

# Engine knock detection using AURIX™

## 32-bit microcontroller

### About this document

#### Scope and purpose

This application note describes how to implement knock detection system using DSADC in AURIX™ 32-bit microcontrollers from Infineon. This offers both the opportunity for cost reduction compared to some conventional systems, and the chance to reduce CPU load. A comparison is made between AURIX™ and the Infineon Audio device.

#### Intended audience

This document is intended for software engineers who are interested in engine control systems involving AURIX™ devices. Engineers should have a good understanding of DSADC, Engine control and signal processing before reading this document.

For more detailed information please refer to the following Application notes:

- AP32343\_Position\_Minus\_Time\_Request\_PMTR
- AP32222\_DSADC\_basics
- AP32015\_Engine Knock detection using TC-1796

### Table of contents

<b>About this document</b>	<b>1</b>
<b>Table of contents</b>	<b>1</b>
<b>1 What is engine knock?</b>	<b>3</b>
<b>2 Signal processing basics</b>	<b>4</b>
2.1 Nyquist sampling theorem	4
2.2 Aliasing	4
2.3 Anti-aliasing	4
2.4 A/D converter	5
2.5 Advantage of oversampling converters	5
2.6 Filter specification	6
<b>3 System concept</b>	<b>8</b>
3.1 Conventional system concept with Audio	8
3.2 System concept with AURIX™	8
<b>4 Code implementation</b>	<b>9</b>
4.1 Resources	9
4.2 Code operation sequence	9
4.2.1 Knock window	10
4.3 DSADC configuration	10
4.4 Group delay	11
4.5 Offset compensation	12
4.6 DMA and double buffering to reduce CPU load	12
4.7 Signal extract	13
<b>5 Performance test</b>	<b>15</b>

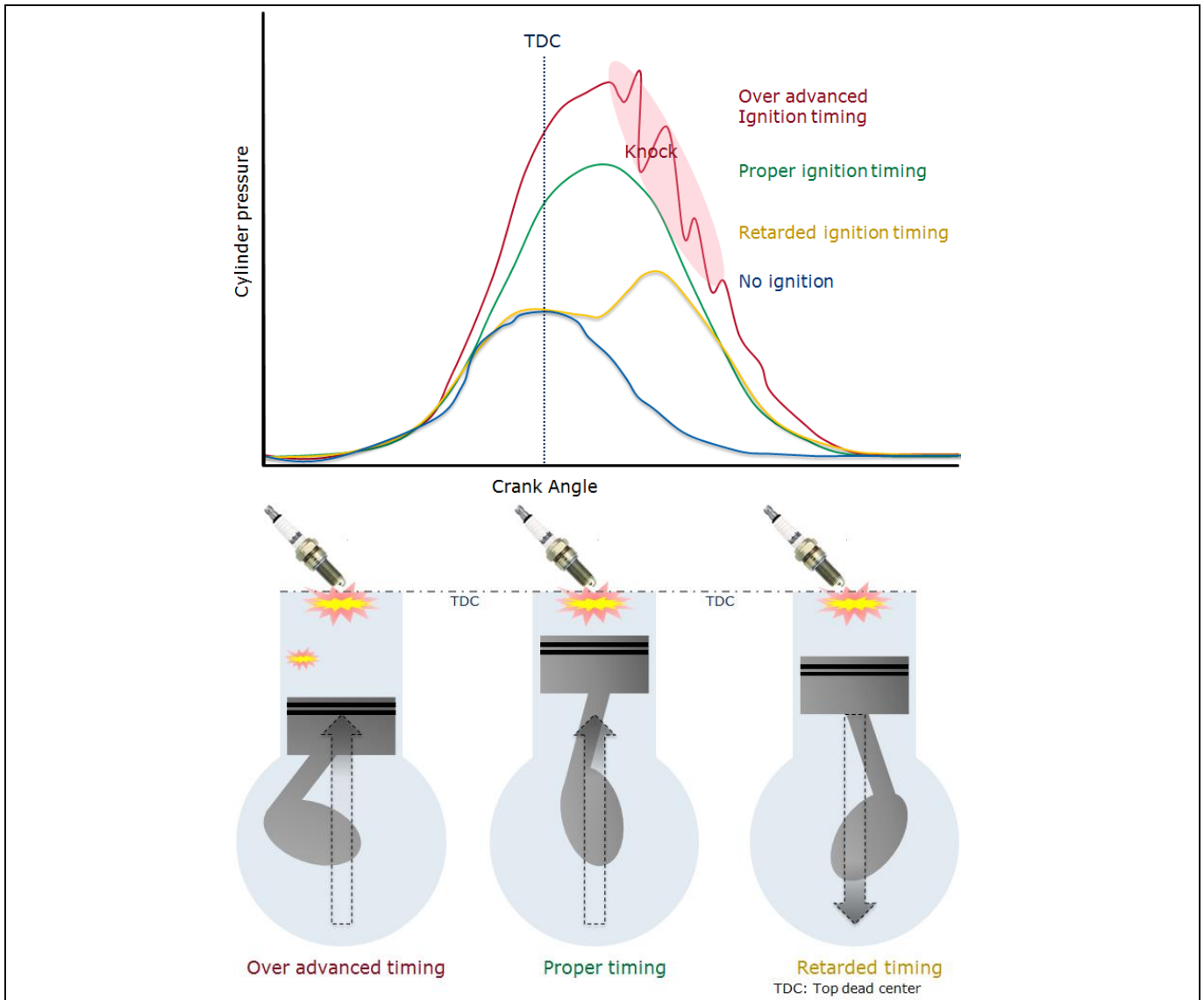
---

**Table of contents**

5.1	Input signal .....	15
5.2	Analog to digital conversion validation test.....	16
5.3	DSADC cut-off frequency validation test .....	17
5.4	Signal extraction validation test.....	19
<b>6</b>	<b>Check points .....</b>	<b>20</b>
<b>7</b>	<b>Driver API.....</b>	<b>22</b>
<b>8</b>	<b>Driver files and folder structure .....</b>	<b>23</b>
<b>9</b>	<b>References .....</b>	<b>24</b>
	<b>Revision history .....</b>	<b>25</b>

## What is engine knock?

# 1 What is engine knock?



**Figure 1 Engine ignition timing**

Generally, advanced ignition timing generates high combustion pressure. But advanced timing cannot be used unconditionally in engine control, because advanced timing leads to the phenomena of engine knocking. Therefore to get high combustion pressure in the cylinder, finding the ‘correct’ timing is very important from an engine control point of view.

In system operation the ECU (Engine Control Unit) continuously tries to find the best timing. For example, the ECU will set the initial advanced ignition timing and then, if no knock signal is detected, the ECU will gradually increase the advanced ignition timing. However, if a knock signal is detected the ECU could set a much more retarded ignition timing compared with the advanced ignition timing.

In order to distinguish normal combustion and abnormal combustion, the ECU receives a signal via a piezoelectric sensor which provides a vibration signal. According to engine RPM, mechanical structure, and cylinder atmosphere, the knock signal has a specific frequency.

## 2 Signal processing basics

### 2.1 Nyquist sampling theorem

The Nyquist sampling theorem provides a prescription for the nominal sampling interval required to avoid aliasing. It may be stated simply as follows:

The sampling frequency should be at least twice the highest frequency contained in the signal.

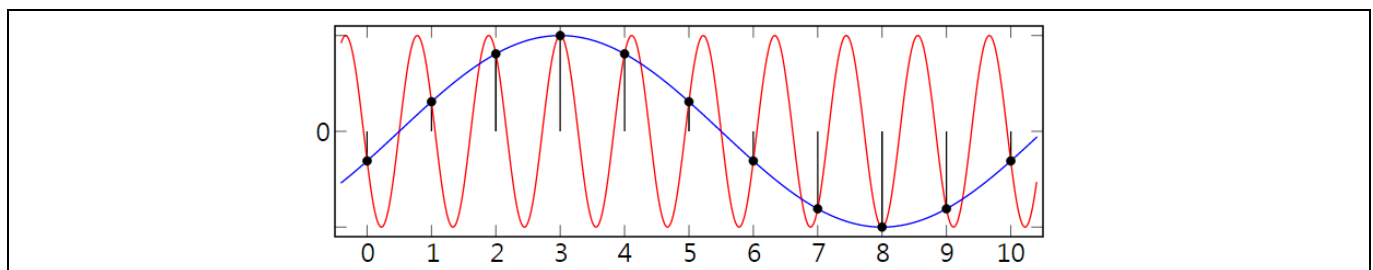
$$f_s \geq 2 f_c$$

Where:

- $f_s$  is the sampling frequency (how often samples are taken per unit of time or space).
- $f_c$  is the highest frequency contained in the signal.

### 2.2 Aliasing

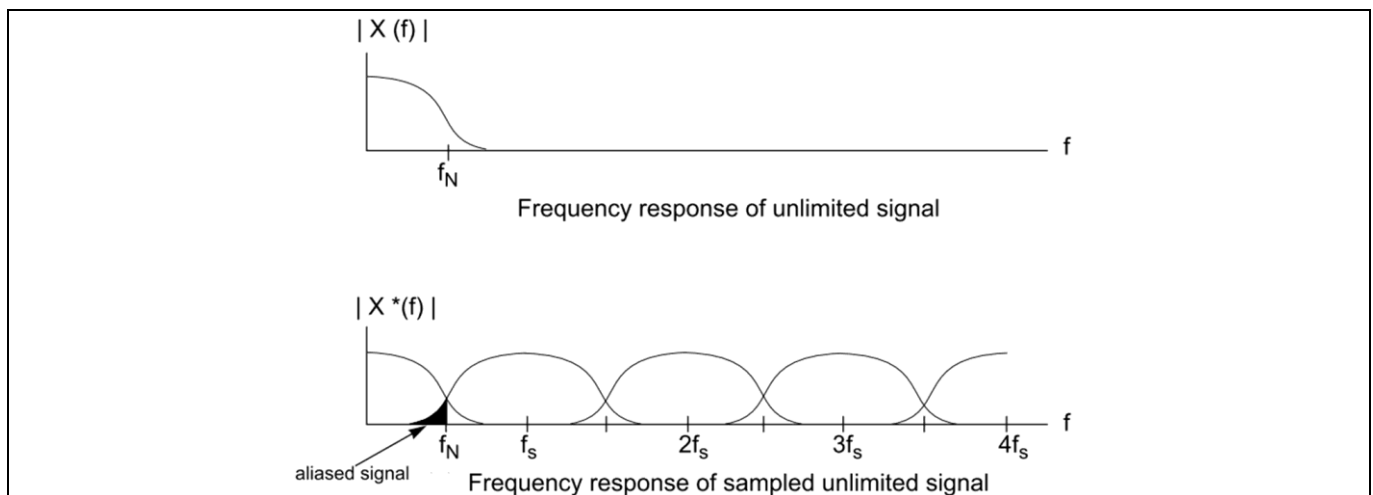
An alias is a duplicate. If the sample is equal to or below the Nyquist rate, the system will be faced with a problem. In the interpolation process, which signal in the graph below is more likely to be reproduced? From the dots, both are valid signals. Two waves hide in the same bunch of samples. This situation is called aliasing.



