Jammer Box

General

The Jammer box can manipulate signals coming from engine speed sensors and to put out the manipulated signals to the wiring harness leading to the ECU. The aim of the tool is to help calibration, validation and testing of the engine speed and synchronization in provoking failures in the vehicle while the engine is running.

The Jammer box can perform these manipulations on one CRK channel and one CAM channel in parallel.

The tool is controlled by a text user interface which allows to select the type of failure one wants to provoke from a pre-defined list.

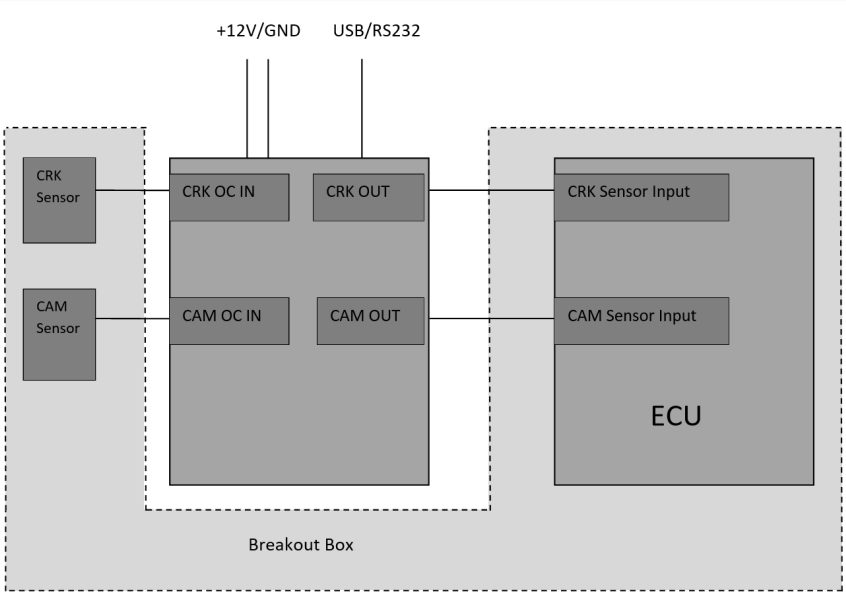
Hardware

The Jammer Box is made to have CRK/CAM input and output of 5V. Having a higher voltage in input could damage the microcontroller.

Wiring

To use the Jammer box, you will need to wire to the Jammer box:

- A power supply use either the 12V of the car battery or a 12V DC power supply.
- A RS232/USB cable to connect the PC to the Jammer Box
- The output of the CRK and CAM sensors to their respective OC entry of the Jammer box. Do not use the ACT entry it is not connected to the Microcontroller.
- Connect the Jammer CRK OUT and CAM OUT to the ECU CRK and CAM Inputs (the ACT output is recommended to not rely on the ECU pull up)



Using the Jammer Box Interface

The Jammer Box Interface is there to control the jammer box with simple commands. This software work on Windows 10 and shouldn't work with another OS.

To initialize the software, you will need:

- The COM port number at which the Jammer box is connected
- The configuration file fullname (with ".cnf" or ".txt") if it is in the same folder as the interface executable or write the full path of the configuration file (read the PatternTemplate.cnf for more details)

To find your COM Port number

1. Open **Device Manager**.
2. Locate **Ports (COM & LPT)** in the list.
3. Check for the com ports by expanding the same.

For the configuration file

- Do not change the name of the Variable
- Do not add a space between the variable name, the value and the equal sign
- Do not add a line break
- Always put a / between Cam edges position

Variables specific cases:

- Cam0Active_edge=b 'b' for both 'f' for falling 'r' for rising
- CrkSensorType=c 'c' for CPDD sensor for another sensor put 'h' for hall sensor
- FirstErSegAngle you may find the value here: AngBegWinEr (in calib file) or AngBegErSeg[0]
- Cam0SensorType value isn't used but a value is still needed (do not change)
- Cam0FilterInMicroSec=30 is only for the tolerances of the Jammer Box (do not change)

The interface will ask for those information's you should then have the main menu appear with the list of action you can make.

By simply entering the one of the numbers and pressing "enter" you will send the corresponding instruction to the Jammer box.

You will need to send the configuration of the CRK and the CAM before to generate any failure.

Once the configuration is done, you may choose the type of failure you want to generate.

