Renesas RA Family

Digital Filtering using the IIR Filter Accelerator

Introduction

This application note provides a brief overview and design considerations of digital filters. It covers the configuration and operation of the IIR Filter Accelerator (IIRFA) peripheral on the MCK-RA6T2. The application example supplementing this note demonstrates filter design, coefficient extraction, peripheral configuration and operation, and output verification for using a bandpass filter to remove noise from a known signal.

This application note enables you to effectively design and implement an IIR filter for your own application running on the MCK-RA6T2.

Target Devices

- RA6T2

Required Resources

To build and run the IIRFA Application example, you will need the following resources:

Development tools and software:

- e^2 studio Integrated Development Environment (IDE), version 2024-01.1
- RA Family Flexible Software Package (FSP) v5.2.0
- JLink RTT Viewer, version 7.94g

The FSP and e^2 studio are bundled in a downloadable platform installer available on Renesas' website at: renesas.com/ra/fsp

- MATLAB, version R2014b or later (https://www.mathworks.com/products/matlab.html)
    DSP System Toolbox (https://www.mathworks.com/products/dsp-system.html)

Hardware:

- RA6T2 (https://www.renesas.com/us/en/products/microcontrollers-microprocessors/ra-cortex-m-mcus/ra6t2-240mhz-arm-cortex-m33-trustzone-high-real-time-engine-motor-control)
- USB-C to USB-A cable
- Host PC running Windows 10

Prerequisites and Intended Audience

This application note assumes you have some experience with the Renesas e^2 studio IDE and RA Family Flexible Software Package (FSP). Before you perform the procedures in this application note, follow the procedure in the *FSP User Manual* to build and run the Blinky project. Doing so enables you to become familiar with e^2 studio and the FSP and validates that the debug connection to your board functions properly. Additionally, this application note assumes that you have some theoretical background on DSP topics relating to IIR filtering.

The intended audience are users who want to develop applications with the IIRFA module on the RA6T2 MCU.

Contents

1.   Introduction to IIR Filters

This section provides a brief background on digital filtering in general and focuses on the theoretical implications of using an IIR filter implementation. If you are already familiar with digital filtering, feel free to skip to Section 2 of this document.

Note: It is assumed that you already have a basic understanding of DSP theory relating to digital filtering. This section discusses the major differences between the design options offered for IIR filter design in MATLAB. The goal is to provide you with enough knowledge to help you choose the best options for your own filtering needs. If you would like a deeper discussion in DSP theory, please refer to section 1.5 for additional reading.

1.1   Definition of IIR and FIR Filters

Digital filters are discrete-time systems that perform algorithmic operations on a sampled signal to reduce or enhance specific aspects of the input signal. Digital filters act on the phase and frequency response of the input to allow certain frequencies of the signal to pass through to the output while blocking (that is, attenuating) unwanted frequencies.

In the time domain, the response of a digital filter to dynamic changes in the input signal is characterized by the impulse response of the system. In the frequency domain, the response of the filter is characterized by the transfer function of the system.

The impulse response is always an infinite-length signal, described by a sequence, and it can contain both zero and nonzero values. The nonzero values of the impulse response are commonly referred to as taps. There are two cases of filters that can be realized based on the impulse response:

Infinite Impulse Response (IIR) Filter

The Infinite Impulse Response (IIR) Filter contains an infinite number of taps in the impulse response. The analogous transfer function of the system contains a feedback component.

Finite Impulse Response (FIR) Filter

The Finite Impulse Response (FIR) Filter contains a finite number of taps in the impulse response.

1.2   Advantages and Disadvantages of IIR Filters

When approaching a filtering problem, it is important to consider the tradeoffs between the two filter types to design a system that meets the desired requirements. This section will only detail the advantages and disadvantages of using an IIR filter. In general, the advantages of IIR filters mirror the disadvantages of FIR filters, and vice versa.

Advantages

Some of the main advantages of an IIR filter include:

- A lower computational cost than an FIR filter with equivalent behavior specifications.
- Shorter input-output signal delay.
- Compact representation.

Disadvantages

Some of the main disadvantages of using an IIR filter include:

- System stability is not guaranteed (however, numerical tools mitigate this issue by picking poles and zeros for a stable system).
- Phase response of system is more difficult to control.
- Designing the filter can be a complex mathematical procedure (however, design is simplified when taking advantage of powerful filter design tools, such as MATLAB).
- Sensitive to numerical precision (see section 2.2 of this document for further discussion).

1.3 Biquad IIR Filters

In the frequency domain, the response of the IIR filter is characterized by the transfer function of the system. In other words, the transfer function mathematically expresses the input to output behavior of the filter.

1.3.1 Filter Order and Cascaded Biquads

IIR transfer functions are expressed as a ratio of polynomials in terms of complex frequency; where the numerator contains the zeros of the system, and the denominator contains the poles of the system. The number of zeros and poles in the transfer function determines the filter order of the system and the values of the zeros and poles describe the frequency response of the filter.

A biquadratic (biquad) filter is a second order filter, containing two poles and zeros. Higher-order filters are often necessary because they can achieve sharper frequency roll-offs, however they are more sensitive to coefficient quantization and thus have the possibility of becoming unstable. This is less of a problem with biquad filters and so typically, higher order filters are implemented as a series of cascaded biquad sections.

1.4 IIR Filter Specifications

Filter design is the process of turning a set of requirements into a well-defined numerical algorithm. IIR filter specifications are formulated in terms of the desired frequency response, analog design, and structure.

1.4.1 Types of Filter Frequency Responses

As mentioned previously, digital filters act on an input signal's frequency response to create an output with enhanced or reduced responses in specific frequency ranges. The common frequency responses of filters are of the following types:

Lowpass: A lowpass filter allows frequencies below a specified cutoff frequency to pass to the output, while blocking frequencies above the cutoff value.

Highpass: A highpass filter allows frequencies above a specified cutoff frequency to pass to the output, while blocking frequencies below the cutoff value.

Bandpass: A bandpass filter is a combination of a highpass filter followed by a lowpass filter. Any frequencies that lie between the two specified cutoff frequencies are passed to the output signal, while frequencies that lie outside the specified band are blocked.

Bandstop: A bandstop filter is a combination of a lowpass filter followed by a highpass filter. Any frequencies that lie outside of the two specified cutoff frequencies are passed to the output signal, while frequencies that lie inside the specified band are blocked.
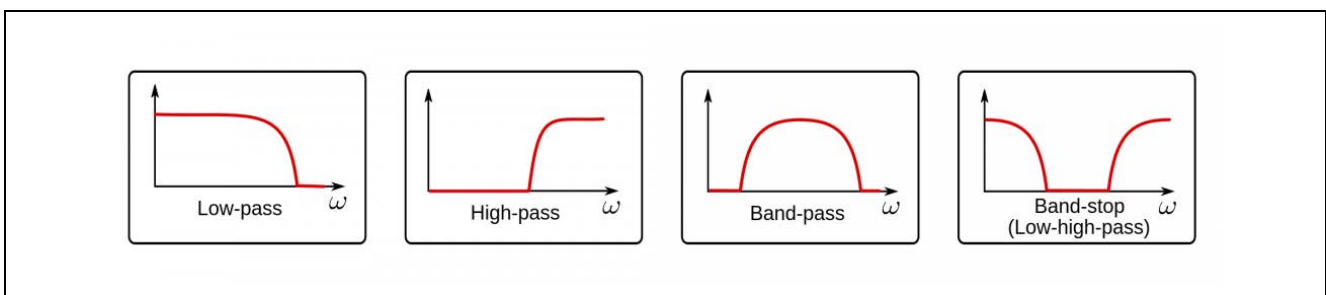

Figure 1.   The four frequency responses

1.4.2 Types of Analog Filters

Due to mathematical limitations, there is no optimal procedure that exists for designing an IIR filter. Instead, IIR filters are made by following established design procedures of existing analog filter designs. The analog filter algorithms' values are tweaked to achieve the design specifications for frequency and phase for a desired IIR filter. DSP numerical software, such as MATLAB, serve as powerful tools for computing optimization heuristics to design the best IIR filter based on your needed specifications.

One type of specification often provided is the analog filter design on which the IIR filter should be based on. There are four options for the analog filter design method offered by MATLAB, each explained further below:

Butterworth

Key Features:

- No ripple in passband or stopband.
- Passband to stopband frequency roll-off is the least steep.
- Moderate phase distortion.

Chebyshev I

Key Features:

- Ripple in passband, no ripple in stopband.
- Frequency roll-off is steeper than Butterworth filter.
- Poorer group delay.

Chebyshev II (Inverse Chebyshev)

Key Features:

- No ripple in passband, ripple in stopband.
- Frequency roll-off is steeper than Butterworth filter, but not as steep as Chebshev I filter.

Elliptical

Key Features:

- Ripple in passband and stopband.
- Frequency roll-off is the steepest.

Comparison

The following image depicts the frequency responses of the different analog filter designs when configured with the same specifications. Notice the differences in passband and stopband ripples as well as the steepness of the frequency roll-off.
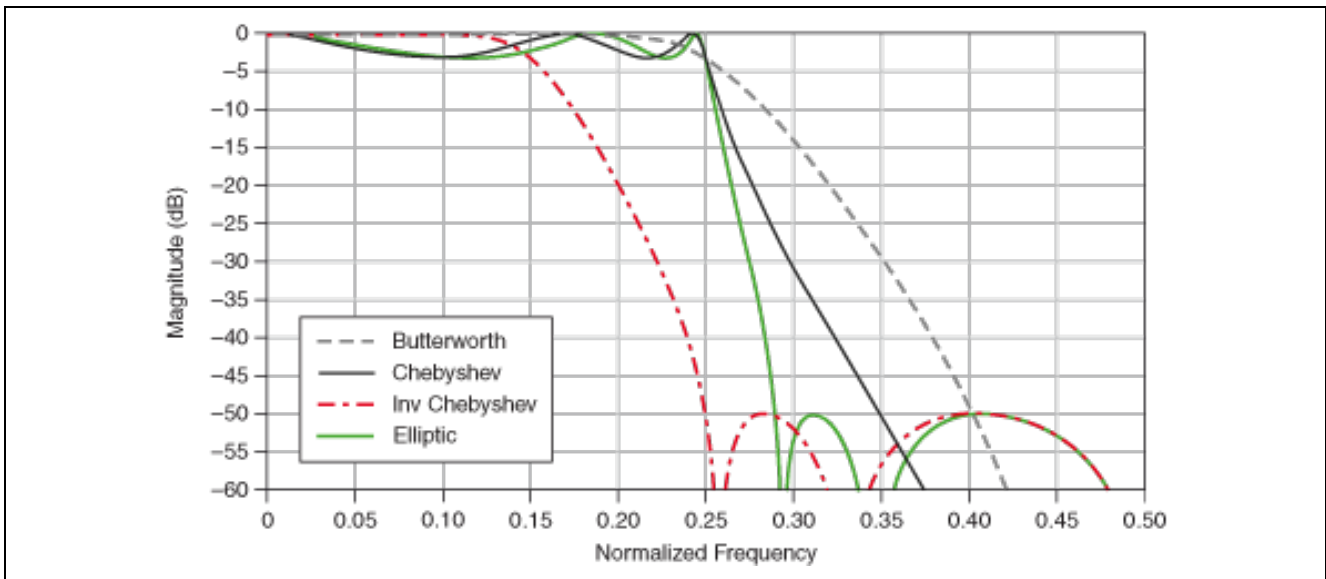

Figure 2.   Differences in the analog filter frequency responses when given the same filter specifications

1.4.3   IIR Filter Architecture

As mentioned previously, the response of a filter is characterized by the transfer function of the system, which can be broken down into a series of biquads with the following general form:

$$H(z) = \prod_{m=1}^{M} \frac{b_0^m + b_1^m z^{-1} + b_2^m z^{-2}}{1 - a_1^m z^{-1} - a_2^m z^{-2}}$$

The numerator contains the feed-forward coefficients, and the denominator contains the feed-back coefficients.

The operation for each biquad IIR filter is defined by the following difference equation:

$$y(n) = b_0 x(n) + b_1 x(n-1) + b_2 x(n-2) + a_1 y(n-1) + a_2 y(n-2)$$

One biquad IIR filter operation is called a stage and cascade-connected stages are called channels.

