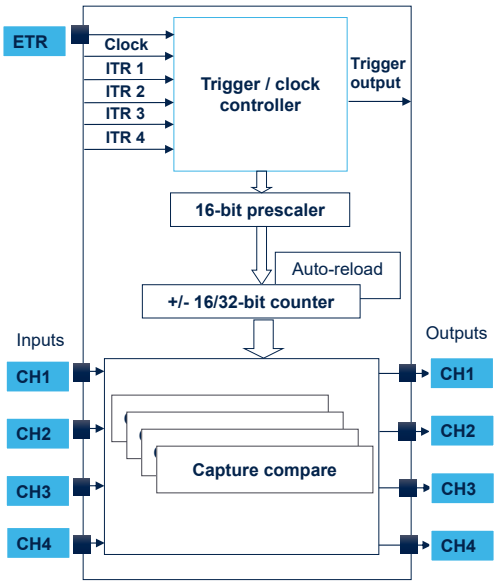
Hello, and welcome to this presentation on the advanced-control, general-purpose and basic timers embedded in STM32 microcontrollers. It covers their main features which are useful for handling any timing-related events, generating waveforms and measuring the timing characteristics of input signals.

Overview



- Multiple timer units providing timing resources
  - Internally (triggers and time-bases)
  - Externally, for outputs or inputs:
    - For waveform generation (PWM)
    - For signal monitoring or measurement (frequency or timing)

Application benefits

- Versatile operating modes reducing CPU burden and minimizing interfacing circuitry needs
- A single architecture for all timer instances offers scalability and ease-of-use
- Also fully featured for motor control and digital power conversion applications

life.augmented

2

The STM32 embeds multiple timers providing timing resources for software or hardware tasks. The software tasks mainly consist of providing time bases, timeout event generation and time-triggers. The hardware tasks are related to I/Os: the timers can generate waveforms on their outputs, measure incoming signal parameters and react to external events on their inputs.
The STM32 timers are very versatile and provide multiple operating modes to off-load the CPU from repetitive and time-critical tasks, while minimizing interfacing circuitry needs. All STM32 timers (with the sole exception of the low-power timer) are based on the same scalable architecture. Once the timer operating principles are known, they are valid for any of the timers. This architecture includes interconnection features and allows several timers to be combined into larger configurations.

Lastly, some of the timers feature specific functions for electrical motor control and digital power conversion such as lighting or digital switched mode power supplies.

Key features

- All timers are based on the same architecture, scalable in terms of:
  - Number of inputs/outputs (from 1 to 9)
  - Resolution (16- or 32-bit)
  - Features (PWM modes, DMA, synchronization, up/down counting)
- Multiple timers can be linked and synchronized
- Each timer channel is configurable independently as an input or output
- Multiple interconnects with other peripherals are available for monitoring or triggering purposes

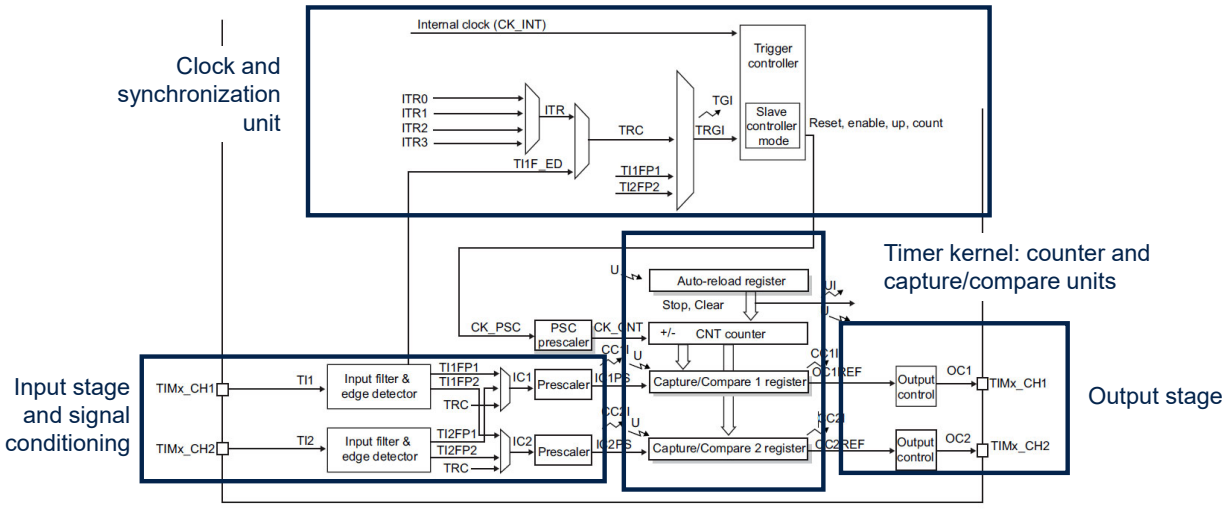
life.augmented

3

Here are the key features of the STM32 timers. All timers are based on the same architecture and are available in several derivatives listed later in this presentation. The timers mainly differ in the number of inputs and outputs they have, from a pure time base without any I/Os to an advanced control version with 9 I/Os. Most of the timers feature 16-bit counters, while some have 32-bit counters. Some features may not be present on the smallest timer derivatives (for example, DMA, synchronization, and up/down counting modes).
Most of the timers can be linked and synchronized to build larger time-base timers, have a higher number of synchronous waveforms, or handle complex timings and waveforms.
Within a timer, each and every channel can be configured independently as an input (typically for capture) or as an

output (typically for a PWM).
The timers can serve as a trigger for other peripherals, for instance to start ADC conversions, or to monitor the internal clocks, thanks to the interconnect matrix.

Block diagram (TIM12)

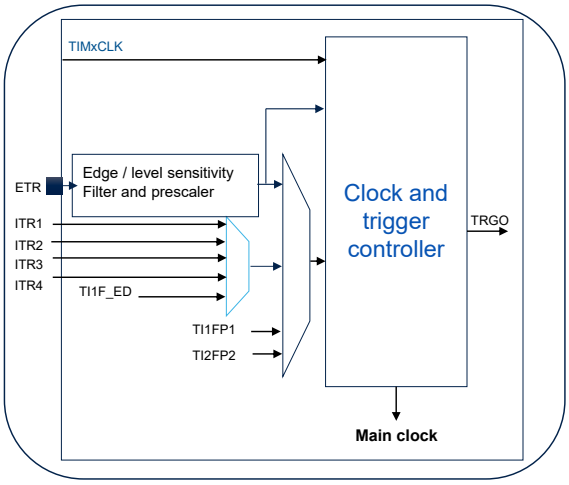


This slide presents the block diagram of the medium-featured TIM12 timer.
The timer kernel consists of a 16-bit up-counter, coupled with an auto-reload register to program the counting period. The 2 timer channels are controlled by 2 capture-compare registers.
The counter is fed by the Clock and Trigger controller, also responsible for the timer chaining.
Shown on the left are the input stage and the input conditioning circuitry while on the right we have the output stage.
Note that TIMxCH1 and TIMxCH2 appear on both sides to indicate they are both input and output capable.

Timer clocking schemes

Multiple internal or external clocking options

- Default clock derived from APB domains
  - Timers are split over APB1 and APB2 domains for low-power optimizations
- External clocking possible with:
  - Other on-chip timers (ITRx inputs)
  - Input pins 1 and 2 (TI1, TI2)
    - Inc. digital filter and programmable edge sensitivity
  - Multi-purpose external trigger input (ETR)
    - Inc. digital filter, programmable edge sensitivity and basic prescaler (/2, /4, /8)
  - Quadrature signals from encoders



life.augmented

5

The timer features multiple clocking options.
The Clock and Trigger controller, also responsible for timer chaining, handles the clock for the counter.
The default clock comes from the reset and clock controller, linked to one of APB clock domains. The various timers are shared on the 2 APB domains to implement low-power schemes (typically one high-speed APB and one low-speed APB to limit the current drawn by the peripherals, including the timers).
External timer clocking allows counting of external events or to have a counting period externally adjusted. The clock source can be provided by other on-chip timers, using one of the 4 internal trigger inputs (ITR1…ITR4). Input pins 1 and 2 can also serve as external clocks, with the option of including digital filters to remove spurious events. The external trigger input (ETR) can be configured as an

external clock, with a digital filter, programmable edge sensitivity and a first basic prescaler stage to reduce the frequency of incoming signals if needed.
Lastly, the quadrature signals from an encoder can be processed to provide a clock and a counting direction, as described later in this presentation.

