

# Модуль ePWM (ШИМ)

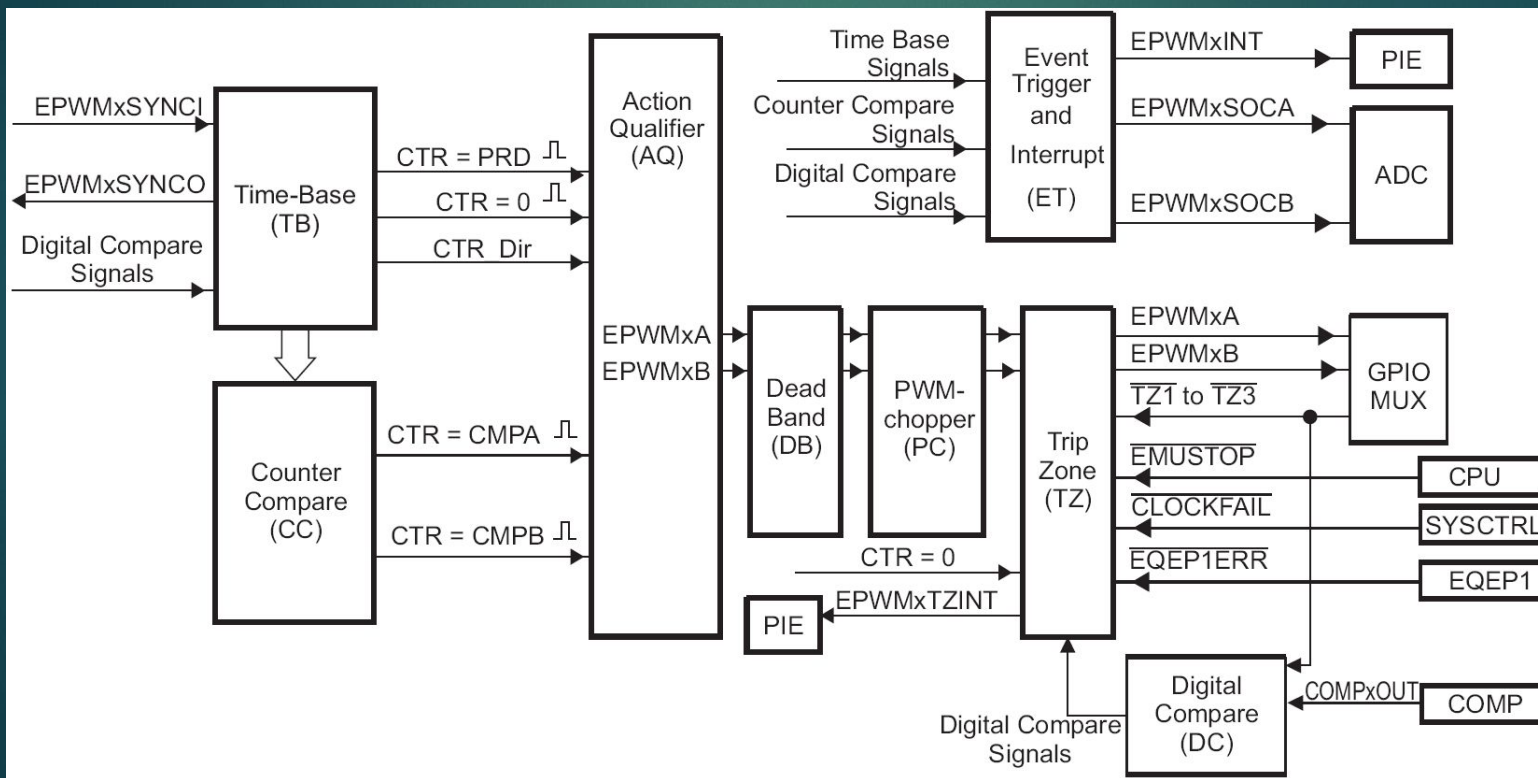
Назначение, внутренняя  
архитектура и возможности

# Назначение модуля ШИМ

- ✓ Модуль ePWM (Enhanced Pulse-Width Modulator) позволяет генерировать ШИМ-сигнал сложной формы с минимальным задействованием процессора.
- ✓ ePWM модуль может изменять уровень GPIO процессора в автоматическом режиме, без необходимости вручную задавать уровень сигнала. Каждый модуль может управлять двумя выводами. Всего в микроконтроллере F28035 есть 6 таких модулей.
- ✓ Этот модуль имеет множество гибких настроек, позволяющих задать частоту и форму несущего сигнала, его синхронизацию с внешними сигналами, реакцию на возникновение аварийной ситуации, автоматическое управление двумя выводами в комплиментарном режиме с обеспечением мёртвого времени и множество других опций.
- ✓ Выходами являются сигналы A и B, которые меняют свой уровень с высокого на низкий и наоборот в зависимости от настроек модуля ePWM. При помощи регистров мультиплицирования GPIO (GPxMUX) эти каналы могут быть выведены на разные выводы микроконтроллера.

