Продвинутые таймеры на STM32F1xx

Кто, почему и зачем?

Кто?

- В микроконтроллере stm32f103 имеются 2 продвинутых таймера

- TIM1
- TIM8

Почему они называются продвинутыми?

Как и остальные таймеры, TIM1 и TIM8 позволяют:

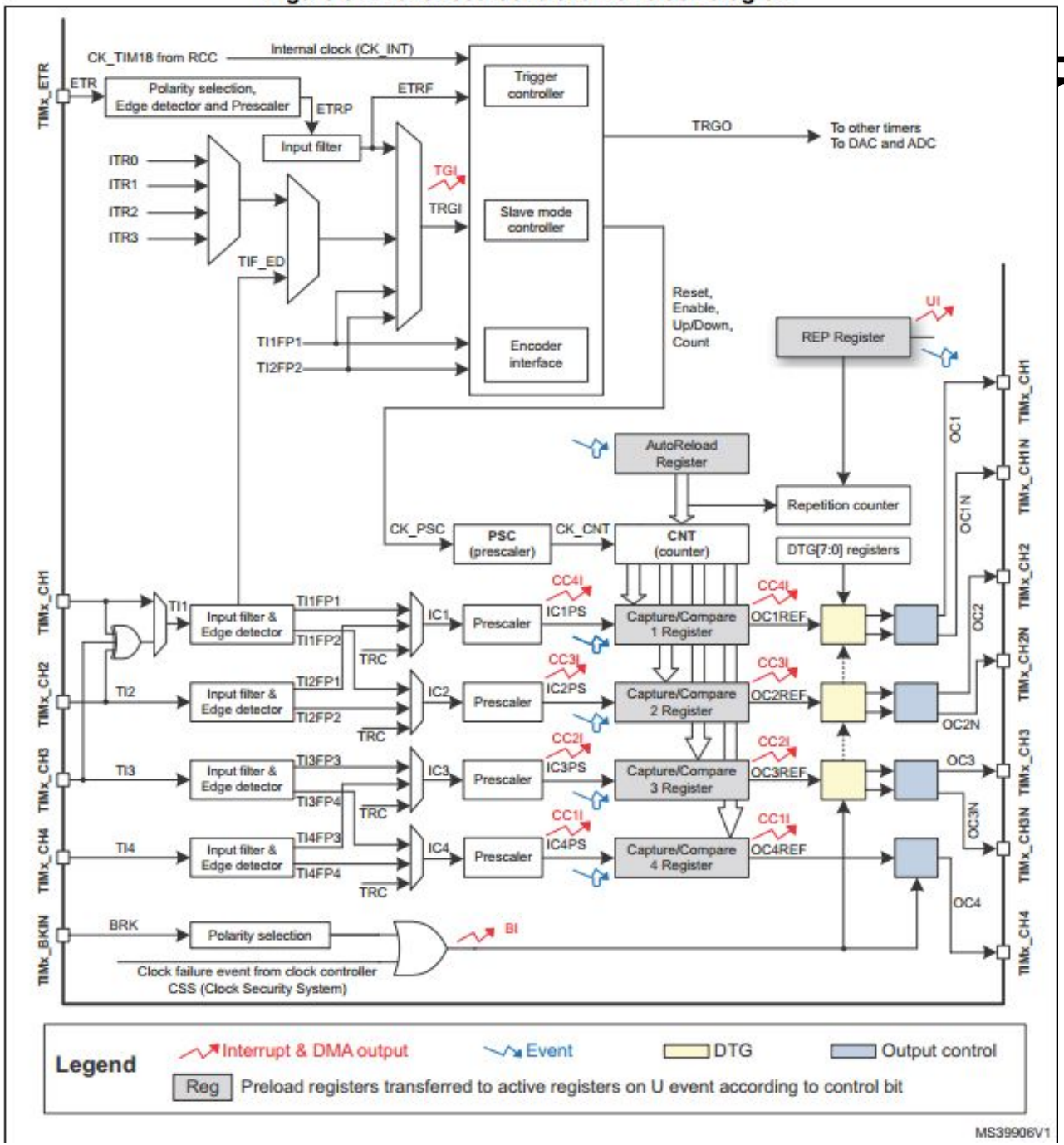
- Захватывать ШИМ-сигналы
- Выводить ШИМ-сигналы
- Считать вверх/вниз/вверх-вниз
- Изменять частоту работы, период заполнения и тд
- Синхронизироваться друг с другом (для увеличения разрядности, например)

НО!