Counting period management

Fine and accurate period setting

- Each timer embeds a 16-bit linear prescaler (1,2,3…65536)
- An auto-reload register defines the counting period
- An update event (interrupt or DMA) is issued on overflow/underflow
  - Triggers transfer of register contents from preload to active (prescaler, period, compare)
  - → Precise period change (prescaler is updated only at overflow)
  - → Glitch-less operation when compare registers are updated
- The update interrupt issuing rate is adjusted with a repetition counter

life.augmented

6

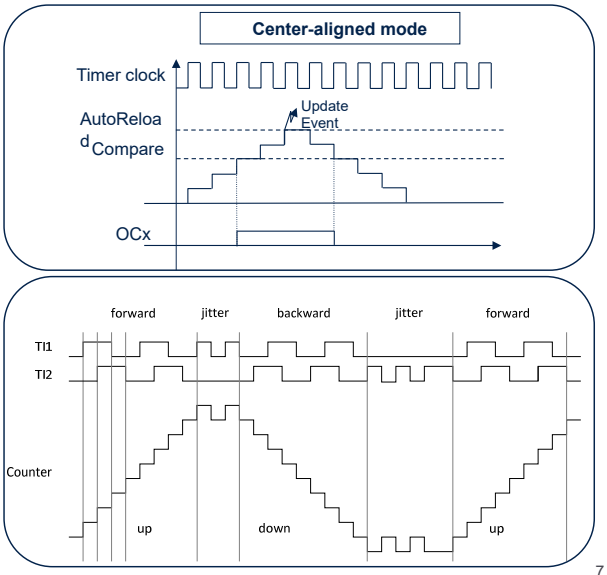
This slide explains how to adjust the timer counting period. Each timer embeds a linear clock prescaler which allows you to divide the clock by any integer between 1 and 65536. This allows the counting pace to be precisely adjusted. For instance, a division by 80 will yield a precise 1 MHz counting rate when the APB clock is 80 MHz.
The autoreload register defines the counting period. In Down-counting mode, the counter is automatically reloaded with the period value when it underflows. In Up-counting mode, the counter rolls over and is reset when it exceeds the auto-reload value.
An update event is issued when the counter underflows or overflows and a new period starts. It triggers an interrupt or DMA request that is used for adjusting timer parameters synchronously with its period, which is useful for real-time control. This update event triggers the transfer from

preload to active registers for multiple parameters, and in particular for the clock prescaler, auto-reload value, compare registers and PWM mode.
An 8-bit programmable repetition counter allows you to decouple the interrupt issuing rate from the counting period, and have, for instance, one interrupt every single, 2nd, 3rd and up to 256th PWM period. This is particularly useful when dealing with high PWM frequencies.

Counting mode

- **Support of incremental / quadrature encoders and motor drive applications**

- Up- and down-counting modes supported
  - On TIM1/8 and TIM2/3/4/5
- Center-aligned PWM generation
  - Direction changes on overflow and underflow
  - Reduces acoustic noise in electric motors
- Built-in support of quadrature encoders
  - Rotary encoder / digital potentiometer
  - Position sensor
  - Allows direct angle reading in timer



life.augmented

7

MS33107V1

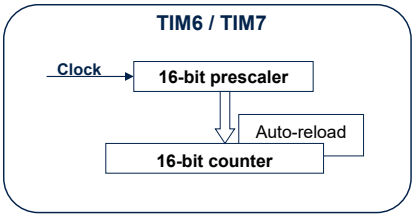
Some of the STM32 timers feature up/down counting modes: the advanced control timers 1 and 8 and the general-purpose timers 2, 3, 4 and 5.
The counting direction can be programmed by software or automatically managed by the timer in center-aligned PWM mode. In this mode, the counting direction changes automatically on counter overflow and underflow. For a given PWM switching frequency, this mode reduces the acoustic noise by doubling the effective current ripple frequency, thus providing the optimum tradeoff between the power stage's switching losses and noise.
The counting direction can also be automatically handled when the timer is in Encoder mode. Quadrature encoders are typically used for high-accuracy rotor position sensing in electrical motors, or for digital potentiometers. From the two outputs of a quadrature encoder sensor (also called an

incremental encoder), the timer extracts a clock on each and every active edge and adjusts the counting direction depending on the relative phase-shift between the two incomings signals. The timer counter thus directly holds the angular position of the motor or the potentiometer.

Timer as internal timing resource

For software and hardware time-base

- The timer can be used as simple time-base
  - For software management
  - To provide a periodic trigger to other peripherals
    - ADC, DAC or other timers
- The update event (on counter overflow) can be used to trigger an interrupt
  - The simplest option, using TIM6 and TIM7 basic timers (without outputs)
- Other means when using general-purpose timers
  - Using the compare events
    - Allows to have multiple events per period
  - Using the trigger output of the timer



life.augmented

8

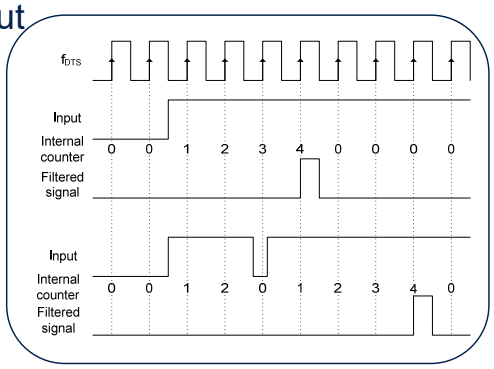
The simplest use case for a timer is to provide an internal time base.
This is commonly used by software routines, either to provide periodic interrupts or single-shot timeout protection. The timer can also provide periodic triggers to other on-chip peripherals, such as the ADC, DAC and other timers.
The update event from the timer (typically on counter overflow) is the usual means to have a software time base interrupt or to trigger a periodic event. The basic timers TIM6 and TIM7 are best suited for such a task, as they are the simplest timer derivatives with no input/output channel. It is also possible to generate internal timings using any other timer, using compare events or using the trigger outputs on any other timer. It is possible to generate multiple timing events with a single timer using multiple

compare channels.

Input capture

Includes signal pre-conditioning to decrease CPU overhead

- Each channel can be configured individually as input capture with the following features:
  - Input remapping (one input can be mapped to 2 capture channels)
  - Programmable edge-sensitivity (rising / falling / both)
  - Event prescaler (1 capture every 1 / 2 / 4 / 8 events)
  - Digital filter (for debouncing and noise removal)
- A capture event causes the counter value to be transferred into the capture register and triggers an interrupt or a DMA request
  - An overcapture flag is set if the capture register is overwritten without having been read



life.augmented

9

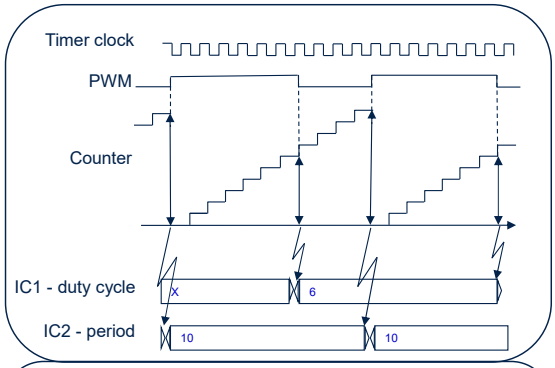
This slide describes the input capture features.
Each channel can be individually configured as input capture with a number of signal conditioning options. An input can be mapped on two capture channels (typically to differentiate rising-edge from falling-edge capture). The edge sensitivity is programmable and can be rising edge, falling edge or both edges. An event prescaler allows capture of one event every 2, 4 or 8 events. This decreases the CPU burden when processing high frequency signals and allows the measurement to be more accurate, since it is performed over multiple input signal periods.
Spurious transition events due to noise or bounces can be removed using a programmable digital filter. The figure shows how a signal is filtered when the filter acceptance is set to 4. In the upper case, a clean rising edge capture is

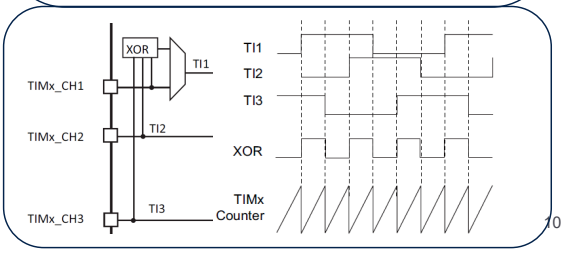
triggered 4 sampling periods after the rising edge, as one can notice looking at the internal counter value. In the lower case, a glitch causes the filter counter to be reset and the capture to happen after 4 successive samples at high level have been counted.
Once the capture trigger is issued, the timer’s counter is transferred into the capture register and an interrupt or a DMA request can be issued. If a new capture occurs before the previous one has been read, the capture register is over-written and an overcapture flag is set for the software to manage this condition if needed.