Direct Form 2 Transposed Structure

Based on the arrangement of terms in this equation, there are a variety of different algorithmic structures, or architectures, for the implementation of a biquad digital filter. The IIRFA peripheral implements the Direct Form 2 Transposed filter which is a canonical form with the minimum number of delay elements.
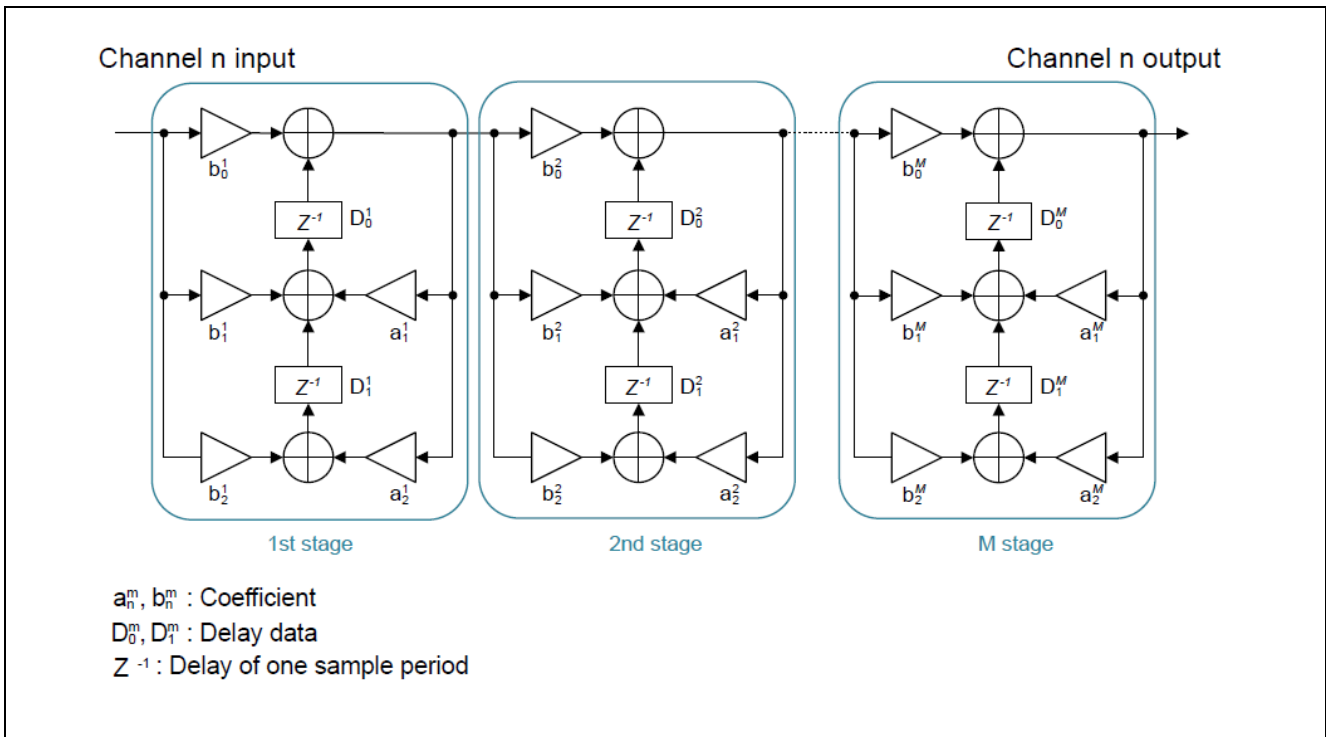

Figure 3.   Cascade-connected stages form a channel

1.5   Additional DSP Resource and Reading

For further reading into DSP and a deeper discussion of the mathematics surrounding digital filtering, please reference:

Understanding Digital Signal Processing by Richard G. Lyons

Signal Processing for Communications by Paolo Prandoni and Martin Vetterli, available online at: https://www.sp4comm.org/webversion.html

2.   Configuring the IIRFA

This section provides an explanation of the user-configurable settings of the IIRFA module. It details any limitations and discusses the differences between the various operation methods to provide a guide for choosing the best configuration for your application.

2.1   Module Settings

IIRFA Configurations

The IIRFA module can be added in e² studio to the **FSP Configuration > Stacks** tab via **New Stack > DSP > IIR Filter Accelerator (r_iirfa).** Table 1 shows the configurable module settings which can be changed in **Properties > Settings** in the **Stacks Configuration.**

Table 1.   Settings available in the configuration.xml

Property	Options	Default	Description
Parameter Checking	• Default (BSP) • Enabled • Disabled	Default (BSP)	If enabled, selected code for function parameter checking is included in the build.
Polling Mode	• Enabled • Disabled	Enabled	If enabled, the IIRFA driver will poll for processing completion before reading the output register. This prevents IIRFA operations from delaying global interrupt processing at the cost of slower filter performance.
Software Loop Unroll Depth	1-12, 16, 20, 24, 32 Samples	1 Sample	Sets the number of samples to process for every loop. Only affects filters with 1 biquad stage.
ECC Support	• Disabled • Enabled • Enabled (no writeback)	Enabled	If enabled, the ECC performs 1-bit error correction and 2-bit error detection on the coefficient and delay data which is retained in local RAM.
Rounding Mode	• Nearest • Toward zero	Nearest	Sets how to round calculation results.
Name	Valid C symbol	g_iirfa0	Module name.
Channel	Non-negative integer	0	Sets the IIRFA channel.

Stage Settings

One biquad IIR filter operation is referred to as a stage and cascade-connected stages are referred to as channels. The following stage settings are available for the IIRFA:

• Up to 32 stages can be cascade-connected (Note: 32 stages are available across all configured channels. In other words, the maximum sum of filter orders across all filters is 64.)
• The stages to be cascade-connected can be selected for each channel.
• The coefficient and delay data can be set independently for each stage.

Interrupts

The following interrupts can be generated during IIRFA processing:

• Output data preparation completion interrupt
• Process completion interrupt
• Operation error interrupt
• ECC error interrupts

Table 2 provides the list of interrupt sources:

Table 2.   Interrupt sources of the IIRFA

Name	Interrupt source	Interrupt flag	Interrupt generation condition
IIRFA_ORDYn (n= 0 to 2)	Channel n output data preparation is completed.	IIRCHnSTS.ORDYF	IIRCHnSTS.ORDYF = 1 and IIRCHnINT.ORDYIE = 1
IIRFA_ORDY3 (m = 3 to 15)	Output data preparation is completed in any of channel m.	IIRCHmSTS.ORDYF	IIRCHmSTS.ORDYF = 1 and IIRCHmINT.ORDYIE = 1
IIRFA_CPRCFn (n = 0 to 2)	Channel n processing is completed.	IIRCHnSTS.CPRCFF	IIRCHnSTS.CPRCFF = 1 and IIRCHnINT.OPRCFIE = 1
IIRFA_CPRCF3 (m = 3 to 15)	Processing is completed in any of channel m.	IIRCHmSTS.CPRCFF	IIRCHmSTS.CPRCFF = 1 and IIRCHmINT.CPRCFIE = 1
IIRFA_ERR (x = 0 to 15)	An operation error has occurred in any channel.	IIRCHxSTS.CERRF	IIRCHxSTS.CERRF = 1 and IIRCHxINT.CERRIE = 1
	An ECC 1-bit error occurred.	IIRECCEF.ESEF	IIRECCEF.ESEF = 1 and IIRECCINT.ESEIE = 1
	An ECC 2-bit error occurred.	IIRECCEF.EDEF	IIRECCEF.EDEF = 1 and IIRECCINT.EDEIE = 1

Rounding Mode

The I/O data, stage coefficients and delay data for the IIR filter accelerator module are retained in the single precision floating-point format specified in the IEEE 754 standard. Floating-point calculations are made internally with extra precision, and then rounded to fit into the destination type. There are two rounding modes for the IIR filter accelerator module supports, which are selectable in the **Properties > Settings** in the **Stacks Configuration** tab. A brief description of each mode and its impact on filter processing are discussed below.

Round to Nearest: In this mode, the results are rounded to the nearest representable value. If the result is midway between two representable values, the even value is chosen. An even value has zero as its least significant bit. This rounding mode is the most precise and prevents statistical bias in a case where there are a lot of midway values calculated.

Round to Zero: In this mode, the results are rounded towards zero, that is, are truncated to the representable value that is closest to zero in all cases.

2.2   Operation Methods

2.2.1   Channel Processing Procedure

Channel processing is a series of operations that are executed when a write access is performed to the input register of a channel. Operations for all stages used by the channel are performed sequentially during the channel processing. If the output data operation is completed in the middle of the channel processing, that is, before the channel processing completion (IIRCHnSTS.CPRCFF) flag is 1, the output data preparation completion flag (IIRCHnSTS.ORDYF) is 1 and the output data (IIRCHnOUT) register) can be read.

The maximum number of channels that can be processed at the same time is 1. Channels are processed sequentially. If a write access to the input data (IIRCHnINP) register is performed during the channel processing of any channel, bus access is forced to wait until the channel processing that is being performed is completed.

2.2.2   Methods for Processing

Based on the configuration of the IIRFA, there are three methods for the execution of channel processing. They differ in the procedure for reading the output data from the IIRCHnOUT register after the channel processing starts.

For definition purposes, single data processing means that the channel processing is performed for one input value and no other channel processing is performed after the processing is completed. Multiple data processing means that channel processing is performed consecutively for multiple input values.

Table 3 shows the procedure for each method:

Table 3. Methods for Processing Data

	Single data processing	Multiple data processing
Method 1	Read without waiting for output data preparation completion.	
Method 2	Read after polling the output data preparation completion flag.	Read after polling the channel processing completion flag.
Method 3	Read after accepting the output data preparation completion interrupt	Read after accepting the channel processing completion interrupt.

Method 1:

Method 1 is the procedure to read the output data without waiting for output data preparation completion. Both polling and interrupts are disabled. Once the input value is written to the IIRCHnINP register, core execution is halted until data is written to the IIRCHnOUT register at the end of IIRFA processing. This method is the fastest of all three, but it may introduce a higher global interrupt latency during processing. The interrupt latency is negligible if your sample processing is the highest priority task.

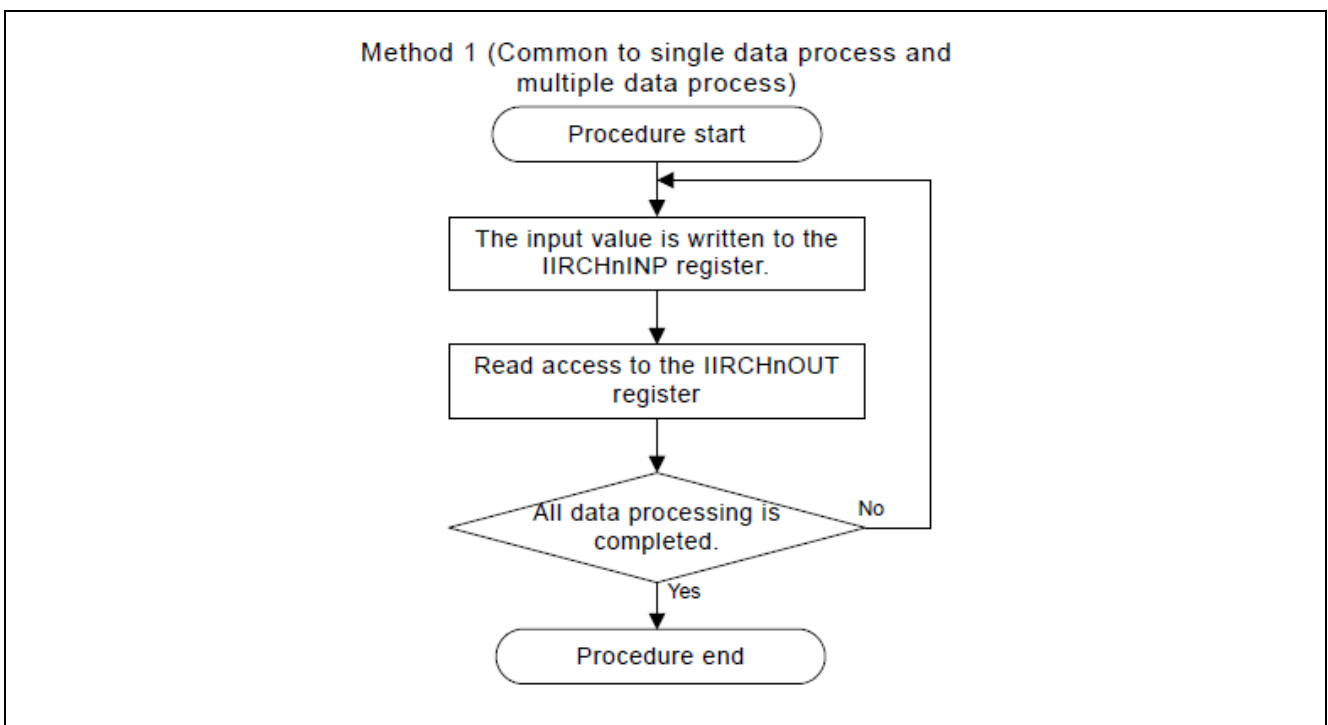Figure 4 is an example procedure flowchart for channel processing with Method 1.

Figure 4. Method 1 Procedural Flow

Key Notes:

- Memory and time overhead are the smallest of all methods since the flag determination processing is not required.
- When a read access to the IIRCHnOUT register is performed the CPU will halt until output data preparation is completed which will hang the System and PSBIU buses .
  Note: While execution waits for IIRCHnOUT no interrupts will be processed by the core. The maximum wait time for a 32-stage filter may be up to approximately 64 ICLK cycles for single sample processing or 224 ICLK cycles for multi-sample processing (decreasing linearly with the number of stages used).

Method 2:

Method 2 is the procedure to read the output data after the data preparation completion flag is set. Polling is enabled and interrupts on the IIR module are disabled. Once the input value is written to the IIRCHnINP register, the IIRFA driver will poll for the completion flags, which indicate that data is available at the IIRCHnOUT register.