# Структура модуля ШИМ

Модуль ШИМ имеет сложную структуру и состоит из множества подмодулей, каждый из которых имеет свою функцию: счетчик для создания несущего сигнала, модуль сравнения уставки с несущим сигналом, модуль действия при сравнении, модуль мёртвого времени и другие.



# Настройка модуля ШИМ

Под настройкой модуля ШИМ подразумевается настройка всех или нескольких его подмодулей. Каждый подмодуль имеет по несколько регистров для конфигурации. Все регистры подмодулей объединяются в группу регистров с названием «*EPwmXRegs*», где X означает номер модуля ePWM.

```
void initPWM (void) {
    // Задать способ счёта "сверху вниз"
    EPwm1Regs.TBCTL.bit.CTRMODE = TB_DOWN;
    // Задать период счётчика равным 1000 тактов
    EPwm1Regs.TBPRD = 1000;
    // Отключать канал A при достижении счётчиком нуля
    EPwm1Regs.AQCTLA.bit.ZRO = AQ_CLEAR;
    // Включать канал A при сравнении счётчика с уставкой A
    EPwm1Regs.AQCTLA.bit.CAD = AQ_SET;
    // Запускать АЦП при каждом достижении нуля
    EPwm1Regs.ETSEL.bit.SOCAEN = 1;
    EPwm1Regs.ETSEL.bit.SOCASEL = ET_CTR_ZERO;
    EPwm1Regs.ETPS.bit.SOCAPRD = 1;
}
```

# Подмодуль счётчика

Подмодуль *Time-Base Submodule* выполняет функцию создания несущего сигнала для ШИМ и фактически представляет собой счетчик. Основные регистры для настройки этого подмодуля (в скобках указано значение по умолчанию):

Имя регистра	Размер	Значение
TBPRD (0)	16 бит	Период счётчика
TBCTL.bit.CTRMODE (3)	2 бита	Способ счёта: 0 – счёт вверх, 1 – счёт вниз, 2 – счёт вверх-вниз, 3 – остановлен
TBCTL.bit.HSPCLKDIV (1)*	3 бита	Делитель частоты счёта 1: 0 → div1 = 1, 1 → div1 = 2, 2 → div1 = 4, 3 → div1 = 6...
TBCTL.bit.CLKDIV (0)*	3 бита	Делитель частоты счёта 2: div2 = 2 <sup>CLKDIV</sup>

\*Итоговая частота счёта считается по формуле:  $f = \frac{SYSCLKOUT}{div1 \cdot div2}$ , где *SYSCLKOUT* равен частоте процессора (60 МГц для F28035).

Здесь приведены не все доступные регистры. С полным списком можно ознакомиться в соответствующей документации – [SPRUGE9E](#)

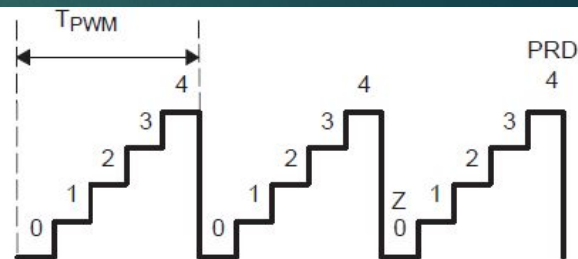
# Подмодуль счётчика

Поведение счётчика при разных настройках регистра `TBCTL.bit.CTRMODE`

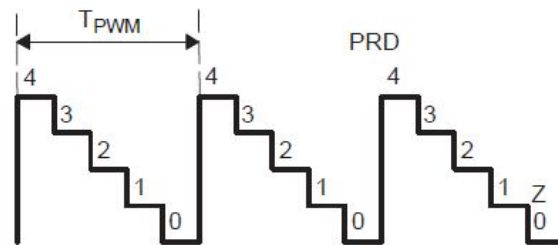
Во всех примерах период равен 4 тактам: `TBPRD = 4`

Бит `TBSTS.bit.CTRDIR` отображает текущее направление счёта:  
1 – счёт вверх, 0 – счёт вниз

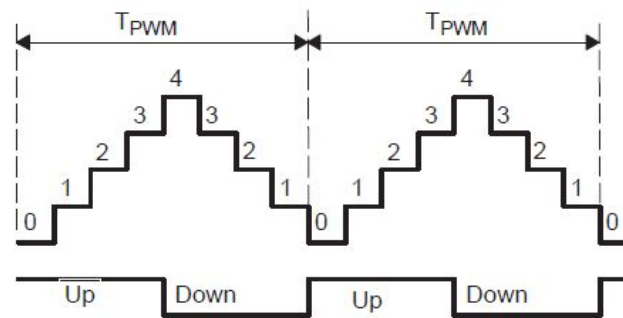
`CTRMODE = 0`  
(счёт вверх)



