Freescale Semiconductor
Application Note

Document Number: AN4485
Rev. 0, 04/2011

Using eFlexPWM with MC56F82xx DSC

by: Richy Ye
Applications Engineering
Shanghai

1 Introduction

The enhanced flexible pulse width modulator (eFlexPWM) module contains PWM submodules, each of which is set up to control a single half-bridge power stage. Fault channel support is provided. The eFlexPWM module can generate various PWM modes, including highly sophisticated waveforms, which can be used to control all known motor types and is ideal for controlling different switching mode power supply (SMPS) topologies as well.

The eFlexPWM module has these main features:

- 16 bits of resolution for center-, edge-aligned, and asymmetrical PWMs
- Fractional delay for enhanced resolution of the PWM period and duty cycle
- PWM outputs that can operate as complementary pairs or as independent channels
- Ability to accept signed numbers for PWM generation

Contents

- Independent control of both rising and falling edges of each PWM output
- Support for synchronization to external hardware or other PWMs
- Double-buffered PWM registers
  - Integral reload rate 1–16
  - Half-cycle reload capability
- Multiple output trigger events can be generated per PWM cycle via hardware
- Support for double switching PWM outputs
- Fault inputs can be assigned to control multiple PWM outputs
- Programmable filters for fault inputs
- Independently programmable PWM output polarity
- Independent top and bottom hardware deadtime insertion
- Each complementary pair can operate with its own PWM frequency and deadtime values
- Individual software control for each PWM output
- All outputs can be programmed to change status simultaneously via a FORCE_OUT event
- PWMX pin can optionally output a third PWM signal from each submodule (please refer to the pin assignment table of the respective device to see which pin is used for this purpose)
- Channels not used for PWM generation can be used for buffered output compare functions
- Channels not used for PWM generation can be used for input capture functions with enhanced dual-edge capture capability (please refer to information in respective chapter of reference manual to see which submodules include this function)
- Option to supply the source for each complementary PWM signal pair from:
  - Crossbar module outputs
  - External ADC input, taking into account values set in ADC high- and low-limit registers

This document includes a module introduction, block diagram, module functions, and design considerations.

2 Block diagram

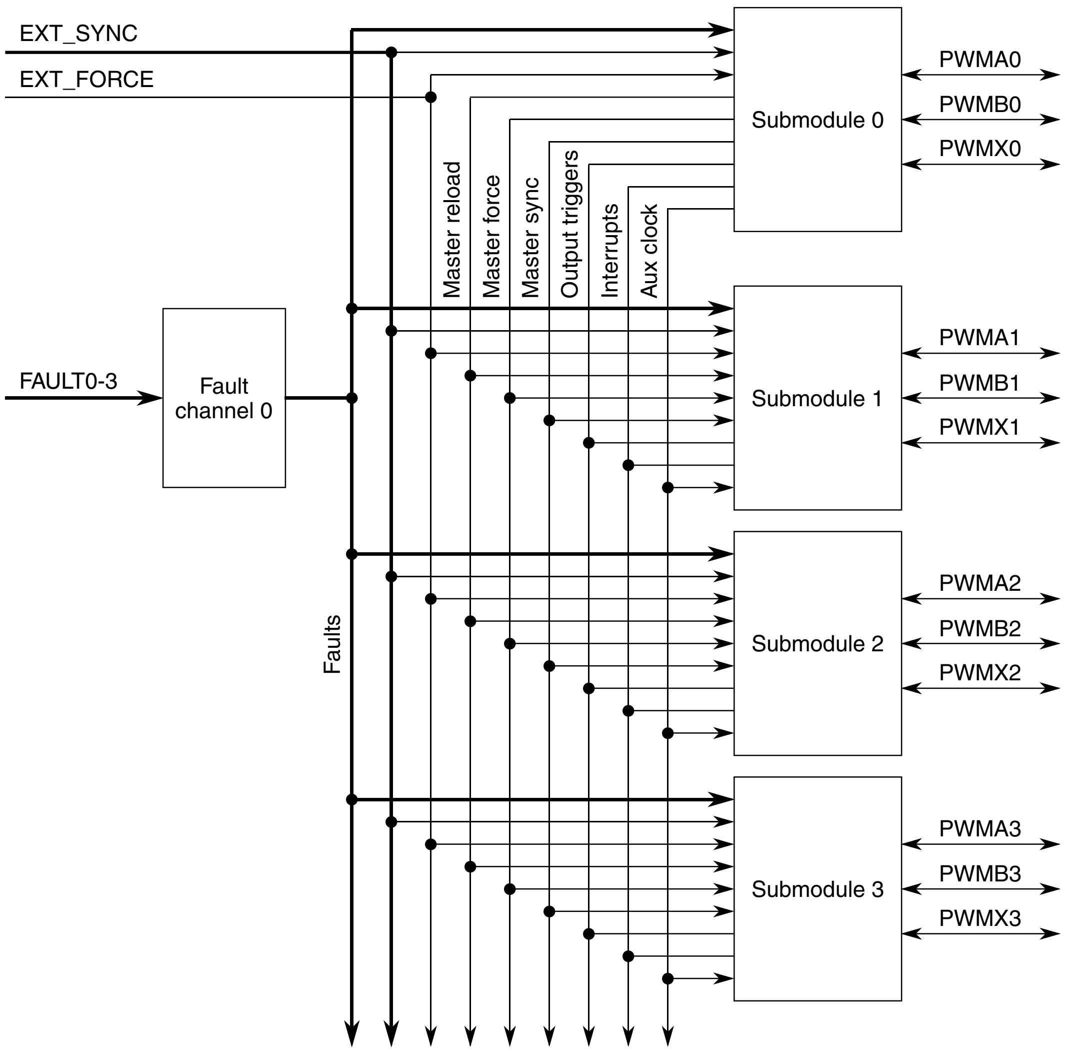

Figure 1. Block diagram of PWM module

The MC56F82xx's eFlexPWM module contains four submodules (each submodule has its own time base) and has one fault channel, fault channel zero, which accommodates four distinct fault inputs. Each FAULTx pin can be mapped arbitrarily to control any combination of the PWM outputs.

By default, submodule 0 is regarded as the master for internal synchronization control. Control signals MASTER RELOAD, MASTER FORCE, MASTER SYNC, and AUX CLOCK are output only by submodule 0, and received by other submodules (1/2/3) or external components. Alternatively, these four submodules can be controlled and synchronized together by external signals EXT_SYNC, EXT_FORCE, and EXT_CLK. In addition, each submodule can generate independent output trigger signals to trigger events in other components, and an interrupt signal for a CPU interrupt response.

Table 1 identifies PWM feature support that varies by submodule (SM).

Table 1. PWM submodule feature support

Submodule	Fractional delay (high resolution)	Enhanced input capture
SM0, SM1, SM2	Yes	No
SM3	No	Yes (1-level FIFO depth)

Please note that only the auxiliary PWM signal of submodule3, PWMX3, is enabled to pinout in the MC56F82xx. Also, signals FAULT[n], PWM[n]_EXT_SYNC, EXT_FORCE, PWM[n]_EXTA, PWM[n]_EXTB, EXT_CLK, and PWM[n]_OUT_TRIGx, where *n* can be 0, 1, 2 or 3 and *x* can be 0 or 1, cannot be found from the pin description table in the data sheet, but may be configured for a wide range of intermodule connections via the crossbar module.

