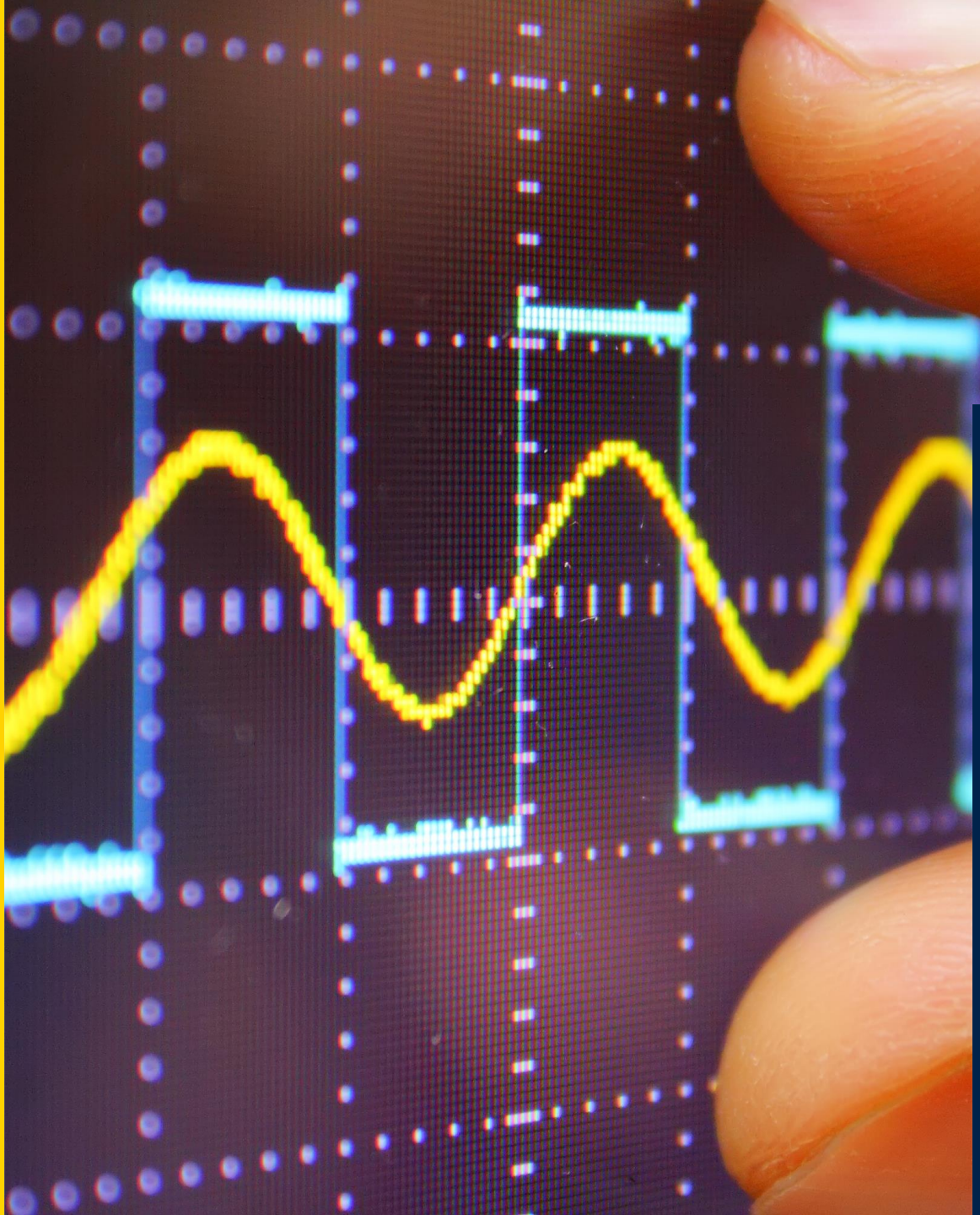


life.augmented

Complementary Variable Frequency PWM

Video Series: Hands-On with STM32 Timers

Saeed Molaie

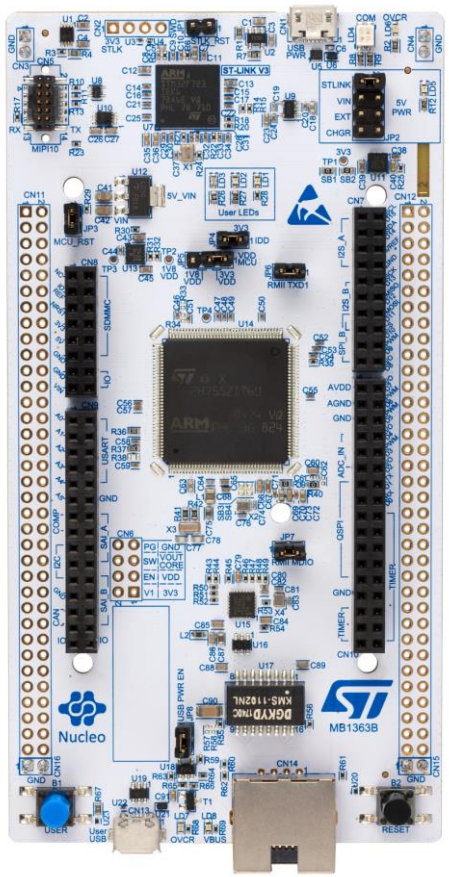
Objective

- Generate center aligned variable frequency PWM signal in run-time for low noise, low power dissipation, smooth motor control applications
- Signal output on multiple complementary channels (CH1, CH1N, CH2N, CH2N, CH3, CH3N)

Link to video series materials

- Link to a zip file with these slides, code, STM32CubeMX project file, and other materials is provided in the description

