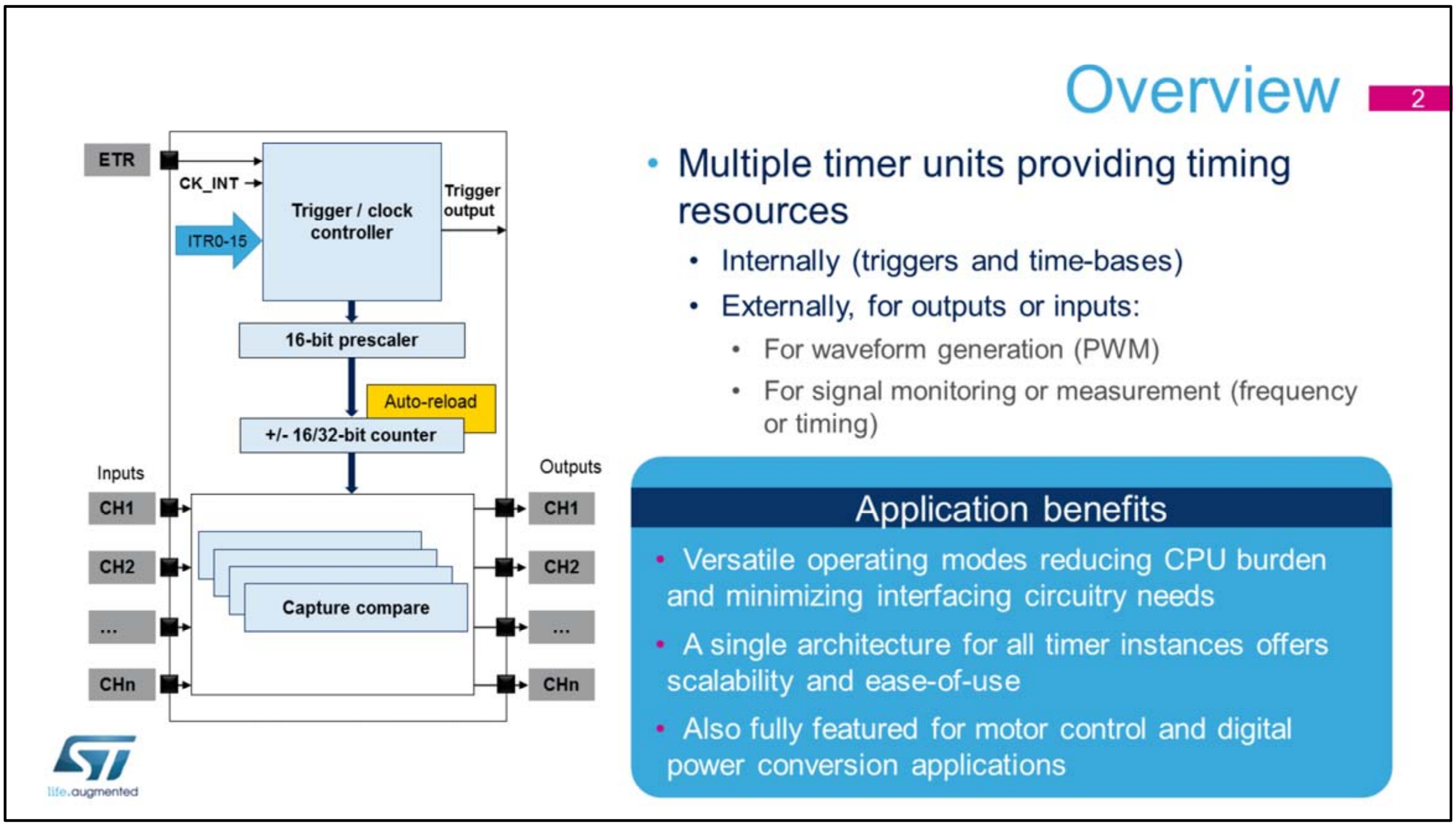
STM32G4 - GPTIM

Advanced-control, general-purpose and basic timers

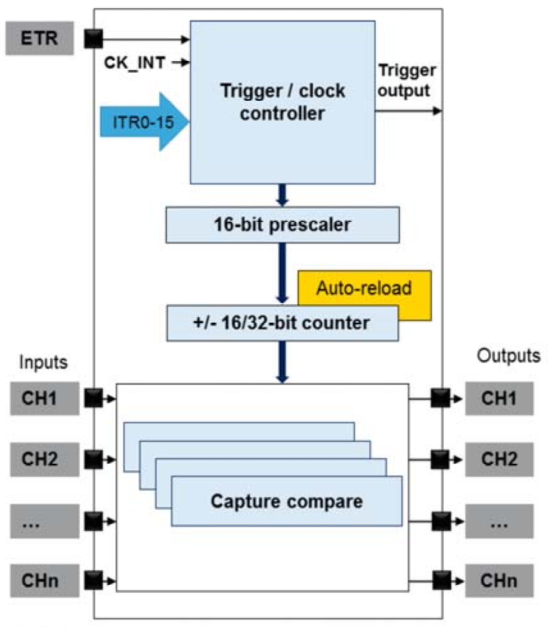
Revision 1.0

life.augmented

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)

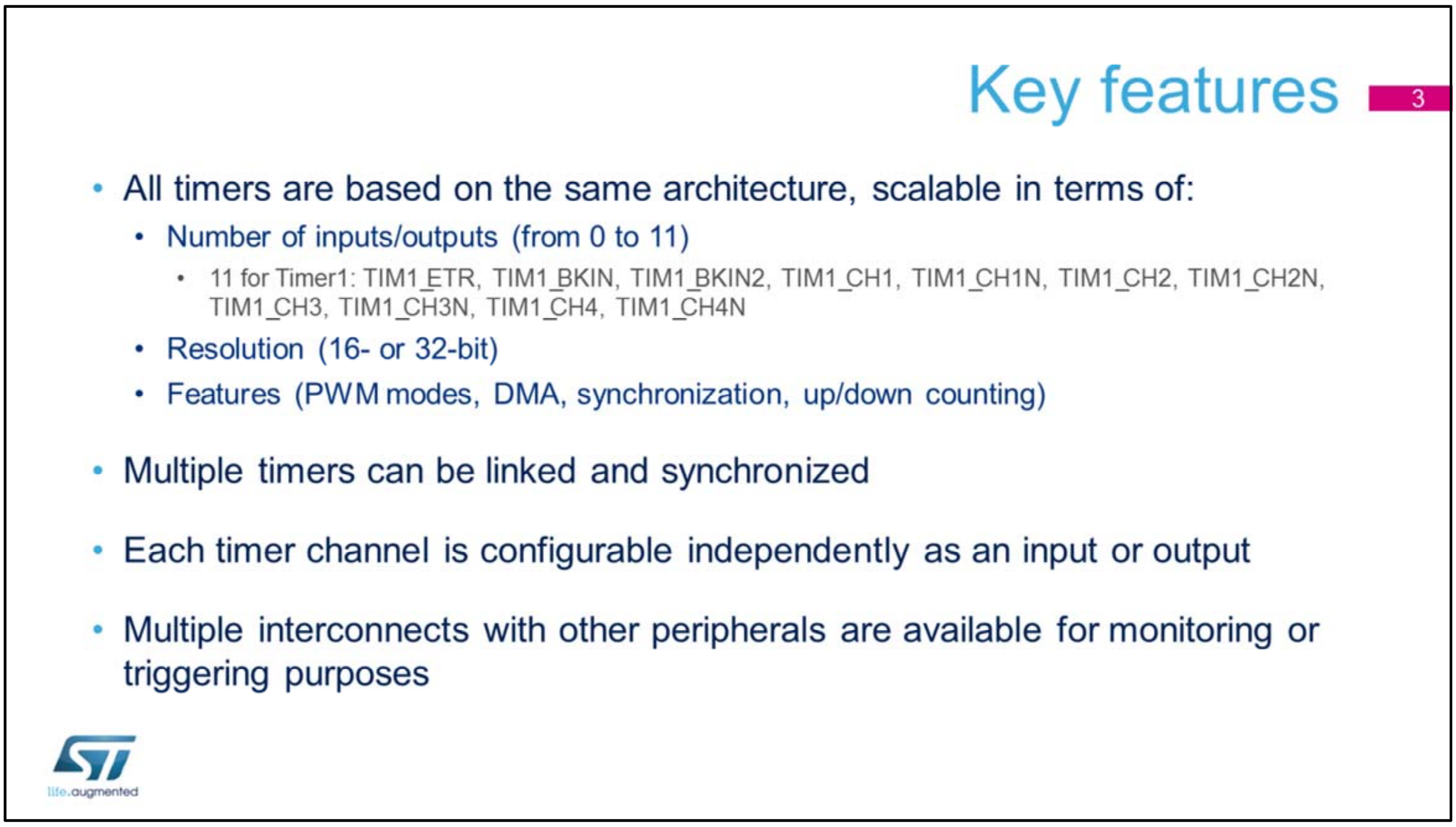
Application benefits

- 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

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 exception of the low-power timer and high resolution 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.



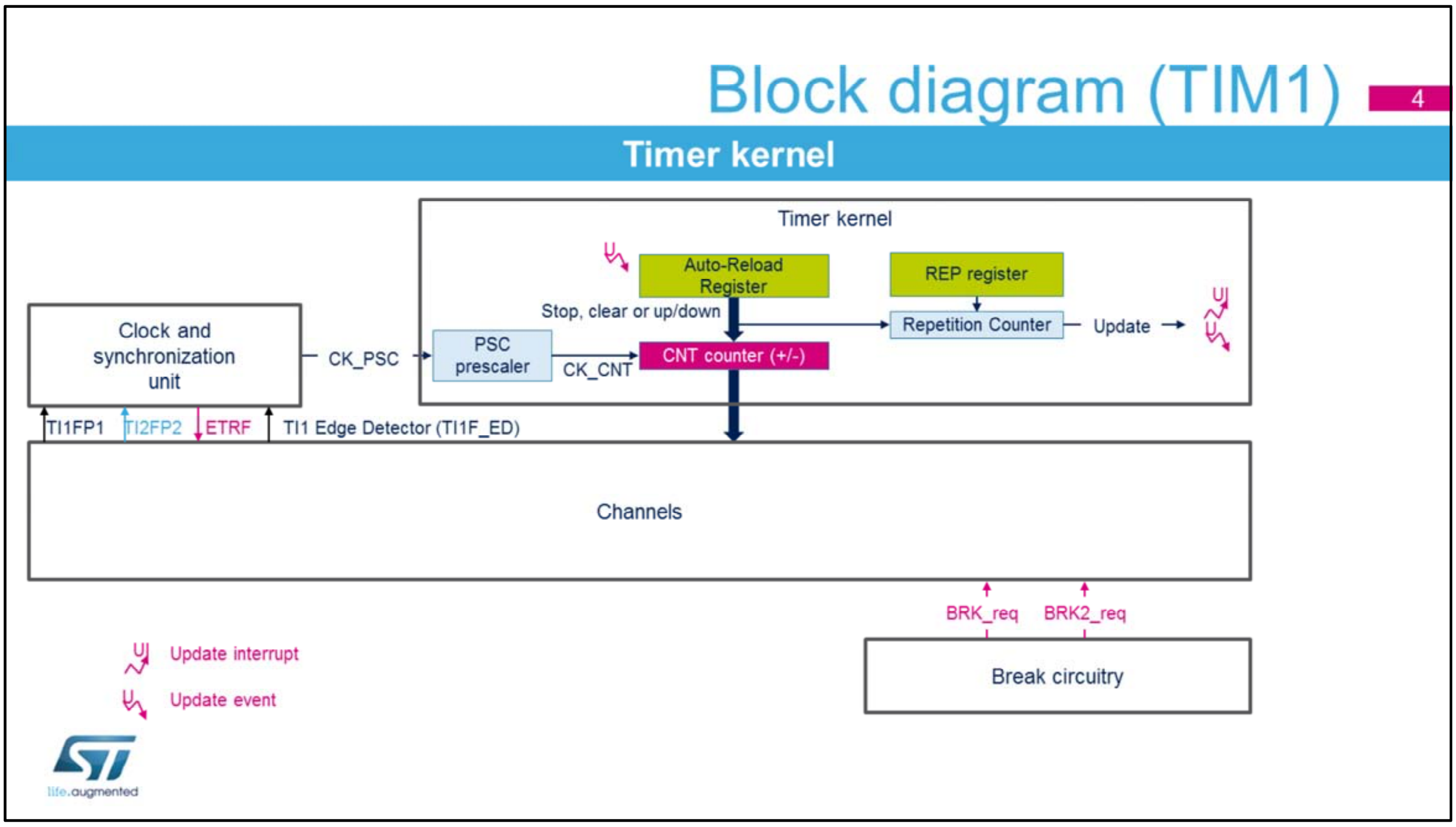
Key features

- All timers are based on the same architecture, scalable in terms of:
  - Number of inputs/outputs (from 0 to 11)
    - 11 for Timer1: TIM1_ETR, TIM1_BKIN, TIM1_BKIN2, TIM1_CH1, TIM1_CH1N, TIM1_CH2, TIM1_CH2N, TIM1_CH3, TIM1_CH3N, TIM1_CH4, TIM1_CH4N
  - 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

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 11 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 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 (TIM1)

4

Timer kernel

This slide presents the block diagram of the TIM1 timer, which has the same architecture as TIM8 and TIM20.
The timer kernel consists of a 16-bit up/down counter, coupled with an auto-reload register to program the counting period.
If the repetition counter is used, the update event is generated after upcounting is repeated for the number of times programmed in the repetition counter register.
Else the update event is generated at each counter overflow.
In order to update the contents of configuration registers, a preload register is implemented.
The content of the preload register is transferred into the shadow register permanently or at each update event.
For instance, this synchronization mechanism may be used to update the contents of the Auto-Reload Register present in this figure.

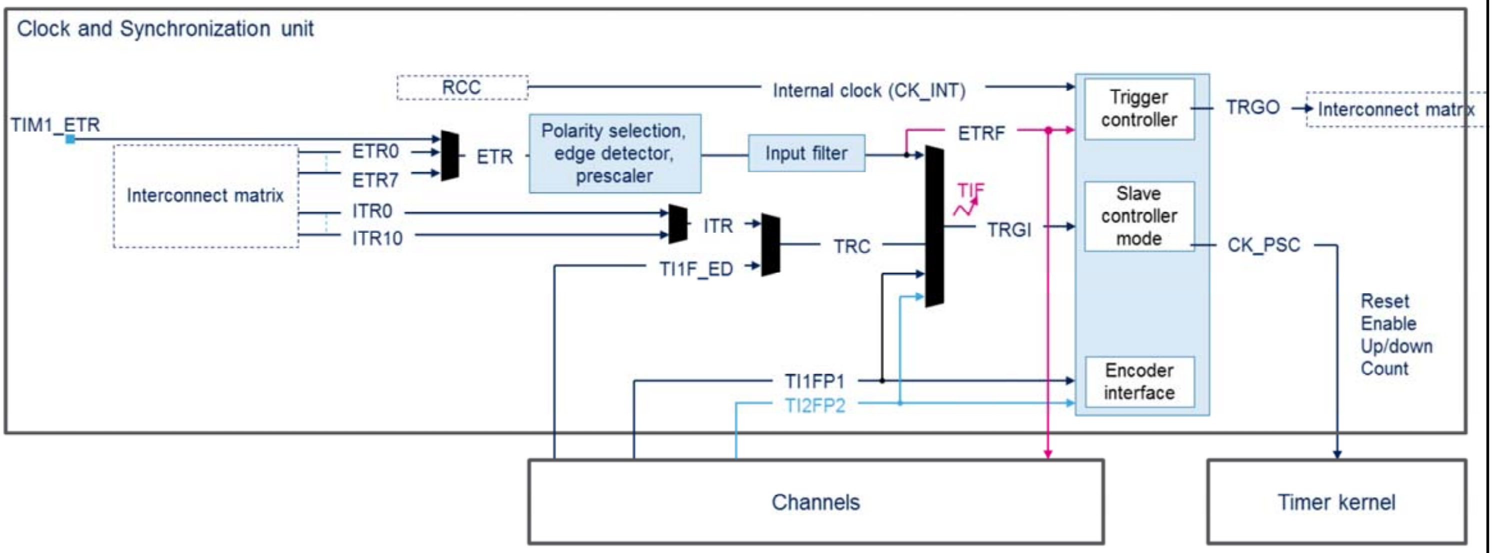
The other units of the timer block are:

- The clock and synchronization unit in charge of providing the timer clock and triggers
- The channels in charge of, input capture, output compare, PWM generation and One-pulse mode output
- The break circuitry in charge of putting the timer’s output signals in a safe user selectable configuration.

These other units will be described in the next slides.

Block diagram (TIM1)

Input stage and signal conditioning



The clock and synchronization unit provides the reference clock to the timer kernel, called CK_PSC.
The counter clock can be provided by the following clock sources

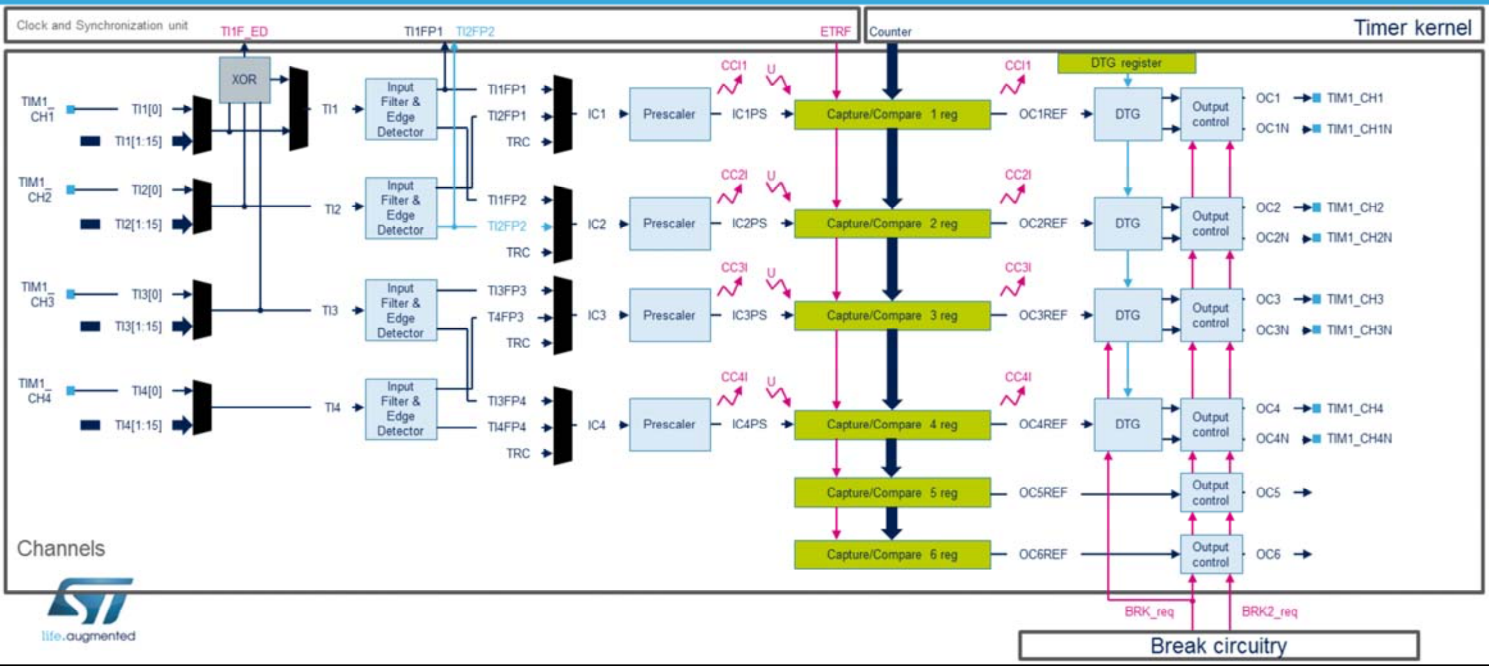
- The internal clock (CK_INT)
- External clock mode1: external input pin, counting each rising or falling edge on a selected input or internal triggers
- External clock mode2: external trigger input ETR or triggers
- Encoder mode, based on TI1FP1 and TI2FP2 coming from the channels.

ETR is a trigger that can be asserted by ADC analog watchdogs, comparators or ETR input.
ITR is a trigger that can be asserted by another timer.
Internal trigger in and out are connected to the Interconnect matrix, refer to the related presentation.

A rising edge of the selected trigger input (TRGI) sets the TIF flag.
When one timer is configured in Master Mode, it can reset, start, stop or clock the counter of another timer configured in Slave Mode.

Block diagram (TIM1) 6

Channels



Timer 1 supports 6 independent channels for:

- Input capture (except channels 5 and 6)
- Output compare
- PWM generation (Edge and Center-aligned Mode)
- One-pulse mode output.

When a capture or compare event occurs, the corresponding CCI flag is set.

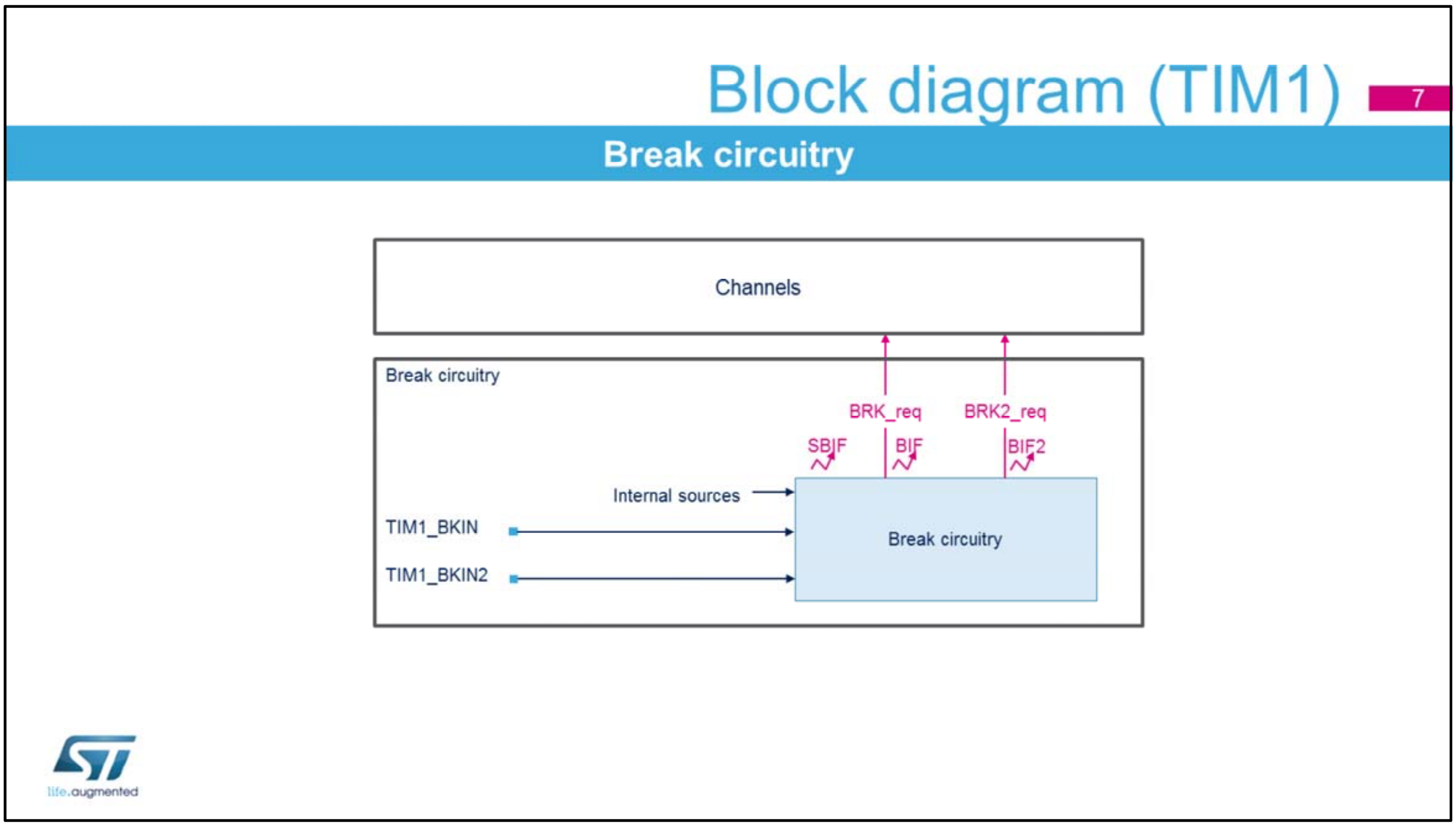
Shown on the left are the input stage and the input conditioning circuitry while on the right we have the output stage.

Note that TIM1CH1 to 4 pins appear on both sides to indicate they are both input and output capable.

Channels 1 to 4 can be output on pins, while channel 5 and 6 are only available inside the microcontroller (for instance, for compound waveform generation or for ADC triggering).

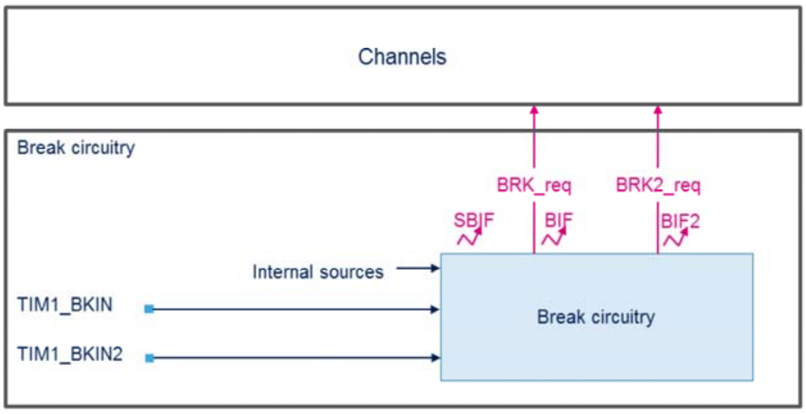
The Timer 1 supports break inputs that can be used to

automatically set the channel outputs in a predefined state whenever a break event occurs.
The Timer 1 also supports a Dead Time Generator unit (DTG) that inserts a dead time on PWM complementary outputs of channels 1 to 4.

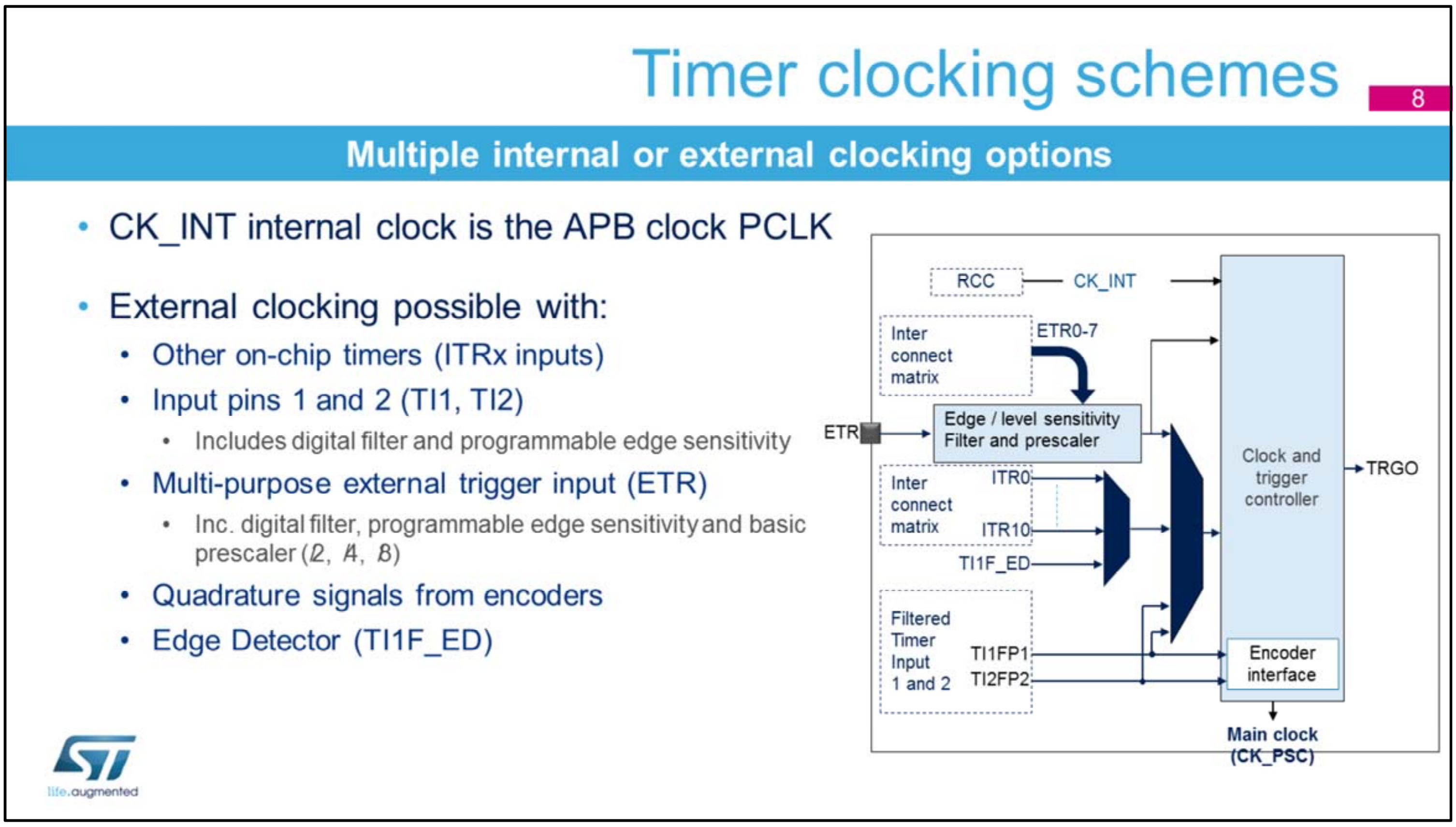


Block diagram (TIM1)

Break circuitry



The purpose of the break function is to protect power switches driven by PWM signals.
The two break inputs are usually connected to fault outputs of power stages and 3-phase inverters.
When activated, the break circuitry shuts down the PWM outputs and forces them to a predefined safe state.
A number of internal MCU events can also be selected to trigger an output shut-down.
The break features two channels.
A break channel which gathers both system-level fault (clock failure, parity error,...) and application fault (from input pins and built-in comparator), and can force the outputs to a predefined level (either active or inactive) after a dead time duration.
A break2 channel which only includes application faults and is able to force the outputs to an inactive state.