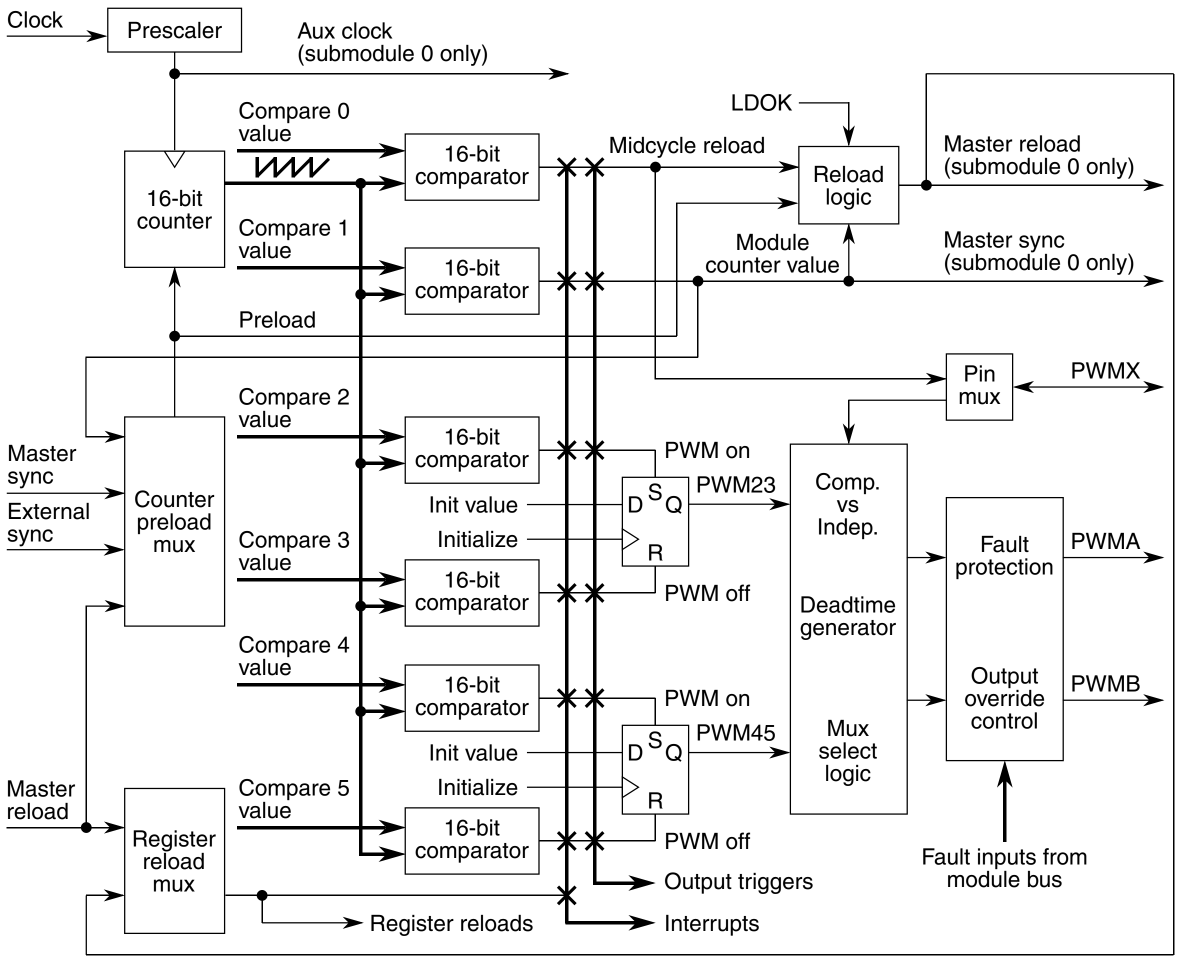

Figure 2. Block diagram of submodule

Figure 2 specifies the PWM submodule details. In each case, two comparators and associated VALx registers are utilized for each PWM output signal. One comparator and VALx register are used to control the turn-on edge, while a second comparator and VALx register control the turn-off edge. The generation of the local sync signal is performed exactly the same way as the other PWM signals in the submodule. While comparator 0 causes a rising edge of the local sync signal, comparator 1 generates a falling edge. Comparator 1 is also hardwired to the reload logic to generate the full cycle reload indicator.

If VAL1 is controlling the modulus of the counter and VAL0 is half of the VAL1 register minus the INIT value, then the half-cycle reload pulse will occur exactly halfway through the timer count period and the local sync will have a 50% duty cycle. On the other hand, if the VAL1 and VAL0 registers are not required for register reloading or counter initialization, they can be used to modulate the duty cycle of the local sync signal, effectively turning it into an auxiliary PWM signal (PWMX), assuming that the PWMX pin is not being used for another function such as input capture or deadtime distortion correction. Including the local sync signal, each submodule is capable of generating three PWM signals where software has complete control over each edge of each of the signals.

If the comparators and edge value registers are not required for PWM generation, they can also be used for other functions such as output compares, generating output triggers, or generating interrupts at timed intervals. The 16-bit comparators shown in Figure 2 are “equal to or greater than,” not just “equal to,” comparators. In addition, if both the set and reset of the flip-flop are asserted, then the flop output goes to 0.

3 Module functions

This section describes the implementation of various functions in the eFlexPWM module in detail, including PWM capability, ADC trigger, enhanced capture, PWM synchronization, and fault protection.

3.1 PWM capability

MC56F82xx's eFlexPWM module contains four submodules, each of which has its own time base and PWM capability. They can work independently of each other or in synchronization. In this section only one submodule is used as an example.

3.1.1 Edge-aligned PWM

An edge-aligned PWM provides a single PWM signal where one out of two edges of the PWM is aligned to the period boundary, and the other edge is determined by the duty cycle. As Figure 3 shows, this allows maximum duty cycle resolution using an edge-aligned PWM, because the PWM duty cycle can be changed per PWM clock.

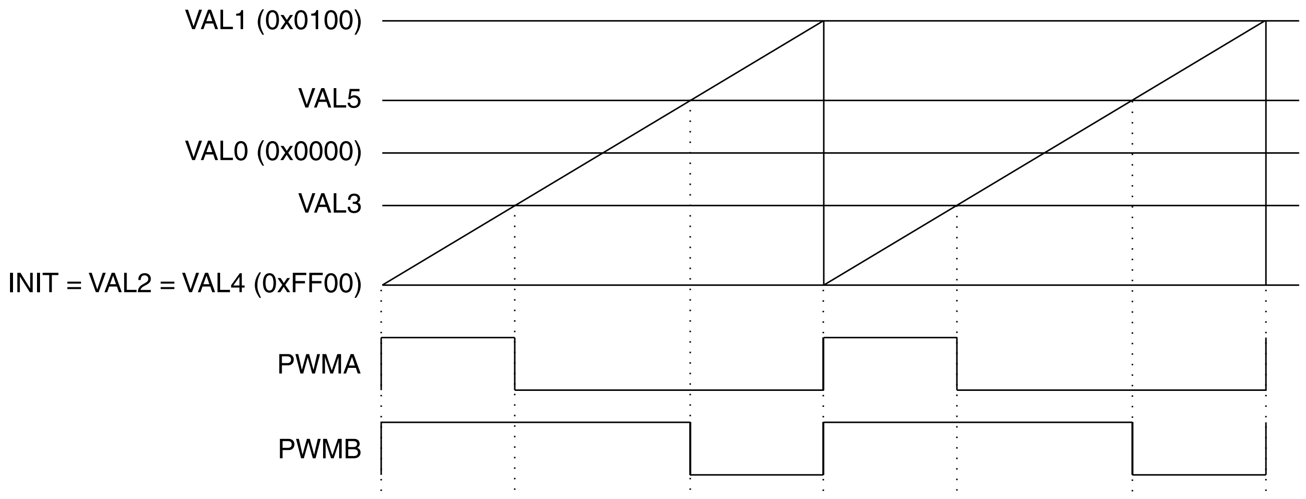

Figure 3. Edge-aligned PWM example (INIT = VAL2 = VAL4)

For an edge-aligned PWM on MC56F82xx, the turn-on edge value (VAL2, VAL4) for each pulse is specified to be the INIT value. Therefore only the turn-off edge value (VAL3, VAL5) needs to be updated to change the duty cycle. The code for an edge-aligned PWM can be found in the software available with this application note.

The edge-aligned PWM is usually used for single power switch converters, such as BUCK, BOOST, and Flyback converters, etc.

3.1.2 Center-aligned PWM

Center-aligned PWM provides a single PWM signal where half of the PWM period appears before a center point, and the other half after the center point. As Figure 4 shows, it will reduce 1-bit duty cycle resolution, because the PWM duty cycle can only be changed twice per PWM clock.

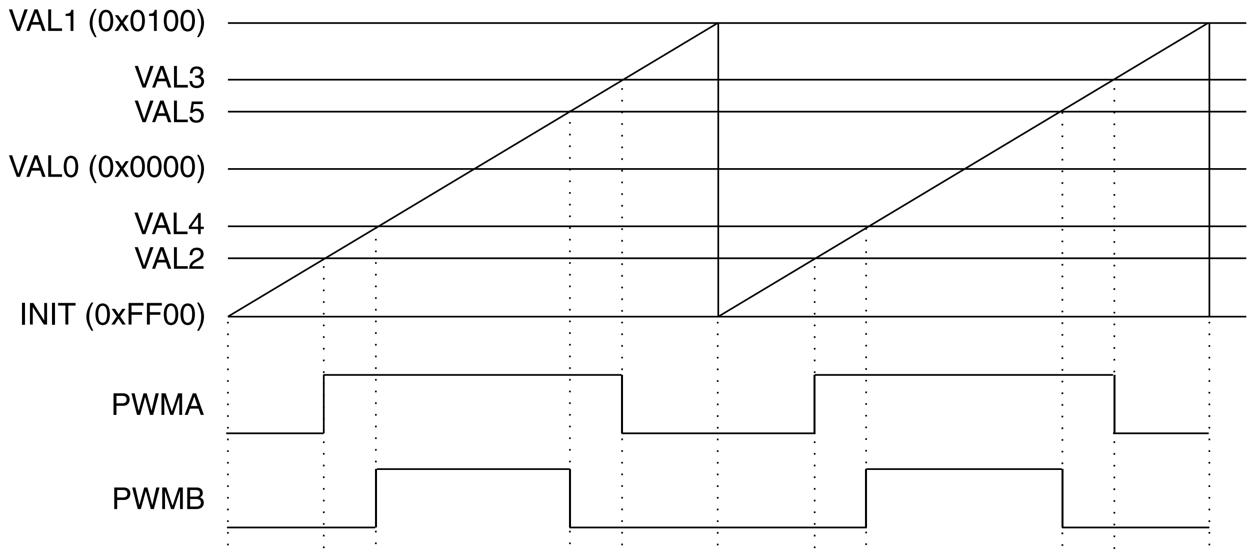

Figure 4. Center-aligned PWM example

For a center-aligned PWM on MC56F82xx, the center point is specified to be half the value of the sum of the INIT value and VAL1. Therefore the turn-on edge value (VAL2, VAL4) for each pulse needs to be updated so that the difference between center point and turn-on edge is half of the duty cycle. Then the turn-off edge value (VAL3, VAL5) also needs to be updated so that the difference between turn-off edge and center point is half of the duty cycle.

If all PWM signal edge calculations follow this same convention, then the signals will be center-aligned with respect to each other. Of course, center alignment between the signals is not restricted to symmetry around the zero count value, as any other number would also work. However, centering on zero provides the greatest range in signed mode and also simplifies the calculations. The code for center-aligned PWM can be found in the software available with this application note.

The center-aligned PWM is usually used for multiple power switches or multi-phase converters, such as half-bridge and full-bridge inverter, etc. Using it will greatly improve system EMI and THD.