**Figure 2** Aliasing

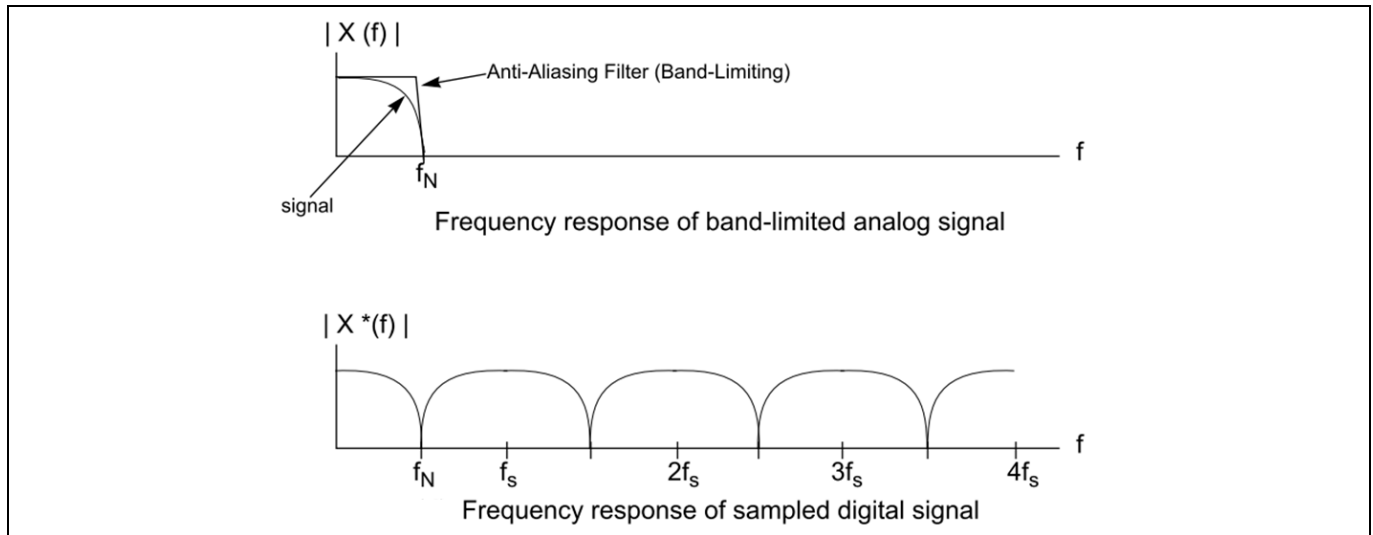
### 2.3 Anti-aliasing

Input signals Nyquist frequency cannot be properly converted and they also create new signals in the base-band, which were not present in the original signal. This non-linear phenomenon is a signal distortion frequently referred to as an aliased signal.



**Figure 3** Aliased signal

Distortion can only be prevented by running the input signal through a low-pass filter, up to the Nyquist frequency. This low-pass filter, sometimes called the anti-aliasing filter, must have a flat response over the frequency band of interest (baseband) and attenuate the frequencies above the Nyquist frequency enough to put them under the noise floor.



**Figure 4** Anti-aliasing

## 2.4 A/D converter

Most A/D converters can be classified into two groups according to the sampling rate criteria.

Nyquist rate converters (for example, Fast ADC in the Infineon Audio product) sample analog signals which have maximum frequencies slightly less than the Nyquist frequency:

$$f_N = f_s/2$$

where:

- $f_s$  is the sampling frequency.

Oversampling converters (for example, DSADC in Infineon AURIX™ products) perform the sampling process at a much higher rate:

$$f_N \ll F_s$$

where:

- $F_s$  denotes the input sampling rate.

## 2.5 Advantage of oversampling converters

The following figure shows the requirements of the anti-aliasing filter and over-sampled Nyquist rate A/D converters. Sampling at the Nyquist rate mandates the use of an anti-aliasing filter with very sharp transition in order to provide adequate aliasing protection without compromising the signal bandwidth  $f_B$ .

The transition band of the anti-aliasing filter of an oversampled A/D converter is much wider than its passband, because anti-aliasing protection is required only for frequency bands between  $NF_s - f_B$  and  $NF_s + f_B$ .

One R-C low-pass filter is sufficient for the oversampled A/D converter anti-aliasing filter.

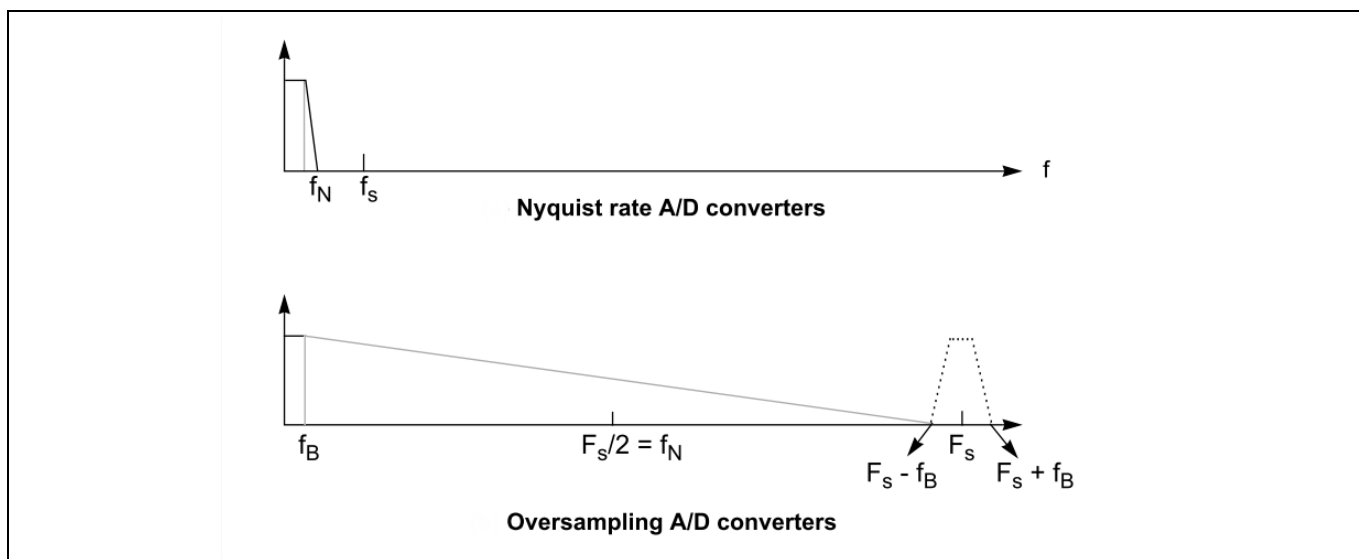


Figure 5 Advantage of over-sampling converter

## 2.6 Filter specification

DSADC has the following filter response specification:

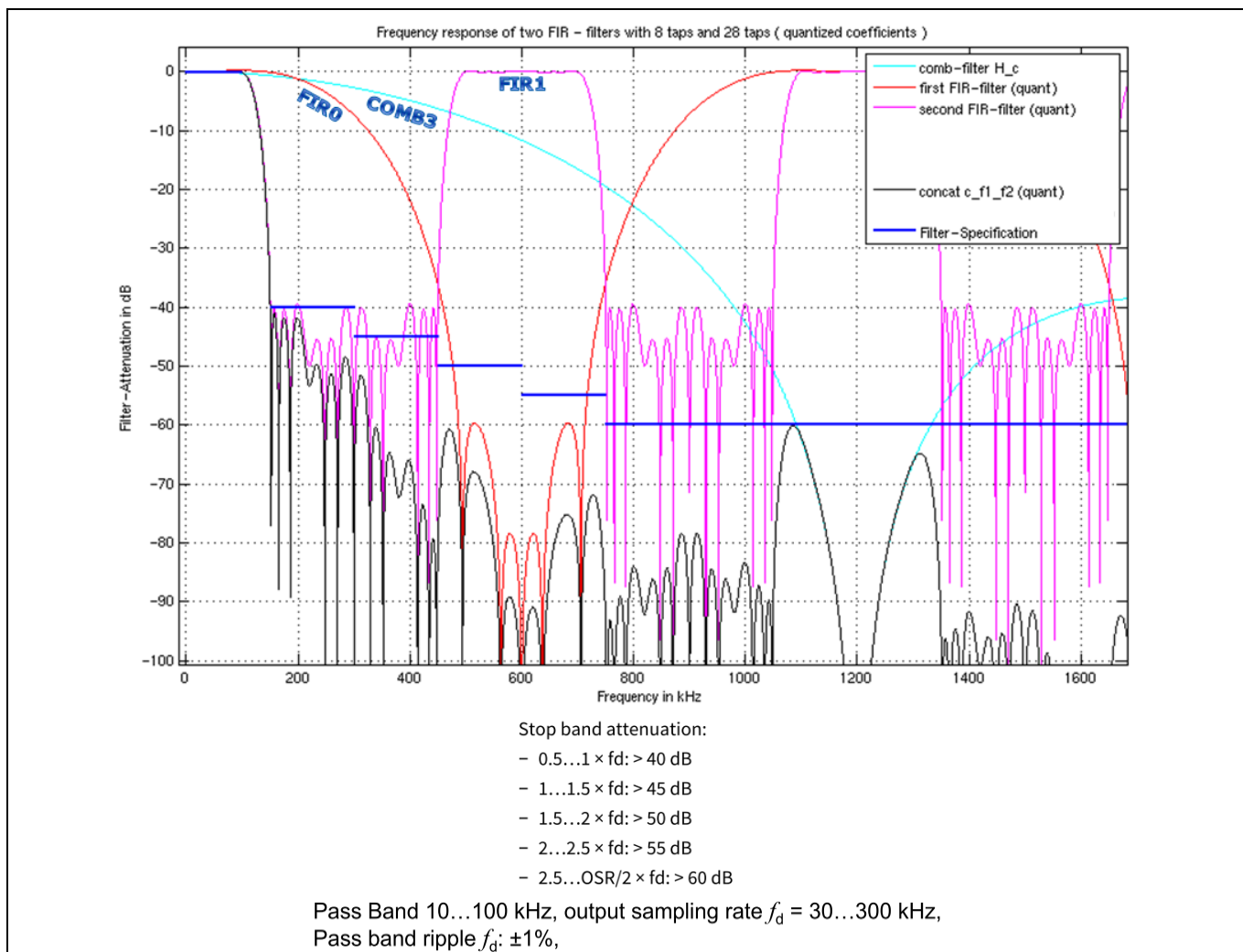
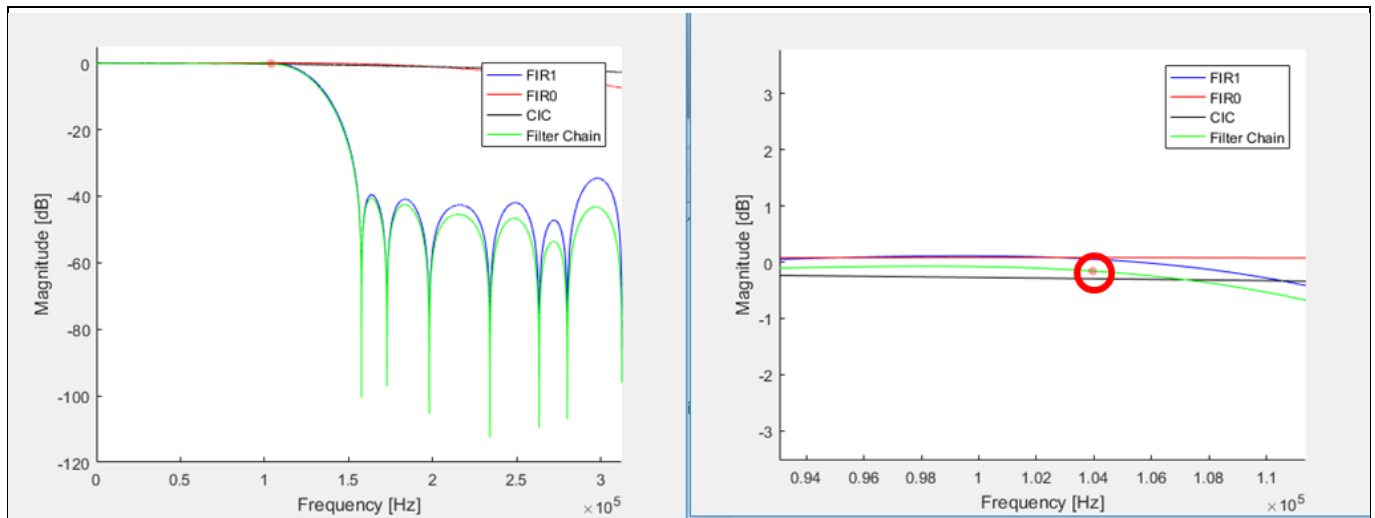
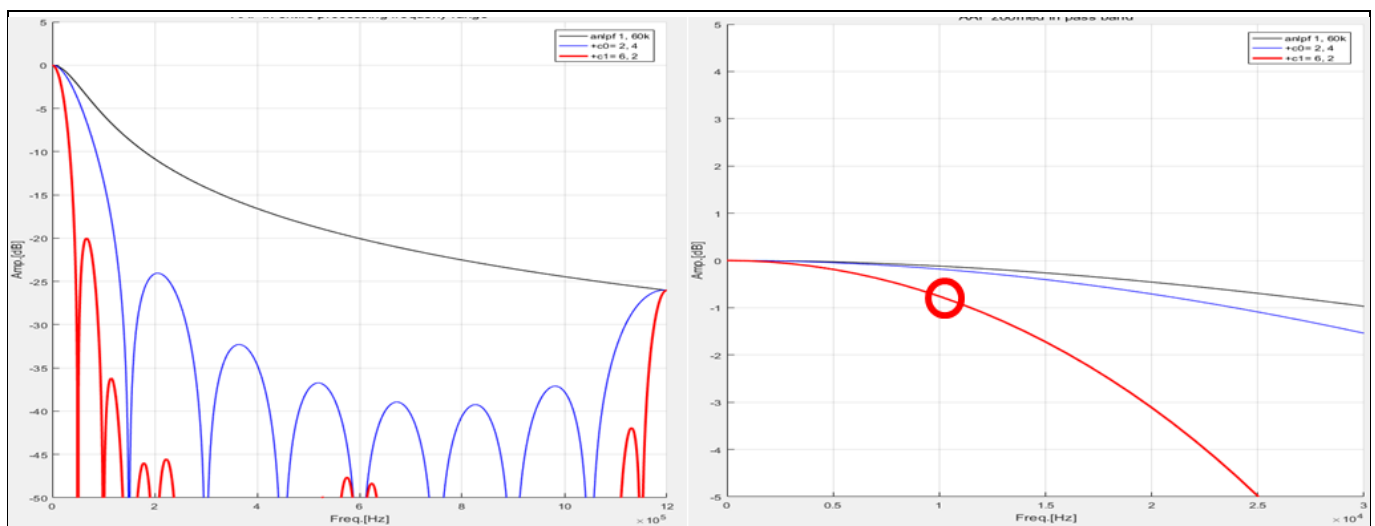


Figure 6 DSADC frequency response

When compared to the FADC, DSADC has the superior filter response. Specifically, without compensation software FIR, the DSADC shows a flat passband when compared to the FADC.



**Figure 7** DSADC passband



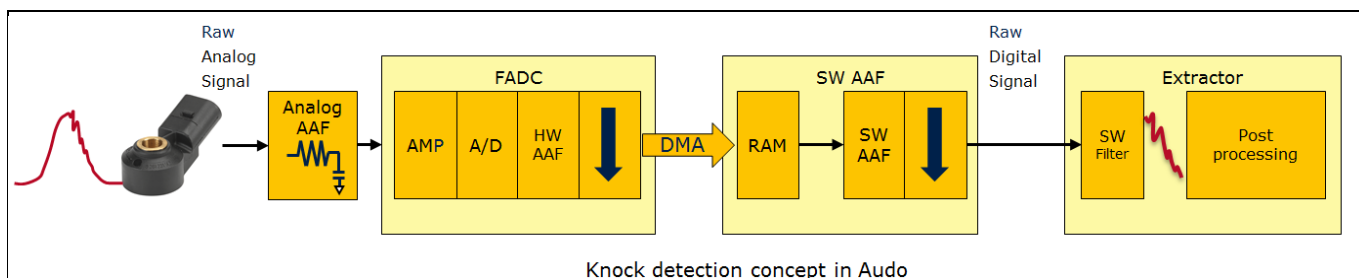
**Figure 8** FADC passband

## System concept

### 3 System concept

#### 3.1 Conventional system concept with Audio

Using FADC in Audio, the following concept can be configured:

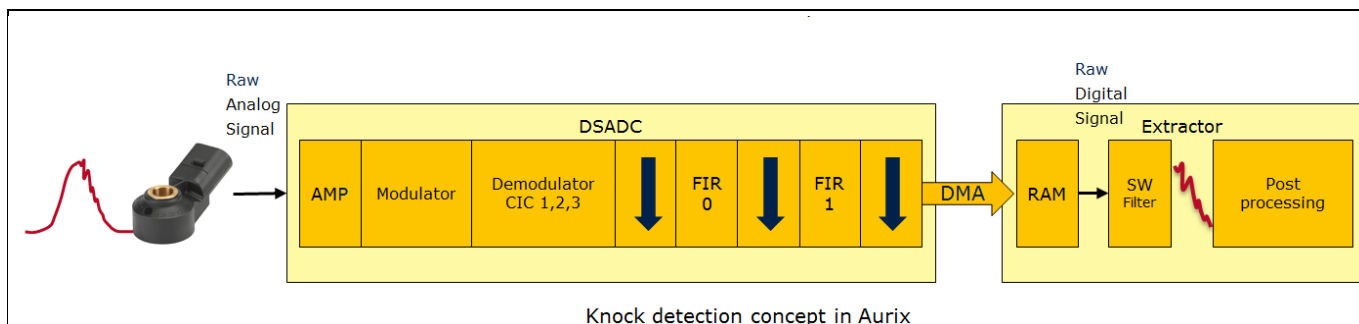


**Figure 9** Audio system concept

#### 3.2 System concept with AURIX™

Compared with the FADC concept, the external analog filter and software AAF are removed in the DSADC system concept, because DSADC provides enough filter performance to implement a knock detection system.

An AURIX™ DSADC implementation offers the opportunity for cost reductions and to lower the CPU load.



**Figure 10** AURIX™ system concept



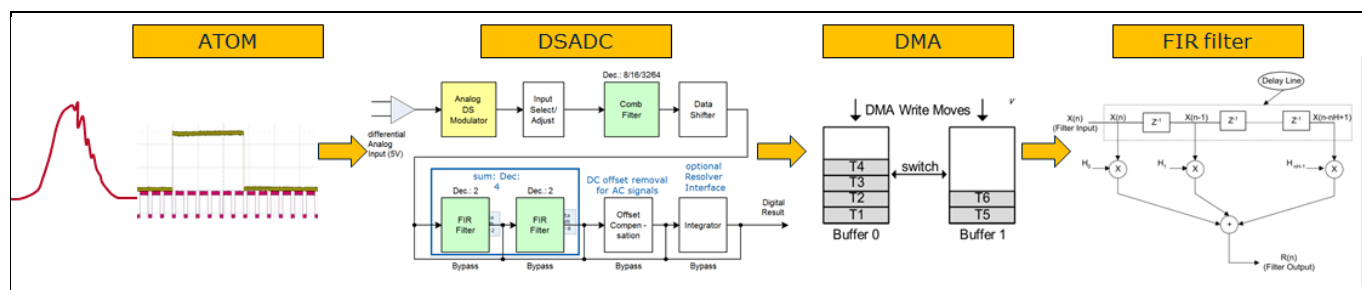
## 4 Code implementation

**Attention:** Code examples are provided *ONLY* for illustration purposes and are not intended to be 'production' ready. Any use of these code examples is entirely at your own risk.

