A coefficients to read (Default)

1 = Selects Bank B coefficients to read

Bit 1: BNA_ACTIVE – Active Coefficient Bank Select

0 = Selects Bank A coefficients as active, coefficients writeable in Bank B (Default)

1 = Selects Bank B coefficients as active, coefficients writeable in Bank A

Bit 0: CLA_ENA – CLA Enable

0 = Disables CLA operation (Default)

1 = Enables CLA operation

5.3 Filter X Data Register 1 (FLTRXDATA1)

Address FFF7E00C – Loop 4 Filter X Data Register 1

Address FFF7E40C – Loop 3 Filter X Data Register 1

Address FFF7E80C – Loop 2 Filter X Data Register 1

Address FFF7EC0C – Loop 1 Filter X Data Register 1

Bit Number	23:16	15:8	7:0
Bit Name	XN	XN2	XN1
Access	R/W	R/W	R/W
Default	0000_0000	0000_0000	0000_0000

Bits 23-16: XN - X_n value, allows processor to initialize filter input value. These bits can only be programmed when the filter is turned off, by setting CLA_EN (Bit 0 of FLTRCTRL) low. Value is sign-extended to 8 bits. 5 LSB bits represent value.

Bits 15-8: XN2 - X_{n-2} value, allows processor to initialize filter previous input value. These bits can only be programmed when the filter is turned off, by setting CLA_EN (Bit 0 of FLTRCTRL) low. Value is sign-extended to 8 bits. 5 LSB bits represent value.

Bits 7-0: XN1 - X_{n-1} value, allows processor to initialize filter previous input value. These bits can only be programmed when the filter is turned off, by setting CLA_EN (Bit 0 of FLTRCTRL) low. Value is sign-extended to 8 bits. 5 LSB bits represent value.

5.4 Filter Y Data Register 1 (FLTRYDATA1)

Address FFF7E010 – Loop 4 Filter Y Data Register 1

Address FFF7E410 – Loop 3 Filter Y Data Register 1

Address FFF7E810 – Loop 2 Filter Y Data Register 1

Address FFF7EC10 – Loop 1 Filter Y Data Register 1

Bit Number	15:0
Bit Name	YN1
Access	R/W
Default	0001_1010_1010_1011

Bits 15-0: YN1 - Y_{n-1} value, allows processor to initialize previous filter output value. These bits can only be programmed when the filter is turned off, by setting CLA_EN (Bit 0 of FLTRCTRL) low.

5.5 Filter Y Data Register 2 (FLTRYDATA2)

Address FFF7E014 – Loop 4 Filter Y Data Register 2

Address FFF7E414 – Loop 3 Filter Y Data Register 2

Address FFF7E814 – Loop 2 Filter Y Data Register 2

Address FFF7EC14 – Loop 1 Filter Y Data Register 2

Bit Number	15:0
Bit Name	YN2
Access	R/W
Default	0001_1010_1010_1011

Bits 15-0: YN2 - Y_{n-2} value, allows processor to initialize previous filter output value. These bits can only be programmed when the filter is turned off, by setting CLA_EN (Bit 0 of FLTRCTRL) low.

5.6 Filter Y' Data Register 1 (FLTRYPDATA1)

Address FFF7E018 – Loop 4 Filter Y' Data Register 2

Address FFF7E418 – Loop 3 Filter Y' Data Register 2

Address FFF7E818 – Loop 2 Filter Y' Data Register 2

Address FFF7EC18 – Loop 1 Filter Y' Data Register 2

Bit Number	15:0
Bit Name	YPN1
Access	R/W
Default	0001_1010_1010_1011

Bits 15-0: YPN1 - Y'_{n-1} value, allows processor to initialize previous filter output value. These bits can only be programmed when the filter is turned off, by setting CLA_EN (Bit 0 of FLTRCTRL) low.

5.7 Filter Nonlinear Response Register 1 (FLTRNLR1)

Address FFF7E01C – Loop 4 Nonlinear Response Register 1

Address FFF7E41C – Loop 3 Nonlinear Response Register 1

Address FFF7E81C – Loop 2 Nonlinear Response Register 1

Address FFF7EC1C – Loop 1 Nonlinear Response Register 1

Bit Number	31:30	29:24	23:18
Bit Name	AFE_GAIN	NOM_GAIN_MULT	POS_MID_GAIN_MULT
Access	R/W	R/W	R/W
Default	11 – Bank A 01 – Bank B	00_0100 – Bank A 01_0000 – Bank B	00_0100 – Bank A 01_0000 – Bank B

Bit Number	17:12	11:6	5:0
Bit Name	POS_LRG_GAIN_MULT	NEG_MID_GAIN_MULT	NEG_LRG_GAIN_MULT
Access	R/W	R/W	R/W
Default	00_0100 – Bank A 01_0000 – Bank B	00_0100 – Bank A 01_0000 – Bank B	00_0100 – Bank A 01_0000 – Bank B

Bits 31-30: AFE_GAIN – AFE Gain Control

Read: FLTRCTRL, Bit 2 (BNA_READ) controls which gain control is read. If BNA_READ is 0, AFE Gain Control from Bank A is read. Otherwise, AFE Gain Control from Bank B is read.

Write: FLTRCTRL, Bit 7 (NL_PG_CONTROL) controls which gain control is programmed. If NL_PG_CONTROL is 0, AFE Gain Control from Bank B is programmed. Otherwise, AFE Gain Control from Bank A is read.

00 = AFE Gain of 1

01 = AFE Gain of 2 (Default for Bank B)

10 = AFE Gain of 4

11 = AFE Gain of 8 (Default for Bank A)

Bits 29-24: NOM_GAIN_MULT – Nominal Gain Bin Multiplier

Read: FLTRCTRL, Bit 2 (BNA_READ) controls which gain multiplier is read. If BNA_READ is 0, gain multiplier from Bank A is read. Otherwise, gain multiplier from Bank B is read.

Write: FLTRCTRL, Bit 7 (NL_PG_CONTROL) controls which gain multiplier is programmed. If NL_PG_CONTROL is 0, gain multiplier from Bank B is programmed. Otherwise, gain multiplier from Bank A is programmed.

Bits 23-18: POS_MID_GAIN_MULT – Positive Mid Gain Bin Multiplier

Read: FLTRCTRL, Bit 2 (BNA_READ) controls which gain multiplier is read. If BNA_READ is 0, gain multiplier from Bank A is read. Otherwise, gain multiplier from Bank B is read.

Write: FLTRCTRL, Bit 7 (NL_PG_CONTROL) controls which gain multiplier is programmed. If NL_PG_CONTROL is 0, gain multiplier from Bank B is programmed. Otherwise, gain multiplier from Bank A is programmed.

Bits 17-12: POS_LRG_GAIN_MULT – Positive Large Gain Bin Multiplier

Read: FLTRCTRL, Bit 2 (BNA_READ) controls which gain multiplier is read. If BNA_READ is 0, gain multiplier from Bank A is read. Otherwise, gain multiplier from Bank B is read.

Write: FLTRCTRL, Bit 7 (NL_PG_CONTROL) controls which gain multiplier is programmed. If NL_PG_CONTROL is 0, gain multiplier from Bank B is programmed. Otherwise, gain multiplier from Bank A is programmed.