3.1.3 Complementary PWM

Complementary PWM provides a PWM output signal on one pin with the complement of the PWM signal on the other output pin (thus providing a pair of complementary channels). As Figure 5 shows, complementary operation allows use of the deadtime insertion feature to avoid shooting through.

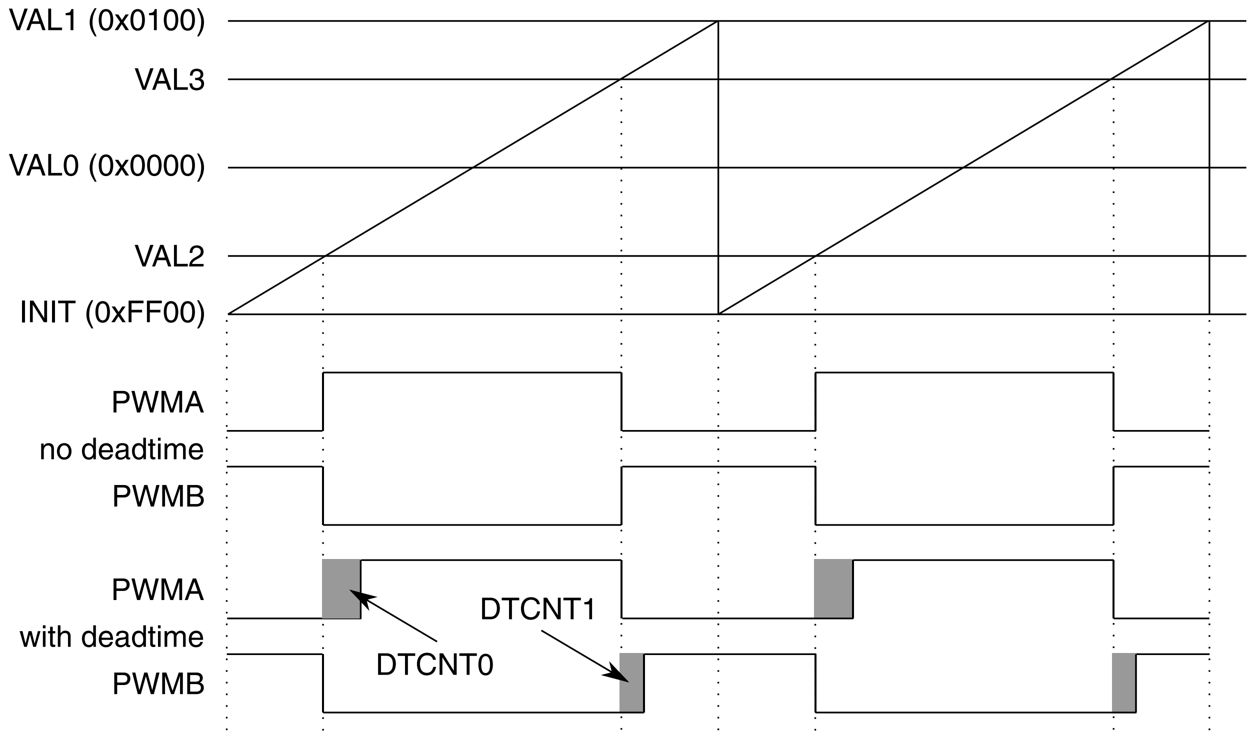

Figure 5. Complementary PWM example (with deadtime insertion)

For complementary PWM on MC56F82xx, writing a logic zero to CTRL2[INDEP] configures the PWM output as a pair of complementary channels. The PWM pins are paired between PWMxA and PWMxB in complementary channel operation. MCTRL[IPOL] determines which signal is connected to the output pin (PWM23 or PWM45). The code for complementary PWM can be found in the software available with this application note.

Complementary PWM is usually used for converters with a bridge leg, such as synchronously switched BUCK, half-bridge, and full-bridge converters, etc.

3.1.4 Push-pull PWM

Push-pull PWM means that a standard PWM output signal goes out on one pin, then the PWM signal (same duty cycle) goes to the other output pin on the next cycle, then the process repeats. The cycle is shown in Figure 6.

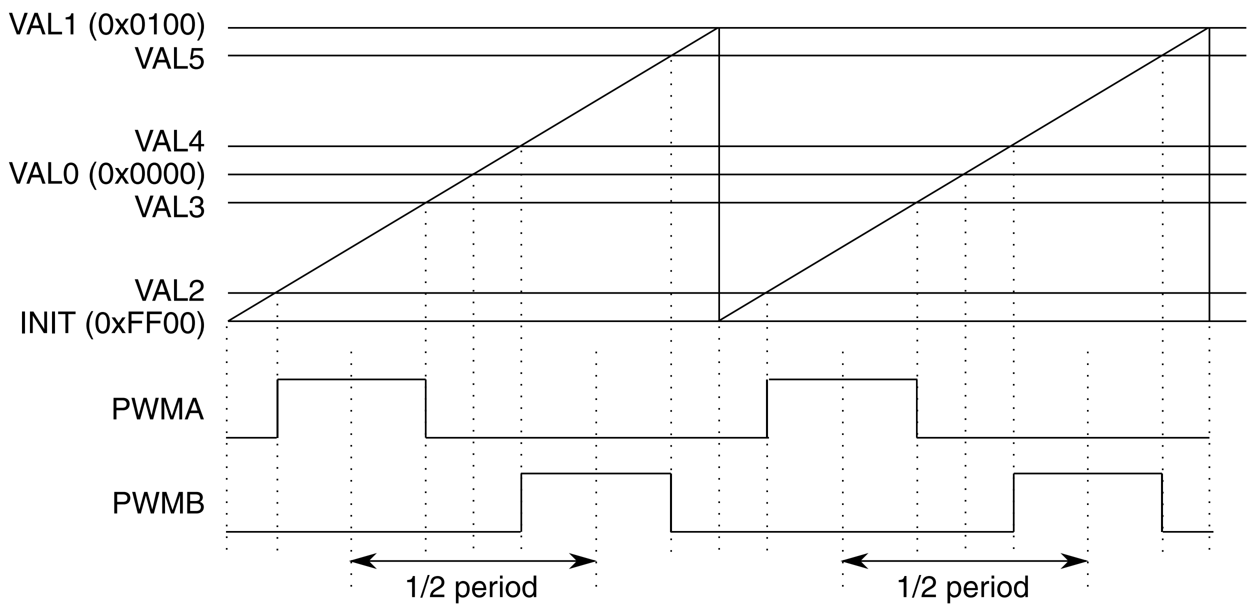

Figure 6. Push-pull PWM example

When using push-pull PWM on MC56F82xx, writing a logic one to CTRL2[INDEP] configures the PWM output as independent channels. The active PWM signal of PWMxA should lie in the former half of the PWM cycle, so that inequality is assured: initial value (INIT) ≤ turn-on edge value (VAL2) ≤ turn-off edge value (VAL3) ≤ half period point value ((VAL1+INIT)/2). Similarly, in order to output the active PWM signal of PWMxB in the second half of the PWM cycle, then the values must be:

> half-period point value ((VAL1+INIT)/2) ≤ turn-on edge value (VAL4) ≤ turn-off edge value (VAL5) ≤ period value (VAL1)

Also, the offset must be kept as half of the PWM period for two turn-on edges (VAL2 and VAL4) and two turn-off edges (VAL3 and VAL5) separately. The code for push-pull PWM can be found in the software available with this application note.

The push-pull PWM is usually used for DC to AC inverters with isolated transformer, such as push-pull, half-bridge, and full-bridge converters, etc.

3.1.5 Multi-phase PWM

Multi-phase PWM allows multiple PWM generators to output PWM signals that are synchronized but constantly shift phase relative to each other, as shown in Figure 7.