Advanced capture options

Direct measurement with no software overhead

- Clear-on-capture mode
  - This mode allows the counter to be reset immediately after the capture has been triggered
- PWM input mode
  - The timer directly captures the period and a pulse width of a signal in two capture registers
- Clear-on-capture mode with XOR function
  - Allows the capture of the interval between any edge of up to 3 inputs
    - → Typically used for Hall sensors in e-motors





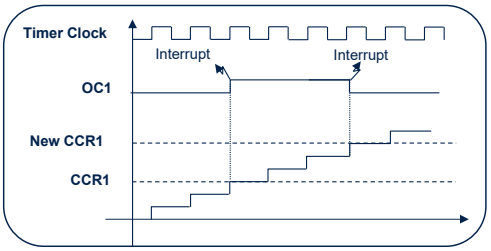
This slide presents some of the more advanced capture-related functions.
The Clear-on-capture mode causes a counter reset immediately after the capture has been triggered. This allows a direct measurement of the period, while a traditional free-running counter would require additional computation to obtain the period following the trigger.
In PWM input mode, the timer is able to capture both the period and the duty cycle of an incoming PWM signal. The input signal is internally routed to 2 capture channels. The signal's rising edge is captured on input capture 2 to provide the period value with the Clear-on-capture mode. The falling edge is captured by the capture 1 channel, which provides the pulse length duration. The duty cycle then simply corresponds to the ratio between input capture 1 and input capture 2.

Lastly, the timer includes an XOR function to combine the three input channels with XOR logic. This is typically used to handle the three 120° phase-shifted signals coming from the Hall sensors in electrical motors. This allows you to have a clear on capture happening on each and every edge of the three signals and have a capture value directly usable for speed regulation.

Output compare

For simple output waveforms or to indicate a period is elapsed

- When the counter matches the compare register value:
  - Corresponding output pin can be programmed to be:
    - Set
    - Reset
    - Toggled
    - Unchanged
  - Set a flag in the interrupt status register
    - Eventually generates an interrupt or DMA request if the corresponding enable bit is set
  - The compare registers can be programmed with or without preload registers
- The programmed output mode can also be preloaded
  - Allows glitch-less transition from one mode to the other (typically from PWM to continuously On or Off state)



life.augmented

11

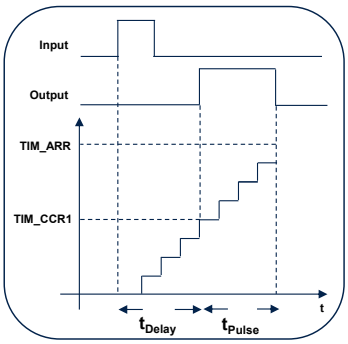
This slide presents the output compare features.
A compare event is generated when the counter matches the value of the compare register. This event can trigger an interrupt or a DMA request and can be reflected on the corresponding output pin by an output set, output reset or output toggle.
The compare register can be preloaded. The preload must be disabled if multiple compare values must be written during a counting period. On the contrary, the use of preload mode must be preferred for applications with real-time constraints, since this gives a higher time margin for the software to update the compare register with the next value. The transfer from the preload to the active value is triggered by an update event, when the counter overflows or underflows.
The output compare mode can also be preloaded, so as to

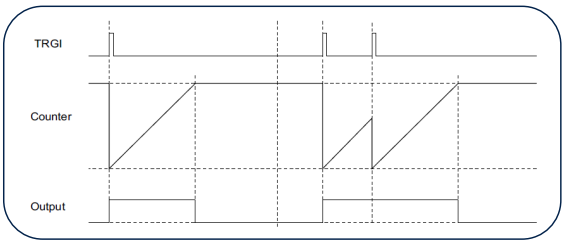
allow glitch-less transition from a PWM mode to a forced On or Off state, for instance.

One-pulse mode

For externally synchronized waveform generation

- Allows the counter to be started in response to a stimulus and to generate a pulse
  - With a programmable length
  - After a programmable delay
- Two software programmable waveforms
  - Single pulse
  - Repetitive pulse
- Retriggerable option
  - Pulse width is extended in case of new trigger





life.augmented

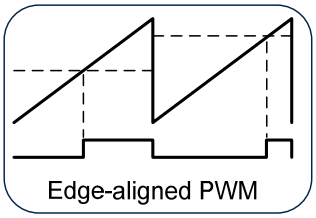
12

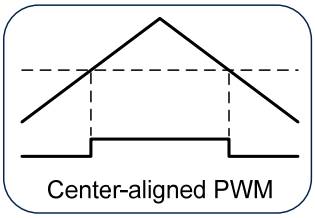
One-pulse mode is used to generate a pulse of a programmable length in response to an external event. The pulse can start as soon as the input trigger arrives or after a programmable delay. The compare 1 register (CCR1) value defines the pulse start time, while the auto-reload register (ARR) value defines the end of pulse. The effective pulse width is then defined as the difference between the ARR and CCR1 register values.
The waveform can be programmed to have a single pulse generated by the trigger, or to have a continuous pulse train started by a single trigger.
One-pulse mode also offers a retriggerable option. In this case, a new trigger arriving before the end of the pulse will cause the counter to be reset and the pulse width to be extended accordingly.

A few PWM modes

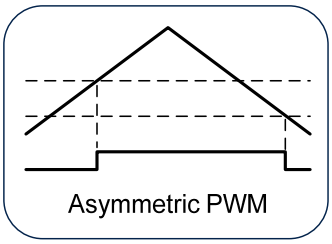
A variety of PWM modes to address multiple applications

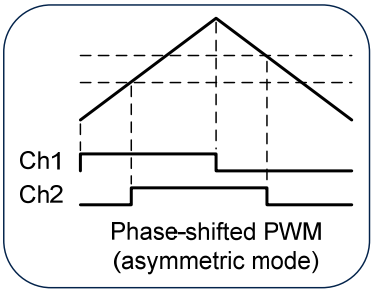
- Basic PWM, edge- or center-aligned





- Asymmetric center-aligned PWM





life.augmented

13

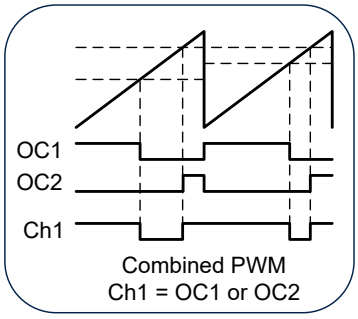
This slide presents some of the PWM modes.
The standard edge-aligned PWM mode is programmed with the auto-reload register defining the period and the compare register defining the duty cycle, the counter being in up-only or down-only counting mode. A single timer can generate up to 4 PWM signals with independent duty cycles and identical frequency. When multiple PWM waveforms are generated by the same timer, all falling edges occur at the same time, hence the term edge-aligned. On the contrary, the rising and falling edges of center-aligned PWMs are not synchronized with the counter roll-over, so that switching time varies with the duty cycle value. This is achieved by programming the counter in up-down mode. This mode is interesting as it spreads the switching noise when multiple PWMs are generated with the same timer. This is a key feature for

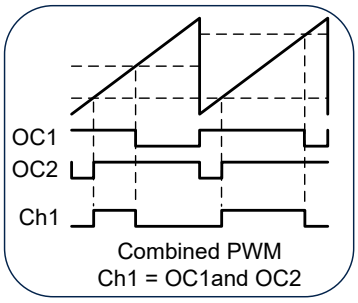
three-phase PWM generation for electric motor drives, since it allows you to double the frequency of the current ripple for a given switching frequency. For instance, a 10 kHz PWM will generate inaudible 20 kHz current ripple. This minimizes the switching losses due to the PWM frequency while guaranteeing silent PWM operation.
A variant of the center-aligned mode is the asymmetric PWM mode, where two compare registers define the turning on and off of the PWM signal. This provides higher resolution for pulse width setting, since turn-on and turn-off times are individually defined. It also allows the generation of phase-shifted PWM signals, necessary to drive DC/DC converters based on the full-bridge phase-shifted topology. In this case, the timer provides two PWM signals with identical frequency, 50% duty cycle, and a phase-shift varying from 0 to 180°.

