Freescale Semiconductor
Application Note

Document Number: AN4209
Rev. 1, 09/2011

Knock Filter

by: Mong Sim
System and Application
Austin
USA

Contents

1 Introduction

A knock sensor is comprised of piezoelectric materials and when impacted generates an electrical signal. The knock sensor monitors the engine's operation to optimize performance. In addition, a knock sensor also protects the engine against power robbing and a potentially destructive engine knock. A computer monitors this electrical signal for irregularity and corrects the timing in the variable valve-timing engine. This ensures the engine operates to it's optimal capacity.

The knock sensor also picks up other vibrations in the system that generate electrical signals corresponding to these vibrations which are undesirable. To filter these unwanted signals and allow a certain spectrum of frequencies to pass through, a special filter circuitry called a Knock Filter is designed. The following sections explain how to design a Knock Filter using the MPC5674F Freescale automotive grade Power Architecture microcontroller (MCU).

Described in the following sections the relevant functions of the individual sub block and how these sub blocks come together to form the Knock Filter design.

Throughout this application note C programming code is used to explain some of the register settings and other programmatic requirements. A basic understanding of C programming language is needed.

2 Knock Filter

In this block diagram Figure 1 has four external components. For the purpose of better understanding and implementation, icons were added to the knock sensor and a signal conditional device. For this application note, both these components were replaced with a function generator with frequency sweeping capability.

In Figure 1 the input signal is sampled by the Analog-to-Digital Converter (ADC). The output from the ADC is processed by the Decimation Filters, the results are then stored in the result FIFO (RFIFO) and retrieved by the Central Processing Unit (CPU) for further processing. The Enhanced Modular Input/Output Subsystem (EMIOS) and the Direct Memory Access (DMA) provide a sampling clock and a command to the Enhanced Queued A/D Converter (EQADC). Three components were added after the CPU to provide visual contact of the Decimation Filter B outputs:

- Deserial/ Serial Peripheral Interface (DSPI)
- Digital-to-Analog Converter (DAC)
- Oscilloscope

In the input signal stage, the EQADC can now be started.