### 4.1 Resources

The main resources to implement Knock detection in the code are as follows:

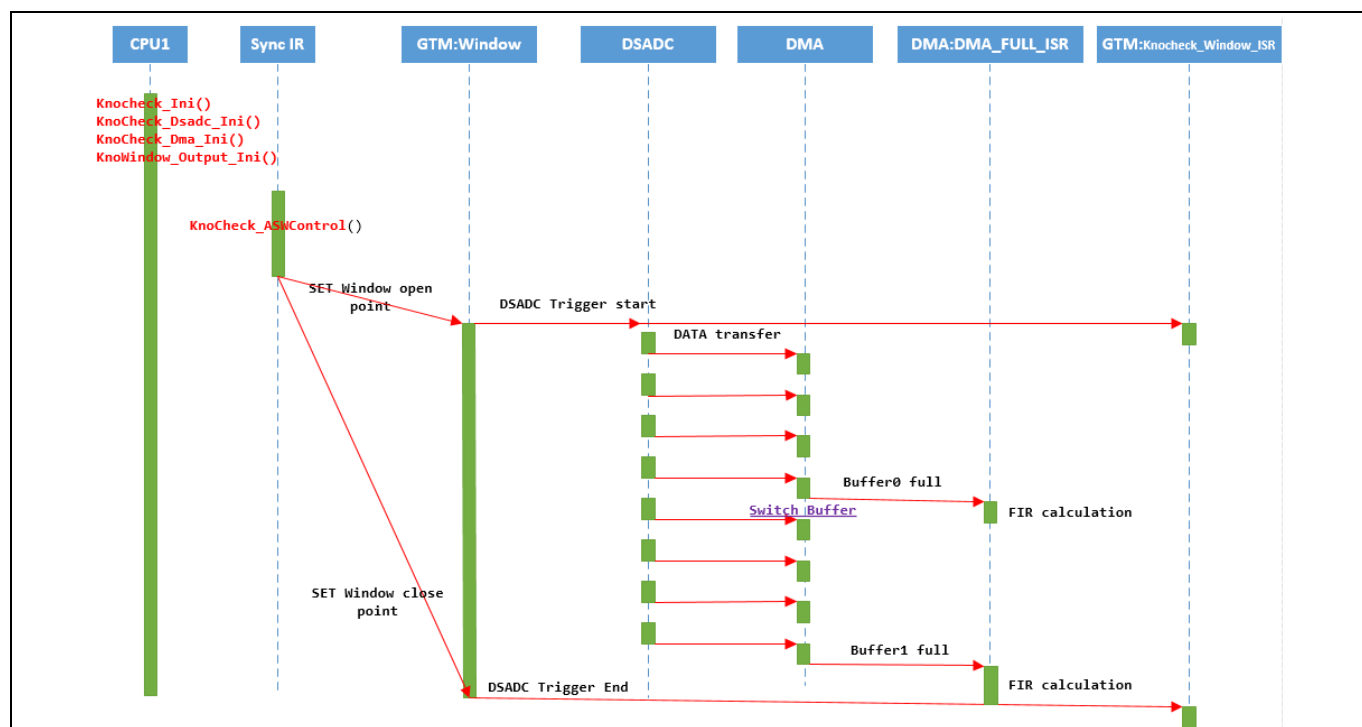
- ATOM2\_CH6 in GTM
- DSADC CH0
- DMA CH0
- Port pin 32.0 (Knock window check)
- FIR filter in DSP library



**Figure 11** Resources for knock detection

### 4.2 Code operation sequence

The overall system operation is as follows:



**Figure 12** Code operation sequence

### 4.2.1 Knock window

In the system the data gathering period is known as the 'knock window open' status. The knock window in the AURIX™ is implemented by ATOM in the GTM.

The ATOM module plays a role as a gate trigger. During the knock window open, DSADC converts the analog to digital data. The knock window open period, and the start and close points, are decided by the application (KnoCheck\_ASWControl API) and will depend on the internal atmosphere of the engine.

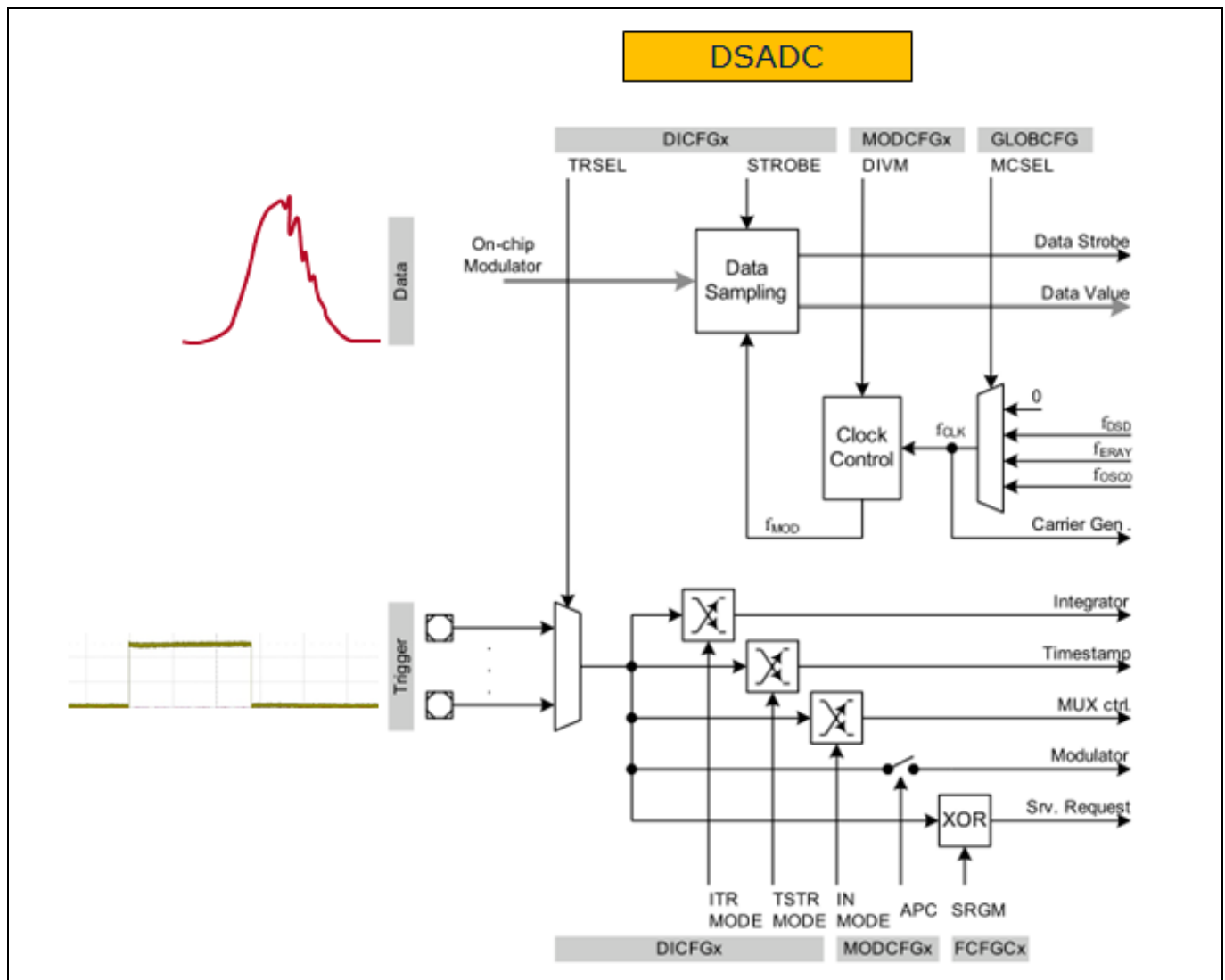


Figure 13 Knock window

### 4.3 DSADC configuration

Here the target output pass band is set as 104.17KHz, where  $F_{spb} = 100\text{MHz}$ .

To get the intended passband OSR (Oversampling Ratio) DSADC can be configured as shown in the following table.

The CIC (Cyclic Integration Comb) filter OSR is configurable by the user, but two FIR filters in DSADC have a fixed decimation rate with 2 respectively.

Divider Factor				8		
Comb Filter	FIR (8/28 9bit TAB)			Filter Input fs	Filter Output fd	Bandwidth (Pass-Band)
OSR COM	OSR FIR	SUM OSR		fs[MHz]	fd[kHz]	fpassmax[kHz]
8	4	32		12.50	390.63	130.21
10	4	40		12.50	312.50	104.17
20	4	80		12.50	156.25	52.08
32	4	128		12.50	97.66	32.55
34	4	136		12.50	91.91	30.64
64	4	256		12.50	48.83	16.28
128	4	512		12.50	24.41	8.14

Figure 14 Pass band configuration

#### 4.4 Group delay

Data that enters a digital filter needs a certain number of filter clocks before it appears at the filter's output, where it can be used by the system. The effective group delay depends on the configuration of the filter chain. The AURIX™ user manual provides a summary table with respect to group delay. When considering group delay, data which comes in to the DSADC from the 'knock window open' until meaningful data is generated, should be discarded.

