SRT / SRAE SOFTWARE DOCUMENTATION SUPER-USER

9. KNOCK

chapter 9

Release : 2.0

REVISIONS DOCUMENT

Release	*Author*	*Date*	*Modifications*
1.0	**M.Mersier**	**05/19/2006**	• **Creation**
2.0	**M.Mersier**	**10/21/2008**	• **Strategy modifications:** **Add Diagnostoc sensor** **Add Initial knock and strong knock detection**

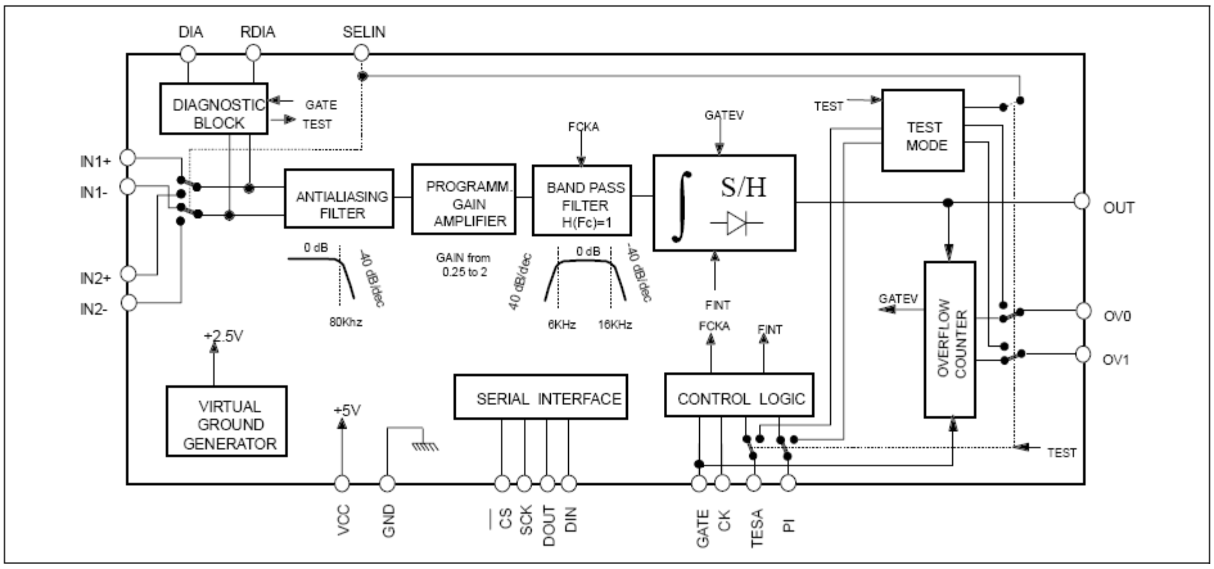
9.1 Knock Function Authorisation

For Knock authorisation, the function must be selected, by **EE.CfgU.AutoCliq_EN** and ECU function must be authorised by a password

9.2 Knock Description

The purpose of the detection and treatment of knock is to prevent detonation possibly leading to engine damage. The sensing of the particular noise, generated by detonation, is achieved by using one or two accelerometers mounted on the engine, and monitoring its signal during a programmable angular “gate”. This signal is then filtered by a band pass filter, of which the two frequencies and the associated gain are programmable. Finally the signal is integrated during the duration of the “gate”. This treatment is made individually for each cylinder.

9.2.1 Knock Hardware Driver Block Diagram



9.2.2 Knock Sensor selection

A multiplexer controlled by the pin SELIN select the sensor differential signal 1 or 2.

For each cylinder it’s possible to assign one of the both knock sensors input by the map **EE.Knk.KnockSensorAssign[CylCliq]**

9.2.3 Antialiasing low-pass filter

This filter as a fixed cut-off frequency at 80 KHz which, at the same time, converts the signal from differential to single-ended and refers it to the virtual ground (2.5V).