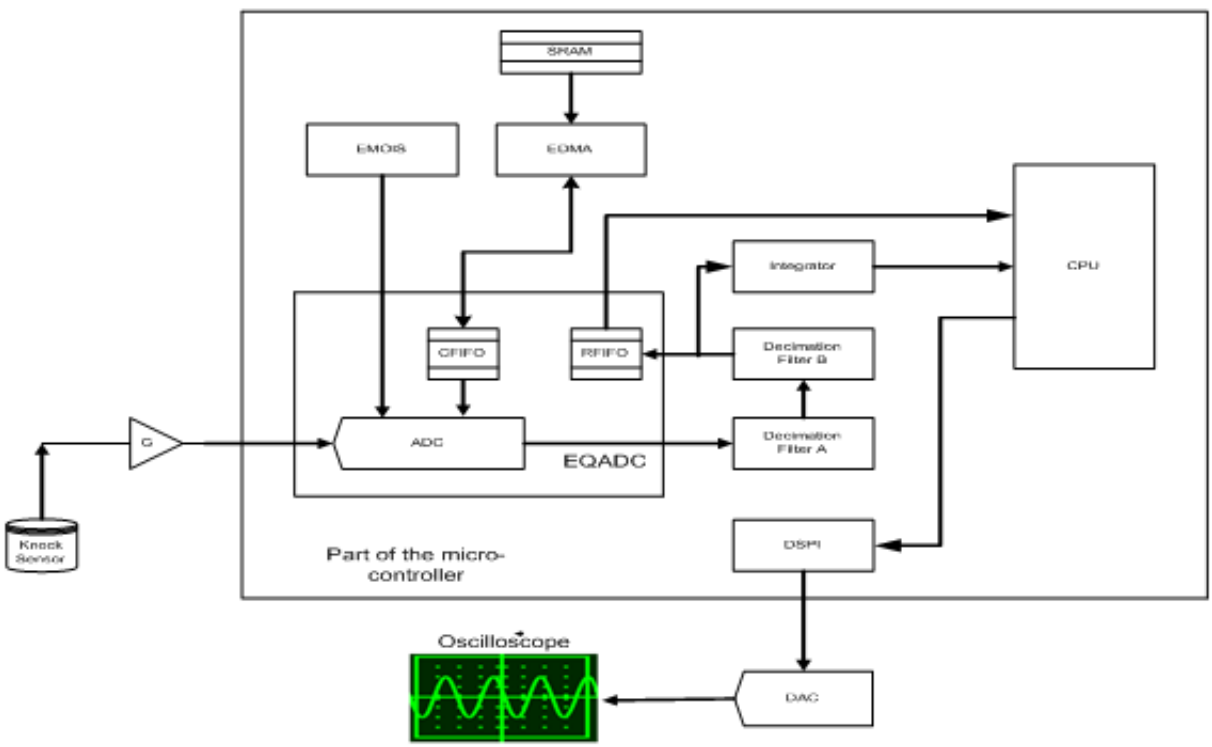


Figure 1. Knock filter anatomy

2.1 Enhanced Queued Analog-to-Digital Converter (EQADC) Stage

The Enhanced Queued Analog-to-Digital Converter (EQADC) module provides fast and accurate conversions for a wide range of applications. There are two EQADC modules on the MPC5674F, the EQADC_A and EQADC_B. Each EQADC module provides a parallel interface to two on-chip independent ADC cores. The EQADC_B has a hardware interface to seven of the eight decimation filter blocks on the MPC5674F. This allows transferring of conversion and filtered values to and from the EQADC_B and decimation filters without CPU or DMA interaction.

The EQADC_B was used because it has better support for the Decimation Filters. In this stage, automate the EQADC_B to sample the input signals and transfer the data to the decimation filter A input without any software overhead.

These are the configuration steps:

1. Invalidate all entities of CFIFO1

   This step makes sure the Command First-In-First-Out memory is clear from any residue that would affect the EQADC operation. Assert the CFINV bit in the Control Register 1 to clear the CFIFO1

   ```
   // invalidate all entries of CFIFO (26.6.2.5 EQADC CFIFO Control Registers (EQADC_CFCR))
   EQADC_B.CFCR[1].B.CFINV = 0x01;
   ```
2. Set raising edge triggering and continuous scanning mode

   Use the EMIOS as an external triggering source, and tell the EQADC how to respond to the triggering signal. To enable this feature, set the appropriate bits in the MODE field of the EQADC Control Register 1.

   ```
   // rising edge trigger, continuous scan (26.6.2.5 EQADC CFIFO Control Registers
   (EQADC_CFCR))
   EQADC_B.CFCR[1].B.MODE = 0x0D;
   ```
3. Select DMA transfer

   To transfer the command to the EQADC without using any CPU resources, use the DMA to transfer the command from a RAM location to the CFIFO1. Assert the field CFFE and CFFS in the EQADC Interrupt and DMA Control Register 1 enables the DMA feature.

   ```
   // enable INT/DMA on CFFF=1 (26.6.2.6 EQADC Interrupt and DMA Control Registers
   (EQADC_IDCR))
   EQADC_B.IDCR[1].B.CFFE = 0x01;

   // select DMA transfer when CFFF sets
   EQADC_B.IDCR[1].B.CFFS = 0x01;
   ```
4. Configure the ADC

   There are two ADC in an EQADC module. In this experiment, choose ADC_1 to provide the analog to digital conversion.

   ```
   // ADC1_CR: ADC1_EN=1, ADC CLK=Sys CLK, Sys clock div by 2
   EQADC_B.CFPR[1].R = 0x00800001;
   ```
5. Configure the destination for the ADC results

   The ADC results must be right adjusted signed and then send the results to the Decimation Filter A input.