Bits 11-6: NEG_MID_GAIN_MULT – Negative Mid Gain Bin Multiplier

Read: FLTRCTRL, Bit 2 (BNA_READ) controls which gain multiplier is read. If BNA_READ is 0, gain multiplier from Bank A is read. Otherwise, gain multiplier from

Bank B is read.

Write: FLTRCTRL, Bit 7 (NL_PG_CONTROL) controls which gain multiplier is programmed. If NL_PG_CONTROL is 0, gain multiplier from Bank B is programmed. Otherwise, gain multiplier from Bank A is read.

Bits 5-0: NEG_LRG_GAIN_MULT – Negative Large Gain Bin Multiplier

Read: FLTRCTRL, Bit 2 (BNA_READ) controls which gain multiplier is read. If BNA_READ is 0, gain multiplier from Bank A is read. Otherwise, gain multiplier from Bank B is read.

Write: FLTRCTRL, Bit 7 (NL_PG_CONTROL) controls which gain multiplier is programmed. If NL_PG_CONTROL is 0, gain multiplier from Bank B is programmed. Otherwise, gain multiplier from Bank A is programmed.

5.8 Filter Coefficient Register 1 (FLTRCOEF1)

Address FFF7E020 – Loop 4 Filter Coefficient Register 1

Address FFF7E420 – Loop 3 Filter Coefficient Register 1

Address FFF7E820 – Loop 2 Filter Coefficient Register 1

Address FFF7EC20 – Loop 1 Filter Coefficient Register 1

Bit Number	27:16	11:0
Bit Name	B01	B11
Access	R/W	R/W
Default	0011_1111_1011 – Bank A 0011_1001_0110 – Bank B	1000_1111_1111 – Bank A 1001_1101_0111 – Bank B

Bits 27-16: B01 – B₀₁ Coefficient

Read: FLTRCTRL, Bit 2 (BNA_READ) controls which bank coefficient is read. If BNA_READ is 0, B₀₁ Coefficient from Bank A is read. Otherwise, B₀₁ Coefficient from Bank B is read.

Write: FLTRCTRL, Bit 1 (BNA_ACTIVE) controls which bank coefficient is programmed. If BNA_ACTIVE is 0, B₀₁ Coefficient for Bank B is programmed. Otherwise, B₀₁ Coefficient for Bank A is programmed.

Bits 11-0: B11 – B₁₁ Coefficient

Read: FLTRCTRL, Bit 2 (BNA_READ) controls which bank coefficient is read. If BNA_READ is 0, B₁₁ Coefficient from Bank A is read. Otherwise, B₁₁ Coefficient from Bank B is read.

Write: FLTRCTRL, Bit 1 (BNA_ACTIVE) controls which bank coefficient is programmed. If BNA_ACTIVE is 0, B₁₁ Coefficient for Bank B is programmed. Otherwise, B₁₁ Coefficient for Bank A is programmed.

5.9 Filter Coefficient Register 2 (FLTRCOEF2)

Address FFF7E024 – Loop 4 Filter Coefficient Register 2

Address FFF7E424 – Loop 3 Filter Coefficient Register 2

Address FFF7E824 – Loop 2 Filter Coefficient Register 2

Address FFF7EC24 – Loop 1 Filter Coefficient Register 2

Bit Number	27:16	15:4	3:0
Bit Name	B21	RESERVED	COEFF_SCALER
Access	R/W	-	R/W
Default	0011_0001_0011 - Bank A 0010_1001_1011 - Bank B	0000_0000_0000	0011 - Bank A 0101 - Bank B

Bits 27-16: B21 – B₂₁ Coefficient

Read: FLTRCTRL, Bit 2 (BNA_READ) controls which bank coefficient is read. If BNA_READ is 0, B₂₁ Coefficient from Bank A is read. Otherwise, B₂₁ Coefficient from Bank B is read.

Write: FLTRCTRL, Bit 1 (BNA_ACTIVE) controls which bank coefficient is programmed. If BNA_ACTIVE is 0, B₂₁ Coefficient for Bank B is programmed. Otherwise, B₂₁ Coefficient for Bank A is programmed.

Bits 15-4: RESERVED – Unused bits

Bits 3-0: COEFF_SCALER – Coefficient Scaler

Read: FLTRCTRL, Bit 2 (BNA_READ) controls which bank scaler is read. If BNA_READ is 0, coefficient scaler from Bank A is read. Otherwise, coefficient scaler from Bank B is read.

Write: FLTRCTRL, Bit 1 (BNA_ACTIVE) controls which bank scaler is programmed. If BNA_ACTIVE is 0, coefficient scaler for Bank B is programmed. Otherwise, coefficient scaler for Bank A is programmed.

5.10 Filter Coefficient Register 3 (FLTRCOEF3)

Address FFF7E028 – Loop 4 Filter Coefficient Register 3

Address FFF7E428 – Loop 3 Filter Coefficient Register 3

Address FFF7E828 – Loop 2 Filter Coefficient Register 3

Address FFF7EC28 – Loop 1 Filter Coefficient Register 3

Bit Number	27:16	11:0
Bit Name	A11	A21
Access	R/W	R/W
Default	0001_0100_0100 – Bank A 0110_0001_0000 – Bank B	1111_1011_1100 – Bank A 1111_1101_1111 – Bank B

Bits 27-16: A11 – A₁₁ Coefficient

Read: FLTRCTRL, Bit 2 (BNA_READ) controls which bank coefficient is read. If BNA_READ is 0, A₁₁ Coefficient from Bank A is read. Otherwise, A₁₁ Coefficient from Bank B is read.

Write: FLTRCTRL, Bit 1 (BNA_ACTIVE) controls which bank coefficient is programmed. If BNA_ACTIVE is 0, A₁₁ Coefficient for Bank B is programmed. Otherwise, A₁₁ Coefficient for Bank A is programmed.

Bits 11-0: A21 – A₂₁ Coefficient

Read: FLTRCTRL, Bit 2 (BNA_READ) controls which bank coefficient is read. If BNA_READ is 0, A₂₁ Coefficient from Bank A is read. Otherwise, A₂₁ Coefficient from Bank B is read.

Write: FLTRCTRL, Bit 1 (BNA_ACTIVE) controls which bank coefficient is programmed. If BNA_ACTIVE is 0, A₂₁ Coefficient for Bank B is programmed. Otherwise, A₂₁ Coefficient for Bank A is programmed.

5.11 Filter Coefficient Register 4 (FLTRCOEF4)

Address FFF7E02C – Loop 4 Filter Coefficient Register 4

Address FFF7E42C – Loop 3 Filter Coefficient Register 4

Address FFF7E82C – Loop 2 Filter Coefficient Register 4

Address FFF7EC2C – Loop 1 Filter Coefficient Register 4

Bit Number	27:16	11:0
Bit Name	B12	A12
Access	R/W	R/W
Default	0000_0000_0000 – Bank A 1110_1011_0100 – Bank B	0000_0000_0000 – Bank A 0000_1111_0110 – Bank B

Bits 27-16: B12 – B₁₂ Coefficient

Read: FLTRCTRL, Bit 2 (BNA_READ) controls which bank coefficient is read. If BNA_READ is 0, B₁₂ Coefficient from Bank A is read. Otherwise, B₁₂ Coefficient from Bank B is read.