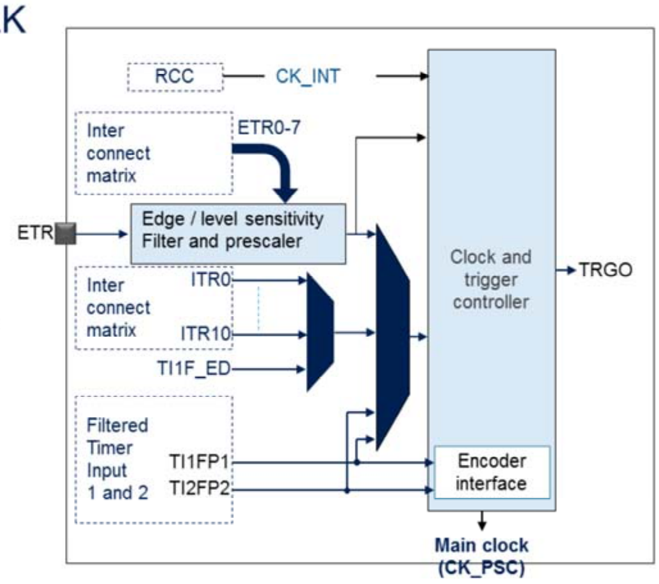


Timer clocking schemes

8

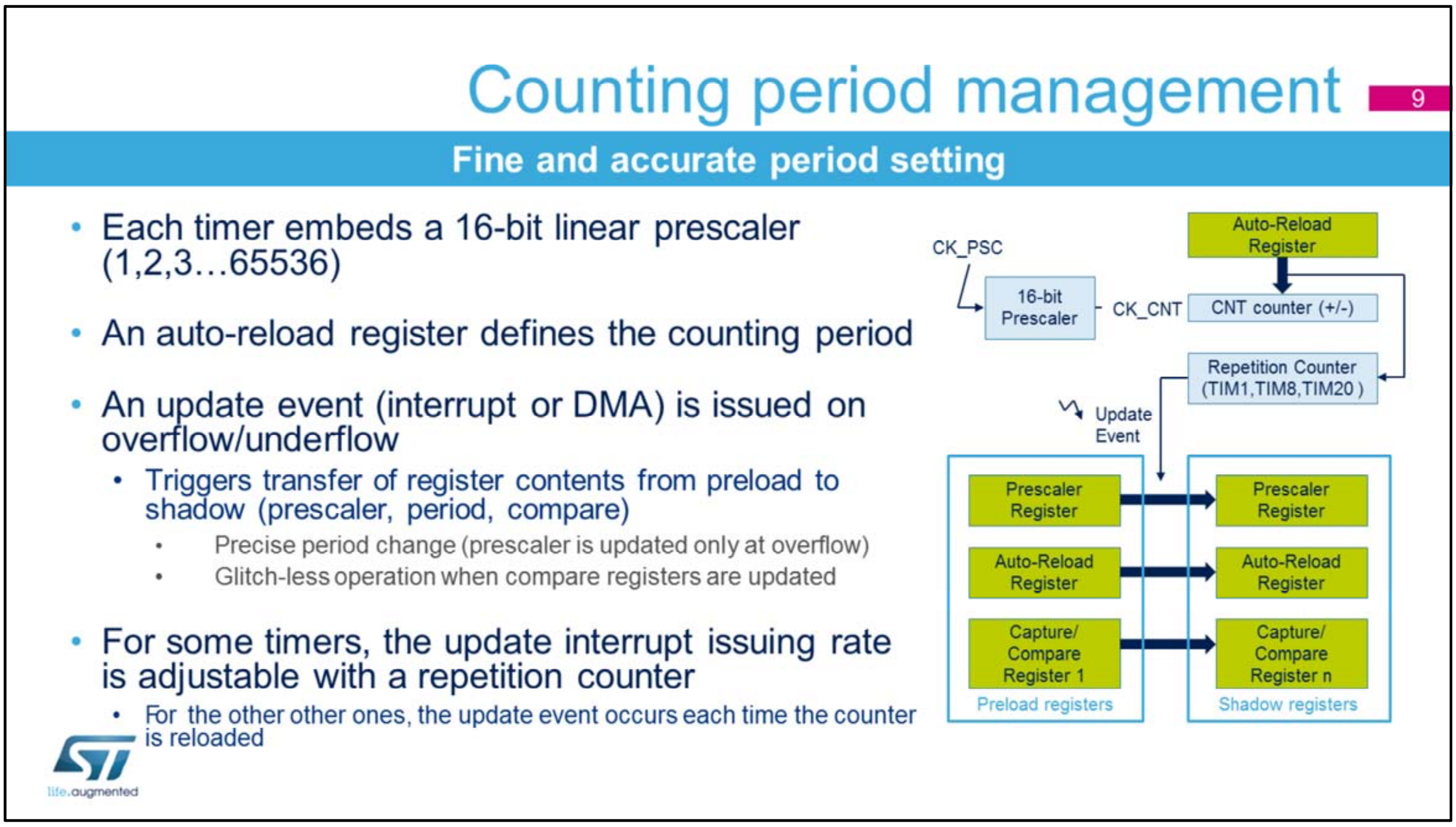
Multiple internal or external clocking options

- CK_INT internal clock is the APB clock PCLK
- External clocking possible with:
  - Other on-chip timers (ITRx inputs)
  - Input pins 1 and 2 (TI1, TI2)
    - Includes 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
  - Edge Detector (TI1F_ED)



Advanced control timers and general-purpose timers feature multiple clocking options.
The Clock and synchronization unit, also responsible for timer chaining, handles the clock for the counter.
The default clock comes from the Reset and Clock Controller.
This timer clock CK_INT is the APB clock PCLK.
External timer clocking makes it possible to count 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 11 internal trigger inputs (ITR0 to ITR10).
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

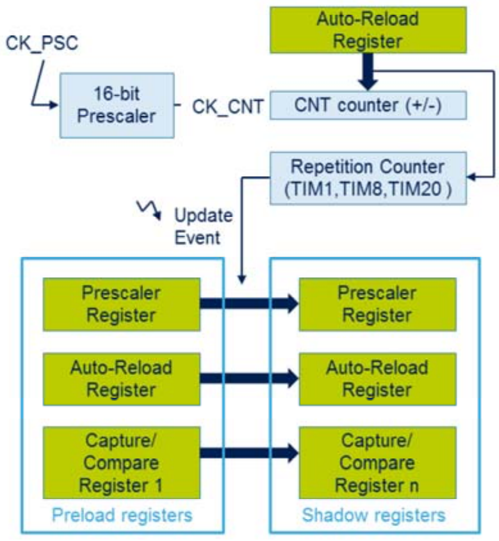
external clock, with a digital filter, programmable edge sensitivity and a first basic prescaler stage to reduce the frequency of incoming signals if needed.
External triggers can be generated by ADC watchdogs and comparators in addition to the ETR pin.
The TI1F_ED Edge Detector input can also be used as the timer clock.
A pulse is generated on both rising and falling edges of the TI1F signal.
Although it was not intended for this, it can serve as a frequency doubler with an external clock (count twice per incoming clock period).
Lastly, the signals from an encoder can be processed to provide a clock and a counting direction, as described later in this presentation.



Counting period management

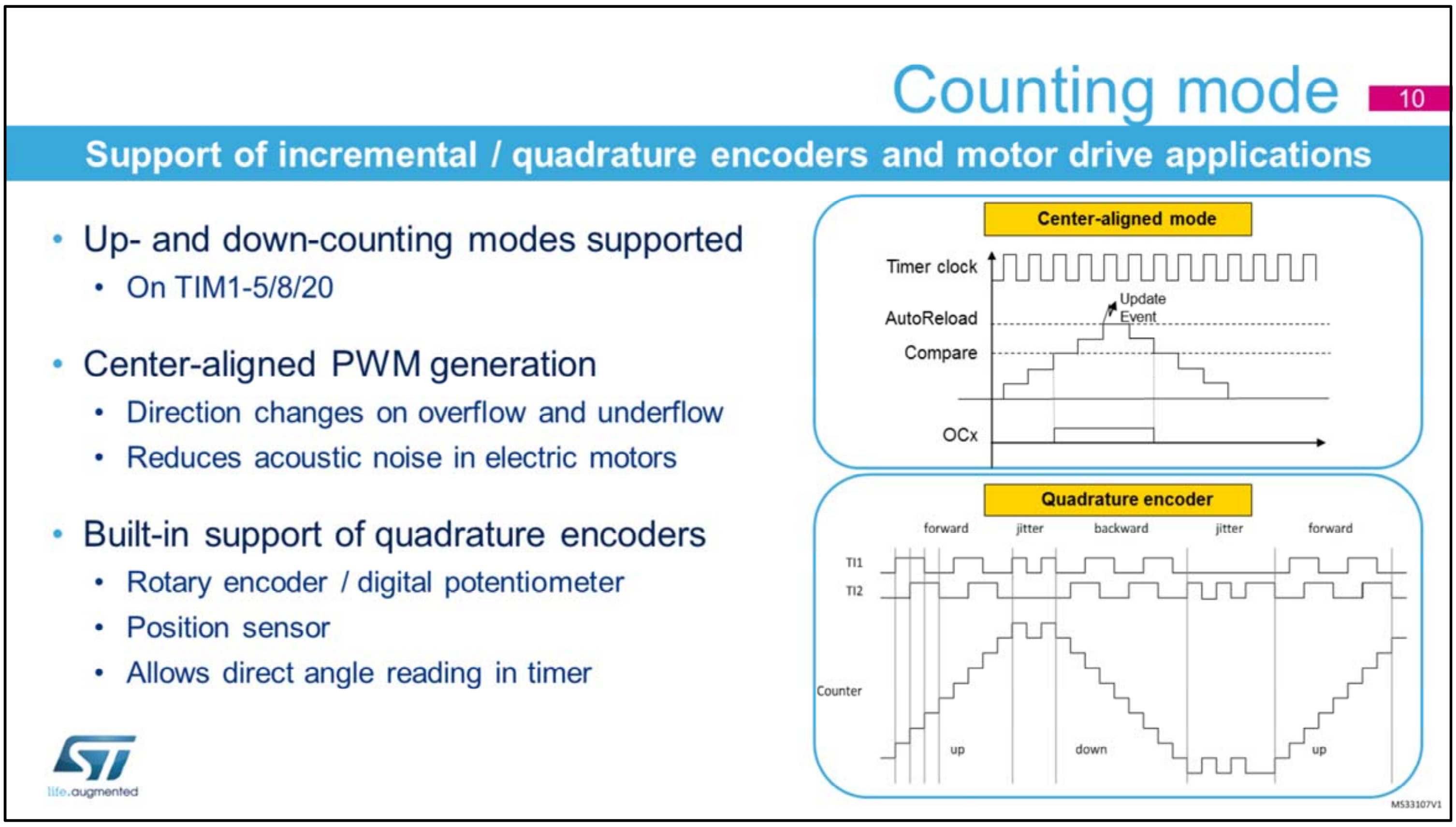
Fine and accurate period setting

- 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 shadow (prescaler, period, compare)
    - Precise period change (prescaler is updated only at overflow)
    - Glitch-less operation when compare registers are updated
- For some timers, the update interrupt issuing rate is adjustable with a repetition counter
  - For the other other ones, the update event occurs each time the counter is reloaded



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.
The counting pace can therefore be precisely adjusted.
For instance, a division by 64 will yield a precise 1 MHz counting rate when the APB clock is 64 MHz.
The auto-reload 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.
In center-aligned mode (up and down counting), the counter counts from 0 to the auto-reload value minus 1, generates a counter overflow event, then counts from the auto-reload value down to 1 and generates a counter

underflow event. Then it restarts counting from 0.
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 shadow registers for multiple parameters, and in particular for the clock prescaler, auto-reload value, compare registers and PWM mode.
A 16-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

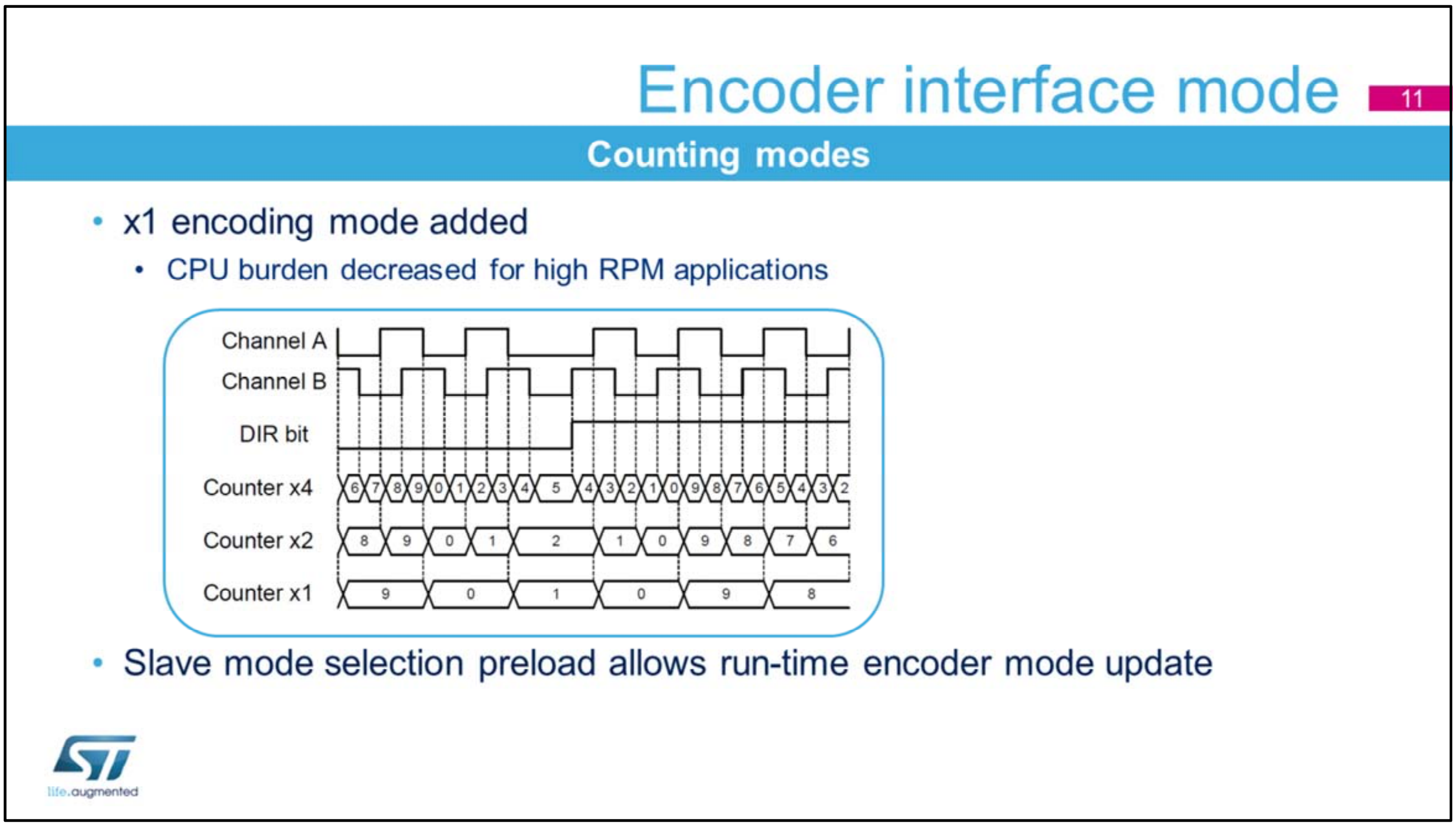
10

Support of incremental / quadrature encoders and motor drive applications

- Up- and down-counting modes supported
  - On TIM1-5/8/20
- 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

Some of the STM32 timers feature up/down counting modes: the advanced control timers 1/8/20 and the general-purpose timers 2/3/4/5.
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

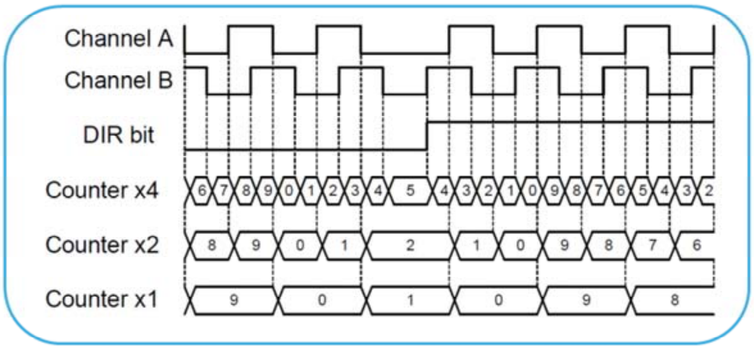
digital potentiometers.
From the two outputs of a quadrature encoder sensor (also called an incremental encoder), the timer extracts a clock on each and every active edge and adjusts the counting direction depending on the relative phase-shift between the two incomings signals.
The timer counter thus directly holds the angular position of the motor or the potentiometer.



Encoder interface mode

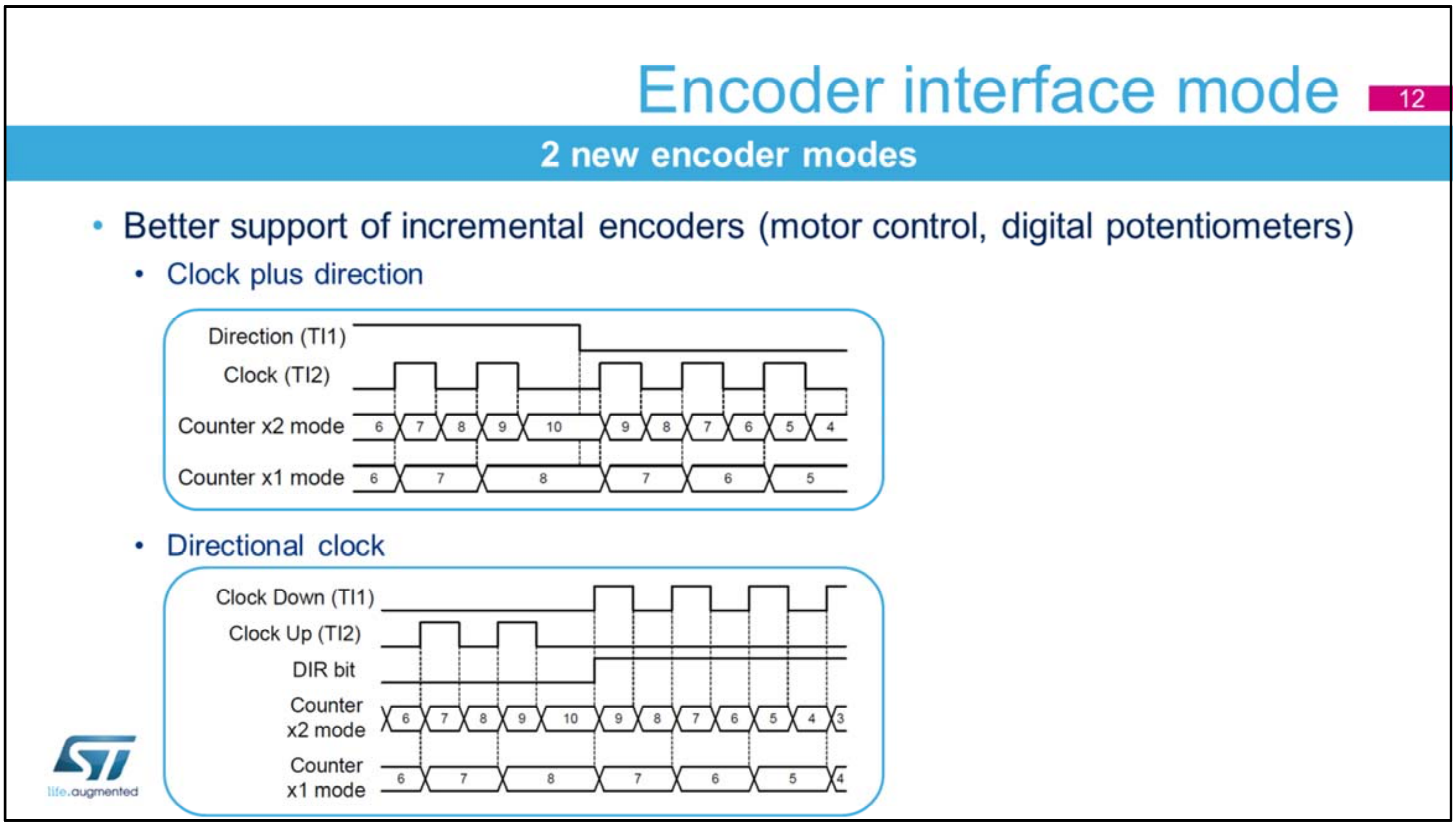
Counting modes

- x1 encoding mode added
  - CPU burden decreased for high RPM applications



- Slave mode selection preload allows run-time encoder mode update

The timers present in the STM32G4 implement a new counting mode when the encoder interface is used.
In addition to x2 and x4 modes, the x1 mode is supported.
As shown in the timing diagram, in x1 mode, the counter value is updated on a single clock edge, depending on the direction.
Here the falling edge of Channel A is used when DIR is equal to 0 and the rising edge of Channel A when DIR is equal to 1.
It may be necessary to switch from one encoder mode to another during run-time.
This is typically done at high-speed to decrease the Update interrupt rate, by switching from x4 to x2 to x1 mode.
For this purpose, the SMS field, which selects the mode, can be preloaded.



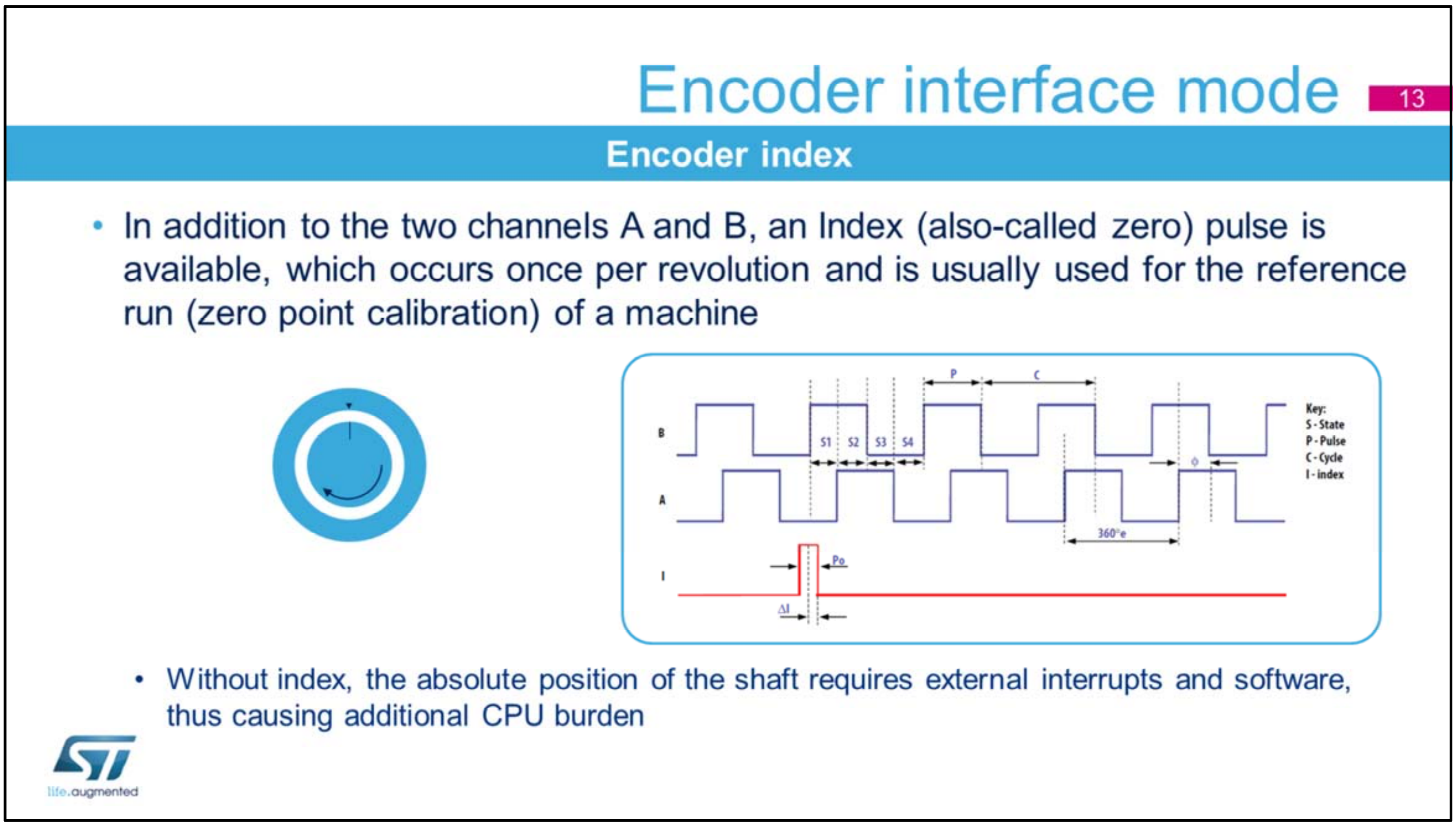
Encoder interface mode

12

2 new encoder modes

- Better support of incremental encoders (motor control, digital potentiometers)
  - Clock plus direction
  - Directional clock

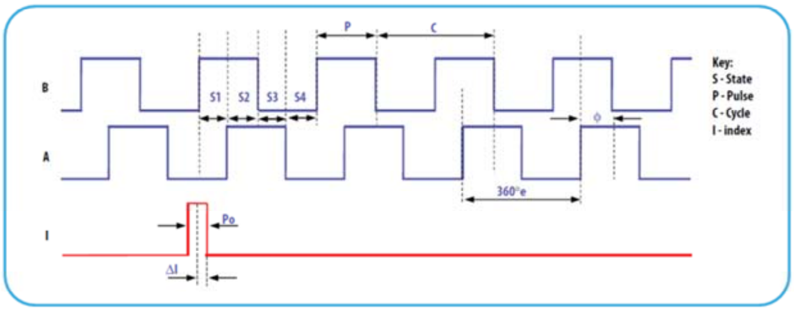
In addition to the quadrature encoder mode, the timer offers support other two types of encoders.
In the “clock plus direction” mode shown on the top timing diagram, the clock is provided on a single line, on TI2, while the direction is forced using the TI1 input.
In the “directional clock” mode shown on the bottom timing diagram, the clocks are provided on two lines, with a single one at once, depending on the direction, so as to have one up-counting clock line and
one down-counting clock line.



Encoder interface mode

Encoder index

- In addition to the two channels A and B, an Index (also-called zero) pulse is available, which occurs once per revolution and is usually used for the reference run (zero point calibration) of a machine

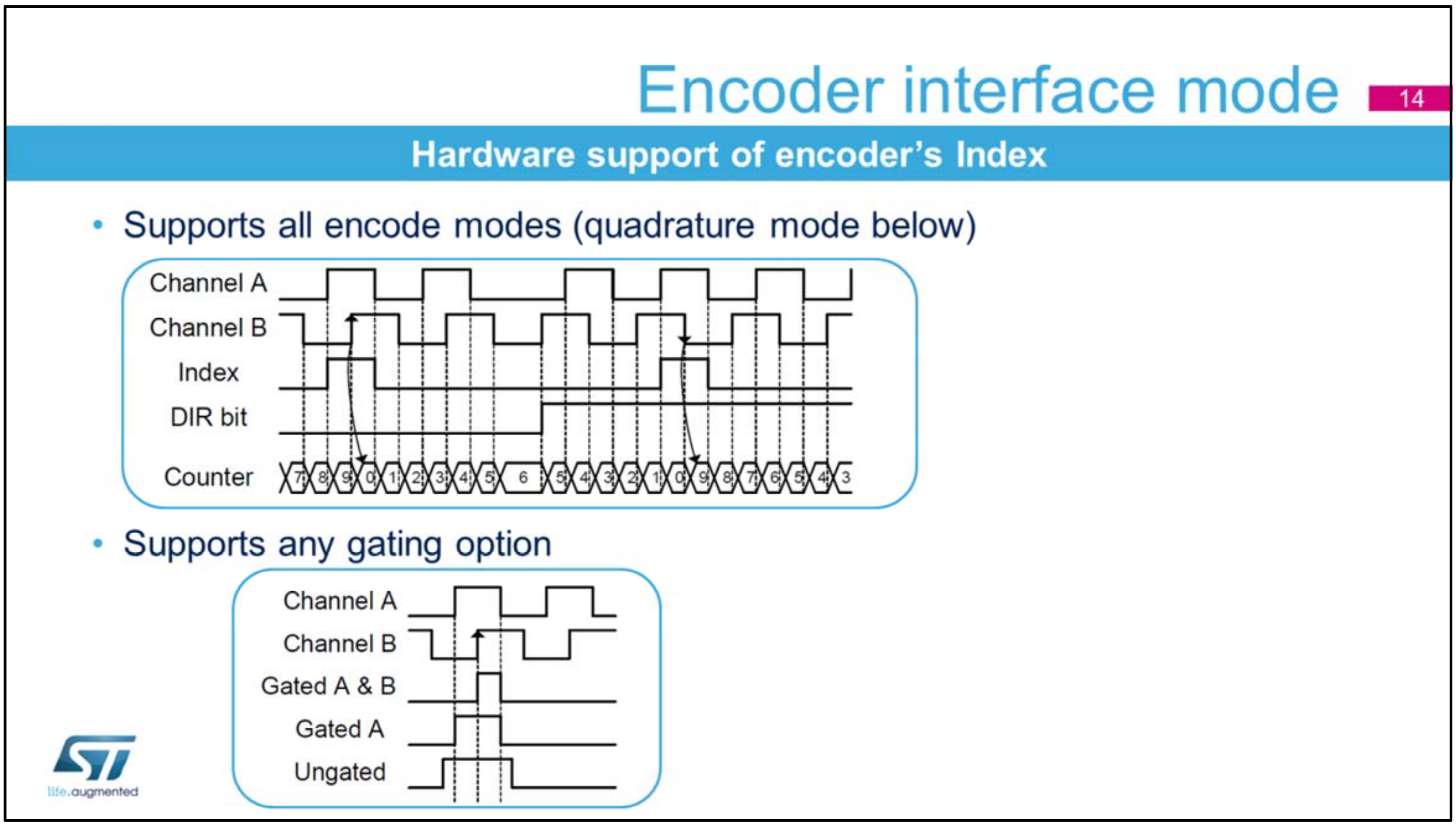


- Without index, the absolute position of the shaft requires external interrupts and software, thus causing additional CPU burden

Two types are encoder are used to driver motors:
- Incremental encoders do not provide a reference position
- Absolute encoders provide an index signal, indicating an absolute reference position.

The counter can be reset by the index signal.
The Index signal must be connected to the TIMx_ETR input.
It can be filtered using the digital input filter.
An encoder with index feature has three outputs: A and B used to control the incrementation / decrementation of the counter and the index, which is asserted once per revolution.
In the timing diagram, the zero pulse is generated when the shaft reaches a particular position.
Without an explicit index, interrupts and software are needed to calculate the index, thus causing additional

CPU burden.

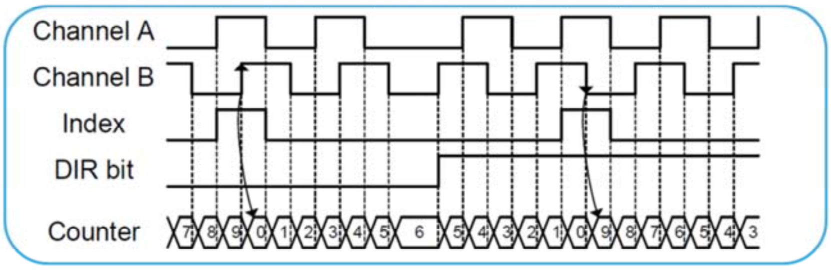


Encoder interface mode

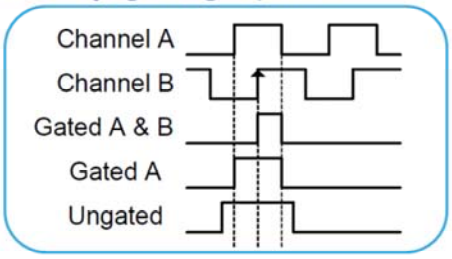
14

Hardware support of encoder's Index

- Supports all encode modes (quadrature mode below)

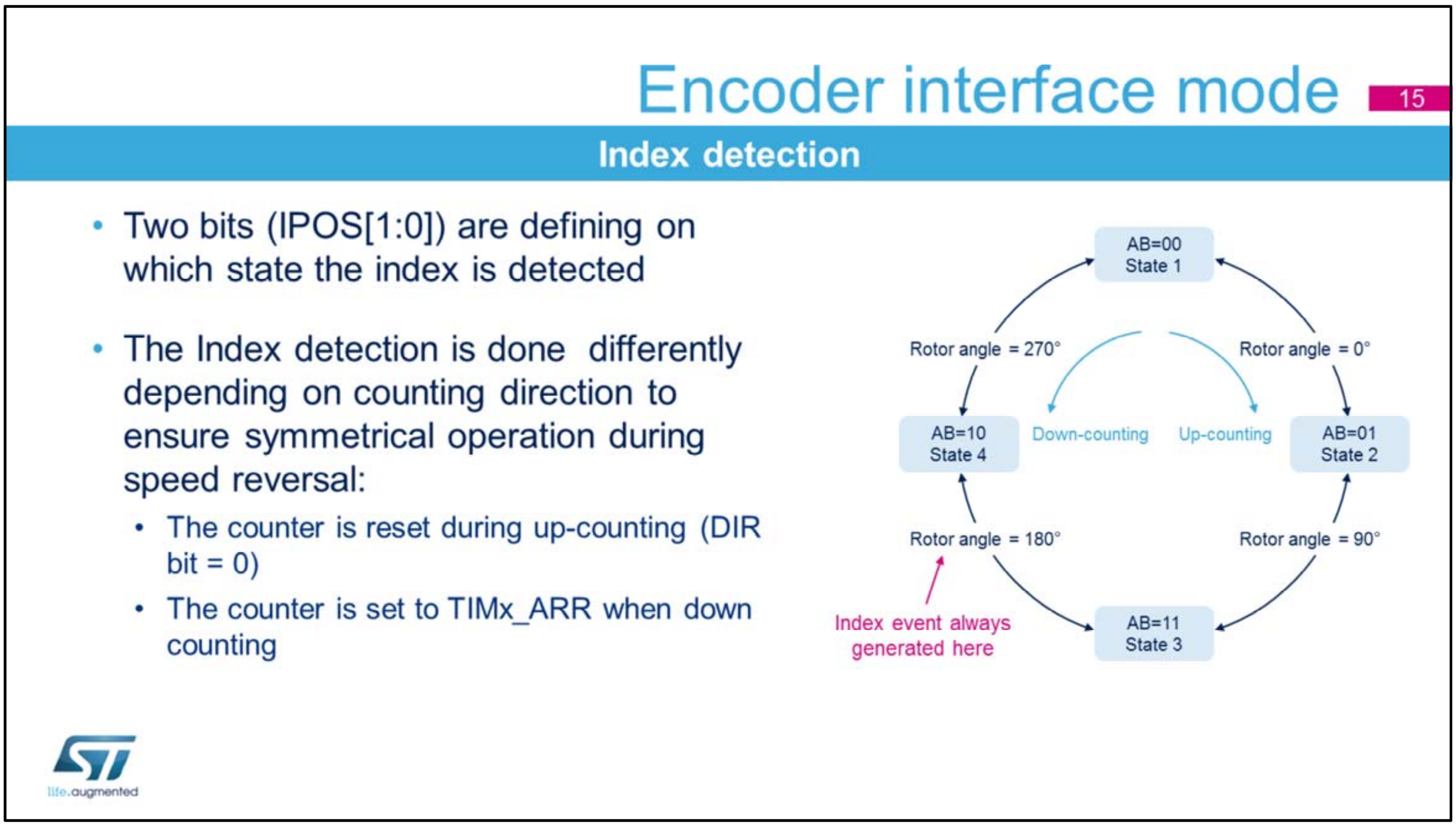


- Supports any gating option



life.augmented

The index feature is supported in all encoder modes: quadrature, clock plus direction, and directional clock. The upper timing diagram assumes quadrature mode is used.
When the index pulse is detected, the counter is reset. Commercially available encoders are proposed with several options for index pulse conditioning:
• Gated with A and B: the pulse width is 1/4 of one channel period, aligned with both A and B edges
• Gated with A (or gated with B): the pulse width is 1/2 of one channel period, aligned with the two edges on channel A (resp. channel B)
• Ungated: the pulse width is up to one channel period, without any alignment to the edges.