   ```
   // ADC_ACR1: DEST=1 for DECFIL A, FMTA=1 for signed format
   //
   EQADC_B.CFPR[1].R = 0x00060030;
   ```

   In this section, there are some fragments of C codes. These codes are used to set the fields of the respective registers. These registers are in the reference manual *MPC5674F Microcontroller Reference Manual* (document number MPC5674FRM) Rev. 3 . In the subsequent sections, this format is followed.

2.2 Enhanced Modular Input and Output System (EMIOS) Stage

The eMIOS200 provides a functionality to generate or measure time events. It has its own configuration of timer channels suitable for a target application. The MPC5674F has one eMIOS200 module.

Before using the EMIOS to trigger the EQADC, connect the selected EMIOS outputs to the EQADC using the System Integration Unit (SIU).

Here are the configuration steps:

1. Set the SIU Pad Configuration (PCR) register pin190 to its primary function

The SIU Pad Configuration (PCR) register is set to its primary function to support the EMIOS11.

```
//configure pin 190 (EMIOS11) for output. (Table 6-22. SIU_PCRn Settings)
// this is the trigger signal used by eQADC_B
SIU.PCR[190].B.PA = 1;
```

2. Make the connection between the EMIOS and the EQADC.

   The EMIOS trigger signal is connected to the EQADC using the SIU Internal Multiplexer (IMUX) select registers.

```
// choose EMIOS 11 for eQADC_B 6.3.1.18 (eQADC Command FIFO Trigger Source Select -
IMUX Select Registers)
(SIU_ISEL[4-7])
SIU.ISEL7.B.CTSEL1_1 = 0x40;
```

3. Configure the EMIOS

   Before using the EMIOS, set the global time base and enable the global prescaler in the eMIOS200 Module Configuration register that has to be set to enable the clock and the enabling state.

```
// required - enable internal counters (22.3.2.1 eMIOS200 Module Configuration Register
(EMIOS_MCR))
EMIOS.MCR.B.GTBE = 1;

// required - enable global prescaler
EMIOS.MCR.B.GPREN = 1;
```

4. Setup an EMIOS channel

   Channel 11 is the triggering source and it has to be configured. Assert the field UCPREN and set the field MODE to 0 x 18 in the Channel 11 Control Register. The UCPEN field enables the prescaler and while in MODE field, select the PWM for the application.

```
// required, even though using div by one (22.3.2.7 eMIOS200 Control Register
(EMIOS_CCR[n]))
EMIOS.CH[11].CCR.B.UCPREN = 1;

// select PWM
EMIOS.CH[11].CCR.B.MODE=0x18;
```

   Next, define the duty cycle and sampling frequency of this triggering signal in the EMIOS A and B registers. The sampling frequency is set to 50 KHz and the duty cycle must be non-zero and less than 0x500.

```
// 128 MHz system clock, use 2560 (0x500) to get 50 kHz EMIOS frequency
// fSYS/fEMIOS (22.3.2.5 eMIOS200 B Register (EMIOS_CBDR[n]), 22.3.2.4 eMIOS200 A
Register (EMIOS_CADR[n]))
EMIOS.CH[11].CBDR.R = 0x500;

// must be non-zero less than CBDR, 0x100 wide enough to be seen on scope
EMIOS.CH[11].CADR.R = 0x100;
```

   The triggering mechanism is set. Finally , setup the DMA to fetch a command from a defined SRAM location to the CFIFO.

2.3 Enhanced Direct Memory Access (EDMA) Stage

The MPC5674F consists of two Enhanced Direct Memory Access controller (EDMA) blocks, EDMA_A and EDMA_B, respectively. The EDMA is a second-generation platform block capable of performing complex data movements through N programmable channels (N=64 for EDMA_A, N=32 for EDMA_B), with minimal to no intervention from the host processor. The hardware micro-architecture includes a DMA engine that performs source and destination address calculations, and actual data movement operations along with an SRAM-based memory containing the transfer control descriptors (TCD) for the channels.