When sending an instruction to the Jammer Box the Jammer will send a confirmation message, if there isn't any then the Jammer box isn't connected, or you are using another COM port that uses a serial communication.

```
Jammer Box interface
Enter the COM Port number used for the JammerBox
> COM: 8
usb com : Open
Enter the file path of the pattern file
> PatternTest.cnf
NumOfTeeth : 60
NumOfGap : 1
NumOfTeethInGap : 2
Tdc0 : 87.75
FirstErSegAngle : 42
NumOfCylinder : 4
CrkSensorType : c
Cam0NumOfEdges : 2
Cam0Active_edge : b
Cam0SensorType : c
Cam0FilterInMicroSec : 30
Cam0EdgePos : 100/300
```

Enter the number corresponding to the Diag you want to generate	
CRK	CAM
1 - Config Crk	3 - Config Cam
2 - Reset Crk config	4 - Reset Cam config
10- Crk short circuit	20- CAM short circuit
11- Crk Tooth Spk	21- CAM Spk
12- Crk Run Out	22- CAM delay
13- Crk Tooth Off	23- CAM Pat Error
14- Crk Gap Not Det	
15- Crk Seg Adp Err Lim	
16- Crk Pulse duration	
17- Crk Posn Eng Stst	
255 - Exit	

```
> 1
 Set CRK config
!1/60/2/1/87.75/42/4/c%
Message received : CRK is configurated
Message received : Successful communication
>
```

The different Failures

You will find the description of every failure, the diagnosis they are generating and how to generate them. For test on car after the engine start mind to have the engine running at more than 2000 rpm when you add the failures. If not the car will surly stall.

11 – Short circuit

Set the CRK output signal to the ground or to the battery, the failure is determined by the user.

This failure generates the diagnosis **EveCrkScg** and **EveCrkScb**.

Procedure: launch the failure at any time.

11 – Crk Tooth Spk

Generate a 10µs pulse at the start of the falling edge of a CRK tooth.

This failure generates the diagnosis **EveCrkSpk**.

Procedure: launch the failure at any time.

12 – Crk Run Out

Generate a short cut to battery or ground (it is determined by the user) during a user given angle in °CRK after 20 °CRK after the gap. The maximum value you can input depends on the number of gaps. If there are two gaps → 360° / 2 = 180 you may only generate a maximum failure of 140 °CRK (letting 20 °CRK at the end to have a normal gap). If you exceed this value, you will generate a short circuit to ground or o the battery depending of the choice you have made.

This failure generates the diagnosis **EveCrkRunOut**, **EveCrkNoiseScb** and **EveCrkNrToothOrng**, typically:

- 50 or more °CRK → **EveCrkNoiseScb** or **EveCrkNoiseScg** (depending on the user choice)
- Around one CRK tooth in °CRK → **EveCrkNrToothOrng** (because you are only removing one tooth)
- 360 or more °CRK → **EveCrkScb** or **EveCrkScg** (depending on the user choice)

Procedure: launch the failure at any time, the value chosen will affect the diag you want to generate

13 – Crk Tooth off

Generate a short cut to battery during a tooth, for a user given number of teeth.

$$\text{Distance between teeth off} = \frac{number\ of\ teeth\ between\ gap}{number\ of\ teeth\ off}$$

This failure generates the diagnosis **EveCrkNrToothOrng** and **EveCrkSynLoss** depending on the number of teeth that are taken off, typically:

- Tooth off --> **EveCrkNrToothOrng**
- 3 Teeth off --> **EveCrkSynLoss**

Procedure: launch the failure at any time, the value chosen will affect the diag you want to generate.

14 – Crk Gap Not Det

Generate a tooth in the middle of the gap, the teeth length depends on the sensor type. If the sensor is a CPDD the added tooth is a 49µs pulse if not it is the size of half the last tooth.

This failure generates the diagnosis **EveCrkGapNotDet**. To generate diagnosis the failure must start at the engine start.

Procedure: launch the failure at the engine start

15 – Crk Seg Adp Err Lim

This failure delays the last CRK tooth of the first ER segment by a user given angle. This is to make one segment bigger than another.

This failure generate the diagnosis **EveSegAdpErLim** may generate a backward diagnosis if the delay angle is too big or also **EveCrkNrToothOrng** if the delay is the length of a tooth.

Typically, we want one segment 0,7% bigger and the backward trigger is if the last tooth is 1,5 or 1,3 time bigger than the last tooth. For a 4-cylinder car a segment is 180 °CRK: 180 * 0,7% = 1,26 to have **EveSegAdpErLim** you will need to delay the tooth for at least 1,26 °CRK.

Procedure: launch the failure at any time, the diag will rise only during a deceleration of the engine without braking or accelerating. For the HIL test, create a pattern with a resembling slop (from 4000 rpm to 1000 rpm in 2-3 seconds)

16 – CRK Pulse duration

This failure changes the pulse duration of the CRK sensor by a user given duration.

This failure generates the diagnosis **EveCrkPlsOrng** and may generate **EveCrkPlsBackPlaus**.

Procedure: launch the failure at any time

17 – CRK Posn Eng Stst

This failure takes off a user determined number of teeth. It is done once to change the number of teeth seen by the ECU at a synchronised start.

This failure generates the diagnosis **EvePosnEngStstLoss** and **EvePosnEngStstOrng** depending on the number of teeth taken off.