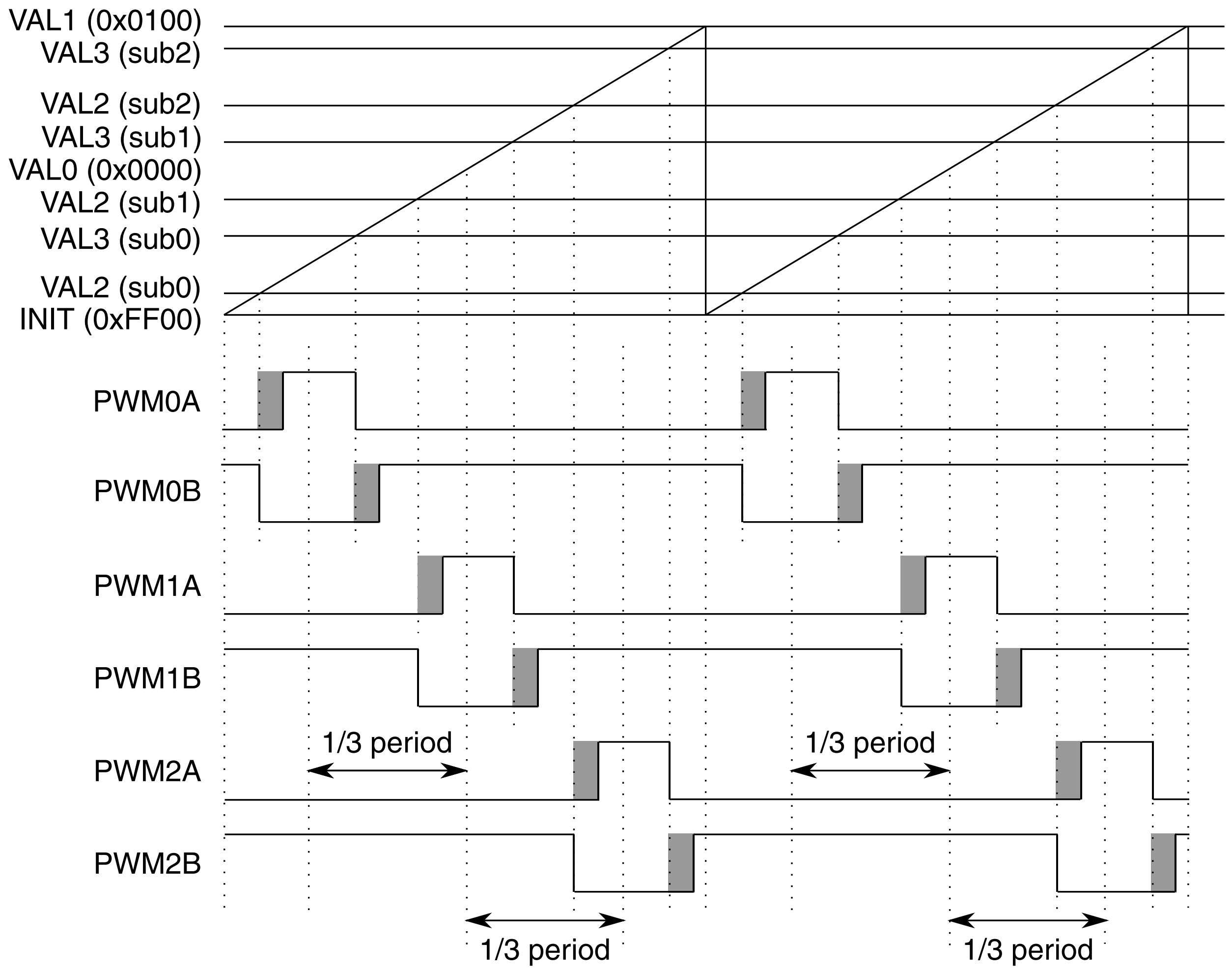

Figure 7. Multi-phase PWM example

When using multi-phase PWM on MC56F82xx, in order to keep time base synchronization among the different phases of the PWM signals, the same initialization control signal must be selected by all PWM submodules. This means that if the master submodule (submodule 0) selects the local sync signal as its initialization control signal, the slave submodules (submodules 1/2/3) should select the master signal (master reload or sync signal) to initialize their counters, or else all submodules will all select the same external sync signal as their initialization control signal. Also, it is important that the phase offsets be set up to evenly distribute the phase shift among the different phases of the PWM signals.

As shown in Figure 7, there are three pairs of PWM signals, so the phase offset among the three pairs of PWM signals should be set to one third of the PWM period,. This means that there is one-third of PWM period phase difference among center point of each pair PWM signals ((PWM2_VAL3+PWM2_VAL2)/2 – (PWM1_VAL3+PWM1_VAL2)/2 = (PWM1_VAL3+PWM1_VAL2)/2 – (PWM0_VAL3+PWM0_VAL2)/2 = (VAL1-INIT)/3) if center-aligned PWM mode is used. If edge-aligned PWM mode is used, there is one-third of PWM period phase difference among edge (turn-on edge or turn-off edge) of each pair of PWM signals (PWM2_VAL3(VAL2) – PWM1_VAL3(VAL2) = PWM1_VAL3(VAL2) – PWM0_VAL3(VAL2) = (VAL1-INIT)/3). The code for multi-phase PWM can be found in the software available with this application note.

Multi-phase PWM is usually used for multi-phase converters with two, three, four, or more phases, such as interleaved BUCK, interleaved BOOST and interleaved flyback converters, etc.

3.1.6 Phase-shifted PWM

Phase-shifted PWM is similar to multi-phase PWM, but the phase relationships are constantly changing. In general, phase-shifted PWM is always set to 50% duty cycle, as shown in Figure 8.

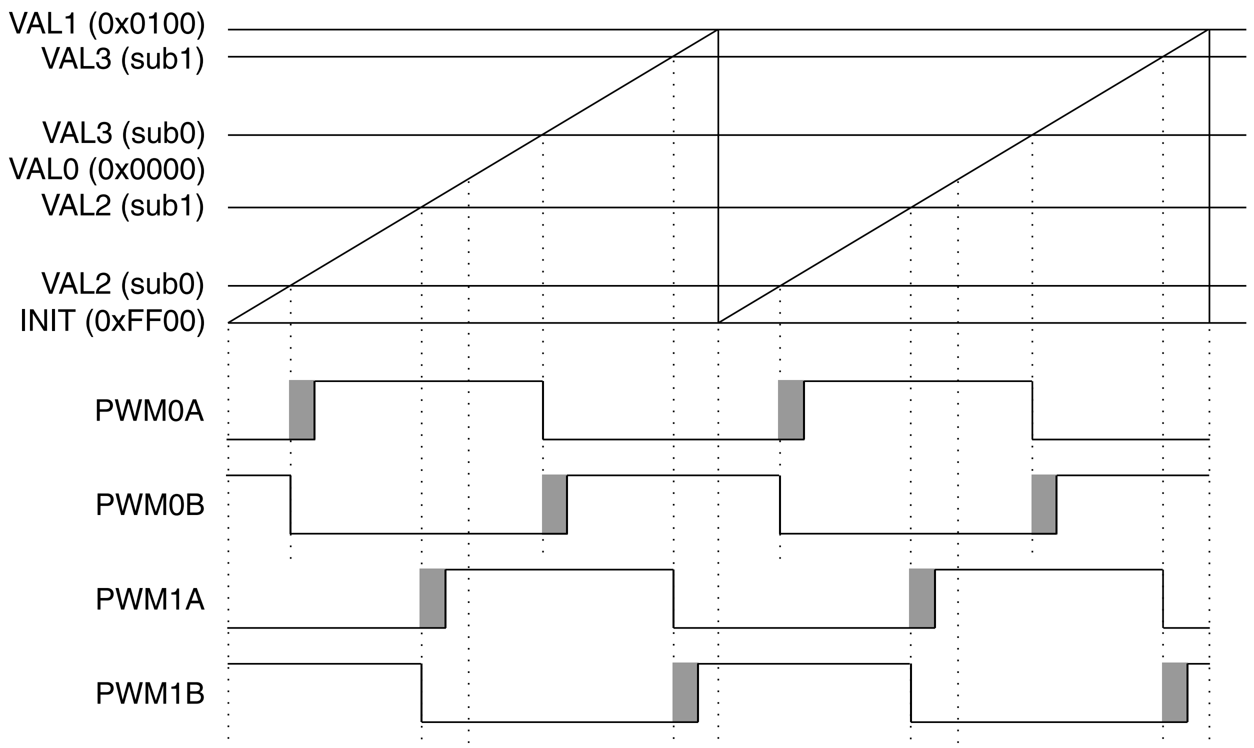

Figure 8. Phase-shifted PWM example

For phase-shifted PWM on MC56F82xx, in order to keep time base synchronization among different phases PWM signals, the same initialization control signal must be selected by all PWM submodules. This means that if the master submodule (submodule 0) selects the local sync signal as its initialization control signal, the slave submodules (submodule 1/2/3) should select the master signal (master reload or sync signal) to initialize their counters, or else all submodules will all select the same external sync signal as their initialization control signal. Also, it is important that the duty cycle of each PWM signal pair be kept unchanged.

As shown in Figure 8, there are two pairs of PWM signals. The first pair of PWM signals (PWM0A, PWM0B) are kept fixed, and the phase relationship between the two pairs of PWM signals is changed only by changing the phase of the secondary pair PWM signals (PWM1A, PWM1B). This means that there is an identical phase offset (phase-shifted quantum) between turn-on edges and turn-off edges of each pair of PWM signals (PWM1_VAL2-PWM0_VAL2 = PWM1_VAL3-PWM0_VAL3 = required phase-shifted quantum). The code for phase-shifted PWM can be found in the software available with this application note.

The phase-shifted PWM is usually used for ZVT converters, such as a full-bridge ZVT converter.

3.1.7 Current-limit PWM

Current-limit PWM mode is a variation of standard fixed-frequency PWM where the turn-on time is truncated when the current reaches a desired maximum value, as shown in Figure 9.