In this implementation, the EDMA fetches a command stored in an internal Static Random Access Memory (SRAM) location. Store the command to the EQADC CFIFO without CPU intervention.

Here are the configuration steps:

1. Define the source
   Define the source, a 32-bit long word, in the MPC5674F internal SRAM. This 32-bit long word is transfered to the destination address EQADC CFIFO by the DMA.

```
// 8000_0000:  EOQ=1: for use in EQADC continuous-scan edge trigger mode
// 0010_0000:  MESSAGE_TAG=0001: to select conversion result into RFIFO1
// 0000_0000:  CHANNEL_NUMBER=0000: select the analog input channel
// 0000_0008:  ALT_CONFIG_SEL=08: choose alternate configuration #1
//  --------------
// 8010_0008 (26.7.2.2.1 Message Formats for On-Chip ADC Operation)

vuint32_t adc_conv_command = 0x80100008;
```

2. Define the destination

   In the EQADC stage define the CFIFO1 as the destination storage location of the DMA transfers. See Enhanced Queued Analog-to-Digital Converter (EQADC) Stage.

3. Setup the transfer control descriptor of the EDMA

   This experiment uses DMA Channel 2. The DMA is setup to transfer the 32-bit long EQADC source command word to the the destination EQADC CFIFO.

   Here are the configuration steps:

```
// source, EQADC command
EDMA_B.TCD[2].SADDR = (vuint32_t) &adc_conv_command;

//destination, CFIFO
EDMA_B.TCD[2].DADDR = (vuint32_t) &(EQADC_B.CFPR[1].R);

   EDMA_B.TCD[2].SSIZE = 2;          // source = 32 bits
   EDMA_B.TCD[2].DSIZE = 2;          // dest = 32 bits
   EDMA_B.TCD[2].NBYTES = 4;         // number of bytes
   EDMA_B.TCD[2].CITER = 1;
   EDMA_B.TCD[2].BITER = 1;

   EDMA_B.TCD[2].SMOD = 0;
   EDMA_B.TCD[2].DMOD = 0;
   EDMA_B.TCD[2].SOFF = 0;           // soffset = 0, don't move
   EDMA_B.TCD[2].SLAST = 0;          // source dest adjustment, don't move
   EDMA_B.TCD[2].CITERE_LINK = 0;
   EDMA_B.TCD[2].DOFF = 0;           // doffset = 0, don't move
   EDMA_B.TCD[2].DLAST_SGA = 0;
   EDMA_B.TCD[2].BITERE_LINK = 0;

   EDMA_B.TCD[2].BWC= 0;             // bandwidth control
   EDMA_B.TCD[2].MAJORLINKCH= 0;     // enable channel-to-channel link
   EDMA_B.TCD[2].MAJORE_LINK= 0;     // enable channel-to-channel link
   EDMA_B.TCD[2].E_SG= 0;            // enable scatter/gather descriptor
   EDMA_B.TCD[2].D_REQ= 0;           // disable ipd_req when done
   EDMA_B.TCD[2].INT_HALF= 0;        // interrupt on citer = (biter >> 1)
   EDMA_B.TCD[2].INT_MAJ= 0;         // interrupt on major loop completion

   EDMA_B.SERQR.R = 2;               // enable DMA requests for CH 2
```

The DMA setup is finished and ready.

2.4 Decimation Stage

There are eight independent Decimation Filter blocks (A through H) on the MPC5674F device. Each Decimation Filter block contains a multiply-accumulate (MAC) unit capable of implementing a 16-bit, fourth order IIR, or eighth order FIR filter. The Decimation Filter blocks can be cascaded to create higher order filters.

Each Decimation Filter has a 32-bit integrator unit which allows the block to accumulate a series of filter outputs. The integrator input has a hardware multiplexer to select receiving data at the input or output of the decimator filter. The integrator also supports software triggering using the CPU or hardware triggering using the EMIOS, EMIOS and PIT and ETPU. The SIU provides two Decimation Filter Register, the SIU_DECFIL1 and SIU_DECFIL2 to control these triggering modes.

