Agenda

- 1st day – Afternoon
  - MC Application
    - Interface
    - Tuning
    - Tasks
    - Classes interaction
      - Current regulation
      - Ramp-up
      - Encoder alignment
  - Speed sensors updates:
    - Sensorless algorithm improvement
  - How to create User Project Interacting with MC Application
  - Dual motor control
    - Resources sharing
    - Supported configurations
    - Code size efficiency
  - Current reading sensor update

Dual motor control requirements

- In present implementation of dual FOC, two advanced timers are required (this limit the feasibility to big piranha)
- Consider that
  - CPU is shared between the two controls
  - FOC execution (i.e. CPU utilization) related to each of the drives must be synchronous with the related PWM signal
  - Depending on current sensing HW topology, ADC peripherals could also require to be shared (e.g. 2x three shunts)
- This limit the possible configurations and force to take some design decisions:
  - The two drive must have PWM frequencies multiple one of the other
  - The two drive may have different repetition counters
  - The two timers must be synchronized so that the update events of the timers do not overlap (related to present implementation)

ADC peripherals usage

- When possible ADCs peripherals have been dedicated to a given drive so that - in its turn - a drive can have dedicated ADC(s)
- When not possible a mechanism for sharing ADCs between the drive must be put in place
- Three shunts and ICS current reading HW topology requires two ADCs (so as to perform simultaneous sampling of two currents)
- Single shunt drive requires only one ADC

Motor 1 → Motor 2 ↓	**Three shunts ICS**	**Single shunt**
Three shunts ICS	ADC1: Three shunt/ICS* ADC2: Three shunt/ICS* ADC3:	ADC1: Three shunt/ICS ADC2: Three shunt/ICS ADC3: Single shunt
Single shunt	ADC1: Three shunt/ICS ADC2: Three shunt/ICS ADC3: Single shunt	ADC1: Single shunt ADC2: ADC3: Single shunt

* Shared resources

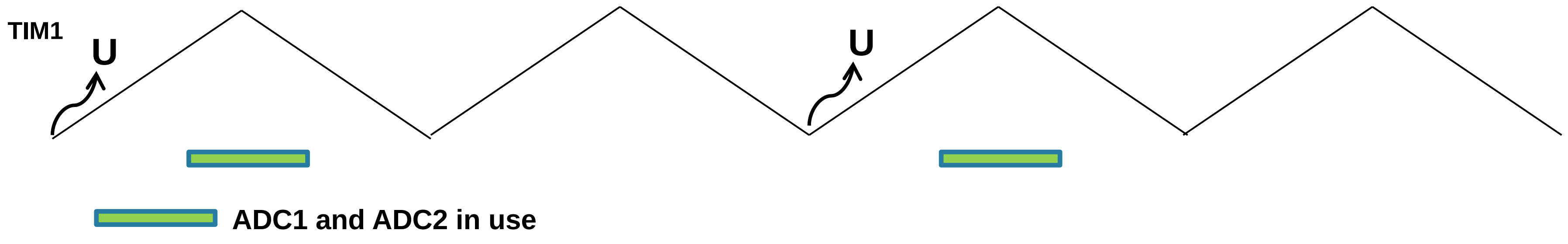
ADC peripherals sharing: basics

- If at least one drive is in single shunt topology no resources sharing is required
  - No limitations occur in this case