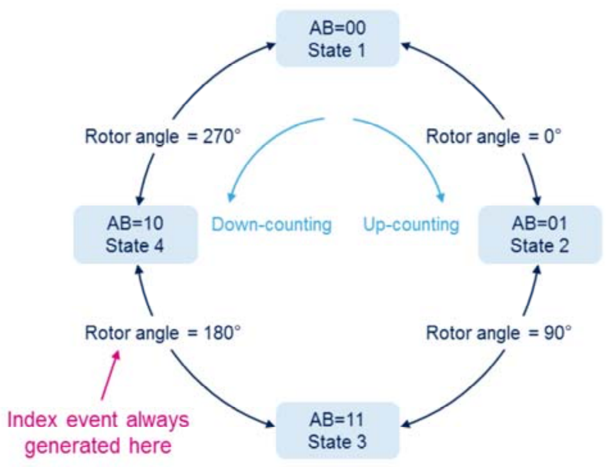


Encoder interface mode

15

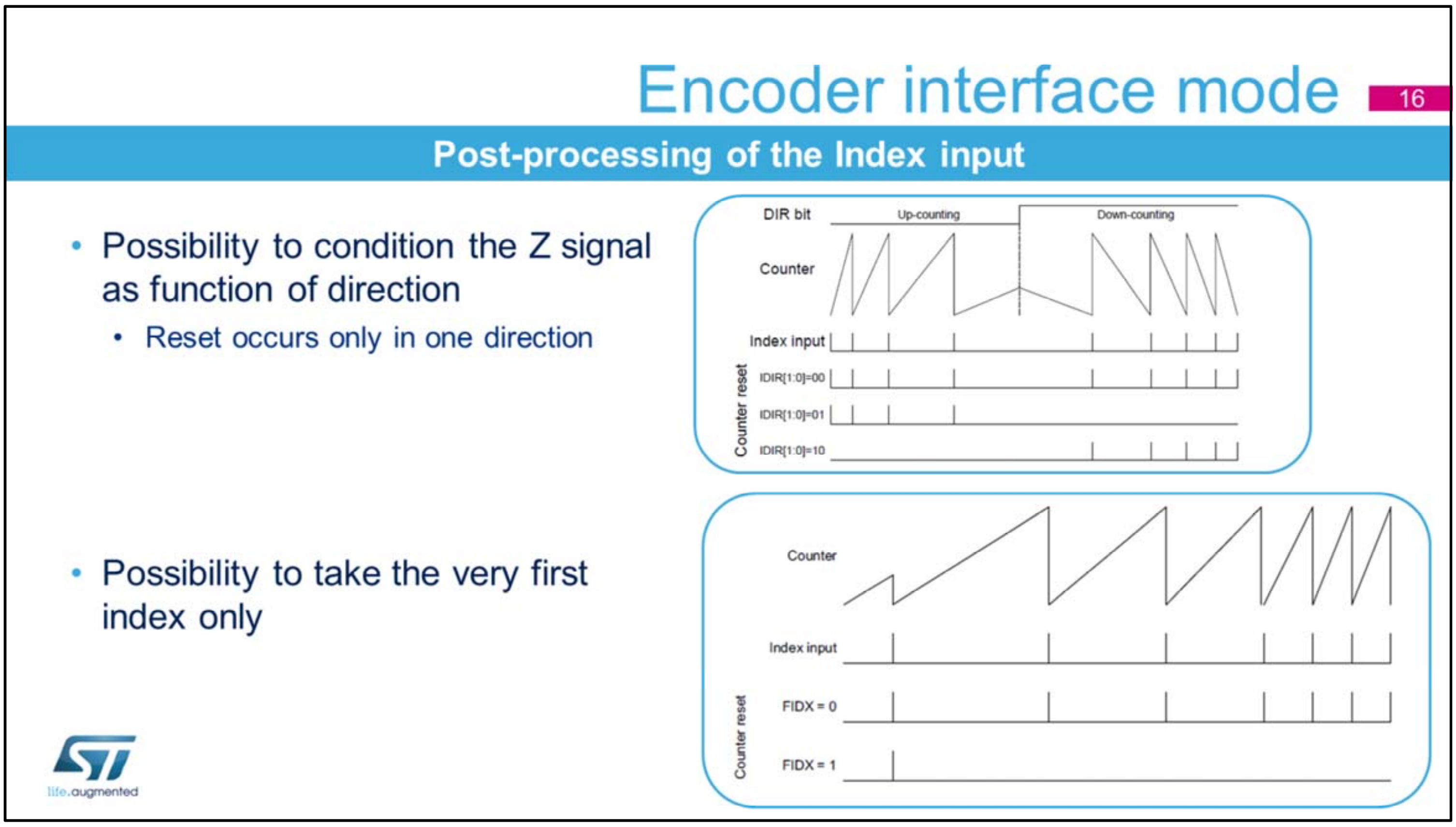
Index detection

- Two bits (IPOS[1:0]) are defining on which state the index is detected
- The Index detection is done differently depending on counting direction to ensure symmetrical operation during speed reversal:
  - The counter is reset during up-counting (DIR bit = 0)
  - The counter is set to TIMx_ARR when down counting



life.augmented

It is necessary to define on which encoder state, that is channel A and channel B state combination, the index must be synchronized, using the IPOS bitfield in the TIMx_ECR register.
The Index detection event will act differently depending on counting direction to ensure symmetrical operation during speed reversal:
• The counter is reset during up-counting (DIR bit = 0).
• The counter is set to TIMx_ARR when down counting.
This allows the index to be generated on the very same mechanical angular position whatever the counting direction.



Encoder interface mode

16

Post-processing of the Index input

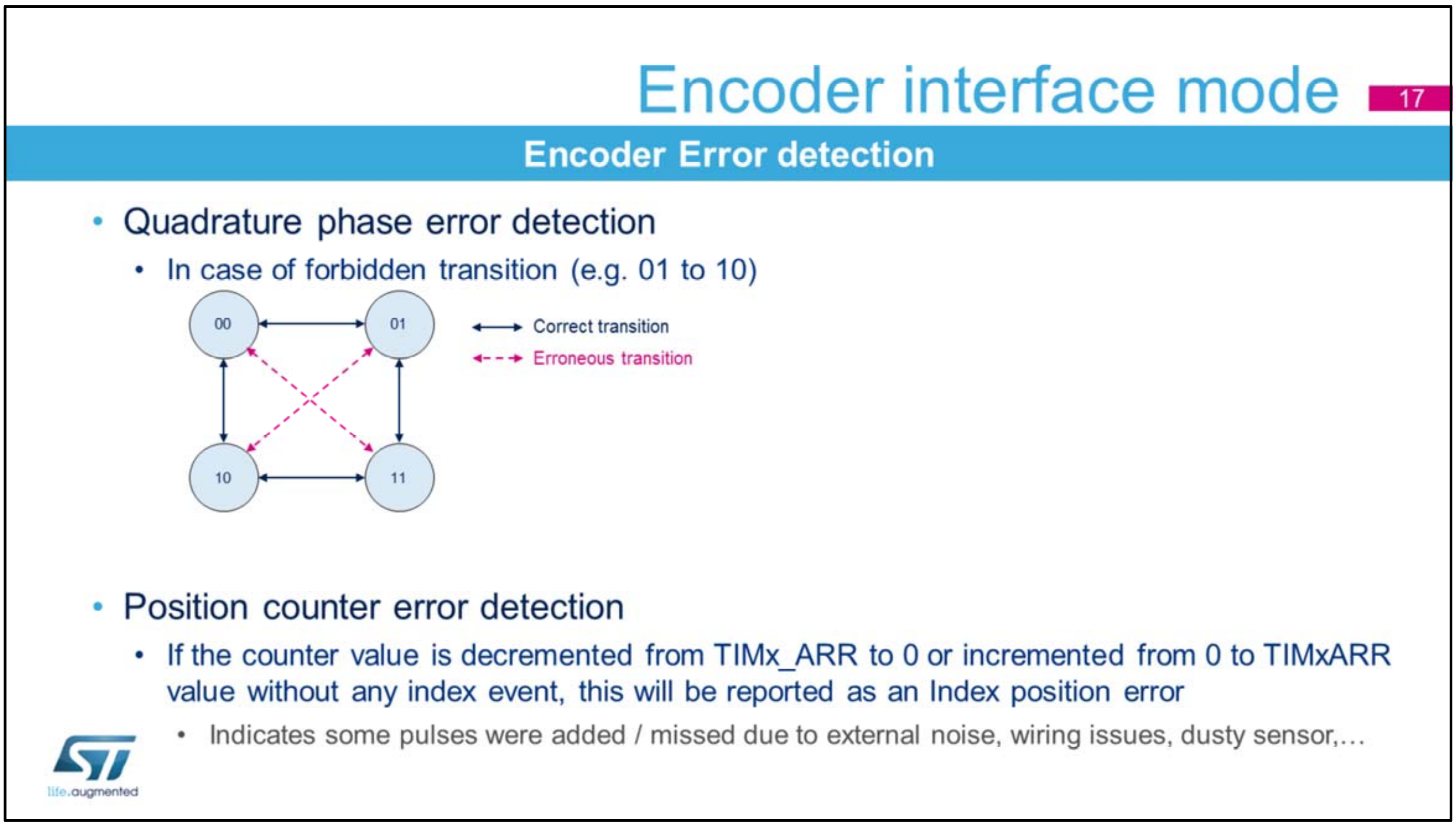
- Possibility to condition the Z signal as function of direction
  - Reset occurs only in one direction
- Possibility to take the very first index only

DIR bit
Up-counting
Down-counting
Counter
Index input
Counter reset
IDIR[1:0]=00
IDIR[1:0]=01
IDIR[1:0]=10
Counter
Index input
Counter reset
FIDX = 0
FIDX = 1
life.augmented

The timer unit supports post-processing of the Index input:

- When Directional index sensitivity is enabled, the index is active only in a selected counting direction.
- The upper timing diagram shows the relationship between index and counter reset events, depending on IDIR field value.
- For example when IDIR=01, index resets the counter when up-counting only.
- The second post-processing option allows the Index to be taken only once, as shown on the bottom timing diagram.

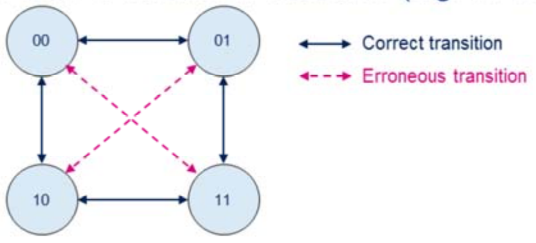
When FIDX=1, once the first index has arrived, any subsequent index will be ignored.



Encoder interface mode

Encoder Error detection

- Quadrature phase error detection
  - In case of forbidden transition (e.g. 01 to 10)

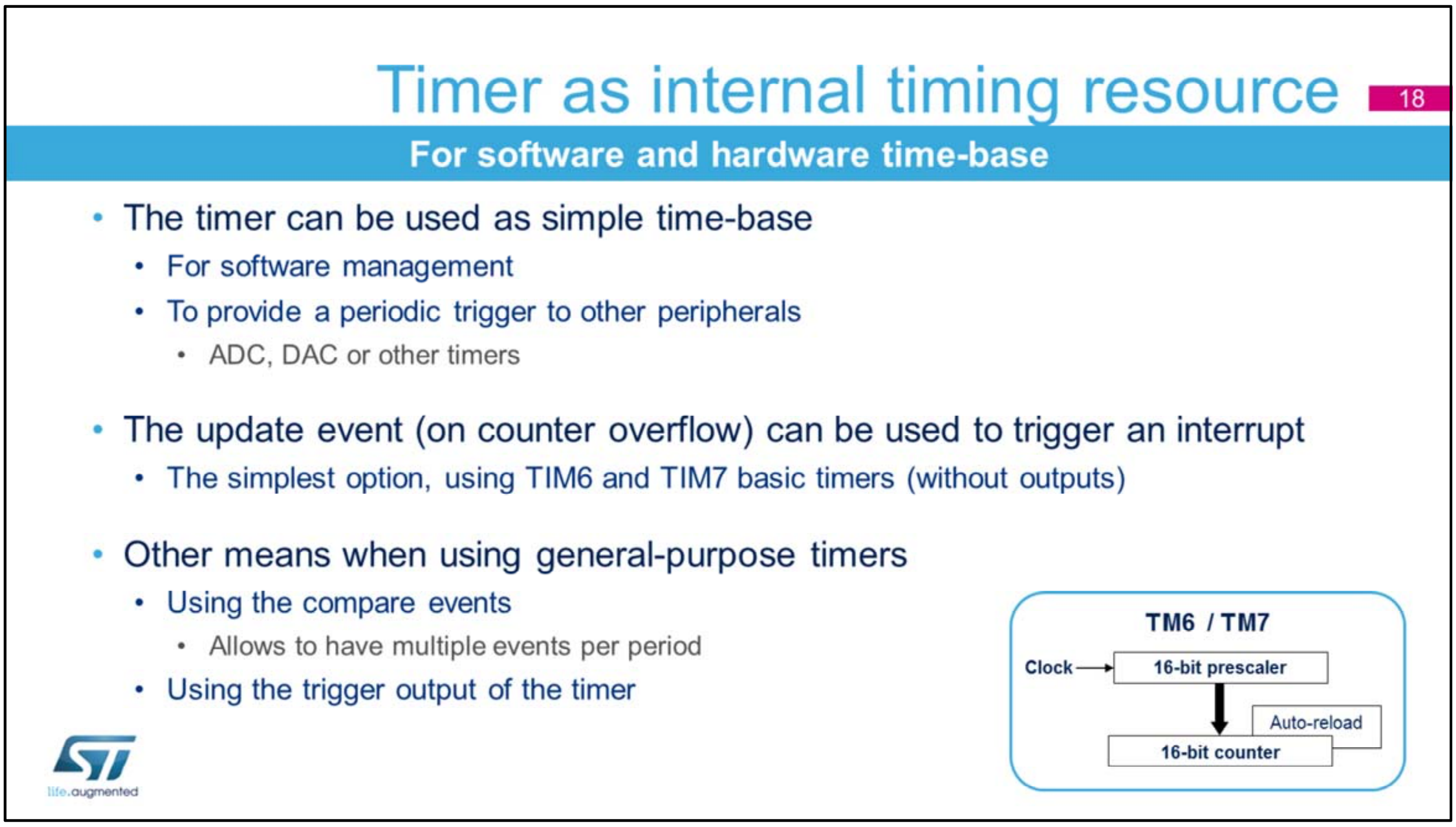


- Position counter error detection
  - If the counter value is decremented from TIMx_ARR to 0 or incremented from 0 to TIMxARR value without any index event, this will be reported as an Index position error
    - Indicates some pulses were added / missed due to external noise, wiring issues, dusty sensor,…

life.augmented

The timer unit detects two types of encoder errors.
For encoder configurations where 2 quadrature signals are available, it is possible to detect transition errors.
The reading on the 2 inputs corresponds to a 2-bit gray code which can be represented as a state diagram.
A single bit is expected to change at once.
A transition error interrupt can be generated when an erroneous transition occurs.
For encoder having an Index signal, it is possible to detect abnormal operation resulting in an excess of pulses per revolution.
An encoder with N pulses per revolution will provide 4xN counts per revolution.
The Index signal will reset the counter every 4xN clock periods.
If the counter value is incremented from 0 to auto-reload value or decremented from auto-reload value to 0

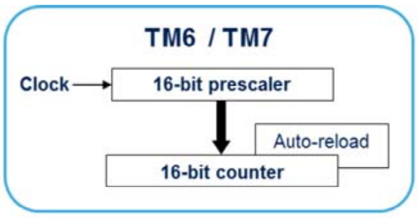
without any index event, this will be reported as an Index position error.
An interrupt can be generated when this error occurs.



Timer as internal timing resource

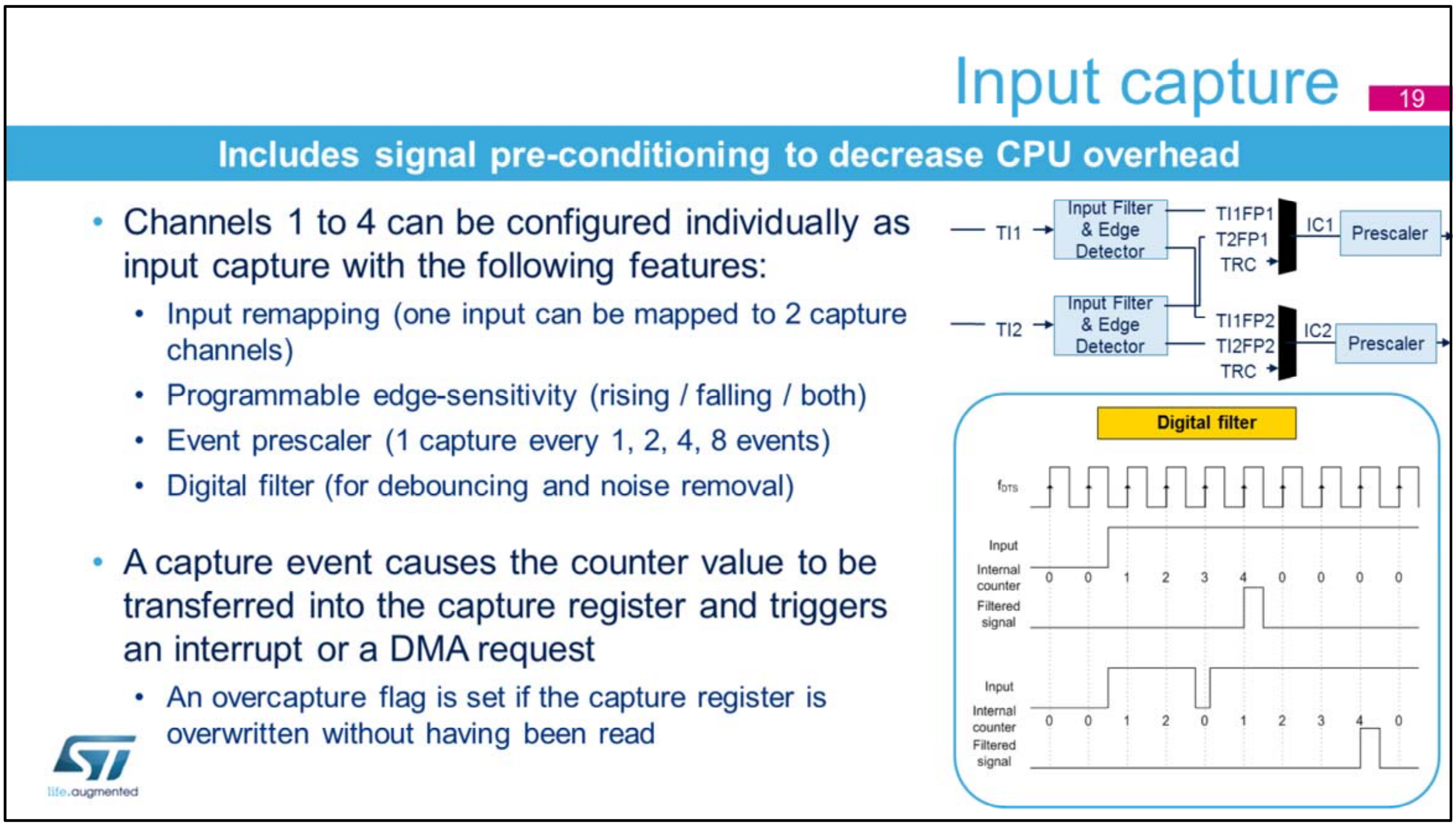
For software and hardware time-base

- 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



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 through the interconnect matrix.
An 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.

Multiple timing events can be generated with a single timer using multiple compare channels.

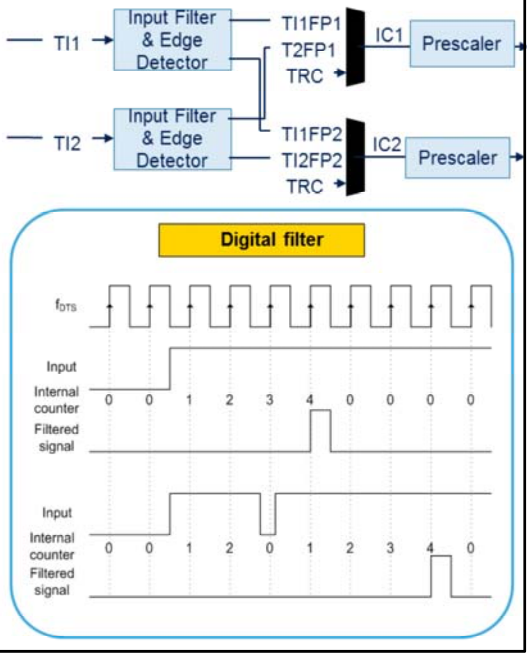


Input capture

19

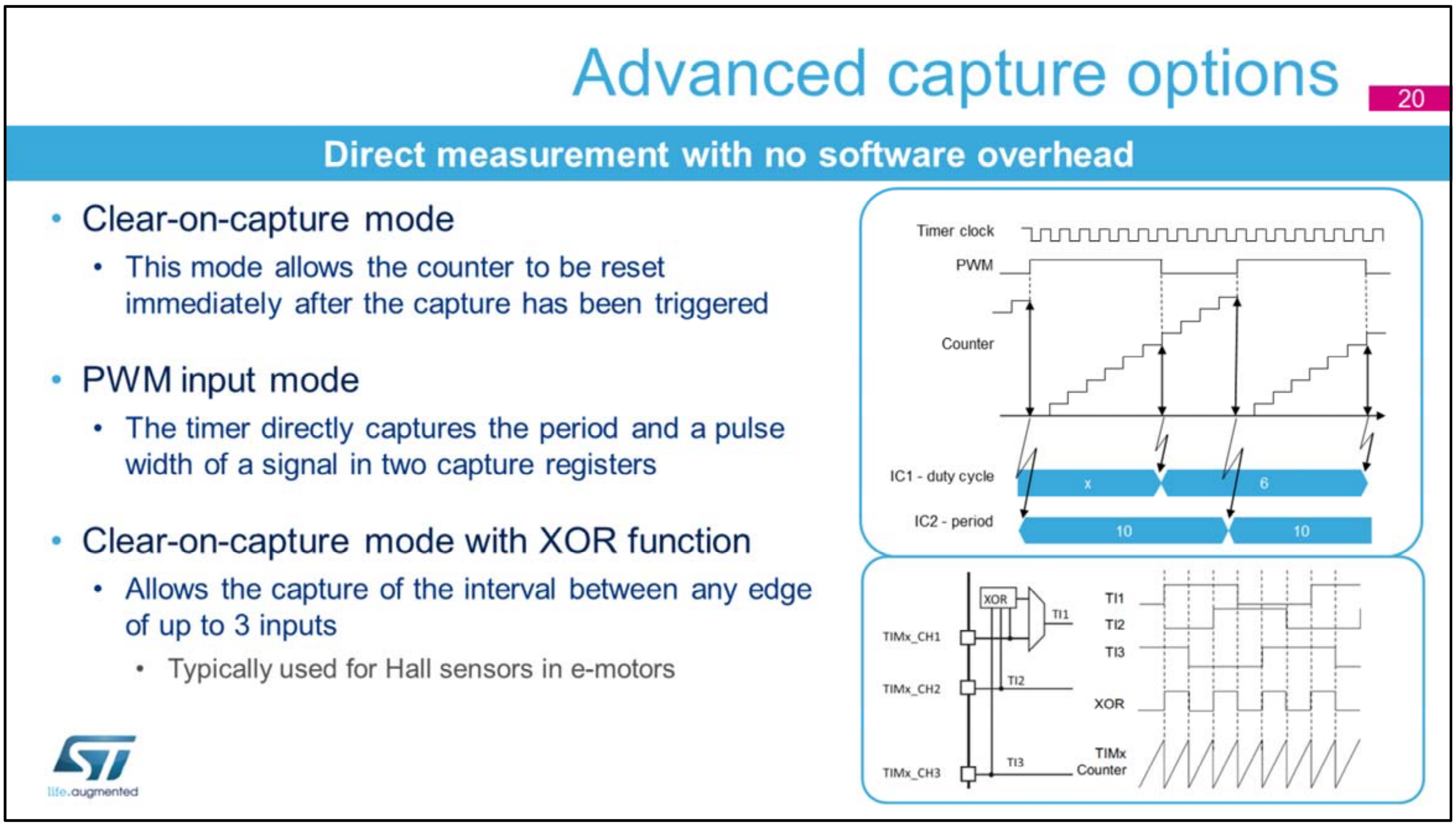
Includes signal pre-conditioning to decrease CPU overhead

- Channels 1 to 4 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



This slide describes the input capture features.
Channels 1 to 4 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), see the figure on the top right.
TRC input can be chosen as the capture trigger, it is connected to one of the ITR inputs (timer-to-timer interconnect).
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 over-capture flag is set for the software to manage this condition if needed.

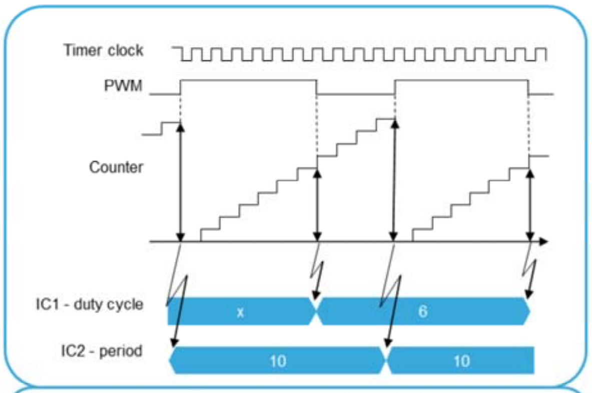


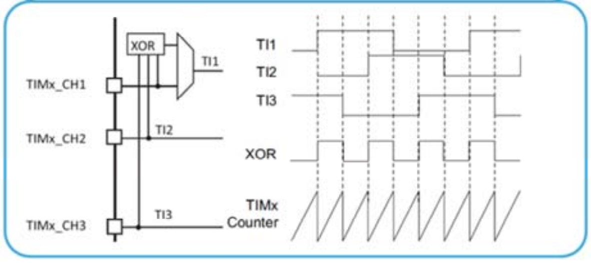
Advanced capture options

20

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

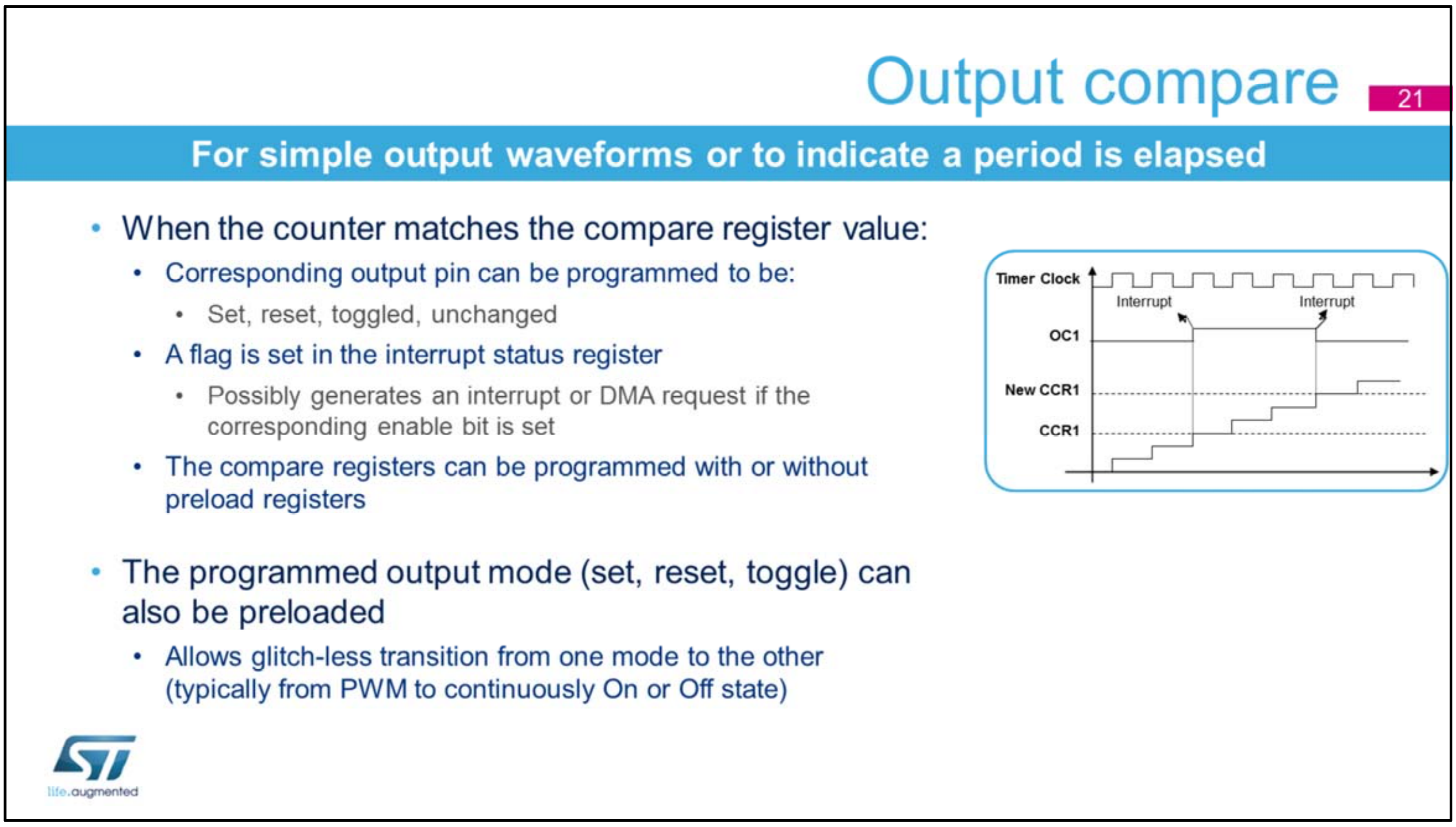




life.augmented

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.