All Decimation Filters support DMA writes to the input data register, and the DMA reads from the output data register. The EQADC_B block has an internal hardware link to seven Decimation Filters. This allows CPU independent transfers of the EQADC_B analog-to-digital conversion result data to the Decimation filter input data registers, and filter output data back to the EQADC_B conversion result FIFOs.

In this stage, two decimation filter blocks are cascaded, A and B respectively using the IIR configuration to implement a Band Pass Filter (BPF).Here is how to configure each of the decimation blocks, how to cascade the two decimation filter blocks, and how to enable this cascade filter to perform.

These are the configuration steps:

1. Design the Band Pass Filter

   The BPF uses a fourth order High Pass IIR filter to cascade with a fourth order Low Pass IIR filter. You can use any off-the-shelf filter design software to get the coefficients for the HPF and LPF.

   These are the specification of the HPF and the LPF.

   The High Pass Filter characteristics as follows:

   Butterworth IIR filter
   Filter type— High Pass
   Pass band— 3125 – higher
   Order— 4
   Filter A

   The Low Pass Filter characteristics as follows:

   Butterworth IIR filter
   Filter type— Low Pass
   Pass band: 0 – 6250 Hz
   Order— 4
   Filter B

   You can choose a different approach to design an eighth order or higher order IIR filter system by cascading two or more Decimation Filter blocks together. The Decimation Filter block provides a Direct-Form 1 fourth order IIR filter design. If you design an eighth order or a higher IIR filter system by using the cascade mode. You have to break the prototype filter into two or more Direct-Form 1 fourth order sections.

   Choose a filter prototype system and design an eighth order IIR BPF. Next, break the eighth order filter into four second order sections. Finally, recombine the second order sections 1 and 2, and 3 and 4 to form two fourth order sections using convolution computation. Now, you have an eighth order IIR filter in cascade form.

2. Configure Decimation Filter A and B

   Appended is the procedure for configuring the decimation filters.

```
// disable decimation filter input
DECFIL_A.MCR.B.IDIS = 1;
DECFIL_B.MCR.B.IDIS = 1;

// tail filter output
DECFIL_B.MCR.B.CASCD = 2;
```

```
// head filter input
DECFIL_A.MCR.B.CASCD = 1;

// sample rate
DECFIL_B.MCR.B.DEC_RATE= 0;
DECFIL_A.MCR.B.DEC_RATE= 0;

// IIR 4th order filter
DECFIL_A.MCR.B.FTYPE = 1;
// enable saturation when underflow/overflow occurs
DECFIL_A.MCR.B.SAT = 0;

// IIR 4th order filter
DECFIL_B.MCR.B.FTYPE = 1;
// enable saturation when underflow/overflow occurs
DECFIL_B.MCR.B.SAT = 0;

// Set the HPF coefficient to the decimation A coefficient registers
DECFIL_A.COEF[0].R= 0x1314E8;
DECFIL_A.COEF[1].R= 0xB3AC5C;
DECFIL_A.COEF[2].R= 0x727D74;
DECFIL_A.COEF[3].R= 0xB3AC5C;
DECFIL_A.COEF[4].R= 0x1314E8;
DECFIL_A.COEF[5].R= 0x5F424E;
DECFIL_A.COEF[6].R= 0x927C70;
DECFIL_A.COEF[7].R= 0x3927C9;
DECFIL_A.COEF[8].R= 0xF49F1C;

// Set the LPF coefficient to the decimation B coefficient registers
DECFIL_B.COEF[0].R= 0x0053A2;
DECFIL_B.COEF[1].R= 0x014E8B;
DECFIL_B.COEF[2].R= 0x01F5D1;
DECFIL_B.COEF[3].R= 0x014E8B;
DECFIL_B.COEF[4].R= 0x0053A2;
DECFIL_B.COEF[5].R= 0x3EFD5C;
DECFIL_B.COEF[6].R= 0xC873D4;
DECFIL_B.COEF[7].R= 0x172EDD;
DECFIL_B.COEF[8].R= 0xFC25C4;

// force reset condition
DECFIL_B.MXCR.R           = 0;
// enable integrator
DECFIL_B.MXCR.B.SENSEL    = 1;
// software reset integrator
DECFIL_B.MXCR.B.SZRO      = 1;
// integrator signal operation selection
DECFIL_B.MXCR.B.SSIG      = 0;
// saturation
DECFIL_B.MXCR.B.SSAT      = 1;

// scaling factor: 0=1; 1=4; 2=8; 3=16
DECFIL_A.MCR.B.SCAL = 1;
DECFIL_B.MCR.B.SCAL = 1;

// enable decimation filter input
DECFIL_B.MCR.B.IDIS = 0;
DECFIL_A.MCR.B.IDIS = 0;

// execute soft reset to initialize decimation filter interface
DECFIL_A.MCR.B.SRES = 1;
DECFIL_B.MCR.B.SRES = 1;
```