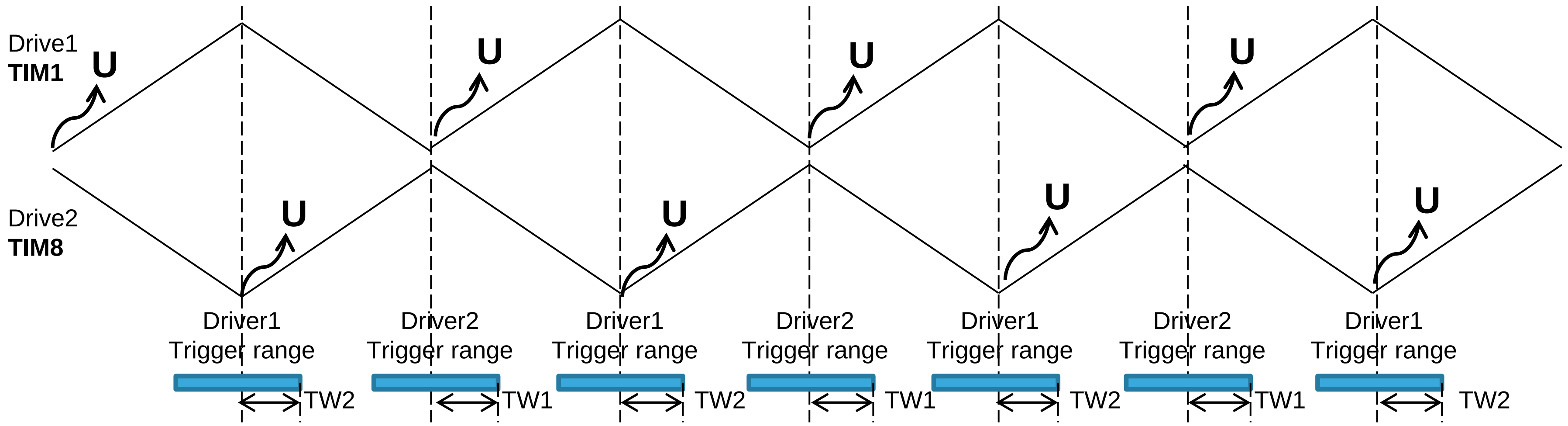
- In case of dual three-shunts, ICS or any combination of them
  - Drives must not require ADCs usage at the same time
    - PWM patterns should be properly shifted each other
  - There must be a moment where ADCs are re-configured so as to serve a given drive
    - For instance the set of channels to be converted and the triggering event must be re-configured
  - This moment will be referred as ‘context switching’

ADC peripheral sharing

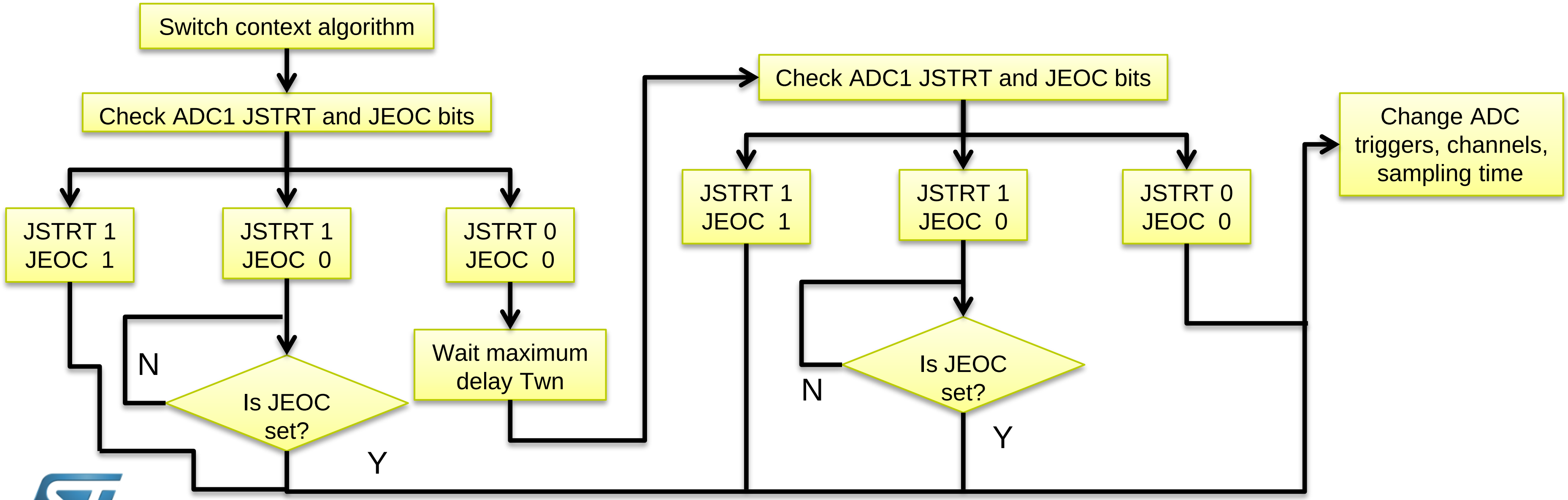
- When context switching should be performed?
- 3 shunts and ICS configurations require ADC conversions to be executed synchronously with PWM
- More precisely conversions must be performed:
  - With the same occurrence rate of the Update events (i.e. once each 1, 2, 3, … PWM periods)
  - In case of 3 shunts, conversions must be performed in proximity of the counter crest (where the three low side switches are closed and the related currents accessible)
  - For simplicity, in case of ICS as well current conversions are executed on counter crest
- In present implementation it has been decided to perform context switching in the TIMx Update ISR leading by half PWM period the ADC usage