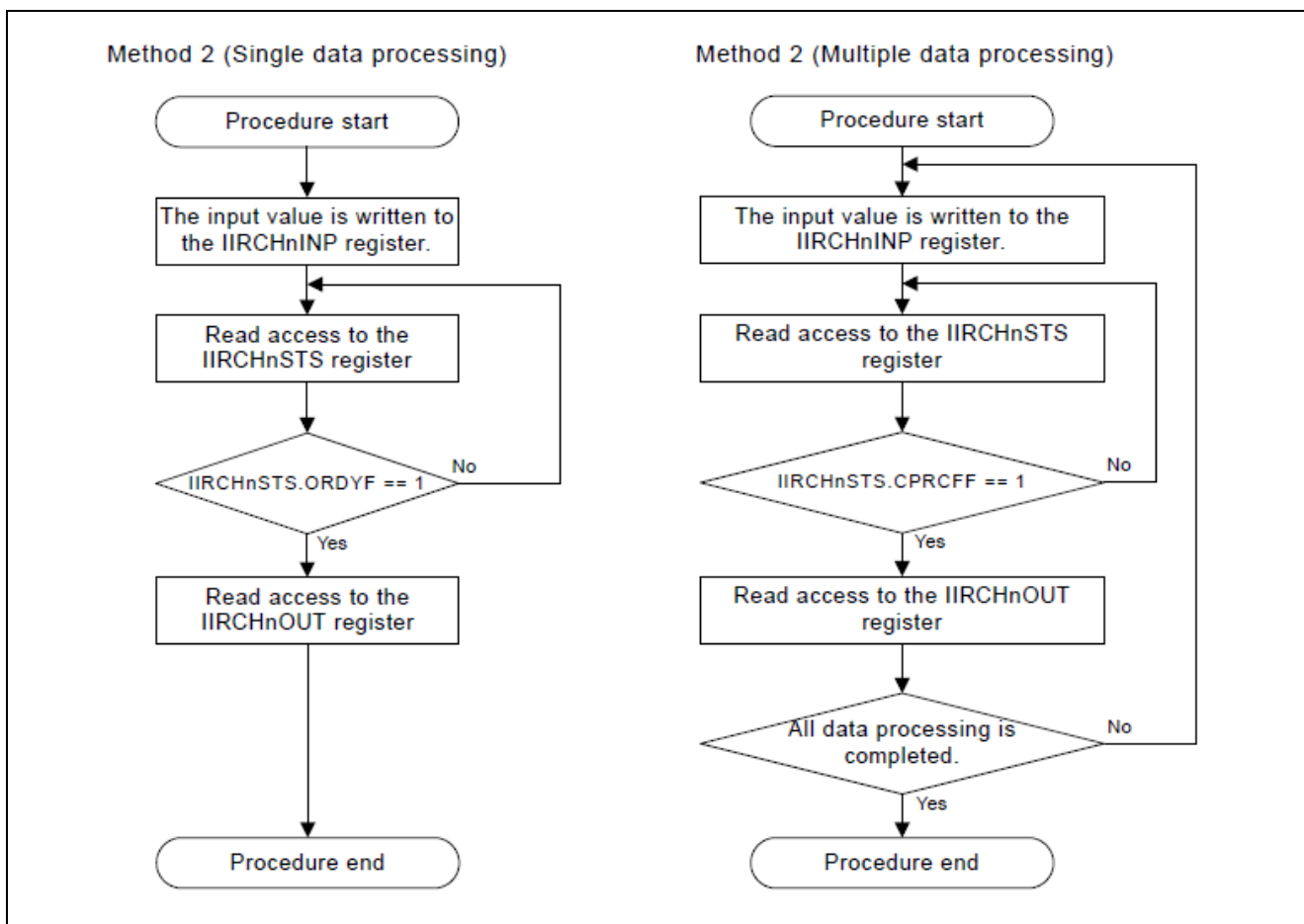An example procedure flowchart for channel processing with Method 2 is given below:

Figure 5.   Method 2 Procedural Flow

Key Notes:

- The overhead is large because flag determination processing is required.
- When a read access to the IIRCHnOUT register is performed, bus access is not forced to wait. Other system interrupts can be processed while waiting for the flag to be set.

Method 3:

Method 3 is the procedure to read the output data after the data preparation completion interrupt occurs. Polling is disabled and interrupts are enabled. Once the input value is written to the IIRCHnINP register, the core can process other instructions until the data preparation completion interrupt signals that data is available at the IIRCHnOUT register.

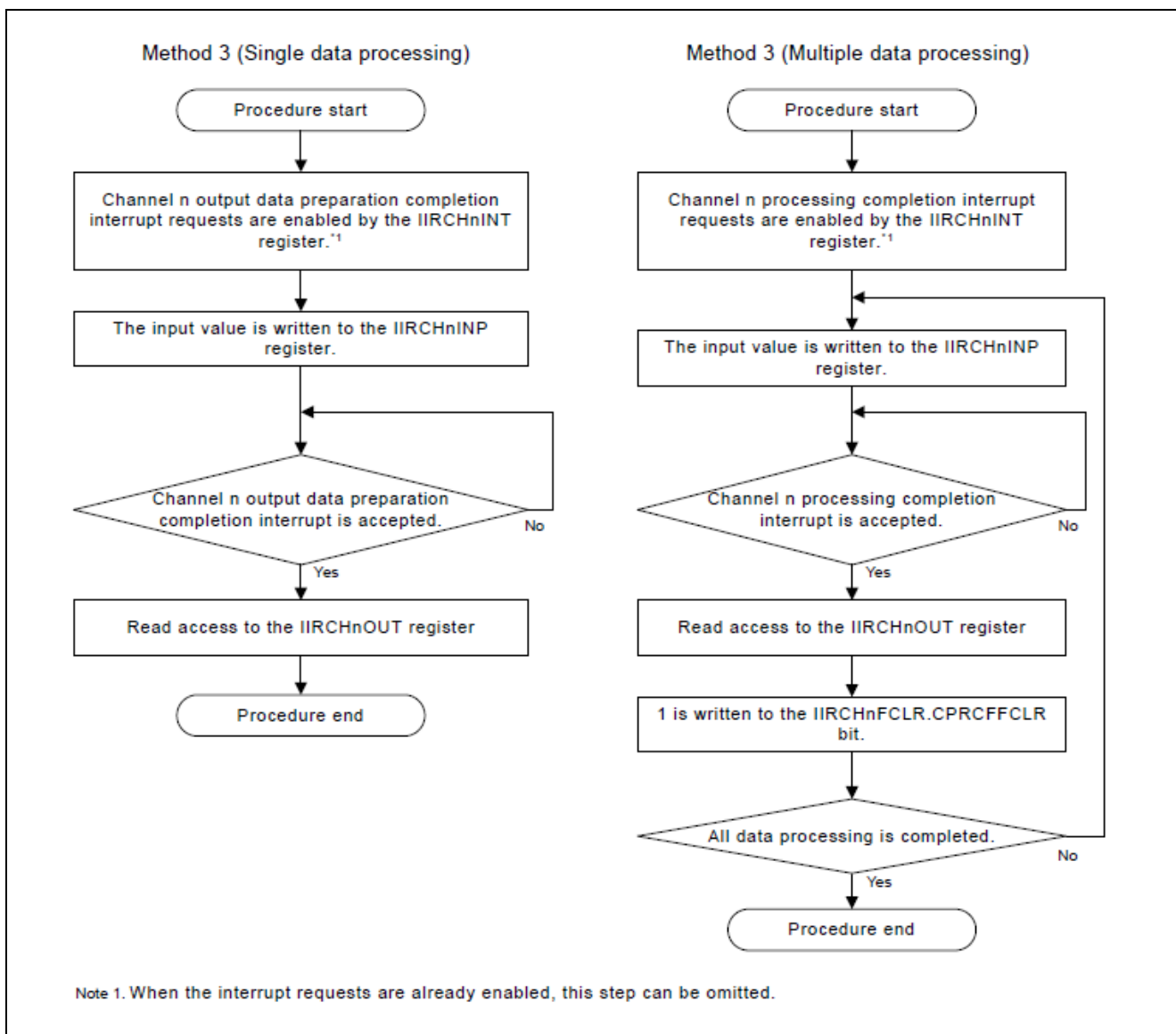Figure 6 is an example procedure flowchart for channel processing with Method 3.

Figure 6.　Method 3 Procedural Flow

Key Notes:

- The overhead is large due to processing at the time of interrupt acceptance.
- When a read access to the IIRCHnOUT register is performed, bus access is not forced to wait.
- After the channel processing starts, the core can execute other until the output data preparation completion interrupt or the channel processing completion interrupt occurs.

2.3　Considerations

2.3.1　Maximizing Performance

The optimum configuration for the IIRFA is application-dependent so it is important to consider the impact of the settings and operation methods available, based on your system's needs.

Each filter stage takes 2 ICLK cycles to process a sample and an additional 5 cycles to write state values back to registers. As such, single-sample operations only take 2 cycles per stage while multi-sample operations take 7. Additional overhead cycles are required to load and store each sample.

The following suggestions may improve performance:

- When using the function R_IIRFA_Filter, choose a larger data chunk size. For realtime filtering it is recommended to perform single sample processing using the R_IIRFA_SingleFIlter inline function.
  Note: the function R_IIRFA_Single filter has no parameter checking, processes exactly one sample, and returns a filtered sample. Polling is supported by this function (if configured).
- If your filter only uses 1 biquad stage, enable the software unroll loop depth setting and provide data in a multiple of the unroll depth.
- Disabling polling significantly improves performance when using a low number of stages; however, it may introduce a higher global interrupt latency during processing.
- When optimizing IIRFA processing time, method 3 should be avoided because the time needed to process the interrupt is much greater than the time needed to filter data. For large data sets, using interrupts can nearly double the total processing time.

2.3.2   Limitations

The key limitations to consider when designing your application are summarized in the following list:

- 32 total stages are available across all configured channels.
- Correct operation of the IIRFA is not guaranteed when using the DTC or DMA and is therefore not supported.
- When polling is disabled, core execution is halted while waiting for data to become available. While core execution is halted, no interrupts can be serviced.

3.   Methods for Filter Design

This section discusses the application space of IIR filters, the implications of numerical data types on filtering, and provides step-by-step instructions to design a Butterworth bandpass filter using the DSP System Toolbox in MATLAB.

3.1   Applications of IIR Filters

The advantages and disadvantages of IIR filters influence which applications they are the best choice for. IIR filters are generally chosen over FIR filters for applications where memory is limited due to the lower computational cost with equivalent behavior specifications, where linear phase is not important, and where the filter specifications require sharper frequency cutoffs and high throughput.

IIR filters have been widely used in telecommunication/RF applications, for example there are transmit and receive filters in digital modems which suppress noise and extract signals for further processing. They are present in IoT smart sensors (temperature, pressure, gas, image, etc.) to remove unwanted noise in the measurement data and make the signals useful for analysis. Additional industry examples of IIR filters include biomedical sensor signal processing, audio equalization, and clock recovery in data communication.

3.2   Data Precision and Its Impact on Digital Filtering

The I/O data, stage coefficient/delay data are retained in the single precision floating-point format specified in the IEEE754 standard.

Figure 7.   Single precision floating-point

The single precision floating-point format supports the following values:

- 0 < E < 255 (normal numbers)
- E = 0 and F = 0 (signed zero)
- E = 0 and F > 0 (denormalized numbers)
- E = 255 and F = 0 (infinity)
- E = 255 and F > 0 (NaN: Not a Number)
  MSB of F is 0. (SNaN: Signaling NaN)
  MSB of F is 1. (QNaN: Quiet NaN)

IIRFA treats the input as +0 if a positive denormalized number is input, −0 if a negative denormalized number is input, and infinity if a NaN (Not a Number) is input.

IIRFA performs addition and multiplication of single precision floating-points multiple times in the cascade connected biquad IIR filter operation. If the result of each addition and multiplication is a positive denormalized number, it is treated as +0. If the result is a negative denormalized number, it is treated as -0. If the result is a NaN (Not a Number), it is treated as infinity. In addition, the rounding mode for each addition and multiplication result can be selected by IIROPCNT.

3.2.1 Data Types and Precision

DSP chips commonly support one (or more) of the following data types: fixed point, single precision float, or double precision float. Fixed point processors are typically cheaper, use lower power and process calculations faster than their float counterparts.

However, float data can represent a larger dynamic range of numbers and tend to have better precision than fixed point data. Double precision floats store 64 bits of data and therefore can represent a much larger dynamic range of numbers than single precision floats which store 32 bits of data.

Overall, using single precision floats strikes a balance between processing speed, dynamic range, and precision.

3.2.2 Note on Bit Depth

When thinking about number representation and data types, another important thing to consider is that typically, the digital filter is just one part of a larger signal system implemented on an MCU. The resolution of the other elements of the system, for example, the ADC, could serve as a bottleneck to your system and should be chosen wisely. The RA6T2 has a 12-bit ADC.

Renesas offers an external D/A converter with a larger bit depth that can be found below:

14-bit High-Speed D/A:

https://www.renesas.com/us/en/products/analog-products/data-converters/digital-analog-converters-dac/isl5961-14-bit-33v-210130msps-commlink-high-speed-da-converter

3.3 Using MATLAB to Extract Coefficients

The example project accompanying this document demonstrates the operation of the IIRFA to implement a bandpass filter to remove unwanted frequencies from a 100-Hz noisy sinusoidal.

The first part of the project covers how to use MATLAB's DSP System Toolbox to design an IIR filter and generate coefficients for each biquad stage.

All Matlab scripts can be found in the **r11an0594/iir_filter_accelerator/tools** folder included with the application project.

Note: This application project covers the steps for filter design for MATLAB version R2014b. If you are using a different version, please refer to the MATLAB Help Center (https://www.mathworks.com/help/matlab/) to verify syntax and correct any possible differences in commands or settings.

3.3.1 Designing with filterBuilder

The following steps detail the process for using the filterBuilder tool in MATLAB's DSP System Toolbox to design a bandpass filter compatible with the IIRFA module that has a passband centered around 100 Hz.