`CTRMODE = 1`  
(счёт вниз)



`CTRMODE = 2`  
(счёт вверх-вниз)



# Подмодуль сравнения

7

Подмодуль *Counter Compare* непрерывно проверяет текущее значение счётчика и сравнивает его с двумя заданными уставками. Когда счётчик сравнивает с этими уставками, подмодуль сравнения генерирует событие *сравнения*, то есть выдаёт импульс, означающий, что счётчик сравнялся с уставкой. По этим событиям другие подмодули могут предпринимать какие-либо действия (переключение вывода, запуск АЦП и пр.). Основные регистры настройки:

Имя регистра	Размер	Значение
CMRA.half.CMPA (0)	16 бит	Уставка сравнения А
CMRB (0)	16 бит	Уставка сравнения В
CMRCTL.bit.LOADAMODE (0) CMRCTL.bit.LOADBMODE (0)	1 бит	Регистры определяют, в какой момент уставки CMRA и CMRB будут перезагружены: 0 – когда счётчик будет равен 0; 1 – когда счётчик будет равен периоду; 2 – когда счётчик будет равен 0 или периоду; 3 - никогда

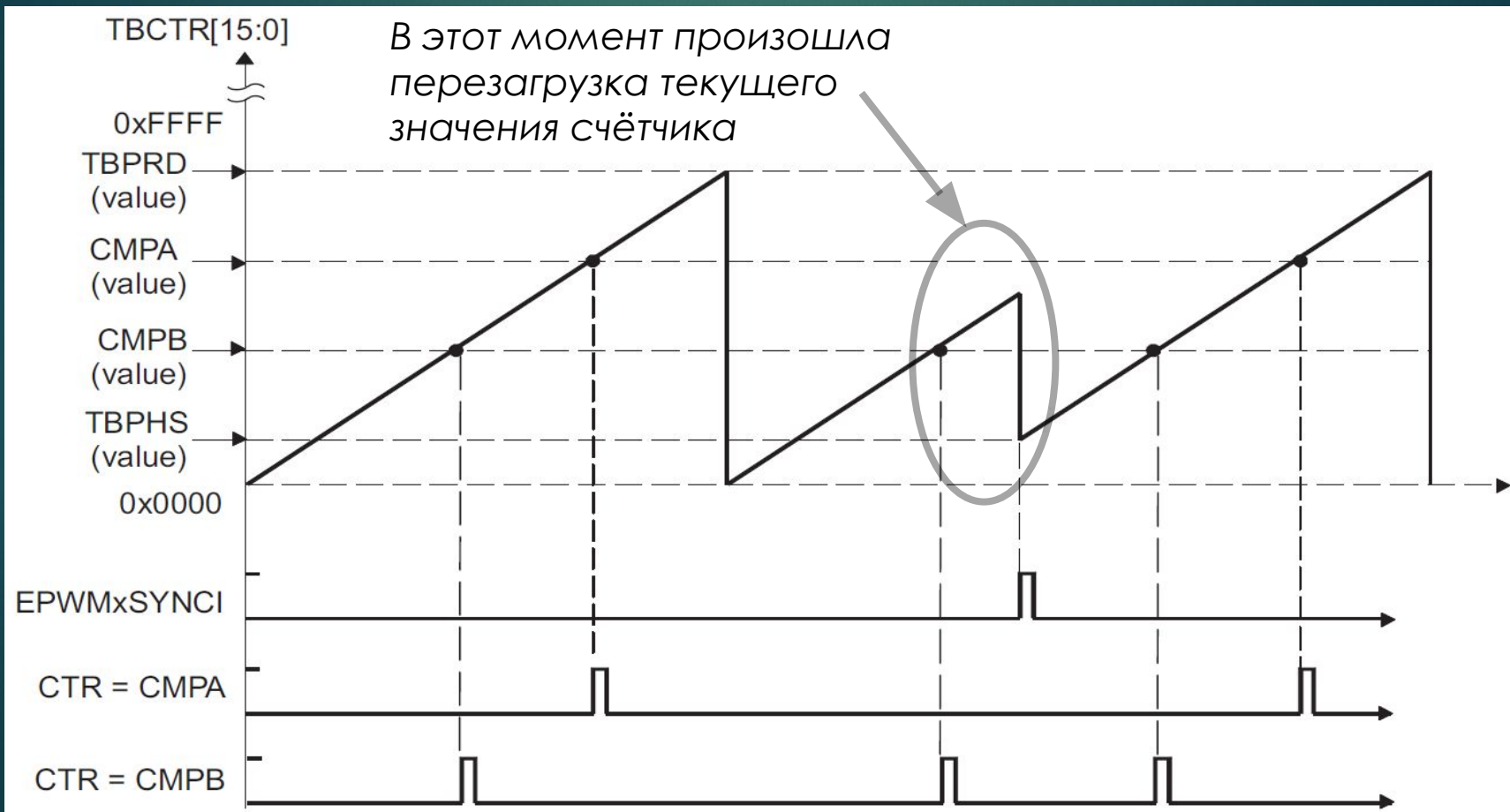
Регистры, хранящие уставки сравнения CMPA и CMPB являются *теневыми*. Это значит, что при записи в них какого-то значения уставки сравнения не обязательно сразу примут записанные значения. Момент фактического применения определяется регистром LOADAMODE/LOADBMODE.