The order of disabling and enabling of filters is important. When disabling the filter, always disable the filter input from the head filter to the tail filter. Conversely, when enabling the filters, always enable the filter input from the tail filter to the head filter.

3. Cascading Decimation Filter

The information in this section is useful for cascading more than two Decimation Filter blocks to form a higher order filter system. The MPC5674F has a simple connection rule; all the cascading blocks must be in sequential order. See Figure 2.

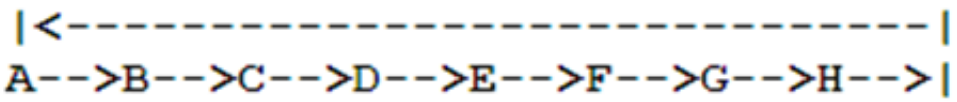


Figure 2. Sequential order

For example, if you decide to use three Decimation Filter blocks for a filter design, choose any Filter A — H as the head filter. However, the subsequence filters must be in sequence with the first filter chosen. For example, if the you choose filter A as the head filter, then the middle filter is B. However, if you choose filter G as the head filter, then the middle filter is H. For these two examples, the tail filters are C and A, respectively.

NOTE

Decimation Filter H is only accessible by the CPU and not accessible by the EQAC_B, therefore when cascading decimation filters do not use filter H as the head filter or the tail filter.

2.5 Integrator Stage

The integrator can operate in window mode controlled by the software or hardware, or operate in a continuous mode. Additionally, the integrator is configurable to support different output modes such as saturate, absolute, and signed. These features are controlled by the Decimation Filter Module Extended Configuration Register (DECFILT_X_MXCR).

Configure the integrator to accumulate outputs from the Decimation Filter block B.

Here are the configuration steps:

1. Configure the Integrator Input

   The Integrator is configurable to accept data from the Decimation Filter Block input or output. Configure the integrator to accumulate the results of Decimation filter B output.

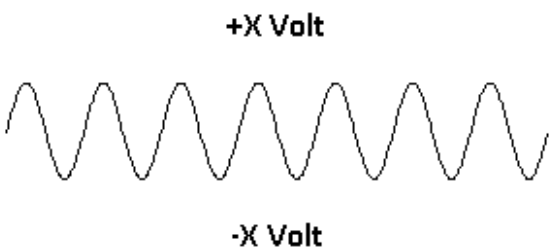
   This is how to configure it.

```
// decimation output feeds the integrator (27.2.2.3 Decimation Filter Module Extended
Configuration Register
(DECFILT_x_MXCR))
DECFIL_A.MXCR.B.SISEL      = 0;
```

2. Configure the Integrator Output

   Before configuring the Integrator output, look at the signal in Figure 3. Figure 3 shows a symmetrical sinusoidal signal with a peak-to-peak voltage of +X volt to –X volt.