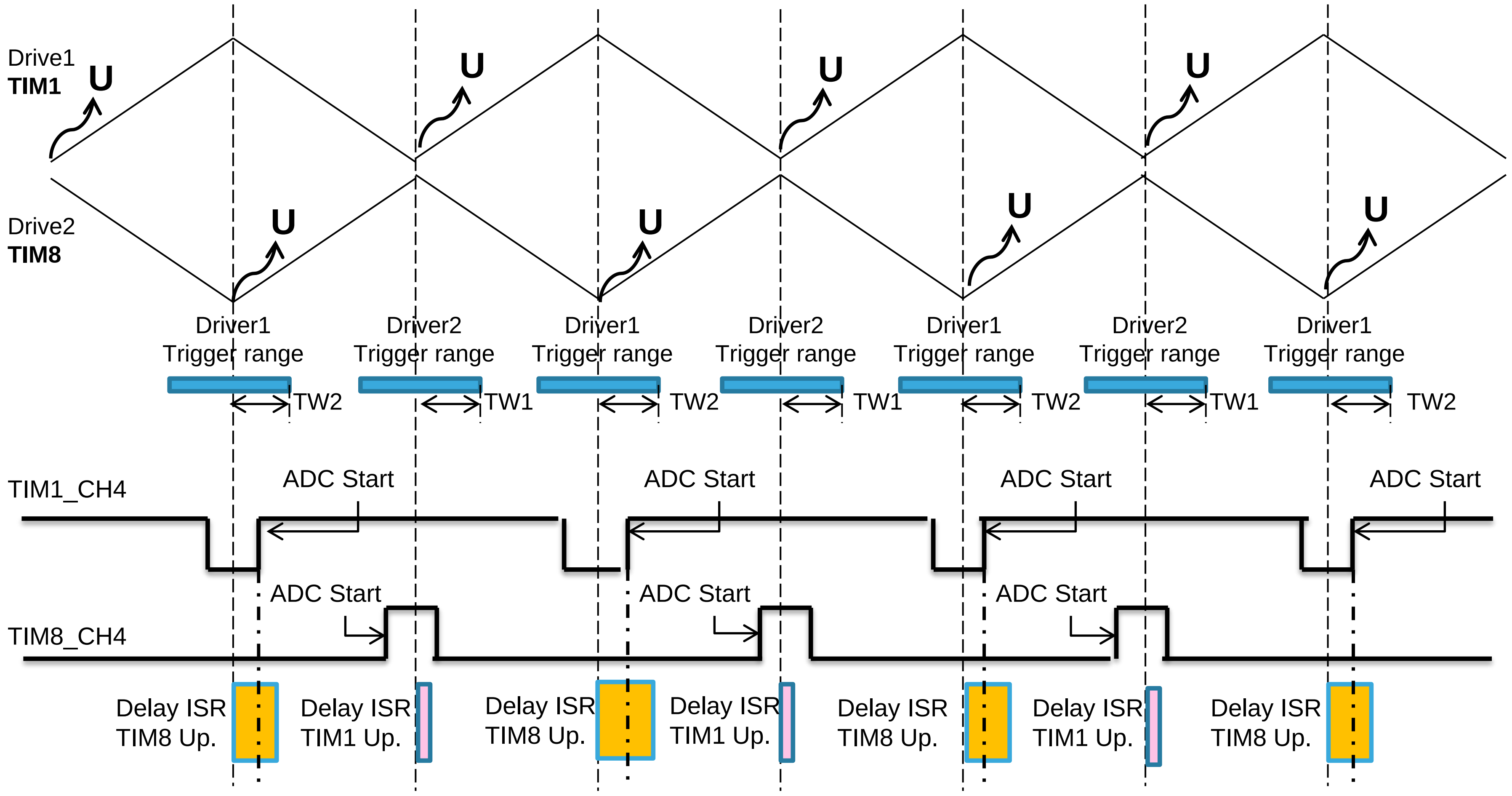
TIMx Update ISR: switch context algorithm