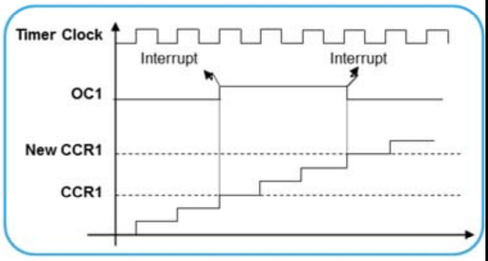
Lastly, the timer includes an XOR function to combine the three input channels with XOR logic.
This is typically used to handle the three 120° phase-shifted signals coming from the Hall sensors in electrical motors.
This allows you to have a clear on capture happening on each and every edge of the three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



Output compare

For simple output waveforms or to indicate a period is elapsed

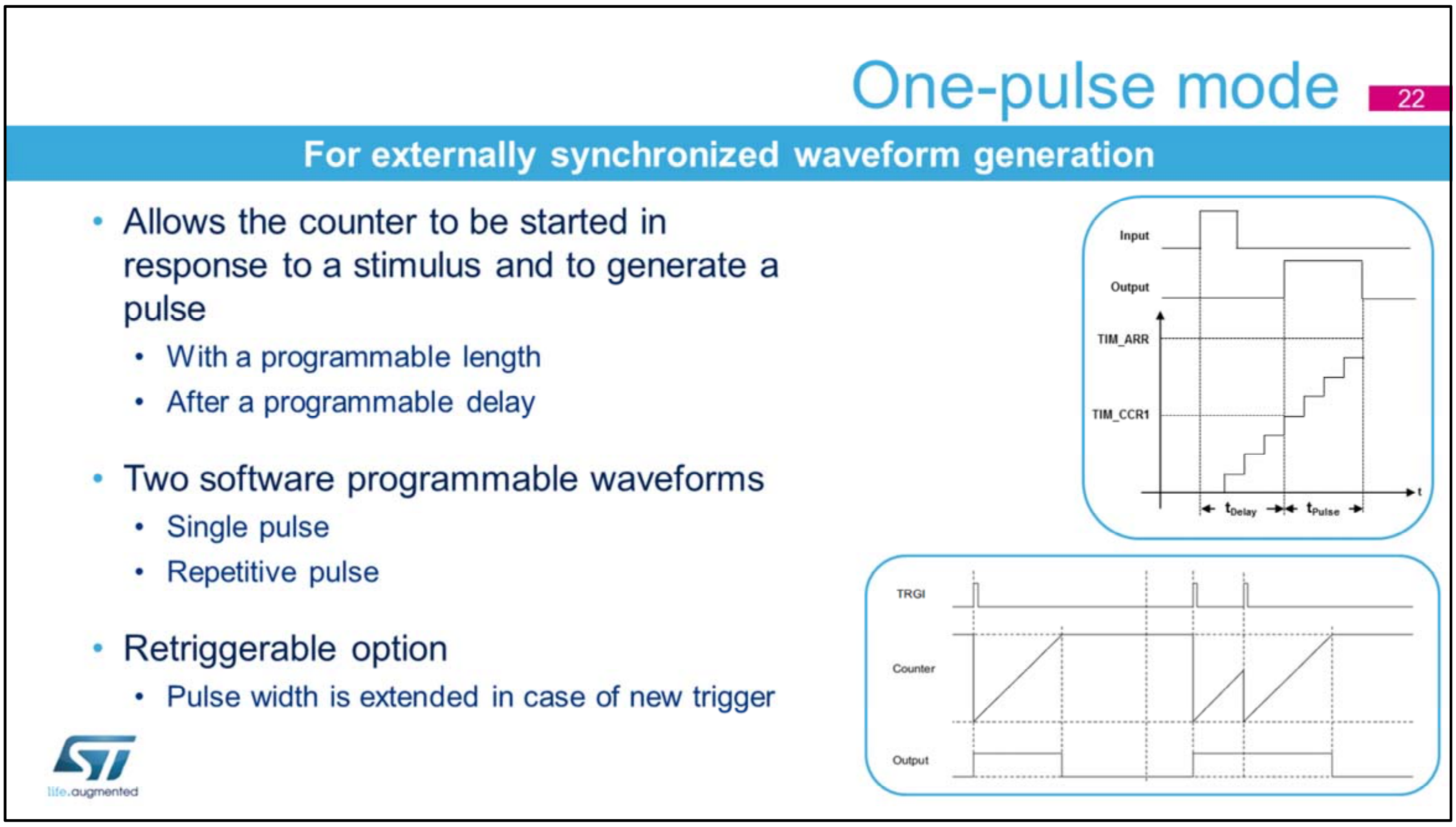
- When the counter matches the compare register value:
  - Corresponding output pin can be programmed to be:
    - Set, reset, toggled, unchanged
  - A flag is set in the interrupt status register
    - Possibly generates an interrupt or DMA request if the corresponding enable bit is set
  - The compare registers can be programmed with or without preload registers
- The programmed output mode (set, reset, toggle) 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

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.
In the timing diagram, preload is disabled.
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.

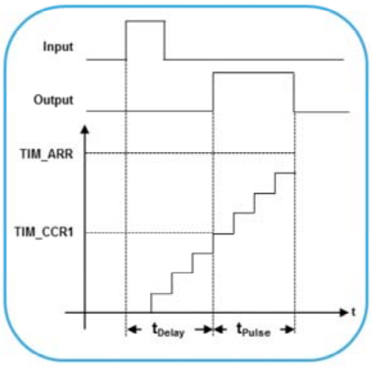


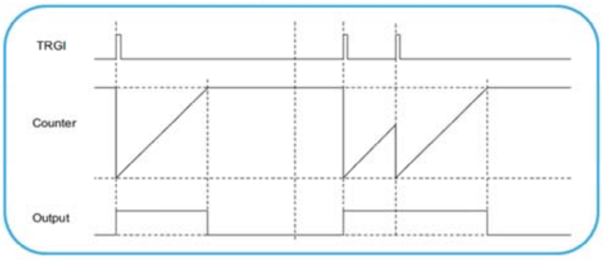
One-pulse mode

22

For externally synchronized waveform generation

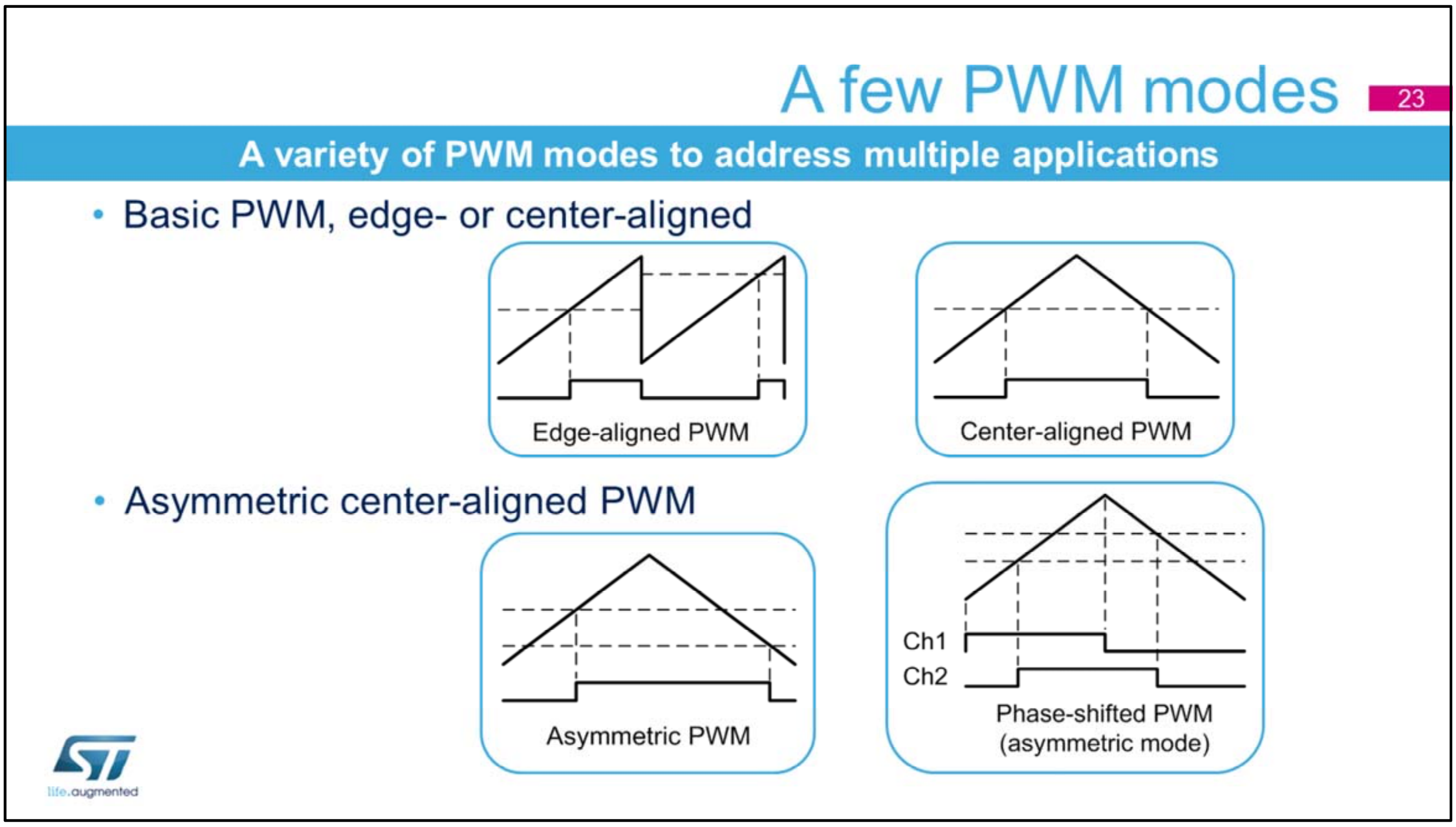
- Allows the counter to be started in response to a stimulus and to generate a pulse
  - With a programmable length
  - After a programmable delay
- Two software programmable waveforms
  - Single pulse
  - Repetitive pulse
- Retriggerable option
  - Pulse width is extended in case of new trigger





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. See the upper timing diagram.
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 re-triggerable option. The timing diagram at the bottom of this slide describes this option.
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

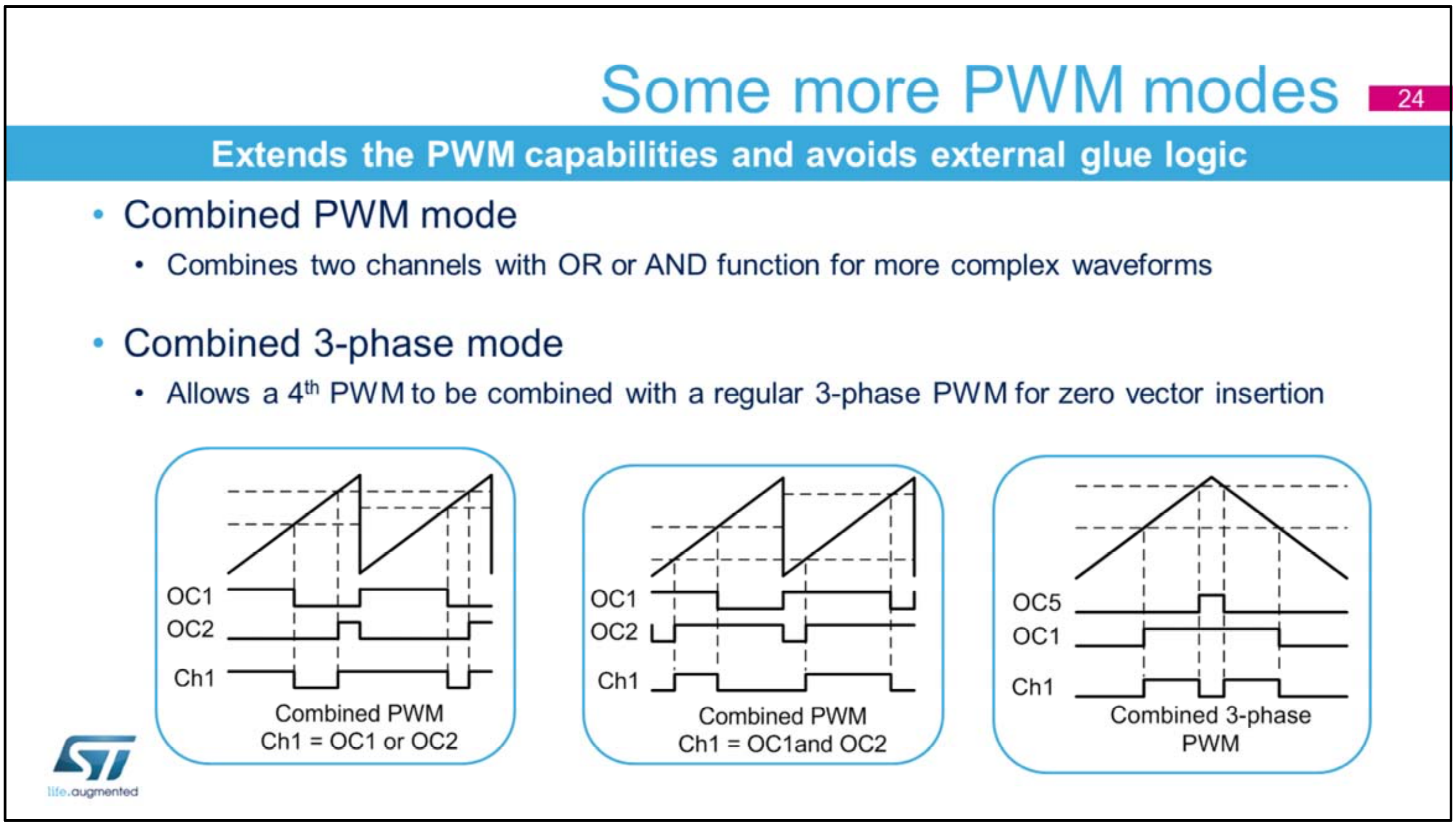
23

A variety of PWM modes to address multiple applications

- Basic PWM, edge- or center-aligned
- Asymmetric center-aligned PWM

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 6 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 the bottom right timing diagram, 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

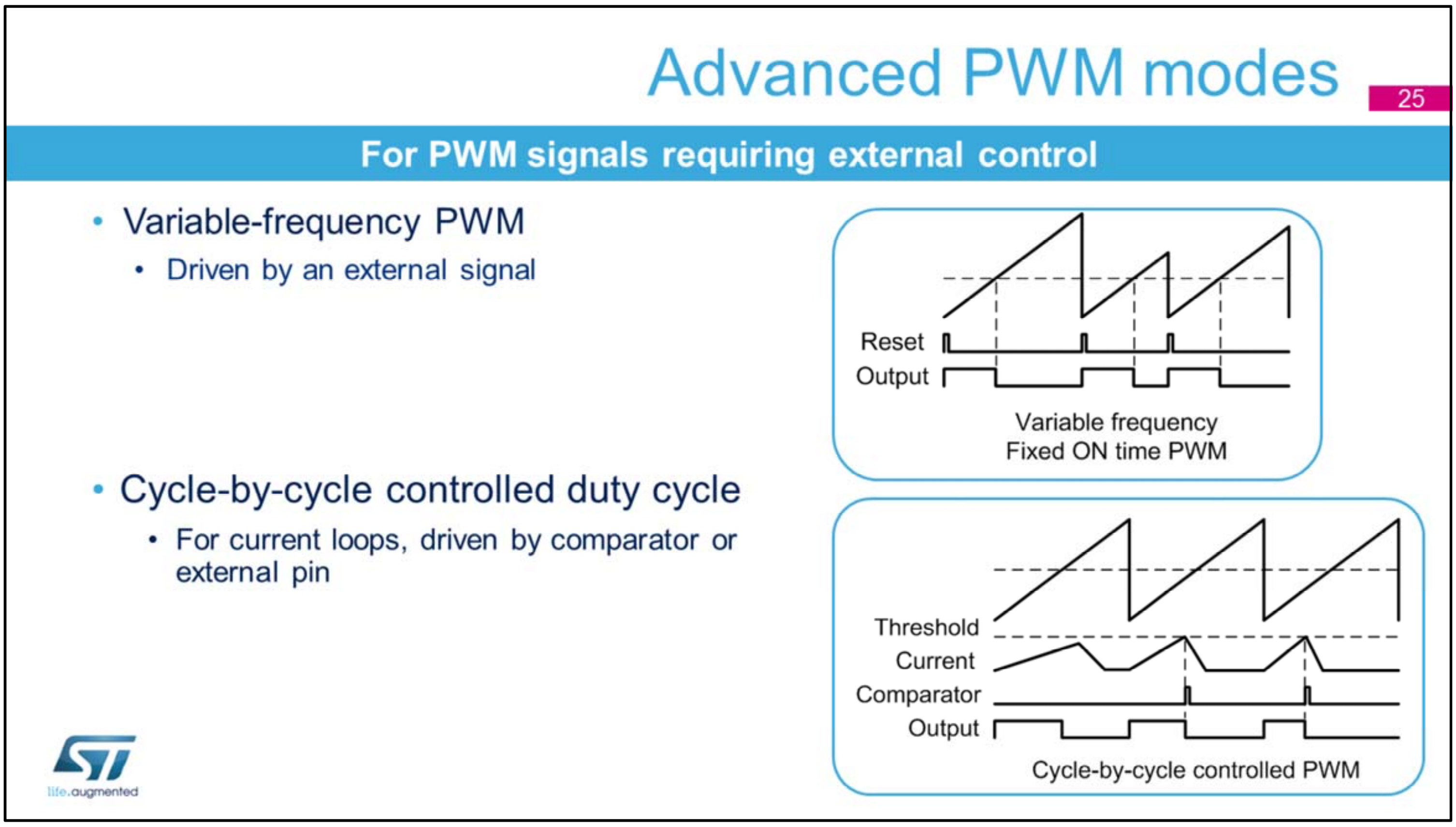
24

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

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 to have two periodic pulses generated with any pulse width and any phase relationship value.
Combined 3-phase PWM mode allows one to three center-aligned PWM signals to be generated with a single programmable signal ANDed in the middle of the pulses.
This 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.

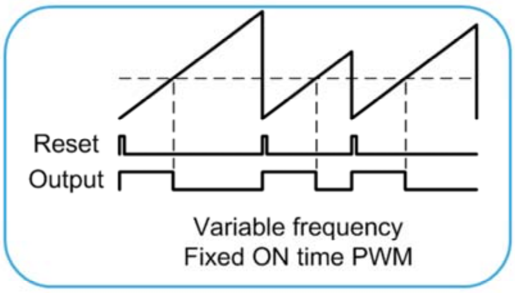


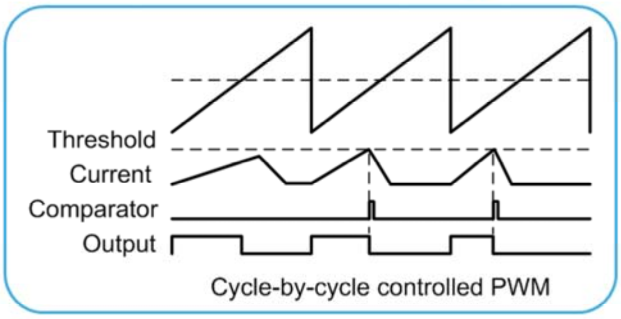
Advanced PWM modes

25

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

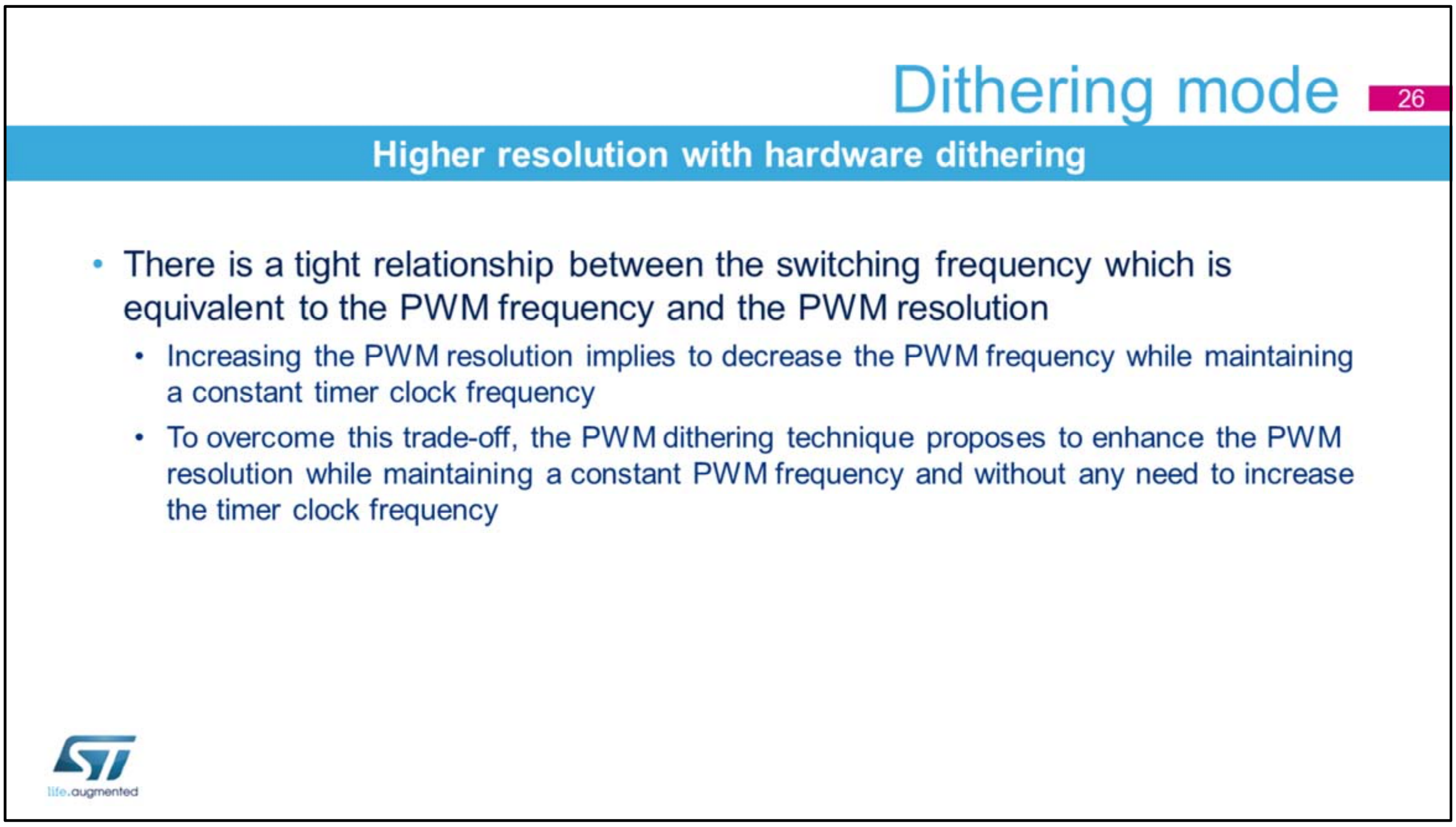




life.augmented

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.



Dithering mode

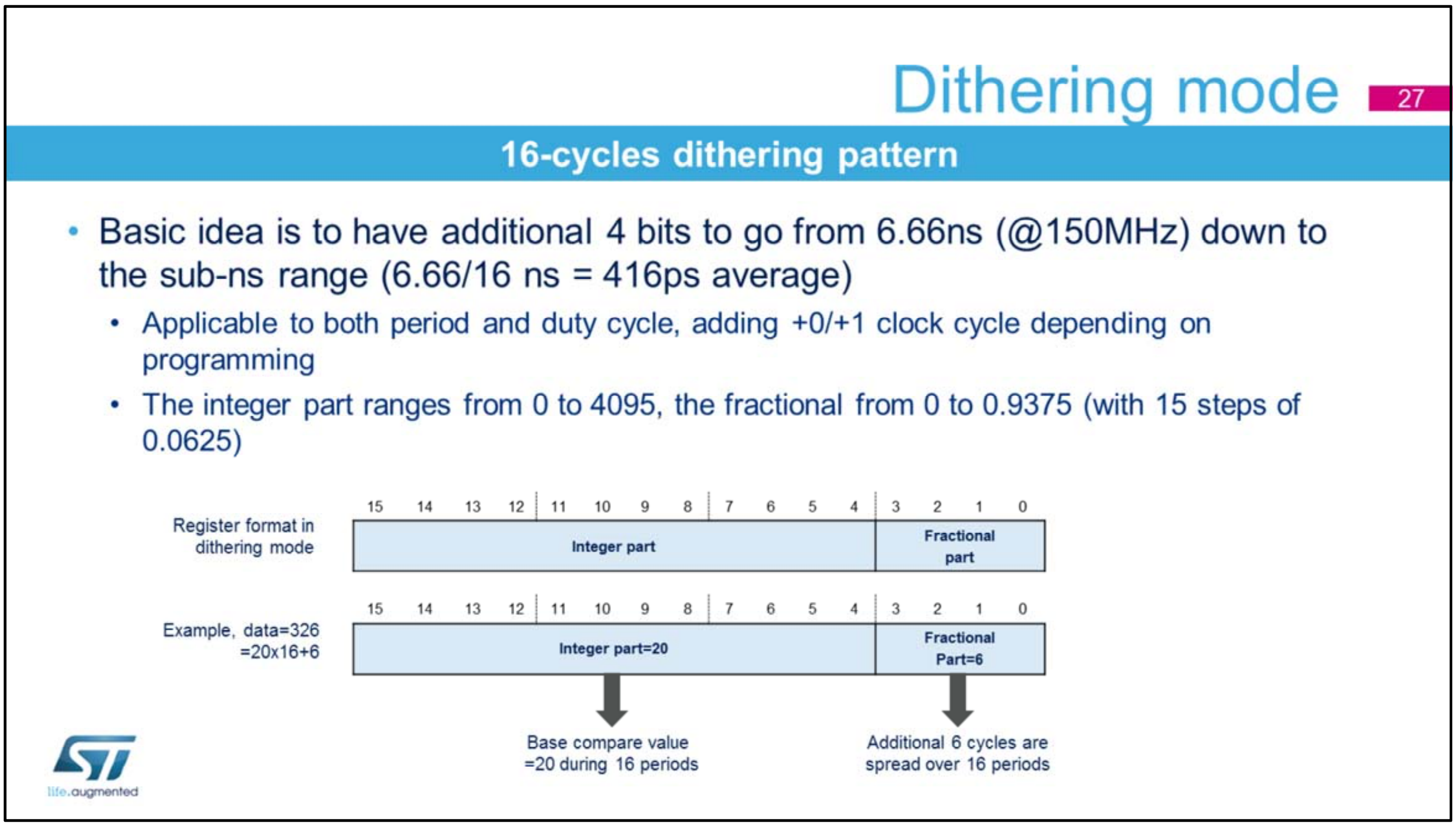
26

Higher resolution with hardware dithering

- There is a tight relationship between the switching frequency which is equivalent to the PWM frequency and the PWM resolution
  - Increasing the PWM resolution implies to decrease the PWM frequency while maintaining a constant timer clock frequency
  - To overcome this trade-off, the PWM dithering technique proposes to enhance the PWM resolution while maintaining a constant PWM frequency and without any need to increase the timer clock frequency

life.augmented

For a timer with a 16-bit resolution, increasing the resolution implies to decrease the frequency.
For a given PWM clock frequency,
-If 14 bits of the timer are used, the PWM period is 2^14 clock cycles and the resolution is the PWM period divided by 2^14.
-If 13 bits of the timer are used, the PWM period is 2^13 clock cycles and the resolution is the PWM period divided by 2^13.
The PWM mode effective resolution can be increased by enabling the dithering mode.



Dithering mode

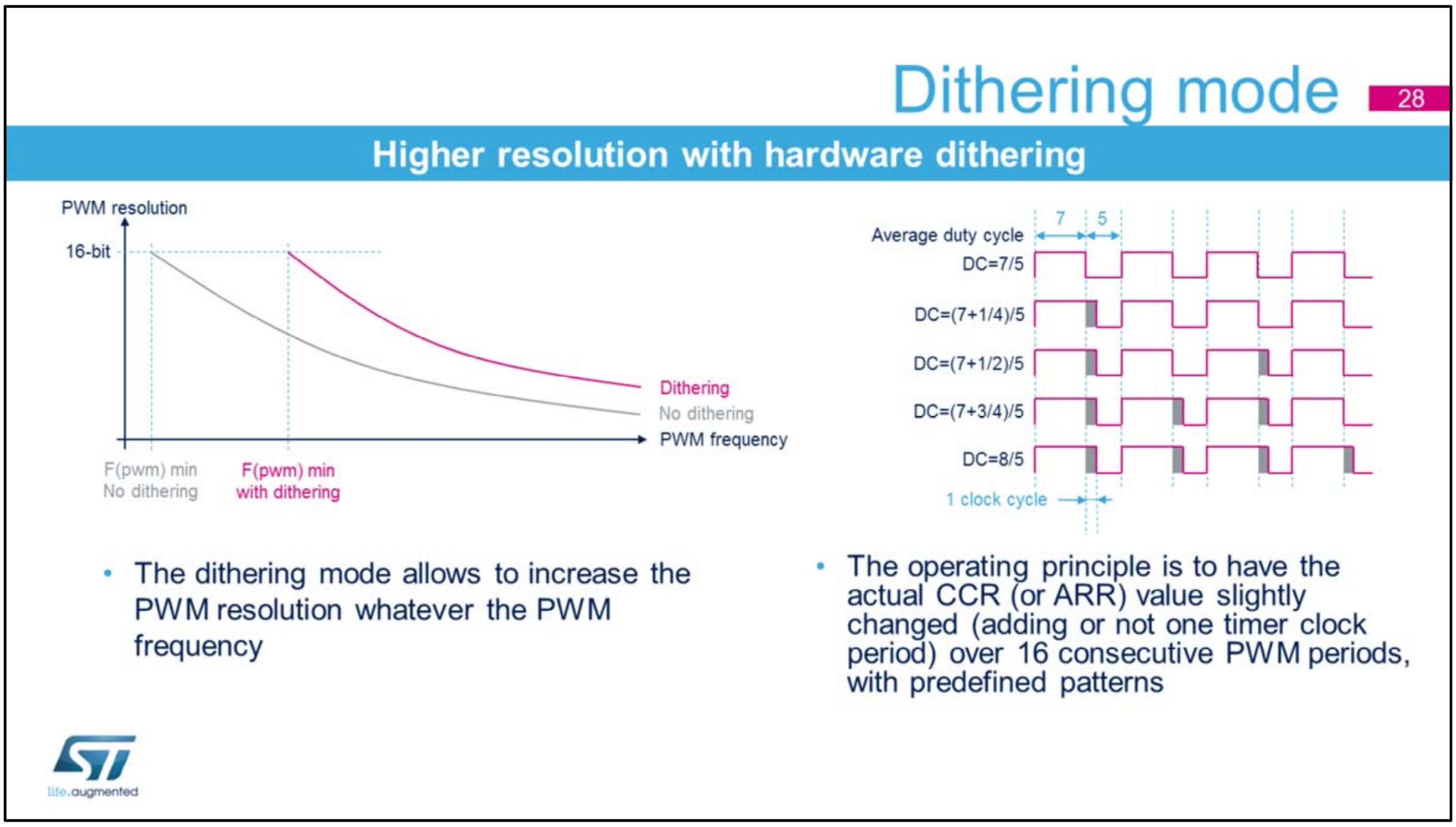
16-cycles dithering pattern

- Basic idea is to have additional 4 bits to go from 6.66ns (@150MHz) down to the sub-ns range (6.66/16 ns = 416ps average)
  - Applicable to both period and duty cycle, adding +0/+1 clock cycle depending on programming
  - The integer part ranges from 0 to 4095, the fractional from 0 to 0.9375 (with 15 steps of 0.0625)

Register format in dithering mode
15 14 13 12 11 10 9 8 7 6 5 4 3 2 1 0
Integer part
Fractional part
Example, data=326 =20x16+6
Integer part=20
Fractional Part=6
Base compare value =20 during 16 periods
Additional 6 cycles are spread over 16 periods
life.augmented

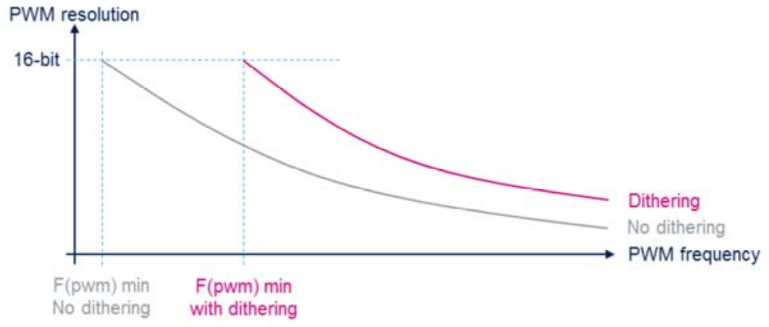
When the dithering mode is enabled, the register coding is changed:

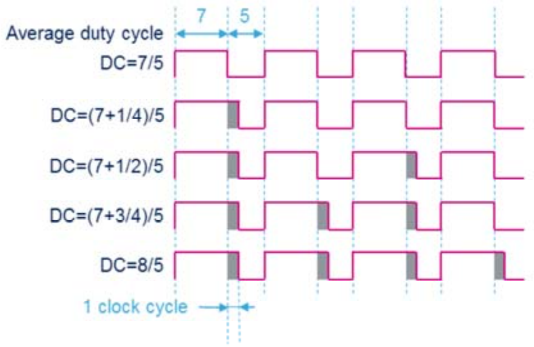
- The 4 least significant bits are coding for the enhanced resolution part (fractional part)
- The most significant bits are left-shifted to the bits 19:4 and are coding for the base value.