Figure 3. Sinusoidal signal



If the above signal is accumulated, the resultant sum is zero. To get a meaningful result using the Integrator, one of the simplest ways is to perform a mathematical transformation to the Decimation output value (V) before accumulating the results as shown in the equation below. This equation simply takes the absolute value of the Decimation output results and accumulates it.

$$\sum_{s=0}^{n} \sqrt[2]{V_s^2}$$

The field SSIG in DECFILT_B_MXCR register allows the Integrator input to take signed or absolute value filter outputs. By disserting this field, the Integrator input takes the absolute value of the filter output.

```
// integrator input takes the absolute value of filter output (27.2.2.3 Decimation
Filter Module Extended Configuration Register
(DECFILT_x_MXCR))
DECFIL_B.MXCR.B.SSIG      = 0;
```

3. Reading the Integrator Output

Before reading the Integrator output, assert the Integrator Request Output (SRQ) bit. After asserting the SRQ bit, the DECFILT_x_FINTVAL will at the point of assertion contain the Integrator output . The MPC5674F allows assertion of this SRQ bit by software or hardware. In this application note the bit is asserted using the CPU.

```
// integrator output request (27.2.2.3 Decimation Filter Module Extended Configuration
Register
(DECFILT_x_MXCR))
DECFIL_B.MXCR.B.SRQ       = 1;
```

3 Knock Filter Demonstration

In this demonstration, the function generator was configured with a five second sweep period, start frequency of 1 KHz, and an end frequency of 20 KHz. Appended are the results of the demonstration program. The blue color signal is the input signal from the function generator and the cyan color signal is the output of the serial DAC.

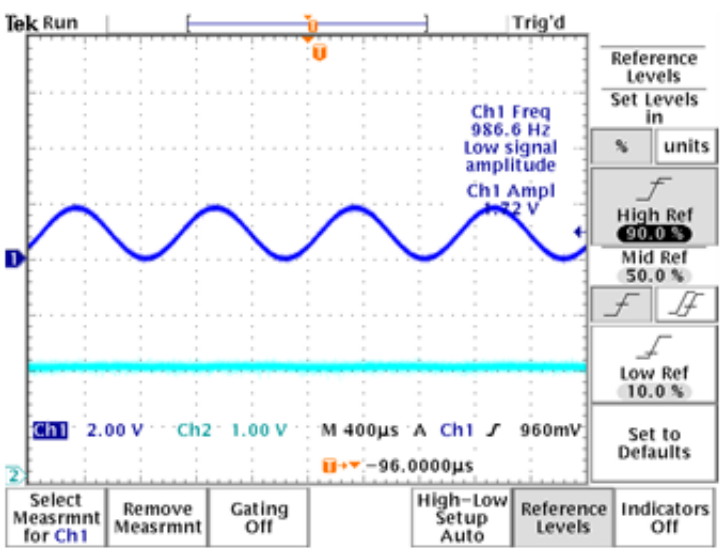


Figure 4. Low frequency plot

Figure 4 demonstrates the characteristic of a band pass filter filtering the lower frequency. Similarly, Figure 6 filters the higher frequency. Figure 5 shows that if the input signal is within the BPF bandwidth, the signals are allowed to pass through the filter.