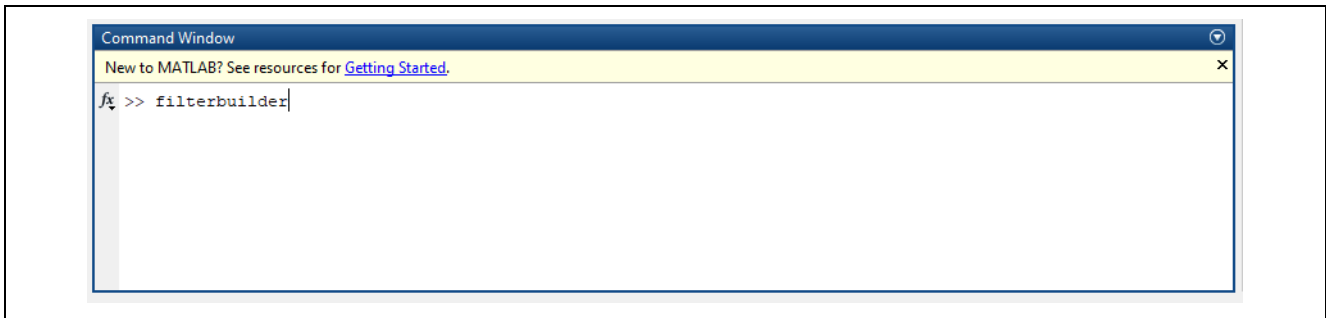Open MATLAB. In the **Command Window**, type the command `filterbuilder` and press **Enter**.

Figure 8.   Run the filterbuilder command

A **Response Selection** window will pop up and prompt you to select a filter response. Select **Bandpass** and press **OK**.
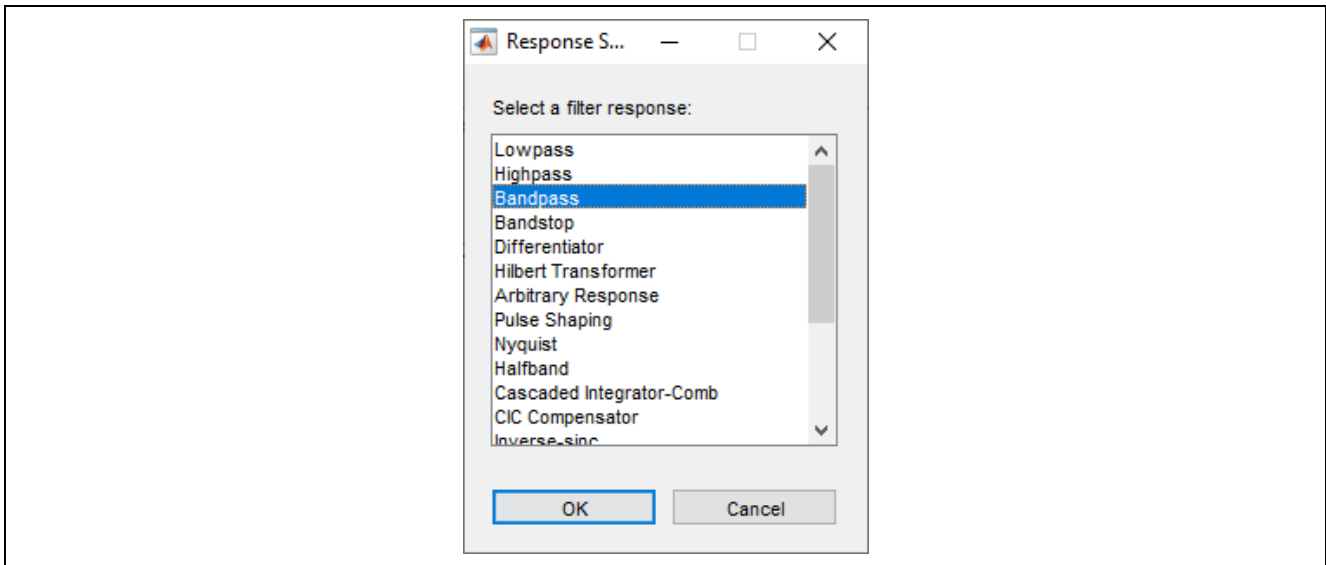

Figure 9.   Select a filter response

FilterBuilder's **Bandpass Design** window will open to the main pane. The **Bandpass Design** window is where the rest of the desired specifications for the filter will be set.

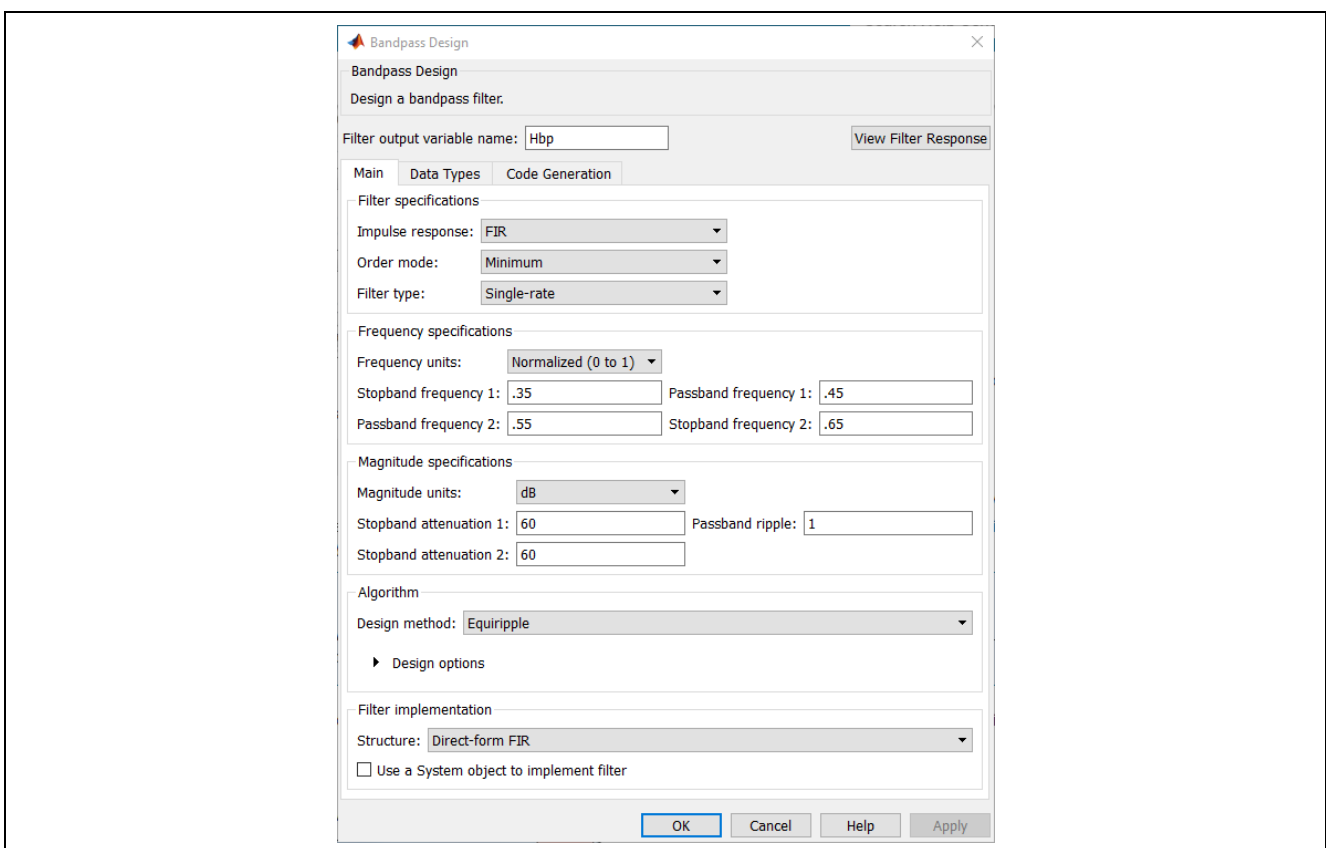The filter output variable is automatically named **Hbp** and will be used throughout the rest of this example.

Figure 10.   Default values of the filterBuilder bandpass design

In the **Filter specifications** section, click the drop-down arrow for **Impulse response** and select **IIR**. Leave the **Order mode** set to **Minimum**.
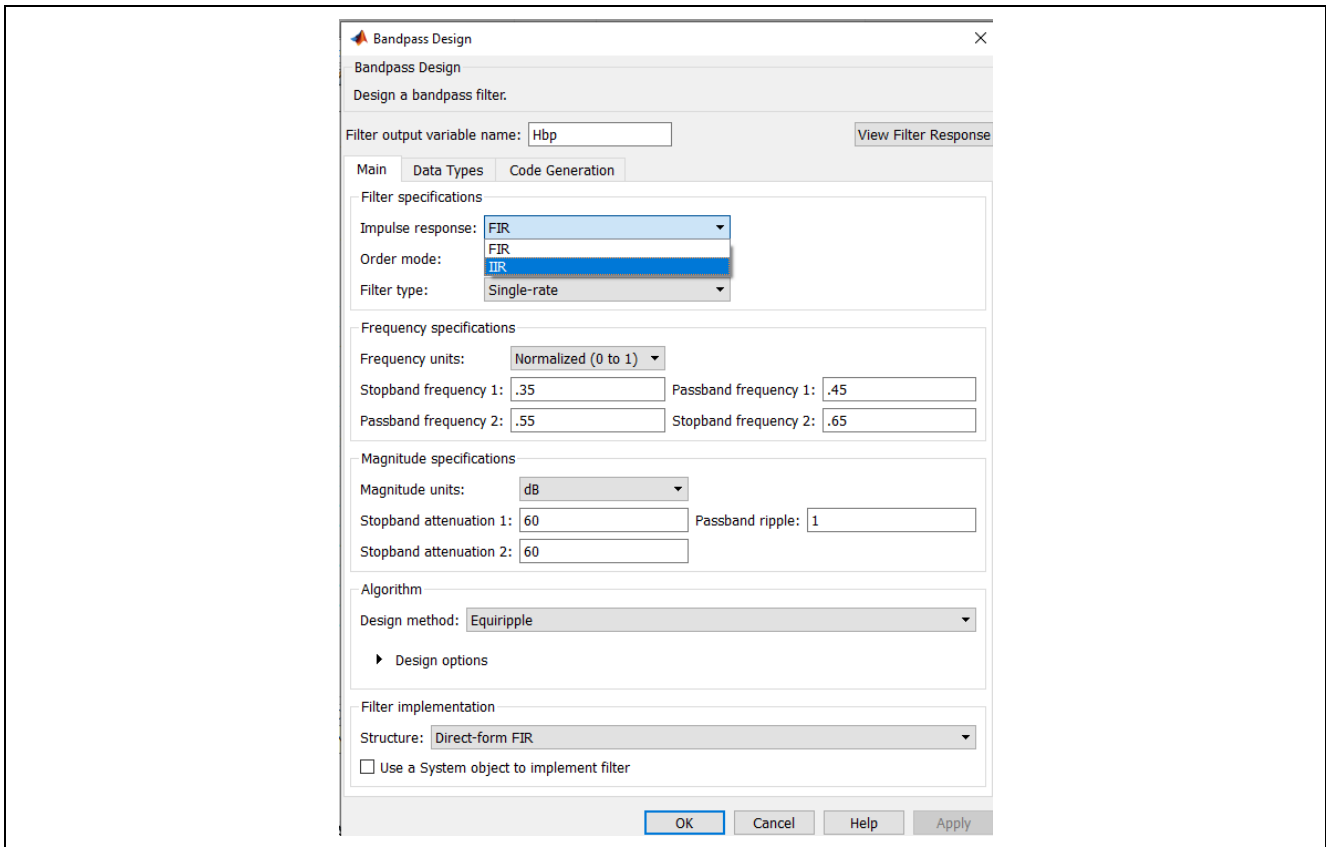
Figure 11. Select the filter specifications

In the **Frequency specifications** section, change the **Frequency units** to **Hz**.

The input signal is a 100-Hz sinusoidal signal with 256 sample points per period. The input sample rate is calculated by 100 Hz * 256 samples = 25600. In the **Input sample rate** box, enter **25600**.

To recover the input signal, the filter's pass band will be centered around 100 Hz. Set the following values:

- **Stopband frequency 1** to **70**
- **Passband frequency 1** to **90**
- **Passband frequency 2** to **110**
- **Stopband frequency 2** to **130**

Figure 12.   Set the frequency specifications

Leave the values in the **Magnitude specifications** and **Algorithm** sections to the defaults. The Butterworth filter is chosen in this application for the maximally flat response in the passband and stopband.

