Software archteture EXPLANATION and theory about spark advance timing

Introduction

The problem to predict angular position

Hardware and Topology

Tips and Tricks

Ways of working

Capacitor Discharger Ignition, why I decide to create a DC CDI?

How to calibrate the spark advance timing curve?

System Validation

Show some measuraments and details about the system response (time to treat an interruption for example)

Maybe I need to create this document with some special topics:

Talk about the evolution of this systems (break pointer, fixed timing spark)

Present the physical problem about the importance to apply the correct spark advance timing during the engine operation

Talk about the differences between 2 stroke and four stroke engines (different curves, but the problem is the same for both)

Discuss about the problem related the engine instrumentation (measure the angular crankshaft displacement), because use a very limited flywheel to identify the angular position (2 different positions)

Explain the adopted strategy to predict the angular position

Discuss about the Kalman filter, Alpha beta gamma filter or another alternative and predict angular position with some accuracy

Present the hardware e discuss about the circuit and the problems about this kind of circuit

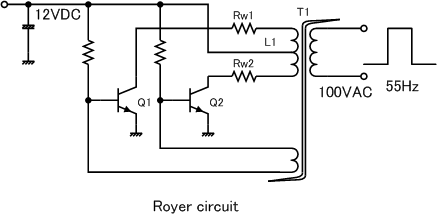
Explain the use of different diodes (Schottky, TVS, fast recovery)

The most important thing at this moment is throw up all knowledge that I gathered about this project and create a material to future consult

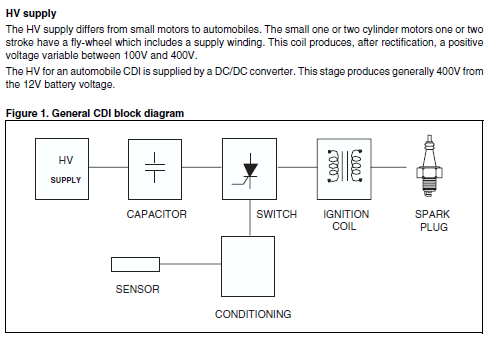
CDI or Capacitor Discharger Ignition is a circuit that charge energy in a capacitor and discharge through a transformer (Igntion Coil), consequently generates a spark that can be use to burn the misture fuel+air inside a combustion engine.

The relation between coils in this ignition coil (transform) generally is more than 1:100, due this reason they need aproximataly 200V to generate 20KV (generates a really good spark).

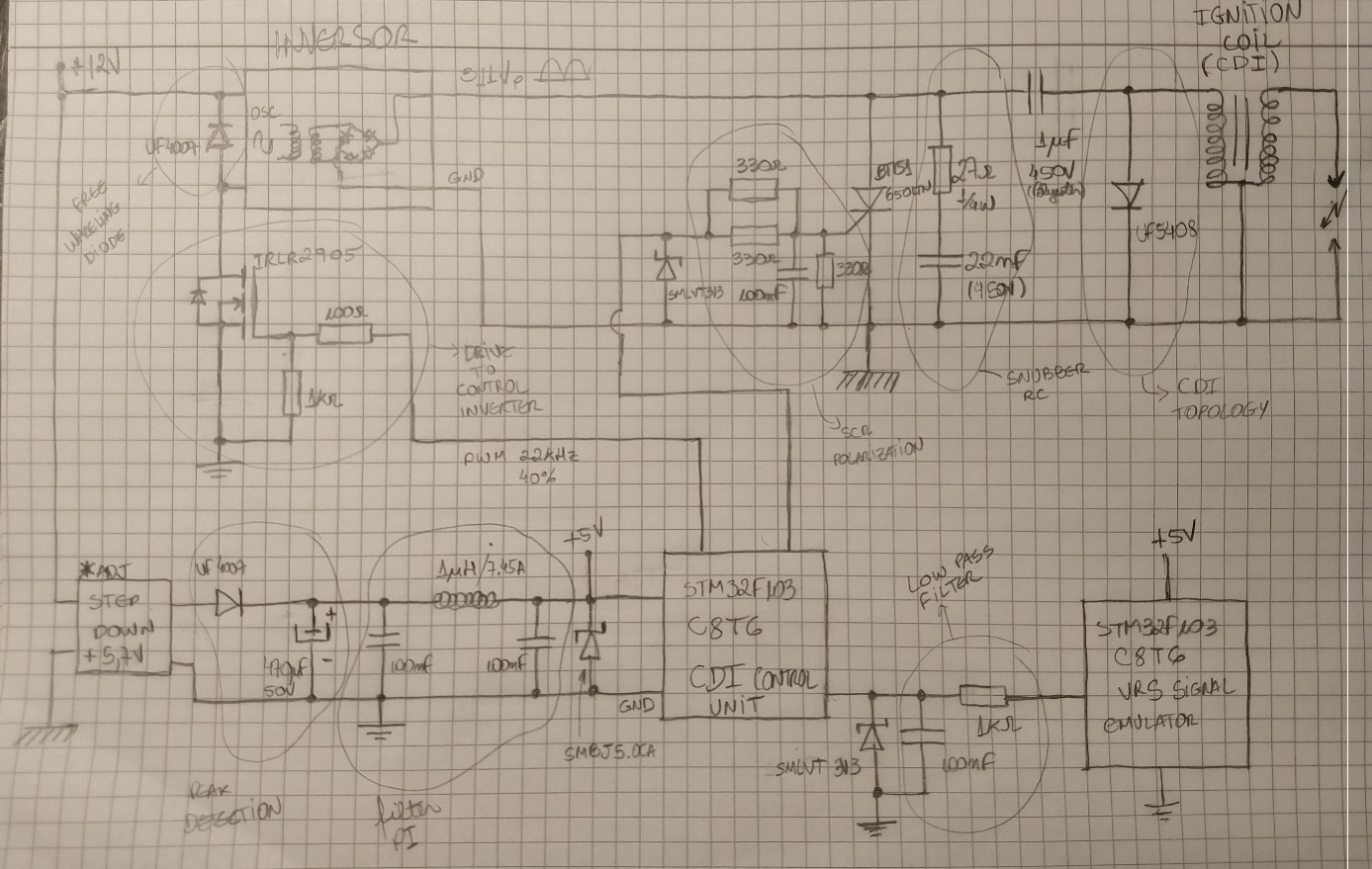
To implement the HV block, I decided to use a small inverter (this common Car Power Inverter that generates 220AC), this inverter was based in a Royer Oscillator.