Write: FLTRCTRL, Bit 1 (BNA_ACTIVE) controls which bank coefficient is programmed. If BNA_ACTIVE is 0, B₁₂ Coefficient for Bank B is programmed. Otherwise, B₁₂ Coefficient for Bank A is programmed.

Bits 11-0: A12 – A₁₂ Coefficient

Read: FLTRCTRL, Bit 2 (BNA_READ) controls which bank coefficient is read. If BNA_READ is 0, A₁₂ Coefficient from Bank A is read. Otherwise, A₁₂ Coefficient from Bank B is read.

Write: FLTRCTRL, Bit 1 (BNA_ACTIVE) controls which bank coefficient is programmed. If BNA_ACTIVE is 0, A₁₂ Coefficient for Bank B is programmed. Otherwise, A₁₂ Coefficient for Bank A is programmed.

5.12 Filter Clamp Register (FLTRCLAMP)

Address FFF7E034 – Loop 4 Filter Clamp Register

Address FFF7E434 – Loop 3 Filter Clamp Register

Address FFF7E834 – Loop 2 Filter Clamp Register

Address FFF7EC34 – Loop 1 Filter Clamp Register

Bit Number	31:16	15:0
Bit Name	CLAMPH	CLAMPL
Access	R/W	R/W
Default	0111_1111_1111_1111	0000_0000_0000_0000

Bits 31-16: CLAMPH - Clamp Upper Value - Sets upper limit of CLA output

Read: FLTRCTRL, Bit 2 (BNA_READ) controls which bank clamp is read. If BNA_READ is 0, Upper Clamp value from Bank A is read. Otherwise, Upper Clamp value from Bank B is read.

Write: FLTRCTRL, Bit 6 (CLAMP_PG_CONTROL) controls which bank clamp is programmed. If CLAMP_PG_CONTROL is 0, Upper Clamp value for Bank B is programmed. Otherwise, Upper Clamp value for Bank A is programmed.

Bits 15-0: CLAMPL - Clamp Lower Value – Sets lower limit of CLA output

Read: FLTRCTRL, Bit 2 (BNA_READ) controls which bank clamp is read. If BNA_READ is 0, Lower Clamp value from Bank A is read. Otherwise, Lower Clamp value from Bank B is read.

Write: FLTRCTRL, Bit 6 (CLAMP_PG_CONTROL) controls which bank clamp is programmed. If CLAMP_PG_CONTROL is 0, Lower Clamp value for Bank B is programmed. Otherwise, Lower Clamp value for Bank A is programmed.

5.13 Filter Nonlinear Response Register 2 (FLTRNLR2)

Address FFF7E038 – Loop 4 Nonlinear Response Register 2

Address FFF7E438 – Loop 3 Nonlinear Response Register 2

Address FFF7E838 – Loop 2 Nonlinear Response Register 2

Address FFF7EC38 – Loop 1 Nonlinear Response Register 2

Bit Number	29:24	21:16	13:8	5:0
Bit Name	LIMIT3	LIMIT2	LIMIT1	LIMIT0
Access	R/W	R/W	R/W	R/W
Default	00_0000	00_0000	00_0000	00_0000

Bits 29-24: LIMIT3 – Positive Large Gain Limit. If error input exceeds this limit, the gain will be set to POS_LRG_GAIN_MULT (FLTRNLR1, Bits 17-12).

Read: FLTRCTRL, Bit 2 (BNA_READ) controls which bank limit is read. If BNA_READ is 0, limit from Bank A is read. Otherwise, limit from Bank B is read.

Write: FLTRCTRL, Bit 7 (NL_PG_CONTROL) controls which bank limit is programmed. If NL_PG_CONTROL is 0, limit from Bank B is programmed. Otherwise, limit from Bank A is programmed.

Bits 21-16: LIMIT2 – Positive Mid Gain Limit. If error input exceeds this limit and is less than LIMIT3, the gain will be set to POS_MID_GAIN_MULT (FLTRNLR1, Bits 23-18).

Read: FLTRCTRL, Bit 2 (BNA_READ) controls which bank limit is read. If BNA_READ is 0, limit from Bank A is read. Otherwise, limit from Bank B is read.

Write: FLTRCTRL, Bit 7 (NL_PG_CONTROL) controls which bank limit is programmed. If NL_PG_CONTROL is 0, limit from Bank B is programmed. Otherwise, limit from Bank A is programmed.

Bits 13-8: LIMIT1 – Nominal Gain Limit. If error input exceeds this limit and is less than LIMIT2, the gain will be set to NOM_GAIN_MULT (FLTRNLR1, Bits 29-24).

Read: FLTRCTRL, Bit 2 (BNA_READ) controls which bank limit is read. If BNA_READ is 0, limit from Bank A is read. Otherwise, limit from Bank B is read.

Write: FLTRCTRL, Bit 7 (NL_PG_CONTROL) controls which bank limit is programmed. If NL_PG_CONTROL is 0, limit from Bank B is programmed. Otherwise, limit from Bank A is programmed.

Bits 5-0: LIMIT0 – Negative Mid Gain Limit. If error input exceeds this limit and is less than LIMIT1, the gain will be set to NEG_MID_GAIN_MULT (FLTRNLR1, Bits 11-6). If error input falls below this limit, the gain will be set to NEG_LRG_GAIN_MULT (FLTRNLR1, Bits 5-0).

Read: FLTRCTRL, Bit 2 (BNA_READ) controls which bank limit is read. If BNA_READ is 0, limit from Bank A is read. Otherwise, limit from Bank B is read.

Write: FLTRCTRL, Bit 7 (NL_PG_CONTROL) controls which bank limit is programmed. If NL_PG_CONTROL is 0, limit from Bank B is programmed. Otherwise, limit from Bank A is programmed.

5.14 Filter Front End Control Register (FLTRFECTRL)

Address FFF7E040 – Loop 4 Filter Front End Control Register

Address FFF7E440 – Loop 3 Filter Front End Control Register

Address FFF7E840 – Loop 2 Filter Front End Control Register

Address FFF7EC40 – Loop 1 Filter Front End Control Register

Bit Number	3	2	1:0
Bit Name	SC_CLK_DIV_2	SC_GAIN_FILTER_SEL	RESERVED
Access	R/W	R/W	-
Default	0	0	01

Bit 3 – SC_CLK_DIV_2 – Switch Cap Clock Divider Select

0 = Switch Cap Clock divide by 1 (Default)

1 = Switch Cap Clock divide by 2

Bit 2 – SC_GAIN_FILTER_SEL – Switch Cap Noise Filter Enable

0 = Disables Switch Cap Noise Filter (Default)

1 = Enables Switch Cap Noise Filter

Bits 1-0: RESERVED – Unused bits, default to 01.

6 Loop 1-4 DPWM Registers

Registers for DPWM modules 1-4 are identical in their bit definitions. The addresses given, and the number of DPWM modules are from the UCD3040. For other devices, consult their respective data sheets.