Twn is the maximum delay between counter crest and latest conversion in drive n



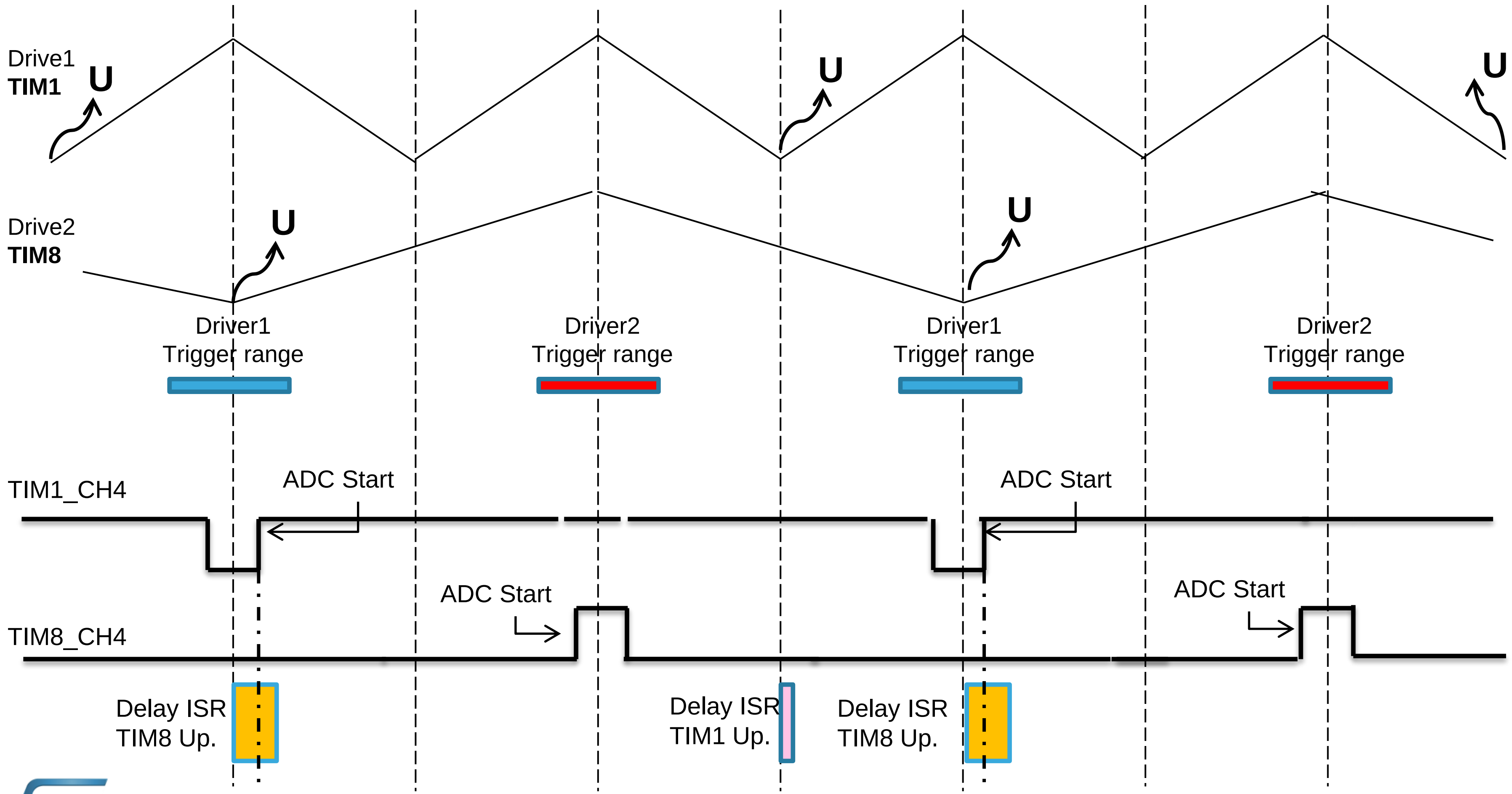
Example 1, drives have same f_{PWM}



- During TIMx Update event ISR, it’s necessary to wait for ADC to finish conversions before switching the context