Equipment utilized



NUCLEO-H745ZI-Q

Micro USB Cable

Oscilloscope

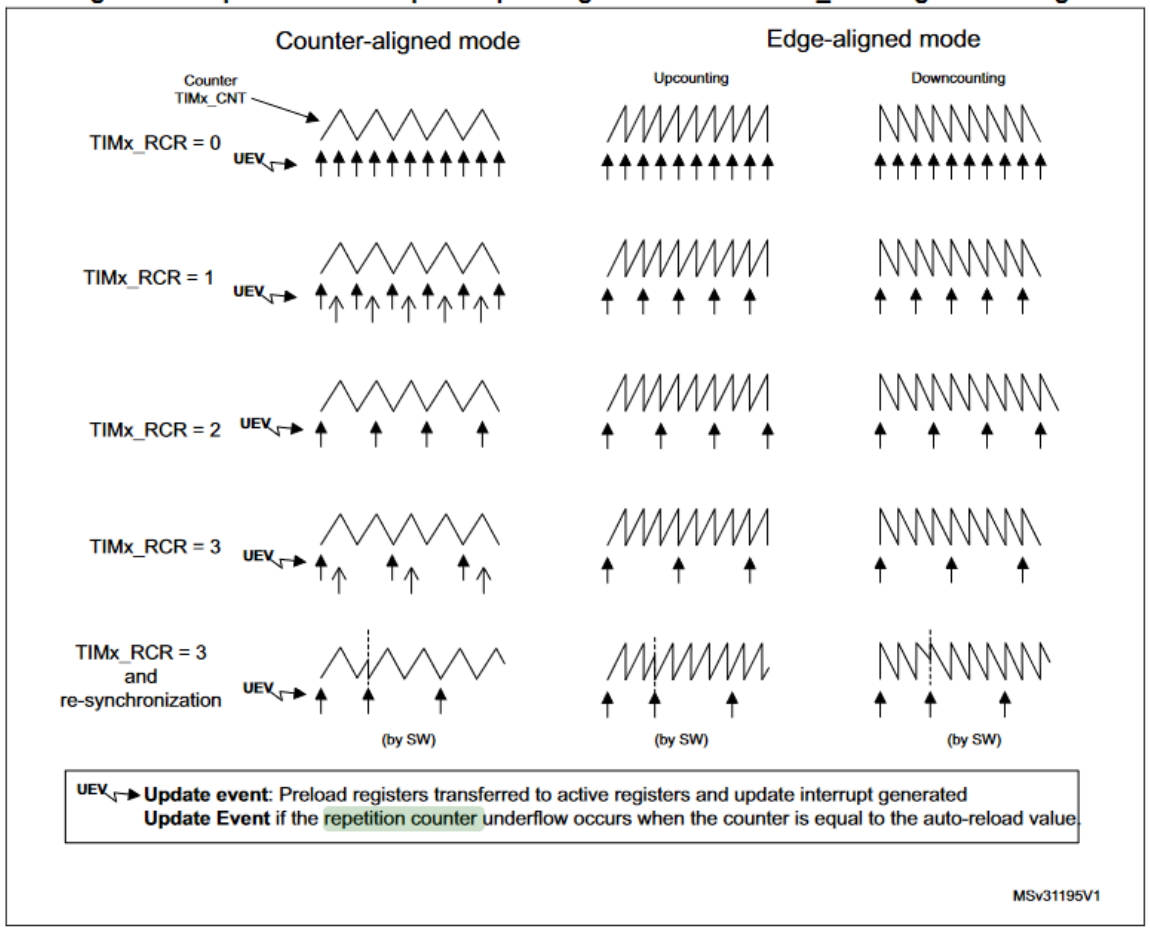
life.augmented

Low level setup of runtime frequency switching

- Smooth frequency change requires separate preload registers for TIMx_ARR & TIMx_CCR
- Preload register must be enabled by setting the
  - CCR: OCxPE bit in the TIMx_CCMRx
  - ARR: ARPE bit in the TIMx_CR1
- Asynchronous update of ARR: Preload register acts as a buffer between the ARR write/read and ARR
- Preload register contents are transferred to shadow registers when:
  - Update event occurs
  - Before counter starts

Low level setup of runtime frequency switching

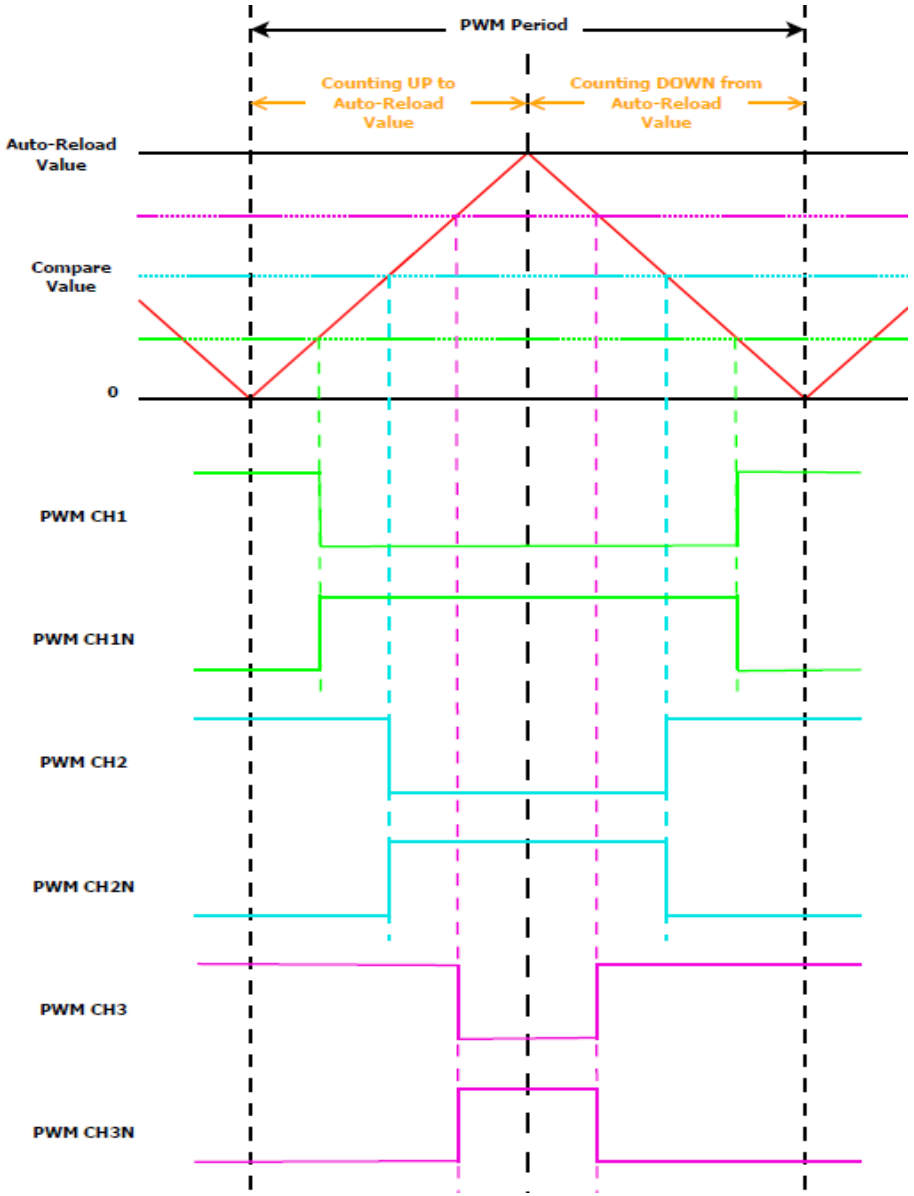
Figure 406. Update rate examples depending on mode and TIMx_RCR register settings



- In center aligned mode: if RCR value odd then UEV occurs on either overflow or underflow
- if the RCR was written before launching the counter, the UEV occurs on the underflow
- If the RCR was written after launching the counter, the UEV occurs on the overflow
- Table in RM0399

life.augmented

Hands-on: Complementary variable frequency PWM



- Output 3 different duty cycles
  - CH1, CH1N – 25%
  - CH2, CH2N – 50%
  - CH3, CH3N – 75%
- Period defined by auto reload register
- The frequency will vary from 4kHz, 8K, to 16K
- Identical frequency across all 3 channels
- Frequency varied during runtime
- Center aligned mode: up/down count
- Repetition counter (RCR) is 3 for this exercise