In the **Filter implementation** box, change the **Structure** to **Direct-form II transposed SOS**.
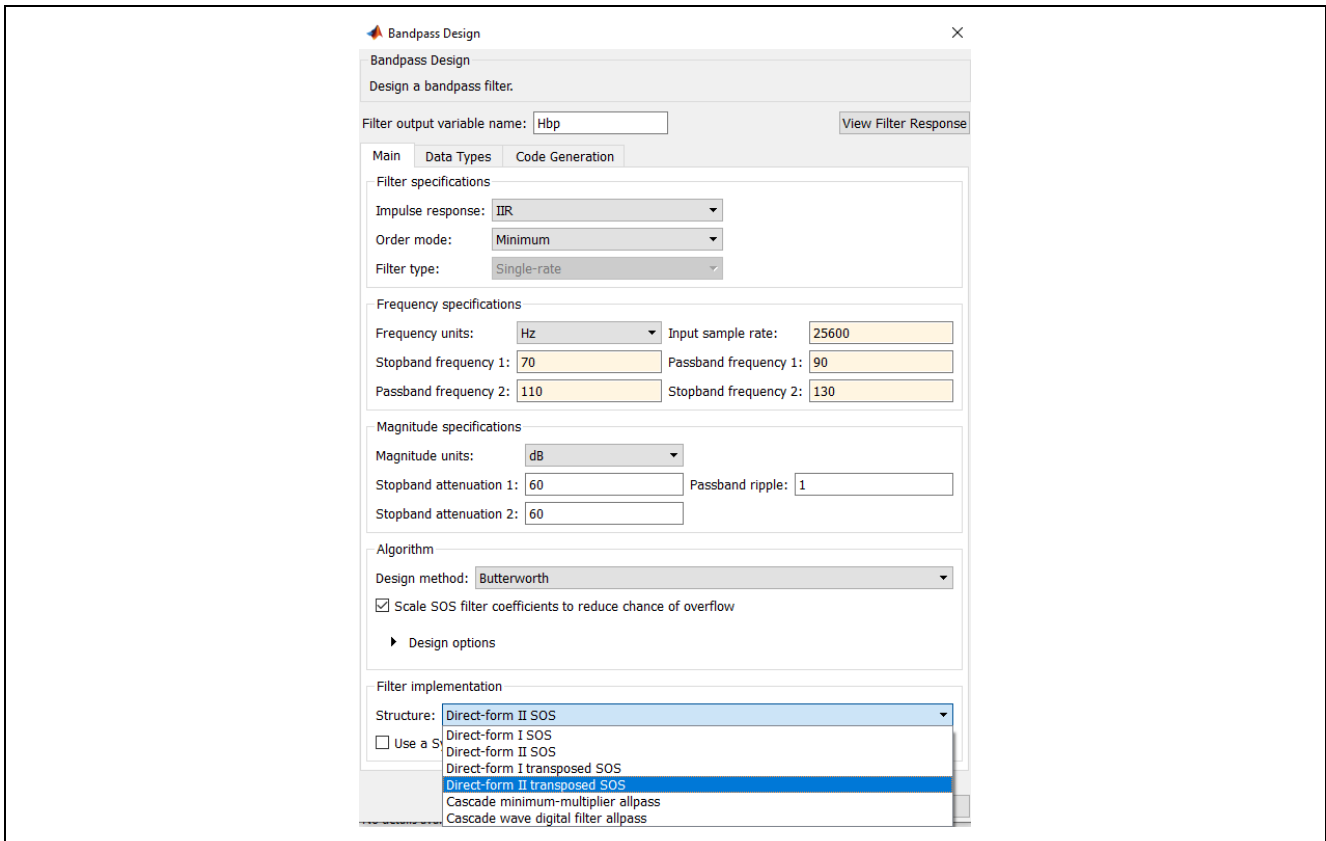
Figure 13.   Set the filter implementation

Next, click on the **Data Types** tab to switch to the **Data Types** pane. Change the **Arithmetic** to **Single precision.**
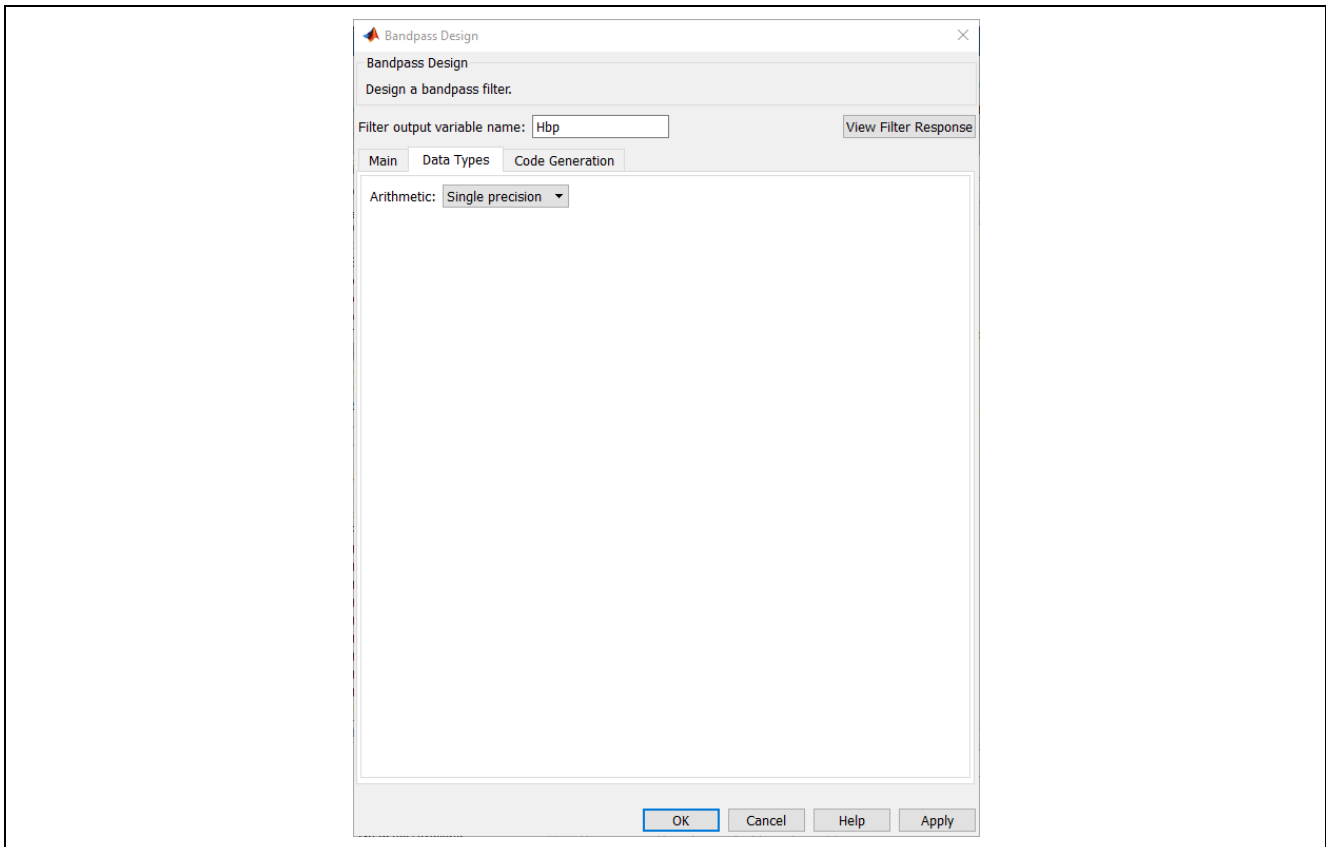
Figure 14.   Set the arithmetic data type

To view the response of the filter before creating the filter object, click the **View Filter Response** button in the top right corner of the window. If prompted to apply changes before visualization, select **Yes**.
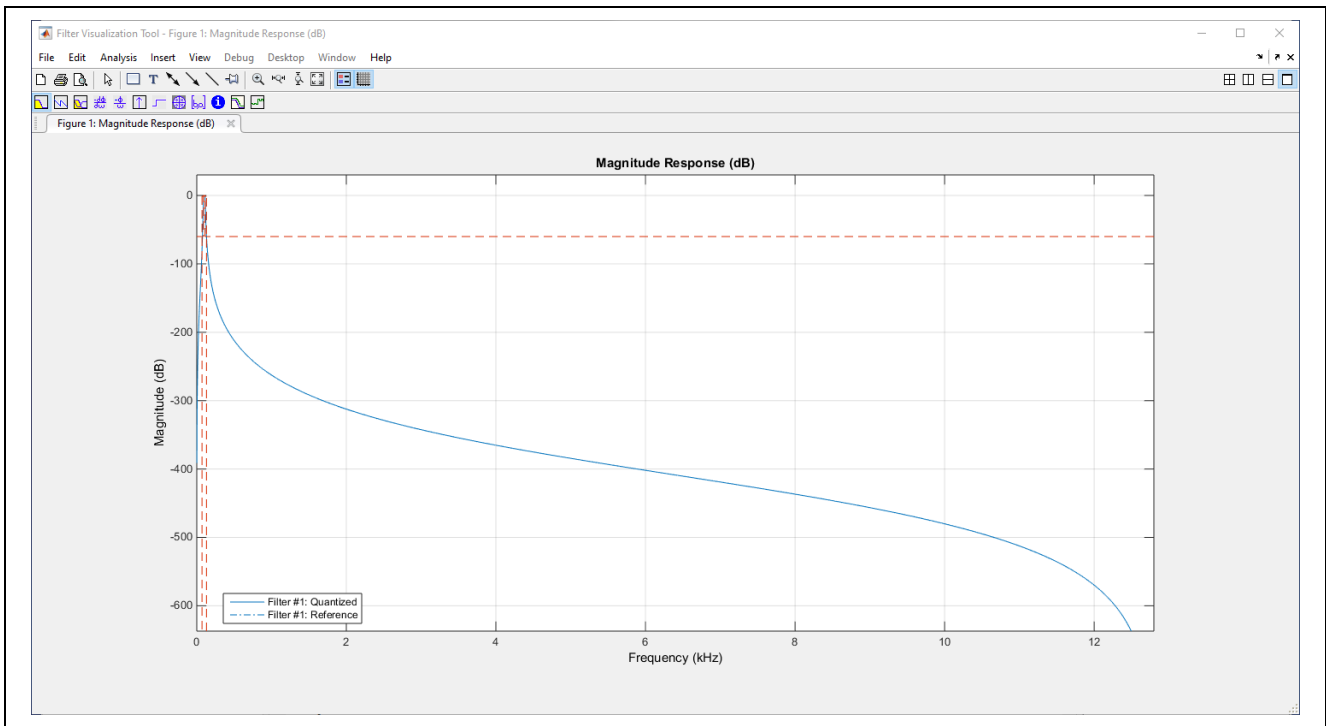
Figure 15. Filter frequency magnitude response

The frequencies displayed range from 0 Hz to the Nyquist frequency (12800 Hz) so it is difficult to view the 100-Hz passband with this view. To zoom in, Click the **Zoom** button in the top bar and draw a window around the passband region.
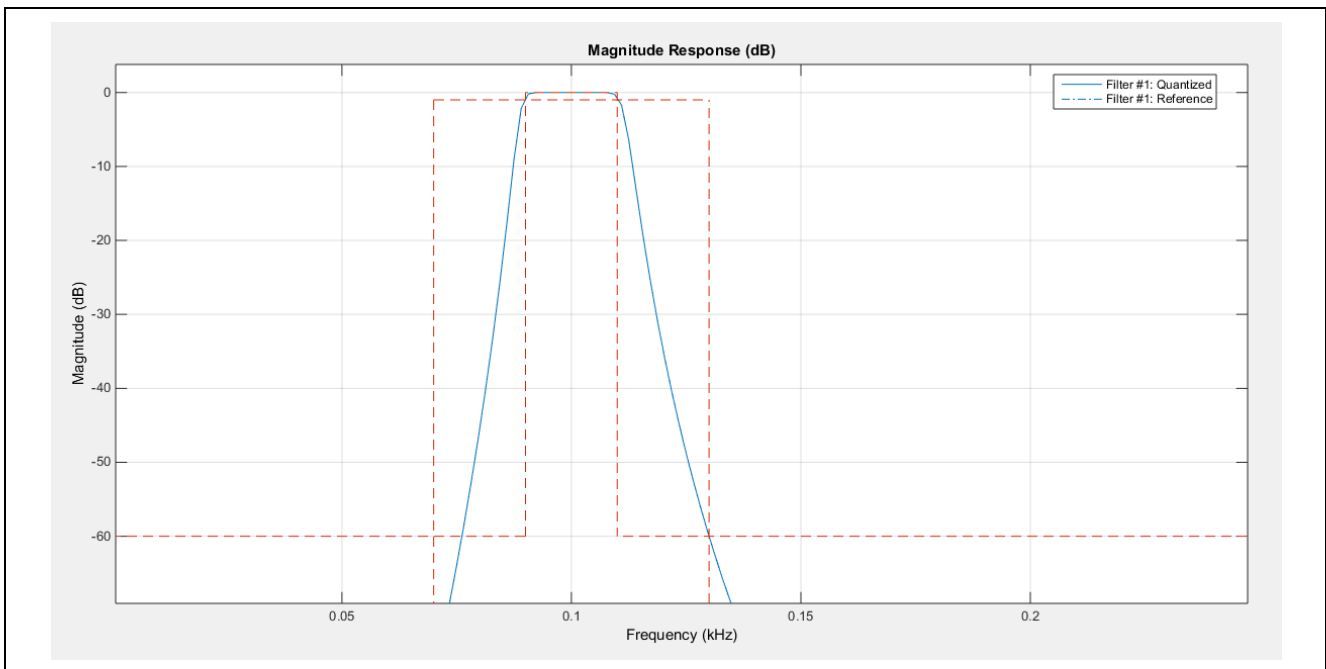
Figure 3.10 Zoom view of filter frequency magnitude response

Close the Filter Visualization Tool once you are done inspecting the frequency magnitude response.

Next, click **OK** to generate the filter variable and a *df2sos* MATLAB object named "Hbp" will populate the workspace.
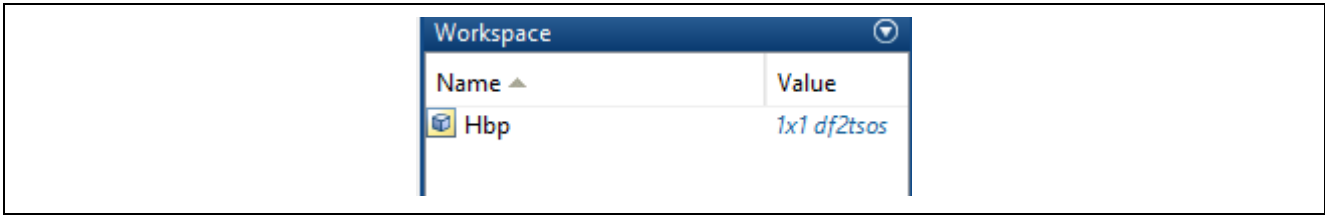
Figure 16.   The Workspace holds the df2tsos object

The *sosMatrix* in the *df2tsos* object holds the coefficients for each second-order-section in the filter.