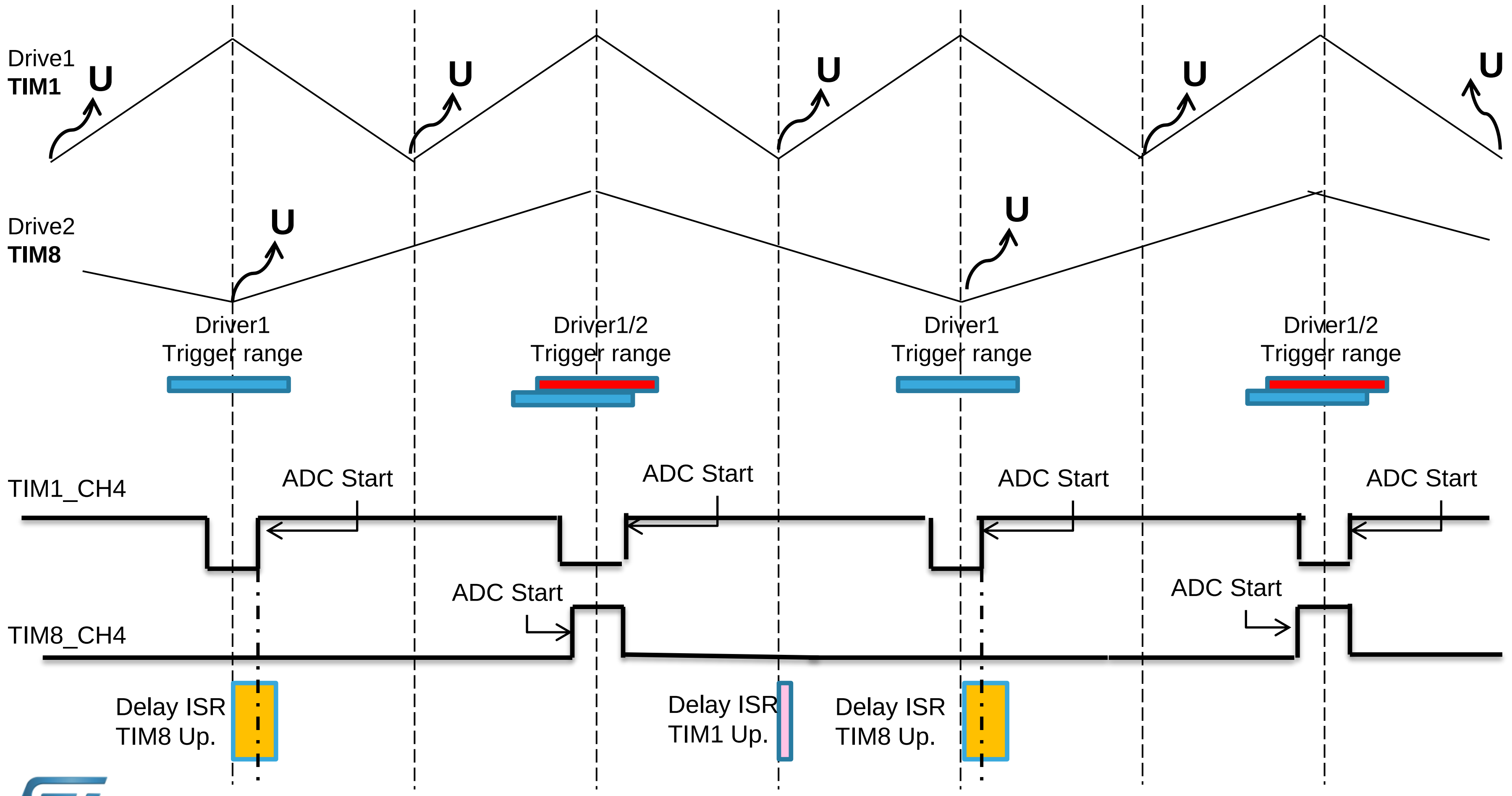
Example 2

- $f_{PWM1} = 2*f_{PWM2}$
- For drive 1 f_{PWM1} / fFOC1 = 2 (FOC executed each 2 PWM cycles)
- For drive 2 f_{PWM2} / fFOC2 = 1 (FOC executed each PWM cycle)