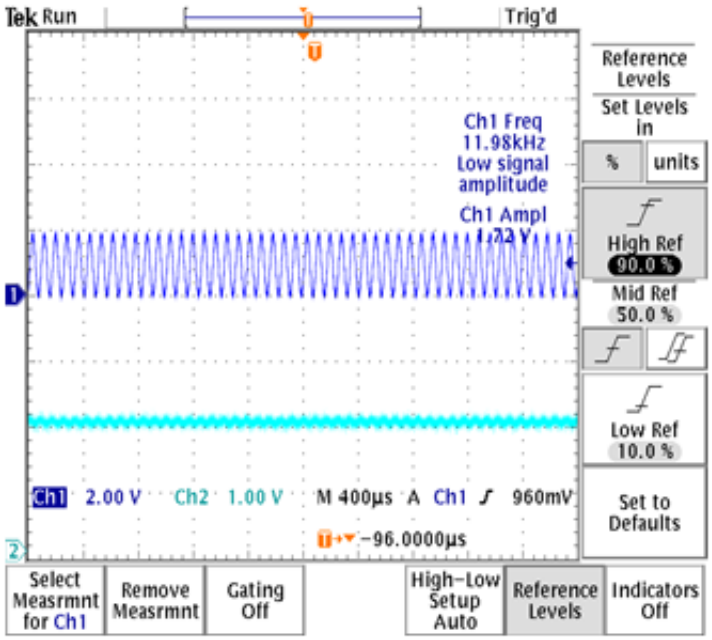


Figure 5. High frequency plot

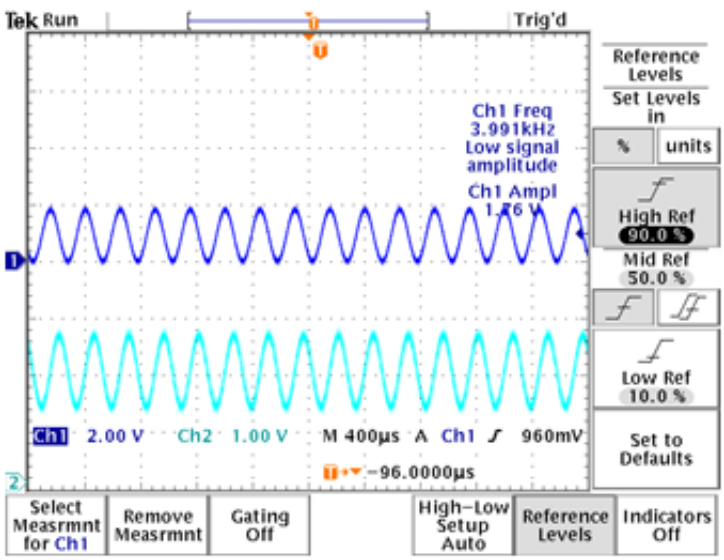


Figure 6. Bandpass frequency plot

The host CPU is used to retrieve data for the RFIFO and send it to the Serial DAC via the DSPI. The waveforms in Figure 4, 5, and 6 are outputs from the Serial DAC. For more information on the setting, please refer to the demonstration program.

4 Conclusion

For more information and learn more about the Freescale Automotive grade Power Architecture Micro-controller, please visit to the Freescale website, www.freescale.com.

How to Reach Us:

Home Page:
www.freescale.com

Web Support:
http://www.freescale.com/support

USA/Europe or Locations Not Listed:
Freescale Semiconductor
Technical Information Center, EL516
2100 East Elliot Road
Tempe, Arizona 85284
+1-800-521-6274 or +1-480-768-2130
www.freescale.com/support

Europe, Middle East, and Africa:
Freescale Halbleiter Deutschland GmbH
Technical Information Center
Schatzbogen 7
81829 Muenchen, Germany
+44 1296 380 456 (English)
+46 8 52200080 (English)
+49 89 92103 559 (German)
+33 1 69 35 48 48 (French)
www.freescale.com/support

Japan:
Freescale Semiconductor Japan Ltd.
Headquarters
ARCO Tower 15F
1-8-1, Shimo-Meguro, Meguro-ku,
Tokyo 153-0064
Japan
0120 191014 or +81 3 5437 9125
support.japan@freescale.com

Asia/Pacific:
Freescale Semiconductor China Ltd.
Exchange Building 23F
No. 118 Jianguo Road
Chaoyang District
Beijing 100022
China
+86 10 5879 8000
support.asia@freescale.com

For Literature Requests Only:
Freescale Semiconductor Literature Distribution Center
1-800-441-2447 or +1-303-675-2140
Fax: +1-303-675-2150
LDCForFreescaleSemiconductor@hibbertgroup.com

Document Number: AN4209
Rev. 1, 09/2011

freescale semiconductor