6.1 DPWM Control Register 1 (DPWMCTRL1)

Address FFF7E100 – Loop 4 DPWM Control Register 1

Address FFF7E500 – Loop 3 DPWM Control Register 1

Address FFF7E900 – Loop 2 DPWM Control Register 1

Address FFF7ED00 – Loop 1 DPWM Control Register 1

Bit Number	31:30	29	28	27:24
Bit Name	CLA_CH_SEL	FAULT_ENA	SYNC_SLAVE_EN	SYNC_OUT_DIV_SEL
Access	R/W	R/W	R/W	R/W
Default	00	0	0	0000

Bit Number	23	22	21:20
Bit Name	PWM_B_OE	PWM_A_OE	MSYNC_CH_SEL

Access	R/W	R/W	R/W
Default	0	0	00

Bit Number	19	18	17	16
Bit Name	GPIO_B_VAL	GPIO_B_ENA	GPIO_A_VAL	GPIO_A_ENA
Access	R/W	R/W	R/W	R/W
Default	0	0	0	0

Bit Number	15	14	13:12
Bit Name	CHECK_OVERRIDE	RSVD	OVERSAMPLE
Access	R/W	R/W	R/W
Default	0	0	00

Bit Number	11	10	9
Bit Name	SFRAME_TRIGGER	SFRAME_ENA	UPDATE_END_PRD_ENA
Access	R/W	R/W	R/W
Default	0	0	0

Bit Number	8	7:5	4
Bit Name	MSYNC_SLAVE_ENA	CLA_SCALE	CLA_ENABLE
Access	R/W	R/W	R/W
Default	0	000	0

Bit Number	3	2	1	0
Bit Name	MULTI_OUT_MODE	RESONANCE_MODE	PHASE_MODE	PWM_ENA
Access	R/W	R/W	R/W	R/W
Default	0	0	0	0

Bits 31-30: CLA_CH_SEL – CLA Select, configures input for DPWM

00 = CLA1 (Default)

01 = CLA2

10 = CLA3

11 = CLA4

Bit 29: FAULT_ENA – Fault PWM Shutdown

0 = Fault signal does not disable PWM signals (Default)

1 = Enables the shutting off of the PWM signals when a fault signal has been received

Bit 28: SYNC_SLAVE_EN – Slave DPWM to external sync

0 = DPWM not synchronized to external sync (Default)

1 = Slave DPWM to external sync

Bits 27-24: SYNC_OUT_DIV_SEL – Sets the divider for generating the Sync Out pulse.

0000 = Sync Out generated on every switching cycle (Default)

0001 = Sync Out generated once every 2 switching cycles

0010 = Sync Out generated once every 3 switching cycles

.....

1111 = Sync Out generated once every 16 switching cycles

Bit 23: PWM_B_OE – Output Enable (Direction) for PWM B

0 = PWM_B output enable asserted, PWM B configured as output (Default)

1 = PWM_B output enable disabled, PWM B configured as input

Bit 22: PWM_A_OE – Output Enable (Direction) for PWM A

0 = PWM_A output enable asserted, PWM A configured as output (Default)

- 1 = PWM_A output enable disabled, PWM A configured as input
- Bits 21-20: MSYNC_CH_SEL** – Select which Sync input to use to do phase offsets
- 00 = DPWM1 (Default)
 - 01 = DPWM2
 - 10 = DPWM3
 - 11 = DPWM4
- Bit 19: GPIO_B_VAL** – Sets value of PWM B output in GPIO mode
- 0 = PWM B driven low in GPIO mode (Default)
 - 1 = PWM B driven high in GPIO mode
- Bit 18: GPIO_B_ENA** – Enables GPIO mode for PWM B output
- 0 = PWM B in DPWM mode (Default)
 - 1 = PWM B in GPIO mode
- Bit 17: GPIO_A_VAL** – Sets value of PWM A output in GPIO mode
- 0 = PWM A driven low in GPIO mode (Default)
 - 1 = PWM A driven high in GPIO mode
- Bit 16: GPIO_A_ENA** – Enables GPIO mode for PWM A output
- 0 = PWM A in DPWM mode (Default)
 - 1 = PWM A in GPIO mode
- Bit 15: CHECK_OVERRIDE** – PWM Check Override
- 0 = DPWM checks mathematical settings within module, correct placement of Event settings/period settings. Invalid configurations are not allowed. (Default)
 - 1 = Overrides checking for invalid configurations and turns off PWM mathematical checking functions.
- Bit 14: RSVD** –
Reserved for future use.
- Bits 13-12: OVERSAMPLE** –
- 00 = Trigger an EADC Sample at PWM Sample Trig Register value (Default)
 - 01 = Trigger an EADC Sample at PWM Sample Trig Register value and at PWM Sample Trig Register value divided by 2
 - 10 = Trigger a EADC Sample at PWM Sample Trig Register value, at PWM Sample Trig Register value divided by 2 and at PWM Sample Trig Register value divided by 4
 - 11 = Trigger a EADC Sample at PWM Sample Trig Register value, at PWM Sample Trig Register value divided by 2, at PWM Sample Trig Register value divided by 4 and at PWM Sample Trig Register value divided by 8
- Bit 11: SFRAME_TRIGGER** –
- 0 to 1 = Trigger to next Single Step Frame
 - 1 to 0 = Resets Trigger
- Bit 10: SFRAME_ENA** – PWM Single Step Frame Mode Enable
- 0 = Disable Single Frame Mode (Default)
 - 1 = Enable Single Step Frame Mode. One EADC sample is requested, CLA then Filters, then one PWM duty cycle performed, then wait on Single Frame Trigger toggle before advancing to next frame.
- Bit 9: UPDATE_END_PRD_ENA** – Update End Period Mode
- 0 = Disable Update End Period Mode (Default)
 - 1 = Enable Update End Period Mode. PWM updates the CLA Value at the end of the frame. Update Lockout Enable Mode (Bit 14) must be set to 0.
- Bit 8: MSYNC_SLAVE_ENA** – Multi-Sync Slave Mode Control
- 0 = PWM not synchronized to another PWM channel (Default)
 - 1 = Enable Multi-Sync Slave Mode, current channel will be slaved from corresponding channel, as specified by CLA_CH_SEL (Bits 31-30)
- Bits 7-5: CLA_SCALE** – Scaling for CLA Input Data
- 000 = CLA Value (Default)
 - 001 = CLA Value multiplied by 2
 - 010 = CLA Value divided by 2
 - 011 = CLA Value multiplied by 4
 - 100 = CLA Value divided by 4

- 101 = CLA Value multiplied by 8
110 = CLA Value divided by 8
111 = CLA Value
- Bit 4: CLA_ENABLE** – CLA Processing Enable
0 = Generate PWM waveforms from PWM Register values (Default)
1 = Enable CLA input
- Bit 3: MULTI_OUT_MODE** – Multi-Output Mode Enable
0 = Disable Multi-Output Mode (Default)
1 = Enable Multi-output mode, A and B independent channels.
Refer to DPWM app note for more info
- Bit 2: RESONANCE_MODE**– Resonance Mode Enable
0 = Disable Resonance Mode (Default)
1 = Enable Resonance Mode, refer to DPWM app note for more info
- Bit 1: PHASE_MODE** – CLA Phase Mode Enable
0 = Disable CLA Phase Mode (Default)
1 = Enable CLA Phase Mode (used in Phase Shifted Full Bridge applications)
- Bit 0: PWM_ENA** – PWM Processing Enable
0 = Disable PWM module, outputs zero (Default)
1 = Enable PWM operation