Example 3 (not supported)

- $f_{PWM1} = 2*f_{PWM2}$
- For drive 1 f_{PWM1} / fFOC1 = 1 (FOC executed each PWM cycles)
- For drive 2 f_{PWM2} / fFOC2 = 1 (FOC executed each PWM cycle)



Dual motor control feasibility summary

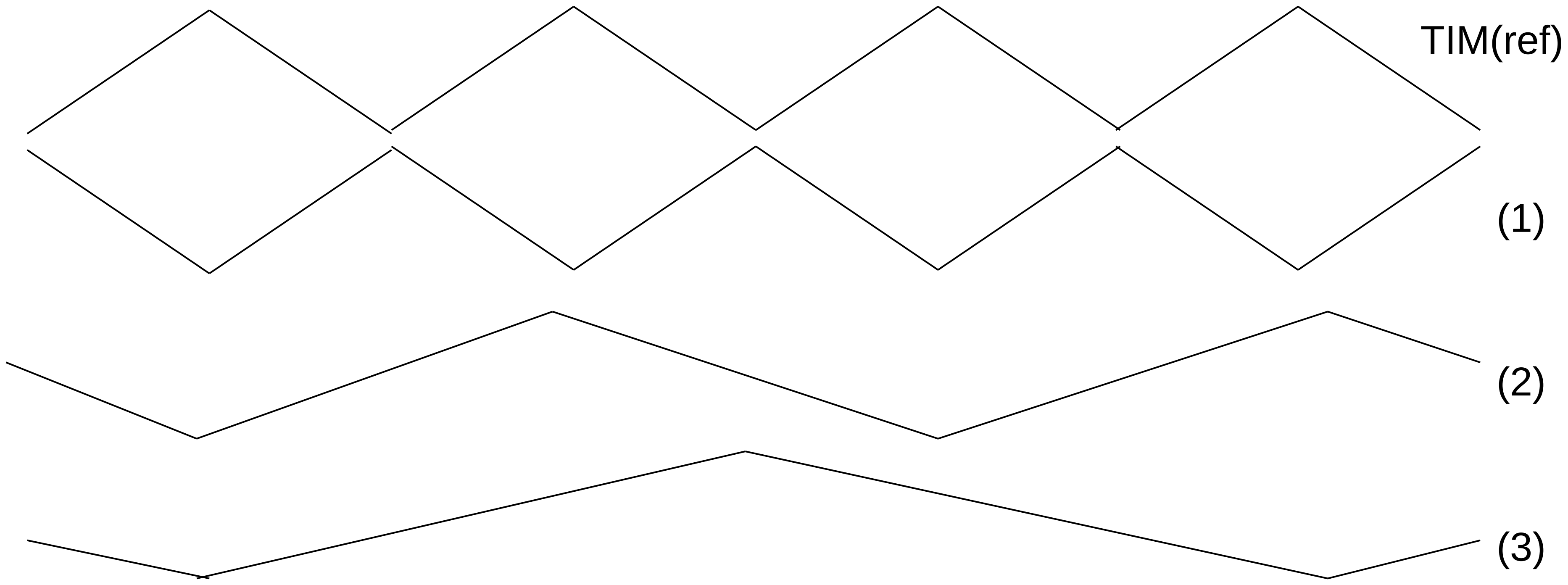
- PWM frequencies must be multiple (x1, x2 or x3)
- If at least one of the two drives is in 1 shunt topology

  →**no limitations**
- If no 1 shunt current reading topologies are present:

fPWMn / fPWMm ↓ fPWMn /fFOCn (Drive1; Drive 2) →	(1:1)	(1:2)	(1:3)	(2:1)	(2:2)	(2:3)	(3:1)	(3:2)	(3:3)
1	✓	✓	✓	✓	✓	✓	✓	✓	✓
2	✗	✗	✗	✓	✓	✓	✗	✗	✗
3	✗	✗	✗	✗	✗	✗	✓	✓	✓