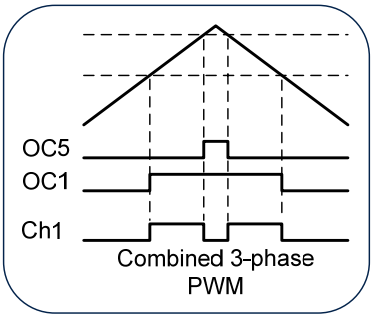
Some more PWM modes

Extends the PWM capabilities and avoids external glue logic

- Combined PWM mode
  - Combines two channels with OR or AND function for more complex waveforms
- Combined 3-phase mode
  - Allows a 4th PWM to be combined with a regular 3-phase PWM for zero vector insertion







life.augmented

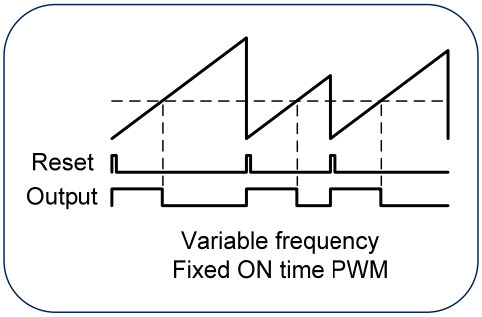
14

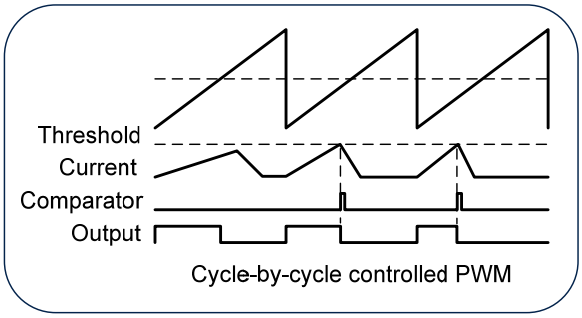
This slide presents the combined PWM modes.
This mode allows a logic combination of two PWM signals to be generated by adjacent channels (output compare 1 and 2 or output compare 3 and 4). The PWMs can be ORed or ANDed to create complex waveforms. Typically, this allows you to have two periodic pulses generated with any pulse width and any phase relationship value.
The combined 3-phase mode specifically targets 3-phase motor control applications. In this case, channel 5 of the timer can be combined with any of the three channels (1, 2 and 3) to insert a low state in the middle of a centered-pattern PWM signal. This mode greatly simplifies the implementation of low-cost current sensing techniques for 3-phase motor control, using a technique usually referred to as zero vector insertion.

Advanced PWM modes

For PWM signals requiring external control

- Variable-frequency PWM
  - Driven by an external signal
- Cycle-by-cycle controlled duty cycle
  - For current loops, driven by comparator or external pin





life.augmented

15

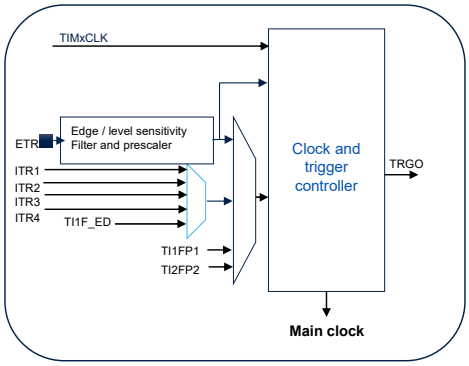
This slide presents some more specific PWM modes, where either the frequency or the duty cycle can be driven by external signals.
The timer can provide variable frequency signals, using an external reset signal connected either on the ETR, or on the channel 1 or 2 inputs. The purpose of this mode is to provide a signal with a fixed On or Off time and a continuously adjusted frequency controlled by the hardware. The timer provides control for the On (or Off) time, using the compare register, while the auto-reload register guarantees that the PWM will not stop if the external reset is missing, thus providing a safe control in boundary conditions. This technique is used for a variety of purposes, such as transition mode PFC (Power Factor Controller) for mains-supplied applications and current-controlled digital LED lighting.

Another mode for the timer is to have the duty cycle controlled by hardware, with either an on-chip comparator or an off-chip signal. The PWM operates at a fixed frequency, the maximum duty cycle is set by the compare register and the actual value controlled cycle-by-cycle. This is used for applications requiring current-controlled PWMs, typically for driving DC motors or solenoids. In this case, a comparator monitors the peak current value into the load. As soon as the current exceeds a programmed threshold, the comparator resets the PWM output, which is then automatically re-started at the next PWM period, thus providing a controlled peak current value.

Cascading timers 1/2

Scalable design for higher flexibility

- The trigger controller provides the ability to cascade multiple timers in a master/slave configuration
  - Slave: the trigger controller gathers inputs on TRGI
    - From an external trigger pin (ETR)
    - From other on-chip resources on ITRx
      - Typically TRGO outputs from other timers
  - Master: internal timer signals are sent to TRGO



- A given timer can simultaneously operate in Slave and Master modes in a cascaded configuration

life.augmented

16

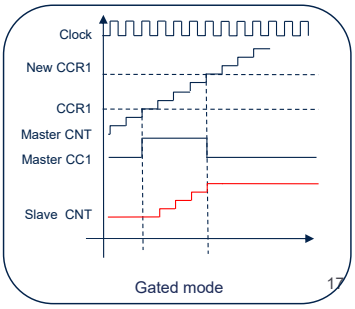
This slide presents the timer's synchronization features. The trigger controller allows you to cascade multiple timers in a master/slave configuration. A timer can control one or more timers as the master timer, or be controlled by another timer as a slave. The Clock and Trigger controller acts as a link between the timers. In Master mode, it can redirect outside the timer, multiple internal control signals, to an on-chip TRGO trigger output. In Slave mode, it gathers multiple inputs on the TRGI (the main trigger input) coming from the external trigger pin (ETR) or from one of the four internal trigger inputs ITR1 to ITR4, connected to the other TRGO outputs. Additionally, the input capture 1 and 2 pins can also be used as an internal trigger (typically to reset the counter).
Slave and Master modes can be programmed independently. A given timer can thus simultaneously be

operating in Slave and Master modes in a cascaded configuration, accepting input triggers while providing output triggers.

Cascading timers 2/2

Multiple signals can be shared across timers

- Master mode: the timer propagates internal signals on the TRGO output
  - Counter reset, counter enable, update event or OC1 compare match
  - Any of the waveforms generated with OC1 to OC4
- Slave mode: the timer is controlled with its TRGI input
  - Triggered mode: the counter start is controlled
  - Reset mode: a rising edge on TRGI reinitializes the counter
  - Combined reset & trigger mode (for retriggerable one-pulse mode)
  - Gated mode: Both the start and stop of the counter are controlled
  - Other modes related to clock:
    - 3 encoder modes
    - External clock