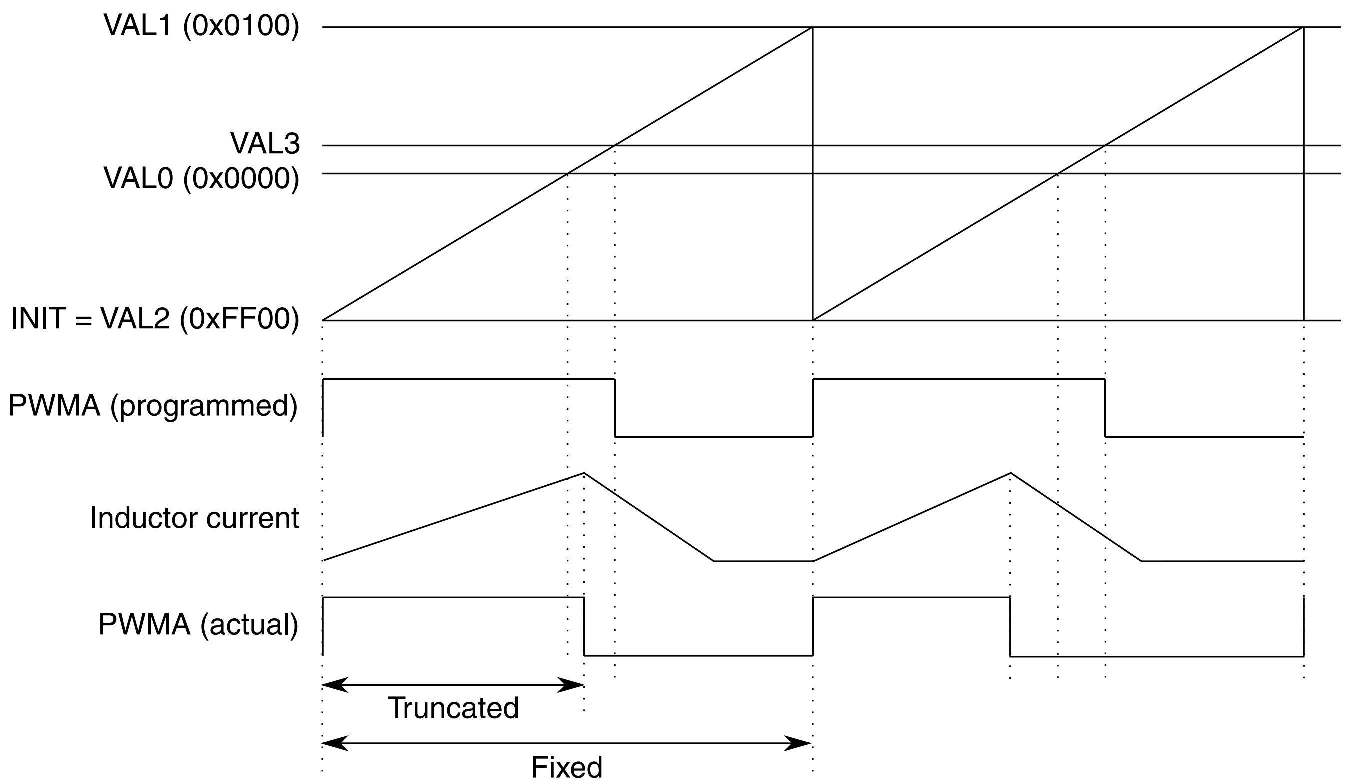

Figure 9. Current-limit PWM example

For current-limit PWM on MC56F82xx, the turn-on edge value (VAL2) is specified to be the INIT value. Then the turn-off edge value (VAL3) is set to a fixed value large enough to meet actual maximum turn-on time requirement. In order to truncate the turn-on time when the current reaches a desired maximum value, an internal comparator can be used to disable the PWM output, where the comparator output will be used as the fault input of the PWM module. Automatic clearing mode with only full cycle clearing timing should be selected for fault clearing mode in the PWM module. The code for current-limit PWM can be found in the software available with this application note.

The current-limit PWM is usually used for a peak-current-controlled converter, such as peak current controlled phase-shifted full-bridge converter, which uses the combination of phase-shifted PWM mode and current-limit PWM mode to drive power MOSFETs. This automatically prevents the output transformer from saturation, which eliminates the need for a capacitor to block DC current. And it should be noted that a subharmonic oscillation issue exists in the control loop when the output inductor current increases to a value so high that it can’t return to its initial value in one PWM cycle. At this moment, an algorithm, called slope compensation, is usually used to correct this issue.

Therefore, a 12-bit DAC on MC56F82xx can be used to implement the slope compensation algorithm using its automatic waveform generation function (down-counting sawtooth waveform). The maximum value of the sawtooth waveform is from the output of the output voltage loop of the phase-shifted full-bridge converter — the minimal value and step will be determined by the desired slope rate, PWM cycle, and DAC updating rate. Then the DAC output is used as the desired maximum value of the output inductor current (if the step is large, an external capacitor can be added on the DAC output pin to smooth DAC output), and connected to the input of an internal comparator, which is used to disable the corresponding PWM output when the actual output inductor current reaches the desired maximum value.

3.1.8 Current-reset PWM

Current-reset PWM mode is a variable frequency mode where the turn-on time is specified, and the turn-off time is truncated when current falls below a desired minimum value, as shown in Figure 10.

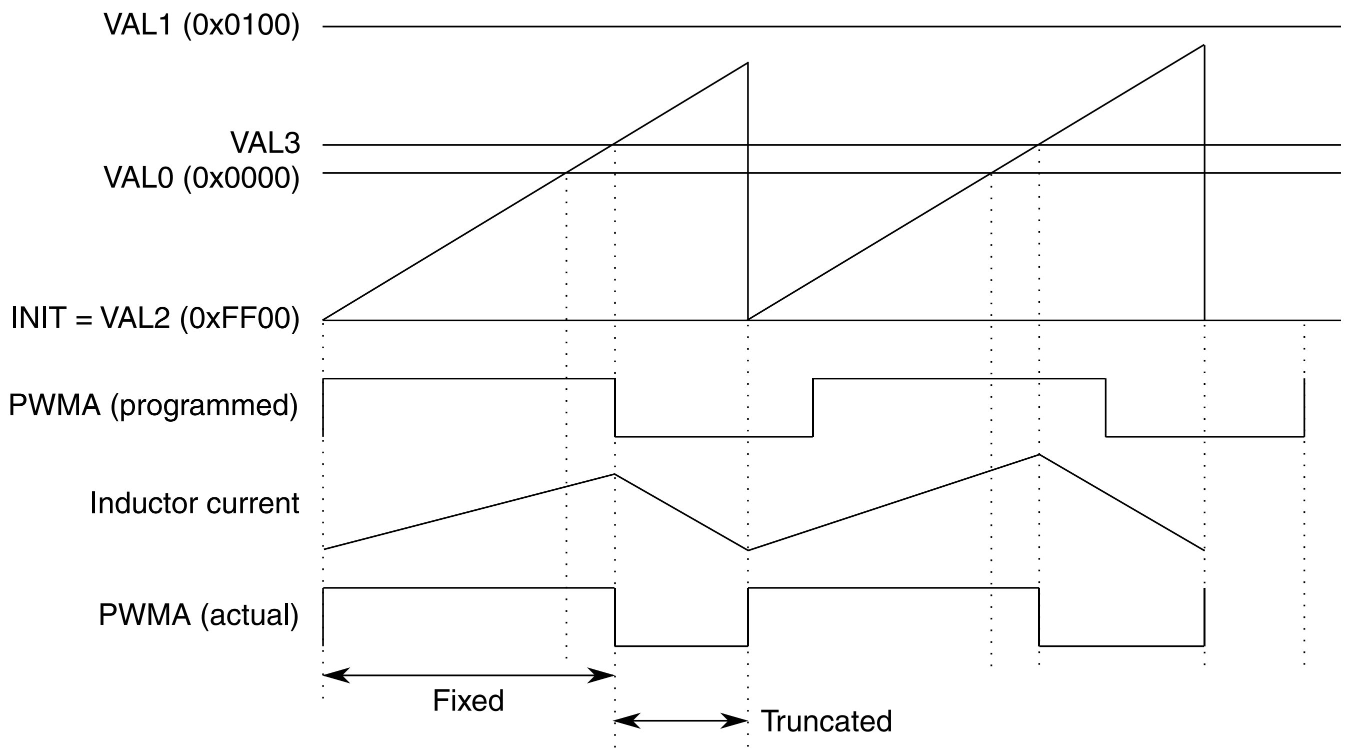

Figure 10. Current-reset PWM example

For current-reset PWM on MC56F82xx, the turn-on edge value (VAL2) is specified to be the INIT value. The turn-off edge value (VAL3) is also specified to assure that the programmed turn-on time is a fixed value. In order to truncate the turn-off time when current falls below a desired minimal value, an internal comparator can be used to initialize the PWM counter. The comparator output will be used as the external sync signal input of the PWM module, and EXT_SYNC should be selected as the initialization control source in the PWM module. The code for current-reset PWM can be found in the software available with this application note.

The current-reset PWM is usually used for transition mode (critical current mode) PFC converters, and will greatly improve converter efficiency in low-power applications. The combination of current-limit

PWM mode and current-reset PWM mode is used in a transition mode PFC converter for better performance.

3.2 PWM synchronization

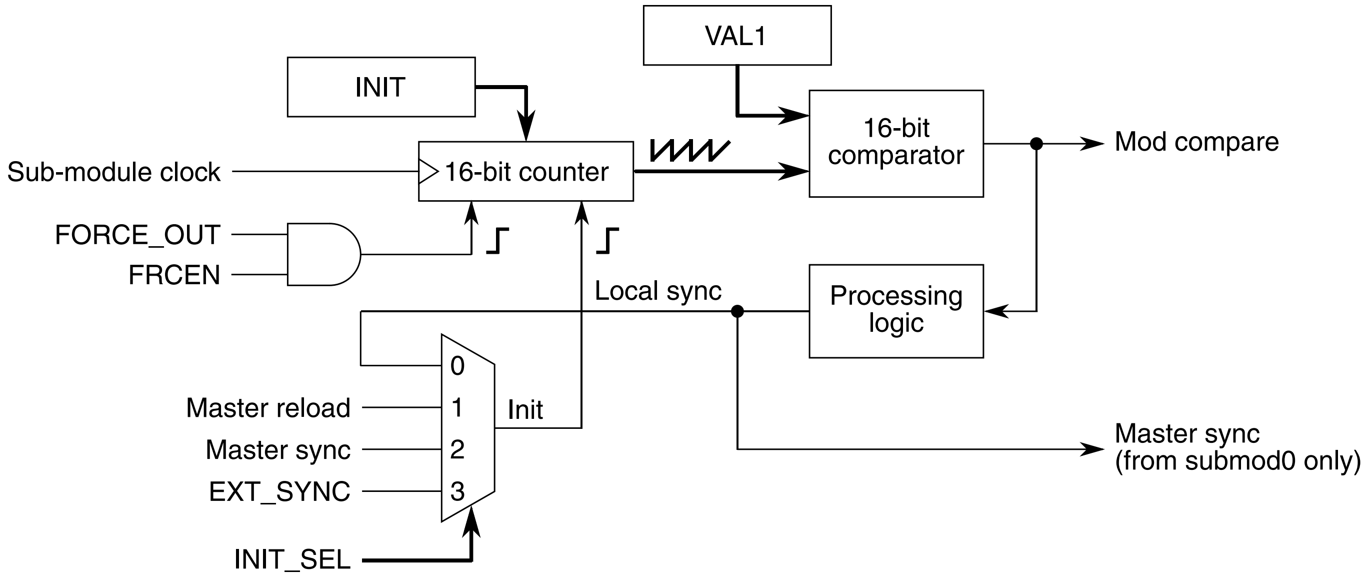

Figure 11. Submodule timer synchronization

The 16-bit counter shown in Figure 11 can be initialized with an INIT value by four possible sources: local sync, master reload, master sync, and EXT_SYNC.