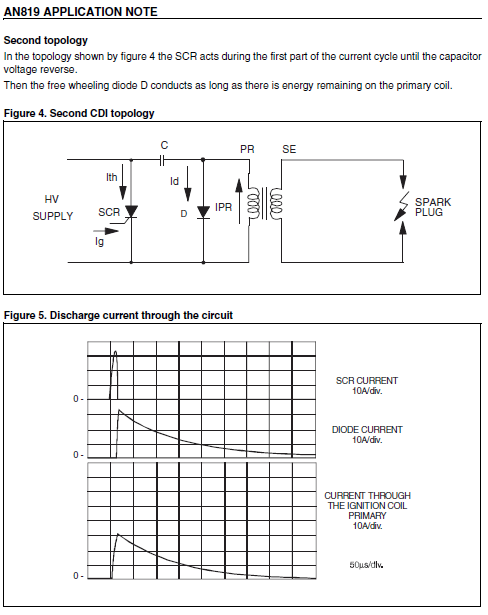


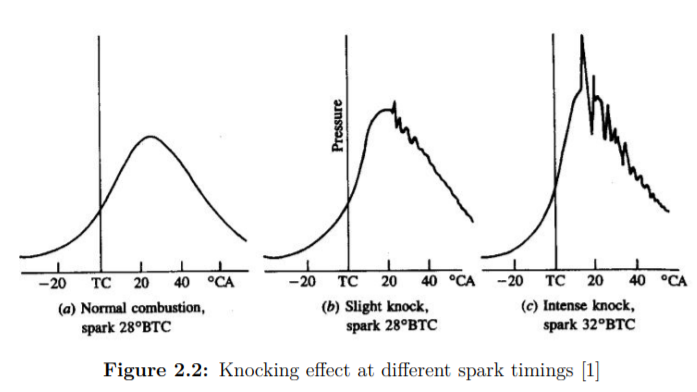
<https://upload.wikimedia.org/wikipedia/commons/e/e5/Royer_Circuit1.gif>



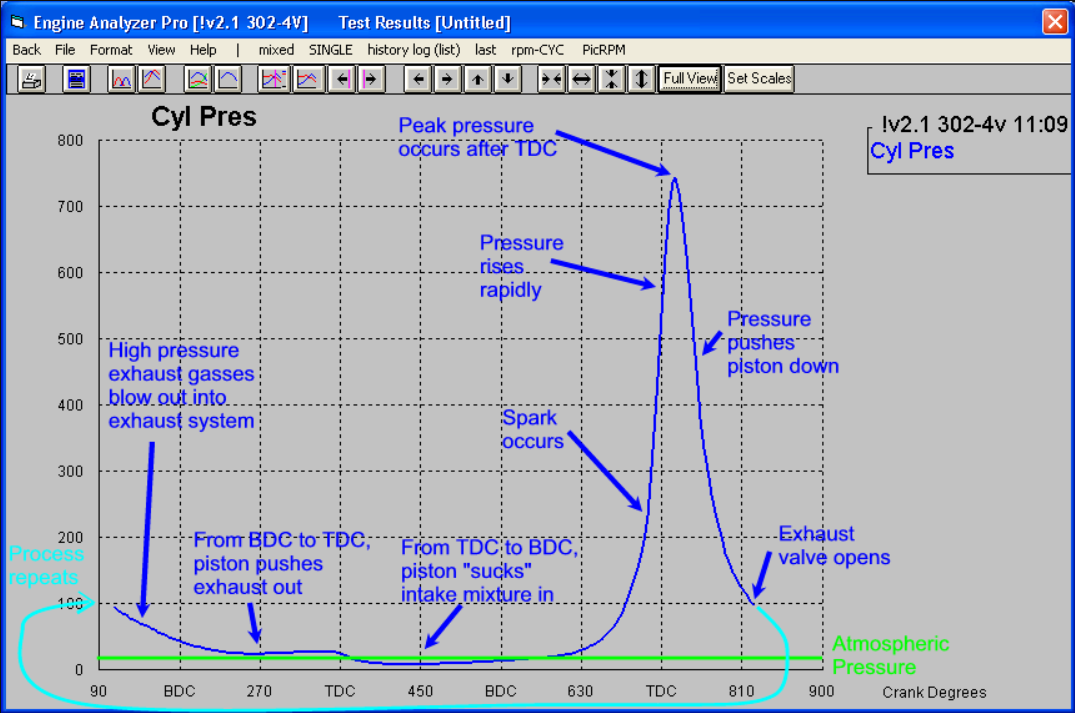
My hardware was projected based on application notes AN819 from STM about Capacitive Discharge Ignition and I choose to use the second topology available in this document because in my point of view I believe is better I do not generate alternate voltage in secondary ignition coil because this ignition coil there is a common GND and the same GND is connected to uC GND, but I believe is really good test the first topology to investigate if there some advantage to generate alternate voltage at the secondary coil, maybe this is a good alternative to generate multi-spark automatically and can help to guarantee a complete fuel combustion.



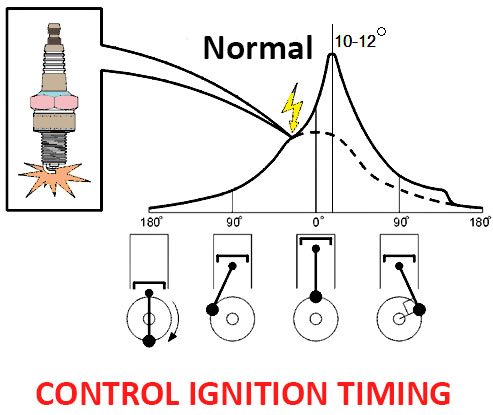




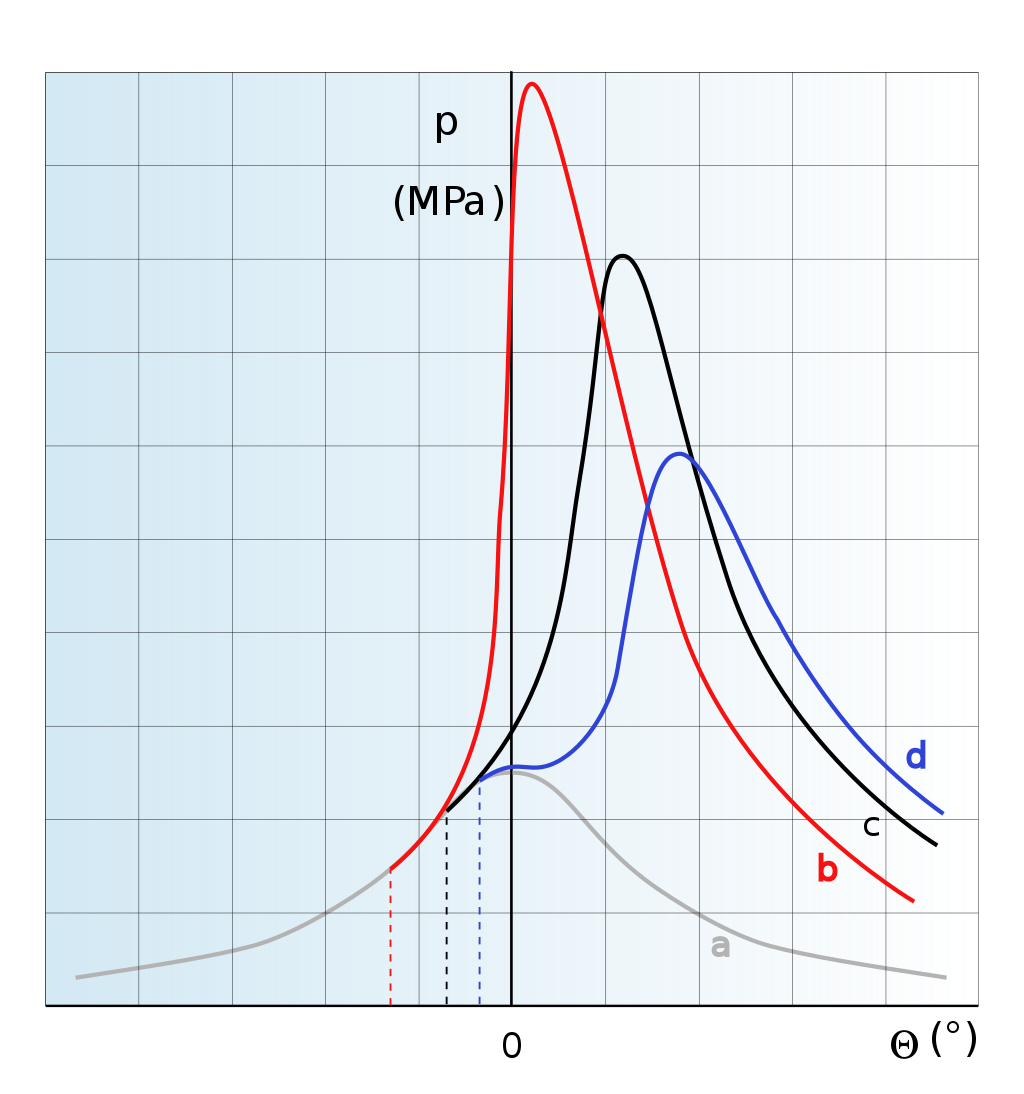
<http://publications.lib.chalmers.se/records/fulltext/255898/255898.pdf>