In the **Command Window**, type `filtord(Hbp.sosMatrix)` and press **Enter** to view the order of the Hbp filter.
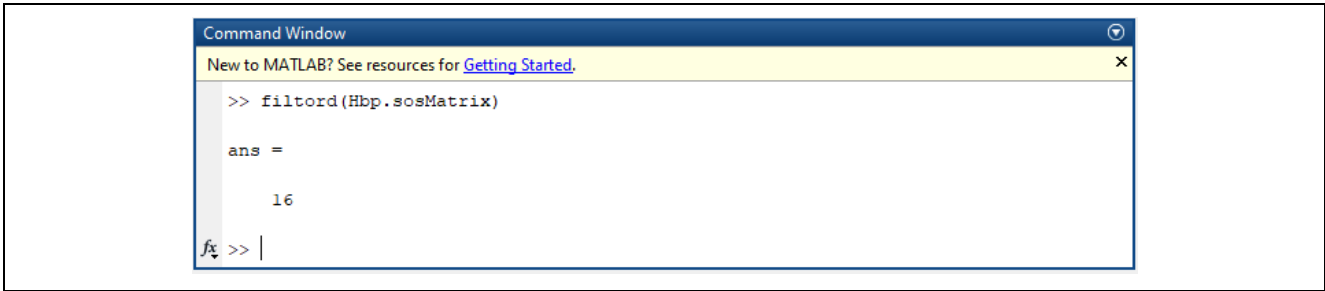
Figure 17.   Run the filtord() command

This filter has an order of 16, and therefore will need 8 cascaded biquads, or IIRFA stages, to implement.

Right-click on the filter object in the workspace, select **Save as** and save the object as `Hbp.mat` in the local directory that contains this example project's MATLAB scripts.

Figure 18.   Save Hbp as a .mat file

3.3.2   Exporting Filter Coefficients

This section explains how to use the included MATLAB script `extract_coefficients.m` to extract the coefficients from the *sosMatrix* in the *df2tsos* filter object to create a .txt file containing the coefficients in C code, formatted for the `iir_filter_coeffs_t` struct. All Matlab scripts can be found in the **r11an0594/iir_filter_accelerator/tools** folder included with the application project.

Double-click the `extract_coefficients.m` script to open it with MATLAB.

Press **Run**. The C code is generated in a new file located in the current directory named `IIRFAcoeff.txt`.
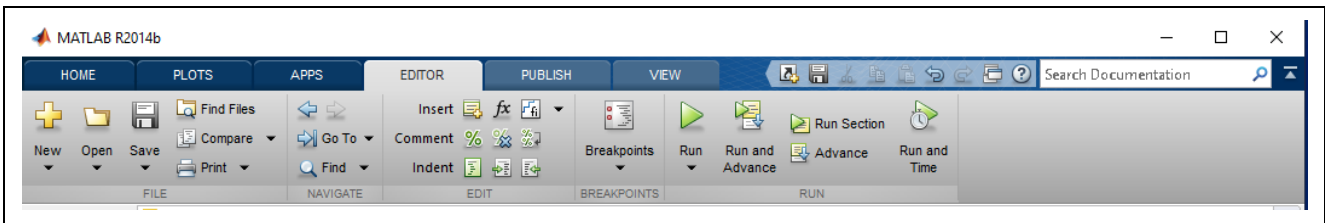

Figure 19.   Run the MATLAB script

3.3.3   Additional Filter Design Tools

GNU Octave is a free DSP numerical software tool and alternative to MATLAB. However, it is beyond the scope of this document to provide the detailed steps for filter design with it. Users wishing to use it should identify steps like those in Section 3.3.1.

GNU Octave is available for download at: https://www.gnu.org/software/octave/download

Signal package (with IIR filter design): https://octave.sourceforge.io/signal/

4.   Running the Example Project

4.1   Importing the Project and Configuring the IIRFA

The following instructions will show you how to import the example project into your e² studio workspace.

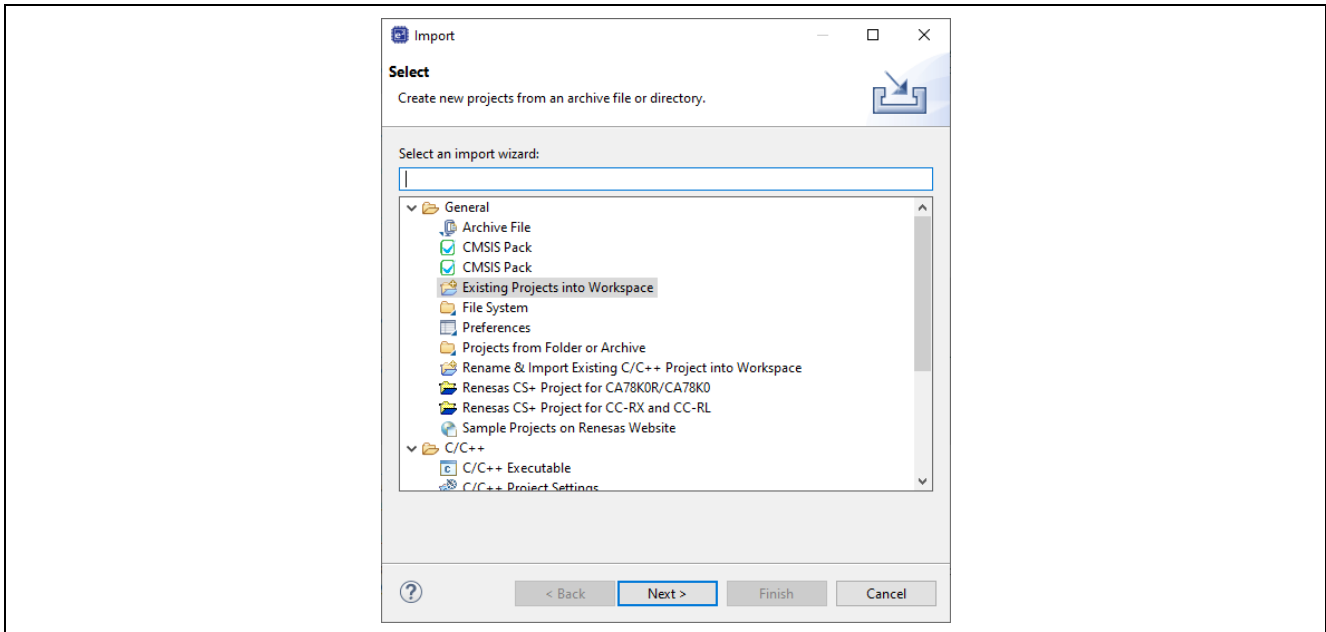Open e² studio and select **File > Import… > Existing Projects into Workspace** and click **Next.**

Figure 20.   Import existing project into workspace

Browse to the location of the application project in the **Select root directory:** section and click **Finish.**

Once the project has finished importing, double click on the `configuration.xml` file in the workspace Project Explorer to open it and click **Generate Project Content.**

4.2   Running the IIRFA Project

Connect the USB C end of the USB cable to the MCK-RA6T2 board and the USB A end to a port on your workstation.

In the Project Explorer, expand the **project > src** folder and double click on `hal_entry.c`. Open the `IIRFAcoeff.txt` file that was created by the `extract_coefficients.m` script (instructions are detailed in section 3.3.2 of this document).

Use the **Ctrl+A** and **Ctrl+C** shortcuts to copy the entire contents of `IIRFAcoeff.txt`. In `hal_entry.c`, paste the coefficients using **Ctrl+V** into the empty `iir_filter_coeffs_t` `gp_iirfa0_filter_coeffs[NUM_STAGES]` struct array, declared above `hal_entry()`.

Figure 21. Insert coefficients into the iir_filter_coeffs_t struct

Click on the [hammer icon] button to build and compile the project.

Click on the [debug icon] button to start a debug session for the project.

Open JLink RTT Viewer and in the **Specify Target Device** section click **…** and browse for the corresponding target device. Click **OK.**
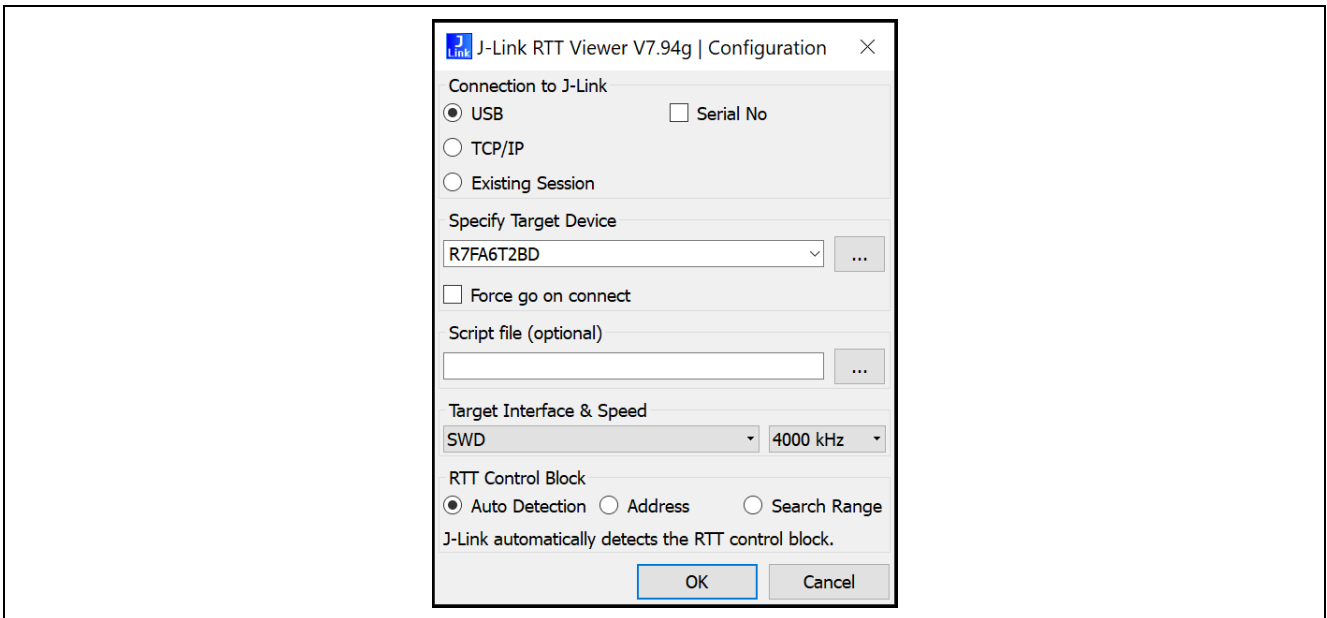

Figure 22. Connect RTT Viewer

In e² studio, click the **Resume** button [resume icon] twice to run the project.

The program will pause at a breakpoint at the end of `hal_entry()` once filter processing is complete and the output signal is ready to be exported from RAM. Leave the debug session running - do not stop the program.

4.3 Exporting Output Signal

This section explains how to export float array in memory to a binary file with the **Memory** debug view in e^2 studio.

Select **Window > Show View > Memory** to open the memory widget in e^2 studio.
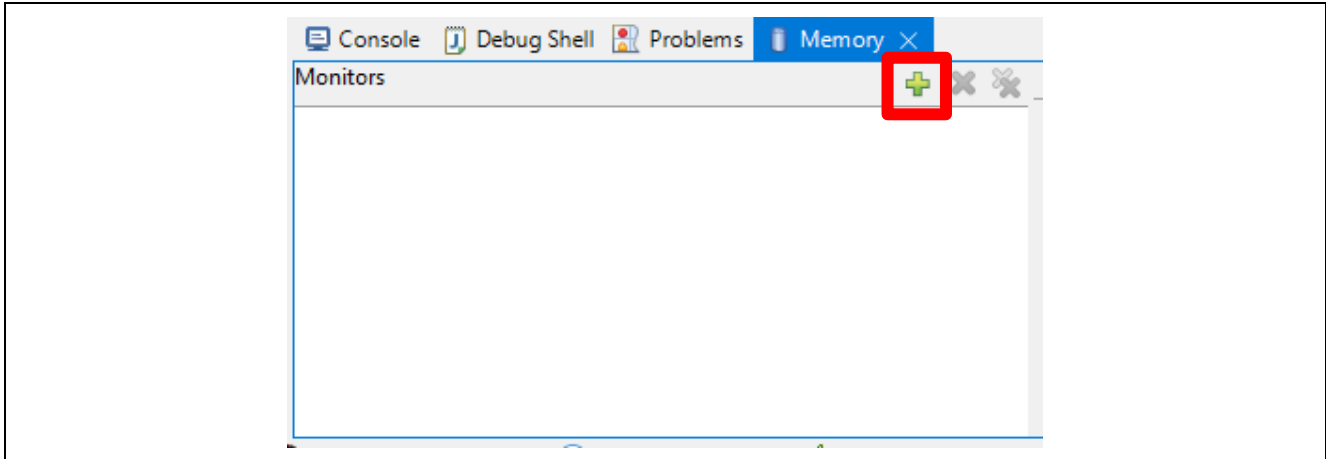
Click the **Add Monitor** button.