life.augmented

Table 5. Timer feature comparison

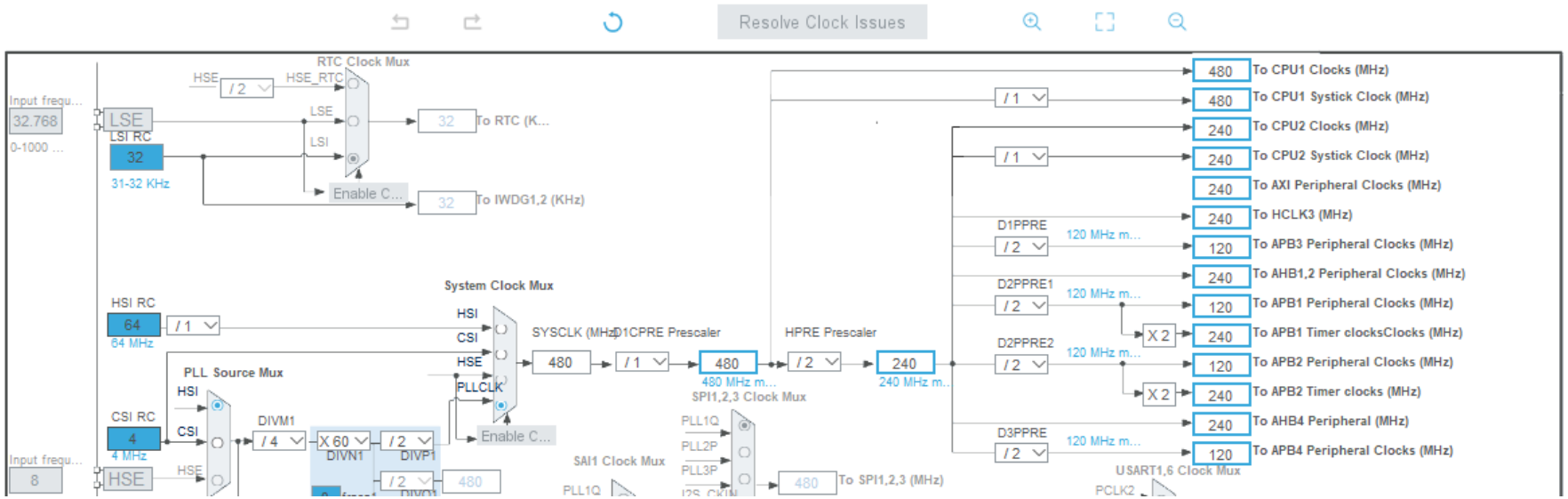
Timer type	Timer	Counter resolution	Counter type	Prescaler factor	DMA request generation	Capture/ compare channels	Comple-mentary output	Max interface clock (MHz)	Max timer clock (MHz) (1)
High-resolution timer	HRTIM1	16-bit	Up	/1 /2 /4 (x2 x4 x8 x16 x32, with DLL)	Yes	10	Yes	480	480
Advanced -control	TIM1, TIM8	16-bit	Up, Down, Up/down	Any integer between 1 and 65536	Yes	4	Yes	120	240
General purpose	TIM2, TIM5	32-bit	Up, Down, Up/down	Any integer between 1 and 65536	Yes	4	No	120	240
	TIM3, TIM4	16-bit	Up, Down, Up/down	Any integer between 1 and 65536	Yes	4	No	120	240
	TIM12	16-bit	Up	Any integer between 1 and 65536	No	2	No	120	240
	TIM13, TIM14	16-bit	Up	Any integer between 1 and 65536	No	1	No	120	240
	TIM15	16-bit	Up	Any integer between 1 and 65536	Yes	2	1	120	240
	TIM16, TIM17	16-bit	Up	Any integer between 1 and 65536	Yes	1	1	120	240

- Timer Feature Comparison Table in STM32H745 datasheet (DS12110)

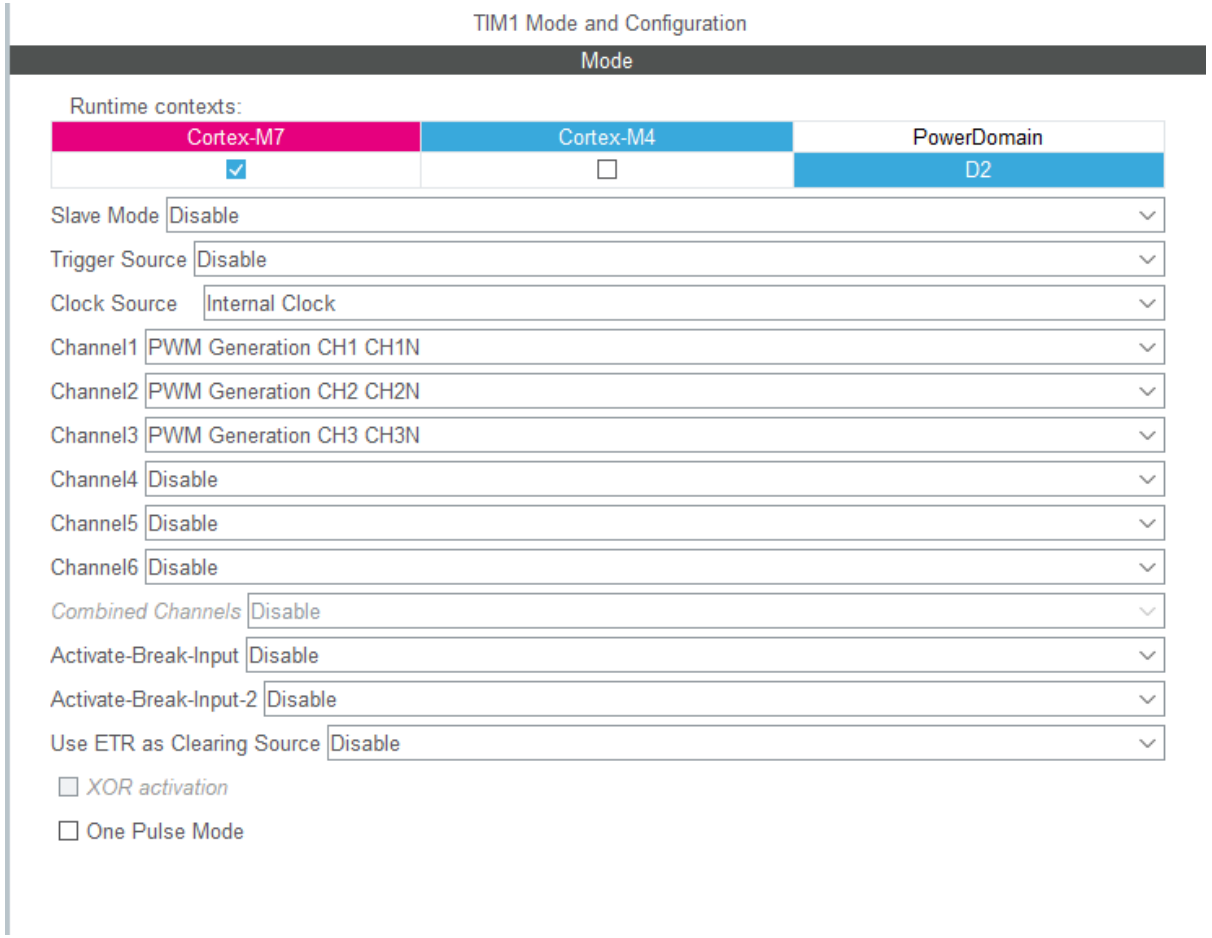
life.augmented

Configure Clock frequency (480MHz)