If local sync is selected as the counter initialization signal, the counter will count up until its output equals VAL1. VAL1 is used to specify the counter modulus value, then VAL1 within the submodule effectively controls the timer period (and thus the PWM frequency generated by that submodule) and everything works on a local level. If the master sync signal (which originates as the local sync from submodule0) is configured to initialize the counter, then the counter period of any submodule can be locked to the period of the counter in submodule0. Then the VAL1 register and associated comparator of the other submodules can be freed up for other functions such as PWM generation, input captures, output compares, or output triggers.

If the master reload signal (which only originates from submodule0) is selected as the source for counter initialization, then the period of the counter of any submodule will be locked to the reload frequency of submodule0. Since the reload frequency is optional and can vary from one to sixteen, it will support generating multi-frequency PWM signals in synchronization.

If the EXT_SYNC signal (which originates on-chip or off-chip depending on the system architecture) is selected as the source for counter initialization, the counter period in all submodules can be controlled by an external source. This makes it easy to implement synchronization between the eFlexPWM module and the external signal.

In addition, the counter can optionally be initialized upon the assertion of the FORCE_OUT signal (which is provided mainly for synchronous switching of multiple PWM outputs), assuming that CTRL2[FRCEN] is set. As indicated by the preceding figure, this constitutes a second initialization input into the counter.

This will cause the counter to initialize regardless of which signal is selected as the counter initialization signal.

In summary, if multiple PWMs with the same frequency are required to synchronize, the master sync signal (submodule0 selects local sync for counter initialization) is recommended to initialize the counters in slave submodules (submodule1/2/3) if all PWM signals are from internal submodules. Otherwise, the EXT_SYNC signal is recommended to initialize the counters in all submodules when some PWM signals are from another PWM module or are off-chip. If it is necessary to synchronize multiple PWMs with different frequencies, two methods are recommended.

- Method one:

1. Configure the highest frequency PWM signal from submodule0 (which uses local sync to initialize the counter for internal PWM synchronization, or EXT_SYNC for external PWM synchronization).
2. Select master reload (reload rate depends on the frequency ratio of different PWMs) signal for counter initialization in slave submodules (submodule1/2/3).

- Method two:

1. Select local sync to initialize counters for those submodules which generate the higher frequency PWM signals.
2. Configure their FORCE source as master sync signal (submodule0 is configured to generate lowest frequency PWMs) or EXT_FORCE signal (other slave submodule is configured to generate lowest frequency PWMs — this submodule is required to output a local sync trigger as an EXT_FORCE signal input for other submodules).
3. Configure the submodule (which generates lowest frequency PWMs) to select local sync to initialize the counter for internal PWM synchronization, or EXT_SYNC for external PWM synchronization.

3.3 Fractional delay logic

For applications requiring greater resolution than a single IPBus clock period (up to 16.67 ns), the fractional delay logic can achieve fine resolution on the rising and falling edges of the PWM outputs. Use the PWM_SMn_FRCTRL register (where *n* is 0, 1, or 2) to enable the fractional delay logic only where needed. The FRACVALx registers (highest five bits are used) act as a fractional clock cycle (IPBus clock period/2^5, up to 520.8 ps) addition to the turn-on and turn-off count specified by the VAL2, VAL3, VAL4, or VAL5 registers. The FRACVAL1 register acts as a fractional increase in the PWM period as defined by VAL1.

For example, assume the following conditions:

INIT = 0x00

VAL1 = 0x0F

VAL2 = 0x00

VAL3 = 0x07

FRACVAL3 = 0x00

these settings would cause the PWM output to have a 50% duty cycle: it would be high from a count value of 0x00 (VAL2) to 0x07 (VAL3), at which point it would go low until the counter reaches the maximum value of 0x0F (VAL1). If FRACVAL3 were set to a value of 0xB800 (real value is 0x17), then the time the PWM output is active would be:

8 cycles + (23/32) cycles = 8.719 cycles

for a high duty cycle of:

(8.719 cycles/16 cycles) × 100% = 54.49%

Another case involves fine tuning the PWM period using the FRACVAL1 register. If a PWM period of 100.25 clock cycles is required, program VAL1 with 0x64 and FRACVAL1 with 0x4000. In this case, the fractional value accumulates so that every four PWM cycles are one clock cycle longer (101 instead of 100). The rising and falling edges of the PWM outputs also use the accumulated fraction to delay their edges and maintain a consistent spacing of 100.25 cycles between corresponding edges, from one cycle to the next.

In order to enable a high-resolution PWM function (fractional delay logic), two independent registers (VALx and FRACVALx, where *x* is 1, 2, 3, 4, or 5) must be programmed to get the desired duty cycle and period. An example can be found, in the software available with this application note, of variable duty cycle and period PWM with high-resolution function, developed to show how to simply set the two registers.

3.4 Fault protection

Fault protection can control any combination of PWM output pins. Faults are generated by a logic one on any of the FAULTx pins. This polarity can be changed via FCTRL[FLVL]. Each FAULTx pin can be mapped arbitrarily to any of the PWM outputs. When fault protection hardware disables PWM outputs, the PWM generator continues to run, but the output pins are forced to logic 0, logic 1, or tri-state depending on the value of SMnOCTRL[PWMxFS].

3.4.1 Automatic fault clearing

When FCTRL[FAUTOx] is set, disabled PWM pins are enabled when the FAULTx pin returns to logic zero and a new PWM full or half-cycle begins. See Figure 12 — if FSTS[FFULLx] is set, then the disabled PWM pins are enabled only at the start of a full cycle and not at the half-cycle. Clearing FSTS[FFLAGx] does not affect disabled PWM pins when FCTRL[FAUTOx] is set. This mode is very useful for cycle-by-cycle protection, and has quicker response in half-cycle resolution, which will help improve system dynamic response.

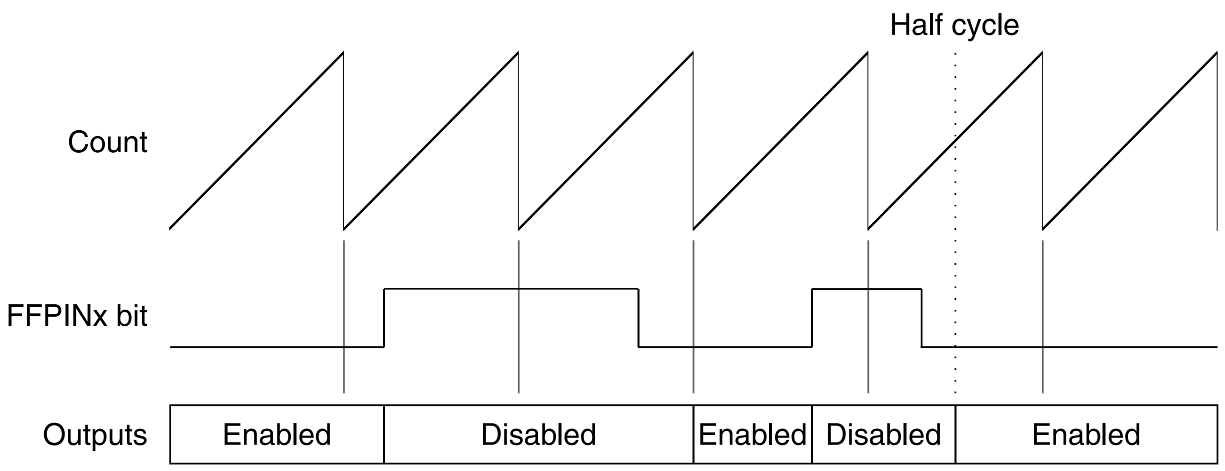

Figure 12. Automatic fault clearing example

3.4.2 Manual fault clearing

Clearing the automatic clearing mode bit, FCTRL[FAUTOx], configures faults from the FAULTx pin for manual clearing:

- If the fault safety mode bits, FCTRL[FSAFEx], are clear, then PWM pins disabled by the FAULTx pins are enabled when:
  - Software clears the corresponding FSTS[FFLAGx] flag.
  - The next PWM full or half cycle begins, regardless of the logic level detected by the filter at the FAULTx pin. See Figure 13. If FSTS[FFULLx] is set, then the disabled PWM pins are enabled only at the start of a full cycle and not at the half-cycle.