<http://performancetrends.com/Definitions/Images/Cylinder-Pressure-Lrg.gif>

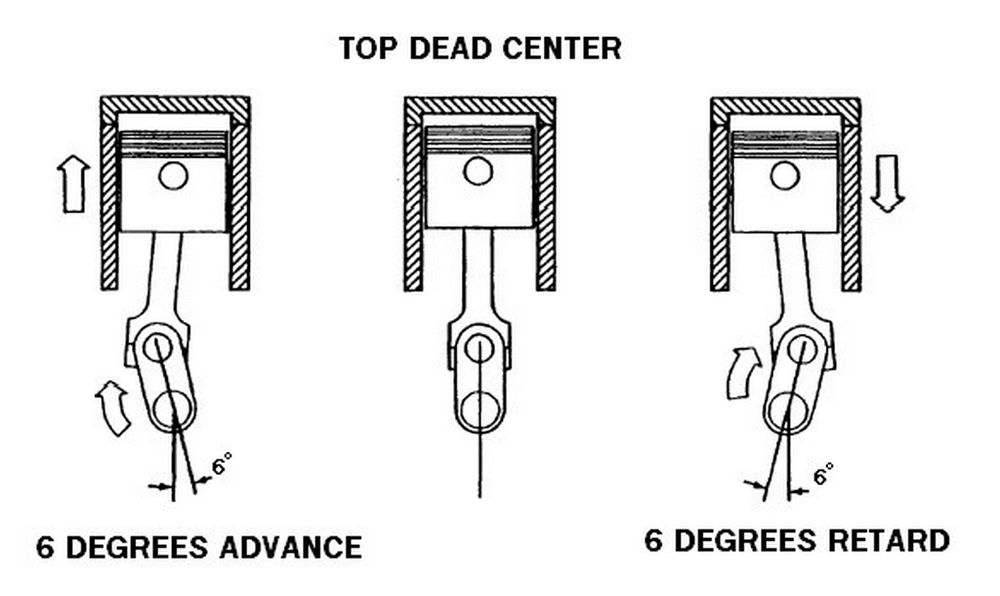


<https://www.newkidscar.com/electrics/ignition-angle-control/>

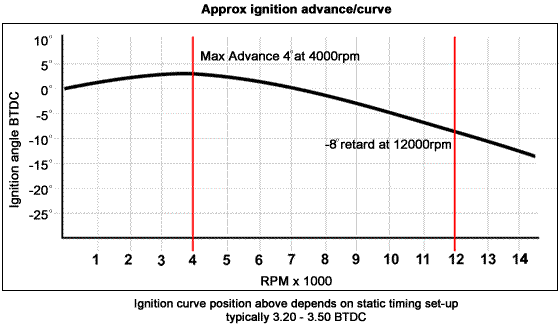


Pressure in cylinder pattern in dependence on ignition timing: (a) - misfire, (b) too soon, (c) optimal, (d) too late.

<https://en.wikipedia.org/wiki/Ignition_timing#/media/File:Pressure_patern_in_dependence_on_ignition_timing.svg>



<https://www.carthrottle.com/post/how-to-advance-your-ignition-timing-for-great-performance-gains/>



<https://advrider.com/f/threads/2-stroke-cdi-on-4-stroke.1122780/>

I decided to use the STM32F103 due your cheap cost and is very easy to buy it… Another positive factor is there is a very famous Bluepill dev board that, this is a shortcut (fast prototyping) because I don´t need to spent time to develop or build my own PCB to start to develop application for this uC… The first idea is integrated simple modules and create this CDI with some external components, but this task is not so really simple, due the electric characteristics for this kind of circuit (generates a lot of noise and EMI).

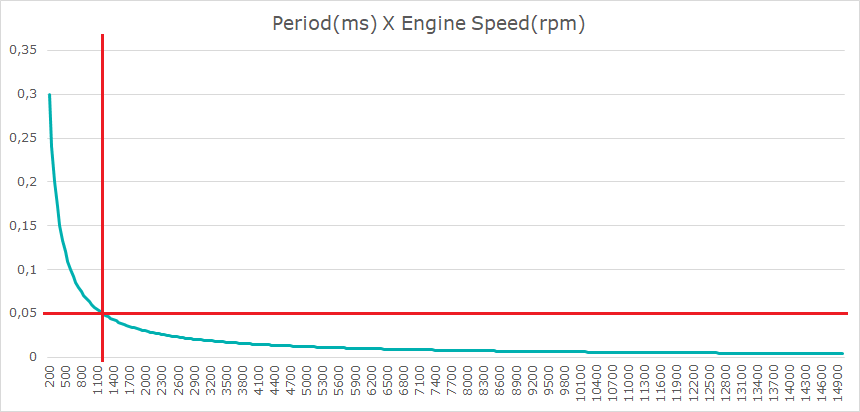
To optimize the uC operation, I choose to use the Timer2 to perform a lot of things with highest prioritization and use the minimal uC resource thinking to expand the functionality in a short future (use all resources in a best way!

The basic thing that needs to be considered in this kind of application is, the ignition timing needs to be respected and can´t be disturbed for another requests, it can´t delayed or postponed… Is MANDATORY to maintain the functionality while the engine operates (to generate the fuel combustion and consequently toque generation), for these reasons was created a software supported by hardware interruptions with high priority.

The software needs to manage 7 concurrent interruptions (which one with your respect priority) as described below:

There are 5 interrupts related Timer2:

1 - Counter overflow (Timer2 is a 16 bits counter (0-65535) and the timer clock period is 13,8ns, the microcontroller running to 72MHz), this counter will be running ever, and this event will happen when the engine speed achieves low values (under 1200rpm or 20Hz).