6.2 DPWM Control Register 2 (DPWMCTRL2)

Address FFF7E104 – Loop 4 DPWM Control Register 2

Address FFF7E504 – Loop 3 DPWM Control Register 2

Address FFF7E904 – Loop 2 DPWM Control Register 2

Address FFF7ED04 – Loop 1 DPWM Control Register 2

Bit Number	7	6	5	4
Bit Name	PWM_B_INV	PWM_A_INV	PWM_B_PROT_DIS	PWM_A_PROT_DIS
Access	R/W	R/W	R/W	R/W
Default	0	0	0	0

Bit Number	3:2	1	0
Bit Name	HIRES_SCALE	ALL_PHASE_CLK_ENA	HIRES_DIS
Access	R/W	R/W	R/W
Default	00	0	0

- Bit 7: PWM_B_INV** – PWM B Output Polarity Control
0 = Non-inverted PWM B output (Default)
1 = Inverts PWM B output
- Bit 6: PWM_A_INV** – PWM A Output Polarity Control
0 = Non-inverted PWM A output (Default)
1 = Inverted PWM A output
- Bit 5: PWM_B_PROT_DIS** – PWM B Asynchronous Protection Disable
0 = Allows asynchronous protection logic to turn off PWM B Output (Default)
1 = Disables asynchronous protection logic from turning off PWM B Output
- Bit 4: PWM_A_PROT_DIS** – PWM A Asynchronous Protection Disable
0 = Allows asynchronous protection logic to turn off PWM A Output (Default)
1 = Disables asynchronous protection logic from turning off PWM B Output
- Bits 3-2: HIRES_SCALE** – Determines resolution of high resolution steps
00 = Resolution of 16 phases. Full resolution enabled. Resolution step = PCLK/16 (Default)
11 = Resolution of 2 phases. Resolution step = PCLK/2
10 = Resolution of 4 phases. Resolution step = PCLK/4
01 = Resolution of 8 phases. Resolution step = PCLK/8

00 = Resolution of 16 phases. Full Resolution enabled.
Resolution step = PCLK/16

Bit 1: ALL_PHASE_CLK_ENA – High Speed Oscillator Phase Enable
0 = Enables only required phases of clock when needed (Default)
1 = Enables all phases of high resolution clock from oscillator

Bit 0: HIRES_DIS – PWM High Resolution Disable
0 = Enable High Resolution logic (Default)
1 = Disable High Resolution logic

6.3 DPWM Period Register (DPWMPRD)

Address FFF7E108 – Loop 4 DPWM Period Register

Address FFF7E508 – Loop 3 DPWM Period Register

Address FFF7E908 – Loop 2 DPWM Period Register

Address FFF7ED08 – Loop 1 DPWM Period Register

Bit Number	13:0
Bit Name	PRD
Access	R/W
Default	00_0011_0100_0001

Bits 13-0: PRD – PWM Period equals the number of PCLK clock periods

6.4 DPWM Event 1 Register (DPWMEV1)

Address FFF7E10C – Loop 4 DPWM Event 1 Register

Address FFF7E50C – Loop 3 DPWM Event 1 Register

Address FFF7E90C – Loop 2 DPWM Event 1 Register

Address FFF7ED0C – Loop 1 DPWM Event 1 Register

Bit Number	13:0
Bit Name	EVENT1
Access	R/W
Default	00_0000_0000_0100

Bits 13-0: EVENT1 – Configures the location of Event 1. Value equals number of PCLK clock periods. Refer to DPWM app note for additional information.

6.5 DPWM Event 2 Register (DPWMEV2)

Address FFF7E110 – Loop 4 DPWM Event 2 Register

Address FFF7E510 – Loop 3 DPWM Event 2 Register

Address FFF7E910 – Loop 2 DPWM Event 2 Register

Address FFF7ED10 – Loop 1 DPWM Event 2 Register

Bit Number	17:0
Bit Name	EVENT2
Access	R/W
Default	0_0000_0001_0110_0011

Bits 17-0: EVENT2 – Configures the location of Event 2. Value equals number of PCLK clock periods in Bits 17:4 and number of high resolution clock phases of PCL in Bits 3:0 (dependent on Bits 3:2 of DPWM Control Register 2). Refer to DPWM app note for additional information.

6.6 DPWM Event 3 Register (DPWMEV3)

Address FFF7E114 – Loop 4 DPWM Event 3 Register

Address FFF7E514 – Loop 3 DPWM Event 3 Register

Address FFF7E914 – Loop 2 DPWM Event 3 Register

Address FFF7ED14 – Loop 1 DPWM Event 3 Register

Bit Number	17:0
Bit Name	EVENT3
Access	R/W
Default	00_0000_0001_1001_0111

Bits 17-0: EVENT3 – Configures the location of Event 3. Value equals number of PCLK clock periods in Bits 17:4 and number of high resolution clock phases of PCL in Bits 3:0. Refer to DPWM app note for additional information.

6.7 DPWM Event 4 Register (DPWMEV4)

Address FFF7E118 – Loop 4 DPWM Event 4 Register

Address FFF7E518 – Loop 3 DPWM Event 4 Register

Address FFF7E918 – Loop 2 DPWM Event 4 Register

Address FFF7ED18 – Loop 1 DPWM Event 4 Register

Bit Number	17:0
Bit Name	EVENT4
Access	R/W
Default	00_0011_0011_1000_1010

Bits 17-0: EVENT4 – Configures the location of Event 4. Value equals number of PCLK clock periods in Bits 17:4 and number of high resolution clock phases of PCL in Bits 3:0. Refer to DPWM app note for additional information.

6.8 DPWM Sample Trigger Register (DPWMSAMPTRIG)

Address FFF7E11C – Loop 4 DPWM Sample Trigger Register

Address FFF7E51C – Loop 3 DPWM Sample Trigger Register

Address FFF7E91C – Loop 2 DPWM Sample Trigger Register

Address FFF7ED1C – Loop 1 DPWM Sample Trigger Register

Bit Number	11:0
Bit Name	SAMPLE_TRIGGER
Access	R/W
Default	0000_0010_0000

Bits 11-0: SAMPLE_TRIGGER – Configures the location of the sample trigger within a PWM period. Value equals the number of PCLK clock periods. Enables start of conversion for EADC. Refer to DPWM app note for additional information.

6.9 DPWM Phase Trigger Register (DPWMPHASETRIG)

Address FFF7E120 – Loop 4 DPWM Phase Trigger Register

Address FFF7E520 – Loop 3 DPWM Phase Trigger Register

Address FFF7E920 – Loop 2 DPWM Phase Trigger Register

Address FFF7ED20 – Loop 1 DPWM Phase Trigger Register

Bit Number	13:0
Bit Name	PHASE_TRIGGER
Access	R/W
Default	00_0000_0000_0000

Bits 13-0: PHASE_TRIGGER – Configures the phase trigger delay within multi-output mode. Value equals the number of PCLK clock periods. Refer to DPWM app note for additional information.