life.augmented

17

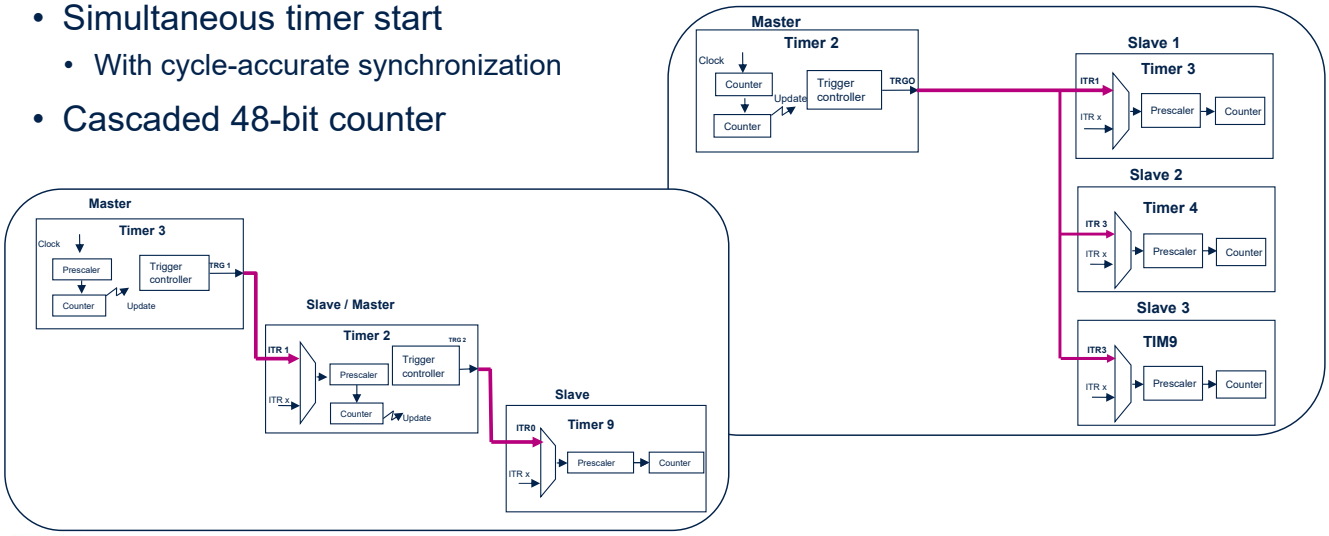
This slide lists the various operating modes and the signals exchanged between timers.
In Master mode, eight options are given for selecting the trigger to be sent on the TRGO output. The output can be a single synchronization pulse issued upon counter reset, counter enable which corresponds to the counter start, the update event or the compare 1 match event. Alternatively, the TRGO output can also transmit one of 4 waveforms generated, including PWM signals, to the other timer modules.
In Slave mode, the timer operating mode is controlled by the TRGI input. In Triggered mode, the counter start is externally controlled. This mode is used for simultaneously starting multiple timers. In Reset mode, the counter is reset by a rising edge on the TRGI input, typically for variable frequency PWM operation. A Combined mode

including reset and trigger can be used for re-triggerable one-pulse mode generation. In Gated mode, shown in the figure, the counter is active only while the level on the input signal is high. This signal either comes from an input or from another timer in Waveform Generation mode. In this case, synchronization pulses issued on reset, enable, update or compare match cannot be used. Lastly, the slave mode selection includes clock-related modes, such as quadrature encoder decoding or external clocking modes mentioned earlier in this presentation.

Examples of synchronized operation

Several timers can be combined for higher flexibility

- Simultaneous timer start
  - With cycle-accurate synchronization
- Cascaded 48-bit counter



This slide gives two examples of synchronized operation. The first example shows how four timers can be simultaneously started. A mechanism allows the master timer to start slightly delayed to compensate for the master/slave link delay, and have all timers synchronized with cycle accuracy. By combining the channels of Timers 2, 3, 4 and 9 as shown, it is possible to have up to 14 synchronized PWM channels.

The second example shows how to create a 48-bit timer by cascading three timers. Here the update event generated on counter roll-over is used as the input clock for the following slave timer, so that Timer 3's counter holds the least significant 16-bits, Timer 2's counter holds the medium bits (bits 16 to 31) and Timer 9's counter holds the upper bits from bit 32 to bit 47.

Motor control features

STM32 timers cover all aspects of motor drives

- PWM generation
  - Center-aligned and combined 3-phase mode
  - Dead time insertion
  - 6-step mode
- Protection (dual-break emergency stop mechanism)
- Speed & position sensing
  - Dedicated modes for encoders, Hall sensors, and tachometer generators
- ADC triggering

life.augmented

19

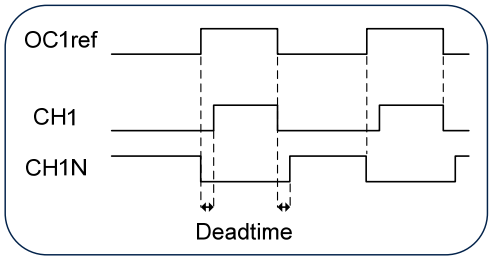
This slide summarizes the timer's 4 main electrical motor control features:
The timer includes specific PWM modes for controlling power switches. In addition to center-aligned and combined 3-phase PWMs previously described, the timer features dead time insertion for complementary PWM generation and 6-step mode for driving brushless DC motors.
It includes power stage protection circuitry with a dual-level emergency stop mechanism to disable the PWM outputs by hardware in case of a fault.
It is able to handle the most common sensors found in motor control systems. Quadrature encoders and Hall sensors are used for fine and coarse position feedback, while tachometer generators are used for cost-effective speed feedback and just require a Clear-on-capture mode.

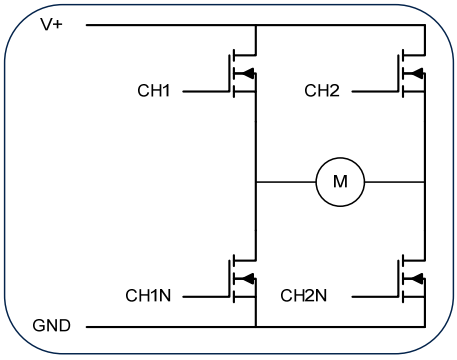
Lastly, the timer includes synchronized ADC triggering options, necessary to properly manage voltage and current sensing and avoid any acquisition issues due to switching noise in power stages.

Dead time insertion

Direct drive of up to 3 half-bridge converters per timer

- Hardware dead time unit generates non-overlapping complementary PWM signals
  - Ability to lock dead time register (read-only) for functional safety
- Prevents cross-conduction in half-bridge and full-bridge converters
  - DC/DC converter, DC motors, and 3-phase brushless motors





Full-bridge DC motor drive

life.augmented

20

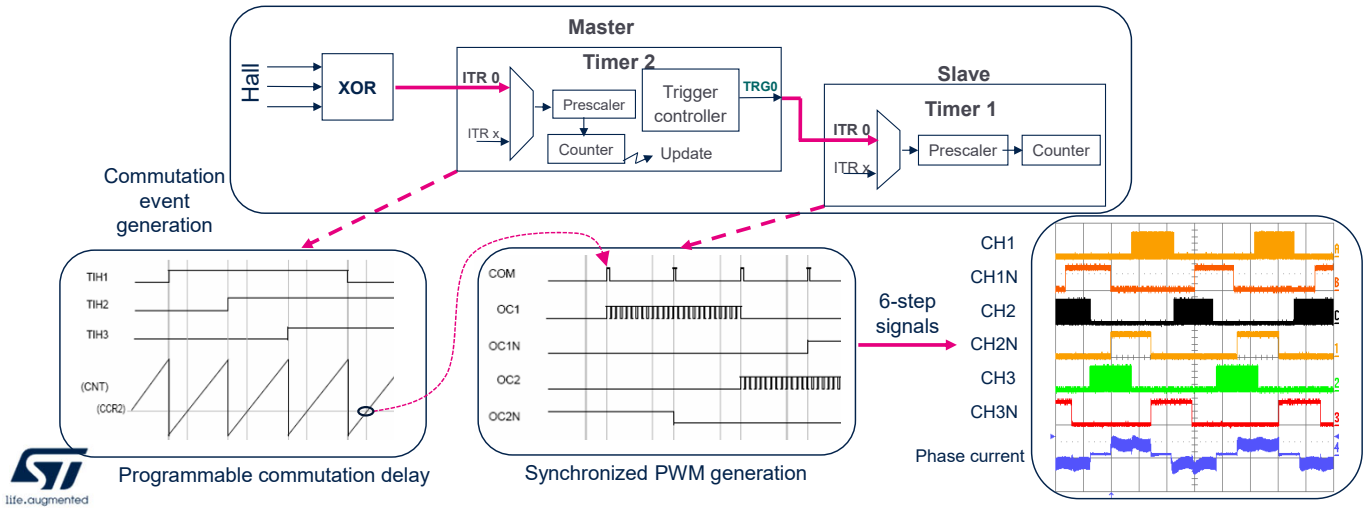
This slide presents the dead time insertion function.
A hardware dead time generator provides two non-overlapping complementary PWMs from a reference PWM signal. The STM32 timers includes up to three dead time generators for OC1, OC2 and OC3 channels. The dead time duration is programmed with an 8-bit value. This value can be locked by the user to prevent this critical value from being corrupted during run-time. This is done by setting a write-once lock bit which switches the dead time register into read-only mode until the next MCU reset. Dead time insertion is necessary when driving half-bridges, where a pair of transistors are connected in series between two power rails. In this case, it is necessary to insert some time before the switch on of one side to allow the other side to switch off, taking into account physical switching characteristics. Half-bridges are usually found in

DC/DC converters, for DC or stepper motor drive, using the full-bridge topology shown here or for 3-phase inverters, with three PWM pairs.