Dithering mode

Higher resolution with hardware dithering

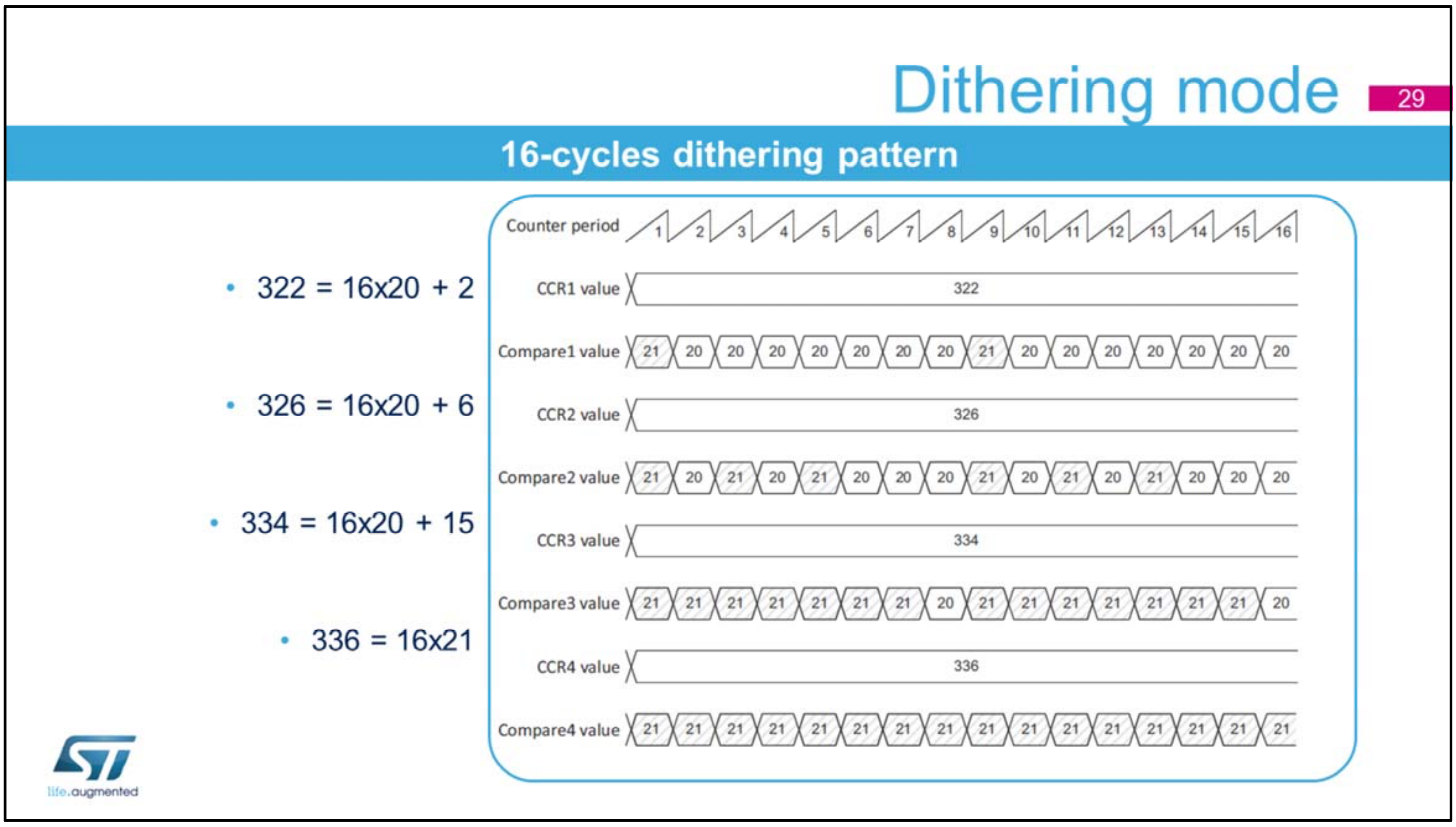




- The dithering mode allows to increase the PWM resolution whatever the PWM frequency
- The operating principle is to have the actual CCR (or ARR) value slightly changed (adding or not one timer clock period) over 16 consecutive PWM periods, with predefined patterns

life.augmented

The curves on the left indicate the PWM resolution according to the PWM frequency.
For a given resolution, the frequency when dithering is enabled is larger than without dithering.
The timing diagram on the right explains the dithering principle: adding or not a timer clock period over 16 consecutive PWM periods, according to the selected duty cycle.
This allows a 16-fold resolution increase, considering the average duty cycle or PWM period

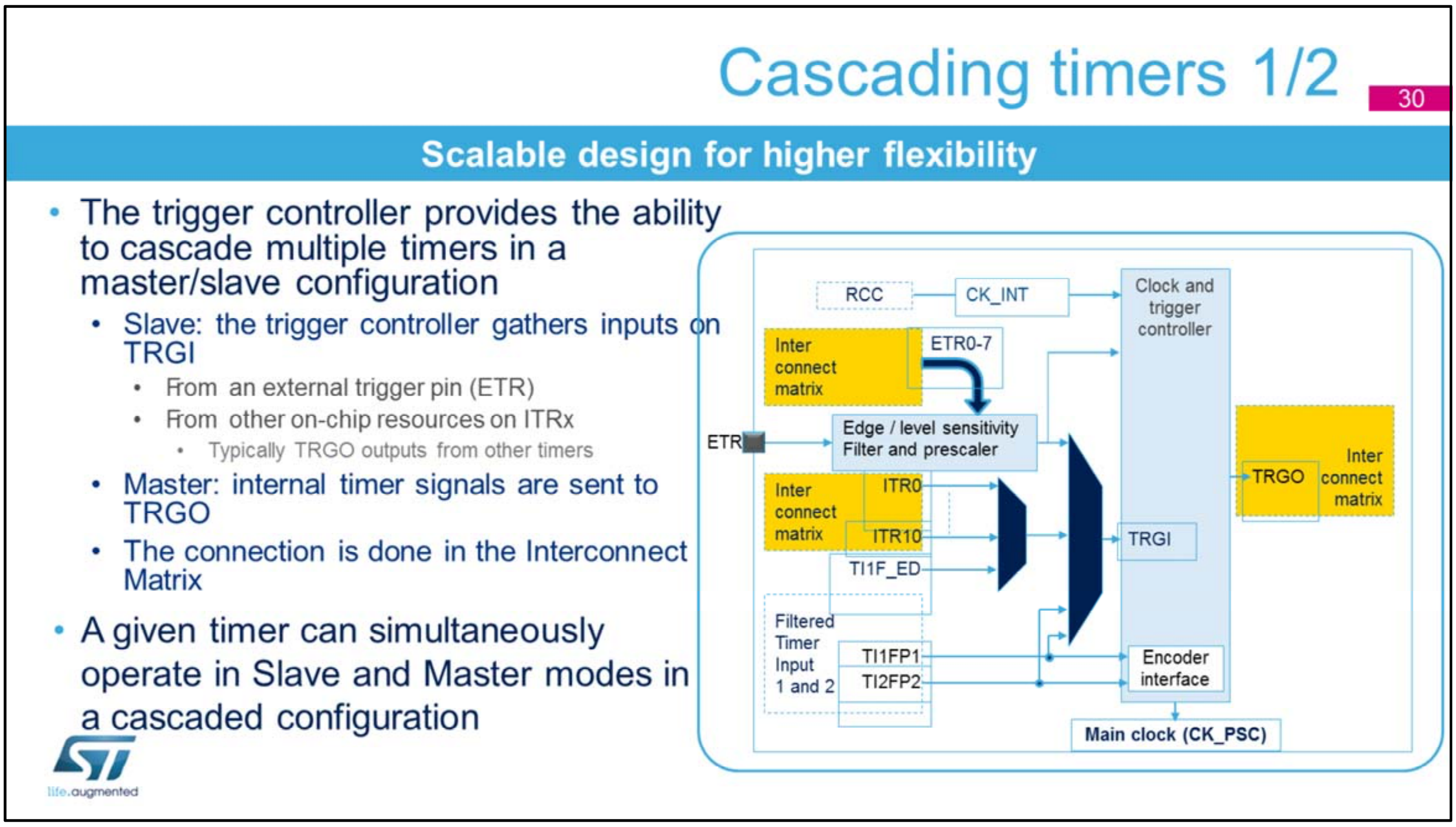


Dithering mode

16-cycles dithering pattern

- 322 = 16x20 + 2
- 326 = 16x20 + 6
- 334 = 16x20 + 15
- 336 = 16x21

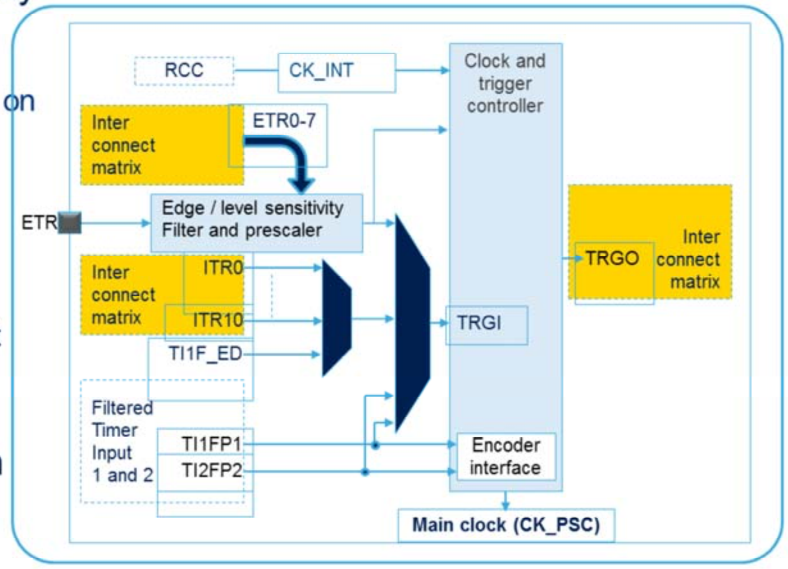
The auto-reload and compare values increments are spread following specific patterns.
In this timing diagram, it is assumed that the counter period is fixed, only the duty cycle will vary.
The pattern will depend on the reminder of the division between the compare value and 16.
On the first timing diagram, the compare value programmed in CCR1 is equal to 322.
322 = 20*16+2.
So the reminder of the division between 322 and 26 is 2.
It means that during two periods over 16 periods, the least significant bit of the compare value is incremented.
The dithering sequence is done to have increments distributed as evenly as possible and minimize the overall ripple.



Cascading timers 1/2

Scalable design for higher flexibility

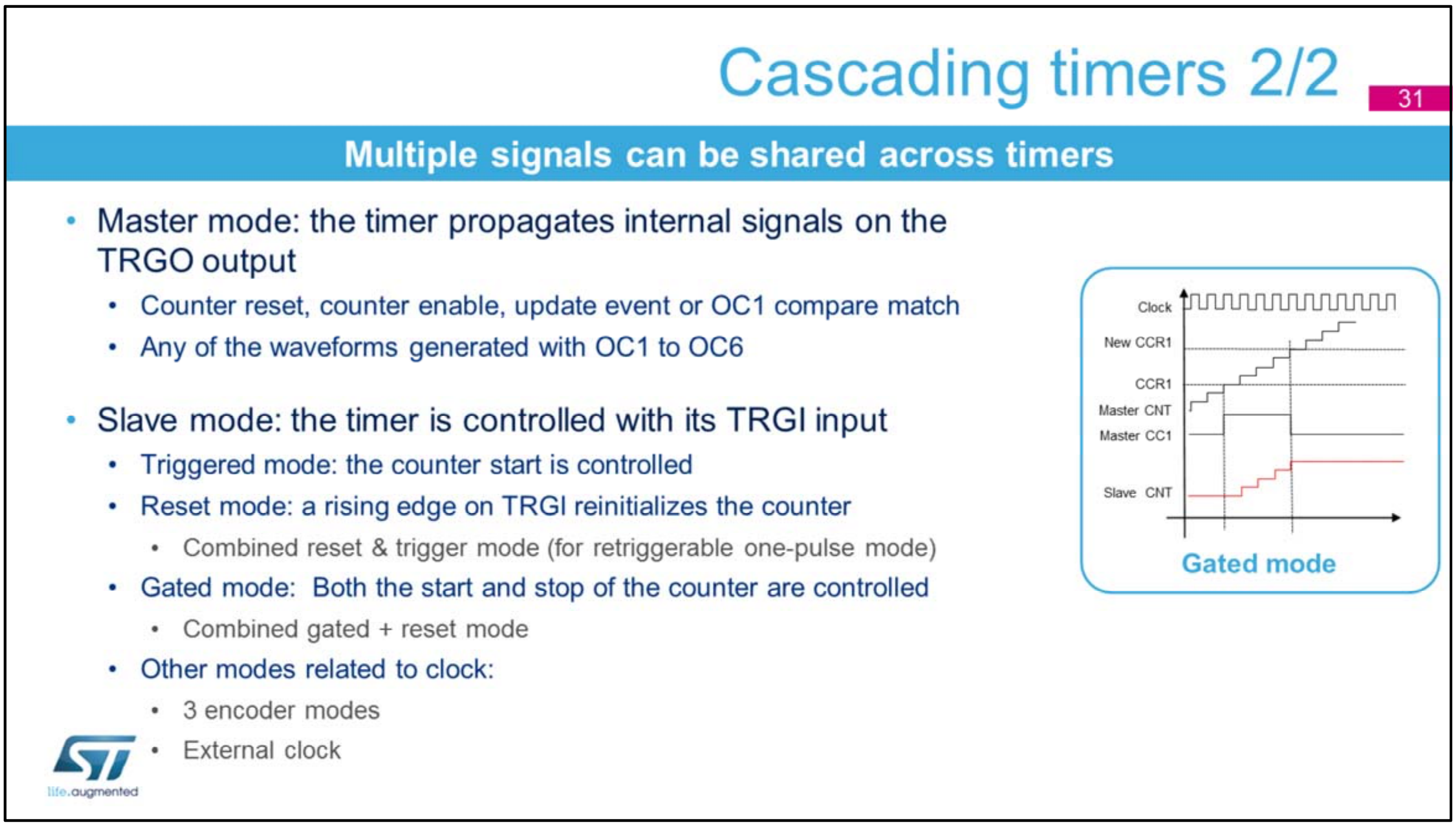
- The trigger controller provides the ability to cascade multiple timers in a master/slave configuration
  - Slave: the trigger controller gathers inputs on TRGI
    - From an external trigger pin (ETR)
    - From other on-chip resources on ITRx
      - Typically TRGO outputs from other timers
  - Master: internal timer signals are sent to TRGO
  - The connection is done in the Interconnect Matrix
- A given timer can simultaneously operate in Slave and Master modes in a cascaded configuration



life.augmented

This slide presents the timer's synchronization features.
The trigger controller can 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 11 internal trigger inputs (ITR0 to ITR10), 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.
Master synchronization and slave synchronization are independently instantiated in the timer units.
See the table at the end of this presentation, summarizing the capabilities of all STM32G4 timers.

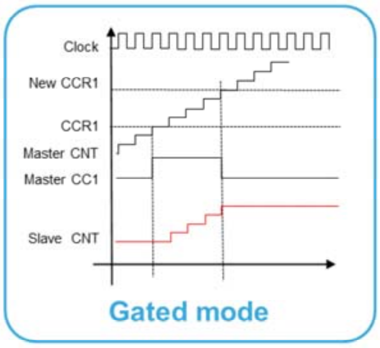


Cascading timers 2/2

31

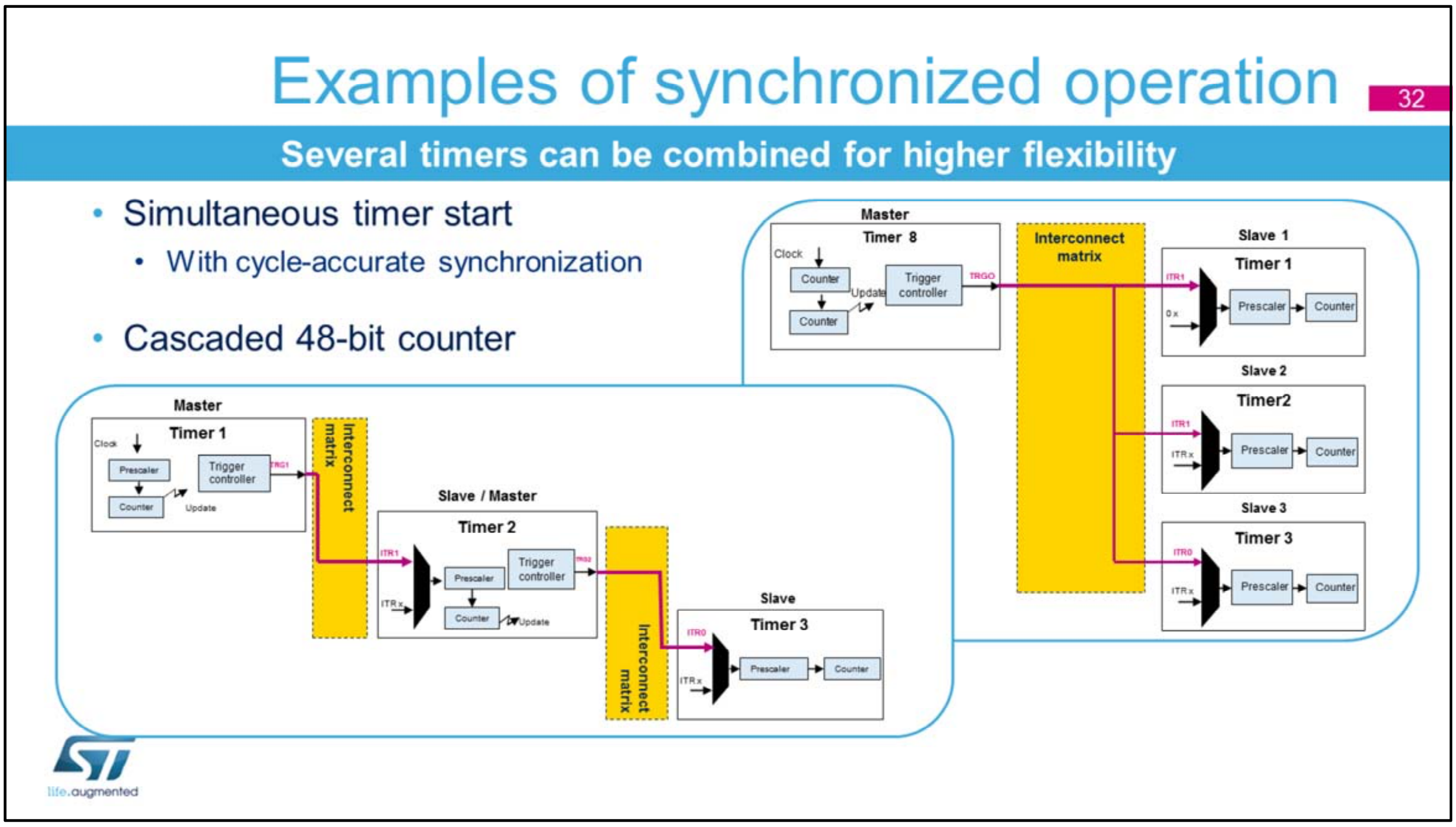
Multiple signals can be shared across timers

- 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 OC6
- 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
    - Combined gated + reset mode
  - Other modes related to clock:
    - 3 encoder modes
    - External clock



This slide lists the various operating modes and the signals exchanged between timers.
In Master mode, 16 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 6 waveforms generated, including PWM signals, to the other timer modules (Compare on channel 1 to 6 and compare pulses on all channels).
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 timing diagram, the counter is active only while the level on the input signal is high.
A combined mode including reset and gated can be used to detect out-of-range PWM signal (duty cycle exceeding a maximum expected value).
The external clock mode 2 can be used in addition to another slave mode (except external clock mode 1 and encoder mode).
In this case, the ETR signal is used as external clock input, and another input can be selected as trigger input.
Lastly, the slave mode selection includes clock-related modes, such as quadrature encoder decoding or external clocking modes mentioned earlier in this presentation.

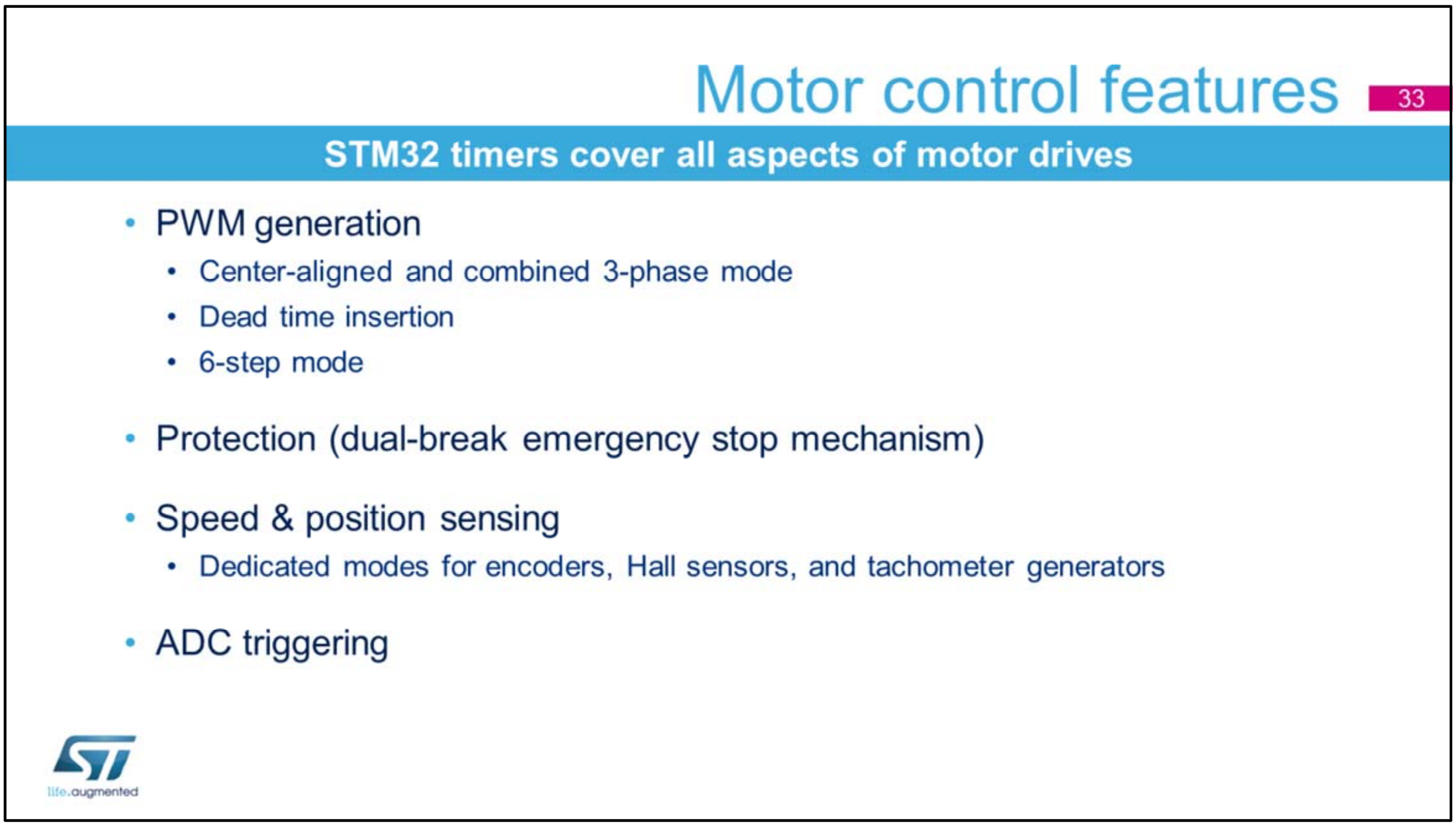


Examples of synchronized operation

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 1, 2, 3 and 8 as shown, it is possible to have up to 16 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 1's counter holds the least significant 16-bits, Timer 2's counter holds the medium bits (bits 16 to 31) and Timer 3's counter holds the upper bits from bit 32 to bit 47.



Motor control features

STM32 timers cover all aspects of motor drives

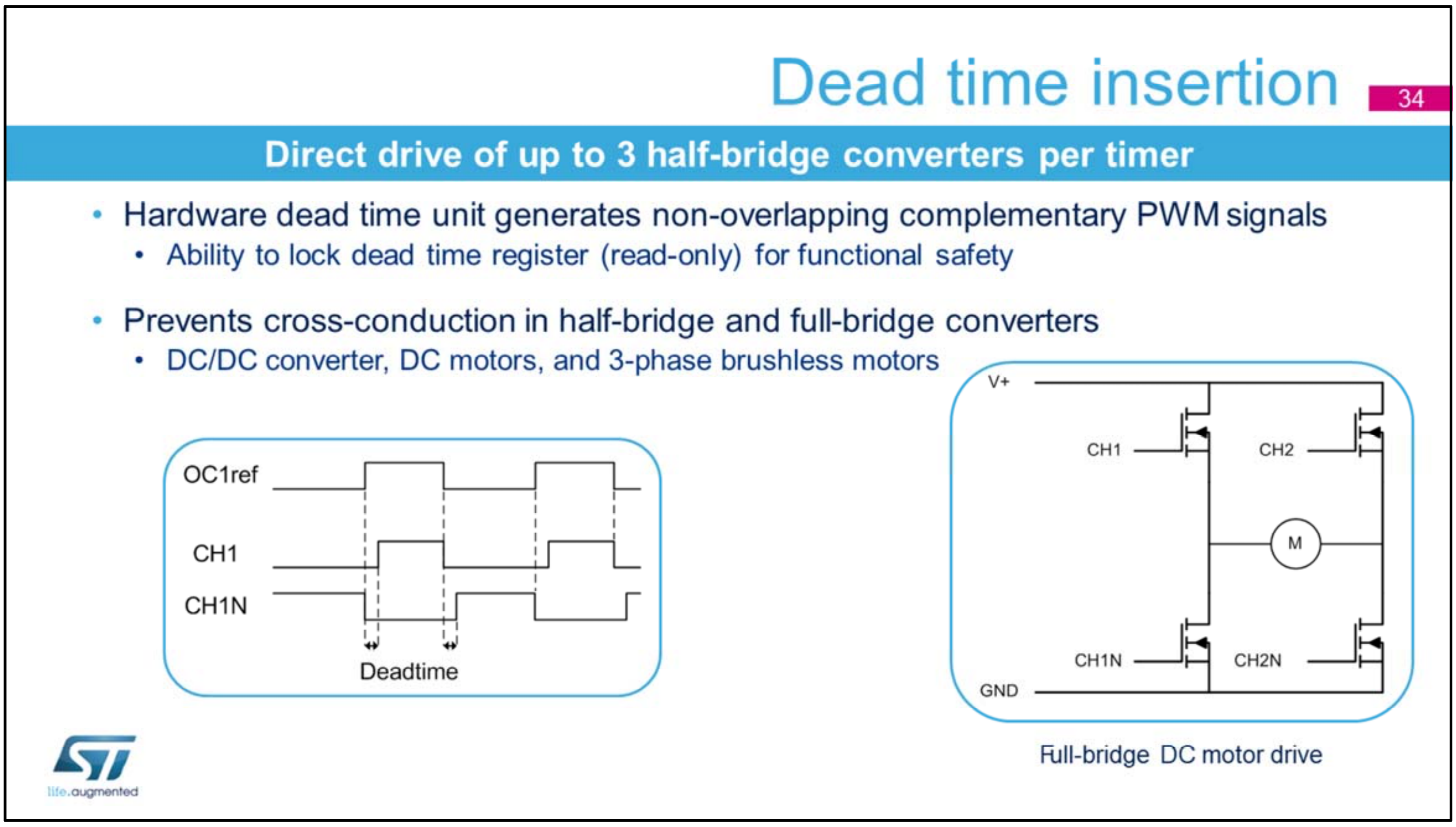
- PWM generation
  - Center-aligned and combined 3-phase mode
  - Dead time insertion
  - 6-step mode
- Protection (dual-break emergency stop mechanism)
- Speed & position sensing
  - Dedicated modes for encoders, Hall sensors, and tachometer generators
- ADC triggering

life.augmented

This slide summarizes the timer’s four 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.

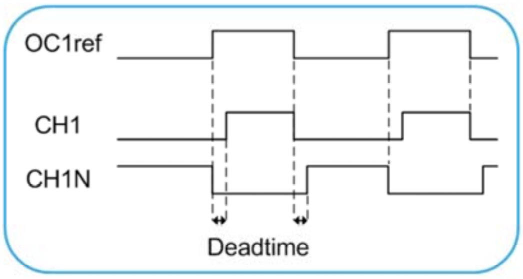
- Lastly, the timer includes synchronized ADC triggering options, necessary to properly manage voltage and current sensing and avoid any acquisition issue due to switching noise in power stages.

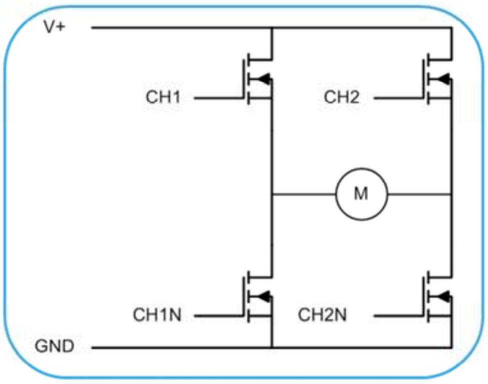


Dead time insertion

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

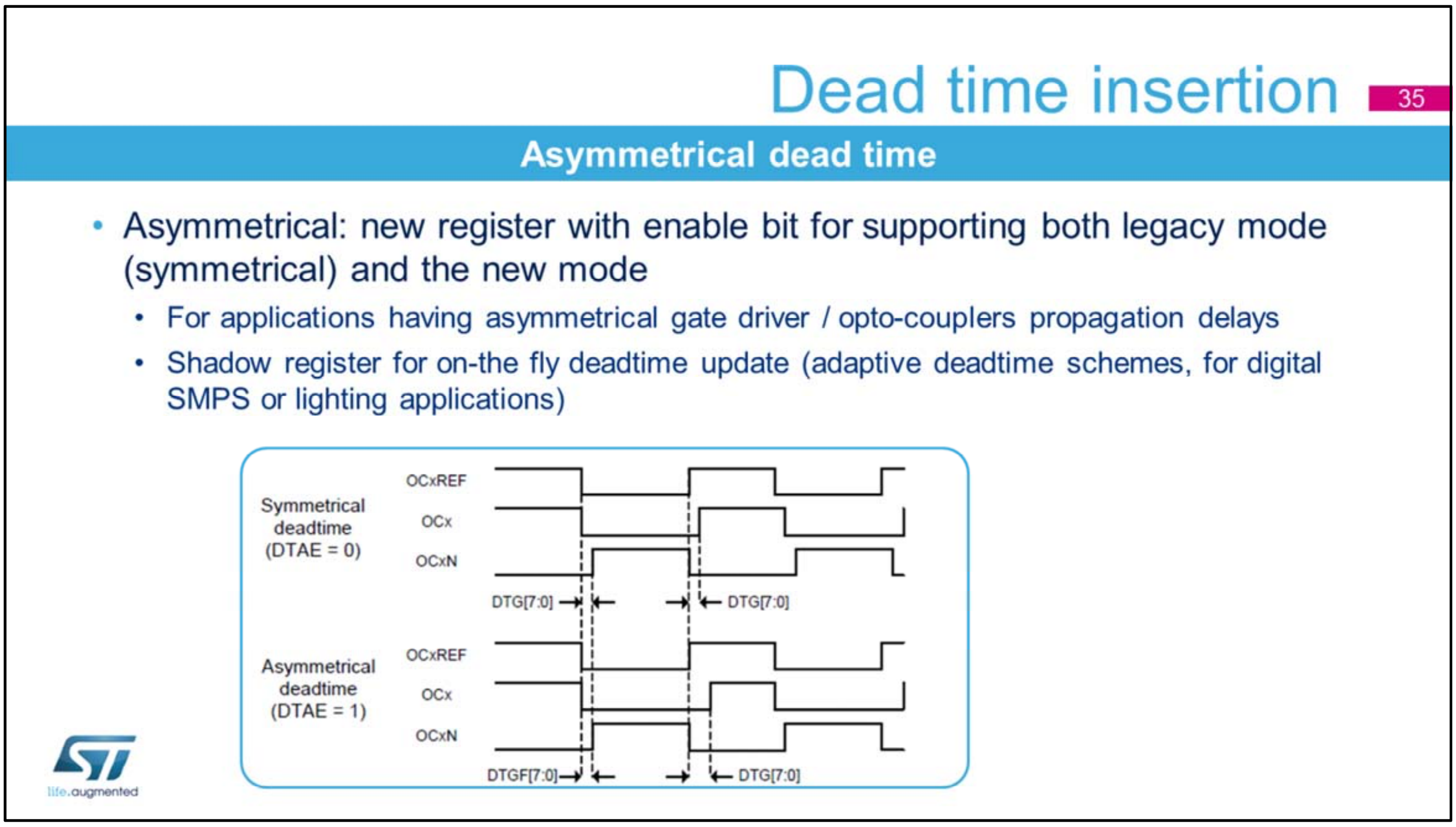




Full-bridge DC motor drive

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 four dead time generators for OC1, OC2, OC3 and OC4 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.