Однако можно настроить подмодуль сравнения так, чтобы уставки сравнения принимали записанное значение сразу же. Об этом можно прочитать в документации.



# Подмодуль сравнения

Данный рисунок демонстрирует работу подмодуля сравнения. Каждый раз, когда значение счётчика совпадает с одной из уставок, генерируется соответствующий импульс.



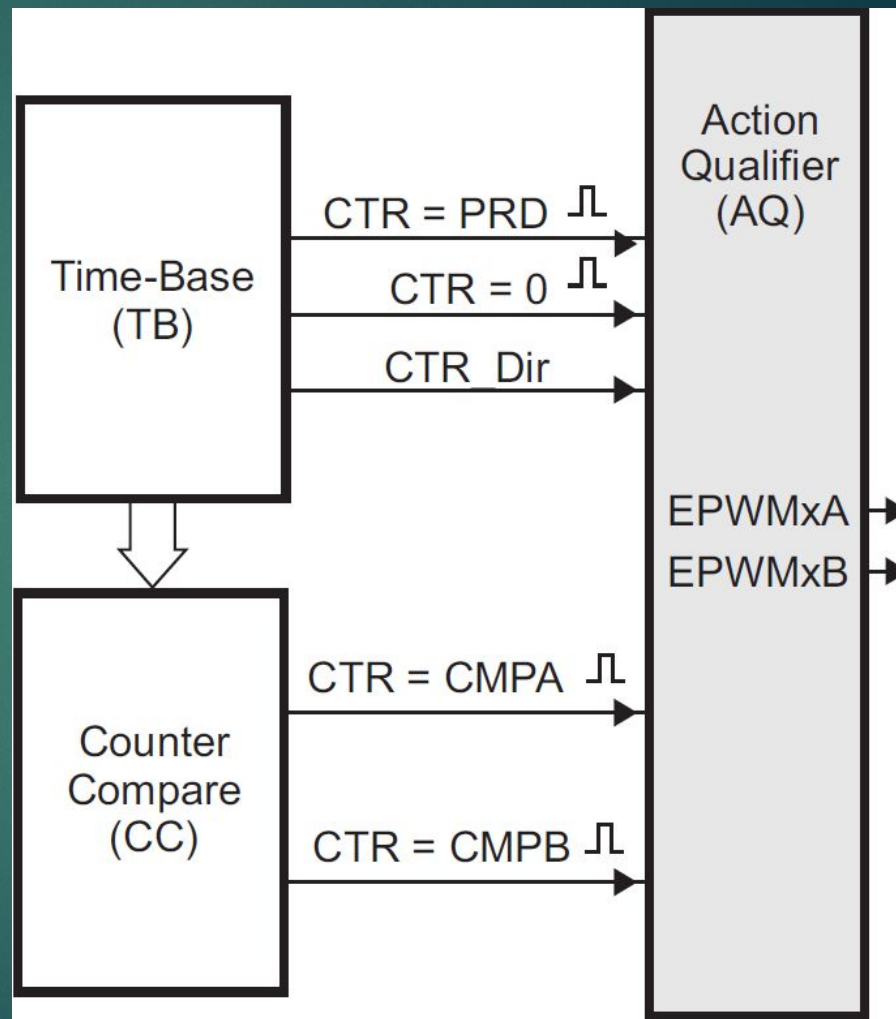
# Подмодуль действия

Подмодуль Action Qualifier играет важнейшую роль в формировании выходного сигнала. Он определяет, какие события в какие действия должны быть преобразованы. Например «Событие «Счётчик равен уставке CMPA» должно установить высокий уровень канала A» и т.п.