Pinout & Configuration | Clock Configuration | Project Manager | Tools



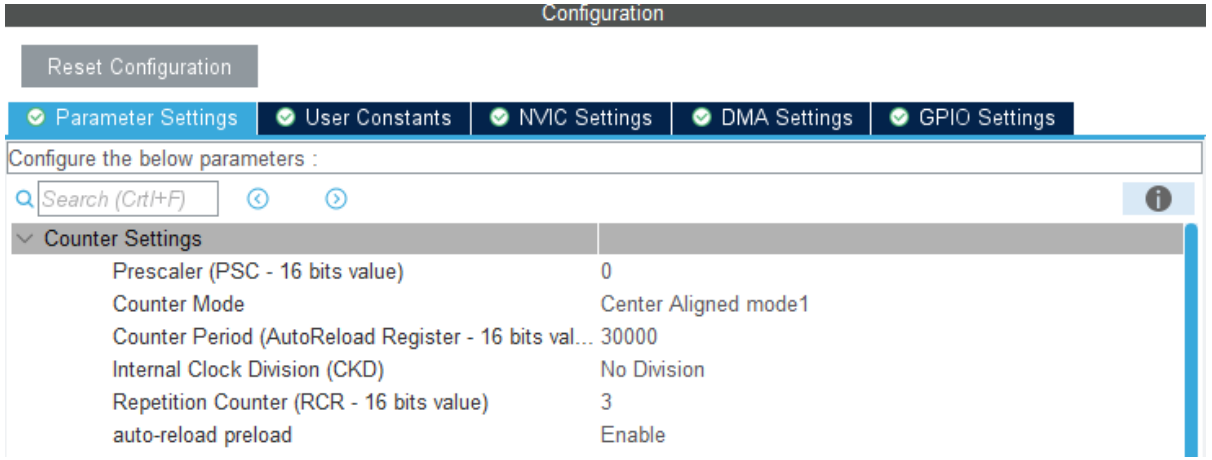
STM32CubeMX - 1



- Select Cortex-M7 & configure timer 1
- For this lab we will configure the project for multi-channel complementary PWM signal output

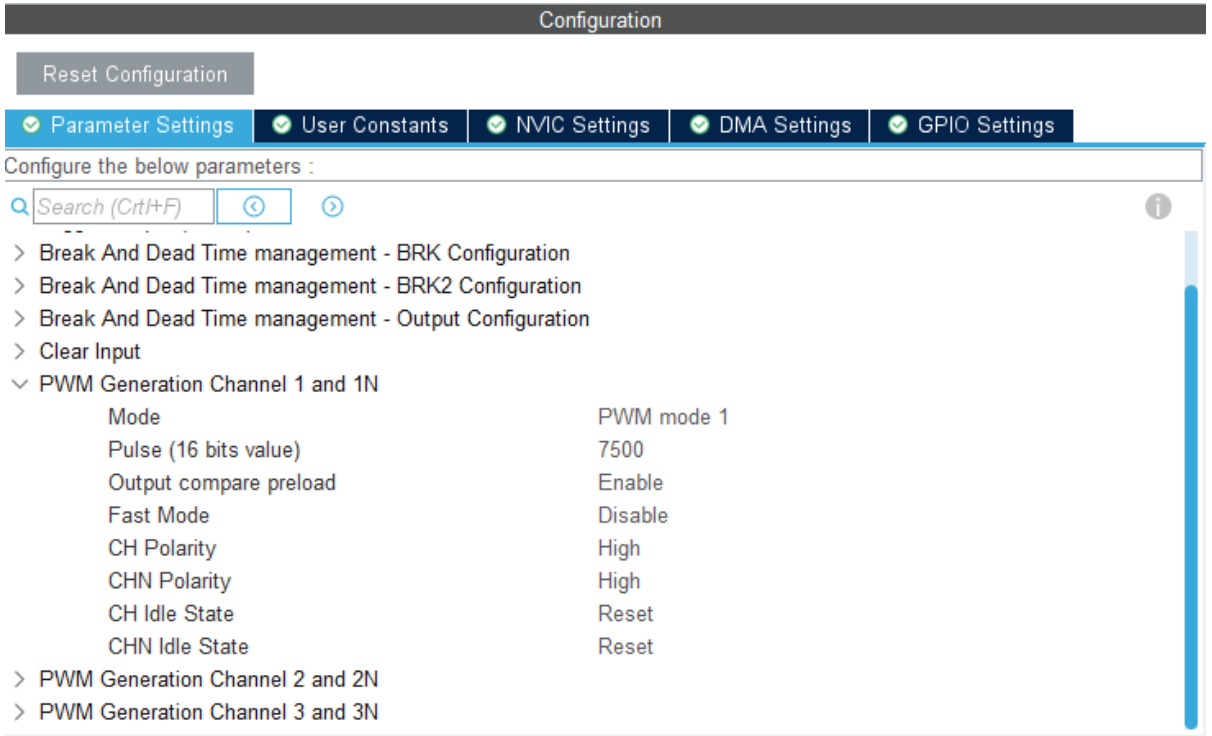
life.augmented

STM32CubeMX - 2



- The PWM frequency is configured for 4 kHz. The Counter Mode is set to Center Aligned mode 1. Counter Period (AutoReload Register) = 30,000. Auto-reload preload is Enable. Repetition counter is set to 3 to complete 2 full PWM time base periods.

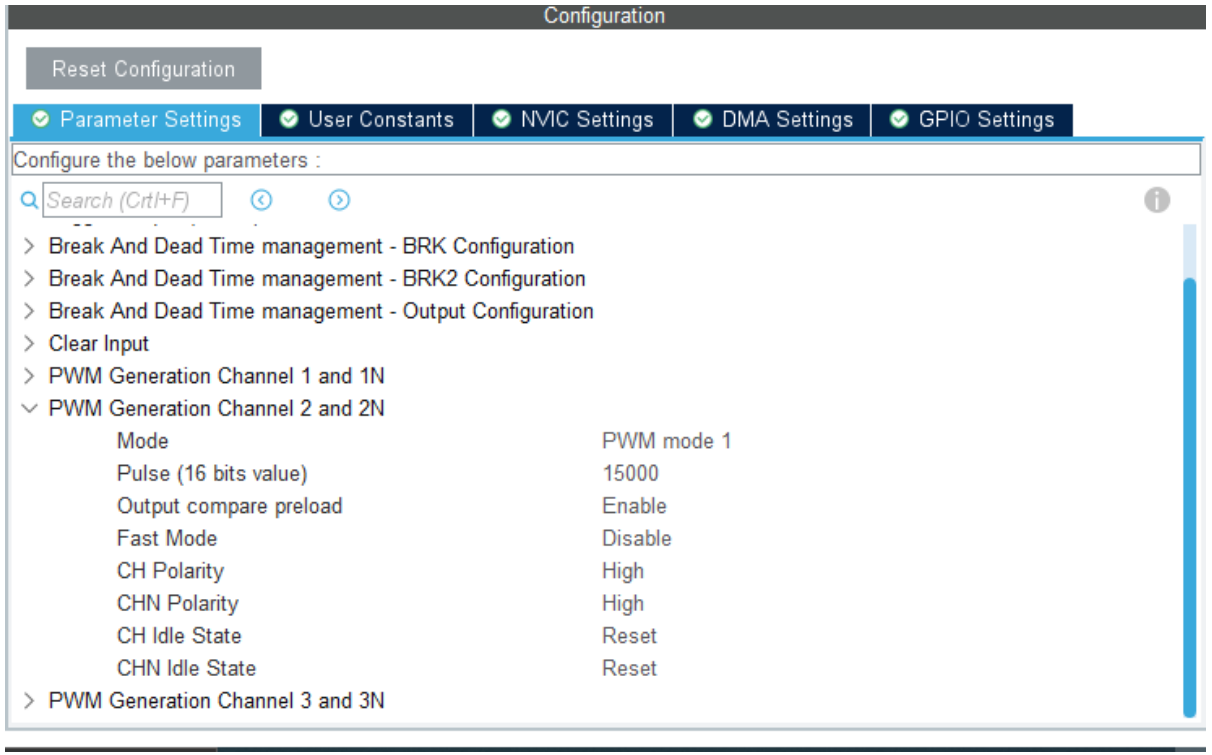
STM32CubeMX - 3



- The PWM Channel 1 duty cycle is configured for 25%. The Mode is set for PWM mode 1. Pulse (16 bits value) = 7500. Output compare preload is enabled.