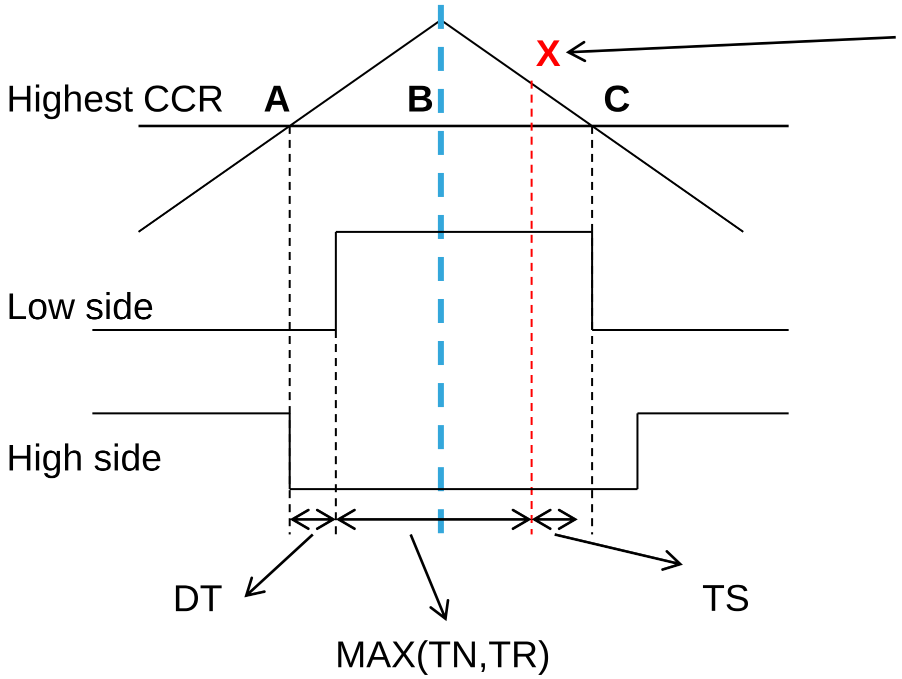
Dual motor control Timers synchronization

- Below graphics show synchronization of timers in case the have same frequency (1), in case one frequency is half the other one (2) and in case one frequency is one third of the other one (3):



Twn computation

- In case of ICS, Twn = 0
- In case of single shunt no need to define Twn (no ADC sharing)
- In case of three shunt is equal to BX in below figure



$$AC_{Min} = DT + MAX(TN,TR) + TS$$
$$AB = BC = AC/2$$
$$BX_{Max} = BC – TS = AC/2 – TS$$
$$BX_{Max} = (DT + MAX(TN,TR) - TS)/2$$

Dual motor control code size efficiency

- With the exception of PWMnCurrentFdbk sub-classes, dual MC utilizes exactly the same classes of single motor

- This makes very efficient the OOP architecture resulting in high code efficiency
  - Everything is shared between the two controls (e.g. sensors base classes, FOC base class, …) is linked only once
  - Example 1:
    - First motor: 3 shunt, sensor-less + PLL, no flux weakening, no MTPA, no feed-Forward
    - Second motor: 3 shunt, sensor-less + PLL, no flux weakening, no MTPA, no feed-forward
    - **Code size is 16.4kB, only +19% vs single motor case**
    - **RAM occupation is 3kB, only +50% vs single motor case**
  - **Example 2:**
    - First motor: 3 shunt, sensor-less + PLL, no flux weakening, no MTPA, no feed-forward
    - Second motor: 1 shunt, sensor-less PLL with MTPA, flux weakening and feed forward
    - **Code size is 21.7kB,**
    - **RAM occupation is 3.2kB RAM**

life.augmented

stm32f10x_MC_it.c editing for dual MC

- Contains all and only the interrupts utilized by motor control
  - ADC1_2, ADC3
  - DMA1, DMA2
  - TIM1_UP, TIM8_UP,
  - TIM2 …. TIM5
- This file may need to be updated (only in case of dual MC) according to the utilized configuration
  - In case of single motor HD, use TIM2 for real speed sensor handling
- Supported by ST MC Workbench: user do not need to manually

life.augmented