События, для которых можно определить действия:

- TBCTR == CMPA
- TBCTR == CMPB
- TBCTR == TBPRD
- TBCTR == 0

Кроме того можно учитывать направление счёта, при котором произошло событие



# Подмодуль действия

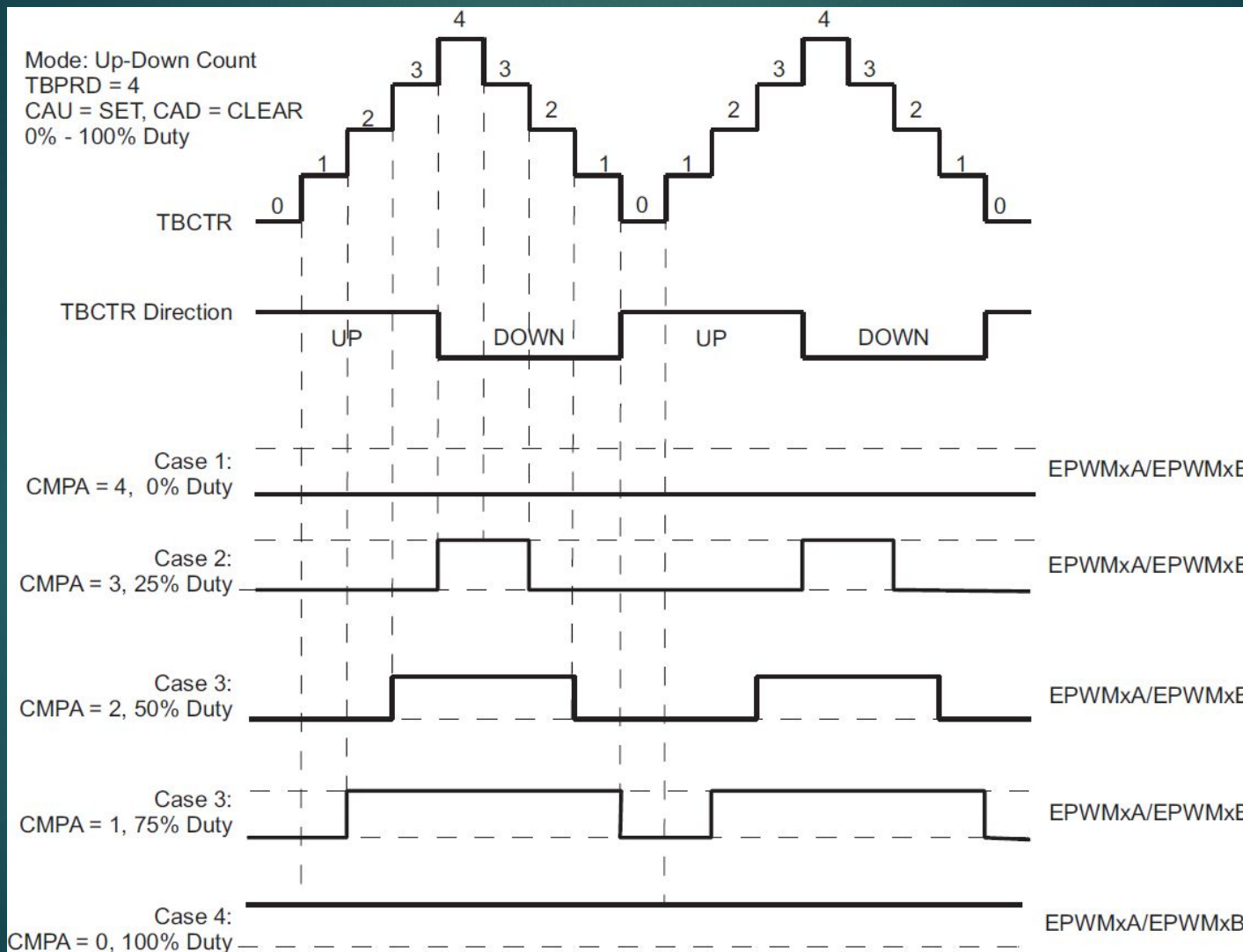
11

Действия, совершаемые с каналами по событиям, настраиваются в регистрах подмодуля. В регистры, перечисленные в таблице, можно записать следующие значения:

**0** – не предпринимать никаких действий, **1** – выдать высокий уровень на канал **2** – выдать низкий уровень на канал **x**, **3** – изменить уровень канала **x** (т.е. если был высокий, то станет низкий, и наоборот)

Для каналов **A** и **B** есть два одинаковых регистра **AQCTLA** и **AQCTLB**, которые определяют действия каналов **A** и **B** соответственно.

