Application Note, V1.0, March 2008

AP16124

XC2000 Family

DSP Examples for C166S V2 Lib

16bit

Microcontrollers

Never stop thinking

Edition 2008-03

Published by
Infineon Technologies AG
81726 Munich, Germany

LEGAL DISCLAIMER

Information

For further information on technology, delivery terms and conditions and prices please contact your nearest Infineon Technologies Office (**www.infineon.com**).

Warnings

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

Infineon Technologies Components may be used in life-support devices or systems only with the express written approval of Infineon Technologies, if a failure of such components can reasonably be expected to cause the failure of that life-support device or system, or to affect the safety or effectiveness of that device or system. Life support devices or systems are intended to be implanted in the human body, or to support and/or maintain and sustain and/or protect human life. If they fail, it is reasonable to assume that the health of the user or other persons may be endangered.

XC2000 Family

Revision History: V1.0, 2008-03

Previous Version(s):
none

Page	Subjects (major changes since last revision)

We Listen to Your Comments
Any information within this document that you feel is wrong, unclear or missing at all? Your feedback will help us to continuously improve the quality of this document. Please send your proposal (including a reference to this document) to:
mcdocu.comments@infineon.com

1 Overview

The C166S V2 Lib, a DSP Library for C166S V2 Core is developed for the microcontroller family XC2000 with MAC unit. The Library consists of C-callable, hand-coded assembly, general purpose signal processing routines. The throughput of the system using the C166S V2 Lib routines is considerably better than those achieved using the equivalent code written in ANSI C language. Therefore it is important to understand the differences and requirements of the functions in each category. This application note presents the usage of finite impulse response (FIR), infinite impulse response (IIR), and fast Fourier transform (FFT). The performance of the DSP libraries in two different compilers (Tasking classic and Tasking VX) is also compared by placing the code in PSRAM and flash.

1.1 XC2000 Microcontroller family with MAC Unit

XC2000 is a new Infineon microcontroller family with MAC unit. It has C166S V2 Core with powerful on-chip peripheral subsystems (for example MultiCAN) and memory units (e.g. embedded Flash). The C166S V2 core is a well-known 16 bit core with two instruction sets. A normal instruction set and a MAC (multiply and accumulate) instruction set. The normal instruction set is used for the general control tasks, and the MAC instruction set is dedicated for DSP and 32 bit mathematical operations, such as filter, FFT. The MAC unit enhances strongly the DSP performance and 32 bit data processing capability of XC2000. To help users shorten the time-to-market in system development, Infineon provides a set of assembly-optimized functions, named C166S V2 lib. Each function in the C166S V2 Lib is designed to produce the best performance by using the MAC instructions. More details on MAC operations is described in "XC2000 family DSP optimization guide for XC2000 and XC166 microcontroller families with MAC unit (AP16113)"

The C166S V2 Lib is further based on the input parameter conditions, to provide parameter-specific optimal performance. Due to the parameter specifics, it is important to understand the differences and requirements of the functions in each category. It is also important to understand potential overhead related to memory hierarchy, to estimate and improve the actual performance of a system being developed.

2 Cycle Measurement

A XC2287-96F66HL easy kit board is used to measure the cycle count. Table 1 lists the key features of XC2287 which is important for performance measurement and optimization. More details on the architecture can be found in "User's Manual, XC2000 derivatives volume1 (of 2): System Units"

Table 1 Features of XC2287

Content	Description
Microcontroller	XC2287-96F66HL
Clock Frequency	66MHz
DPRAM(2Kb Dual port RAM)	Single cycle data access
DSRAM(16Kb data SRAM)	Single cycle data access
PSRAM(64Kb program SRAM)	Single cycle code access
Flash(12 segments - 768KB)	Code access with 4 wait states
Instruction pipeline	5 Stages processing pipeline (plus 2 stages fetch pipeline)