- 1 Tooth off --> **EvePosnEngStstOrng**
- 3 Teeth off --> **EvePosnEngStstLoss**

Procedure: launch the failure at the engine start

20 – CAM Short circuit

Set the CAM output signal to the ground or to the battery, the failure is determined by the user.

This failure generates the diagnosis **EveCamScg** and **EveCamScb**.

Procedure: launch the failure at any time

21 – Cam Delay

It delays the CAM signal by a user given value in °CRK.

This failure generates the diagnosis **EveCamPosnOrng**, **EveCrkSynLoss** and **EveCamToothOff**. Typically:

- 20 °CRK before the sync → **EveCrkSynLoss**
- 20 °CRK after the sync → **EveCamPosnOrng**
- One Crk tooth delay → **EveCamToothOff**

Procedure: launch the failure at any time, the value chosen will affect the diag you want to generate.

22 – Cam Spk

It generates a pulse of 10µs on the CAM signal at every rising edge and falling edge of the CAM.

This failure generates the diagnosis **EveCamSpk**.

Procedure: launch the failure at any time.

23 – Cam Pat Err

For every 7 active CAM edges the rising or falling edge will be ignored and not send to the output.

This failure generates the diagnosis **EveCamPatErr**.

Procedure: launch the failure at any time.

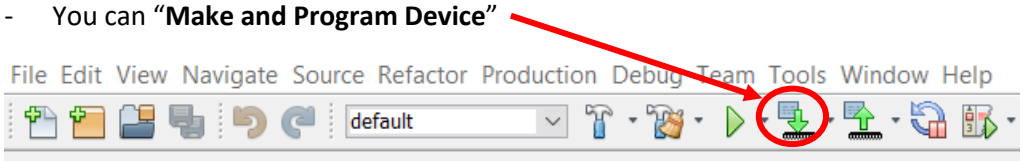
How to reprogram the Jammer box

- Download MPLAB X with the XC16 compiler

https://www.microchip.com/mplab/mplab-x-ide

https://www.microchip.com/mplab/compilers

- import the project of the embedded software
- make a "**Clean and Build Project**" (to be sure there are no problems)
- Use a PicKit or an ICD debugger and connect it to the Jammer box with the RJ-11 port
- You can "**Make and Program Device**"



The Jammer box should be reprogrammed. To test it just use the user interface and try to launch a new failure. The Jammer box should send "Failure active". If not, be sure everything is wired normally.

How to add a new failure

In the embedded software:

To add a failure function, you will first need to know went the function is called, for this you have multiple possibilities. You can use the interruption caused by the rising and/or the falling edge of the CRK and/or the CAM. Go check:

- Output_CRK in failure.c, called at the CRK interruption
- Output_CAM in failure.c, called at the CAM interruption
- IC1Interrupt(void) in main.c, CRK rising edge interrupt
- IC2Interrupt(void) in main.c, CAM rising edge interrupt
- IC3Interrupt(void) in main.c, CRK falling edge interrupt
- IC4Interrupt(void) in main.c, CAM falling edge interrupt

You can also use the "Failure_processing" that is called in the main loop and defined in the failure.c file.

Those are all the moment were your failure function can be called, depending on what you want or need you will have to play with this.

As tools you have multiple global variables and 9 timers at your disposal, the timers are all used in various functions so think to always reset the timer with the corresponding reset function. For more information look in the timer.c file. There is a big list of global variables, so I won't list them all but give some of the most important:

- failure_identify a variable that is used to select the failure to generate, it is set in the UART_receive function in uart.c
- T_TOOTH_RAW the duration of the last CRK tooth it is an integer to have the value in seconds multiply the value by 1.73us (similar variable exists for the previous teeth's)
- CRK_signal/CAM_signal those Boolean indicate the state of the CRK/CAM signal gnd is false and 5v is true
- failure_active a simple Boolean variable used to set or not the failure
- teeth_count_CRK count the number of teeth seen, it is reset when the gap is reached

Now you know most of the tool at your disposal to

In the uart.c add a case in the switch(message_identify) case of the UART_receive function. To initialize the failure Id and receive all the data wanted to start the failure or not. Think to use an unused message_identify character.

Note that you define the number of expected variables and if the number doesn't correspond a "Com error" is send to the PC.

In the interface software:

If you want to add a failure you will have to add a case in the switch case of the main function. Then create a char array with the message you want to send. The frame of the message is simple:

- Start char = "!"
- Separation char = "/"
- End char = "%"
- First char after the "!" is the message_identify this will define what action will be done by the jammer box in the UART_receive function in the uart.c file.

After the message_identify add a separation char "/" if you want to add some variable, if not you may add the end char "%". For more information read UART_ProtocolOverview.xlsx.

Use the function "`SendData()`" to send the message to the Jammer box.

You may use a start/stop functionality to start or stop the failure at will. It is simple just create a start message and a stop message and use the "`Start_Stop_Command()`". To command the star/stop we send with the message_identify a start/stop variable. This variable then is used on the embedded software to determine if it should start or not the failure.