6-step / block commutation

Offload CPU for BLDC motor drive

- One timer can handle the Hall-sensor feedback and trigger an advanced timer for synchronized PWM generation



This slide shows how the 6-step drive (also called block commutation) is managed with the STM32 timer.
It consists of chaining two timers, one handling the three Hall sensor signals while the other manages the PWM generation synchronized with the rotor angular position, generating six successive steps.
The first timer operates in clear-on-capture mode, triggered by the three inputs. A compare register (here compare 2), is responsible for adding a programmable delay between the raw angular position and the commutation time. The capture register 1 holds the timing interval between successive Hall sensor edges and is necessary for the speed regulation loop.
The compare 2 match event is propagated to the slave timer through the TRGO output. These events serve as commutation events and trigger changes for PWM

generation. For each of the six steps of the sequence, the states of the six outputs are defined to be either forced active or inactive, or generating a PWM signal. The transition from one step to the other is preloaded by software, in the commutation interrupt routine, and automatically transferred by hardware to re-program the output operating mode when the next commutation arrives. The figure at right shows the six PWM signals for two consecutive, complete 6-step sequences, together with the current in one of the motor phases.

Break function 1/2

Best-in-class protection scheme

- A break event disables the PWM outputs
  - By hardware (minimal latency)
  - Asynchronous (clock-less operation, no clock-related delay)
  - Programmable safe state (High / Low or Hi-Z)
- Available on timers with complementary outputs
  - Timers 1 and 8
- 2 break channels
  - Allows dual-level protection schemes
    - All outputs OFF or some outputs forced ON and some OFF
    - Dead time insertion guarantees no risks of shoot-through

life.augmented

22

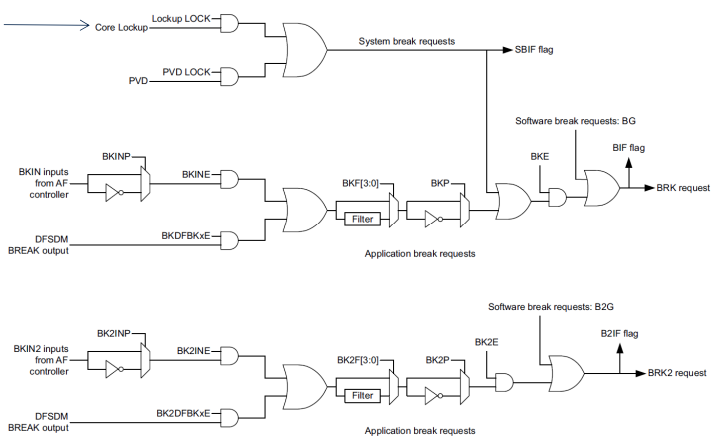
This slide presents the break function.
A break event triggers a hardware protection mechanism that automatically disables the PWM outputs, and forces them to a user-configurable state, either low impedance with high or low level, or high impedance. The logic circuitry works asynchronously, without any clock. This guarantees the functionality even in case of a system clock failure, and avoids any clock-related propagation time that would tend to delay the protection.
This feature is available on all timers having complementary PWM outputs, which are capable of performing power conversion tasks: Timers 1 and 8.
Timers 1 and 8 have two separated break channels. This provides a dual-level protection scheme, where for instance a low priority protection with all switches off can be overridden by a higher priority protection with low-side

switches active. Furthermore, a dead time delay can be inserted immediately before entering the fault mode for safely disabling the power stage. This prevents potential shoot-through conditions. Let’s consider for instance that the fault occurs when the high-side PWM is ON, while the safe state is programmed to have high-side switched OFF and low-side switched ON. At the time the fault occurs the system will first disable the high-side PWM, and insert a dead time before switching ON the low side.

Break function 2/2

Several emergency stop input sources

- Includes system break source from CSS (Clock Security System) and PVD (Programmable Voltage Detection)
- Digital filter for glitch removal
- Programmable polarity

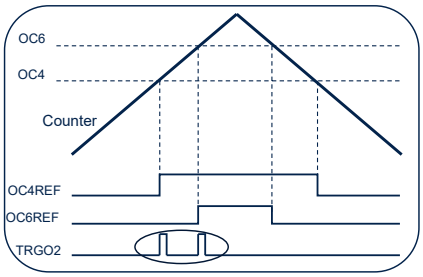


This slide presents how the break function sources are managed.
Multiple break sources can be combined for triggering a break event. Two system level sources can be selected: the Clock Security System (CSS) indicating an external clock failure, and the Programmable Voltage Detector (PVD) indicating a drop out on the Vdd supply or on the PVD_IN input.
Break inputs can also be selected with the alternate function controller, on the product pinout, or from the DFSDM Sigma Delta Demodulator output.
External sources can be conditioned before entering the break detection unit. This enables the selection of the proper polarity and discarding of spurious glitches by means of a digital filter.

ADC triggering

Multiple options for ADC triggers

- ADC trigger generation can be managed using:
  - Compare events
  - TRGO outputs
- Timers 1 and 8 also have an additional TRGO2 output dedicated to ADC
  - 16 possible triggering sources
  - Allows dual ADC triggers per PWM period
  - TRGO is free for synchronization purposes



Source	
TIM1_CC1 event	
TIM1_CC2 event	
TIM1_CC3 event	
TIM2_CC2 event	
TIM5_TRGO event	
TIM4_CC4 event	
TIM3_CC4	
TIM8_TRGO event	
TIM8_TRGO(2) event	
TIM1_TRGO event	
TIM1_TRGO(2) event	
TIM2_TRGO event	
TIM4_TRGO event	
TIM6_TRGO event	
EXTI line11	

life.augmented

24

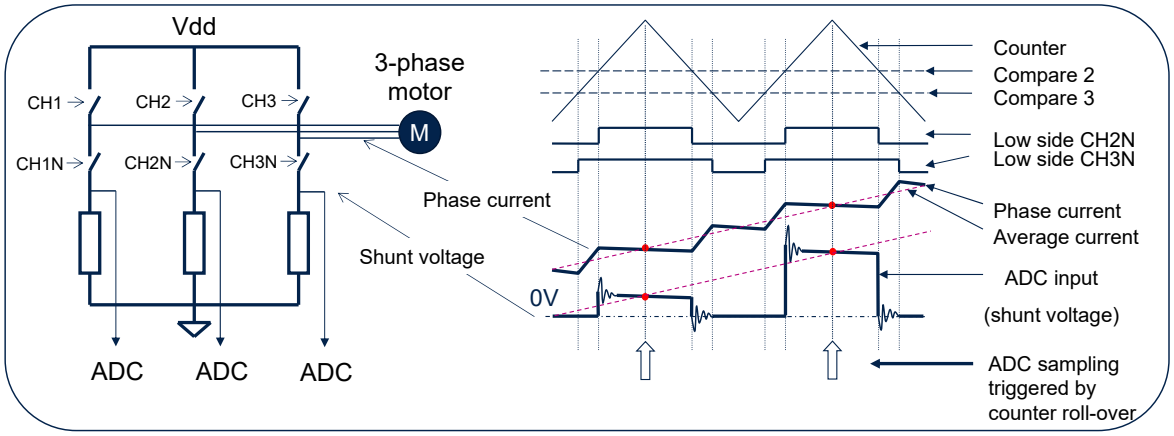
This slide presents the ADC triggering options related to the timers.
The ADCs can be triggered with most of the STM32 timers, with three options.
This can be done using compare events: the ADC conversion will start on a given compare match. The list of supported compare events varies from one timer to the other, as shown on the table.
The TRGO event can also be used on certain timers. This gives extra flexibility since the TRGO can be any of the compare events or timer internal control signals, such as register update, counter reset or trigger input. On the other hand, this prevents the TRGO from being used for synchronization purposes.
For this reason, Timers 1 and 8 also have an additional TRGO2 output, fully devoted to ADC triggering.