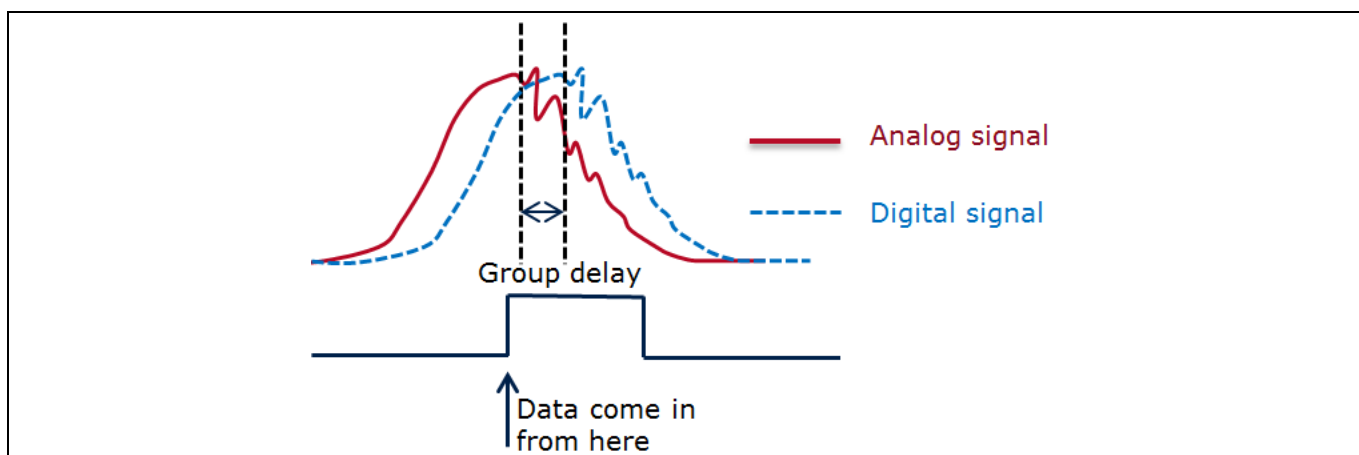


Figure 15 Group delay

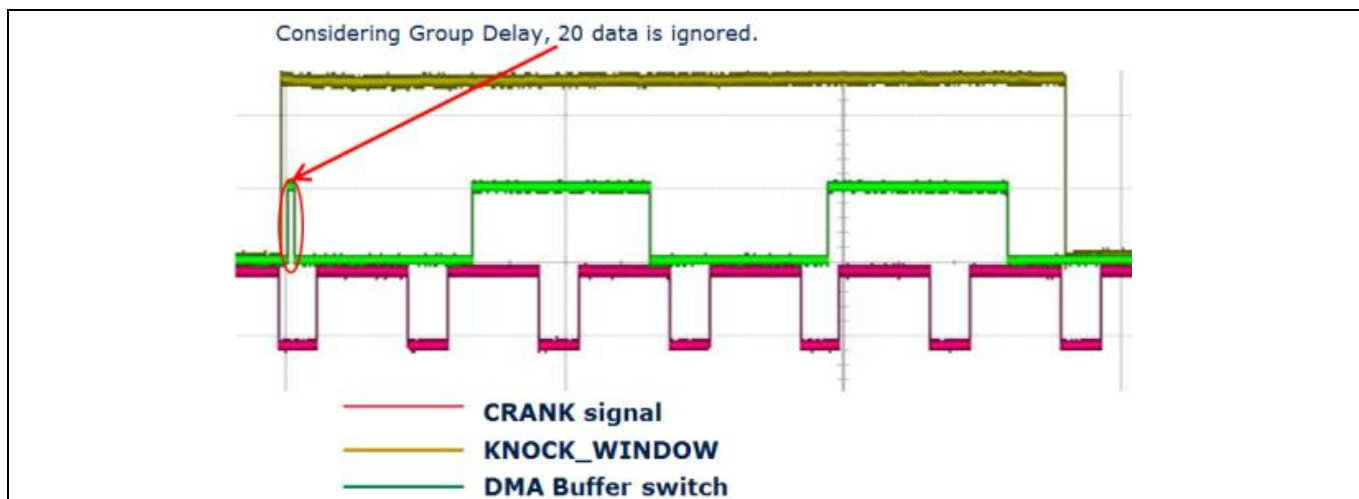


Figure 16 Group delay consideration in code

## 4.5 Offset compensation

Fundamentally DSADC has deterministic offset, as demonstrated in the following table:

Added Input Offset Value for	Differential inputs [mV]	Single ended inputs [mV]
Gain setting = 1:	150	150
Gain setting = 2:	75	75
Gain setting = 4:	37.5	37.5
Gain setting = 8:	16.25	16.25
Gain setting =16:	8.125	8.125

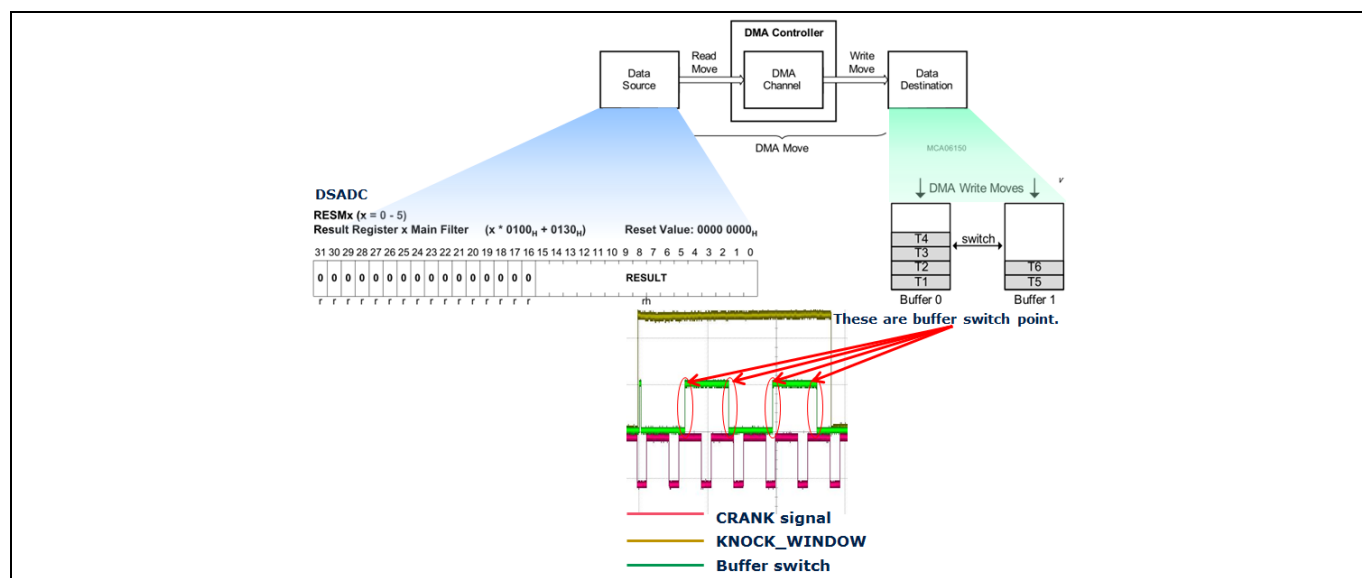
**Figure 17 DSADC internal offset (TC27x B and C steps)**

*Note: The value can vary +/-35. You must consider the offset when calibrating the offset value.*

*Note: The offset is different for each device step. For more specific information please refer to appropriate AURIX™ data specification for the device that you are using.*

## 4.6 DMA and double buffering to reduce CPU load

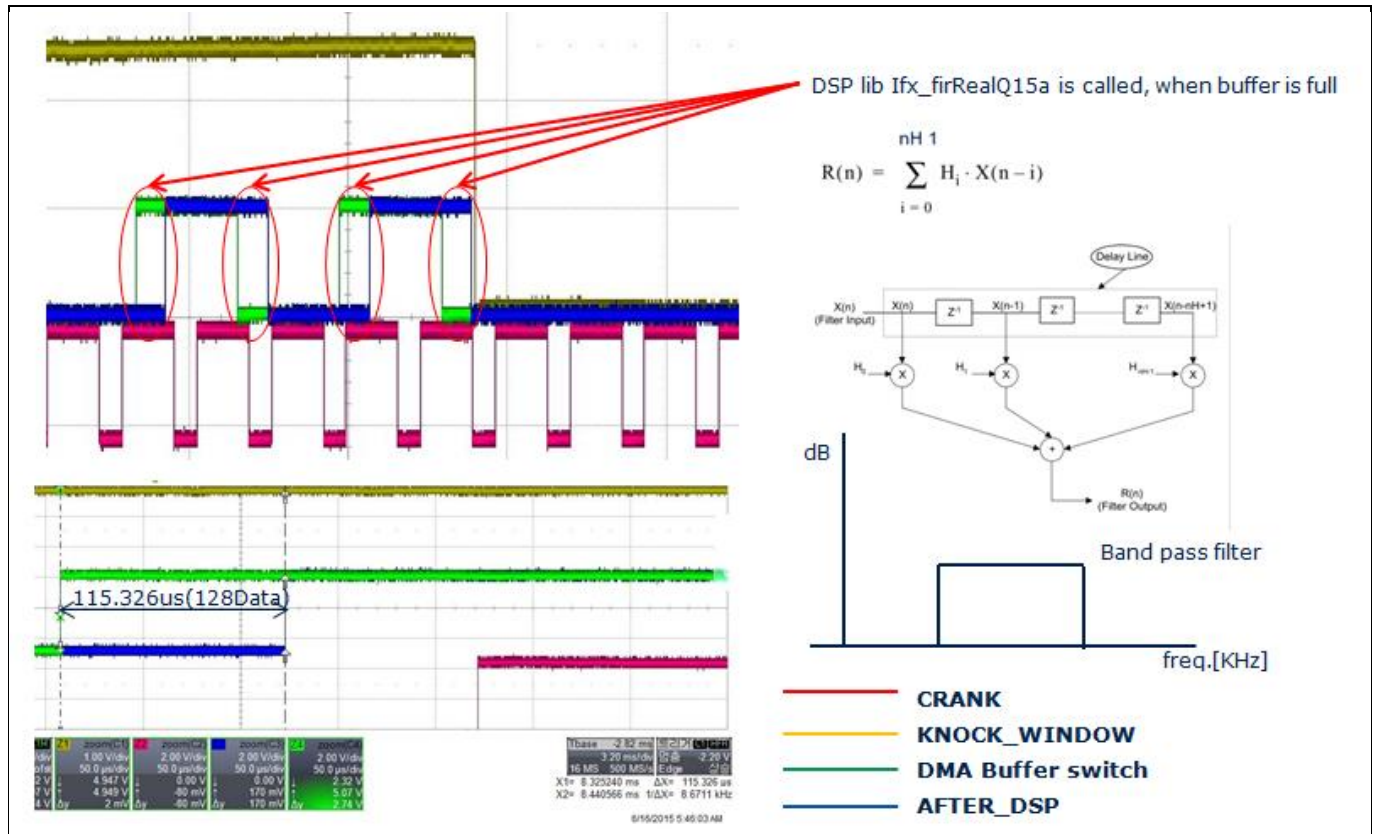
Digital Signal Processing arithmetic takes a long time. To distribute the CPU load, DSP calculations only occur after 128 bits of data have entered the RAM buffer. For effective data handling a 'double buffering' function inside the DMA is used. Double buffering provides automatic buffer change functionality and means that during the calculation phase, while working on the available set of data, more incoming data can still be safely stored.