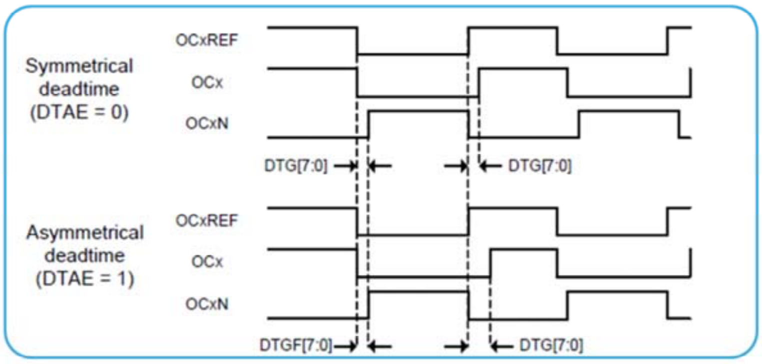


Dead time insertion

35

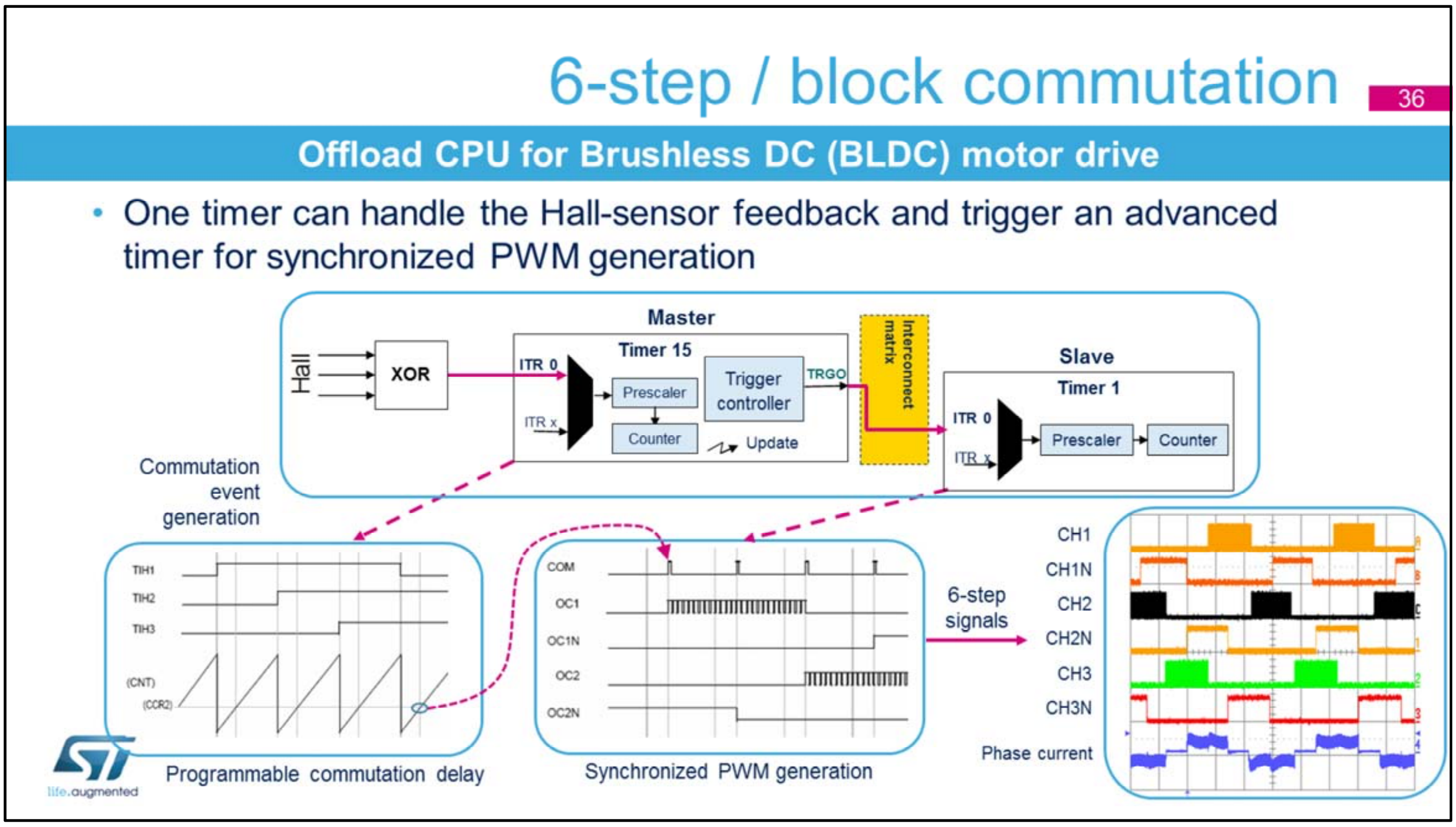
Asymmetrical dead time

- Asymmetrical: new register with enable bit for supporting both legacy mode (symmetrical) and the new mode
  - For applications having asymmetrical gate driver / opto-couplers propagation delays
  - Shadow register for on-the fly deadtime update (adaptive deadtime schemes, for digital SMPS or lighting applications)


Symmetrical deadtime (DTAE = 0)
OCxREF
OCx
OCxN
DTG[7:0]
DTG[7:0]
Asymmetrical deadtime (DTAE = 1)
OCxREF
OCx
OCxN
DTGF[7:0]
DTG[7:0]

life.augmented

In symmetrical mode, the dead time duration is the same on both falling and rising edge of the OCx output.
In asymmetrical mode, the rising edge-referred deadtime is defined by the DTG[7:0] bitfield, while the falling edge-referred is defined by the DTGF[7:0] bitfield.
It is useful for applications having asymmetrical gate driver or opto-coupler propagation delays.
It is possible to have the deadtime value updated on-the-fly during pwm operation, using a preload mechanism.
The preload value is loaded in the active register on the next update event.



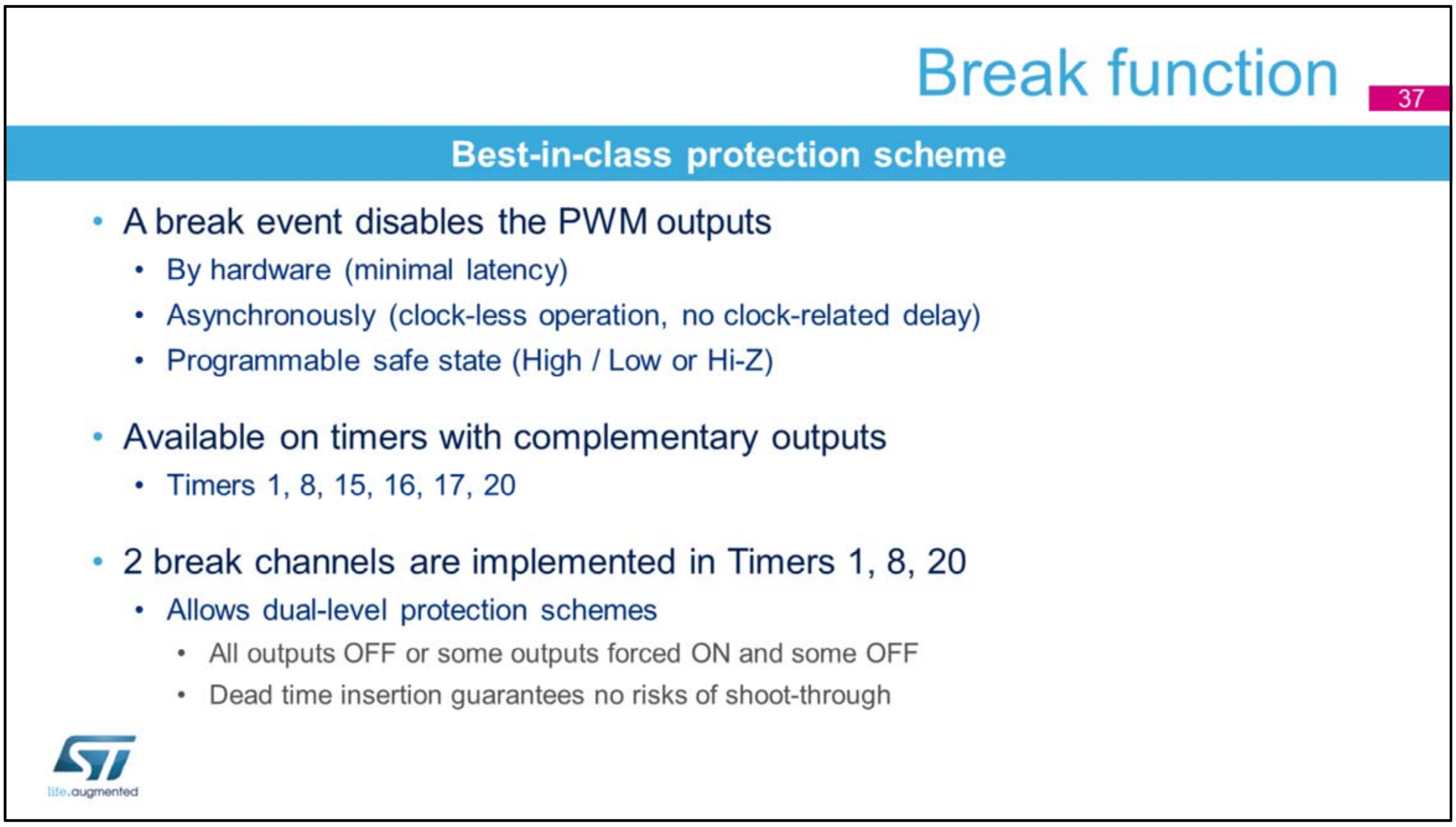
6-step / block commutation

Offload CPU for Brushless DC (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

37

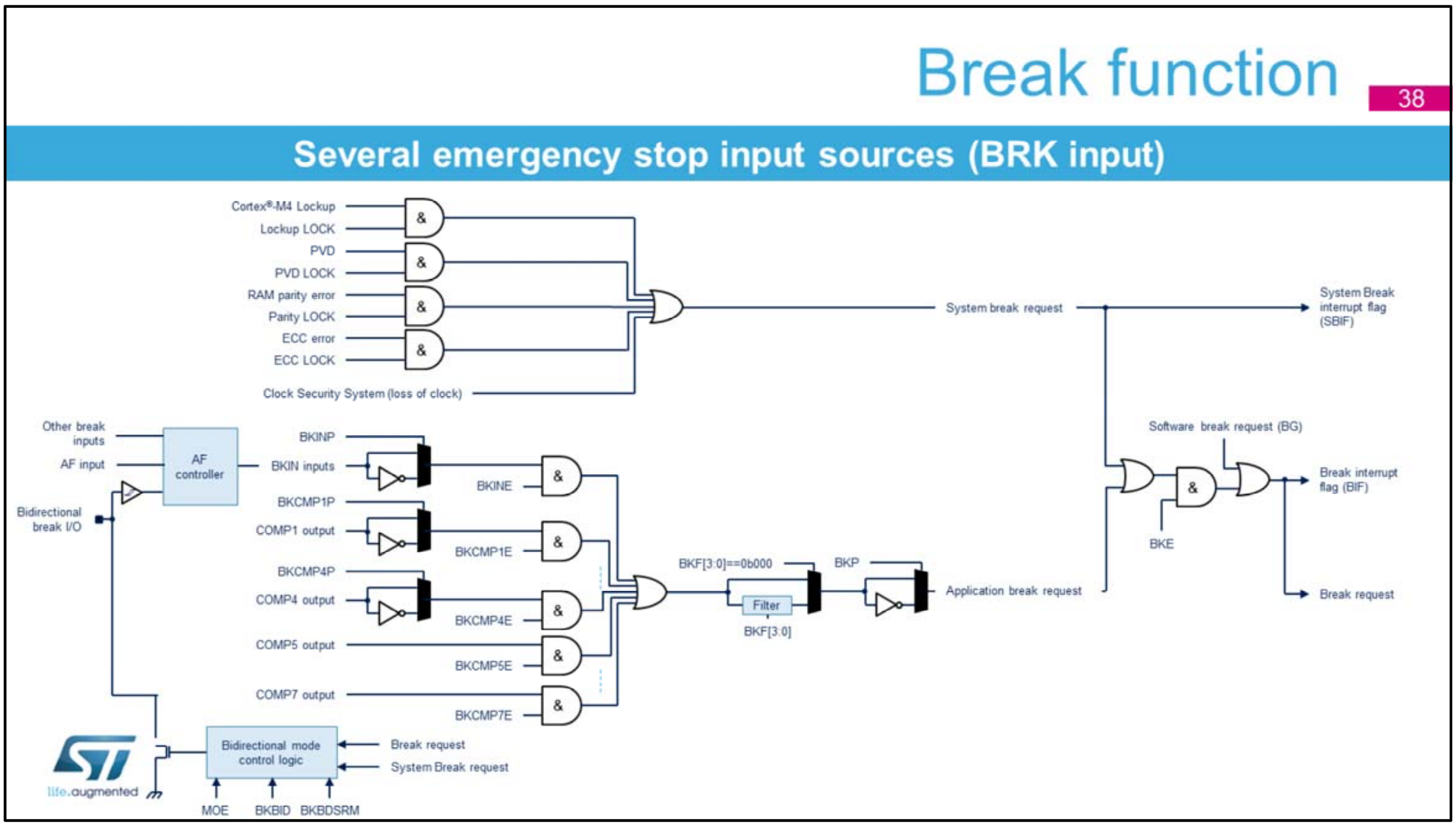
Best-in-class protection scheme

- A break event disables the PWM outputs
  - By hardware (minimal latency)
  - Asynchronously (clock-less operation, no clock-related delay)
  - Programmable safe state (High / Low or Hi-Z)
- Available on timers with complementary outputs
  - Timers 1, 8, 15, 16, 17, 20
- 2 break channels are implemented in Timers 1, 8, 20
  - 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

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, 8, 15, 16, 17 and 20.
Timers 1, 8 and 20 have two separated break channels, while Timers 15, 16 and 17 support a unique break channel.

Supporting two break channels 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

38

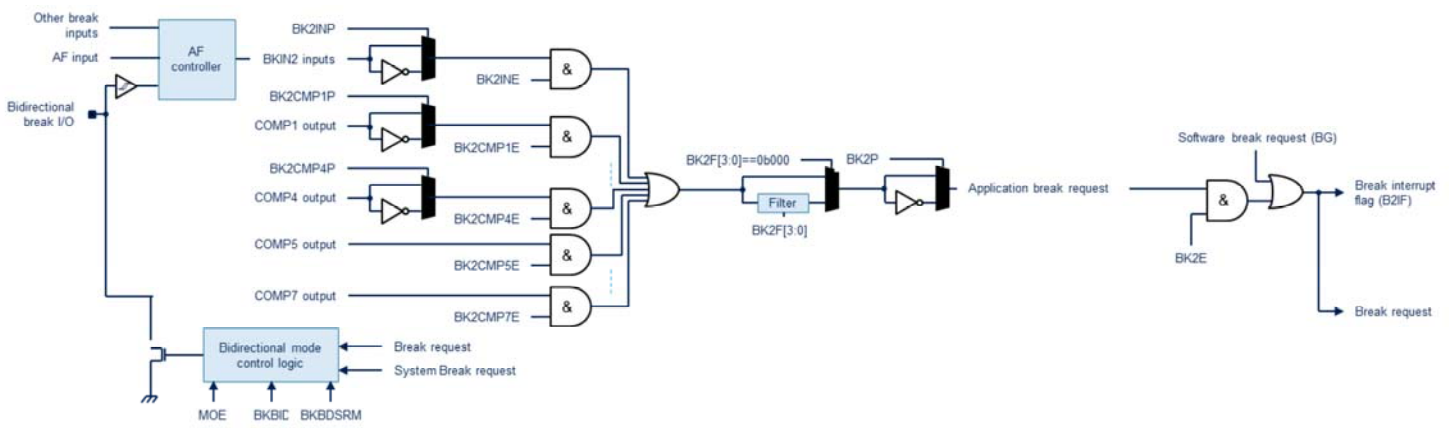
Several emergency stop input sources (BRK input)

This slide describes the break function related to the BRK input.
Multiple break sources can be combined for triggering a break event.
A system break request results from serious errors detected in the MCU: CPU lockup, Power Voltage Drop, RAM parity error, flash ECC error and loss of clock.
The Lock signals are write-once enable bits located in the SYSCFG peripheral.
They are reset by default (fault error is masked).
Once set (fault detection is enabled), they cannot be reset unless the whole MCU is reset, for functional safety.
An application break request results from a board failure.
Break inputs can also be selected with the alternate function controller, on the microcontroller pinout.
External sources can be conditioned before entering the

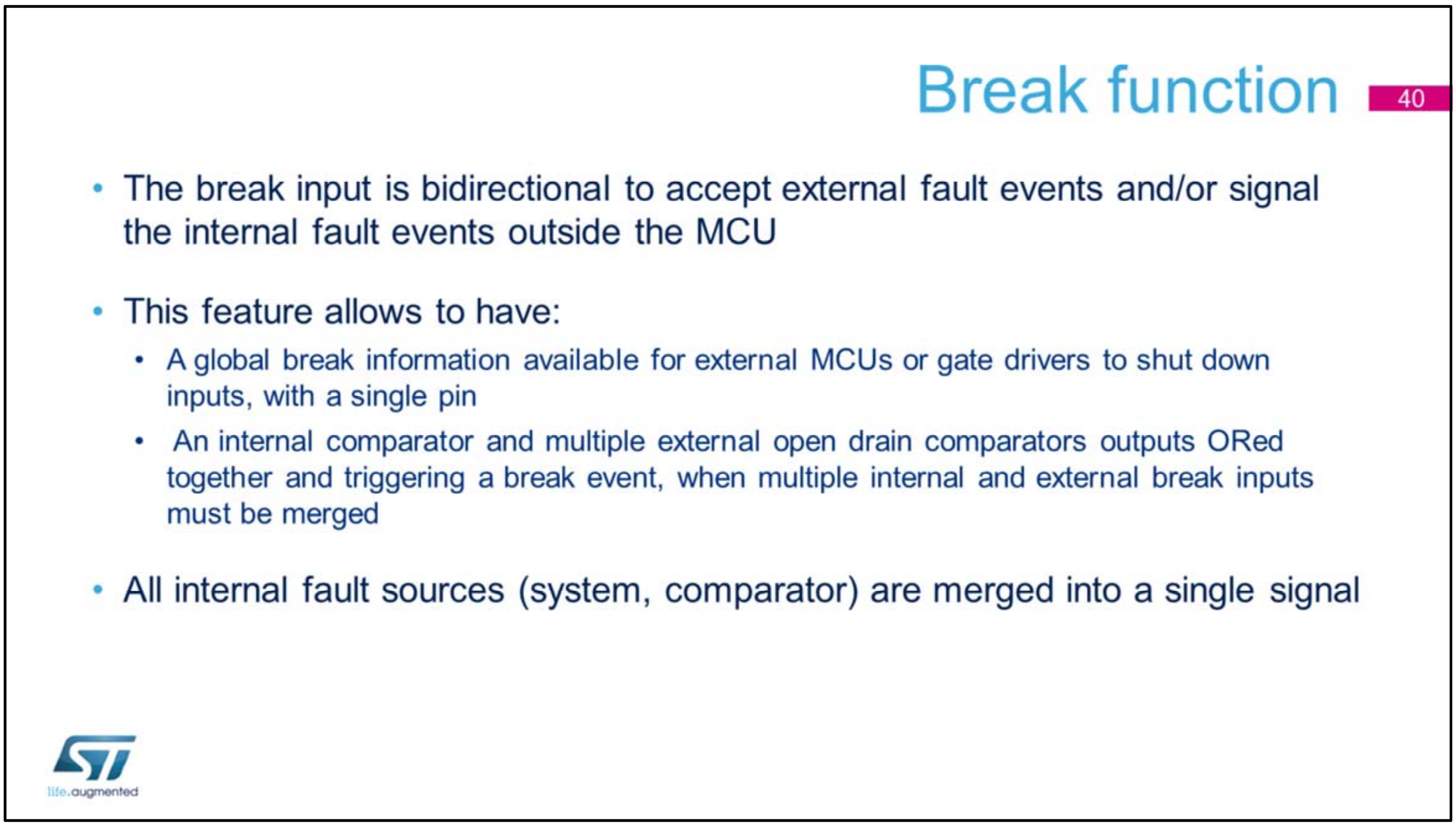
break detection unit in order to select the proper polarity and discarding of spurious glitches by means of a digital filter.
Software is also capable of requesting a break, typically when diagnosing an unrecoverable error condition.
When one of these break requests occurs, an interrupt or DMA request is asserted, in addition to set the PWM outputs in a safe state.

Break function

Several emergency stop input sources (BRK2 input)



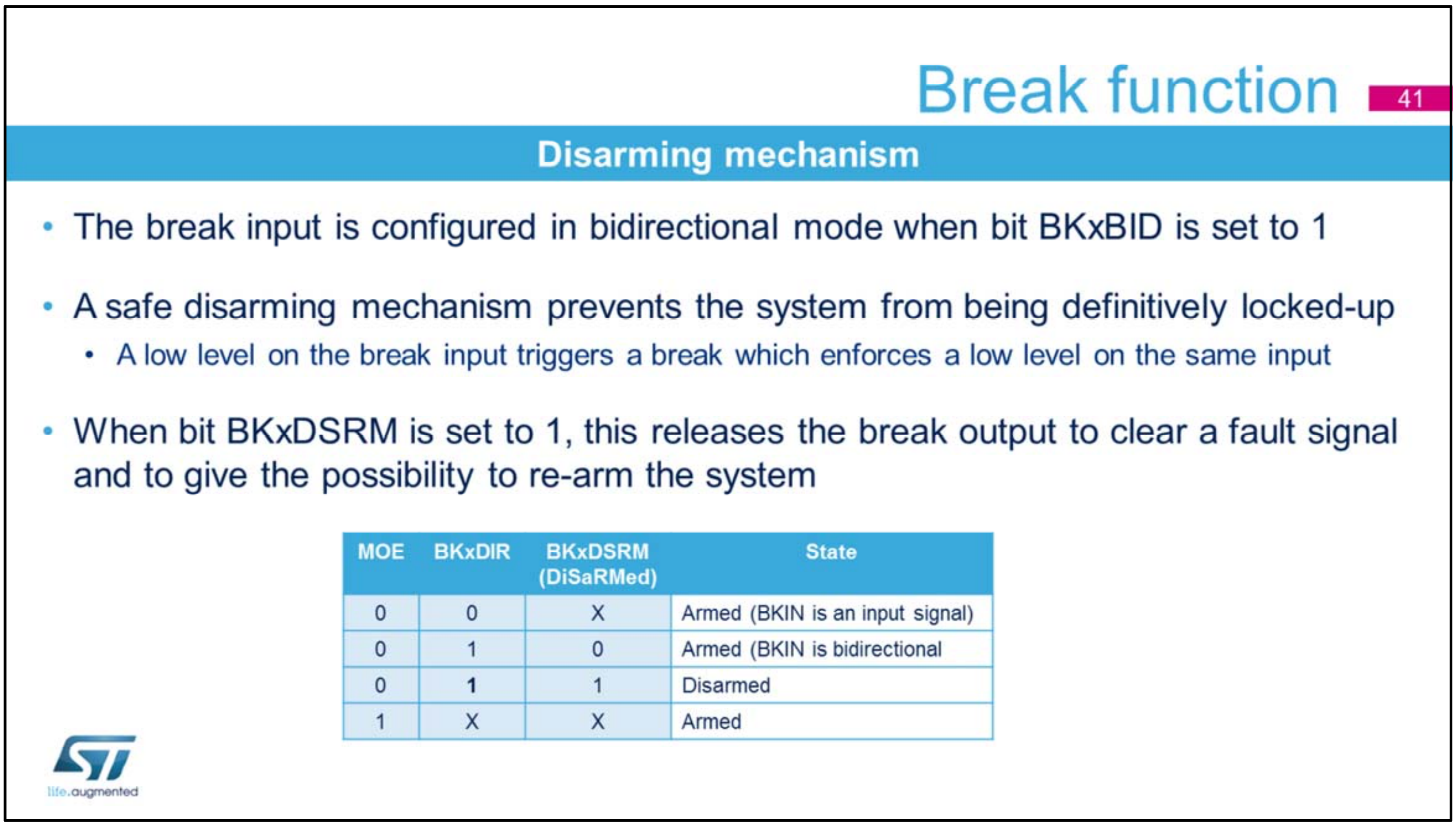
This slide describes the break function related to the BRK2 input.
Unlike the BRK input, the BRK2 event can be caused neither by a system break request nor by a software break request.



Break function

- The break input is bidirectional to accept external fault events and/or signal the internal fault events outside the MCU
- This feature allows to have:
  - A global break information available for external MCUs or gate drivers to shut down inputs, with a single pin
  - An internal comparator and multiple external open drain comparators outputs ORed together and triggering a break event, when multiple internal and external break inputs must be merged
- All internal fault sources (system, comparator) are merged into a single signal

The STM32G4 implements a bidirectional break.
The break IO pad is used to report board-level failure, but can also be used as an output to signal a microcontroller internal failure.
The bidirectional mode is available for both the Break and Break 2 inputs, and require the I/O to be configured in open-drain mode with active low polarity.
A global break information detected inside the STM32G4 can therefore be output to other CPUs or gate drivers.
Internal break sources and multiple external open drain comparator outputs are ORed together to trigger a unique break event, when multiple internal and external break sources must be merged.



Break function

41

Disarming mechanism

- The break input is configured in bidirectional mode when bit BKxBID is set to 1
- A safe disarming mechanism prevents the system from being definitively locked-up
  - A low level on the break input triggers a break which enforces a low level on the same input
- When bit BKxDSRM is set to 1, this releases the break output to clear a fault signal and to give the possibility to re-arm the system

MOE	BKxDIR	BKxDSRM (DiSaRMed)	State
0	0	X	Armed (BKIN is an input signal)
0	1	0	Armed (BKIN is bidirectional
0	1	1	Disarmed
1	X	X	Armed

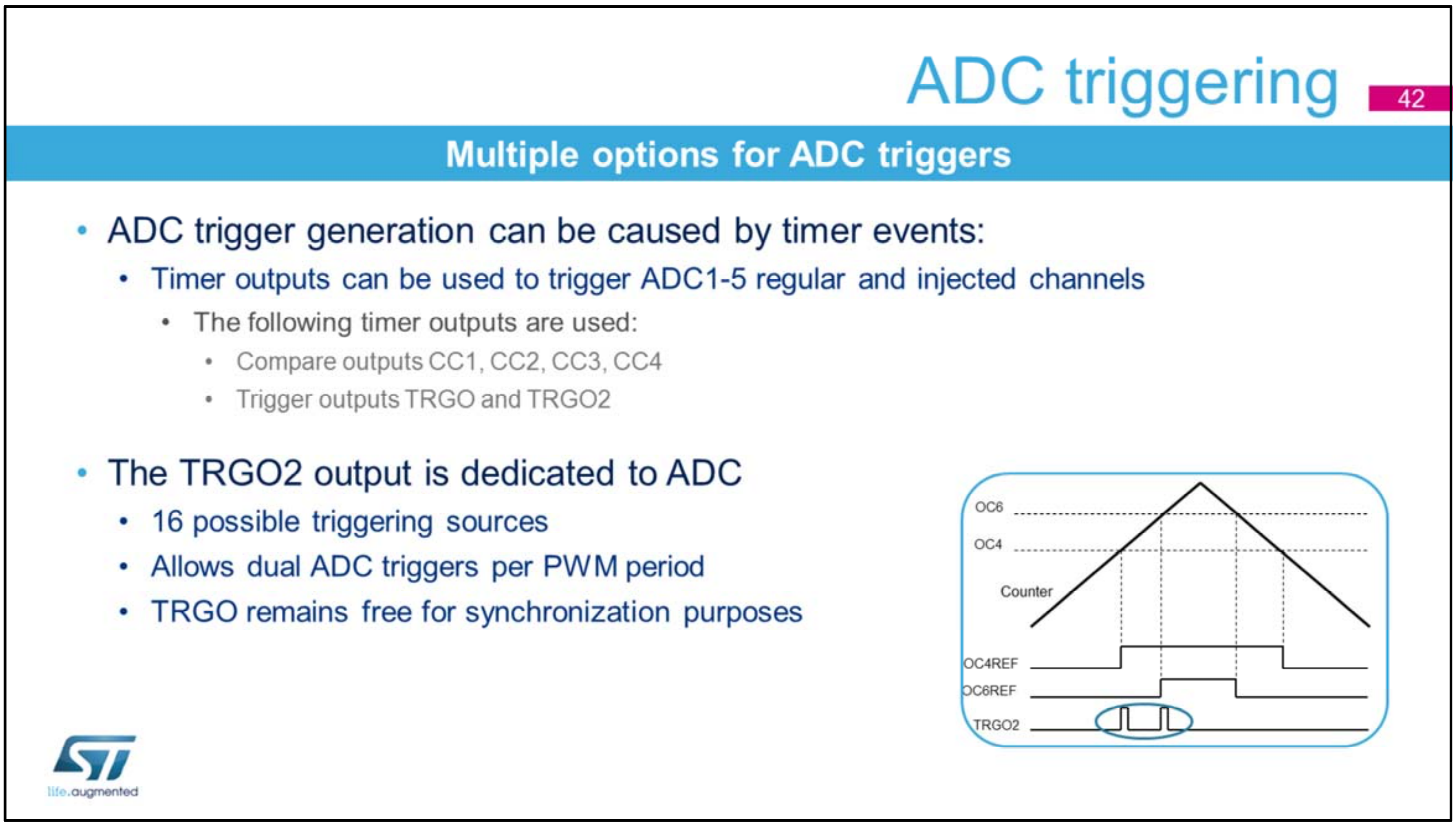
life.augmented

This slide explains how to arm and disarm the break circuitry.
The bidirectional mode is active when the BKxBID bit is set. x means BK or BK2 break signal.
Since the break IO pad is bidirectional, a low level on the break input triggers a break which enforces a low level on the same pad.
Therefore a disarming mechanism is required to exit the break condition.
The Main Output Enable (MOE) bit is relevant when a channel is configured as an output.
It is cleared asynchronously by hardware as soon as one of the break inputs is active to disable OC and OCN outputs.
The following sequence should be used to re-arm the protection after a break event:

- The BKDSRM (BK2DSRM) bit must be set to release the output control

- The software must wait until the system break condition disappears (if any) and clear the SBIF status flag (or clear it systematically before re-arming)
- The software must poll the BKDSRM (BK2DSRM) bit until it is cleared by hardware (when the application break condition disappears)

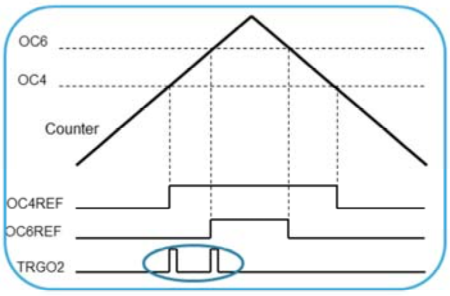
From this point, the break circuitry is armed and active, and the MOE bit can be set to re-enable the PWM outputs.