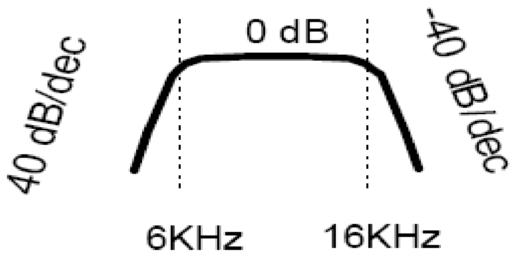
9.2.4 Programmable Gain Op-Amp

The signal is then sent to an operational amplifier with 3 possible gain choices (0.25, 0.5, 1) that are programmed via SPI. The default value is 1.

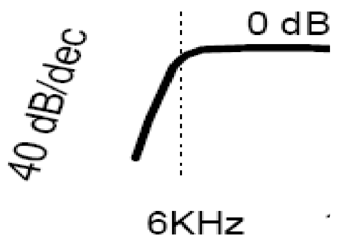
The Gain is programmable by the parameter **EE.Knk.Gain .**

9.2.5 Switched Capacitor Programmable Band Pass Filter

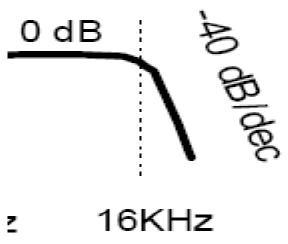
The readout chain continues with a 4th order High Pass and a 2nd order Low Pass Filters with 64 steps programmable pass band frequency, programmed via SPI.



The low cutting frequency is done for the High Pass Filter and is programmable by the parameter **EE.Knk.LowFreq**



The high cutting frequency is done for the Low Pass Filter and is programmable by the parameter **EE.Knk.HighFreq**



The frequencies are directly expressed in kHz.

Low Cutting Frequency		High Cutting Frequency	
KD=2	KD=3	KD=2	KD=3
150,00	100,00	400,00	266,67
75,00	50,00	200,00	133,33
50,00	33,33	133,33	88,89
37,50	25,00	100,00	66,67
30,00	20,00	80,00	53,33
25,00	16,67	66,67	44,44
21,43	14,29	57,14	38,10
18,75	12,50	50,00	33,33
16,67	11,11	44,44	29,63
15,00	10,00	40,00	26,67
13,64	9,09	36,36	24,24
12,50	8,33	33,33	22,22
11,54	7,69	30,77	20,51
10,71	7,14	28,57	19,05
10,00	6,67	26,67	17,78
9,38	6,25	25,00	16,67
8,82	5,88	23,53	15,69
8,33	5,56	22,22	14,81
7,89	5,26	21,05	14,04
7,50	5,00	20,00	13,33
7,14	4,76	19,05	12,70
6,82	4,55	18,18	12,12
6,52	4,35	17,39	11,59
6,25	4,17	16,67	11,11
6,00	4,00	16,00	10,67
5,77	3,85	15,38	10,26
5,56	3,70	14,81	9,88
5,36	3,57	14,29	9,52
5,17	3,45	13,79	9,20
5,00	3,33	13,33	8,89
4,84	3,23	12,90	8,60
4,69	3,13	12,50	8,33
4,55	3,03	12,12	8,08
4,41	2,94	11,76	7,84
4,29	2,86	77,43	7,62
4,17	2,78	11,11	7,41
4,05	2,70	10,81	7,21
3,95	2,63	10,53	7,02
3,85	2,56	10,26	6,84
3,75	2,50	10,00	6,67
3,66	2,44	9,76	6,50
3,57	2,38	7,52	6,35
3,49	2,33	9,30	6,20
3,41	2,27	9,09	6,06
3,33	2,22	8,89	5,93
3,26	2,17	8,70	5,80
3,19	2,13	8,51	5,67
3,13	2,08	8,33	5,56
3,06	2,04	8,16	5,44
3,00	2,00	8,00	5,33
2,94	1,96	7,84	5,23
2,88	1,92	7,69	5,13
2,83	1,89	7,55	5,03
2,78	1,85	7,41	4,94
2,73	1,82	7,27	4,85
2,68	1,79	7,14	4,76
2,63	1,75	7,02	4,68
2,59	1,72	6,90	4,60
2,54	1,69	6,78	4,52
2,50	1,67	6,67	4,44
2,46	1,64	6,56	4,37
2,42	1,61	6,45	4,30
2,38	1,59	6,35	4,23
2,34	1,56	6,25	4,17