6.10 DPWM Cycle Adjust A Register (DPWMCYCADJA)

Address FFF7E124 – Loop 4 DPWM Cycle Adjust A Register

Address FFF7E524 – Loop 3 DPWM Cycle Adjust A Register

Address FFF7E924 – Loop 2 DPWM Cycle Adjust A Register

Address FFF7ED24 – Loop 1 DPWM Cycle Adjust A Register

Bit Number	15:0
Bit Name	CYCLE_ADJUST_A
Access	R/W
Default	0000_0000_0000_0000

Bits 15-0: CYCLE_ADJUST_A – Adjusts PWM A output signal. 16-bit signed number allows output signal to be delayed or sped up. Refer to DPWM app note for additional information.

6.11 DPWM Cycle Adjust B Register (DPWMCYCADJB)

Address FFF7E128 – Loop 4 DPWM Cycle Adjust B Register

Address FFF7E528 – Loop 3 DPWM Cycle Adjust B Register

Address FFF7E928 – Loop 2 DPWM Cycle Adjust B Register

Address FFF7ED28 – Loop 1 DPWM Cycle Adjust B Register

Bit Number	15:0
Bit Name	CYCLE_ADJUST_B
Access	R/W
Default	0000_0000_0000_0000

Bits 15-0: CYCLE_ADJUST_B – Adjusts the PWM B output signal. 16-bit signed number allows output signal to be delayed or sped up. Refer to DPWM app note for additional information.

6.12 DPWM Current Limit Flag Control Register (DPWMCLFCTRL)

Address FFF7E12C – Loop 4 DPWM Current Limit Flag Control Register

Address FFF7E52C – Loop 3 DPWM Current Limit Flag Control Register

Address FFF7E92C – Loop 2 DPWM Current Limit Flag Control Register

Address FFF7ED2C – Loop 1 DPWM Current Limit Flag Control Register

Bit Number	24	23:16	15:11	10
------------	----	-------	-------	----

Bit Name	CLF	COUNT	RESERVED	COUNT_CONT
Access	R	R	-	R/W
Default	-	-	00000	0

Bit Number	9	8	7:0
Bit Name	COUNT_CLR	CLF_ENA	COUNT_MAX
Access	R/W	R/W	R/W
Default	0	0	0000_0000

Bit 24: CLF – Current Limit Flag Status

0 = Current Limit Flag Not Set

1 = Current Limit Flag Set

Bits 23-16: COUNT – Current Limit Indicator Counter, equals the number of received current limit indicators, detected by CLF logic at end of DPWM cycle.

Bits 15-11: RESERVED – Unused bits in current application

Bit 10: COUNT_CONT – Current Limit Indicator Counter Configuration

0 = Current Limit Indicator Counter (Bits 23-16) clears if no Current Limit Indicator is found at the end of the DPWM cycle (Default)

1 = Current Limit Indicator Counter provides a cumulative count of Current Limit Indicators and does not clear on an absence of a Current Limit Indicator at the end of the DPWM cycle

Bit 9: COUNT_CLR – Current Limit Indicator Counter Clear

0 = No Clear of Current Limit Counter (Default)

1 = Clears Current Limit Counter

Bit 8: CLF_ENA – Current Limit Detect Enable

0 = Disables CLF detect circuitry (Default)

1 = Enables CLF detect circuitry

Bits 7-0: COUNT_MAX – Current Limit Count, sets the number of received Current Limit Indicators before asserting the CLF Flag (Bit 24)

7:0 = Binary Count ranging from 0 to 127

6.13 DPWM Overflow Register (DPWMOVERFLOW)

Address FFF7E130 – Loop 4 DPWM Overflow Register

Address FFF7E530 – Loop 3 DPWM Overflow Register

Address FFF7E930 – Loop 2 DPWM Overflow Register

Address FFF7ED30 – Loop 1 DPWM Overflow Register

Bit Number	5	4	3:0
Bit Name	GPIO_B_IN	GPIO_A_IN	OVERFLOW
Access	R	R	R
Default	-	-	-

Bit 5: GPIO_B_IN – value on PWM B as input

0 = Low signal on DPWMB pin if used as input

1 = High signal on DPWMB pin if used as input

Bit 4: GPIO_A_IN – value on PWM A as input

0 = Low signal on DPWMA pin if used as input

1 = High signal on DPWMA pin if used as input

Bit 3: OVERFLOW[3] – CLA Event 4 Overflow Status

0 = CLA Event 4 has not overflowed

1 = Overflow condition found on CLA Event 4

Bit 2: OVERFLOW[2] – PWM Event 4 Overflow Status

0 = PWM Event 4 has not overflowed

1 = Overflow condition found on PWM Event 4

Bit 1: OVERFLOW[1] – CLA Event 3 Overflow Status

0 = CLA Event 3 has not overflowed

1 = Overflow condition found on CLA Event 3

Bit 0: OVERFLOW[0] – CLA Event 2 Overflow Status

0 = CLA Event 2 has not overflowed

1 = Overflow condition found on CLA Event 2

6.14 DPWM Interrupt Register (DPWMINT)

Address FFF7E134 – Loop 4 DPWM Interrupt Register

Address FFF7E534 – Loop 3 DPWM Interrupt Register

Address FFF7E934 – Loop 2 DPWM Interrupt Register

Address FFF7ED34 – Loop 1 DPWM Interrupt Register

Bit Number	7	6	5	4	3:0
Bit Name	CLF	PRD	CLF_INT_ENA	PRD_INT_ENA	PRD_INT_SCALE
Access	R	R	R/W	R/W	R/W
Default	-	-	0	0	1111

Bit 7: CLF – Current Limit Flag

0 = Current Limit Flag is not asserted

1 = Current Limit Flag is set

Bit 6: PRD – PWM Period Interrupt Flag

0 = PWM Period Interrupt Flag is not asserted

1 = PWM Period Interrupt Flag is set

Bit 5: CLF_INT_ENA – Current Limit Flag Interrupt Enable

0 = Disables generation of interrupt for Current Limit Events (Default)

1 = Enables generation of interrupt if number of current limit flags exceed limit configured in Bits 7-0 of the DPWM Current Limit Flag Control Register.

Bit 4: PRD_INT_ENA – PWM Period Interrupt Enable

0 = Disables generation of periodic PWM interrupt (Default)

1 = Enables generation of periodic PWM interrupt, flag will be set Bit 6 of this register
Bits 3-0: PRD_INT_SCALE – This value scales the period interrupt signal from an interrupt every switching cycle to 16 switching cycles

0000 = Period Interrupt generated every switching cycle
 0001 = Period Interrupt generated once every 2 switching cycles
 0010 = Period Interrupt generated once every 4 switching cycles
 0011 = Period Interrupt generated once every 8 switching cycles
 0100 = Period Interrupt generated once every 16 switching cycles
 0101 = Period Interrupt generated once every 16 switching cycles
 0110 = Period Interrupt generated once every 32 switching cycles
 0111 = Period Interrupt generated once every 48 switching cycles
 1000 = Period Interrupt generated once every 64 switching cycles
 1001 = Period Interrupt generated once every 80 switching cycles
 1010 = Period Interrupt generated once every 96 switching cycles
 1011 = Period Interrupt generated once every 128 switching cycles
 1100 = Period Interrupt generated once every 160 switching cycles
 1101 = Period Interrupt generated once every 192 switching cycles
 1110 = Period Interrupt generated once every 224 switching cycles
 1111 = Period Interrupt generated once every 256 switching cycles (Default)