Имя регистра	Размер	Значение
AQCTLx.bit.CAU	2 бита	Действие при TBCTR == CMPA, когда счётчик считал вверх
AQCTLx.bit.CAD	2 бита	Действие при TBCTR == CMPA, когда счётчик считал вниз
AQCTLx.bit.CBU	2 бита	Действие при TBCTR == CMPB, когда счётчик считал вверх
AQCTLx.bit.CBD	2 бита	Действие при TBCTR == CMPB, когда счётчик считал вниз
AQCTLx.bit.PRD	2 бита	Действие при TBCTR == TBPRD
AQCTLx.bit.ZRO	2 бита	Действие при TBCTR == 0

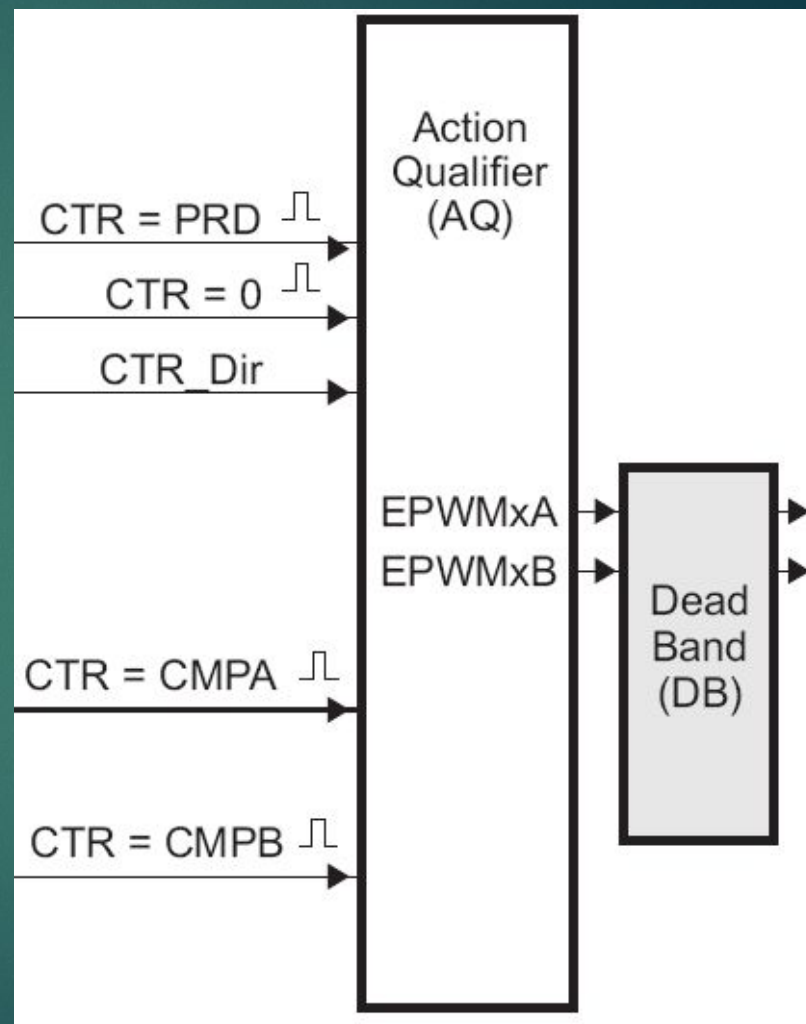


# Подмодуль мёртвого времени

13

Подмодуль *Dead Band* позволяет вносить задержку на включение или выключение каналов А и/или В. Таким образом можно создать паузу между, например, отключением ключа в верхней стойке инвертора и включением верхнего, чтобы избежать протекания сквозных токов в стойке. Подразумевается, что оба ключа управляются каналами А и В одного модуля ePWM.

Кроме того этот модуль позволяет автоматически создавать на одном из каналов сигнал, комплиментарный другому.