ADC triggering

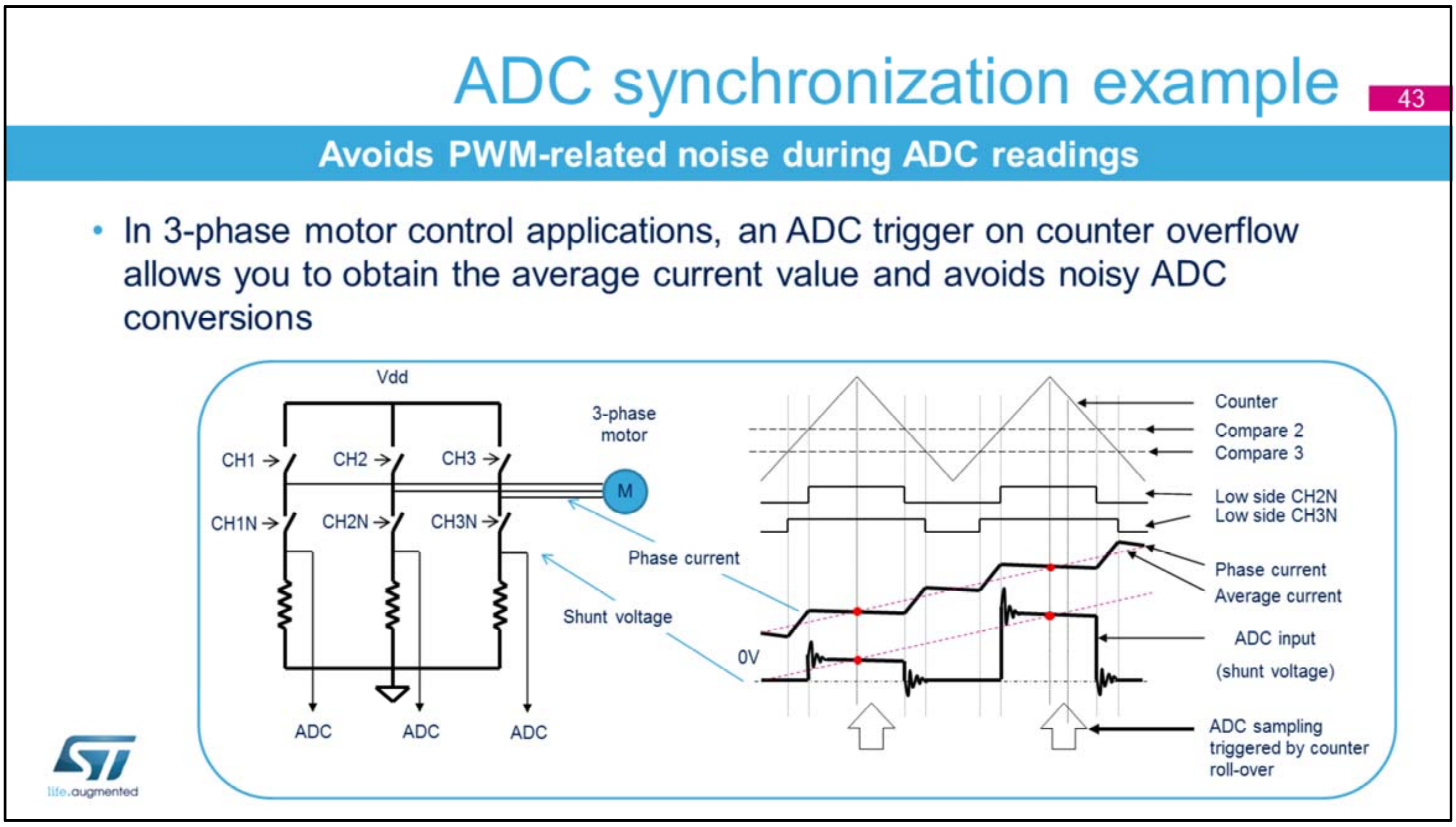
Multiple options for ADC triggers

- ADC trigger generation can be caused by timer events:
  - Timer outputs can be used to trigger ADC1-5 regular and injected channels
    - The following timer outputs are used:
      - Compare outputs CC1, CC2, CC3, CC4
      - Trigger outputs TRGO and TRGO2
- The TRGO2 output is dedicated to ADC
  - 16 possible triggering sources
  - Allows dual ADC triggers per PWM period
  - TRGO remains free for synchronization purposes



This slide presents the ADC triggering options related to the timers.
The ADCs can be triggered with most of the STM32 timers.
This can be done using compare events: the ADC conversion will start on a given compare match. This applies to timer channels 1 to 4.
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 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, as shown in the figure, or compare 5 and 6 events.
This also leaves the TRGO free for multiple timer synchronization schemes.



ADC synchronization example

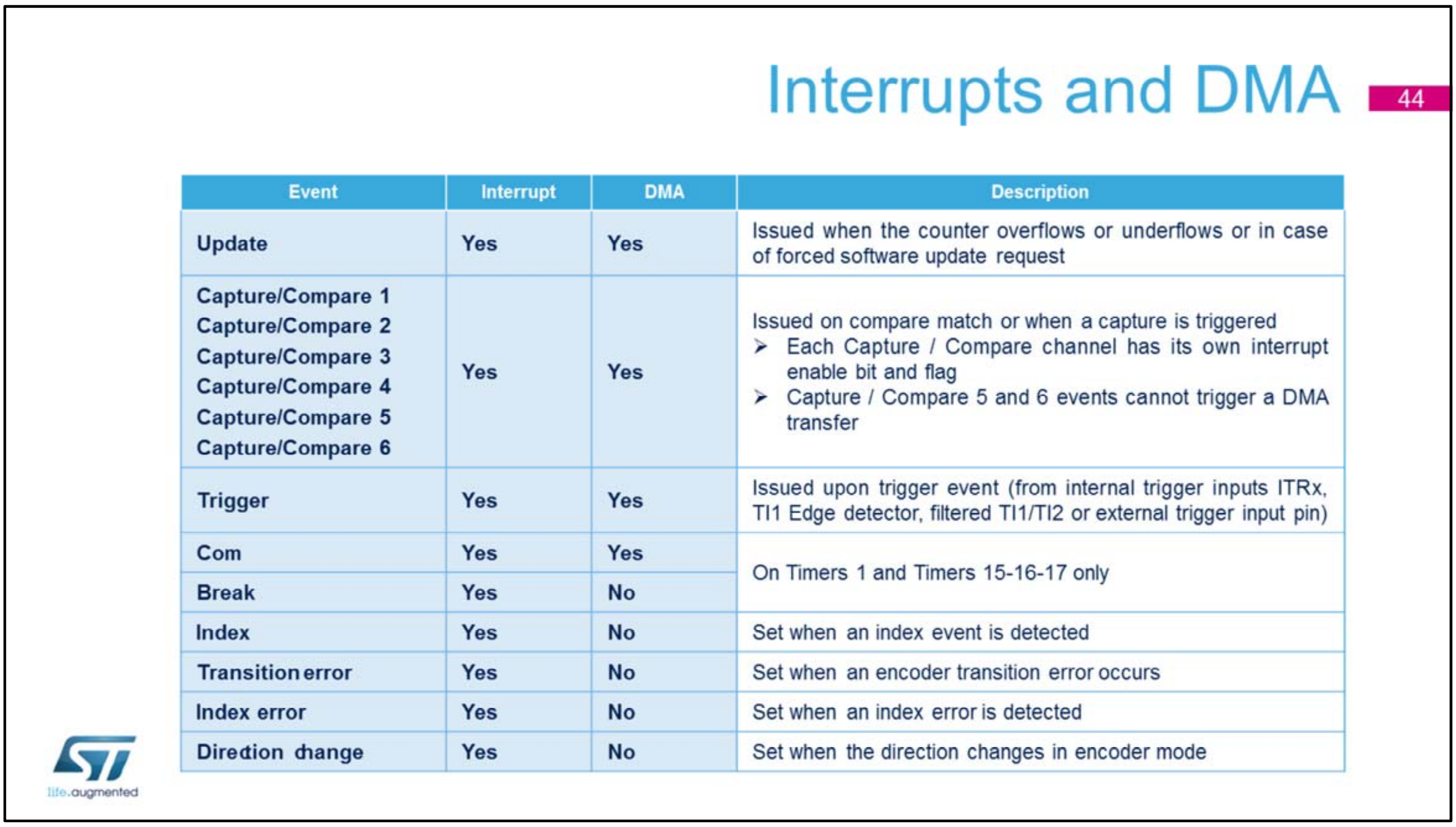
43

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

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.



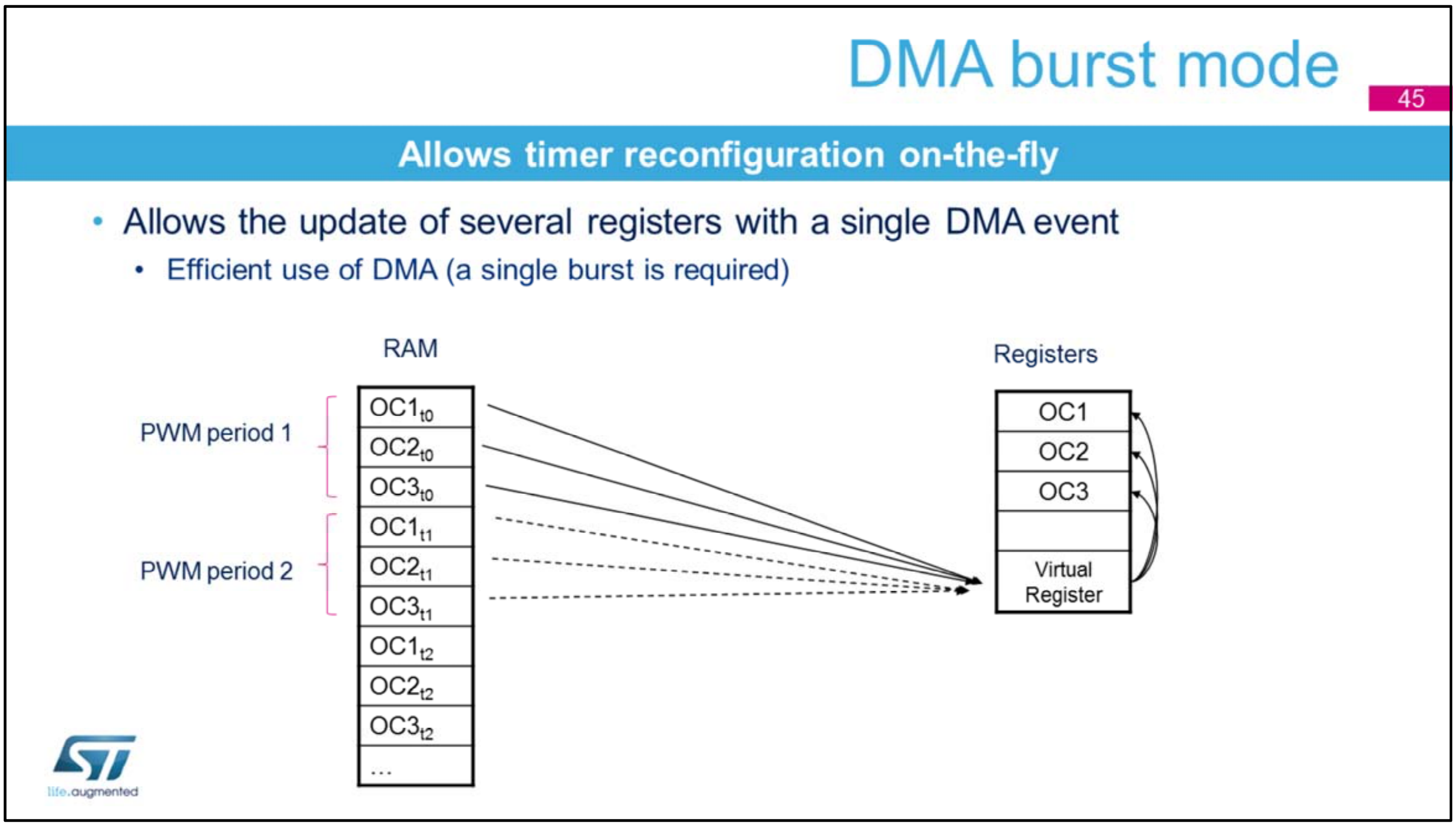
Interrupts and DMA

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 Capture/Compare 5 Capture/Compare 6	Yes	Yes	Issued on compare match or when a capture is triggered ➤ Each Capture / Compare channel has its own interrupt enable bit and flag ➤ Capture / Compare 5 and 6 events cannot trigger a DMA transfer
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 Timers 15-16-17 only
Break	Yes	No	
Index	Yes	No	Set when an index event is detected
Transition error	Yes	No	Set when an encoder transition error occurs
Index error	Yes	No	Set when an index error is detected
Direction change	Yes	No	Set when the direction changes in encoder mode

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 six capture/compare events have their own interrupt.
Only channels 1 to 4 can trigger a DMA transfer.
A trigger event on the TRGI input (regardless of the trigger source) can trigger an interrupt or DMA request.
In encoder mode, the counter can be reset by the an Index signal coming from the encoder, indicating an

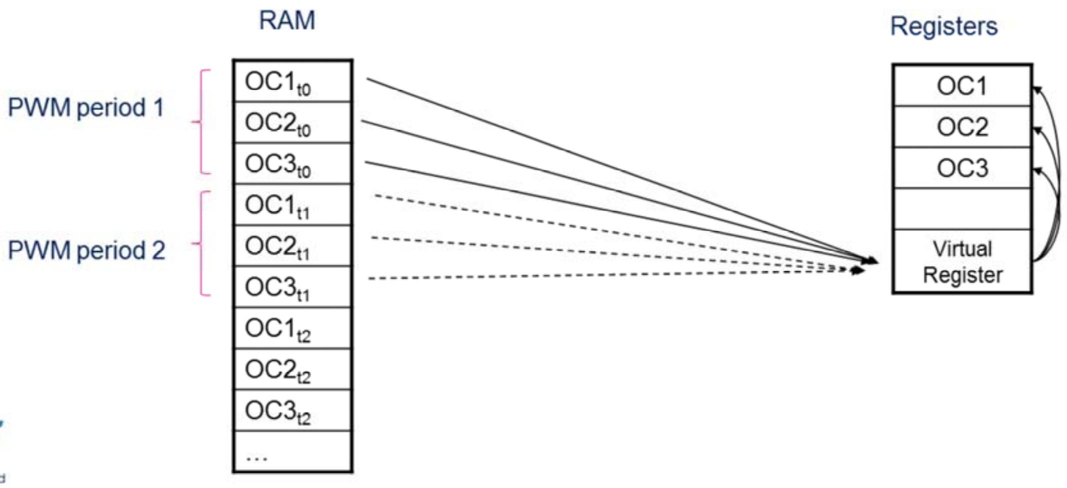
absolute reference position.
This index event can also cause an interrupt.
Furthermore, the encoder mode can detect two errors: transition error and index error that can also cause interrupts.
An interrupt can be caused by the detection of a direction change in encoder mode.
Lastly, additional sources of interrupts and DMA requests are the commutation and break events.



DMA burst mode

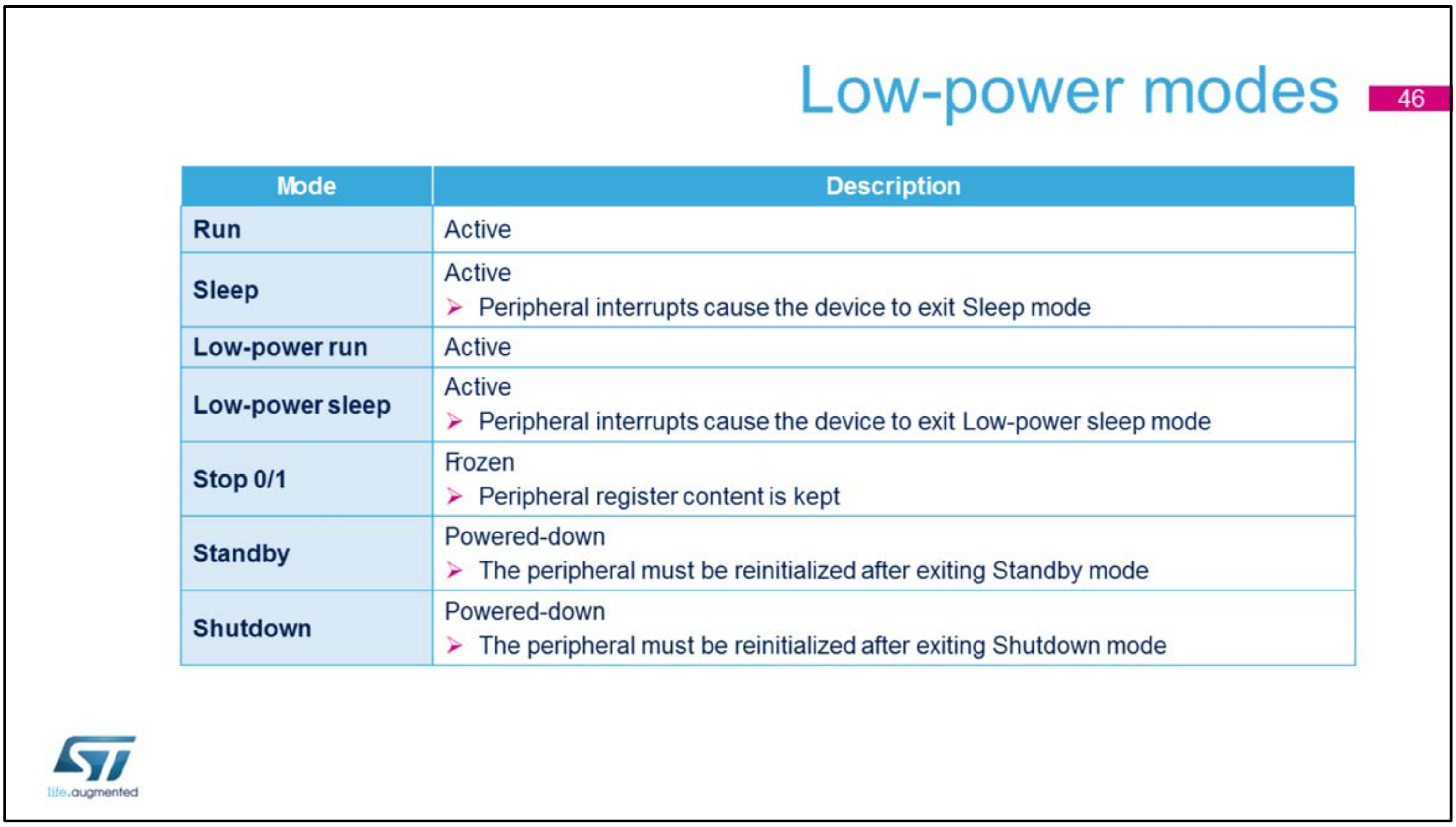
Allows timer reconfiguration on-the-fly

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



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 burst 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.

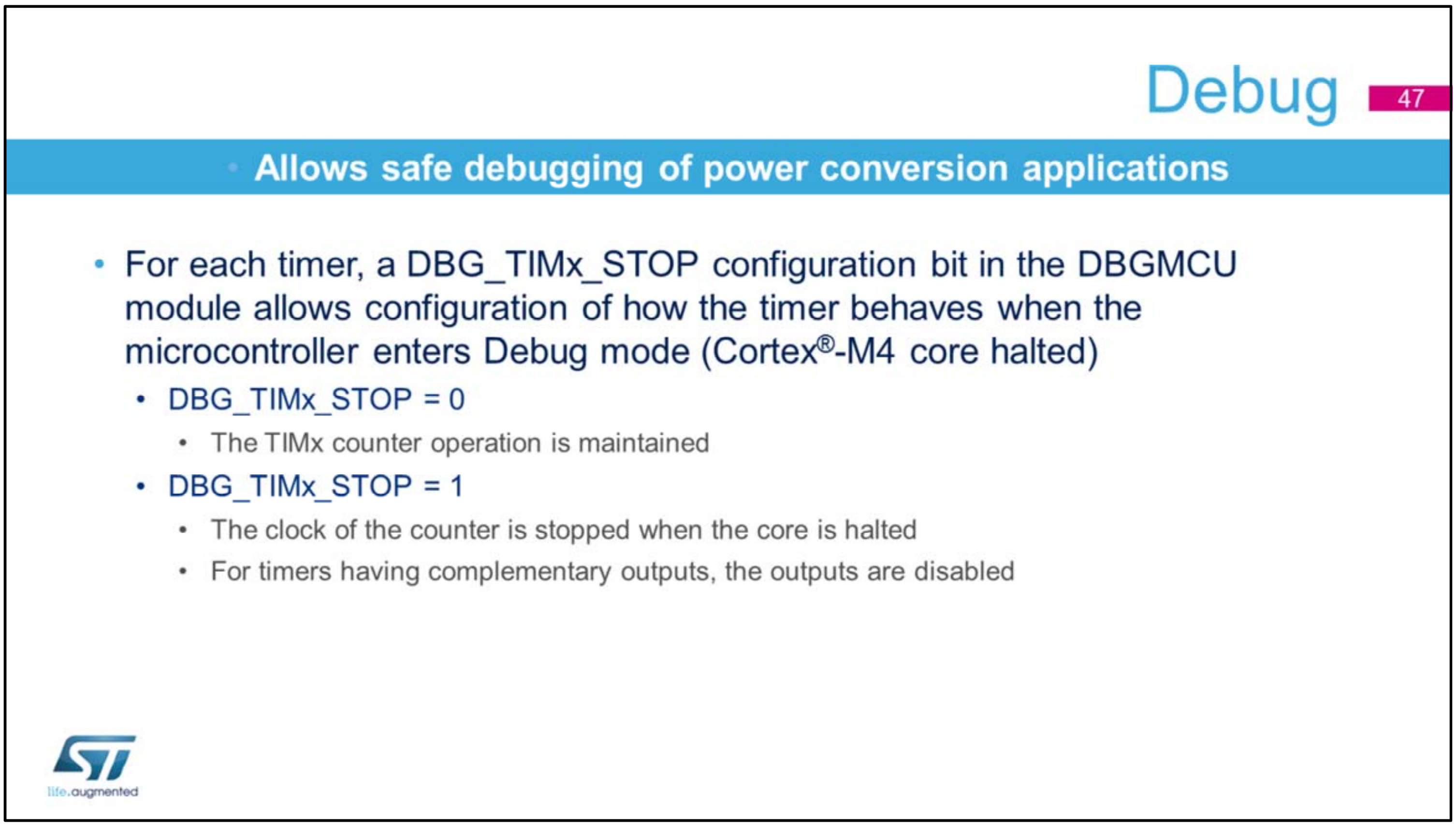


Low-power modes

Mode	Description
Run	Active
Sleep	Active ➢ Peripheral interrupts cause the device to exit Sleep mode
Low-power run	Active
Low-power sleep	Active ➢ Peripheral interrupts cause the device to exit Low-power sleep mode
Stop 0/1	Frozen ➢ Peripheral register content is kept
Standby	Powered-down ➢ The peripheral must be reinitialized after exiting Standby mode
Shutdown	Powered-down ➢ The peripheral must be reinitialized after exiting Shutdown mode

life.augmented

This table indicates the state of the general-purpose timers (GPT) according to the current MCU power mode.
In Run, Low-power run, Sleep and Low-power sleep modes, the general-purpose timers are active.
In Sleep and Low-power sleep modes, GPT interrupts cause the device to exit respectively Sleep or Low-power sleep mode.
In Stop 0 and 1 modes, the general-purpose timers are frozen. The peripheral register content is kept. No reconfiguration of the GPT is needed when exiting these modes.
In Standby and Shutdown modes, the general-purpose timers are powered down and must therefore be reinitialized upon exit of these modes.
Note that for low-power use cases, the LPTIMER offers extended features, such as full operation and wake-up capabilities in Stop 0 and 1 modes.



Debug

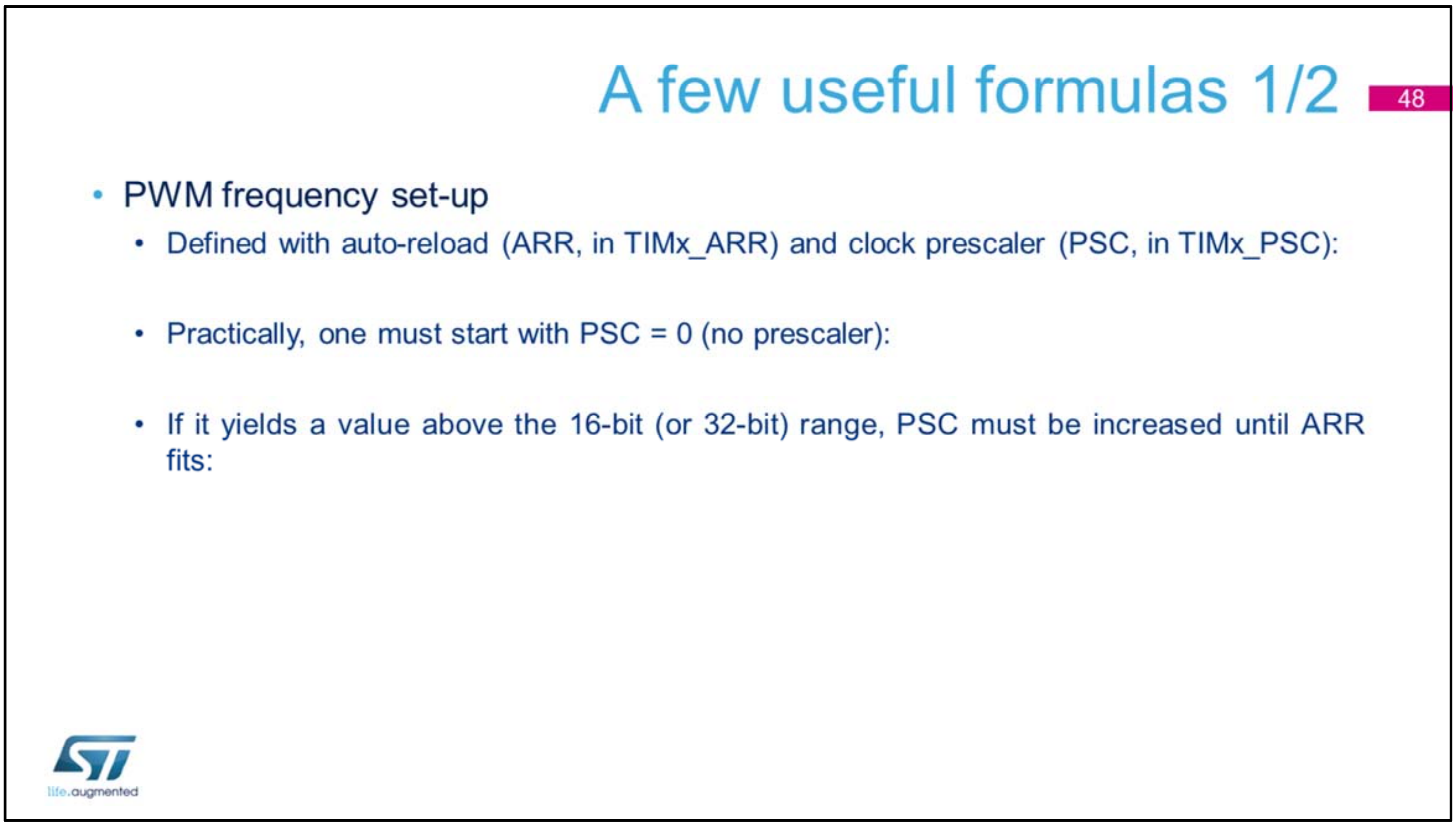
Allows safe debugging of power conversion applications

- For each timer, a DBG_TIMx_STOP configuration bit in the DBGMCU module allows configuration of how the timer behaves when the microcontroller enters Debug mode (Cortex®-M4 core halted)
  - DBG_TIMx_STOP = 0
    - The TIMx counter operation is maintained
  - DBG_TIMx_STOP = 1
    - The clock of the counter is stopped when the core is halted
    - For timers having complementary outputs, the outputs are disabled

The timer’s state in Debug mode can be configured with one configuration bit per timer.

➢ 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

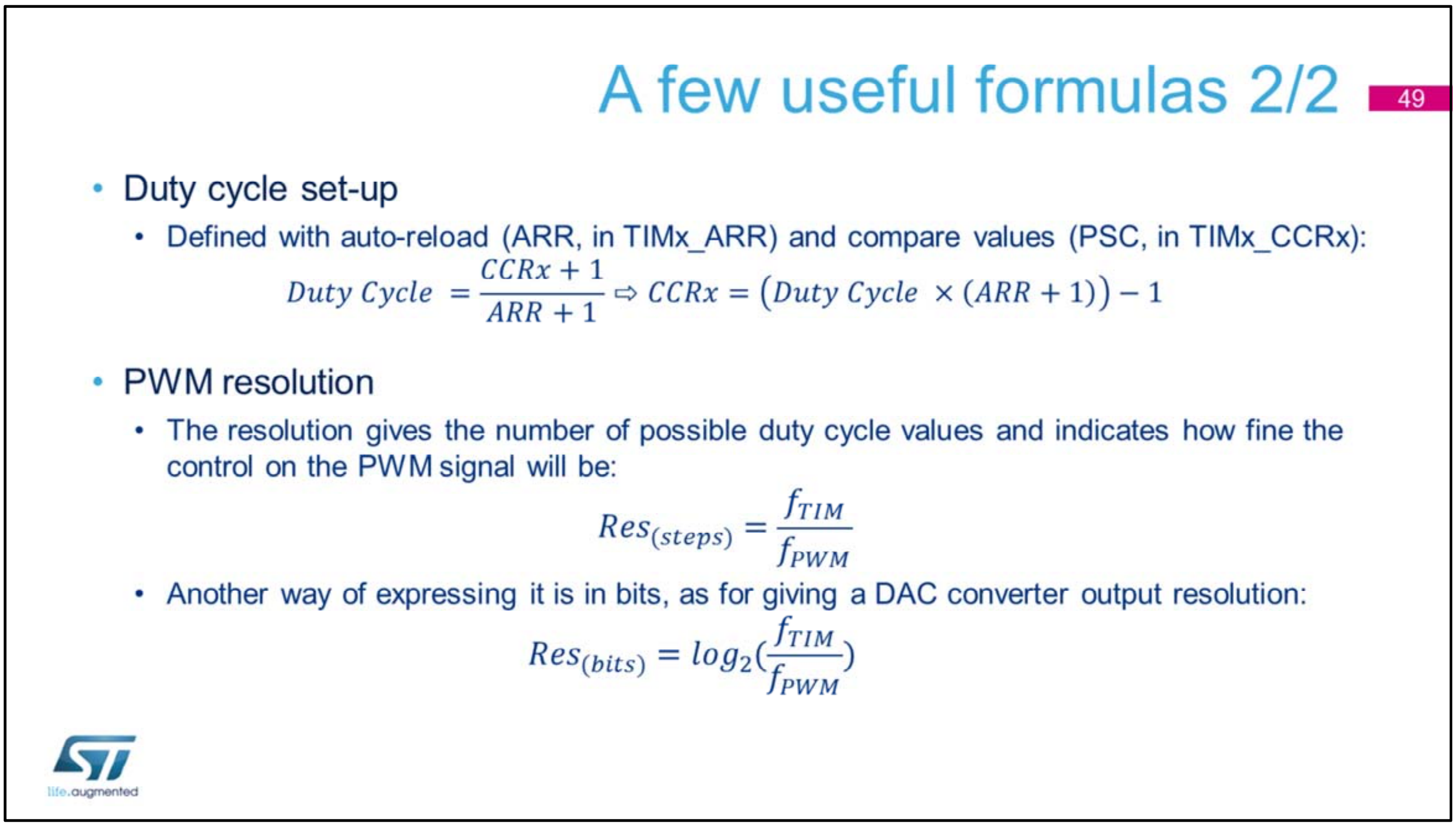
48

- PWM frequency set-up
  - Defined with auto-reload (ARR, in TIMx_ARR) and clock prescaler (PSC, in TIMx_PSC):
  - Practically, one must start with PSC = 0 (no prescaler):
  - If it yields a value above the 16-bit (or 32-bit) range, PSC must be increased until ARR fits:

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

49

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

$$Duty\ Cycle = \frac{CCRx + 1}{ARR + 1} \Rightarrow CCRx = \big(Duty\ Cycle \times (ARR + 1)\big) - 1$$