The built-in GPT12E unit is used to measure cycle counts for the DSP examples. The following sample code shows how to measure the cycle using GPT12E

```
/* Computation of overhead*/
GPT1_vStartTmr_GPT1_TIMER_3();
GPT1_vStopTmr_GPT1_TIMER_3();
Overhead_Time = GPT12E_T3;

/*DSP function under test*/
GPT1_vStartTmr_GPT1_TIMER_3();
Func_test ();
GPT1_vStopTmr_GPT1_TIMER_3();

/*Concatenation of two 16 bit timers to form a 32 bit timer*/
Timer_Low = GPT12E_T3;
Timer_high = GPT12E_T2;
cycle_count = 4*((Timer_high*65536+Timer_Low)-Overhead_Time)
```

The Func_test time is subtracted from the overhead time to get the actual cycle count. The maximum resolution of the GPT12E in XC2287 is 4 CPU cycles since the input to the timer is fixed to the CPU clock, divided by four. Overhead time will vary depending on the function parameters for different functions. The compilers used areTaskingv8.6r2 and Tasking VX v2.1r2

2.1 Memory overhead and Expected performance

The performance of the function is measured by placing the code in two different program memories

- Code in Internal flash
- Code in PSRAM

2.1.1 Code in Internal flash

Flash access wait states affect only non-sequential access.The flash memory in XC2000 requests 4 wait states for system frequency over 20 MHz. The bus width of XC2000 flash is 64 bits, which means that in each time 64 bits data can be read without wait states. For MAC instruction 64 bits correspond to 2 MAC instructions and 4 normal instructions because each MAC instruction has 4 bytes code size, and most of the normal instruction has only 2 bytes. If the code is executed from flash in XC2000 microcontroller, every 2 CoXXX instructions or every 4 normal instructions could have 4 cycles delay. We can use these 4 cycles for the delay caused by pipeline conflict.

```
Example:

Assembly without optimization (12 cycles)
.............
ADD R1,#2
MOV IDX0,R1
CoMAC [IDX0], [R0] ; 3 cycle stall
MOV R3, R1 ; 4 cycle waitstates
ADD R2, R3
.............

Assembly with optimization (5 cycles)
.............
MOV R3, R1
ADD R2, R3
MOV IDX0, R1 ; 3 cycle stall in 4 cycle waitstates
CoMAC [IDX0], [R0] ; 1 cycle delay
.............
```

2.1.2 Code in PSRAM

XC2000 family has 64 Kbytes PSRAM that can be used to store codes and dates. PSRAM has no waitstates. So, we can locate the DSP routines in PSRAM to get better performance in comparison with flash. In general, the code in PSRAM is two times faster than in flash for XC2000 family.

2.1.3 Data in DPRAM

The MAC unit on XC2000 and XC166 microcontroller has a Harvard architecture implementation, which means that every CPU-cycle allows one opcode fetch, two operand reads and one optional operand write. The XC2000 and XC166 microcontrollers are designed with Von Neumann architecture, where the memory space in is unified and the code and data share the same linear addressing space. In order to make it possible to fetch the code and data in one cycle a specific addressing mode is used in the MAC unit. When the MAC instruction needs two operands, one of them must be located in DPRAM. FIR and IIR filters in the C166S V2 Lib assume that one operand is stored in DPRAM.

3 Usage and performance of C166S V2 Lib

This section presents the usage and performance of Finite impulse response filter (FIR), Infinite impulse response filter (IIR) and Fast Fourier transform (FFT)

3.1 Finite Impulse response (FIR) Filters

The general form for a linear time-invariant FIR system's output y(n) at time n is given by

$$y(n) = \sum_{i=0}^{N-1} h_i \cdot x(n-i)$$

The C166S V2 DSPLib provides three FIR functions

- Fir_16 (16 bit coefficients, sample processing)
- Fir_32 (32 bit coefficients, sample processing)
- Fir_16_Blk (16 bit coefficients, Block processing)

The prototype and the assumptions for the FIR filter are as follows:

- **Fir_16** Short Fir_16 (short* h, short* IN, short N_h, short* D_buffer

Where,

IN points to the input sample in 1Q15 format

h points to the filter coefficients in 1Q15 format. The coefficients should be placed in reverse order

N_h is the filter order

D_buffer is the delay buffer. Delay buffer should be placed in the DPRAM area (0xf200 - 0xfe00).