There are two events using Capture feature (Measure Input signal frequency):

2 - Channel 1: Pulse rising edge detect (Capture the counter value), this is the synchronism event. The treatment starts to set zero Timer2 counter, this counter will use during the input pulse measurement and output pulse generation. According this premise, this counter can measure Engine Speed after 1200rpm (50ms or 65455 timer count) up to 15000rpm (4ms or 5236 timer count). Under speeds below than 1200rpm I will count the Overflow events times 65536u plus Timer2 count value, I use this value to measure the Engine Speed, but under the 1200 the spark timing advance is fixed in the minimum value (falling edge pulse) because is very hard to reach the correct spark timing in low speeds (there is no accuracy due the non-linearization curve), use the falling edge event to trigger the ignition is a good solution for this low speed range, will ensure that engine have cranked successfully. After this event, a flag is set and as soon as possible, a calculation will be performed to program the events Output Compare Channel 3 and Channel 4, responsible to generate the trigger signal, according the advance timing calibrated in function the engine speed. If the engine speed is under 1200rpm, set the output and program the Channel 4 interrupt for 1ms later and happened push down the signal (return to zero).

One of the most important and effective things to avoid problems with electrical interference/EMI noise caused by spark ignition in input signal from VRS sensor is activate the input filter for external interrupt, it was configured for 8 clocks (internal clock period), after elapsed this period, in case that the input status do not change level, it is considered a valid transition (in this case zero to high level, rising edge detect) and trigger the internal interrupt event… This setup must be validated in different situations, until this moment was set as 3 in STM32Cube for test purpose and it works properly, but maybe in new context will need to reconfigure.

3 - Channel 2: Pulse falling edge detect (Capture the counter value, at this moment I can measure the time between rising edge and falling edge events (Duty cycle) and I know this Duty is equivalent to 22 degree angle displacement (Crankshaft).

There are two events related Output Compare (generate Output signal: Trigger pulse to SCR gate)

4 - Channel 3: Generate the rising edge pulse

5 – Channel 4: Generate the falling edge pulse after a calibratable time (1ms for Duty Cycle)

6 - Systick counter interruption: is responsible for system clock (the scheduler is based on this counter) and increment the counter which 1ms, this is a 24bits timer (need to confirm in datasheet).

7 - Timer4 is responsible to generate the PWM signal, this signal is responsible to control the Inverter module (a Royer Oscillator that generates the high voltage: 311,1Vp) to charger the Capacitor. The operation frequency is 25KHz (htim4.Init.Period = 1440, the period is 2x1440) to avoid generate some audible noise during the operation and the PWM is 20% (sConfigOC.Pulse = 600) approximately…

Interrupt prioritization and reasons to do it:

Timer2 Overflow / Input pulse rising edge / Input pulse falling edge (they will never happen at the same time, 0 – Highest priority))

Output pulse rising edge / Output pulse falling edge (they will never happen at the same time, 1)

Timer4 – 2 (it must do not disturb the input and output signals)

Systick – 3 (this is the lowest priority because it is fundamental for this system accomplish the pulse management (read VRS sensor and generate a output pulse) than do the other things in a precise time and this arbitration didn´t cause any bad consequence in this context…

It was created a scheduler with tree different temporal tasks: slow, medium and fast, which one with your respective execution time period, but this task scheduler, will be executed after scenario calcs are performed before… This scheduler module is based on Systick counter!

To manage the spark advance, the approach is under the 1200 rpm, the advance is fixed in minimal advanced possible (18 degree in my simulation), and I created an engine speed limitation to cut the ignition when the speed overtakes the engine speed limit

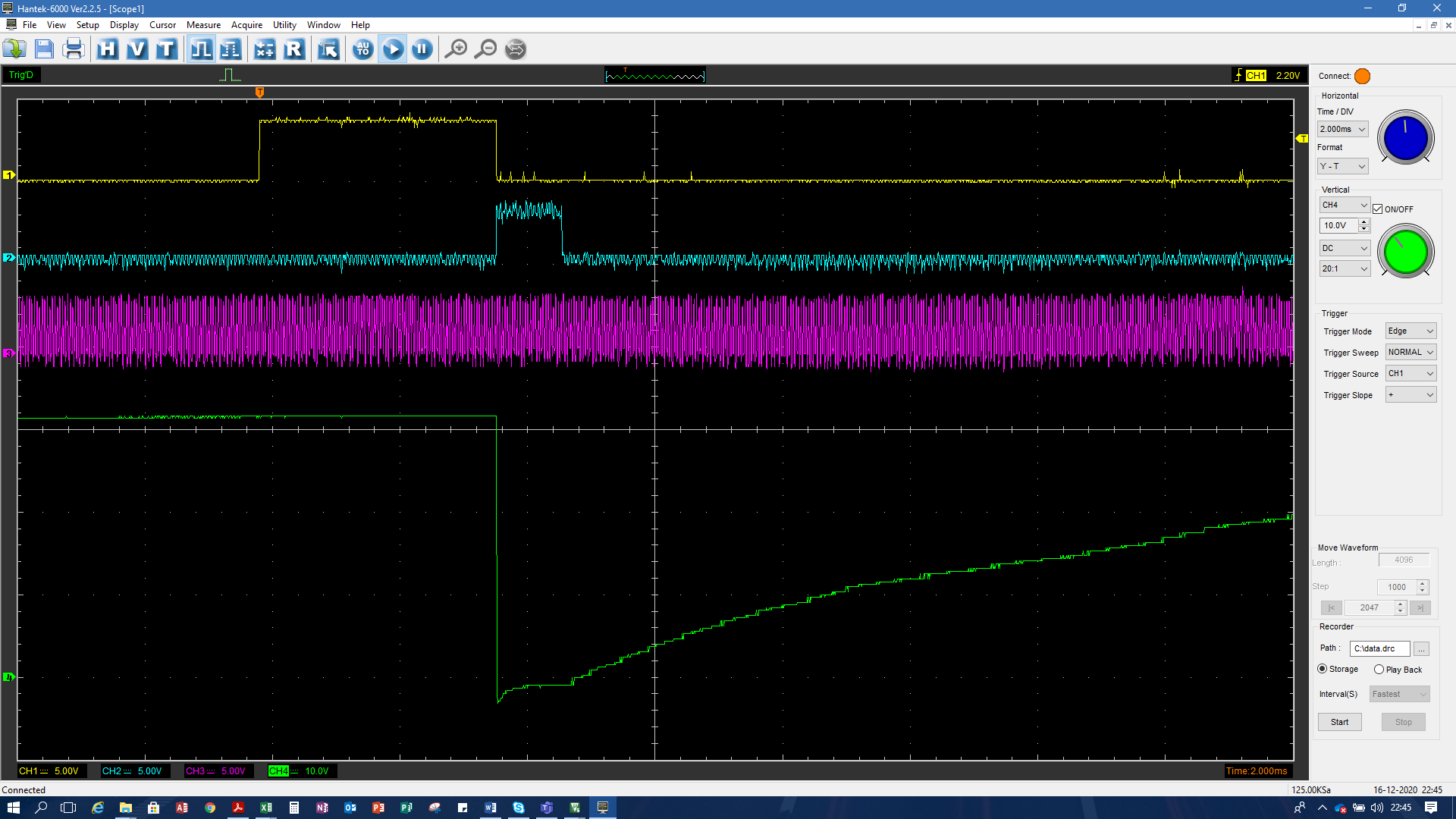
Software and hardware validation

I have created a dedicate software that generates the pulse simulation to emulate the real VRS sensor, it generates a square wave with the same pattern that the pulse came from VRS sensor. I need to check in details the system accuracy, to perform this test I will build a simulator using a stroboscopic light and a electrical motor to measure how the system performing in dynamic conditions, because using the discrete simulation is not enough to prove the system quality and its reliability.