6.15 EADC Control Register (EADCCTRL)

Address FFF7E140 – Loop 4 EADC Control Register

Address FFF7E540 – Loop 3 EADC Control Register

Address FFF7E940 – Loop 2 EADC Control Register

Address FFF7ED40 – Loop 1 EADC Control Register

Bit Number	1	0
Bit Name	SCFE_ENA	EADC_ENA
Access	R/W	R/W
Default	1	1

Bit 1: SCFE_ENA – Switch Cap Front Enable

0 = Disables Switch Cap Front End logic

1 = Enables Switch Cap Front End logic (Default)

Bit 0: EADC_ENA – EADC Enable

0 = Disables EADC

1 = Enables EADC (Default)

6.16 EADC Status Register (EADCST)

Address FFF7E14C – Loop 4 EADC Status Register

Address FFF7E54C – Loop 3 EADC Status Register

Address FFF7E94C – Loop 2 EADC Status Register

Address FFF7ED4C – Loop 1 EADC Status Register

Bit Number	0
Bit Name	EOC
Access	R
Default	-

Bit 0: EOC – Indicates EADC end of conversion

6.17 EADC DAC Value Register (EADCDAC)

Address FFF7E150 – Loop 4 EADC DAC Value Register

Address FFF7E550 – Loop 3 EADC DAC Value Register

Address FFF7E950 – Loop 2 EADC DAC Value Register

Address FFF7ED50 – Loop 1 EADC DAC Value Register

Bit Number	9-0
Bit Name	DAC_VALUE
Access	R/W
Default	00_1111_1111

Bits 9-0: DAC_VALUE - Programmable DAC Value

7 Other Registers mentioned

Three other registers not in the Compensator and DPWM sections are also mentioned in the user's guide.

The first two, SYNCCTRL and CLFCTRL, are in the Miscellaneous Analog Control area. The third one, CLKCNTL, is in the SYS area.

7.1 Comparator Control Register (COMPCTRL)

Address FFF7F000

Bit Number	25	24	23:18
Bit Name	COMP_INT_ENA	COMP_DIS	COMP_ADJ_D
Access	R/W	R/W	R/W
Default	0	1	000000

Bit Number	17:12	11:6	5:0
Bit Name	COMP_ADJ_C	COMP_ADJ_B	COMP_ADJ_A
Access	R/W	R/W	R/W
Default	000000	000000	000000

Bit 25: COMP_INT_ENA – Analog Comparators Interrupt Enable

0 = Disables Analog Comparator Interrupt (Default)

1 = Enables Analog Comparator Interrupt

Bit 24: COMP_DIS – Analog Comparators Disable

0 = Enables Analog Comparators

1 = Disables Analog Comparators (Default)

Bits 23-18: COMP_ADJ_D – Reference Select for Analog Comparator D

000000 = Comparator Reference of 31.250 mV (Default)

000001 = Comparator Reference of 62.5 mV

.....

111111 = Comparator Reference of 2.0 V

Bits 17-12: COMP_ADJ_C – Reference Select for Analog Comparator C

000000 = Comparator Reference of 31.250 mV (Default)

000001 = Comparator Reference of 62.5 mV

.....

111111 = Comparator Reference of 2.0 V

Bits 11-6: COMP_ADJ_B – Reference Select for Analog Comparator B

000000 = Comparator Reference of 31.250 mV (Default)

000001 = Comparator Reference of 62.5 mV

.....

111111 = Comparator Reference of 2.0 V

Bits 5-0: COMP_ADJ_A – Reference Select for Analog Comparator A

000000 = Comparator Reference of 31.250 mV (Default)

000001 = Comparator Reference of 62.5 mV

.....

111111 = Comparator Reference of 2.0 V

7.2 Sync Control Register (SYNCCTRL)

Address FFF7F024

Bit Number	17	16	15:6	5:4
Bit Name	SYNC_OUT_IN	SYNC_INPUT_IN	RESERVED	SYNC_MUX_SEL

Access	R	R	-	R/W
Default	-	-	0000_0000_00	00

Bit Number	3	2	1	0
Bit Name	RESERVED	SYNC_OUTPUT_DIR	SYNC_INPUT_OUT	SYNC_INPUT_DIR
Access	-	R/W	R/W	R/W
Default	0	1	1	1

Bit 17: SYNC_OUT_IN – Value of SYNC_OUTPUT pin when configured as GPIO

0 = Logic level low present on SYNC_OUTPUT pin

1 = Logic level high present on SYNC_OUTPUT pin

Bit 16: SYNC_INPUT_IN – Value of SYNC_INPUT pin when configured as GPIO

0 = Logic level low present on SYNC_INPUT pin

1 = Logic level high present on SYNC_INPUT pin

Bit 15-6: RESERVED – Unused bits in current application

Bits 5-4: SYNC_MUX_SEL – Selects which loop controls SYNC_OUTPUT

00 = Loop 1 Sync Output (Default)

01 = Loop 2 Sync Output

10 = Loop 3 Sync Output

11 = Loop 4 Sync Output

Bit 3: RESERVED – Unused bits in current application

Bit 2: SYNC_OUTPUT_DIR – Configures direction of SYNC_OUTPUT pin

0 = SYNC_OUTPUT pin configured as an output pin

1 = SYNC_OUTPUT pin configured as an input pin (Default)

Bit 1: SYNC_INPUT_OUT – Configure output value for SYNC_INPUT pin, if used as an output

0 = SYNC_INPUT pin driven low in output mode

1 = SYNC_INPUT pin driven high in output mode (Default)

Bit 0: SYNC_INPUT_DIR – Configure direction of SYNC_INPUT pin

0 = SYNC_INPUT pin configured as an output pin

1 = SYNC_INPUT pin configured as an input pin (Default)

7.3 Comparator Read Register (COMPREAD)

Address FFF7F01C

Bit Number	4	3	2	1	0
Bit Name	COMP_INT	COMP_D	COMP_C	COMP_B	COMP_A
Access	R	R	R	R	R
Default	-	-	-	-	-

Bit 4: COMP_INT – Analog Comparator Interrupt

0 = Analog Comparator Interrupt is not asserted

1 = Analog Comparator Interrupt is asserted

Bit 3: COMP_D - Comparator D Status

0 = Analog Comparator D is unset

1 = Analog Comparator D has been set (Tripped)

Bit 2: COMP_C - Comparator C Status

0 = Analog Comparator C is unset

1 = Analog Comparator C has been set (Tripped)

Bit 1: COMP_B - Comparator B Status

0 = Analog Comparator B is unset

1 = Analog Comparator B has been set (Tripped)

Bit 0: COMP_A - Comparator A Status

0 = Analog Comparator A is unset

1 = Analog Comparator A has been set (Tripped)

7.4 CLF Control Register (CLFCTRL)

Address FFF7F030

Bit Number	7:6	5:4	3:2	1:0
Bit Name	L4_CLF_CTRL	L3_CLF_CTRL	L2_CLF_CTRL	L1_CLF_CTRL
Access	R/W	R/W	R/W	R/W
Default	11	10	01	00