**Figure 18 DMA double buffering concept**

## 4.7 Signal extract

FIR filter software is needed to get a target frequency signal. Infineon provides a DSP library. To extract the signal which has the target frequency the FIR filter API **Ifx\_firRealQ15a** is used. To use this API the filter should have a multiple of 8 coefficients. It takes 115.326us to calculate 128 bits of data (AURIX™ TC277).



**Figure 19** FIR filter calculation points

After extraction the signal which has the knock frequency is then manipulated.

- The signed signal is first converted to an absolute signal.
- The absolute signal is added to get an integration value.
- The more digitalized the signals are in pass band, the greater the integration value.

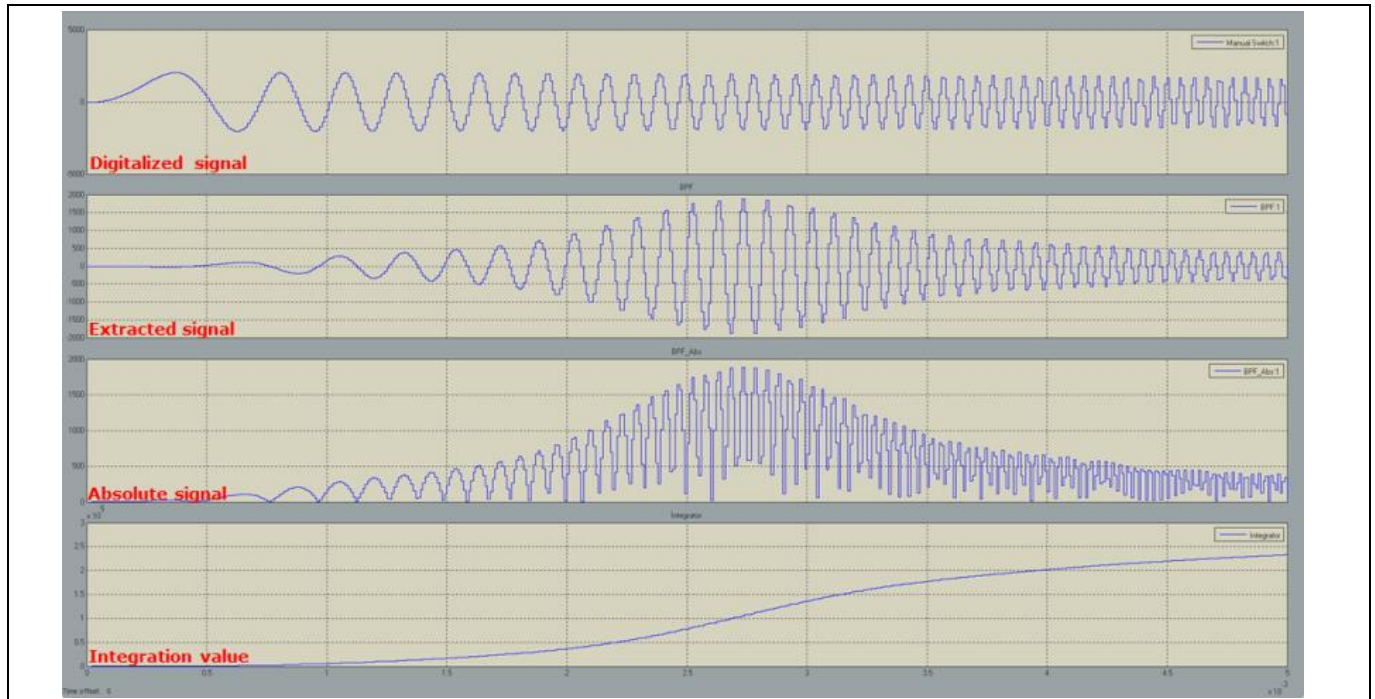


Figure 20 Integration value

## 5 Performance test

### 5.1 Input signal

A sine wave signal is used for the test.

According to the test, the sinwave frequencies are as follows:

- 11KHz sinewave (Analog to digital conversion validation test).
- 0~200Khz swept sinewave for 4s (DSADC cut-off frequency validation test).
- 0~104Khz swept sinewave for 7s (Signal extraction validation test).



**Figure 21** Test input signal

## 5.2 Analog to digital conversion validation test

As shown in the figure that follows, the sine signal is digitalized.

- A digitalized signal can be checked through the variable Buffer0\_Dest, Buffer1\_Dest.
- An 11 KHz signal is used for the test in this figure.

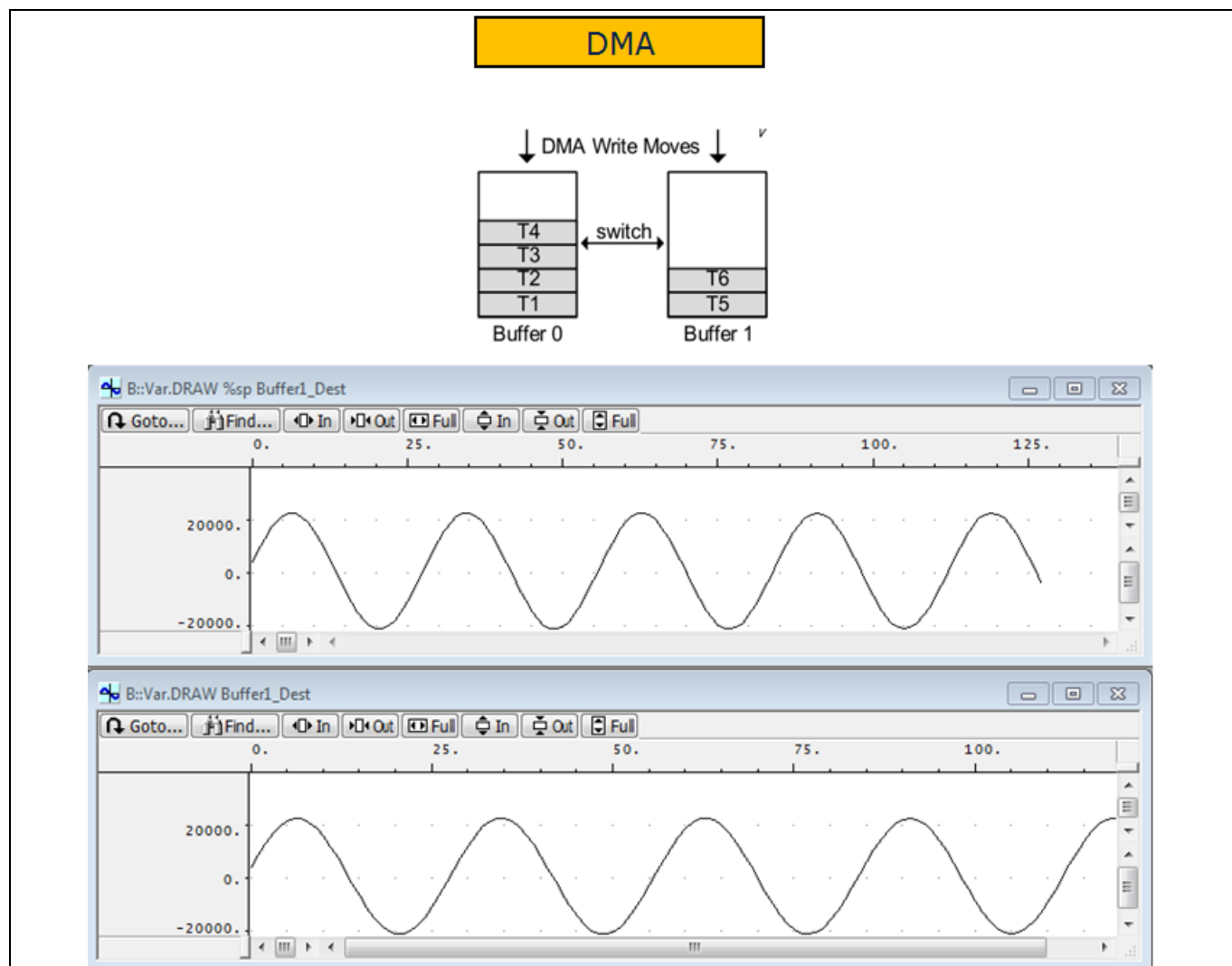


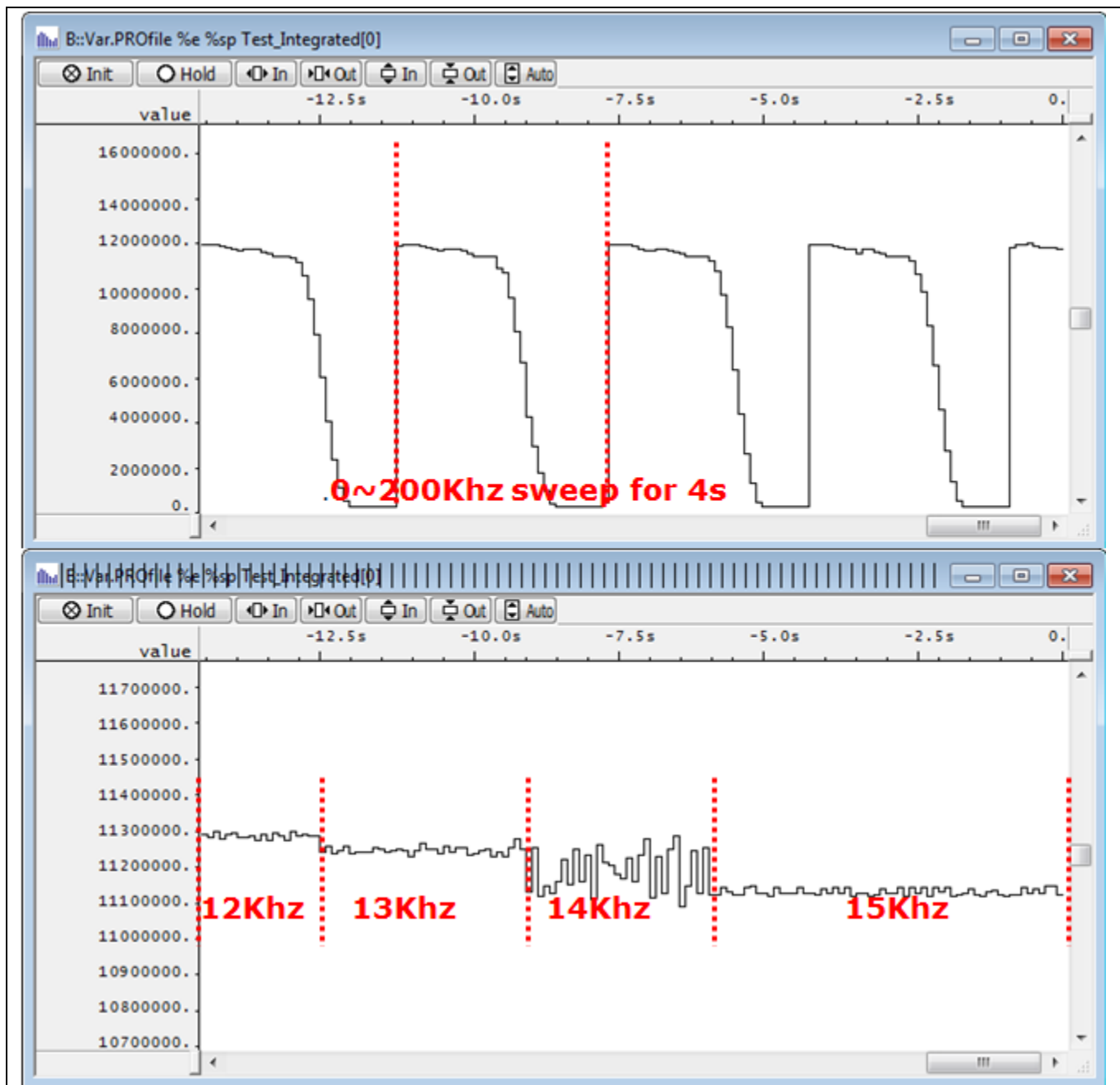
Figure 22 Analog to digital conversion validation test (Sinewave with 11KHz)