Почему они называются продвинутыми?

- Выводить на выходы микроконтроллера комплементарный ШИМ-сигнал с настраиваемым «Мёртвым временем»
- Задавать счетчик повторов, который позволит выставлять флаг прерывания через заданное количество срабатываний события, порождающего прерывание (переполнения, например)
- Обрабатывать данные с квадратурного энкодера и датчика Холла
- Сбрасывать выходы микроконтроллера, на которые выводился ШИМ в предустановленное состояние Reset
- Использовать Прямой Доступ к Памяти по любому поводу, описанному ранее

Figure 52. Advanced-control timer block diagram



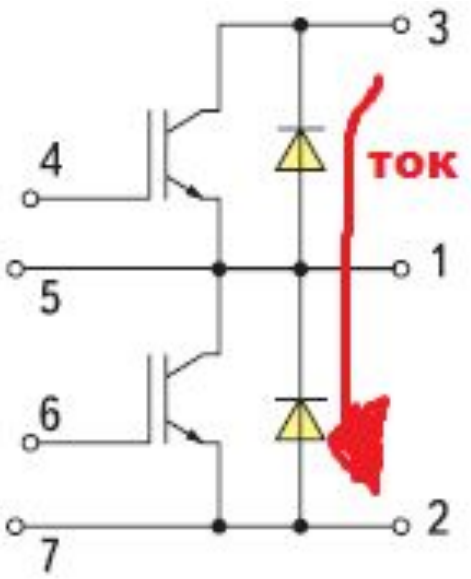
Зачем они нам нужны?

1) При управлении полумостом на полевых или IGBT транзисторах необходимо дожидаться полного закрытия одного транзистора полумоста перед открытием второго, иначе оба транзистора будут открыты и ток потечет через них, игнорирую нагрузку.

С этим помогает

мёртвое время

Зачем они нам нужны?



1, 5 – выход на нагрузку

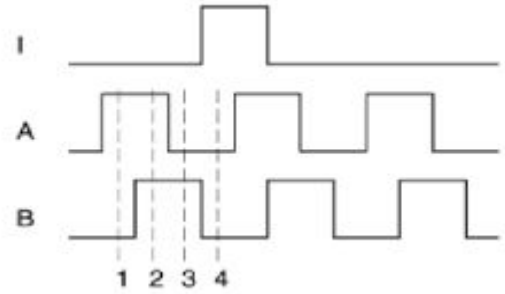
2, 7 – общая точка (земля)

3 – напряжение питания полумоста

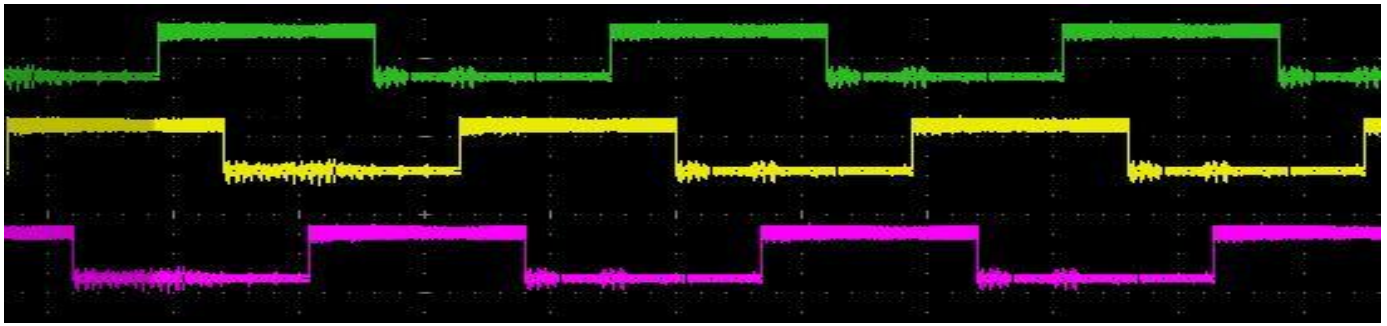
4, 6 – управляющие сигналы на ключи полумоста

Зачем они нужны?