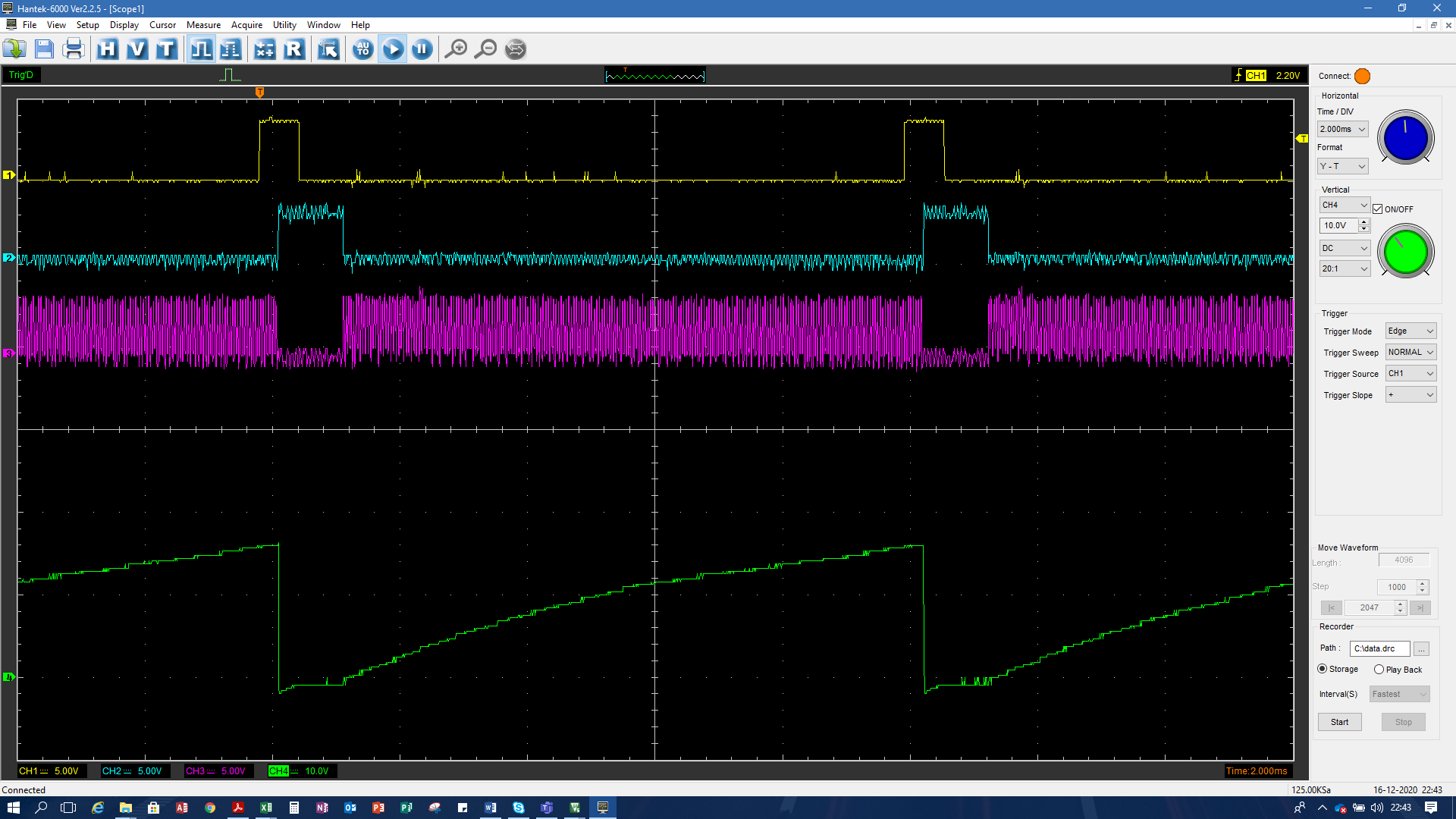
To implement the timer scheduler (based on systick counter: 24 bits down counter, but the implementation for HAL library increment a variable uwTick each 1ms event and this variable is defined as integer 32bits, is it possible count time elapsed up to 50 days uninterrupted), I read the actual uwTick value and I put this value added the time interval that I want to execute some function in a variable, I continuing reading this uwTick in each loop repetition and compare with this variable (inside the while loop), using this technic I can execute some functions periodically…

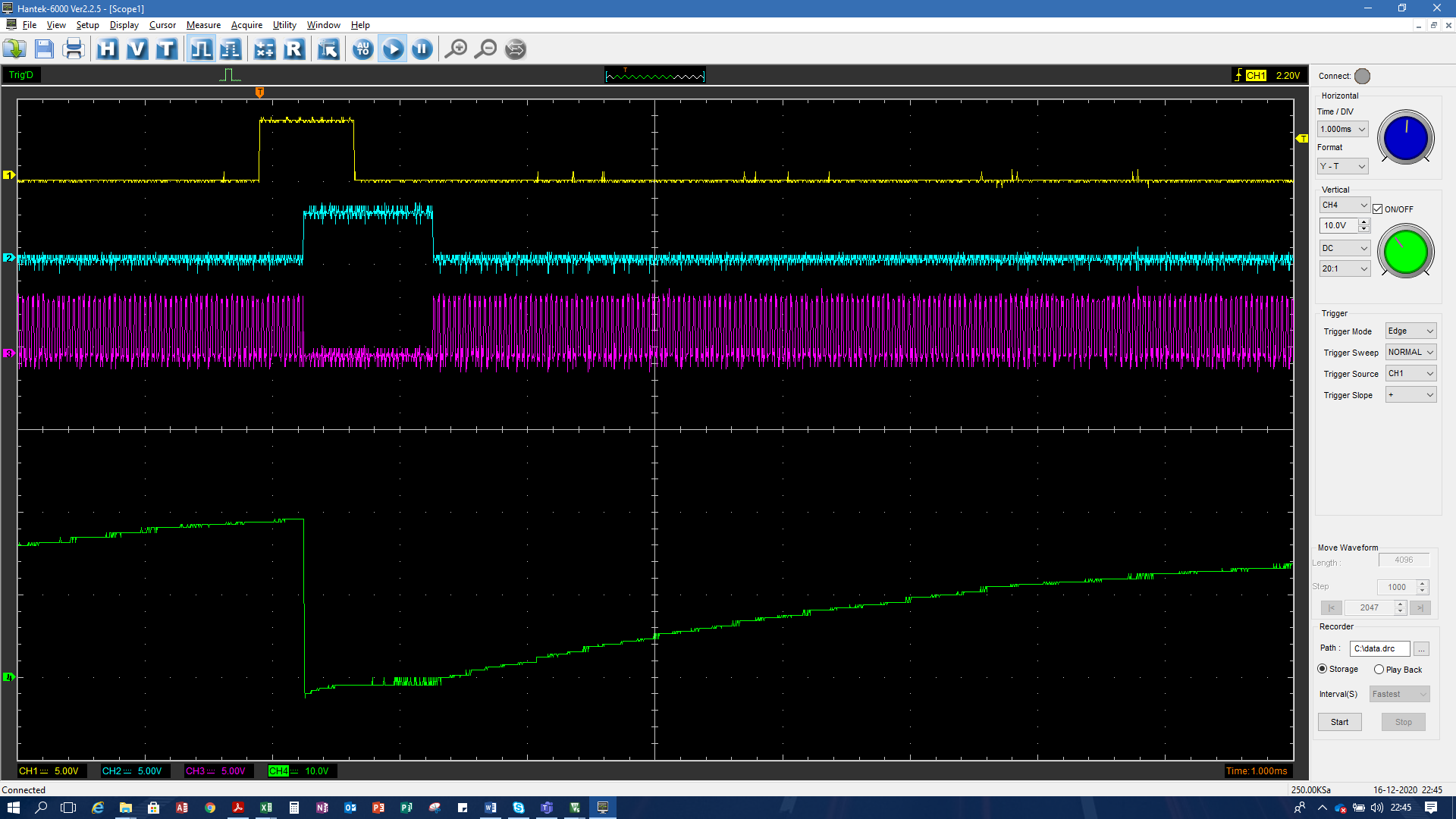
To facilitate and make fast my work, I have used STM32CubeMX to perform the basic static configuration, is a simple way to configure the microcontroller and its respective registers without made mistakes related to syntax errors or without need to understand details about the uC architecture.