Filter output is in 1Q15 format

- **Fir_32** Short Fir_32 (short* h, short* IN, short N_h, short* D_buffer)

Where,

IN points to the input sample in 1Q15 format

h points to the filter coefficients in 1Q31 format. The coefficients should be placed in reverse order

N_h is the filter order

D_buffer is the delay buffer. Delay buffer should be placed in the DPRAM area (0xf200 - 0xfe00).

Filter output is in 1Q15 format

- **Fir_16_Blk** void Fir_16_Blk(short* h, short* IN, short* R, short* D_buffer, short N_h, short N_x)

Where,

IN points to the input sample in 1Q15 format

h points to the filter coefficients in 1Q31 format. The coefficients should be placed in reverse order

N_h is the filter order

N_x Length of the new input sample

R pointer to filter output in 1Q15 format

D_buffer is the delay buffer. Delay buffer should be placed in the DPRAM area (0xf200 - 0xfe00)

3.1.1 FIR Filter Example

This example demonstrates the use of C166S V2 DSP Library FIR filtering capabilities. The input sequence used to test the FIR filter (Fir_16) is derived as follows:

X[i] = K * [sin (2*pi*f1*i/Fs) + sin (2*pi*f2*i/Fs)]

Where 'K' is the scale factor which depends on the filter coefficients and should be adjusted to prevent the overflow, f1 and f2 are the input frequencies, Fs is the sampling frequency. In this particular example 'K' is taken as 0.25, f1 is 370 Hz and f2 is 18500 Hz, It is sampled (Fs) at 44100 Hz. Later, the quantized filter coefficients(16 bits) with design specifications as mentioned in **Table 2** are generated in matlab using filter design and analysis tool (Matlab command: fdatool). The coefficients of the filter in 1Q15 format are listed in **Appendix** . The corresponding frequency response is shown in **Figure 1**. Analysis of finite word length effects for this case showed that the input quantization noise and the deviation in the frequency response due to coefficient quantization are both insignificant. **Figure 2** shows the input with frequencies f1 and f2 (top graph) and output (bottom graph) of the FIR filter. Frequency f2 (18500Hz) is attenuated since it is

above the cutoff frequency and only f1 (370Hz) sinusoidal remains as shown in **Figure 2** (bottom graph).