Figure 23.   Add a memory monitor

Type **&output** in the address/expression field and click **OK**.
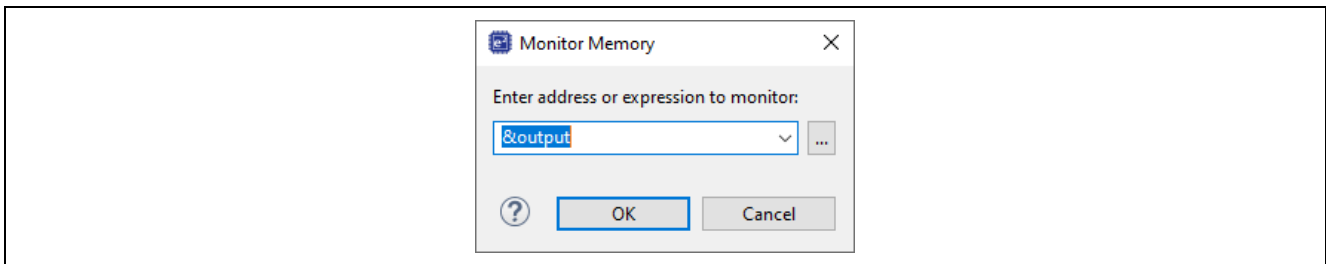
Figure 24.   Set the variable expression to monitor

Click the **Export Memory** button located in the **Memory View** toolbar.

Figure 25.   Export the memory

In the **Export Memory** pop up window, set the **Format** to **RAW Binary** and enter a byte **Length** of **16384** (the length of the **output** array is 4096*4 bytes = 16384 bytes).

Browse to the folder location that contains the MATLAB scripts for this project and save the file as `output_iirfa.bin`. Click **OK** to generate the binary file.

Figure 26.   Save as a RAW Binary file

4.4   Verification of Filter Operation

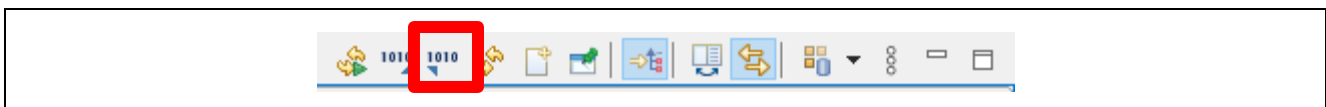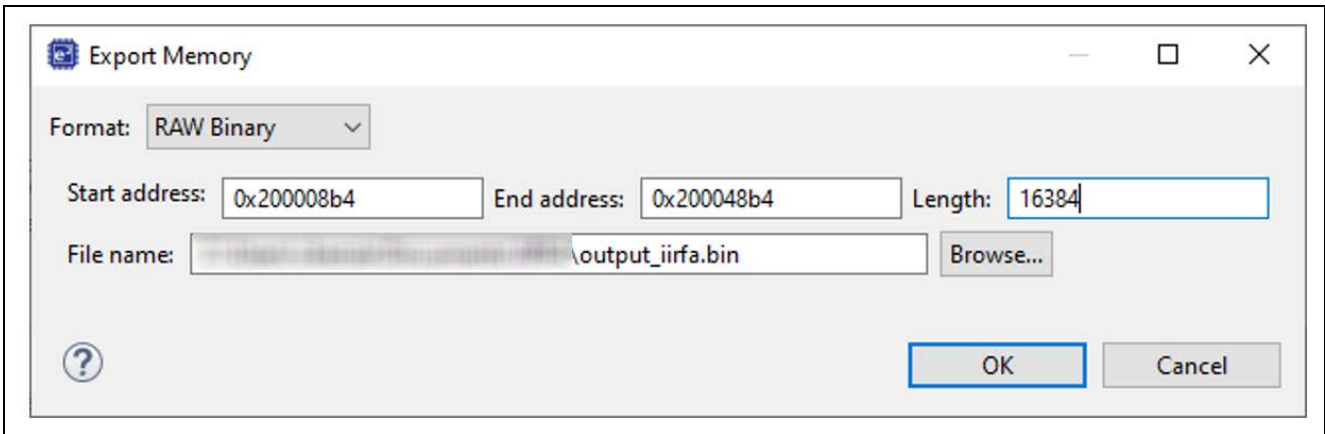This project comes with a script that uses the functionality of MATLAB to import and plot the output_iirfa.bin binary file for a visual verification of the filter operation in both the time and frequency domains.

Open verify_filter_performance.m with MATLAB.

Press **Run**.

Two figure windows will pop-up. MATLAB's figure 1 window, shown below as Figure 27, depicts the input and signals in the time domain. The top pane of figure 1 is a plot of the noisy input signal and the bottom pane is a plot of both the output of the IIRFA (solid blue line) and MATLAB (dashed red line).
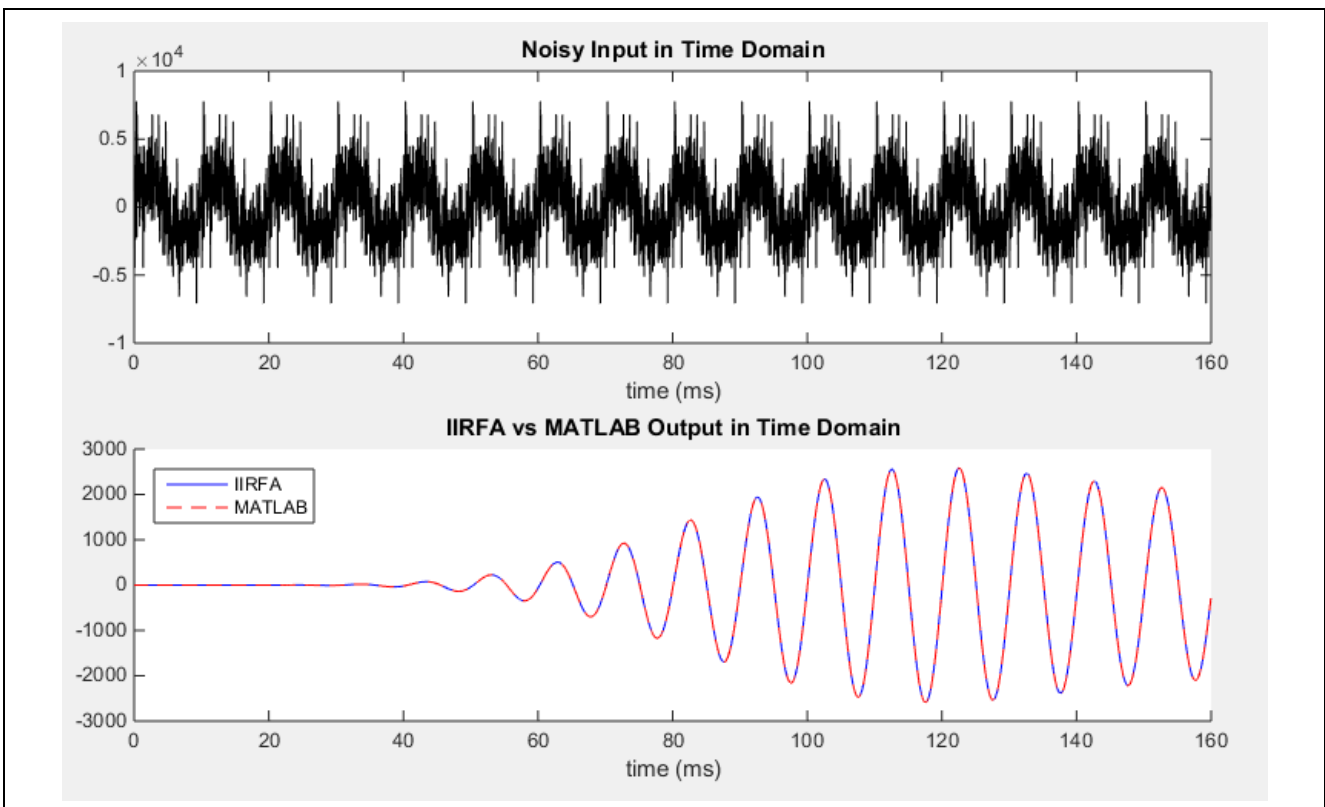
Figure 27.   Time-domain response of input and output signals of IIRFA and MATLAB

MATLAB's figure 2 window, shown below as Figure 28, is the single-sided frequency spectrum of the input signal and the output signal from the IIRFA filter processing. It depicts the peak amplitude of each sinusoidal component making up the time-domain signal, from 0 Hz to the Nyquist frequency at 12800 Hz.
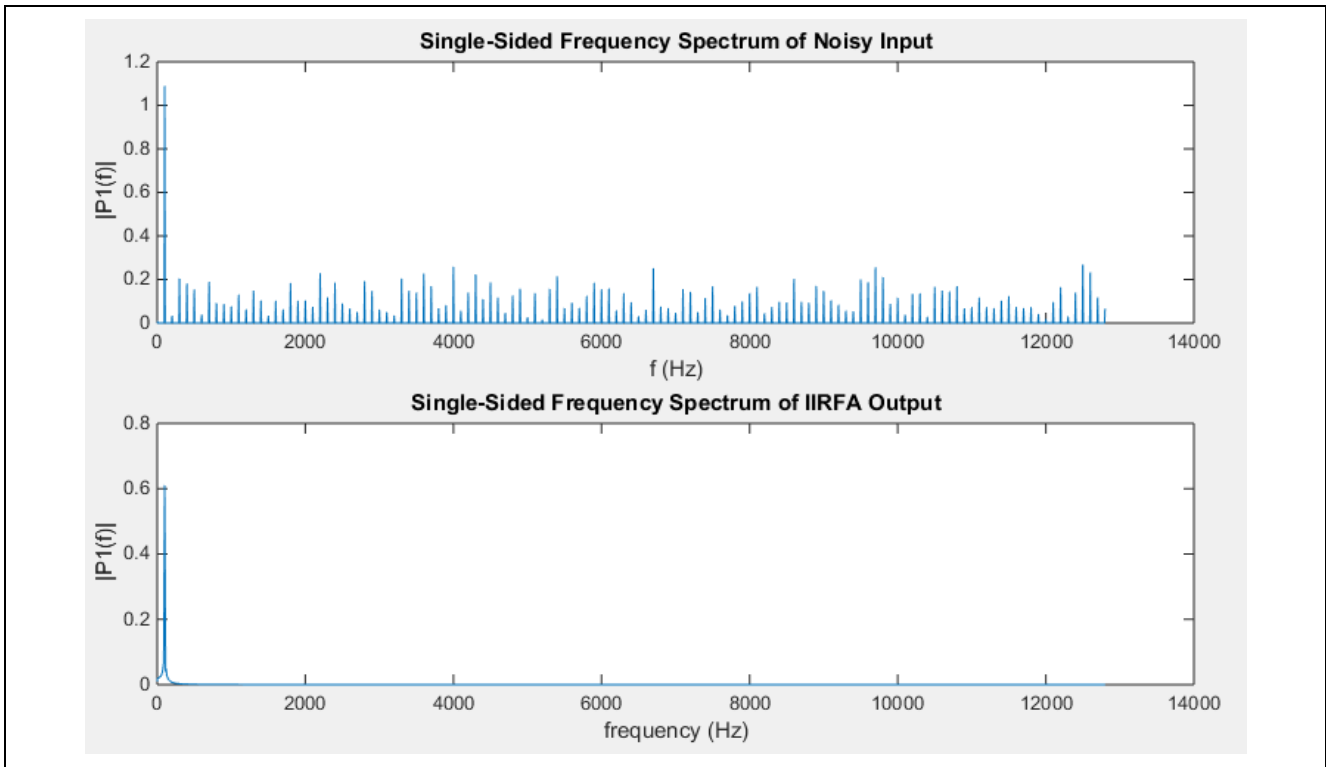
Figure 28.   Frequency-domain response of input and output of IIRFA

To zoom in, Click the **Zoom** button 🔍 in the top bar and draw a window around the region you would like to see in greater detail.
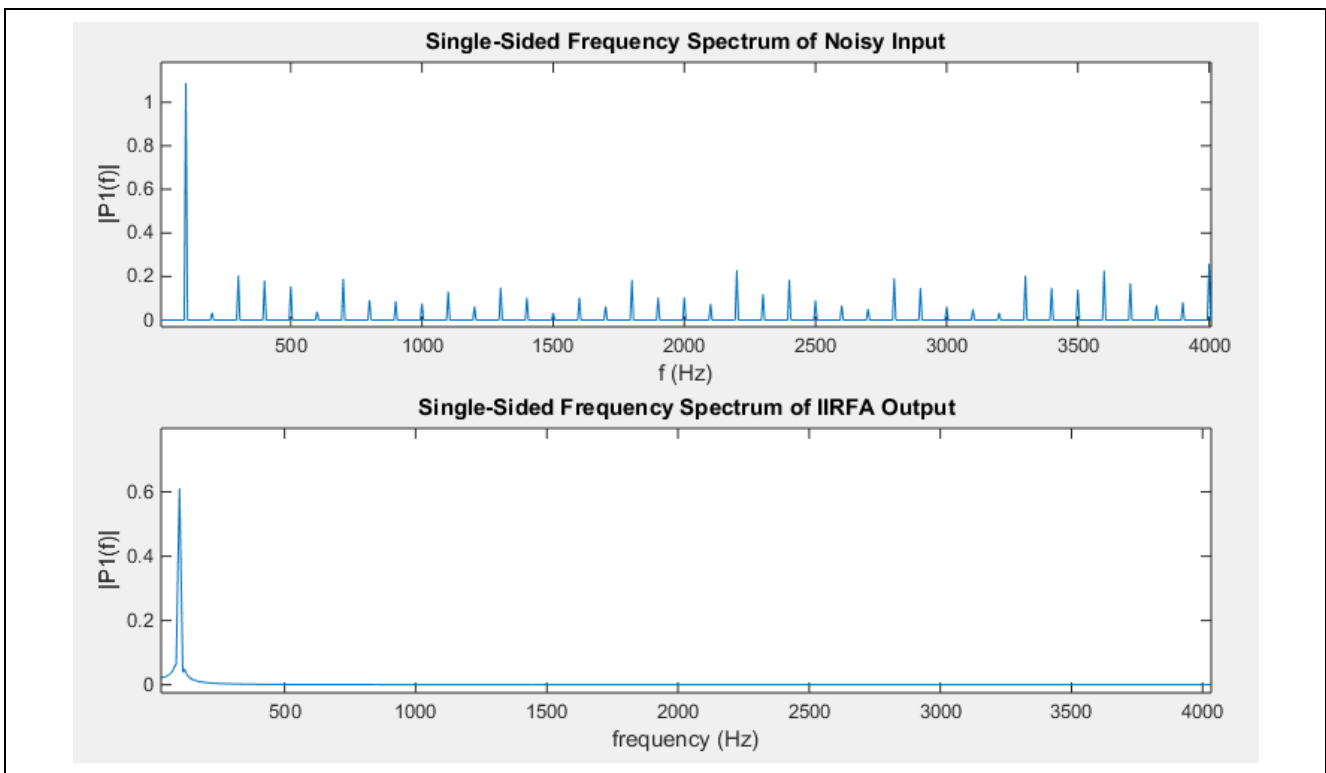
Figure 29.   Zoom view frequency-domain response of input and output of IIRFA

5. Next Steps and References

The latest versions of the following documents are available on the Renesas Electronics website.