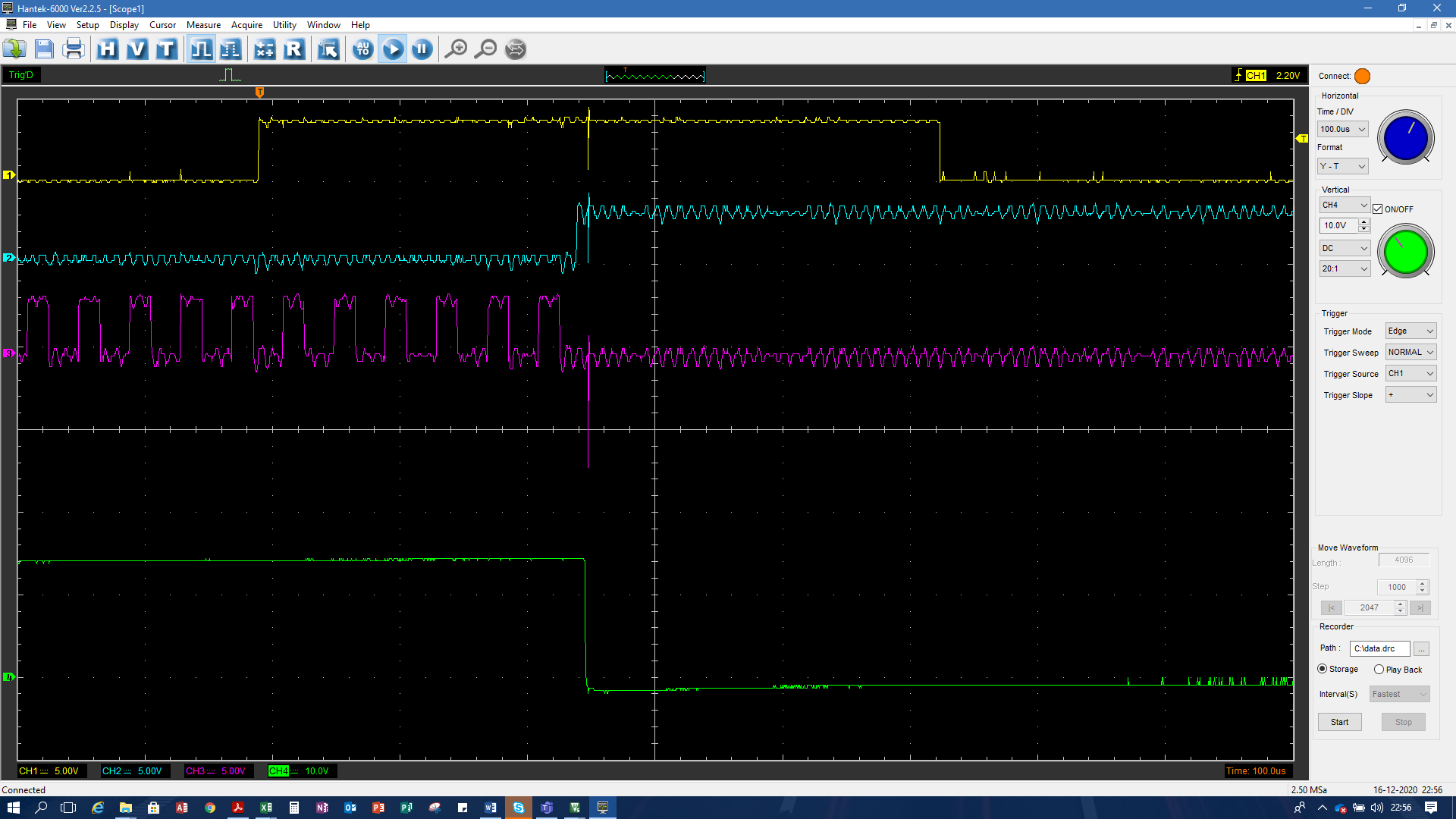
Bellow the 1200rpm, the ignition angle is fixed and is the minimal value:



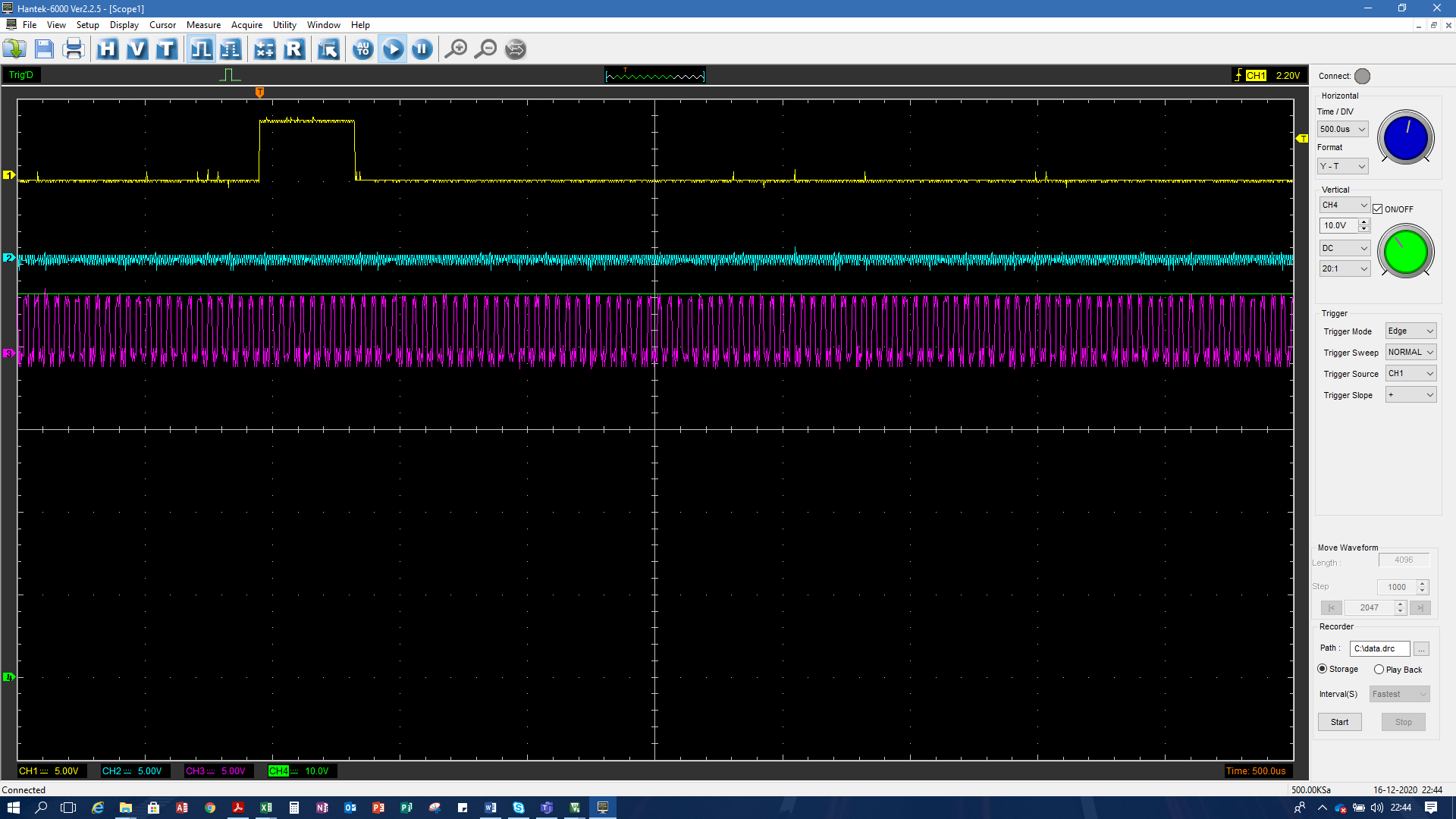
Above the 1200rpm the ignition angle is calibrate in a table and the set value is 31 (middle the duty cycle):