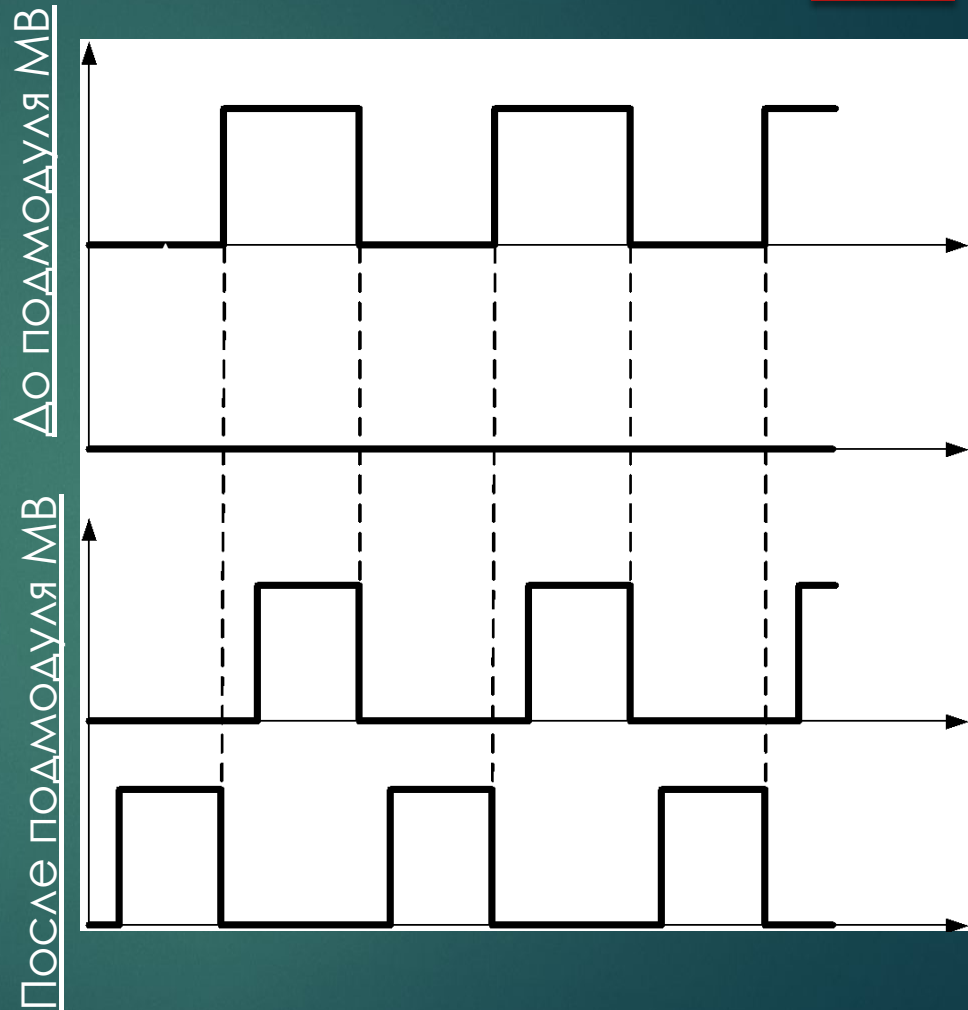
# Подмодуль мёртвого времени

14

Рисунок демонстрирует результат настройки подмодуля мёртвого на генерацию комплиментарного сигнала канала В с добавлением мёртвого времени.

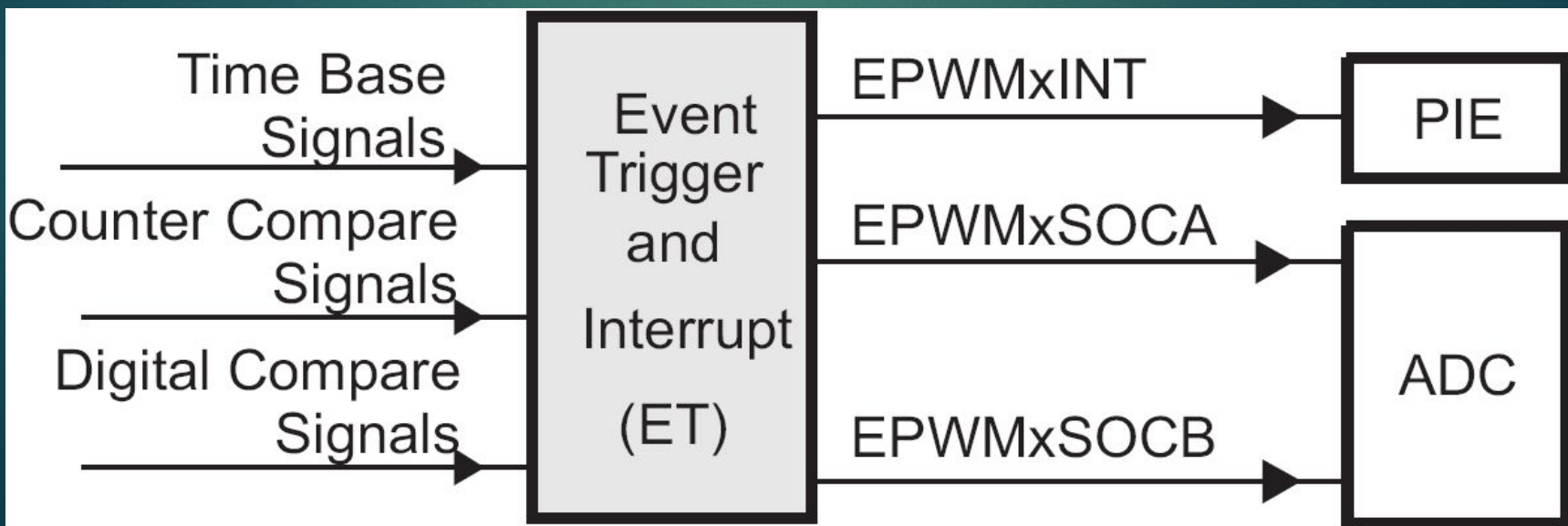
Как видно, модуль добавляет задержку на включение канала А и генерирует сигнал канала В, внося задержку между его включением и отключением канала А.