9.2.6 Programmable Time Constant Switched Capacitor Integrator

The programmable time constant switched capacitor integrator uses the following formula:

$$\boxed{\text{Tint} = (\text{KINT} - \text{KD})/\text{Fck}}$$

Where:

- **Fck** is the system clock frequency (20MHz)
- **KD** is a coefficient determine by software according to the low and high frequency choose
- **KINT** is programmable by the calibration **EE.Knk.Kint** There is three possible Kint choice : 1600 ; 3200 and 6400

Kd	KInt	Tint = CliqIntTimeConst (**µs**)
2	1600	160
2	3200	320
2	6400	640
3	1600	240
3	3200	480
3	6400	960

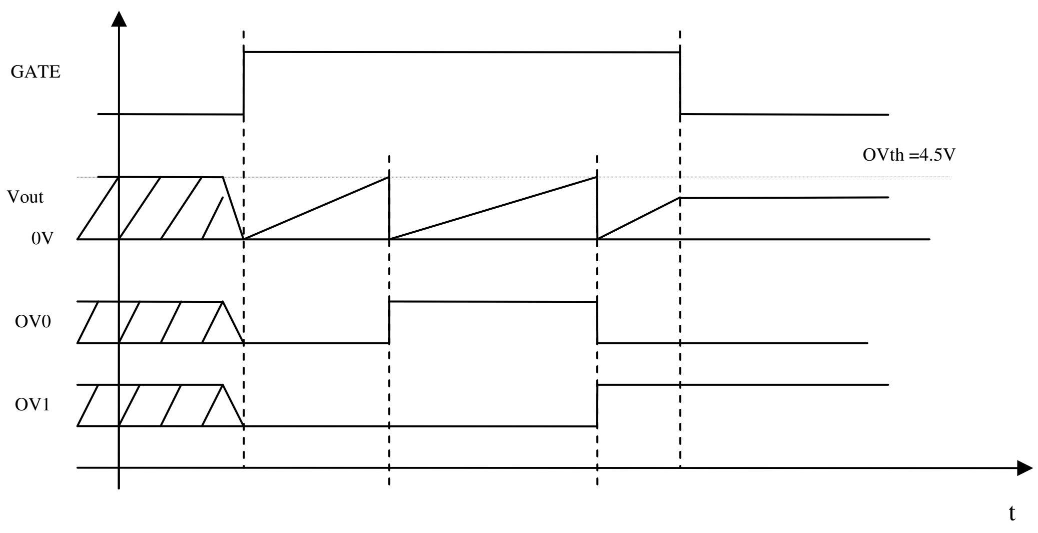
9.2.7 Measure signal

The Gate signal is used to measure the knock signal only during the knock peak, and to avoid the noise generated by the valves.

When the GATE goes high the integrator capacitor and the overflow counter output are reset and integration starts, if the integrated voltage reaches the typical value of 4.5 V the overflow counter is incremented, the feedback capacitor is discharged, and integration continues until GATE goes down. A maximum of three integrator resets are allowed. When GATE is low the integrator output and overflow counter outputs are hold

The real integrated voltage will be calculated as:

$$\boxed{Vout\_real = Vout + (2 * OV1 * OVth) + (OV0 * OVth)}$$



9.2.8 Gate Definition

Two tables allow the definition of the angular “gate” for acquisition of the noise as a function of engine speed. **EE.Knk.AngleStartFen** defines the opening position of the “gate”, and **EE.Knk.AngleStopFen**defines the closing position of the “gate” The both map are expressed in crankshaft degrees as a function of RPM

The gate has a maximum opening point of 32° BTDC and maximum closing point of 223° ATDC.Knock

In case of VTC using, to keep the gate position as function of the valve position, the came position **FD.VVT.VVT1.PhaseReal** is decreased from the **AngleStartFen** and the **AngleStopFen** witch are programmed.