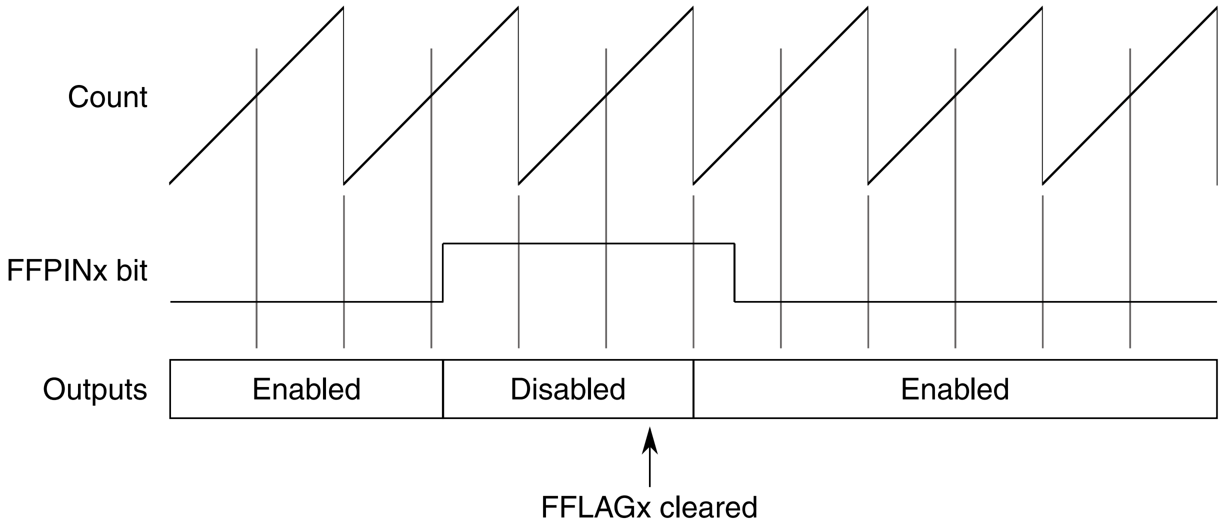

Figure 13. Manual fault clearing example (FCTRL[FSAFEx] = 0)

- If the fault safety mode bits, FCTRL[FSAFEx], are set, then PWM pins disabled by the FAULTx pins are enabled when:
  - •Software clears the corresponding FSTS[FFLAGx] flag.

— The filter detects a logic zero on the FAULTx pin at the start of the next PWM full or half-cycle boundary (see Figure 14). If FSTS[FFULLx] is set, then the disabled PWM pins are enabled only at the start of a full cycle and not at the half-cycle.

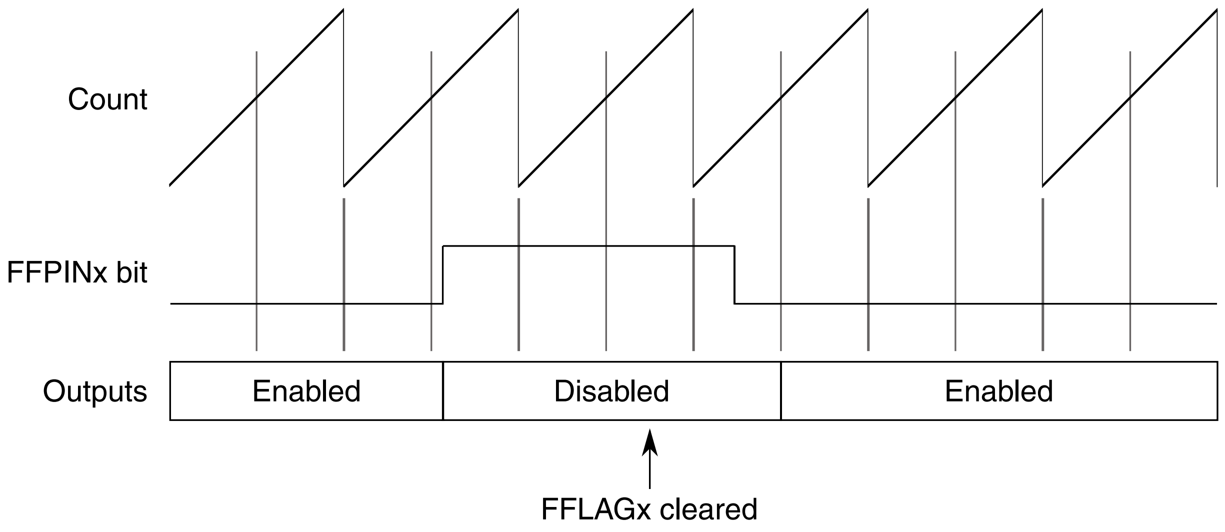

Figure 14. Manual fault clearing example (FCTRL[FSAFEx] = 1)

This mode is very useful for latch-up protection, and has flexible enable mode and quick response in half-cycle resolution, which will help to improve system design flexibility and dynamic response.

3.5 Enhanced capture capability

When eFlexPWM submodule3 pins (PWM3A, PWM3B, PWM3X) aren't being used for PWM generation, they can be used to perform the input capture function. Commensurate with the idea of controlling both edges of an output signal, the enhanced capture (E-capture) logic is designed to measure both edges of an input signal. As a result, when a submodule pin is configured for input capture, the CVALx/x+1 registers associated with that pin are used to record the edge values.

Figure 15 is a block diagram of the E-capture circuit. Upon entering the pin input, the signal is split into two paths: one goes straight to a mux input where software can pass the signal directly to the capture logic for processing. The other path connects the signal to an 8-bit counter which counts both the rising and falling edges of the input signal. The output of this counter is compared to an 8-bit value that is specified by the user (EDGCMPx) and when the two values are equal, the comparator generates a pulse that resets the counter. This pulse is also supplied to the mux input where software can select it to be processed by the capture logic. This feature allows the module to count a specified number of edge events and then perform a capture and interrupt.

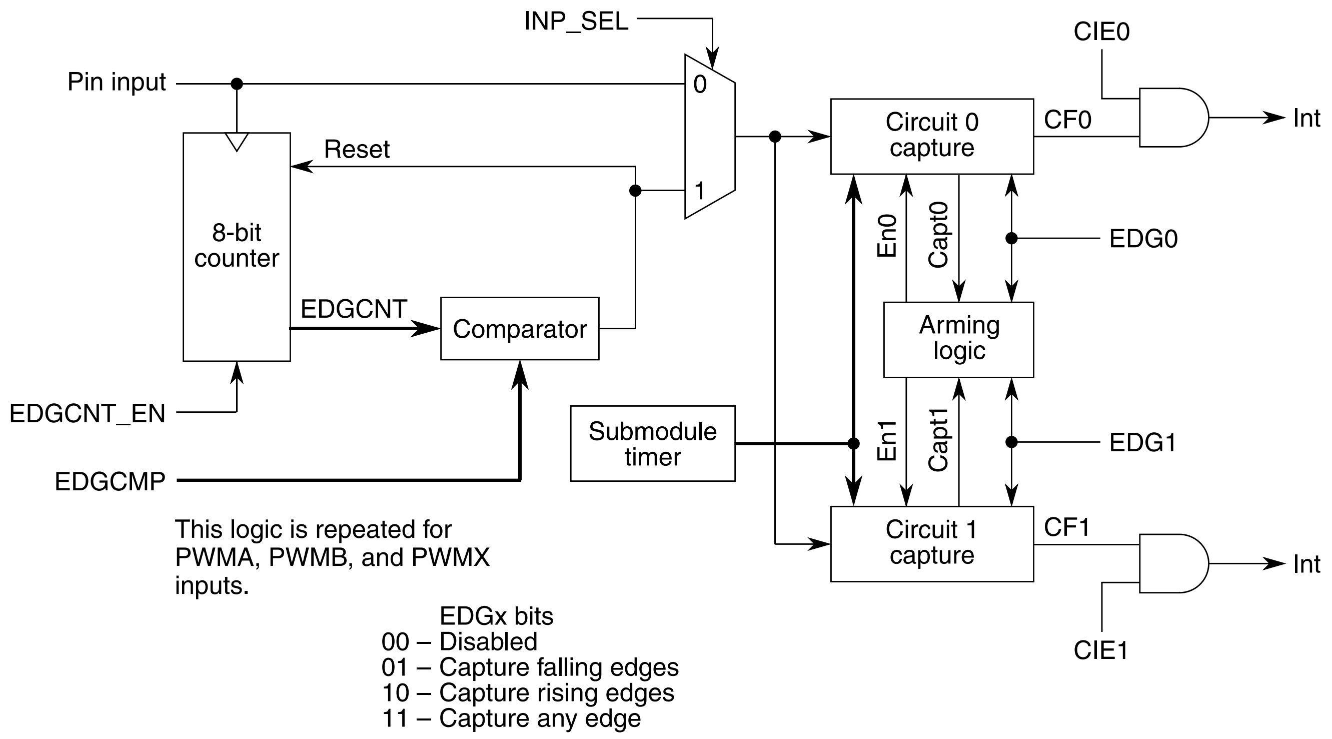

Figure 15. Block diagram of enhanced capture unit

Based on the mode selection, the mux selects either the pin input or the comparator output from the counter/compare circuit to be processed by the capture logic. The selected signal is routed to two separate capture circuits which work in tandem to capture sequential edges of the signal. The type of edge to be captured by each circuit is determined by CAPTCTRLx[EDGx1] and CAPTCTRLx[EDGx0].

Also, controlling the operation of the capture circuits is the arming logic, which allows captures to be performed in a free-running (continuous) or one-shot fashion. In free-running mode, the capture sequences will be performed indefinitely. If both capture circuits are enabled, they will work together in a ping-pong style where a capture event from one circuit leads to the arming of the other and vice-versa. In one-shot mode, only one capture sequence will be performed. Similarly, if only capture circuits are enabled, the capture event will occur on the enabled capture circuit. Both capture circuits are capable of generating an interrupt to the CPU. The duty cycle meter code for E-capture function on PWM3X can be found in the software available with this application note.