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

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

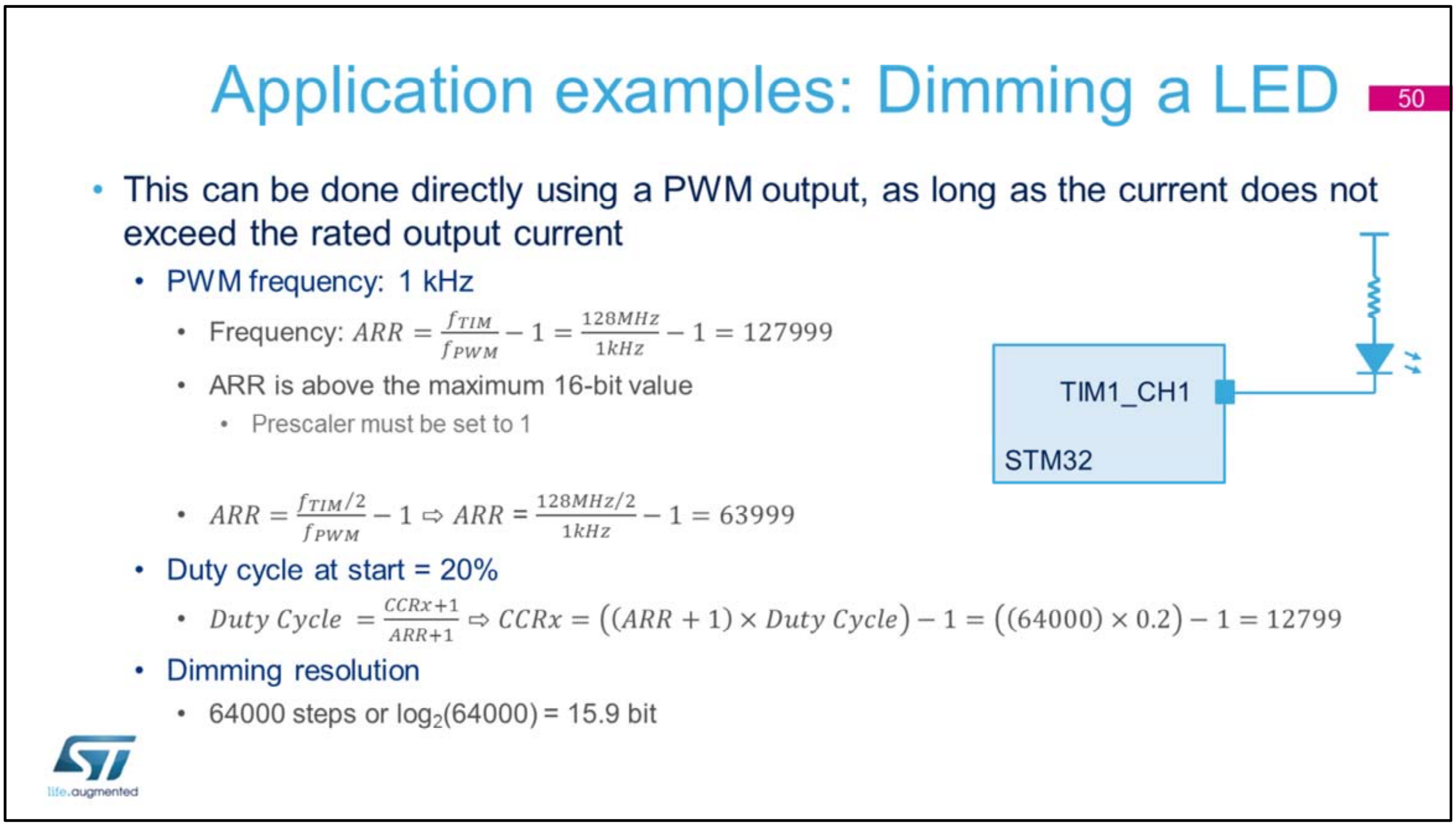
  - Another way of expressing it is in bits, as for giving a DAC converter output resolution:

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

life.augmented

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

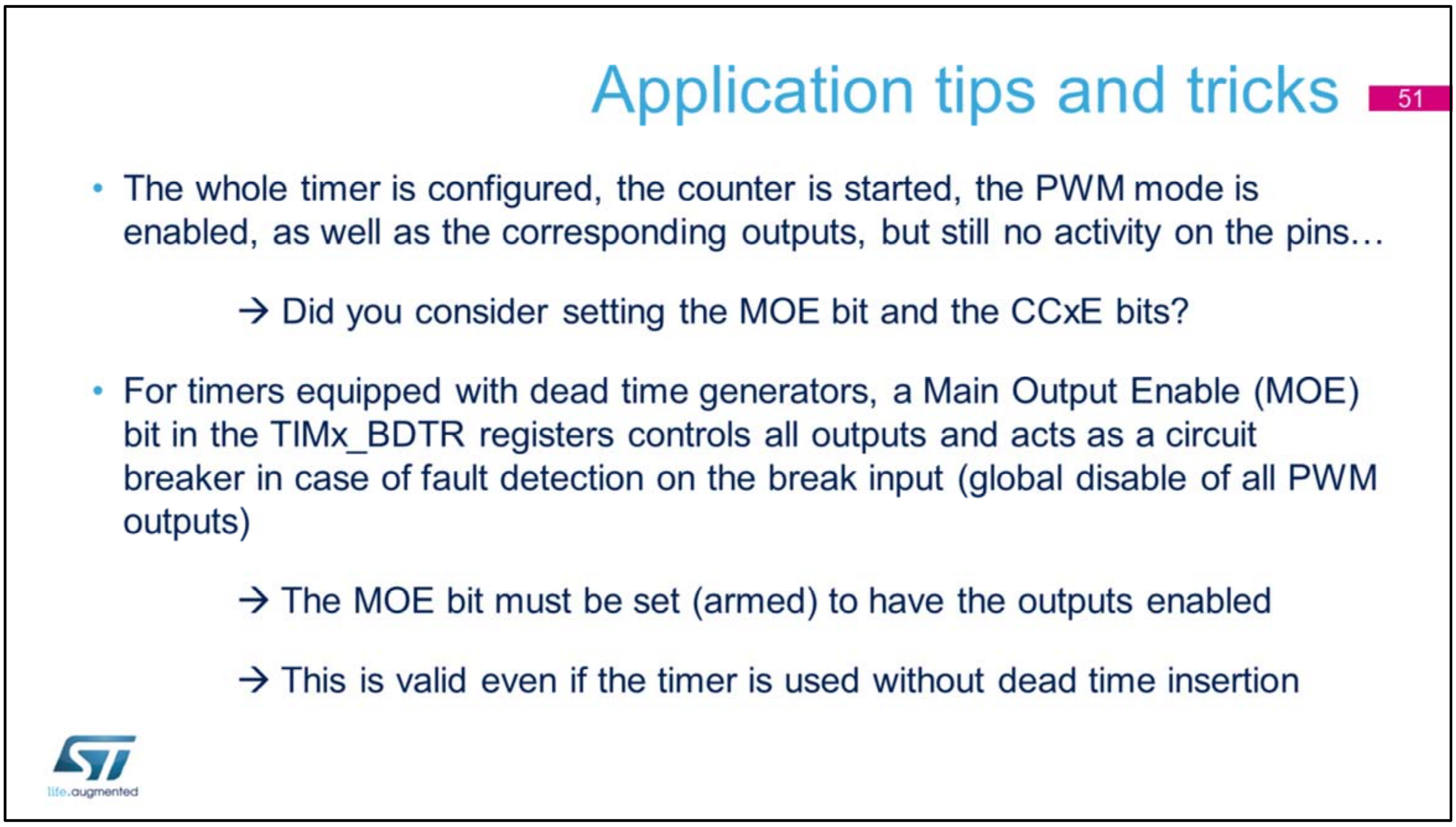
50

- 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: $ARR = \frac{f_{TIM}}{f_{PWM}} - 1 = \frac{128MHz}{1kHz} - 1 = 127999$
    - ARR is above the maximum 16-bit value
      - Prescaler must be set to 1
    - $ARR = \frac{f_{TIM}/2}{f_{PWM}} - 1 \Rightarrow ARR = \frac{128MHz/2}{1kHz} - 1 = 63999$
  - Duty cycle at start = 20%
    - $Duty\ Cycle = \frac{CCRx+1}{ARR+1} \Rightarrow CCRx = \big((ARR+1) \times Duty\ Cycle\big) - 1 = \big((64000) \times 0.2\big) - 1 = 12799$
  - Dimming resolution
    - 64000 steps or $\log_2(64000) = 15.9$ 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 128 MHz, the result is 127999, which is above the 16-bit range that can be used with Timer 1.
The timer prescaler must be set to 1 to have the timer operating at 64 MHz and this results in a valid value of 63999 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 12799.
Lastly, the dimming resolution can be computed from formulas presented in the previous slides. With a timer

running at 64 MHz, a 1 kHz PWM provides 640000 dimming steps, which corresponds to an equivalent resolution of 15.9 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?

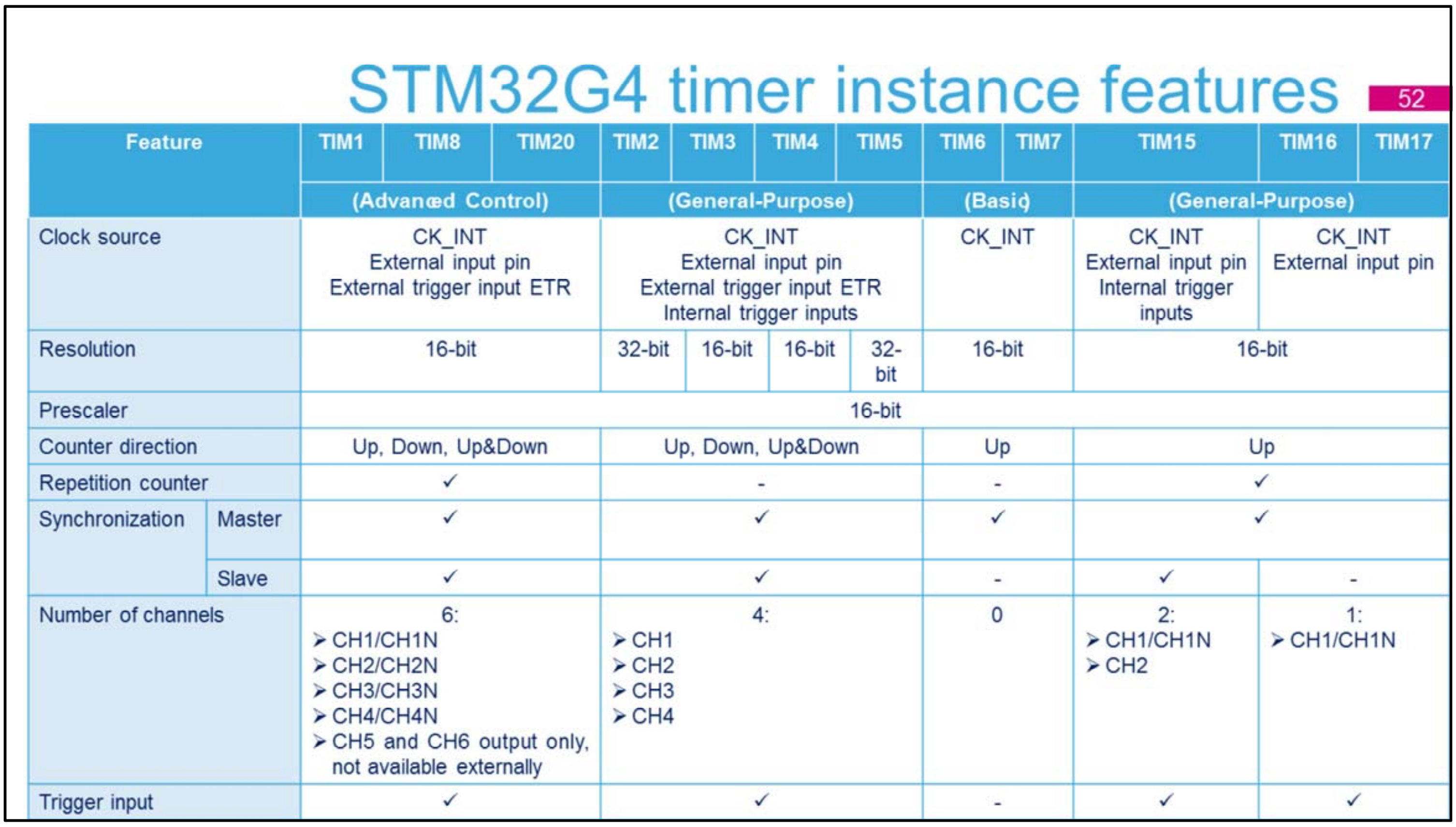
- For timers equipped with dead time generators, a Main Output Enable (MOE) bit in the TIMx_BDTR registers controls all outputs and acts as a circuit breaker in case of fault detection on the break input (global disable of all PWM outputs)

  → 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

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, 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.

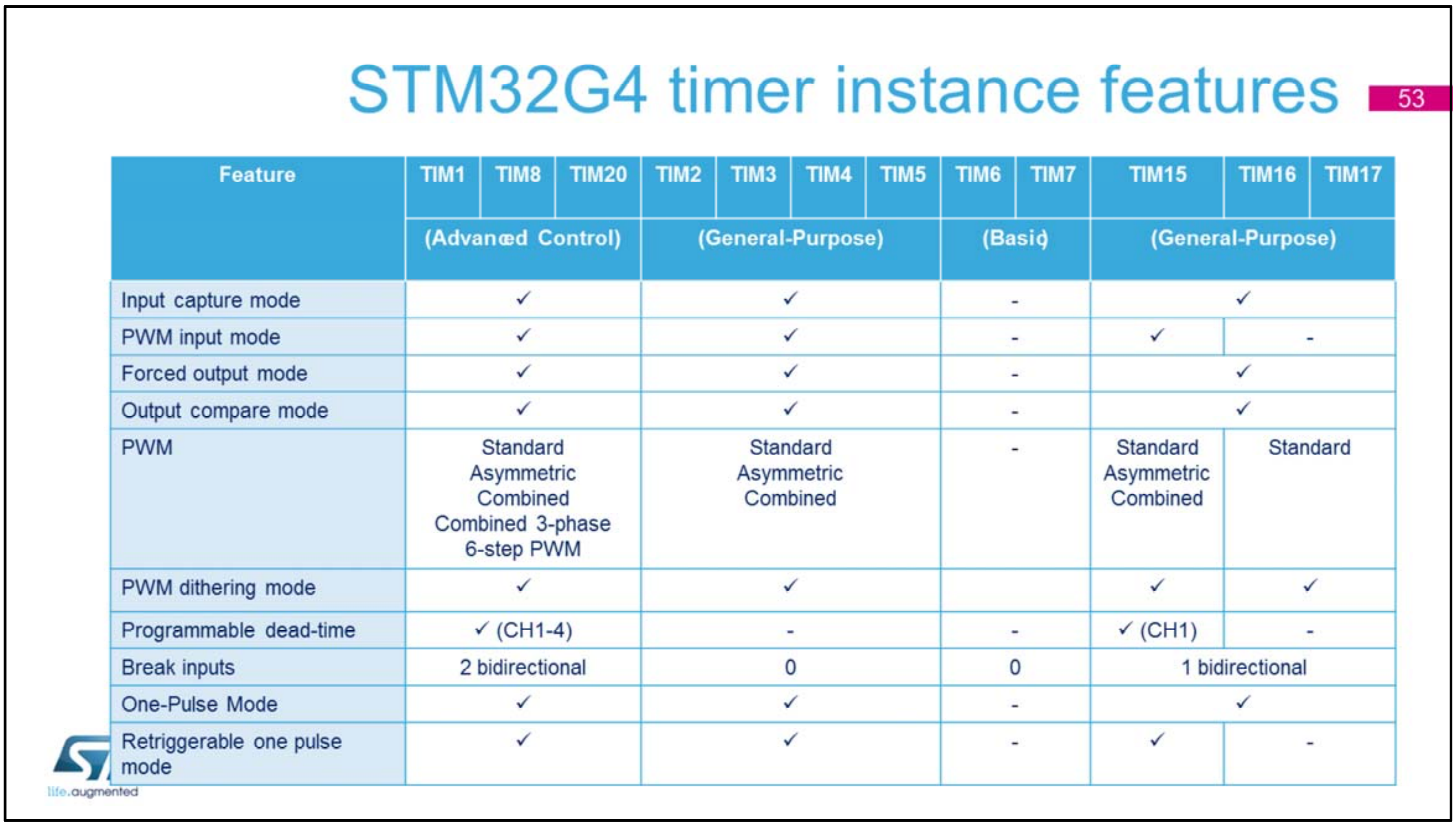


STM32G4 timer instance features

52

Feature		TIM1	TIM8	TIM20	TIM2	TIM3	TIM4	TIM5	TIM6	TIM7	TIM15	TIM16	TIM17
		(Advanced Control)			(General-Purpose)				(Basic)		(General-Purpose)		
Clock source		CK_INT External input pin External trigger input ETR			CK_INT External input pin External trigger input ETR Internal trigger inputs				CK_INT		CK_INT External input pin Internal trigger inputs	CK_INT External input pin	
Resolution		16-bit			32-bit	16-bit	16-bit	32-bit	16-bit		16-bit		
Prescaler		16-bit											
Counter direction		Up, Down, Up&Down			Up, Down, Up&Down				Up		Up		
Repetition counter		✓			-				-		✓		
Synchronization	Master	✓			✓				✓		✓		
	Slave	✓			✓				-		✓	-	
Number of channels		6: ➢ CH1/CH1N ➢ CH2/CH2N ➢ CH3/CH3N ➢ CH4/CH4N ➢ CH5 and CH6 output only, not available externally			4: ➢ CH1 ➢ CH2 ➢ CH3 ➢ CH4				0		2: ➢ CH1/CH1N ➢ CH2	1: ➢ CH1/CH1N	
Trigger input		✓			✓				-		✓	✓	

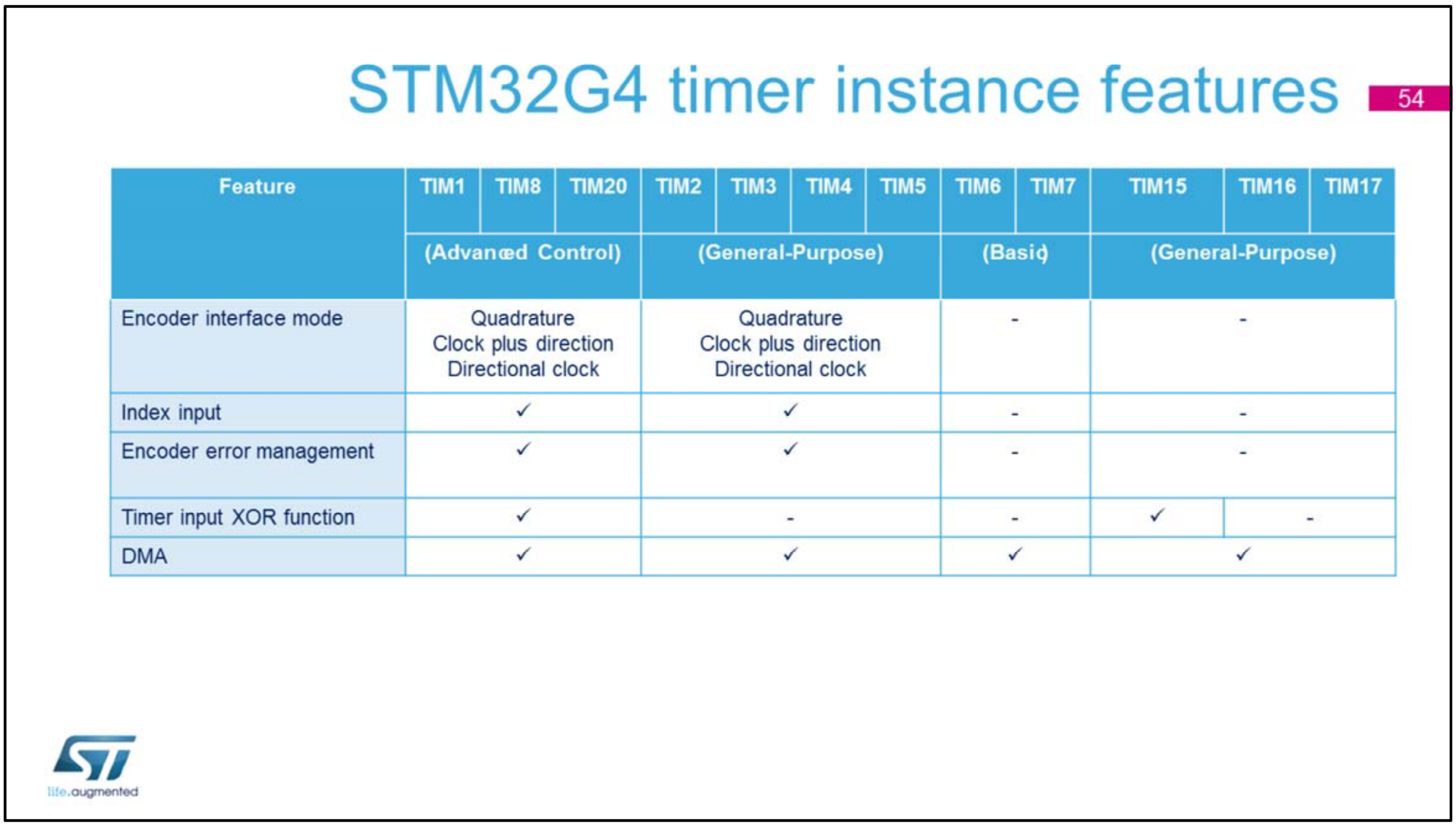
This slide lists the timer instances present in STM32G4 microcontrollers.
Timers 1, 8 and 20 are full-featured timer, motor control capable, including all PWM options and six compare channels for being able to generate simultaneously 3-phase PWM signals and have two independent ADC triggers.
Timers 15, 16, 17 are general-purpose timers.
Only Timer 15 supports a complementary channel and advanced PWM modes.
Timers 2, 3, 4 and 5 are general-purpose timers, including advanced PWM modes, up-down counting capability and 4 channels. Timer 2 and 5 additionally offer a 32-bit counting range.
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.



STM32G4 timer instance features

Feature	TIM1	TIM8	TIM20	TIM2	TIM3	TIM4	TIM5	TIM6	TIM7	TIM15	TIM16	TIM17
	(Advanced Control)			(General-Purpose)				(Basic)		(General-Purpose)		
Input capture mode	✓			✓				-		✓		
PWM input mode	✓			✓				-		✓	-	
Forced output mode	✓			✓				-		✓		
Output compare mode	✓			✓				-		✓		
PWM	Standard Asymmetric Combined Combined 3-phase 6-step PWM			Standard Asymmetric Combined				-		Standard Asymmetric Combined	Standard	
PWM dithering mode	✓			✓						✓	✓	
Programmable dead-time	✓ (CH1-4)			-				-		✓ (CH1)	-	
Break inputs	2 bidirectional			0				0		1 bidirectional		
One-Pulse Mode	✓			✓				-		✓		
Retriggerable one pulse mode	✓			✓				-		✓	-	

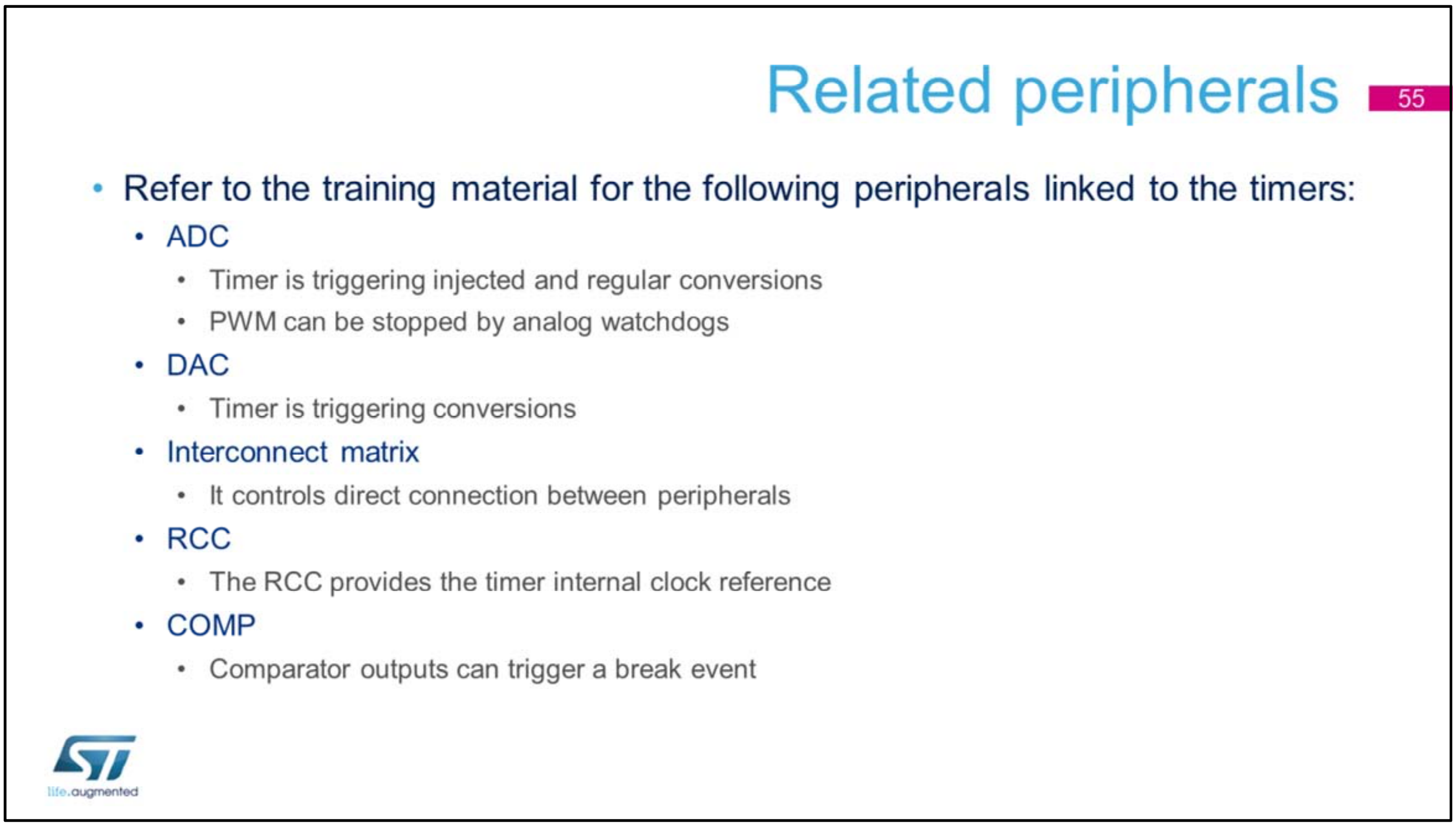
Input capture and output compare is not supported by Timers 6 and 7.
Any timer capable of PWM supports the dithering mode.
Programmable dead-time makes sense when the timer has complementary outputs, which is the case for Timer 1, 8, 20 (channels 1 to 4) and Timer 15 (channel 1).
Timers 1, 8, 20, 15, 16, and 17 support a break input.
Retriggerable one pulse mode is not supported by Timers 16, and 17.



STM32G4 timer instance features

Feature	TIM1	TIM8	TIM20	TIM2	TIM3	TIM4	TIM5	TIM6	TIM7	TIM15	TIM16	TIM17
	(Advanced Control)			(General-Purpose)				(Basic)		(General-Purpose)		
Encoder interface mode	Quadrature Clock plus direction Directional clock			Quadrature Clock plus direction Directional clock				-		-		
Index input	✓			✓				-		-		
Encoder error management	✓			✓				-		-		
Timer input XOR function	✓			-				-		✓	-	
DMA	✓			✓				✓		✓		

Only Timers 1, 8, 20 and 2, 3, 4, 5 have the 3-mode encoder interface: quadrature, clock plus direction and directional clock.
Any timer having this encoder interface supports the index pulse and detects encoder errors.
XOR function that combines channels is only available in Timers 1, 8, 20 and 15.
All timers can request DMA transfers through the DMAMUX unit, including basic timers.



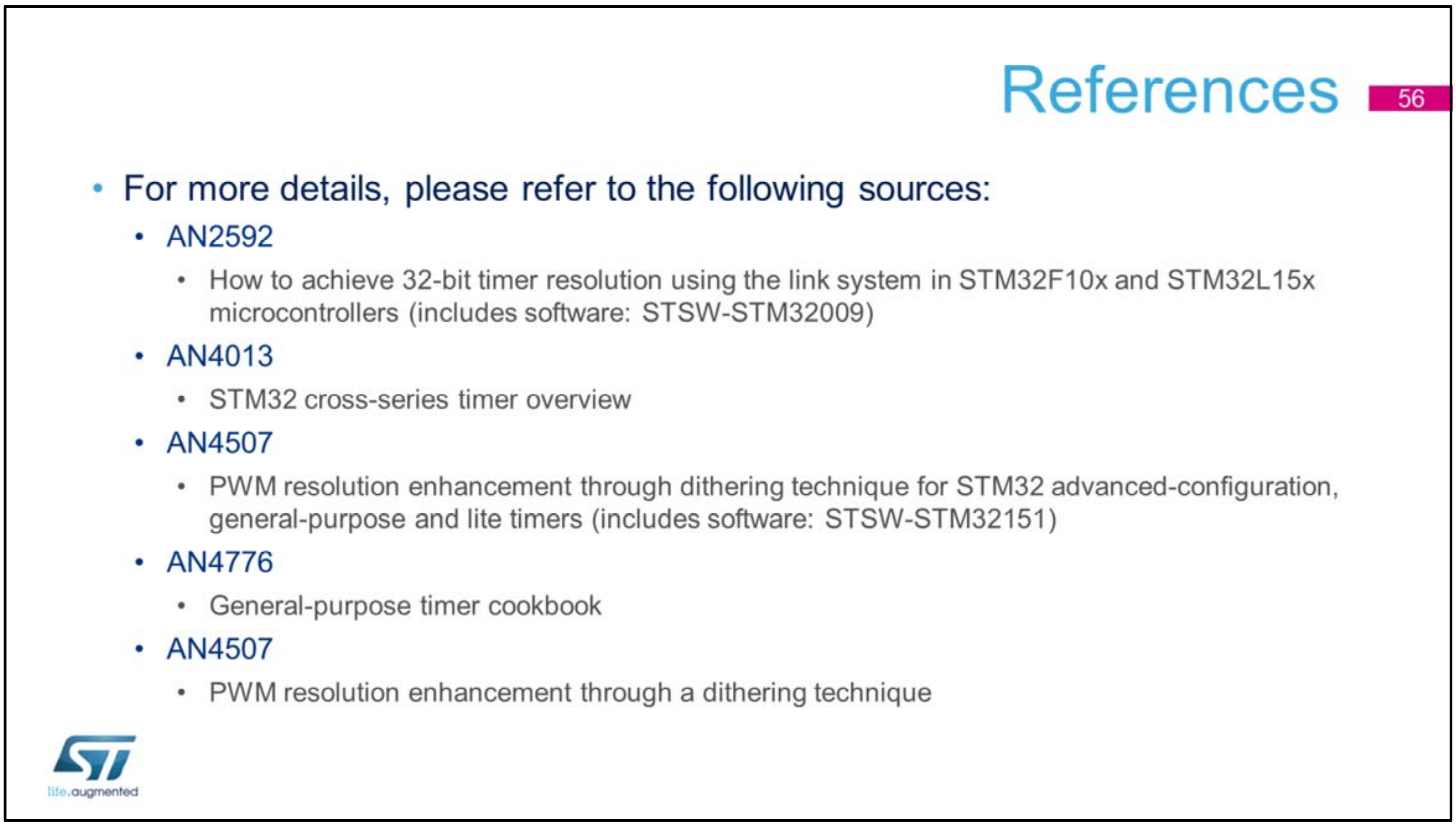
Related peripherals

55

- 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
  - Interconnect matrix
    - It controls direct connection between peripherals
  - RCC
    - The RCC provides the timer internal clock reference
  - COMP
    - Comparator outputs can trigger a break event

life.augmented

The timer is linked with multiple on-chip peripherals.
It serves as a trigger source for the ADC and the DAC converter.
The interconnect matrix enables timer to timer direct connection, using the master and slave interfaces, and also direct connection between timers and other peripherals.
The Reset and Clock Control unit, called RCC, provides the internal clock reference for all timers.
At last, the comparator units can detect abnormal temperature or voltage conditions and cause a timer break event.



References

- For more details, please refer to the following sources:
  - AN2592
    - How to achieve 32-bit timer resolution using the link system in STM32F10x and STM32L15x microcontrollers (includes software: STSW-STM32009)
  - AN4013
    - STM32 cross-series timer overview
  - AN4507
    - PWM resolution enhancement through dithering technique for STM32 advanced-configuration, general-purpose and lite timers (includes software: STSW-STM32151)
  - AN4776
    - General-purpose timer cookbook
  - AN4507
    - PWM resolution enhancement through a dithering technique

Three application notes complement the timer sections in the reference manual:
AN2592 gives a practical implementation of a 32-bit timer made of two synchronized 16-bit timers, and is useful for better understanding the overall timer synchronization mechanism. It comes with a software example.
AN4013 provides a more detailed overview of all timer features and available firmware examples.
AN4507 presents an implementation of PWM resolution enhancement by means of dithering techniques. It comes with a software example.
AN4776 starts with few reminders on timer operating principles and contains a collection of examples for standard timer use cases. It comes with a software example.
AN4507 presents a dithering technique that enhances

the PWM resolution. It comes with a software example.