2)Считывать показания с датчиков Холла или квадратурного энкодера



Состояние	Канал A	Канал B
1	High	Low
2	High	High
3	Low	High
4	Low	Low



Зачем они нужны?

Figure 95. Example of hall sensor interface

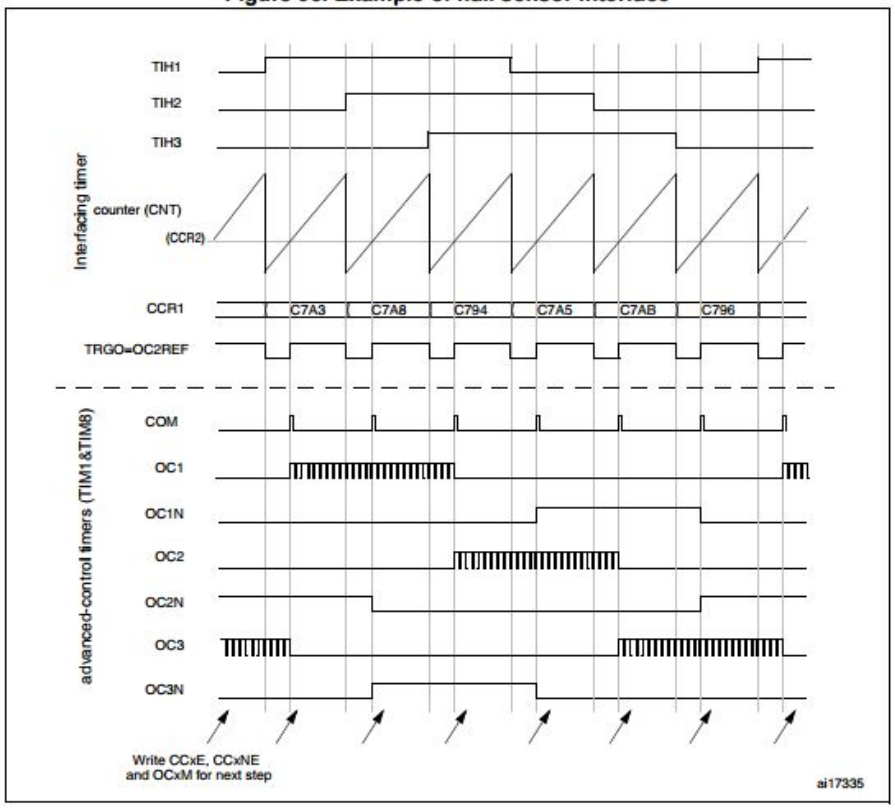
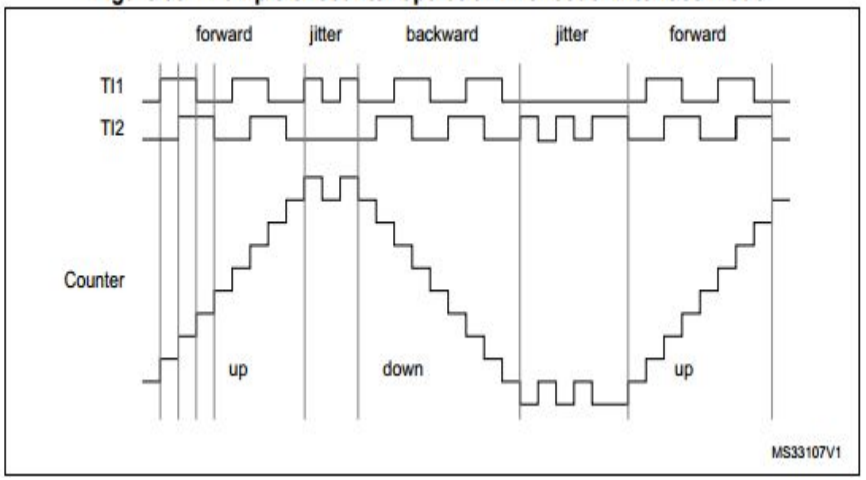