- Refer to the following GitHub repository for various FSP modules example projects and application projects (https://github.com/renesas/ra-fsp-examples/).

- *RA6T2 Group User's Manual: Hardware* (R01UH0951).

- *FSP v5.2.0 User's Manual* (www.renesas.com/RA/FSP).

6.   References

- Renesas FSP User's Manual   renesas.github.io/fsp
- Renesas RA MCU datasheet   Select the relevant MCUs from the www.renesas.com/ra
- Example Projects   github.com/renesas/ra-fsp-examples

DSP IIR Topics

- *Signal Processing for Communications* by Paolo Prandoni and Martin Vetterli:
  https://www.sp4comm.org/webversion.html
- https://zone.ni.com/reference/en-XX/help/371361R-01/lvanlsconcepts/lvac_iir_filter_types/

Website and Support

Visit the following vanity URLs to learn about key elements of the RA family, download components and related documentation, and get support.

RA Product Information   www.renesas.com/ra
RA Product Support Forum   www.renesas.com/ra/forum
RA Flexible Software Package   www.renesas.com/FSP

Renesas Support   www.renesas.com/support

Revision History

		Description	
Rev.	**Date**	**Page**	**Summary**
1.00	Jun.07.22	—	First release document
1.01	Oct.13.23	—	Updated for FSP v4.4.0
1.02	May.31.24	—	Updated for FSP v5.2.0

General Precautions in the Handling of Microprocessing Unit and Microcontroller Unit Products

The following usage notes are applicable to all Microprocessing unit and Microcontroller unit products from Renesas. For detailed usage notes on the products covered by this document, refer to the relevant sections of the document as well as any technical updates that have been issued for the products.

1. Precaution against Electrostatic Discharge (ESD)

    A strong electrical field, when exposed to a CMOS device, can cause destruction of the gate oxide and ultimately degrade the device operation. Steps must be taken to stop the generation of static electricity as much as possible, and quickly dissipate it when it occurs. Environmental control must be adequate. When it is dry, a humidifier should be used. This is recommended to avoid using insulators that can easily build up static electricity. Semiconductor devices must be stored and transported in an anti-static container, static shielding bag or conductive material. All test and measurement tools including work benches and floors must be grounded. The operator must also be grounded using a wrist strap. Semiconductor devices must not be touched with bare hands. Similar precautions must be taken for printed circuit boards with mounted semiconductor devices.

2. Processing at power-on

    The state of the product is undefined at the time when power is supplied. The states of internal circuits in the LSI are indeterminate and the states of register settings and pins are undefined at the time when power is supplied. In a finished product where the reset signal is applied to the external reset pin, the states of pins are not guaranteed from the time when power is supplied until the reset process is completed. In a similar way, the states of pins in a product that is reset by an on-chip power-on reset function are not guaranteed from the time when power is supplied until the power reaches the level at which resetting is specified.

3. Input of signal during power-off state

    Do not input signals or an I/O pull-up power supply while the device is powered off. The current injection that results from input of such a signal or I/O pull-up power supply may cause malfunction and the abnormal current that passes in the device at this time may cause degradation of internal elements. Follow the guideline for input signal during power-off state as described in your product documentation.

4. Handling of unused pins

    Handle unused pins in accordance with the directions given under handling of unused pins in the manual. The input pins of CMOS products are generally in the high-impedance state. In operation with an unused pin in the open-circuit state, extra electromagnetic noise is induced in the vicinity of the LSI, an associated shoot-through current flows internally, and malfunctions occur due to the false recognition of the pin state as an input signal become possible.

5. Clock signals

    After applying a reset, only release the reset line after the operating clock signal becomes stable. When switching the clock signal during program execution, wait until the target clock signal is stabilized. When the clock signal is generated with an external resonator or from an external oscillator during a reset, ensure that the reset line is only released after full stabilization of the clock signal. Additionally, when switching to a clock signal produced with an external resonator or by an external oscillator while program execution is in progress, wait until the target clock signal is stable.

6. Voltage application waveform at input pin

    Waveform distortion due to input noise or a reflected wave may cause malfunction. If the input of the CMOS device stays in the area between V_{IL} (Max.) and V_{IH} (Min.) due to noise, for example, the device may malfunction. Take care to prevent chattering noise from entering the device when the input level is fixed, and also in the transition period when the input level passes through the area between V_{IL} (Max.) and V_{IH} (Min.).

7. Prohibition of access to reserved addresses

    Access to reserved addresses is prohibited. The reserved addresses are provided for possible future expansion of functions. Do not access these addresses as the correct operation of the LSI is not guaranteed.

8. Differences between products

    Before changing from one product to another, for example to a product with a different part number, confirm that the change will not lead to problems. The characteristics of a microprocessing unit or microcontroller unit products in the same group but having a different part number might differ in terms of internal memory capacity, layout pattern, and other factors, which can affect the ranges of electrical characteristics, such as characteristic values, operating margins, immunity to noise, and amount of radiated noise. When changing to a product with a different part number, implement a system-evaluation test for the given product.

Notice

1. Descriptions of circuits, software and other related information in this document are provided only to illustrate the operation of semiconductor products and application examples. You are fully responsible for the incorporation or any other use of the circuits, software, and information in the design of your product or system. Renesas Electronics disclaims any and all liability for any losses and damages incurred by you or third parties arising from the use of these circuits, software, or information.

2. Renesas Electronics hereby expressly disclaims any warranties against and liability for infringement or any other claims involving patents, copyrights, or other intellectual property rights of third parties, by or arising from the use of Renesas Electronics products or technical information described in this document, including but not limited to, the product data, drawings, charts, programs, algorithms, and application examples.

3. No license, express, implied or otherwise, is granted hereby under any patents, copyrights or other intellectual property rights of Renesas Electronics or others.

4. You shall be responsible for determining what licenses are required from any third parties, and obtaining such licenses for the lawful import, export, manufacture, sales, utilization, distribution or other disposal of any products incorporating Renesas Electronics products, if required.

5. You shall not alter, modify, copy, or reverse engineer any Renesas Electronics product, whether in whole or in part. Renesas Electronics disclaims any and all liability for any losses or damages incurred by you or third parties arising from such alteration, modification, copying or reverse engineering.

6. Renesas Electronics products are classified according to the following two quality grades: "Standard" and "High Quality". The intended applications for each Renesas Electronics product depends on the product's quality grade, as indicated below.

    "Standard": Computers; office equipment; communications equipment; test and measurement equipment; audio and visual equipment; home electronic appliances; machine tools; personal electronic equipment; industrial robots; etc.

    "High Quality": Transportation equipment (automobiles, trains, ships, etc.); traffic control (traffic lights); large-scale communication equipment; key financial terminal systems; safety control equipment; etc.

    Unless expressly designated as a high reliability product or a product for harsh environments in a Renesas Electronics data sheet or other Renesas Electronics document, Renesas Electronics products are not intended or authorized for use in products or systems that may pose a direct threat to human life or bodily injury (artificial life support devices or systems; surgical implantations; etc.), or may cause serious property damage (space system; undersea repeaters; nuclear power control systems; aircraft control systems; key plant systems; military equipment; etc.). Renesas Electronics disclaims any and all liability for any damages or losses incurred by you or any third parties arising from the use of any Renesas Electronics product that is inconsistent with any Renesas Electronics data sheet, user's manual or other Renesas Electronics document.

7. No semiconductor product is absolutely secure. Notwithstanding any security measures or features that may be implemented in Renesas Electronics hardware or software products, Renesas Electronics shall have absolutely no liability arising out of any vulnerability or security breach, including but not limited to any unauthorized access to or use of a Renesas Electronics product or a system that uses a Renesas Electronics product. RENESAS ELECTRONICS DOES NOT WARRANT OR GUARANTEE THAT RENESAS ELECTRONICS PRODUCTS, OR ANY SYSTEMS CREATED USING RENESAS ELECTRONICS PRODUCTS WILL BE INVULNERABLE OR FREE FROM CORRUPTION, ATTACK, VIRUSES, INTERFERENCE, HACKING, DATA LOSS OR THEFT, OR OTHER SECURITY INTRUSION ("Vulnerability Issues"). RENESAS ELECTRONICS DISCLAIMS ANY AND ALL RESPONSIBILITY OR LIABILITY ARISING FROM OR RELATED TO ANY VULNERABILITY ISSUES. FURTHERMORE, TO THE EXTENT PERMITTED BY APPLICABLE LAW, RENESAS ELECTRONICS DISCLAIMS ANY AND ALL WARRANTIES, EXPRESS OR IMPLIED, WITH RESPECT TO THIS DOCUMENT AND ANY RELATED OR ACCOMPANYING SOFTWARE OR HARDWARE, INCLUDING BUT NOT LIMITED TO THE IMPLIED WARRANTIES OF MERCHANTABILITY, OR FITNESS FOR A PARTICULAR PURPOSE.

8. When using Renesas Electronics products, refer to the latest product information (data sheets, user's manuals, application notes, "General Notes for Handling and Using Semiconductor Devices" in the reliability handbook, etc.), and ensure that usage conditions are within the ranges specified by Renesas Electronics with respect to maximum ratings, operating power supply voltage range, heat dissipation characteristics, installation, etc. Renesas Electronics disclaims any and all liability for any malfunctions, failure or accident arising out of the use of Renesas Electronics products outside of such specified ranges.

9. Although Renesas Electronics endeavors to improve the quality and reliability of Renesas Electronics products, semiconductor products have specific characteristics, such as the occurrence of failure at a certain rate and malfunctions under certain use conditions. Unless designated as a high reliability product or a product for harsh environments in a Renesas Electronics data sheet or other Renesas Electronics document, Renesas Electronics products are not subject to radiation resistance design. You are responsible for implementing safety measures to guard against the possibility of bodily injury, injury or damage caused by fire, and/or danger to the public in the event of a failure or malfunction of Renesas Electronics products, such as safety design for hardware and software, including but not limited to redundancy, fire control and malfunction prevention, appropriate treatment for aging degradation or any other appropriate measures. Because the evaluation of microcomputer software alone is very difficult and impractical, you are responsible for evaluating the safety of the final products or systems manufactured by you.

10. Please contact a Renesas Electronics sales office for details as to environmental matters such as the environmental compatibility of each Renesas Electronics product. You are responsible for carefully and sufficiently investigating applicable laws and regulations that regulate the inclusion or use of controlled substances, including without limitation, the EU RoHS Directive, and using Renesas Electronics products in compliance with all these applicable laws and regulations. Renesas Electronics disclaims any and all liability for damages or losses occurring as a result of your noncompliance with applicable laws and regulations.

11. Renesas Electronics products and technologies shall not be used for or incorporated into any products or systems whose manufacture, use, or sale is prohibited under any applicable domestic or foreign laws or regulations. You shall comply with any applicable export control laws and regulations promulgated and administered by the governments of any countries asserting jurisdiction over the parties or transactions.

12. It is the responsibility of the buyer or distributor of Renesas Electronics products, or any other party who distributes, disposes of, or otherwise sells or transfers the product to a third party, to notify such third party in advance of the contents and conditions set forth in this document.

13. This document shall not be reprinted, reproduced or duplicated in any form, in whole or in part, without prior written consent of Renesas Electronics.

14. Please contact a Renesas Electronics sales office if you have any questions regarding the information contained in this document or Renesas Electronics products.

(Note1) "Renesas Electronics" as used in this document means Renesas Electronics Corporation and also includes its directly or indirectly controlled subsidiaries.

(Note2) "Renesas Electronics product(s)" means any product developed or manufactured by or for Renesas Electronics.

(Rev.5.0-1 October 2020)

Corporate Headquarters

TOYOSU FORESIA, 3-2-24 Toyosu,
Koto-ku, Tokyo 135-0061, Japan
www.renesas.com

Trademarks

Renesas and the Renesas logo are trademarks of Renesas Electronics Corporation. All trademarks and registered trademarks are the property of their respective owners.

Contact information

For further information on a product, technology, the most up-to-date version of a document, or your nearest sales office, please visit:
www.renesas.com/contact/.