life.augmented

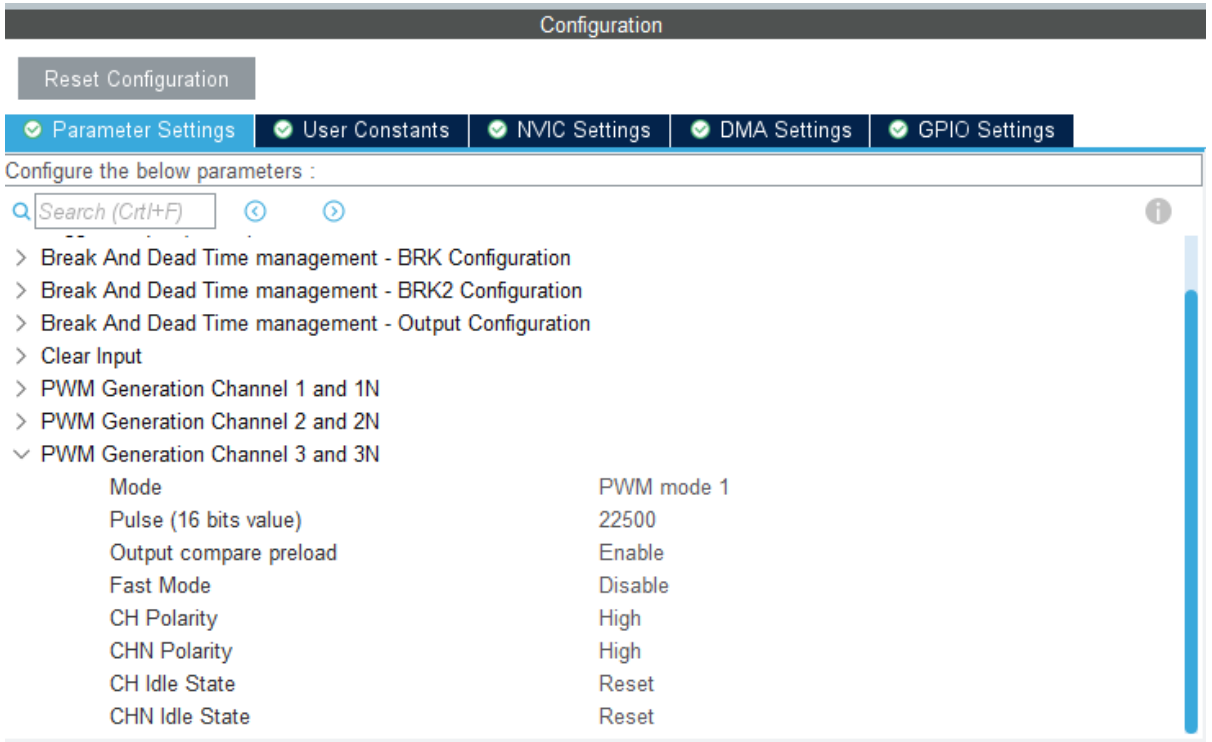
STM32CubeMX - 4



- The PWM Channel 2 duty cycle is configured for 50%. The Mode is set for PWM mode 1. Pulse (16 bits value) = 15000. Output compare preload is enabled.

life.augmented

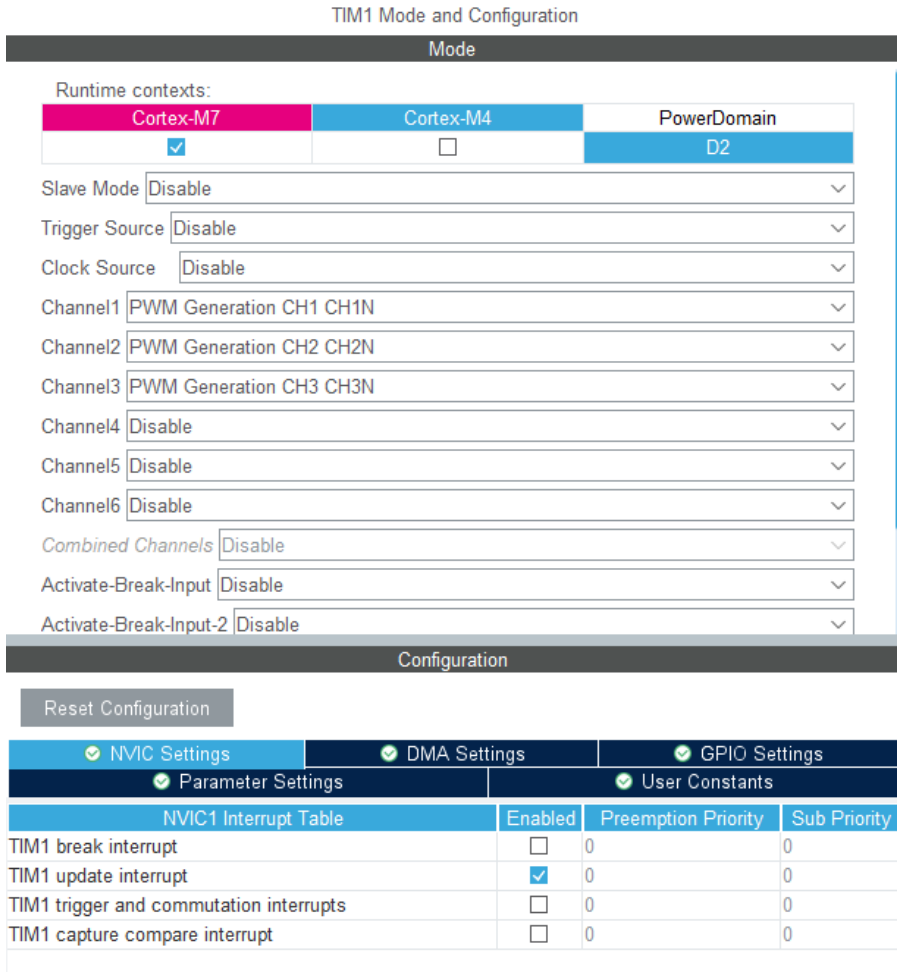
STM32CubeMX - 4



- The PWM Channel 3 duty cycle is configured for 75%. The Mode is set for PWM mode 1. Pulse (16 bits value) = 22500. Output compare preload is enabled.