If authorised by the parameter **EE.Knk.External_Gate_EN**, the gate signal can be redirected on output IGN_6, if this output is not used for coil driving.

9.3 Knock Treatment

9.3.1 Engine Noise

When starting engine and during **initbruit** mSec (125mS). The measured noise **BruitMoy** and **LastBruit** are initialised by the knock acquisition **FD.Knk.CliqBrut[CylKn]** and limited by a minimum value **EE.Knk.BruitMini**, in mVolts

A noise threshold is calculated by multiplying the average noise (**BruitMoy**) by a gain **EE.Knk.Kbruit .**

- If the measured noise **FD.Knk.CliqBrut[CylKn]** is greater than this calculated threshold, it is said to be "abnormal measurement" :
  - the last measured "normal" noise is kept as counting for the average noise calculation
  - whereas the new one (above threshold) is taken into account in the detonation (*Cliquetis*) calculation (only the noise above the average noise is kept for the detonation calculation) :

$$\boxed{FD.Knk.Deton_i = Knock_i - BruitMoy_i}$$

- If the measured noise **FD.Knk.CliqBrut[CylKn]** is lower than this threshold, it is said to be "normal measurement" and it is kept in the variable **LastBruit** (only if it is greater than the minimum value **EE.Knk.BruitMini)** for the average noise calculation:

$$\boxed{FD.Knk.Deton_i = 0}$$

The average noise (**BruitMoy**) is then calculated with the following filter using the last "normal" measurement value and limited by the minimum value **EE.Knk.BruitMini**:

$$\boxed{\boldsymbol{BruitMoy_i = BruitMoy_i + ((LastBruit_i - BruitMoy_i)^* EE.Knk.KfiltBruit)}}$$

Where:

- ***LastBruit***, last "normal" measurement
- **EE.Knk.KfiltBruit** : Noise filter coefficient.
- ***i***, cylinder number

9.3.2 Knock Conditions

The knock detection and correction is active, if authorise by the parameter **EE.CfgU.AutoCliq** :

- within a RPM zone define between **EE.Knk.RegCliqMin** and **EE.Knk.RegCliqMax**
- above a minimum throttle position threshold **EE.Knk.PapCliqMin**

Within this zone the knock detection is authorised and the advance correction is applied, but outside this zone the advance correction applied is null.

9.3.3 Strong Knock Detection

The knock is declared harmful and the ignition advance may be corrected when:

$$\boxed{FD.Knk.Deton_i > FD.Knk.Scliq\_Fort}$$

Where:

- **FD.Knk.Scliq_Fort**: Knock threshold interpolated in the map **EE.Knk.ScliqFort** as a function of engine RPM
- *i*, cylinder number

9.3.4 Initial Knock Detection

If Strong Knock is not detected, the knock is declared harmful and the ignition advance may be corrected when:

$$\boxed{FD.Knk.Deton_i > FD.Knk.Scliq\_Naissant}$$

Where:

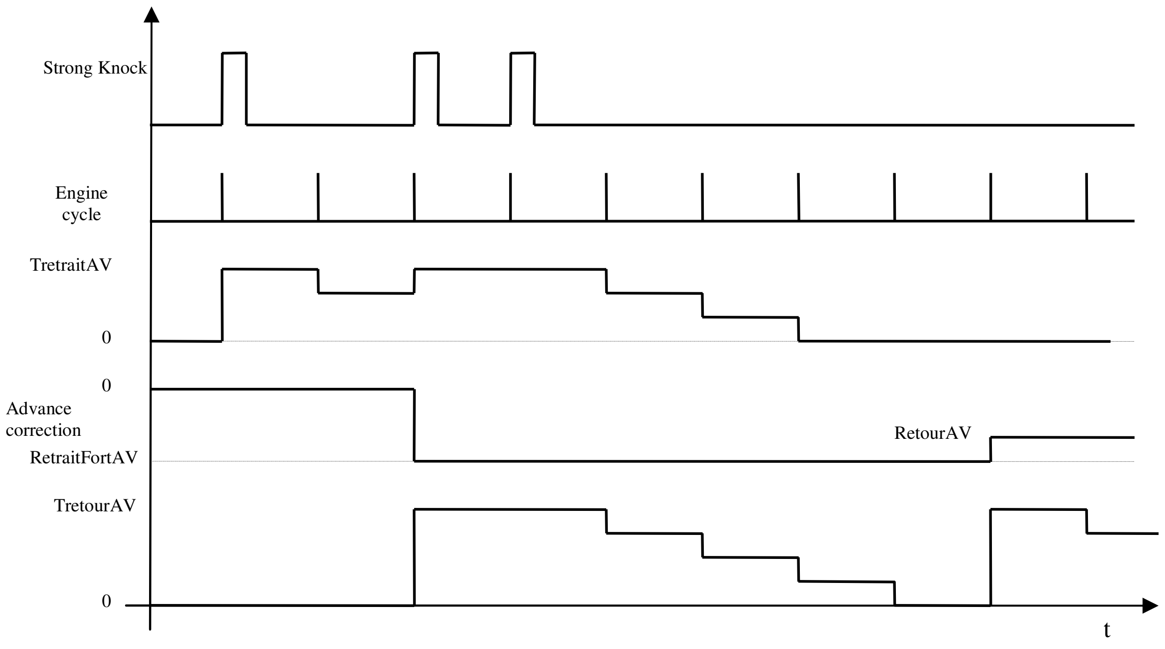
- **FD.Knk.Scliq_Naissant**: Knock threshold interpolated in the map **EE.Knk.Scliq** as a function of engine RPM
- *i*, cylinder number

9.3.5 Advance Correction

9.3.5.1 For Strong Knock

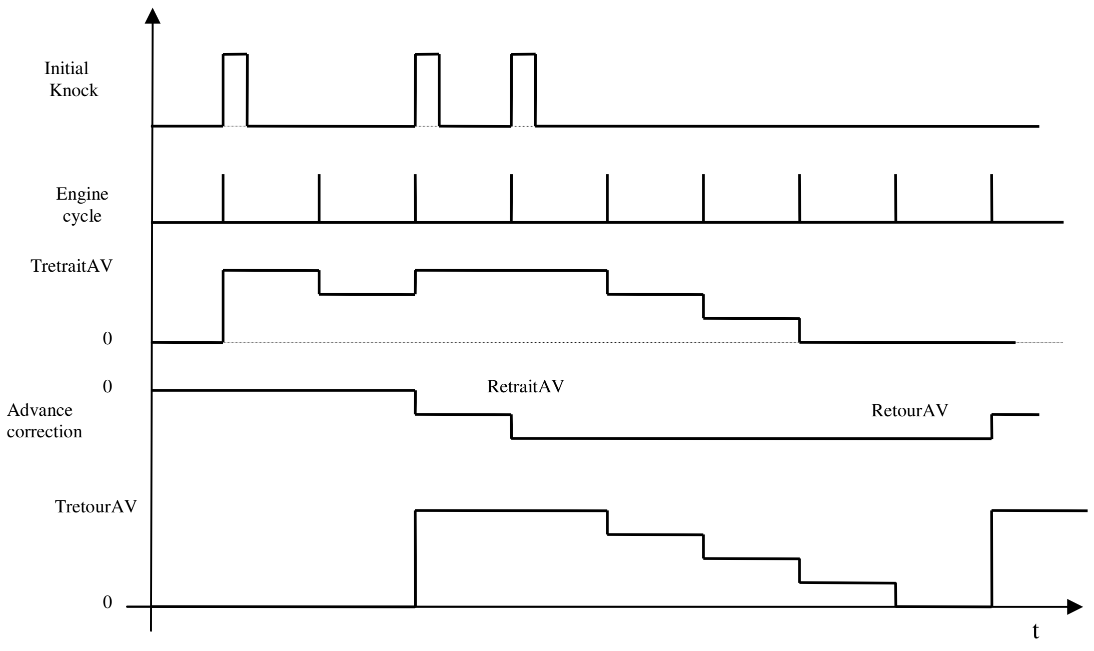
When, on the same cylinder, two detection of strong detonation occur within less than a certain number of measurements **EE.Knk.TretraitAV** (in number of engine cycles), the ignition advance of this cylinder is reduced. By the advance reduction **FD.Knk.AvCliq[CylKn]** done by the parameter **EE.Knk.RetraitFortAV**