### 5.3 DSADC cut-off frequency validation test

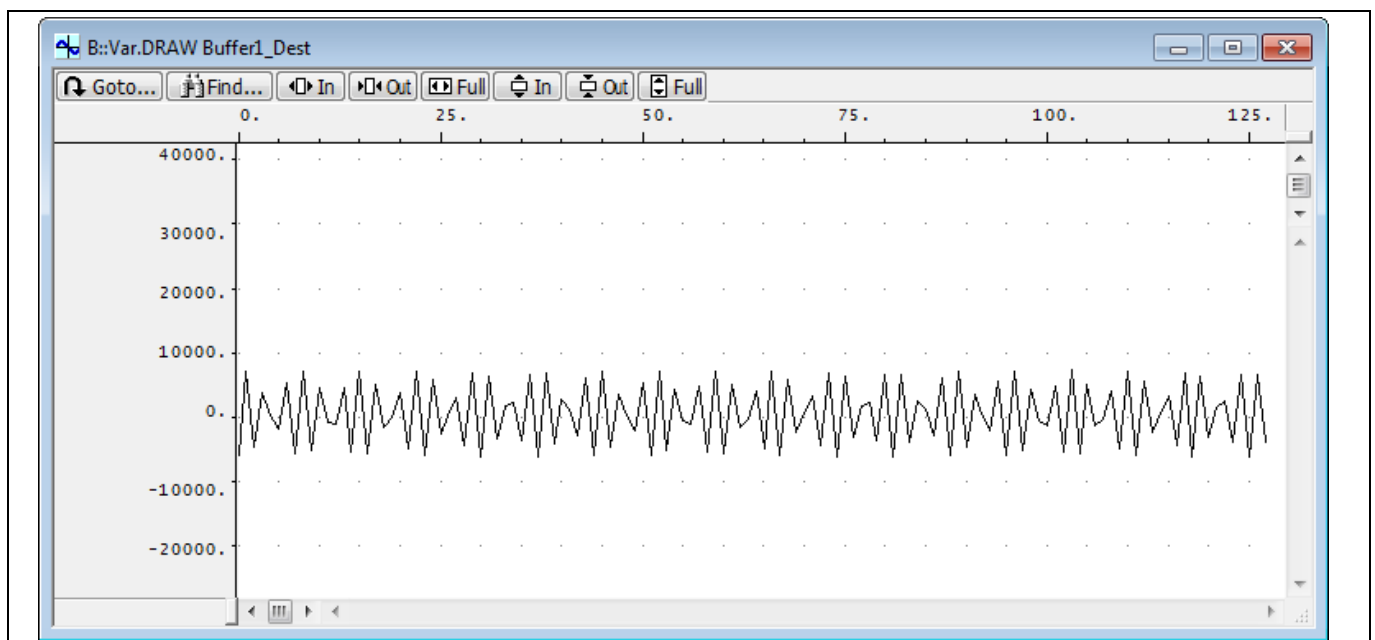
The target cut-off DSADC frequency is 104.17KHz in this system. The following figure shows the integration value.

- For the test, a sine-wave signal which sweeps from 0 to 200KHz for 4s is used.
- The cut-off frequency can be checked in the second picture below. The integration value dramatically drops down from around the 104KHz.



**Figure 23** Cut-off frequency validation test

The next figure shows a re-constructed sinewave signal which has 130 KHz. The re-constructed sinewave signal is very different from the original.



**Figure 24** Sinewave with 130KHz

## 5.4 Signal extraction validation test

For the test, a 30 KHz (+2KHz) Bandpass filter coefficient is used. The following figure shows the signal extraction result.

Note: The Filter type, frequency, and filtering range can all be changed based on the project.

Note: A 30 KHz sinewave signal has highest integration value.

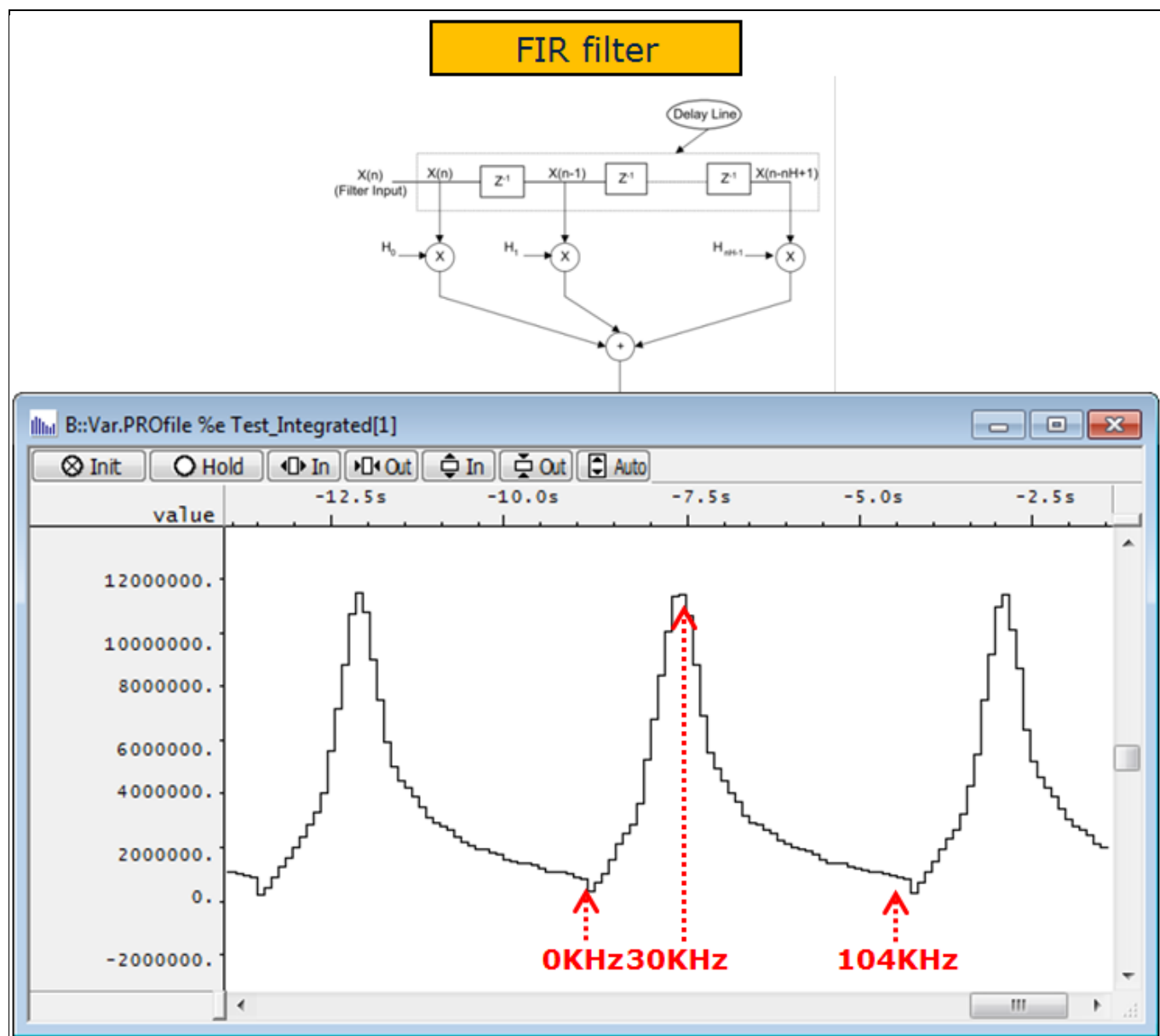


Figure 25 Signal extraction validation test

## 6 Check points

The operating situation can be easily checked through the available variables.

### Check points

- Buffer0\_Dest.
- Buffer1\_Dest.
- Test\_Integrated.
- Knocheck\_Integrated.