TRGO2 offers 16 possibilities, including the six compare events and the possibility to have a dual trigger per PWM period, by combining the compare 4 and 6 events. This also leaves the TRGO free for multiple timer synchronization schemes.

ADC synchronization example

Avoids PWM-related noise during ADC readings

- In 3-phase motor control applications, an ADC trigger on counter overflow allows you to obtain the average current value and avoids noisy ADC conversions



life.augmented

25

This slide presents an example of PWM-synchronized ADC trigger.
For 3-phase motor control, it is mandatory to have ADC readings synchronized with the PWM generated for controlling the power stage. This allows extraction of the average value out of the current waveform ripple, and makes sure the ADC reading is done at an adequate distance from the ringing due to the power switches.
Shown here on the left is a 3-phase motor inverter. The six switches are controlled by three complementary PWM pairs with dead time inserted, while the current in the motor windings is measured using shunt resistors placed in the three half-bridges' bottom side. The right side shows the timer's counter, compare 1 and compare 2 values and corresponding PWM outputs for the low-side switches controlled by CH1N and CH2N. The two bottom

waveforms represent the current in the motor phase and the image of this current obtained on the shunt resistors. With this low-cost topology, the voltage can only be measured when the low-side switches are ON, which explains the square-wave-shaped signal obtained on the ADC input. In this case, the ADC trigger is generated on the counter roll-over. This allows the reading to be done precisely in the middle of the period and get the average value of a signal with significant ripple. Additionally, using a PWM-synchronized ADC trigger also guarantees that the ADC conversion will be done away from the ringing noise present on the shunt voltages.

Interrupts and DMA

Event	Interrupt	DMA	Description
Update	**Yes**	**Yes**	Issued when the counter overflows or underflows or in case of forced software update request
Capture/Compare 1 **Capture/Compare 2** **Capture/Compare 3** **Capture/Compare 4**	**Yes**	**Yes**	Issued on compare match or when a capture is triggered. Each Capture / Compare channel has its own interrupt and DMA enable bit and flag
Trigger	**Yes**	**Yes**	Issued upon trigger event (from internal trigger inputs ITRx, TI1 Edge detector, filtered TI1/TI2 or external trigger input pin)
Com	**Yes**	**Yes**	On Timers 1 and 8 only
Break	**Yes**	**No**	

life.augmented

This slide lists the interrupts and DMA requests sources. Most of events are able to generate either an interrupt or a DMA request, and even the two simultaneously. The update is issued when the counter overflows or underflows. It is mainly used to refresh the timer's run-time settings at the beginning of the PWM period and maximize the interval before the next register update. The repetition counter allows you to skip some PWM periods and decrease the number of interrupts or DMA requests at high PWM frequency.

Each of the four capture/compare events have their own interrupt and DMA. A trigger event on the TRGI input (regardless of the trigger source) can also trigger an interrupt or DMA request.

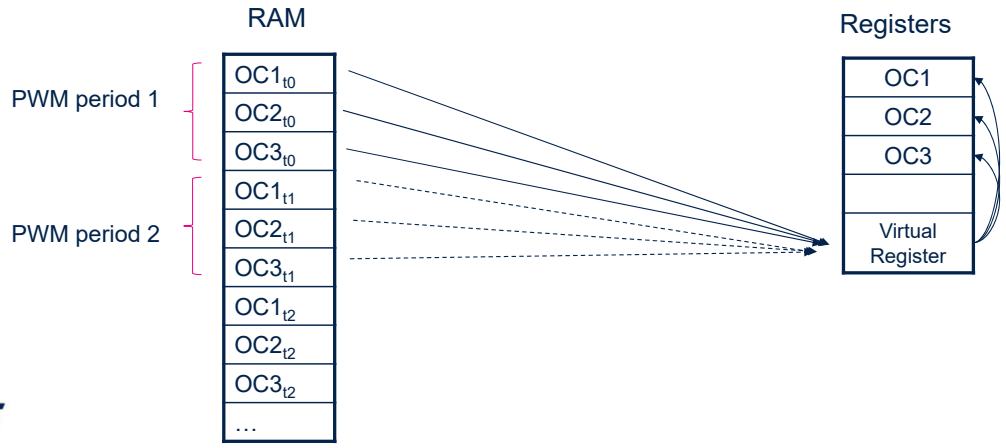
Lastly, additional sources of interrupts and DMA requests are the commutation and break events on Timers 1, 8, 15,

16 and 17 only. Note that the break event does not generate DMA requests.

DMA burst mode

Allows timer reconfiguration on-the-fly

- Allows the update of several registers with a single DMA event
  - Efficient use of DMA (a single stream is required)



life.augmented

27

The timer includes a DMA burst mode to have multiple registers re-programmed with a single DMA stream. This allows the modification of several run-time parameters simultaneously (for instance duty cycle and frequency of several channels) or dynamically change the timer configuration by writing the configuration registers.
The example shows how a table containing three compare values can be transferred into the compare registers with a single DMA stream when a new PWM period starts.
The DMA must be programmed in memory to peripheral mode, pointing to a unique location in the timer (virtual register TIMx_DMAR). When the update event occurs, the timer sends a number of DMA requests corresponding to the programmed burst length. Each value is then automatically redirected from the virtual register into the active register targeted.

On the next update event, three new compare values are transferred again. In this example, this mechanism saves two DMA streams that would normally be necessary for such an update scheme.

Low-power modes

Mode	Description
Run	Active.
Sleep	Active. Peripheral interrupts cause the device to exit Sleep mode.
Stop	Frozen. Peripheral registers content is kept.
LP-Stop mode	Frozen. Peripheral registers content is kept.
LPLV-Stop mode	Frozen. Peripheral registers content is kept.
Standby	Powered-down. The peripheral must be reinitialized after exiting Standby mode.

life.augmented

The timer is active in the Run and Sleep modes, while it is frozen in Stop, LP-Stop and LPLV-Stop modes: the timer state and register content are preserved and the timer directly resumes operation when the MCU is woken up.
In Standby mode, the timer is powered-down and must be completely re-initialized when exiting from this mode.

Debug

- **Allows safe debugging of power conversion applications**

- For each timer, 2 configuration bits in the DBGMCU module are used to configure how the timer behaves when one core enters Debug mode
  - 1 TIMx bit in DBGMCU_APB1FZ1 (Cortex®-A7 core halted)
  - 1 TIMx bit in DBGMCU_APB1FZ2 (Cortex®-M4 core halted)
- TIMx = 0
  - The TIMx counter operation is maintained
- TIMx = 1
  - The clock of the counter is stopped when the selected core is halted
  - For timers having complementary outputs, the outputs are disabled

life.augmented

29

The timer's state in Debug mode can be configured with 2 configuration bits per timer, one for the Cortex-A7 and one for the Cortex-M4.

If the debug bit is reset, the timer clock is maintained during a breakpoint.

If the debug bit is set, the timer's counter is stopped as soon as the core is halted. Additionally, the outputs of the timers having complementary outputs are disabled and forced to an inactive state. This feature is extremely useful for applications where the timers are controlling power switches or electrical motors. It prevents the power stages from being damaged by excessive current, or the motors from being left in an uncontrolled state when hitting a breakpoint.

A few useful formulas 1/2

- PWM frequency set-up
  - Defined with auto-reload (ARR, in TIMx_ARR) and clock prescaler (PSC, in TIMx_PSC):

$$f_{PWM} = \frac{f_{TIM}}{(ARR + 1) \times (PSC + 1)}$$

  - Practically, one must start with PSC = 0 (no prescaler):

$$ARR = \frac{f_{TIM}}{f_{PWM} \times (PSC + 1)} - 1 \rightarrow ARR = \frac{f_{TIM}}{f_{PWM}} - 1$$

  - If it yields a value above the 16-bit (or 32-bit) range, PSC must be increased until ARR fits:

$$ARR = \frac{f_{TIM}/2}{f_{PWM}} - 1 \rightarrow ARR = \frac{f_{TIM}/3}{f_{PWM}} - 1 \rightarrow ARR = \frac{f_{TIM}/4}{f_{PWM}} - 1 \rightarrow \ ...$$

life.augmented

This slide explains how to set the timer's PWM frequency. This parameter is defined using the autoreload value (ARR) programmed in the TIMx_ARR register and the clock prescaler programmed in the TIMx_PSC register.
The PWM frequency is given by the timer operating frequency (f_{TIM}) divided by ARR+1 times the clock prescaler+1.
Practically, finding both register values is an iterative process, where one must start from PSC = 0, i.e. no clock division. This guarantees that the PWM will have the finest possible resolution.
In this case, the ARR value is simply the ratio between the timer clock frequency and the PWM frequency, the whole minus 1.
If this equation yields an ARR value above the timer's ARR range, either a 16-bit or 32-bit value depending on the

selected timer, the computation must be re-done with a higher prescaler value, with the following sequence:
An ARR value equal to timer clock frequency divided by two over the PWM frequency, the whole minus 1, then an ARR value equal to timer clock frequency divided by three over the PWM frequency, the whole minus 1, and so on up to the point where the ARR value fits within the programmable range.