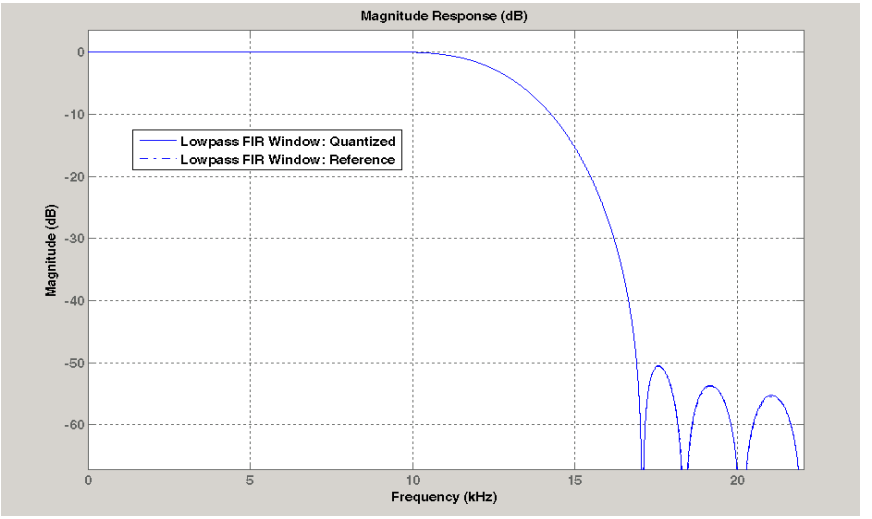

Figure 1 Frequency response of a low pass FIR filter

The output of the FIR filter is delayed from the input by a certain number of samples. This is due to the fact that, fewer delay lines in the FIR filter structure has to be activated in order to get the expected output. Higher delay is expected for FIR filters of high order.

Table 2 FIR Filter design Specifications

Content	Description
Filter Type	Low-Pass
Order	19
Design Method	Window (Kaiser)
Sampling Frequency	44,100 Hz
Pass band/Stop band frequency	1000/17000 Hz
Pass band/Stop band Attenuation	1/50 Hz

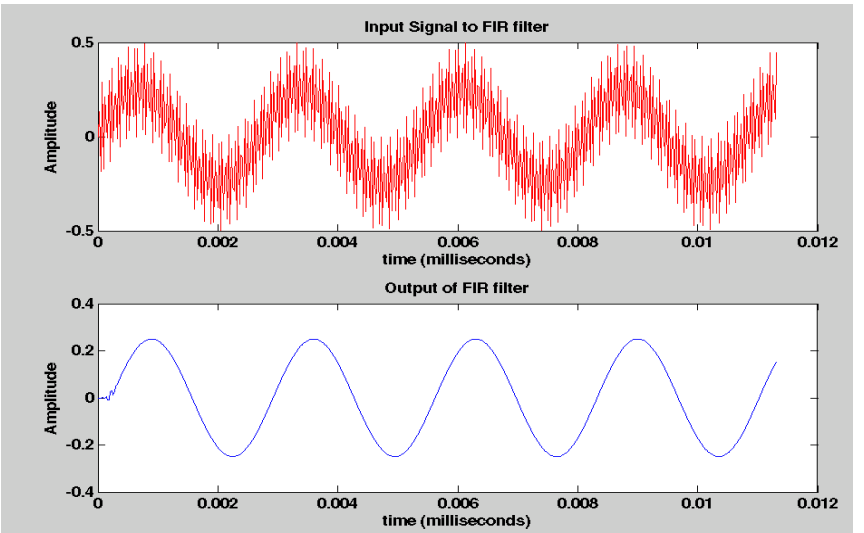

Figure 2 FIR filter Pass band Input (above) and output (below)

Table 3 lists the performance of the three FIR functions. Coefficients quantized to 16 bit are sufficient for this particular case. Therefore increasing the coefficient wordlength to 32 bits [using Fir_32] does not provide significant changes in the output. It is better to use Fir_16_blk (block processing) instead of Fir_16 and Fir_32(sample processing), since the function call overhead is less in Fir_16_blk (block processing) compared to Fir_16 and Fir_32 (sample processing). Visually, the output of the Fir_32 filter is the same as that of Fir_16 filter and is therefore not shown.

Table 3 FIR filter Benchmarks (Tasking Classic)

Functions	No of cycles (20 filter coefficients, 500 Input samples)	
	Flash	PSRAM
Fir_16	67992	34508
Fir_32	105000	56008
Fir_16_blk	36592	18056

Table 4 FIR filter Benchmarks (Tasking VX)

Functions	No of cycles (20 filter coefficients, 500 Input samples)	
	Flash	PSRAM
Fir_16	67992	32004

Functions	No of cycles (20 filter coefficients, 500 Input samples)	
	Flash	PSRAM
Fir_32	102496	52996
Fir_16_blk	36576	18048

Note: The cycle count is measured by including the function call overhead.