Модуль имеет очень гибкие и сложные настройки, поэтому они не здесь рассматриваются. Ознакомиться с ними можно в документации.



Подмодуль *Event Trigger* позволяет генерировать прерывания и запускать АЦП по событиям, приходящим от модуля сравнения (достижение счётчиком нуля или периода, сравнение счётчика с уставками). Это похоже на поведение подмодуля действия с той разницей, что вместо изменения состояния каналов А и В запускается АЦП или прерывание.

Кроме перечисленных событий, триггером для запуска может служить ещё и сигнал от компаратора (это одно из периферийных устройств микроконтроллера).



# Подмодуль запуска событий

16

В таблице ниже перечислены основные и наиболее часто используемые регистры для настройки этого модуля.

Имя регистра	Размер	Значение
ETSEL.bit.SOCAEN(0)	1 бит	Разрешение запуска АЦП: 0 – запрещено, 1 – разрешено
ETSEL.bit.INTEN(0)	1 бит	Разрешение вызова прерывания: 0 – запрещено, 1 – разрешено
ETSEL.bit.SOCASEL(0)	3 бита	Событие для запуска АЦП
ETSEL.bit.INTSEL(0)	3 бита	Событие для вызова прерывания
ETPS.bit.SOCAPRD(0)	2 бита	Периодичность запуска АЦП
ETPS.bit.INTPRD(0)	2 бита	Периодичность вызова прерывания

Возможные события для регистров SOCASEL и INTSEL: 0 – событие от компаратора,

1 –  $TBCTR == 0$ , 2 –  $TBCTR == TBPRD$ , 3 –  $TBCTR == 0$  или  $TBCTR == TBPRD$ ,  
4 –  $TBCTR == CMPA$  при счёте вверх, 5 –  $TBCTR == CMPA$  при счёте вниз,  
6 –  $TBCTR == CMPB$  при счёте вверх, 7 –  $TBCTR == CMPB$  при счёте вверх.

Возможные варианты для регистров SOCAPRD и INTPRD: 0 – никогда не запускать АЦП / вызывать прерывание, 1 – делать это при каждом событии, 2 – делать это при каждом втором событии, 3 – делать это при каждом третьем событии



Кроме рассмотренных подмодулей, есть ещё и другие.

- Подмодуль *Trip Zone* позволяет мгновенно отключать силовые ключи при появлении определённого сигнала (низкого или высокого) на каком либо выводе микроконтроллера. На этот вывод, например, можно завести сигнал аппаратной аварии ключа.
- Подмодуль *PWM Chopper* позволяет преобразовать высокий сигнал канала А или В в последовательность коротких импульсов.
- Подмодуль *Digital Compare* обеспечивает интерфейс между модулем ePWM и компаратором и позволяет настраивать реакцию ePWM на сигналы компаратора.

Прерывания от подмодуля *Trip Zone* имеют более высокий приоритет, чем другие прерывания ePWM.

	INTx.8	INTx.7	INTx.6	INTx.5	INTx.4	INTx.3	INTx.2	INTx.1
<b>INT1.y</b>	WAKEINT (LPM/WD) 0xD4E	TINT0 (TIMER 0) 0xD4C	ADCINT9 (ADC) 0xD4A	XINT2 Ext. int. 2 0xD48	XINT1 Ext. int. 1 0xD46	Reserved - 0xD44	ADCINT2 (ADC) 0xD42	ADCINT1 (ADC) 0xD40
<b>INT2.y</b>	Reserved - 0xD5E	EPWM7_TZINT (ePWM7) 0xD5C	EPWM6_TZINT (ePWM6) 0xD5A	EPWM5_TZINT (ePWM5) 0xD58	EPWM4_TZINT (ePWM4) 0xD56	EPWM3_TZINT (ePWM3) 0xD54	EPWM2_TZINT (ePWM2) 0xD52	EPWM1_TZINT (ePWM1) 0xD50
<b>INT3.y</b>	Reserved - 0xD6E	EPWM7_INT (ePWM7) 0xD6C	EPWM6_INT (ePWM6) 0xD6A	EPWM5_INT (ePWM5) 0xD68	EPWM4_INT (ePWM4) 0xD66	EPWM3_INT (ePWM3) 0xD64	EPWM2_INT (ePWM2) 0xD62	EPWM1_INT (ePWM1) 0xD60