.



9.3.5.2 For Initial Knock

When, on the same cylinder, two detection of initial detonation occur within less than a certain number of measurements **EE.Knk.TretraitAV** (in number of engine cycles), the ignition advance of this cylinder is reduced by the advance reduction **FD.Knk.AvCliq[CylKn]** done by steps **EE.Knk.RetraitAV** as long as detonation keeps being detected.



9.3.5.3 Advance Knock Correction Limitation

The advance reduction **FD.Knk.AvCliq[CylKn]** is limited to a maximum advance **EE.Knk.MaxAV.**

9.3.5.4 Without Knock detected

When there are no more knock detection within the allotted time, the ignition advance correction returns 0 by steps **EE.Knk.RetourAV** every ***EE.Knk.TretourAV*** engine cycles.

9.3.6 Advance Adaptation

There is an integral correction designed to maintain an optimal advance, just below the limit of knock, for each cylinder. This term contains negative values (ignition advance reduction if knock is detected) but may also contain positive values depending on the base map values.

This integral correction is saved in the stanby Ram.

The saved value and the calculation are using 8 breakpoints (**IdReg**) of engine speed equi-spaced between the minimum and maximum RPM thresholds for the authorisation of the strategy.

$$\boxed{Adapt16_{i,n} = Adapt16_{i,n} + \left(FD.Knk.AvCliq_i + EE.Knk.OfsAdp\right) * EE.Knk.Kadp}$$

Where:

- ***Adapt16***, integral correction as a function of RPM (n) and cylinder (i)
- ***FD.Knk.AvCliq***, knock correction as a function of cylinder (i)
- ***EE.Knk.OfsAdp***, adaptation offset allowed in a positive evolution in °crank
- **EE.Knk.*Kadp***, integral term gain

The integral correction is clamped between the max limits **EE.Knk.MaxAdp** , and the min limits **EE.Knk.MinAdp**

9.3.7 Knock Sensor Diagnostic

9.3.7.1 Low signal diagnostic

For each cylinder, if during more than **EE.Knk.CptKnockLow** mSec**,** the knock acquisition **FD.Knk.CliqBrut[CylKn]** is successively lower than **EE.Knk.BruitMini**, the assign input sensor for the cylinder consider is declared HS, and the flag **DIAG_KNOCK1(or2)_LOW** is set in **FD.Knk.DiagKnockSensor.**

9.3.7.2 High signal diagnostic

For each cylinder, if more than **EE.Knk.CptKnockHigh** successive strong knock are detected, the assign input sensor for the cylinder consider is declared HS, and the flag **DIAG_KNOCK1(or2)_HIGH** is set in **FD.Knk.DiagKnockSensor.**

9.3.8 Knock Advance Correction

If the Knock conditions are performed (see §9.3.2) the advance correction for each cylinder is:

- If the knock sensor assign to this cylinder is declared HS

$$\boxed{FD.Knk.CorAvCliq_i = FD.Knk.AvCliq_i = EE.Knk.DiagRetraitAV}$$

- If the knock sensor assign to this cylinder is declared Okey

$$\boxed{FD.Knk.CorAvCliq_i = Adapt16_{i,n} + FD.Knk.AvCliq_i}$$

Where:

- ***Adapt16***, integral correction as a function of RPM (n) and cylinder (i)
- ***FD.Knk.AvCliq***, knock correction as a function of cylinder (i)

Otherwise the advance global correction is null.