The performance of the libraries in Tasking VX compiler is better compared to Tasking classic due to the following reasons:

- The code generated by Tasking VX for function call is better compared to Tasking classic.
- Tasking VX uses 8 CPU registers for function parameters whereas Tasking classic uses only 4 CPU registers for function parameters.

3.2 Infinite Impulse response (IIR) filters

The general form for an infinite impulse response (IIR) filter's output yen) at time n is given by

$$y(n) = \sum_{i=1}^{N} a(i-1) \cdot y(n-i) + \sum_{i=0}^{M} b(i) \cdot x(n-i)$$

Where, a(i) and b(i) are filter coefficients.

IIR filters generally have non linear phase responses, but can meet magnitude response in much lower order compared to FIR filters. However due to the nature of instability care must be taken to design quantized filter coefficients.

The C166S V2 DSP Lib provides five IIR functions

- IIR_1 (16 bit filter coefficients, Direct form 1, sample processing)
- IIR_2 (16 bit filter coefficients, Direct form 2, sample processing)
- IIR_bi_1 (Biquad, Direct form 1, 16 bit coefficients, sample processing)
- IIR_bi_2 (Biquad, Direct form 2, 16 bit coefficients, sample processing)
- IIR_bi_2_Blk (Biquad, Direct form 2, 16 bit coefficients, Block processing)

The prototype and the assumptions for the IIR filter are as follows:

- **IIR_1** Short IIR_1 (short* h, short* IN, short N, short x_y)

Where,

IN points to the input sample in 1Q15 format

h points to the filter coefficients in 2Q14 format. The coefficients should be placed in reverse order. Coefficients a(i) should be followed by coefficients b(i)

N is the filter order

x_y is the pointer to the delay buffer. The delay buffer should be located in DPRAM

Filter output is in 1Q15 format

- **IIR_2** Short IIR_2 (short* h, short* IN, short N, short* u)

Where,

IN points to the input sample in 1Q15 format

h points to the filter coefficients in 2Q14 format. The coefficients should be placed in reverse order.

N is the filter order

u is the pointer to the state variable vector. The state variable vector should be located in DPRAM (0xf200 - 0xfe00).

Filter output is in 1Q15 format

- **IIR_bi_1** Short IIR_bi_1 (short* h, short* IN, short N, short* u_w)

Where,

IN points to the input sample in 1Q15 format

h points to the filter coefficients in 1Q15 format. The coefficients should be placed in reverse order.

N is the filter order

u_w is the pointer to the state variable vector. The state variable vector should be located in DPRAM (0xf200 - 0xfe00).

Filter output is in 1Q15 format

- **IIR_bi_2** Short IIR_bi_2 (short* h, short* IN, short N, short* u)

Where,

IN points to the input sample in 1Q15 format

h points to the filter coefficients in 1Q15 format. The coefficients should be placed in reverse order.

N is the filter order

u is the pointer to the state variable vector. The state variable vector should be located in DPRAM (0xf200 - 0xfe00).

Filter output is in 1Q15 format

- **IIR_bi_2_Blk** Short IIR_bi_2_Blk (short* h, short* IN, short N, short* u_w, short* y, short N_x)

Where,

IN points to the input sample in 1Q15 format

h points to the filter coefficients in 1Q15 format. The coefficients should be placed in reverse order.

N is the filter order

u_w is the pointer to the state variable vector. The state variable vector should be located in DPRAM (0xf200 - 0xfe00).