3.6 ADC triggering

In cases where timing of ADC triggering is critical, scheduling must be done as a hardware event instead of by software. With the eFlexPWM module, multiple ADC triggers can be generated by the compare output capability in hardware per PWM period without the need of another timer module. Figure 16 shows how this is accomplished in a submodule. For example, when specifying complementary PWM mode operation, only two edge comparators are required to generate the output PWM signals for a given submodule. This means that the other comparators are free to perform other functions, such as a compare output for an ADC trigger event. In this example, the software does not need to quickly respond after the first triggering to set up other triggerings that must occur in the same PWM period.

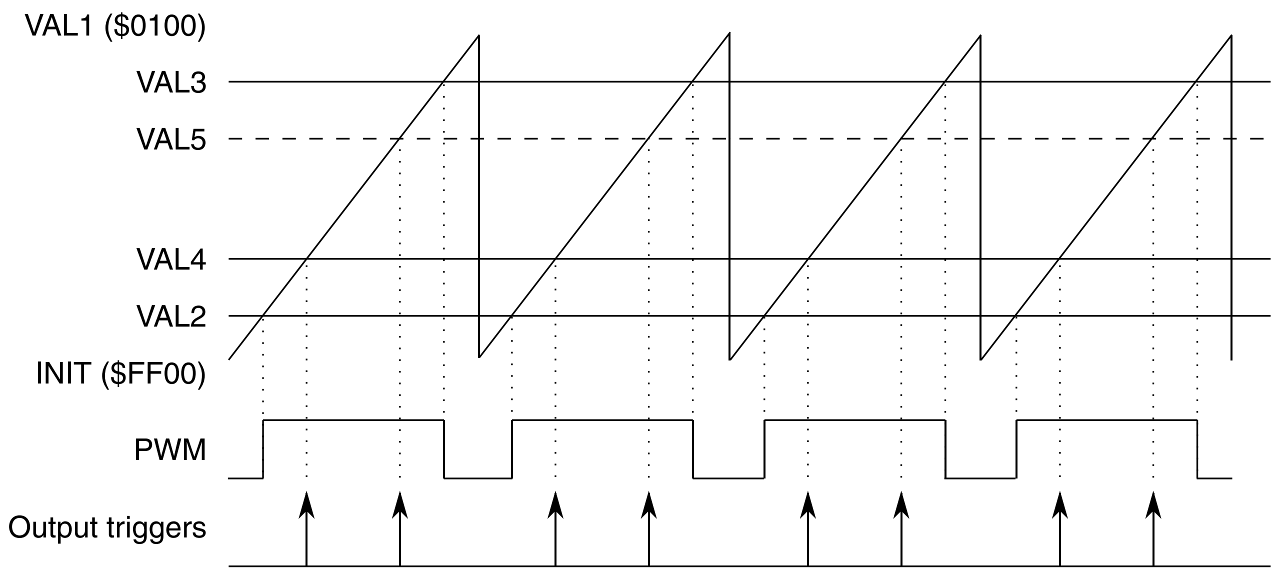

Figure 16. Multiple output trigger generation in PWM period

Because each submodule has its own timer, it is possible for each submodule to run at a different frequency. One of the possible options with the eFlexPWM module is to have one or more submodules running at a lower frequency only to trigger outputs, but still synchronized to those timers in other submodules, which are used to generate PWM signals. Figure 17 shows how this feature can be used to schedule ADC triggers over multiple PWM periods. A suggested use for this configuration would be to use the lower-frequency submodule to control the sampling points where multiple ADC triggers can now be scheduled over the entire sampling period. In this figure, all submodule comparators are shown being used for ADC trigger generation.

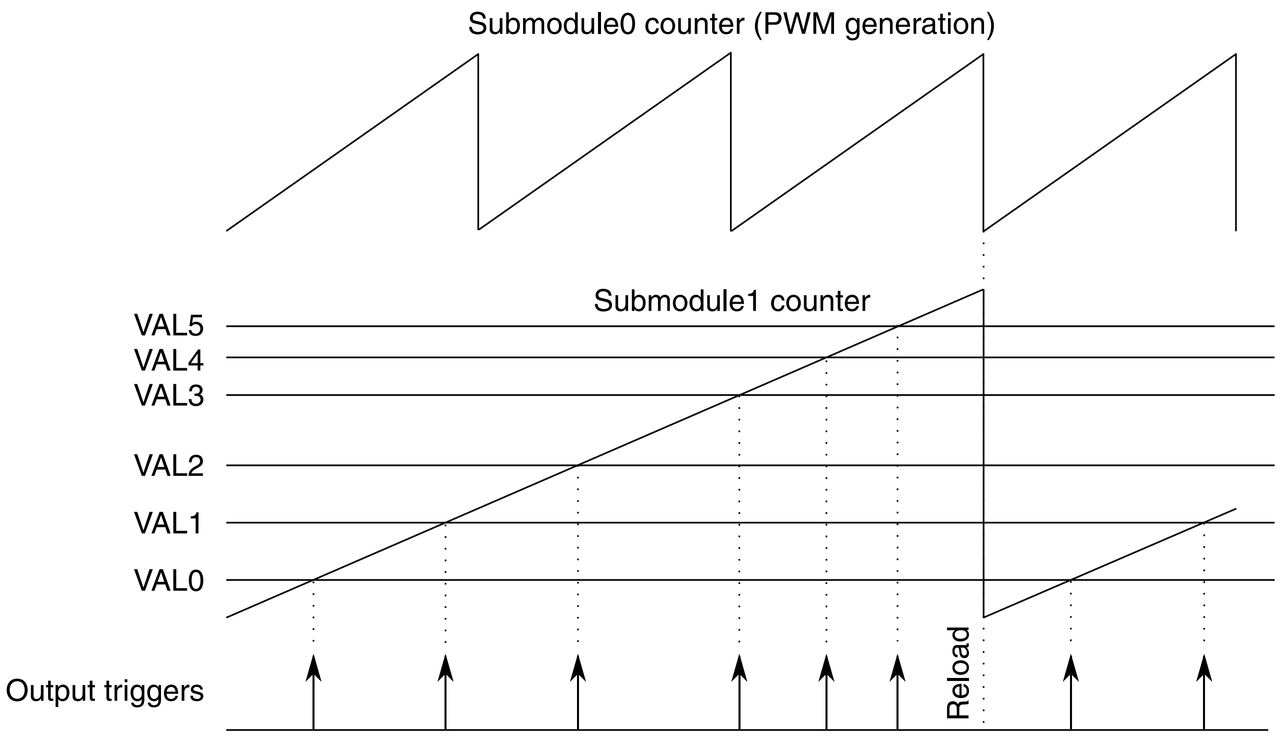

Figure 17. Multiple output trigger generation over several PWM periods

4 Design considerations

Use the following list of considerations to ensure correct operation of the eFlexPWM module on MC56F82xx:

- The PWM output pins are set to input ports with an internal weak pullup resistor (about 2.0 V) after reset, or by default it is required to connect a strong external pulldown or pullup resistor (1–10 kΩ) close to the pin to ensure safety status under uncertain conditions.
- There are two methods to manually disable PWM outputs regardless of duty cycle and clock settings:
  - Disable PWM output by clearing corresponding bit of PWM_OUTEN, which will result in tri-state output on the PWM pin.
  - Disable PWM output by setting corresponding bit of PWM_MASK followed by a FORCE_OUT event, which will result in logic zero output prior to output polarity on the PWM pin.
- It is recommended to meet these boundary conditions for correct PWM generation:

  INIT ≤ VAL2 (VAL4), VAL3 (VAL5) ≤ VAL1
- When variable frequency PWMs are required, it is recommended to keep INIT constant and change frequency only through changing VAL1.
- When enabling high-resolution PWM (fractional delay logic) function:
  - If automatically generating initialization code from Processor Expert in CodeWarrior v8.3, there will be a wrong value assignment for the PWM_SMnFRCTRL register. The correct value

needs to be set according to the bit definition of the PWM_SMnFRCTRL register in the reference manual. Otherwise incorrect PWM behavior may result.

— In order to assure correct high-resolution PWM generation, at least three integer PWM clock (up to 60 MHz) cycle duration (except for deadtime) is required between the edge value (VAL2, VAL3, VAL4, and VAL5) and the period boundary value (INIT and VAL1).

— If complementary high-resolution PWM mode (high resolution enabled for both duty cycle and period) is specified, besides enabling fractional delay logic for the complementary PWM channel, these conditions must also be met for correct PWM generation:

PWM_SMnFRACVAL2 = PWM_SMnFRACVAL5

PWM_SMnFRACVAL3 = PWM_SMnFRACVAL4

— If complementary high-resolution PWM mode is specified, and the inverted generated PWM is selected as the source for deadtime generation, swap the programming for the turn-on edge VAL2/FRACVAL2 (VAL4/FRACVAL4) and turn-off edge VAL3/FRACVAL3 (VAL5/FRACVAL5) registers to ensure correct PWM output.

5 Conclusion

This note summarizes the structure and functions of the eFlexPWM module on MC56F82xx DSC, and provides example code to help better understand implementation of the functions, making it easy for users to properly use this module in their projects. Finally, some design considerations are shared with users to help them better use the module.

The example project was developed on the MC56F8257 with CodeWarrior v8.3 IDE. The macro of MODE_SEL is designed to select different function routines in the project.

Document Number: AN4485
Rev. 0
04/2011