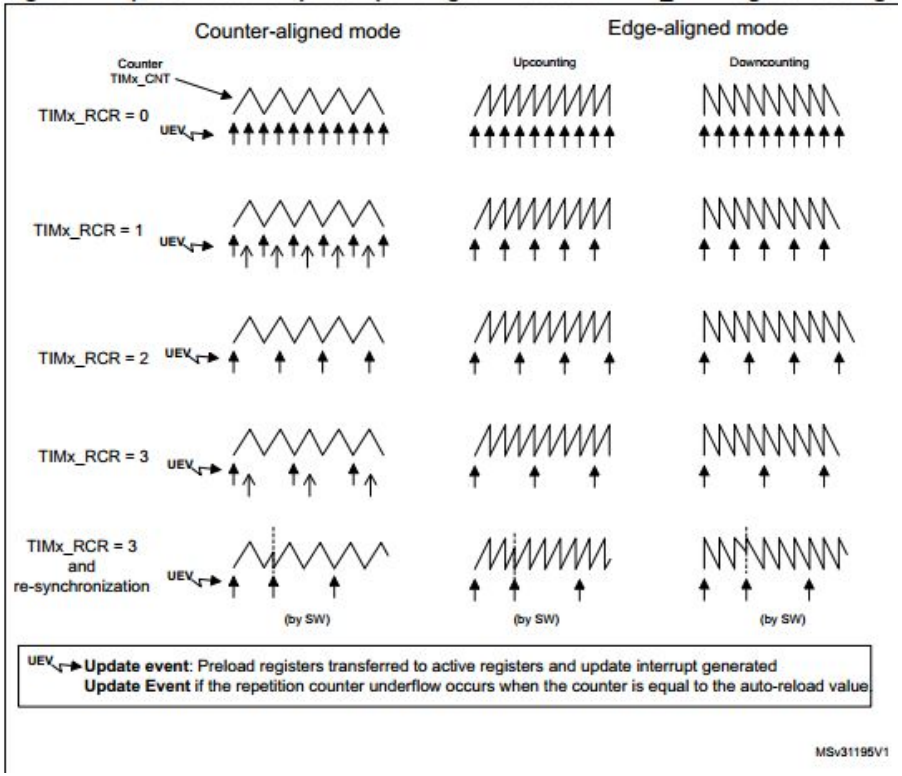
Figure 93. Example of counter operation in encoder interface mode.



Зачем они нужны?

3)Прерывания после заданного числа событий

Figure 72. Update rate examples depending on mode and TIMx_RCR register settings



Как их настроить?

1) Запускаем тактирование таймера
RCC_APB2PeriphClockCmd(RCC_APB2Periph_TIMx , ENABLE);
2) Создаём экземпляр структуры основных настроек таймера
TIM_TimeBaseInitTypeDef
Заполняем её поля, глядя на её определение в файле stm32f10x_tim.h

```
typedef struct
{
  uint16_t TIM_Prescaler;        /*!< Specifies the prescaler value used to divide the TIM clock.
                                  This parameter can be a number between 0x0000 and 0xFFFF */

  uint16_t TIM_CounterMode;      /*!< Specifies the counter mode.
                                  This parameter can be a value of @ref TIM_Counter_Mode */

  uint16_t TIM_Period;           /*!< Specifies the period value to be loaded into the active
                                  Auto-Reload Register at the next update event.
                                  This parameter must be a number between 0x0000 and 0xFFFF.  */

  uint16_t TIM_ClockDivision;    /*!< Specifies the clock division.
                                 This parameter can be a value of @ref TIM_Clock_Division_CKD */

  uint8_t TIM_RepetitionCounter;  /*!< Specifies the repetition counter value. Each time the RCR downcounter
                                  reaches zero, an update event is generated and counting restarts
                                  from the RCR value (N).
                                  This means in PWM mode that (N+1) corresponds to:
                                    - the number of PWM periods in edge-aligned mode
                                    - the number of half PWM period in center-aligned mode
                                  This parameter must be a number between 0x00 and 0xFF.
                                  @note This parameter is valid only for TIM1 and TIM8. */
} TIM_TimeBaseInitTypeDef;
```

3) Вызвать TIM_TimeBaseInit

Как их настроить?