N_x Number of Input samples

y points to the filter output in 1Q15 format

3.2.1 IIR Filter Direct form-1 Example

This example demonstrates the use of C166S V2 DSP Library IIR filtering direct form-1 capabilities. First input data to the IIR filter is generated as follows:

X[i] = K * [sin (2*pi*f1*i/Fs) + sin (2*pi*f2*i/Fs)]

Where 'K' is the scale factor which depends on the filter coefficients and should be adjusted to prevent the overflow, f1 and f2 are the input frequencies, Fs is the sampling frequency. In this particular example 'K' is taken as 0.25, f1 is 370 Hz and f2 is 18500 Hz, It is sampled (Fs) at 44100 Hz. Later, the quantized filter coefficients (16 bits) with design specifications as mentioned in **Table 5** are generated in matlab using filter design and analysis tool (Matlab command: fdatool). The coefficients of the filter in 2Q14 format are listed in **Appendix** .The corresponding frequency response is shown in **Figure 3**. Analysis of finite word length effects for this case showed that the input quantization noise and the deviation in the frequency response due to coefficient quantization are both insignificant. **Figure 4** shows the input with frequencies f1 and f2 (top graph) and output (bottom graph) of the IIR filter. Frequency f2 (18500Hz) is attenuated since it is above the cutoff frequency and only f1 (370Hz) sinusoidal remains as shown in **Figure 4** (bottom graph). The attenuated frequency is well within the stop band frequency of the filter. Therefore we can observe some high frequency distortions at the output.

Table 5 IIR filter Direct form-1 design specifications

Content	Description
Filter Type	Low-Pass
Order	2
Design Method	Butterworth

Content	Description
Sampling Frequency	44,100 Hz
Pass band/Stop band frequency	8000/22000 Hz
Pass band/Stop band Attenuation	1/50 Hz