life.augmented

NVIC Settings



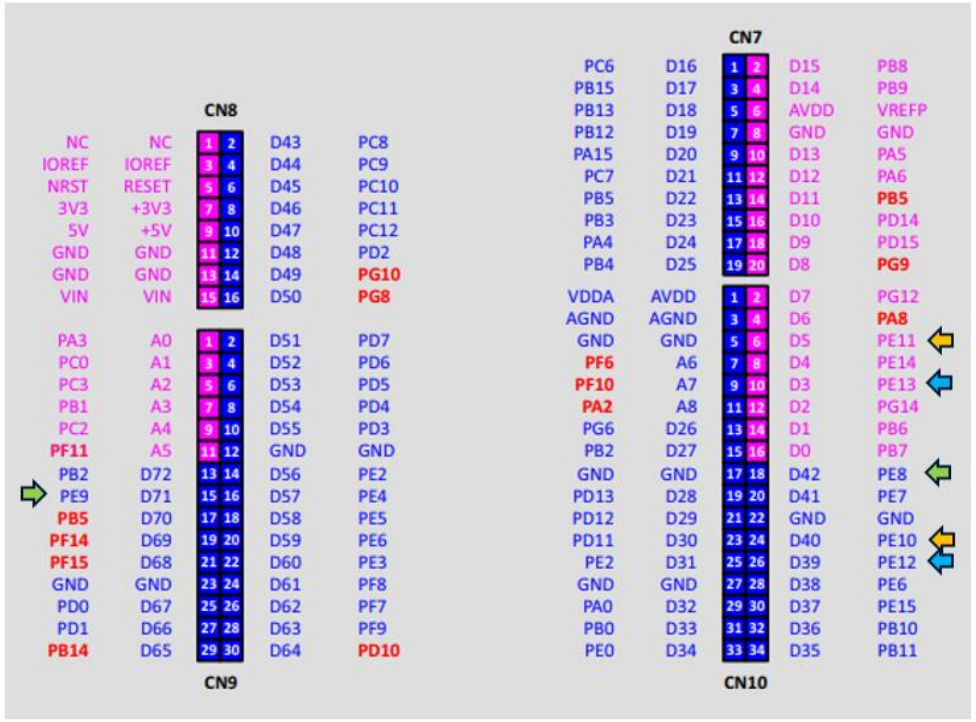
- Enable timer 1 interrupt to generate interrupt after each update event (UEV). When repetition counter reaches 0, we change the frequency in the timer callback function.

life.augmented

STM32CubeMX final pin configuration

GPIO Pin	PWM Channel
PE8	CH1N (N - negated)
PE9	CH1
PE10	CH2N
PE11	CH2
PE12	CH3N
PE13	CH3

Nucleo Pinout



CH1(PE9) & CH1N(PE8)

CH2(PE11) & CH2N(PE10)

CH3(PE13) & CH3N(PE12)

- Connect your scope to the Nucleo-H745ZI. The PWM regular and complementary outputs are shown on the marked pins below on the CN9 and CN10 headers.

Define duty cycle for each frequency

```
/* USER CODE END Header */
/* Includes ------------------------------------------------------------------*/
#include "main.h"

/* Private includes ----------------------------------------------------------*/
/* USER CODE BEGIN Includes */

/* USER CODE END Includes */

/* Private typedef -----------------------------------------------------------*/
/* USER CODE BEGIN PTD */
#define TIM1_PWM_FREQ_4K      30000
#define TIM1_PWM_FREQ_8K      15000
#define TIM1_PWM_FREQ_16K      7500

#define TIM1_PWM_4K_25DUTY      ( TIM1_PWM_FREQ_4K * 0.25 )
#define TIM1_PWM_4K_50DUTY      ( TIM1_PWM_FREQ_4K * 0.50 )
#define TIM1_PWM_4K_75DUTY      ( TIM1_PWM_FREQ_4K * 0.75 )

#define TIM1_PWM_8K_25DUTY      ( TIM1_PWM_FREQ_8K * 0.25 )
#define TIM1_PWM_8K_50DUTY      ( TIM1_PWM_FREQ_8K * 0.50 )
#define TIM1_PWM_8K_75DUTY      ( TIM1_PWM_FREQ_8K * 0.75 )

#define TIM1_PWM_16K_25DUTY      ( TIM1_PWM_FREQ_16K * 0.25 )
#define TIM1_PWM_16K_50DUTY      ( TIM1_PWM_FREQ_16K * 0.50 )
#define TIM1_PWM_16K_75DUTY      ( TIM1_PWM_FREQ_16K * 0.75 )
/* USER CODE END PTD */
```

- We define these macros for modifying the duty cycle, at run-time for each frequency (used in our timer 1 callback function)
- Frequency is varied across all channels to values: 4KHz, 8KHz, 16KHz
- Duty cycle is unique to each channel:
  - CH1/1N: 25%
  - CH2/2N: 50%
  - CH3/3N: 75%

life.augmented

Starting timer and complementary channels

```
130 /* USER CODE END Boot_Mode_Sequence_2 */
131
132   /* USER CODE BEGIN SysInit */
133
134   /* USER CODE END SysInit */
135
136   /* Initialize all configured peripherals */
137   MX_GPIO_Init();
138   MX_TIM1_Init();
139   /* USER CODE BEGIN 2 */
140   HAL_TIM_Base_Start_IT( &htim1 );
141   HAL_TIM_PWM_Start( &htim1, TIM_CHANNEL_1 );
142   HAL_TIM_PWM_Start( &htim1, TIM_CHANNEL_2 );
143   HAL_TIM_PWM_Start( &htim1, TIM_CHANNEL_3 );
144   HAL_TIMEx_PWMN_Start( &htim1, TIM_CHANNEL_1 );
145   HAL_TIMEx_PWMN_Start( &htim1, TIM_CHANNEL_2 );
146   HAL_TIMEx_PWMN_Start( &htim1, TIM_CHANNEL_3 );
147
148   /* USER CODE END 2 */
149
150   /* Infinite loop */
151   /* USER CODE BEGIN WHILE */
152   while (1)
153   {
154     /* USER CODE END WHILE */
155
156     /* USER CODE BEGIN 3 */
157   }
158   /* USER CODE END 3 */
159 }
160
```

- Using HAL API, we start regular timer and complementary channels for generating the signal