4) Создаём экземпляр структуры настроек Output Compare таймера TIM_OCInitTypeDef

Заполняем её поля, глядя на её определение в файле stm32f10x_tim.h

```
typedef struct
{
  uint16_t TIM_OCMode;      /*!< Specifies the TIM mode.
                                 This parameter can be a value of @ref TIM_Output_Compare_and_PWM_modes */

  uint16_t TIM_OutputState;  /*!< Specifies the TIM Output Compare state.
                                 This parameter can be a value of @ref TIM_Output_Compare_state */

  uint16_t TIM_OutputNState;  /*!< Specifies the TIM complementary Output Compare state.
                                 This parameter can be a value of @ref TIM_Output_Compare_N_state
                                 @note This parameter is valid only for TIM1 and TIM8. */

  uint16_t TIM_Pulse;       /*!< Specifies the pulse value to be loaded into the Capture Compare Register.
                                 This parameter can be a number between 0x0000 and 0xFFFF */

  uint16_t TIM_OCPolarity;    /*!< Specifies the output polarity.
                                 This parameter can be a value of @ref TIM_Output_Compare_Polarity */

  uint16_t TIM_OCNPolarity;   /*!< Specifies the complementary output polarity.
                                 This parameter can be a value of @ref TIM_Output_Compare_N_Polarity
                                 @note This parameter is valid only for TIM1 and TIM8. */

  uint16_t TIM_OCIdleState;   /*!< Specifies the TIM Output Compare pin state during Idle state.
                                 This parameter can be a value of @ref TIM_Output_Compare_Idle_State
                                 @note This parameter is valid only for TIM1 and TIM8. */

  uint16_t TIM_OCNIdleState;  /*!< Specifies the TIM Output Compare pin state during Idle state.
                                 This parameter can be a value of @ref TIM_Output_Compare_N_Idle_State
                                 @note This parameter is valid only for TIM1 and TIM8. */
} TIM_OCInitTypeDef;
```

5) Вызвать для каждого канала TIM_OCxInit(TIMy, &TIM_OCInitStructure);

Как их настроить?

6) Создаём экземпляр структуры настроек BDTR таймера TIM_OCInitTypeDef

Заполняем её поля, глядя на её определение в файле stm32f10x_tim.h

```
typedef struct
{

  uint16_t TIM_OSSRState;       /*!< Specifies the Off-State selection used in Run mode.
                                   This parameter can be a value of @ref OSSR_Off_State_Selection_for_Run_mode_state */

  uint16_t TIM_OSSIState;       /*!< Specifies the Off-State used in Idle state.
                                   This parameter can be a value of @ref OSSI_Off_State_Selection_for_Idle_mode_state */

  uint16_t TIM_LOCKLevel;       /*!< Specifies the LOCK level parameters.
                                   This parameter can be a value of @ref Lock_level */

  uint16_t TIM_DeadTime;        /*!< Specifies the delay time between the switching-off and the
                                   switching-on of the outputs.
                                   This parameter can be a number between 0x00 and 0xFF  */

  uint16_t TIM_Break;           /*!< Specifies whether the TIM Break input is enabled or not.
                                   This parameter can be a value of @ref Break_Input_enable_disable */

  uint16_t TIM_BreakPolarity;    /*!< Specifies the TIM Break Input pin polarity.
                                   This parameter can be a value of @ref Break_Polarity */

  uint16_t TIM_AutomaticOutput;  /*!< Specifies whether the TIM Automatic Output feature is enabled or not.
                                   This parameter can be a value of @ref TIM_AOE_Bit_Set_Reset */
} TIM_BDTRInitTypeDef;
```

7) Вызвать для каждого канала TIM_BDTRConfig

Как их настроить?

8)Вызываем для каждого прямого ШИМ-выхода

TIM_CCxCmd(TIM_TypeDef* TIMx, uint16_t TIM_Channel, uint16_t TIM_CCx)

9)Вызываем для каждого инверсного ШИМ-выхода

TIM_CCxNCmd(TIM_TypeDef* TIMx, uint16_t TIM_Channel, uint16_t TIM_CCx)

ВАЖНО!

10) Функция разрешения ШИМ-выхода

TIM_CtrlPWMOutputs(TIM1, ENABLE);

11) Запускаем таймер

TIM_Cmd(TIMx, ENABLE);

12) Готово

Фичи для успеха в лабораторной работе

- Настроить таймер в режим Center-align
- Настроить 4-й канал таймера на вывод ШИМа с заполнением PWM_PERIOD-1
- Настроить триггер запуска преобразования АЦП на переполнение 4-го канала

Figure 85. Center-aligned PWM waveforms (ARR=8)