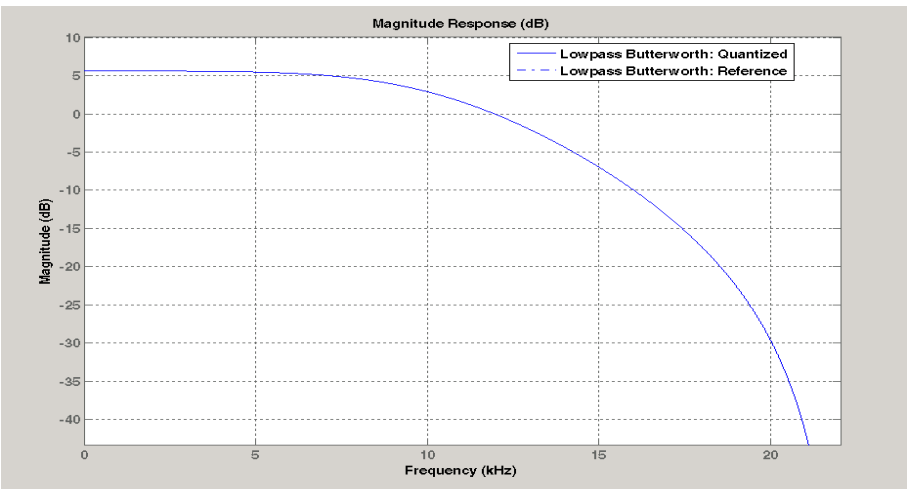

Figure 3 Frequency response of a Low pass IIR filter Direct form-1

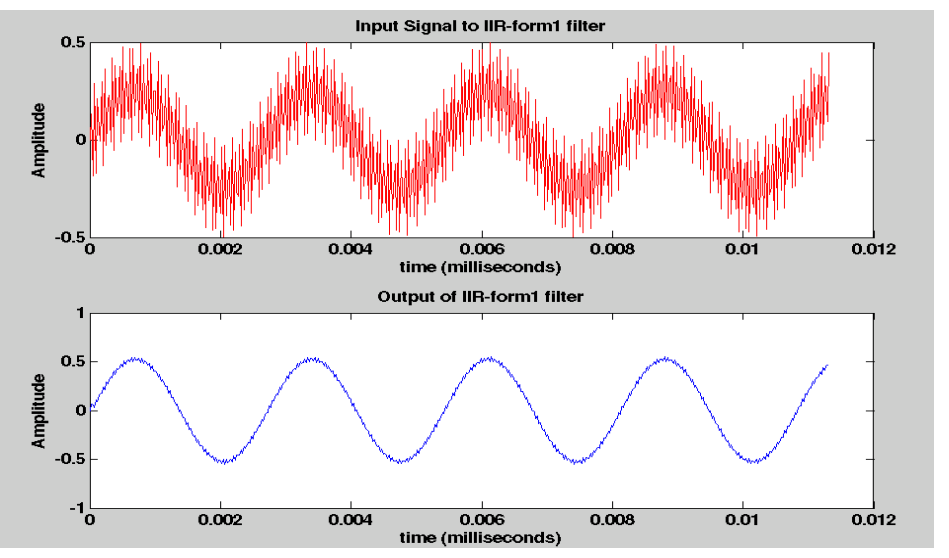

Figure 4 IIR Filter Direct form-1 Pass band Input (above) and Output (below)

3.2.2 IIR Filter Biquad Direct form-1 Example

This example demonstrates the use of C166S V2 DSP Library IIR filter Biquad Direct form-1 capabilities. First input data to the IIR filter is generated as follows:

X[i] = K * [sin (2*pi*f1*i/Fs) + sin (2*pi*f2*i/Fs)]

Where 'K' is the scale factor which depends on the filter coefficients and should be adjusted to prevent the overflow, f1 and f2 are the input frequencies, Fs is the sampling frequency. In this particular example 'K' is taken as 0.25, f1 is 370 Hz and f2 is 18500 Hz, It is sampled (Fs) at 44100 Hz. Later, the quantized filter coefficients(16 bits) with design specifications as mentioned in **Table 6** are generated in matlab using filter design and analysis tool (Matlab command: fdatool). The coefficients of the filter in 1Q15 format are listed in **Appendix** . The corresponding frequency response is shown in **Figure 5**. Analysis of finite word length effects for this case showed that the input quantization noise and the deviation in the frequency response due to coefficient quantization are both insignificant. **Figure 6** shows the input with frequencies f1 and f2 (top graph) and output (bottom graph) of the IIR filter. Frequency f2 (18500Hz) is attenuated since it is above the cutoff frequency and only f1 (370Hz) sinusoidal remains as shown in **Figure 6** (bottom graph).

Table 6 IIR Filter Biquad Direct form-1 Design Specifications

Content	Description
Filter Type	Low-Pass
Order	2
Design Method	Butterworth
Sampling Frequency	44,100 Hz
Pass band/Stop band frequency	8000/22000 Hz
Pass band/Stop band Attenuation	1/50 Hz