A few useful formulas 2/2

- Duty cycle set-up
  - Defined with auto-reload (ARR, in TIMx_ARR) and compare values (PSC, in TIMx_CCRx):

$$Duty\ Cycle\ = \frac{CCRx + 1}{ARR + 1} \rightarrow CCRx = \left(Duty\ Cycle\ \times (ARR + 1)\right) - 1$$

- PWM resolution
  - The resolution gives the number of possible duty cycle values and indicates how fine the control on the PWM signal will be:

$$Res_{(steps)} = \frac{f_{TIM}}{f_{PWM}}$$

  - Another way of expressing it is in bits, as for giving a DAC converter output resolution:

$$Res_{(bits)} = log_2(\frac{f_{TIM}}{f_{PWM}})$$

life.augmented

31

This slide explains how to program a duty cycle for a given PWM frequency.

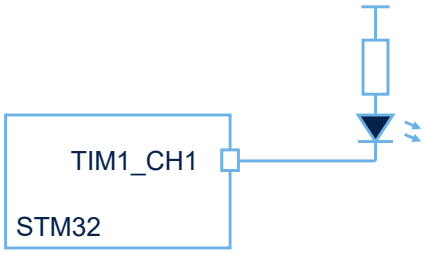
This parameter is defined using the autoreload value (ARR) programmed in the TIMx_ARR register and the compare value programmed in the TIMx_CCRx register. The duty cycle does not depend on the PWM frequency and is given by the compare value +1 over the autoreload value +1.

Another useful indication is the PWM resolution.
This gives the number of possible duty cycle values and indicates how fine the control on the PWM signal will be. The resolution, expressed in number of duty cycle steps, is simply equal to the ratio between the timer clock frequency and the PWM frequency, the whole minus 1.

Another way of expressing it is in bits, as for giving a DAC converter output resolution. In this case, the resolution is the base 2 logarithm of the ratio between the timer clock frequency and the PWM frequency, the whole minus 1.

Application examples: Dimming a LED

- This can be done directly using a PWM output, as long as the current does not exceed the rated output current
  - PWM frequency: 1 kHz
    - Frequency: $ARR = \frac{f_{TIM}}{f_{PWM}} - 1 = \frac{200MHz}{1kHz} - 1 = 199999$
    - ARR is above the maximum 16-bit value
    - → Prescaler must be set to 4
    - $ARR = \frac{f_{TIM}/5}{f_{PWM}} - 1 \rightarrow ARR = \frac{200MHz/5}{1kHz} - 1 = 39999$
  - Duty cycle at start = 20%
    - $Duty\ Cycle = \frac{CCRx+1}{ARR+1} \rightarrow CCRx = \big((ARR+1) \times Duty\ Cycle\big) - 1 = \big((40000) \times 0.2\big) - 1 \cong \mathbf{7999}$
  - Dimming resolution
    - 40000 steps or $\log_2(40000)$ = 15.3 bit



life.augmented

This slide shows a simple practical example of PWM usage, for dimming a low-power LED.
This can be done directly using a PWM output, as long as the current does not exceed the rated output current.
The 1st step is to program the frequency, to be set to 1 kHz. When doing the ARR value computation with no prescaler and a timer operating frequency of 200 MHz, the result is 199999, which is above the 16-bit range that can be used with Timer 1.

The timer prescaler must be set to 4 to have the timer operating at 40 MHz and this results in a valid value of 39999 for the ARR register.
The second step consists of computing the Compare register value to have a 20% duty cycle. This yields a value of 7999.

Lastly, the dimming resolution can be computed from formulas presented in the previous slides. With a timer running at 40 MHz, a 1 kHz PWM provides 40000 dimming steps, which corresponds to an equivalent resolution of 15.3 bits.

Application tips and tricks

- The whole timer is configured, the counter is started, the PWM mode is enabled, as well as the corresponding outputs, but still no activity on the pins…

  → Did you consider setting the MOE bit and the CCxE bits?

- For timers equipped with dead time generators (Timers 1 and 8), a Main Output Enable (MOE) bit in the TIMx_BDTR registers controls all outputs and acts as a circuit breaker in case of fault detection on the break input (global disable of all PWM outputs)

  → The MOE bit must be set (armed) to have the outputs enabled

  → This is valid even if the timer is used without dead time insertion

life.augmented

33

This slide explains a common support case, where the whole timer is configured, the counter is started, the PWM mode is enabled, as well as the corresponding outputs, but still there's no activity on the pins.

Usually, this is because the MOE bit or the CCxE bit was not set.
The CCxE bit in the TIMxCCER register defines the configuration of a CCx channel as input or output. The CC1E bit must be set to get a PWM signal on the CH1 channel.
For timers equipped with dead time generators (Timers 1, 8, 15, 16 and 17), a Main Output Enable (MOE) bit in the TIMx_BDTR registers controls all outputs and acts as a circuit breaker in case of fault detection on the break input (global disable of all PWM outputs).

The MOE bit must be set (armed) to have the outputs enabled.
This is valid even if the timer is used without dead time insertion, and the timer is used for general-purpose applications.

Related peripherals

- Refer to the training material for the following peripherals linked to the timers:
  - ADC
    - Timer is triggering injected and regular conversions
    - PWM can be stopped by analog watchdogs
  - DAC
    - Timer is triggering conversions

life.augmented

The timer is linked with multiple on-chip peripherals. It serves as a trigger source for the ADC and the DAC converter.

STM32MP1 instances features

	Counter resolution	Counter Type	PWM Mode	Dead time and Break input	DMA	Capture Compare channels	Synchronization	
							Master	Slave
Advanced control TIM1 and TIM8	16 bit	Up/Down	Standard + combined + asymmetric	Yes	Yes	6	Yes	Yes
General-purpose TIM2 and TIM5	32 bit	Up/Down	Standard + combined + asymmetric	No	Yes	4	Yes	Yes
General-purpose TIM3 and TIM4	16 bit	Up/Down	Standard + combined + asymmetric	No	Yes	4	Yes	Yes
General-purpose TIM12	16 bit	Up	Standard + combined	No	No	2	Yes	Yes
General-purpose TIM13, TIM14	16 bit	Up	Standard	No	No	1	Yes	No
General-purpose TIM15	16 bit	Up	Standard + combined	Yes	No	2	Yes	Yes
General-purpose TIM16, TIM17	16 bit	Up	Standard	Yes	No	1	Yes	No
Basic TIM6 and TIM7	16 bit	Up	Standard	No	Yes	0	No	No

life.augmented

This slide lists the timer instances present in STM32MP1 microcontrollers.
Timers 1 and 8 are full-featured timers, motor control capable, including all PWM options and six compare channels for being able to generate simultaneously 3-phase PWM signals and have two independent ADC triggers.
Timers 2, 3, 4 and 5 are general-purpose timers, including all PWM modes, up-down counting capability and 4 channels. Timers 2 and 5 additionally offer a 32-bit counting range.
Timers 12, 13 and 14 are lite timers, with support for standard PWM only, with 1 or 2 channels and up-counting mode only. They complement the other timers whenever additional independent time bases are necessary.
Timers 15, 16 and 17 are also lite timers, with support for

standard PWM only, with 1 or 2 channels and up-counting mode only. They complement the other timers whenever additional independent time bases are necessary. They also have dead time insertion and break input features for driving simple power systems with only a single PWM pair. Lastly, Timers 6 and 7 are pure time bases with no outputs, used principally to trigger the DAC converters or to provide software time bases.

Thank you

life.augmented