life.augmented

Interrupt handler/callback definition

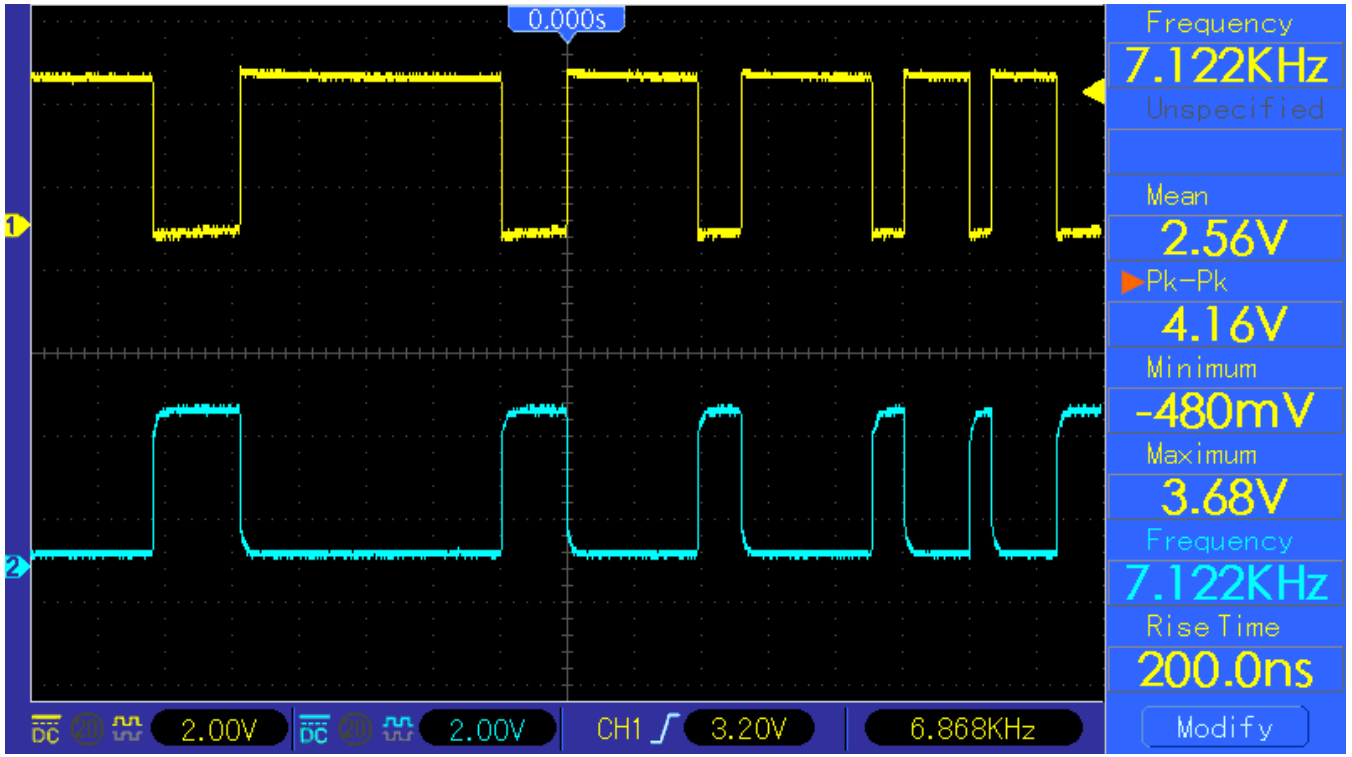
```
319
320 /* USER CODE BEGIN 4 */
321 void HAL_TIM_PeriodElapsedCallback(TIM_HandleTypeDef *htim)
322 {
323   if ( timer_ar_value == TIM1_PWM_FREQ_4K )
324   {
325     timer_ar_value = TIM1_PWM_FREQ_8K;
326     __HAL_TIM_SET_AUTORELOAD( &htim1, TIM1_PWM_FREQ_8K );
327     __HAL_TIM_SET_COMPARE( &htim1, TIM_CHANNEL_1, TIM1_PWM_8K_25DUTY );
328     __HAL_TIM_SET_COMPARE( &htim1, TIM_CHANNEL_2, TIM1_PWM_8K_50DUTY );
329     __HAL_TIM_SET_COMPARE( &htim1, TIM_CHANNEL_3, TIM1_PWM_8K_75DUTY );
330   }
331   else if ( timer_ar_value == TIM1_PWM_FREQ_8K )
332   {
333     timer_ar_value = TIM1_PWM_FREQ_16K;
334     __HAL_TIM_SET_AUTORELOAD( &htim1, TIM1_PWM_FREQ_16K );
335     __HAL_TIM_SET_COMPARE( &htim1, TIM_CHANNEL_1, TIM1_PWM_16K_25DUTY );
336     __HAL_TIM_SET_COMPARE( &htim1, TIM_CHANNEL_2, TIM1_PWM_16K_50DUTY );
337     __HAL_TIM_SET_COMPARE( &htim1, TIM_CHANNEL_3, TIM1_PWM_16K_75DUTY );
338   }
339   else if ( timer_ar_value == TIM1_PWM_FREQ_16K )
340   {
341     timer_ar_value = TIM1_PWM_FREQ_4K;
342     __HAL_TIM_SET_AUTORELOAD( &htim1, TIM1_PWM_FREQ_4K );
343     __HAL_TIM_SET_COMPARE( &htim1, TIM_CHANNEL_1, TIM1_PWM_4K_25DUTY );
344     __HAL_TIM_SET_COMPARE( &htim1, TIM_CHANNEL_2, TIM1_PWM_4K_50DUTY );
345     __HAL_TIM_SET_COMPARE( &htim1, TIM_CHANNEL_3, TIM1_PWM_4K_75DUTY );
346   }
347   else
348   {
349     timer_ar_value = TIM1_PWM_FREQ_4K;
350     __HAL_TIM_SET_AUTORELOAD( &htim1, TIM1_PWM_FREQ_4K );
351     __HAL_TIM_SET_COMPARE( &htim1, TIM_CHANNEL_1, TIM1_PWM_4K_25DUTY );
352     __HAL_TIM_SET_COMPARE( &htim1, TIM_CHANNEL_2, TIM1_PWM_4K_50DUTY );
353     __HAL_TIM_SET_COMPARE( &htim1, TIM_CHANNEL_3, TIM1_PWM_4K_75DUTY );
354   }
355 }
356 /* USER CODE END 4 */
```

- Change PWM frequency and duty cycle at runtime
- Set the TIM Autoreload Register value at runtime without calling another timer any Init function: *__HAL_TIM_SET_AUTORELOAD*
- Set the TIM Capture Compare Register value at runtime without calling another timer Config Channel function: *__HAL_TIM_SET_COMPARE*