*Note: If you are using a T32 debugger the following commands are recommended:*

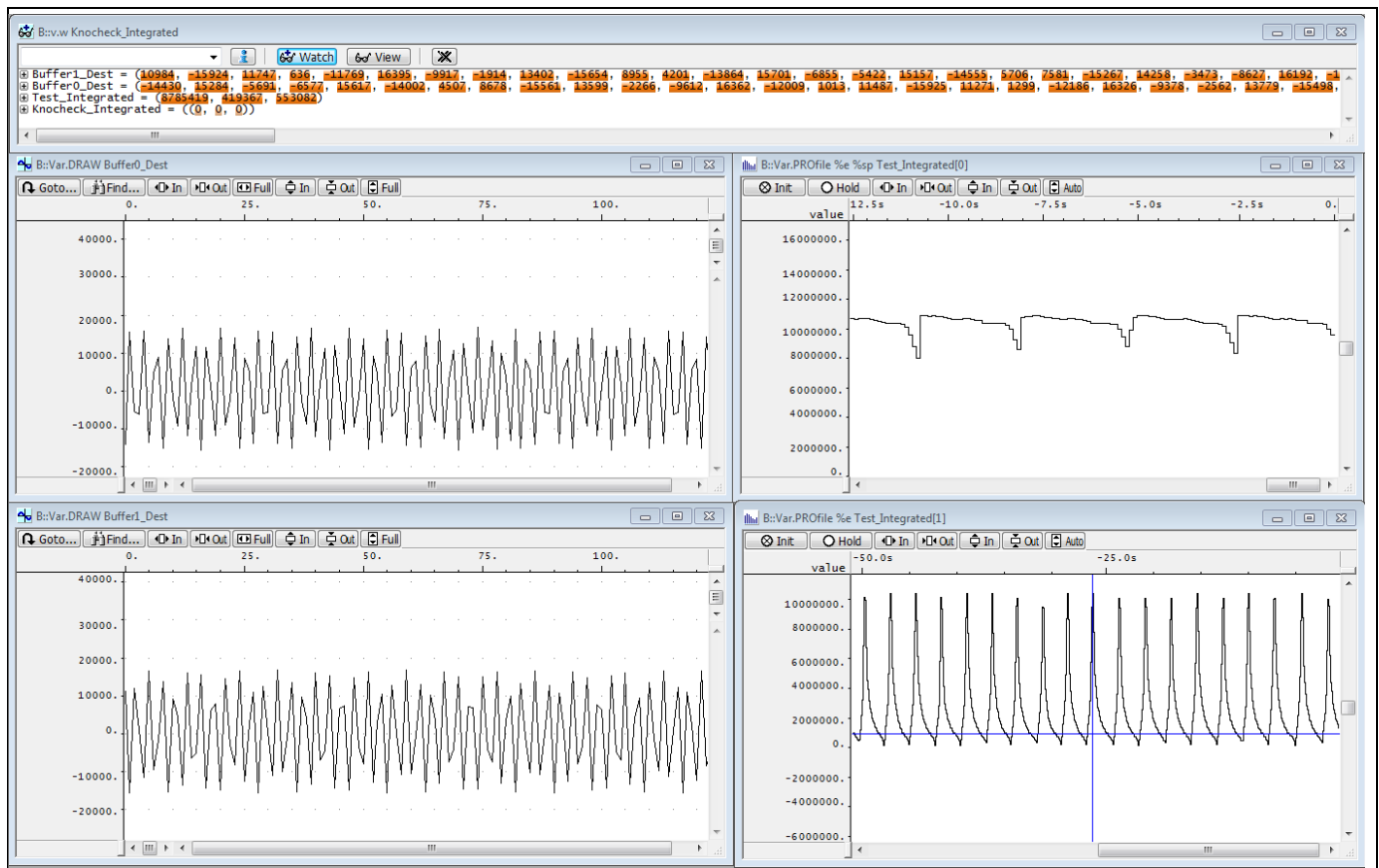
```
// v.w Buffer0_Dest Buffer1_Dest Test_Integrated Knocheck_Integrated
// v.Draw Buffer0_Dest
// v.Draw Buffer1_Dest
```

- The variables **Buffer0\_Dest** and **Buffer1\_Dest** show the digitalized knock signal.

```
// v.Draw Test_Integrated[0]
// v.Draw Test_Integrated[1]
```

- The variable **Test\_Integrated** holds the filter made by software and the integrated signal. The filter frequency is decided by the variable **FilterIndices[i]**.
- **Test\_Integrated[0]** holds the bypass, which means the software filter is not adjusted, and only the integrated process is applied in the following figure.
- **Test\_Integrated[1]** holds the bandpass filter which has 30KHz and integrated signal in the flowing figure.
  - For more detailed information please refer to the DSADC cut-off frequency validation test and the Signal extraction validation test sections.

## Check points



**Figure 26 Check point window**

## 7 Driver API

*Note: Because this document does not focus on the DPLL implementation, any comments only relate to the Sync point related API in the DPLL driver.*

**Table 1 DPLL driver API**

Function	Signature	Description
SyncPoint_TE3_ISR	IFX_INTERRUPT(SyncPoint_TE3_ISR, 0, TE3_INT_NUM)	Sync point is generated every 180°.

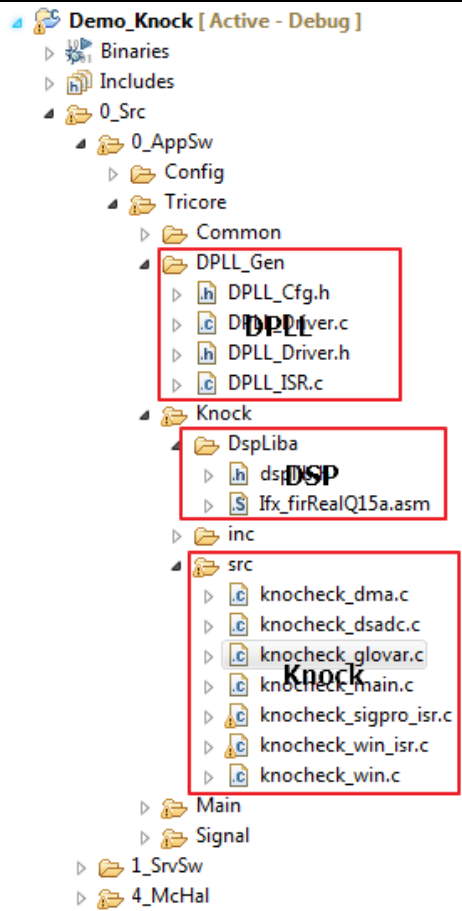
**Table 2 Knock driver API**

Function	Signature	Description
Knocheck_Ini	void Knocheck_Ini(void);	Initialize Knock window, gain, Integration value.
KnoCheck_Dma_Ini	void KnoCheck_Dma_Ini(void);	Initialize Konck related DMA function
KnoCheck_Dsadc_Ini	void KnoCheck_Dsadc_Ini(void);	Initialize Konck related DSADC function
KnoCheck_ASWControl	void KnoCheck_ASWControl(void)	Called at Every Sync position. Set the gain and Knock window open ,close position
KnoCheck_Filter	void KnoCheck_Filter(void);	Calculate FIR filter
KnoCheck_Memcopy	void KnoCheck_Memcopy(void);	Copy the DMA buffer value to the Ram region
KnoCheck_ClearCoeffDly	void KnoCheck_ClearCoeffDly(void);	After integration, clear the buffer in FIR filter
KnoCheck_MemCopy64	void KnoCheck_MemCopy64(uint64* xDest_pu64, uint64* xSrc_pcu64, uint16 numBytes);	Copy the DMA buffer value to the Ram region
knoCheck_GetIntegData	void knoCheck_GetIntegData(uint8 Ch, sint32* IntegratedValues, uint8 ValueLength);	Provide Integration result value

## 8 Driver files and folder structure

An outline of the driver file structure:

**Table 3 Folder structure**

Folder structure	Module	Description
	DPLL	Initialize and operate DPLL functionality in GTM. This module only covers simple DPLL function.
	DspLiba	Infineon DSP Library
	Knock	Knock function code.

## 9 References

- [1] AP32343\_Position\_Minus\_Time\_Request\_PMTR
- [2] AP32222\_DSADC\_basics
- [3] AP32015\_Engine Knock detection using TC-1796
- [4] <http://wolfcrow.com/blog/professor-samplers-notes-aliasing>
- [5] [https://en.wikipedia.org/wiki/Nyquist%E2%80%93Shannon\\_sampling\\_theorem](https://en.wikipedia.org/wiki/Nyquist%E2%80%93Shannon_sampling_theorem)



## Revision history

### Major changes since the last revision

Page or Reference	Description of change
V1.0	First release

#### Trademarks of Infineon Technologies AG

μHVIC™, μIPM™, μPFC™, AU-ConvertIR™, AURIX™, C166™, CanPAK™, CIPOS™, CIPURSE™, CoolDP™, CoolGaN™, COOLiR™, CoolMOS™, CoolSET™, CoolSiC™, DAVE™, DI-POL™, DirectFET™, DrBlade™, EasyPIM™, EconoBRIDGE™, EconoDUAL™, EconoPACK™, EconoPIM™, EiceDRIVER™, eupec™, FCOS™, GaNpowIR™, HEXFET™, HITFET™, HybridPACK™, iMOTION™, IRAM™, ISOFACE™, IsoPACK™, LEDriviR™, LITIX™, MIPAQ™, ModSTACK™, my-d™, NovalithiC™, OPTIGA™, OptiMOS™, ORIGA™, PowIRaudio™, PowIRstage™, PrimePACK™, PrimeSTACK™, PROFET™, PRO-SIL™, RASIC™, REAL3™, SmartLEWIS™, SOLID FLASH™, SPOC™, StrongIRFET™, SuplRBuck™, TEMPFET™, TRENCHSTOP™, TriCore™, UHVIC™, XHP™, XMC™

Trademarks updated November 2015

#### Other Trademarks

All referenced product or service names and trademarks are the property of their respective owners.

**Edition 2016-09-05**

**Published by**

**Infineon Technologies AG**

**81726 Munich, Germany**

**© 2016 Infineon Technologies AG.**

**All Rights Reserved.**

**Do you have a question about this document?**

**Email: [erratum@infineon.com](mailto:erratum@infineon.com)**

**Document reference**

**AP32348**

#### IMPORTANT NOTICE

The information contained in this application note is given as a hint for the implementation of the product only and shall in no event be regarded as a description or warranty of a certain functionality, condition or quality of the product. Before implementation of the product, the recipient of this application note must verify any function and other technical information given herein in the real application. Infineon Technologies hereby disclaims any and all warranties and liabilities of any kind (including without limitation warranties of non-infringement of intellectual property rights of any third party) with respect to any and all information given in this application note.

The data contained in this document is exclusively intended for technically trained staff. It is the responsibility of customer's technical departments to evaluate the suitability of the product for the intended application and the completeness of the product information given in this document with respect to such application.

For further information on the product, technology, delivery terms and conditions and prices please contact your nearest Infineon Technologies office ([www.infineon.com](http://www.infineon.com)).

#### WARNINGS

Due to technical requirements products may contain dangerous substances. For information on the types in question please contact your nearest Infineon Technologies office.

Except as otherwise explicitly approved by Infineon Technologies in a written document signed by authorized representatives of Infineon Technologies, Infineon Technologies' products may not be used in any applications where a failure of the product or any consequences of the use thereof can reasonably be expected to result in personal injury.