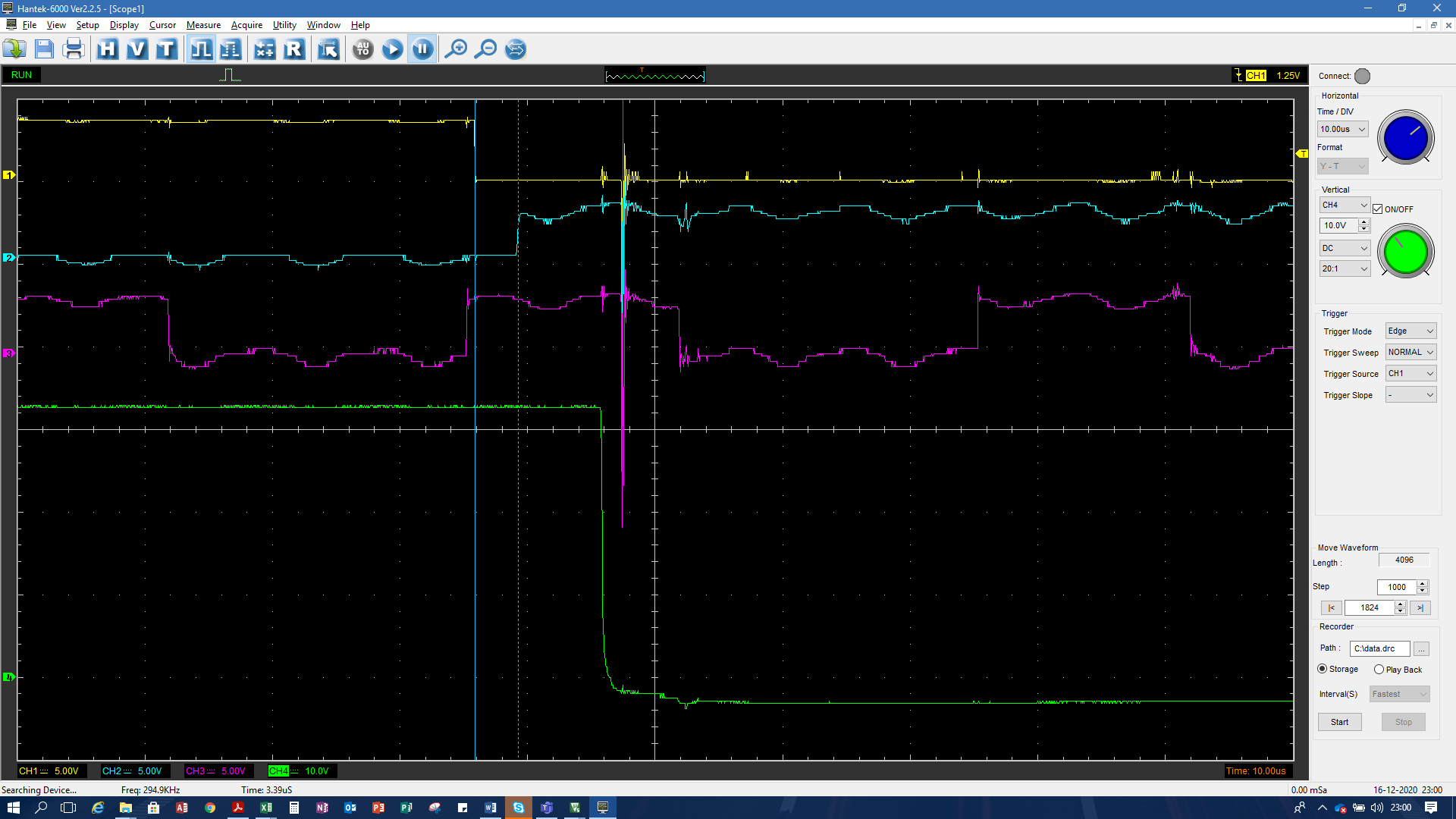


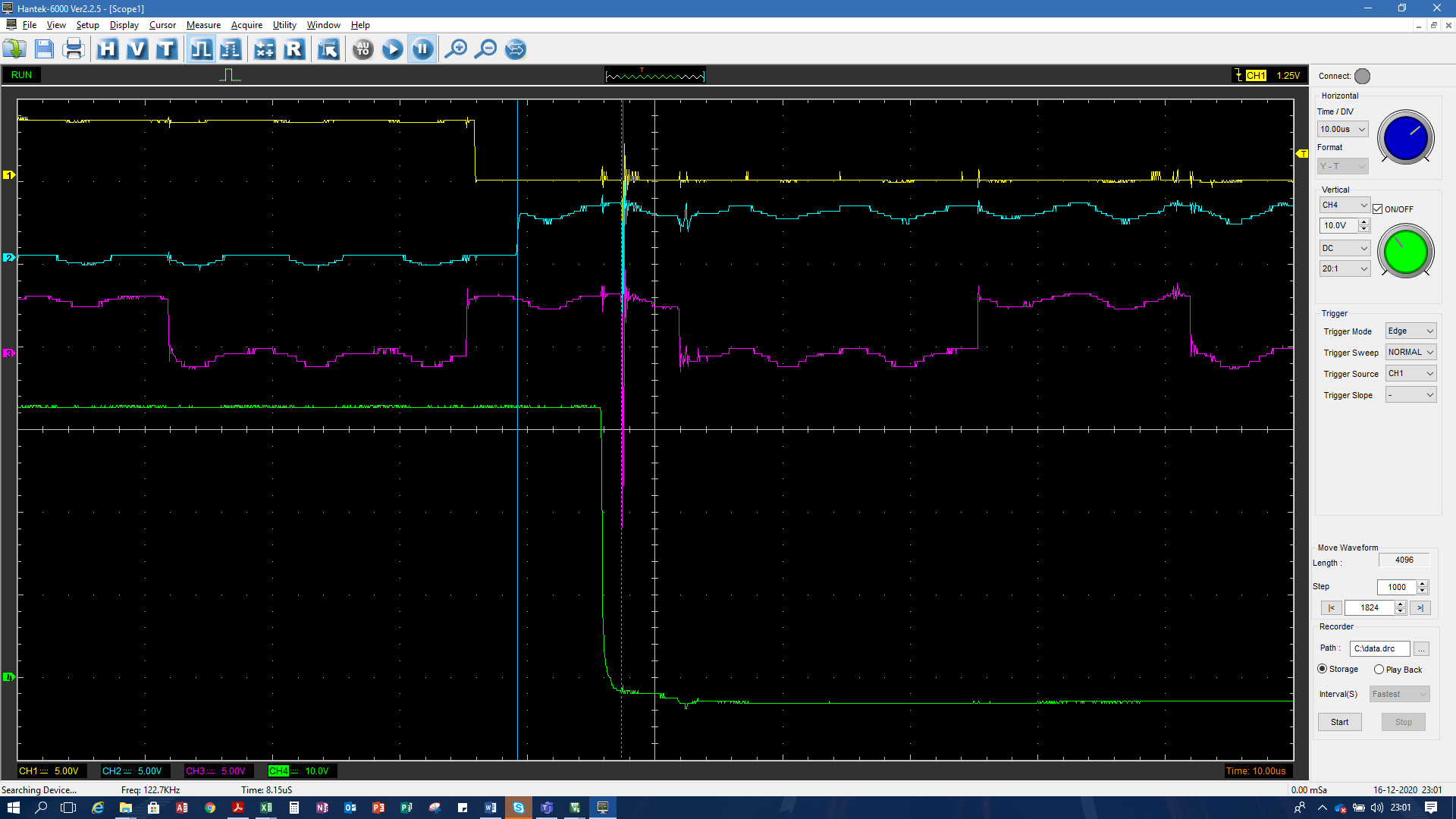




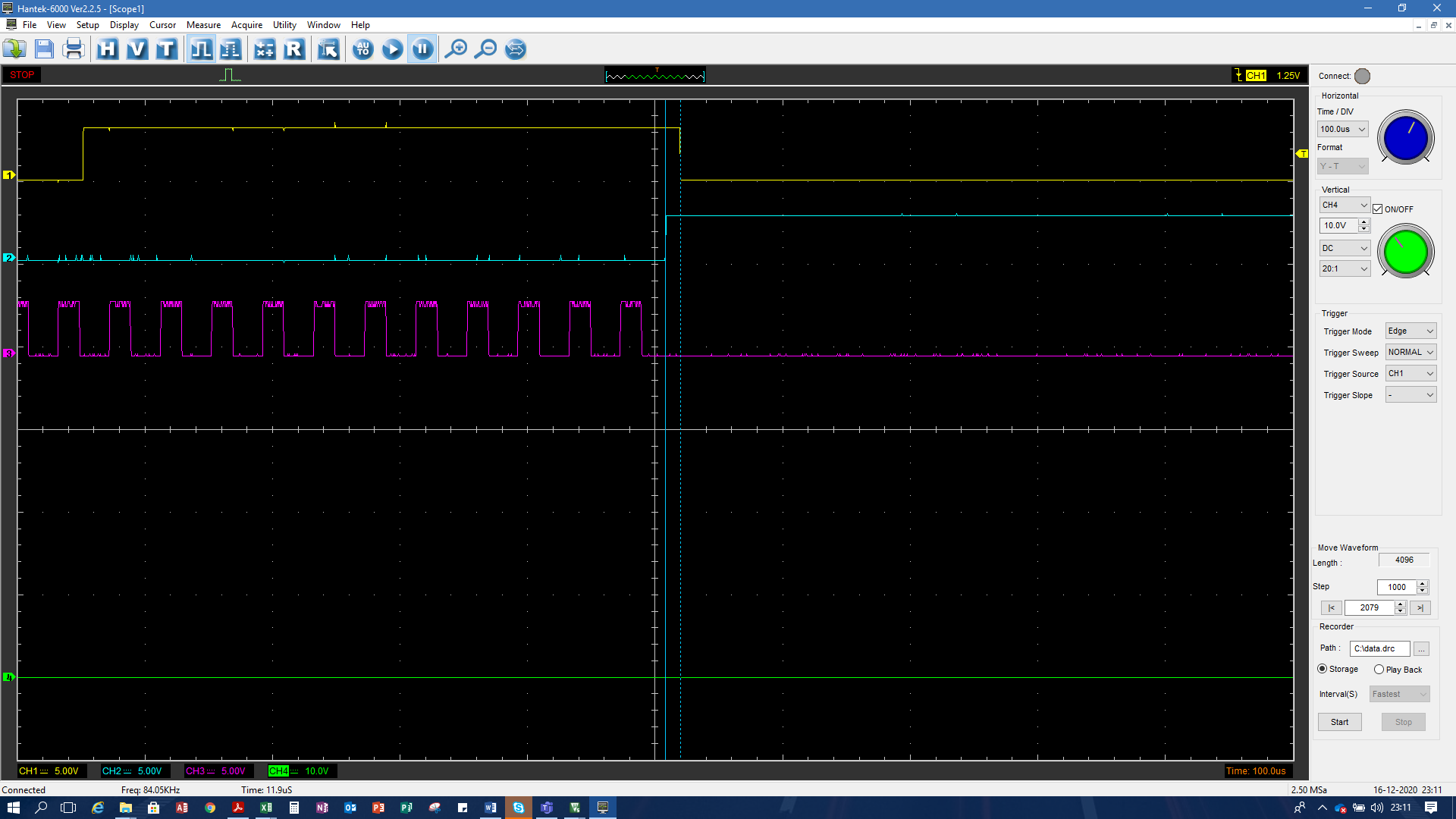
Cut the ignition:







Fixed angle…



With 64, difference between falling edge (VRS signal) and trigger

In my latest software I have implemented the Inverter control using a PWM signal generate by uC, previously I tested a pulse that started 1ms after the falling edge pulse from trigger happened and with some milliseconds of Duty (2,5 ms) and had worked very well, I need to compare this two different solutions and understand the advantages and disadvantages to use each implementation, because the PWM signal generate some undesirable characteristics like reduction of spark energy due the Voltage on Capacitor was reduced when increase the engine speed (reduce the period between sparks).

To assure unexpected software behaviors I activated the IWDG (Independent Watchgog) timer to reset the uC and restart the software, I use the fast task that was set to 20ms to reinit the counter using the syntax HAL\_IWDG\_Init(&hiwdg), prescaler was set as 4, there is a specific clock for WDG timer where this clock operates in 40KHz…

My lastest software version I returned to generate a pulse to control the royer oscillator, I decided to return to this first solution because when I used PWM to control the oscillator, in high engine speeds the voltage discharger to capacitor was reduced a lot the voltage output, using an only pulse the efficiency was increased…

About simulator

To implement the simulator, It was created a DC motor driver (for brushed motor) controlled by bluepill board, the idea is to implement a digital PI controller to maintain the motor speed constantly and create a calibrate curve to generate different engine motor speed simulations and create many scenarios to validate the CDI logic and circuit.

To start the code I used the same software that I used to create the CDI, for this reason maybe I need to create some