Result: channel 1 and 1N scope capture

Yellow: CH1 (GPIO Pin PE8) – 25% duty

Cyan: CH1N (GPIO Pin PE9) – 25% duty



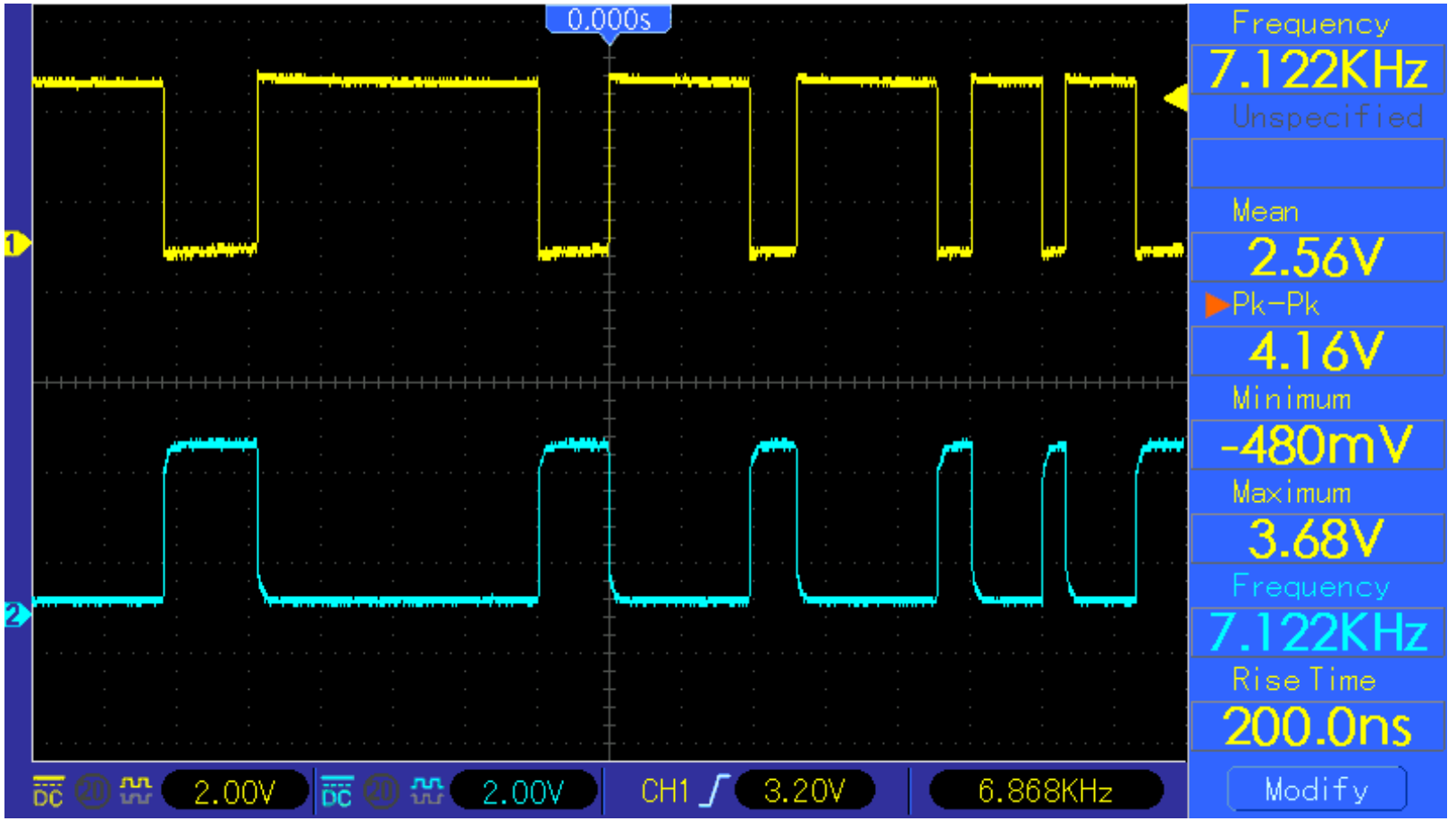
Center aligned (PWM Mode 1) :

$$Duty\ Cycle = \frac{CCR}{ARR}$$

Result: channel 2 and 2N scope capture

Yellow: CH1 (GPIO Pin PE10) – 50% duty

Cyan: CH1N (GPIO Pin PE11) – 50% duty

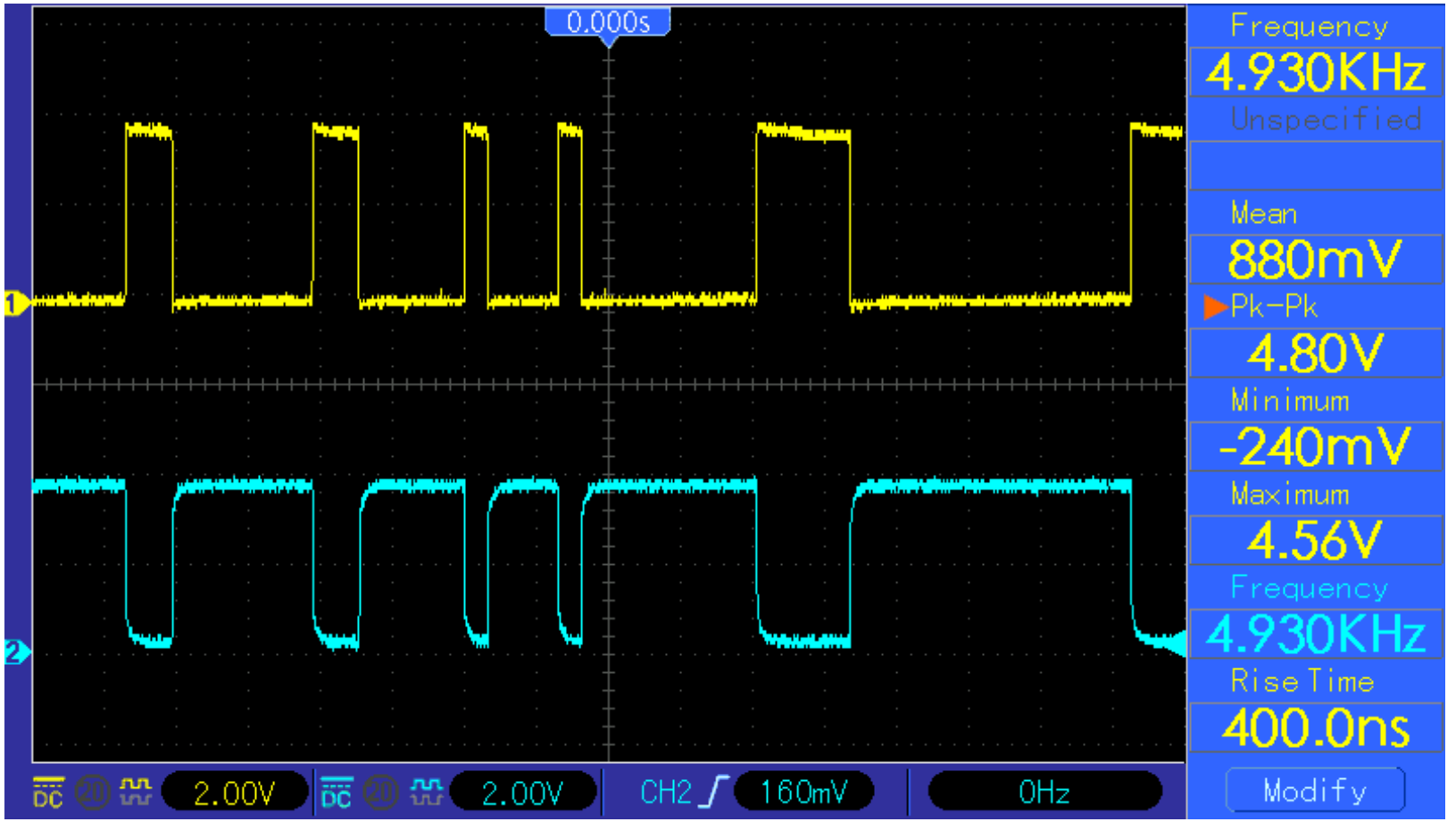


life.augmented

Result: channel 3 and 3N scope capture

Yellow: CH1 (GPIO Pin PE12) – 75% duty

Cyan: CH1N (GPIO Pin PE13) – 75% duty



life.augmented

Links

- AN4013: https://www.st.com/resource/en/application_note/dm00042534-stm32-crossseries-timer-overview-stmicroelectronics.pdf

- STM32H745 reference manual: https://www.st.com/resource/en/reference_manual/dm00176879-stm32h745755-and-stm32h747757-advanced-armbased-32bit-mcus-stmicroelectronics.pdf

- STM32H745 datasheet: https://www.st.com/resource/en/datasheet/stm32h745zg.pdf

Thank you

life.augmented