Bits 7-6: L4_CLF_CTRL – Selects connection for Loop 4 CLF Input

- 00 = Routed from output of Analog Comparator A
- 01 = Routed from output of Analog Comparator B
- 10 = Routed from output of Analog Comparator C
- 11 = Routed from output of Analog Comparator D (Default)

Bits 5-4: L3_CLF_CTRL – Selects connection for Loop 3 CLF Input

- 00 = Routed from output of Analog Comparator A
- 01 = Routed from output of Analog Comparator B
- 10 = Routed from output of Analog Comparator C (Default)
- 11 = Routed from output of Analog Comparator D

Bits 3-2: L2_CLF_CTRL – Selects connection for Loop 2 CLF Input

- 00 = Routed from output of Analog Comparator A
- 01 = Routed from output of Analog Comparator B (Default)
- 10 = Routed from output of Analog Comparator C
- 11 = Routed from output of Analog Comparator D

Bits 1-0: L1_CLF_CTRL – Selects connection for Loop 1 CLF Input

- 00 = Routed from output of Analog Comparator A (Default)
- 01 = Routed from output of Analog Comparator B
- 10 = Routed from output of Analog Comparator C
- 11 = Routed from output of Analog Comparator D

7.5 Clock Control Register (CLKCNTL)

Address FFFFFFFD0

The clock control Register configures the SYNC_OUTPUT pin and a bit that controls the module low power mode. CLKCNTL is accessible in user and privilege mode and supports byte, half-word and word accesses. Any access to this Register takes two SYSCLK cycles.

Bit Number	9:8	7	6:5	4	3	2:0
Bit Name	M_DIV_RATIO	RESERVED	CLKSR	CLKDIR	CLKDOUT	RESERVED
Access	-	-	R/W	R/W	R/W	-
Default	00	0	00	0	0	000

Bits 9-8: M_DIV_RATIO – MCLK (Processor Clock) Divide Ratio

- 00 = MCLK frequency equals High Frequency Oscillator divided by 8 (Default)
- 01 = MCLK frequency equals High Frequency Oscillator divided by 16
- 10 = MCLK frequency equals High Frequency Oscillator divided by 32
- 11 = MCLK frequency equals High Frequency Oscillator divided by 64

Bit 7: RESERVED

Bit 6-5: CLKSR – These bits control the source/function of the SYNC_OUTPUT pin.

- 00 = Configured as a digital input/output pin (Default)
- 01 = Driven by the interface clock (ICLK)
- 10 = Driven by the CPU clock (MCLK)

- 11 = Driven by the system clock (SYSCLK)
- Bit 4: CLKDIR** – This bit represents the SYNC_OUTPUT pin direction.
0 = CLKOUT pin is configured as an input (Default)
1 = CLKOUT pin is configured as an output
- Bit 3: CLKDOUT** – This pin represents the SYNC_OUTPUT output pin data.
0 = CLKOUT driven to logic low in output mode (Default)
1 = CLKOUT driven to logic high in output mode
- Bits 2-0: RESERVED**

7.6 Clock Control Register2 (CLKCTRL)

Address FFF7F004

The clock control Register2 allows control of the module high and low frequency oscillator.

Bit Number	4	3	2:1	0
Bit Name	HI_OSC_FLT_ENA	LO_OSC_DIS	RESERVED	RESERVED
Access	R/W	R/W	R/W***	R/W***
Default	0	0	***	***

******* These three bits (0~2) **MUST NOT** be changed at any time. If the user wants to reconfigure the other two bits (3~4), then he **MUST** do so by ensuring that the original (factory programmed) values for these three bits (0~2) are preserved at all time.

Bit 4: HI_OSC_FLT_ENA – High Frequency Oscillator Low Noise Filter Enable
0 = Disables 30kHz Low Noise Filter (Default)
1 = Enables 30kHz Low Noise Filter

Bit 3: LO_OSC_DIS – Low Frequency Oscillator Disable
0 = Enables Low Frequency Oscillator Clock (Default)
1 = Disables Low Frequency Oscillator Clock

Bits 2-0: RESERVED – These three bits (0~2) **MUST NOT** be changed at any time. If the user wants to reconfigure the other two bits (3~4), then he **MUST** do so by ensuring that the original (factory programmed) values for these three bits (0~2) are preserved at all time.

8 Appendix 1 – C program which emulates compensator

The program below emulates the numerical engine in the compensator. This can be used to evaluate potential control functions without using the chip itself. It is also useful as a description of the details of the compensator, such as scaling, truncation and rounding.

```
//=====
// Fixed point multiply
//  c is the filter coefficient
//  e is the variable to be multiplied
//  Scale is the number of bits that are dropped after the multiple
//=====
inline long UCD9200_CLA::fixedMult(long c, long e, const Scale)
```

```

{
    e *= c;      // multiply by coefficient
    c = (e & (1<<(Scale-1)) ) >> (Scale-1); // get MSB of bits to be shifted off
    return ((e >> Scale) + c); // shift and add r to round value
}

//=====
// Clamp (Saturate) a value at Dlimit.max or Dlimit.min
//=====
inline long UCD9200_CLA::clamp(long val)
{
    if( val > Dlimit.max) // feedback clamp
        val = Dlimit.max;
    if( val < Dlimit.min)
        val = Dlimit.min;
    return val;
}

//=====
//Simulate the Direct Mode CLA implemented in the ALR device
// e[n] contains the error voltage states
// y[n] contains the filter output states
// B1[n] contains the fixed point 2nd order numerator coefficients
// A1[n] contains the fixed point 2nd order denominator coefficients
// B2[n] contains the cascaded 1st order numerator coefficients
// A2[n] contains the cascaded 1st order denominator coefficients
//=====
int UCD9200_CLA::CLA_fixed_ALR(int err)
{
    long    tempx, tempy, w, r; // temp variables used to do the
    int     duty_out;          // difference equation

    e[2] = e[1]; // store Verror values for next time
    e[1] = e[0];
    e[0] = err; // get latest ADC reading

    tempx = fixedMult(B1[0], e[0], 3);
    tempx += fixedMult(B1[1], e[1], 3);
    tempx += fixedMult(B1[2], e[2], 3);

    tempy = fixedMult(A1[1], y[0], 8);
    tempy += fixedMult(A1[2], y[1], 8);

    w = (tempx + tempy) << Kscale; // scale up by K
    r = (w & (1<<(3-1)) ) >> (3-1);
    w = (w >> 3) + r; // drop bottom three bits and round

    y[1] = y[0];
    y[0] = clamp(w);

    //=====
    // cascaded 1st order filter section
    //=====
    tempy = y[0] << 3;

```

```
tempy += fixedMult(B2[1], y[1], 8);  
tempy += fixedMult(A2[1], yp, 8);  
r = (tempy & (1<<(3-1)) ) >> (3-1);  
w = (tempy >> 3) + r; // drop bottom three bits and round  
  
yp = clamp(w);  
  
if( yp < 0) // saturate at 0 before applying to PWM  
    duty_out = 0;  
else  
    duty_out = (int)yp;  
  
return duty_out;  
}
```

REVISION CONTROL

1/13/2010 – Section 7.6 on Clock Control Register 2 (CLKCNTL) added by Shamim Choudhury