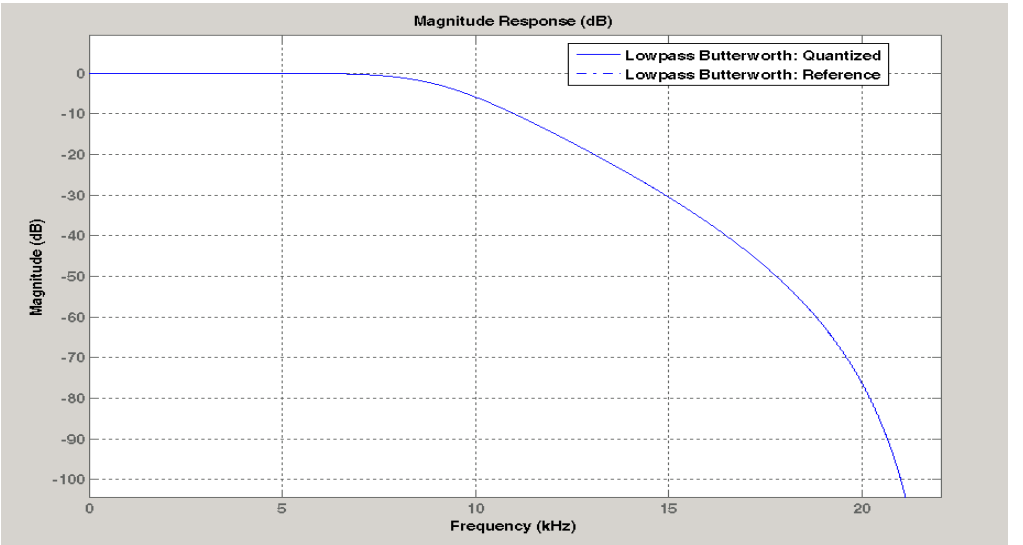

Figure 5 Frequency response of a Low pass IIR Filter Biquad Direct form - 1

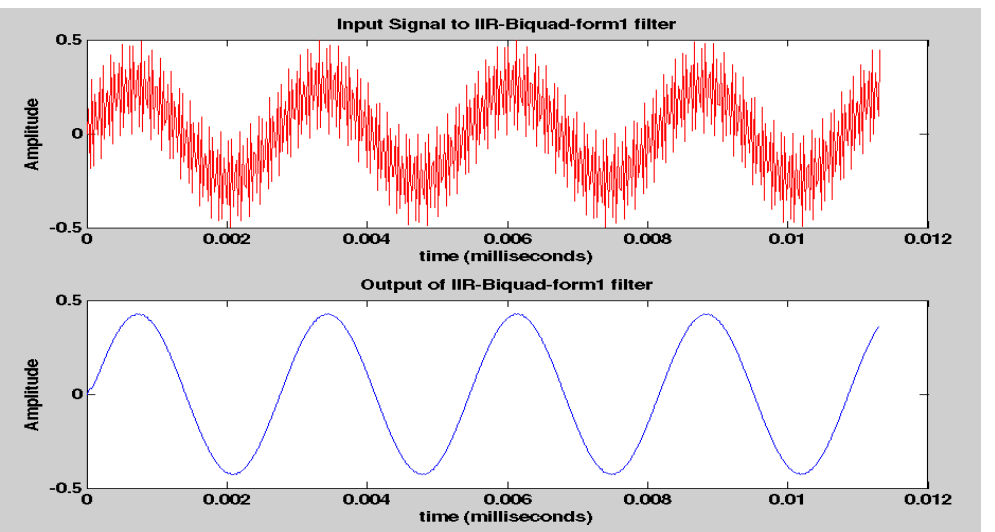

Figure 6 IIR Filter Biquad Direct form-1 Pass band Input (above) and Output (below)

Table 7 lists the performance of the five IIR functions

Table 7 IIR Filter Benchmarks (Tasking Classic)

Functions	No of cycles (20 filter coefficients, 500 Input samples)	
	Flash	PSRAM
IIR_1	78996	44496
IIR_2	74996	36008
IIR_bi_1	78492	49512
IIR_bi_2	84492	42012
IIR_bi_2_Blk	44592	25068

Table 8 IIR Filter Benchmarks (Tasking VX)

Functions	No of cycles (20 filter coefficients, 500 Input samples)	
	Flash	PSRAM
IIR_1	69492	31004
IIR_2	69980	32504
IIR_bi_1	74976	42504
IIR_bi_2	81488	38004
IIR_bi_2_Blk	42080	25048

Note: The cycle count is measured by including the function call overhead.

3.3 Fast Fourier Transform (FFT)

FFT is widely used for frequency-domain processing and spectrum analysis. It is a computationally efficient discrete Fourier transform (DFT) defined as:

$$X(k) = \sum_{n=0}^{N-1} x(n) W_N^{nk}$$

where,

$$W_N = e^{-j2\pi/N} = \cos(2\pi nk/N) - j\sin(2\pi nk/N)$$

The definitions and requirements of the FFT function is as follows,

void real_DIT_FFT (short* x, short* index, short exp, short* table, short* X)

Where,

x is the 16 bit real input vector

index is the bit reversed input index vector

exp is the exponent of the input block size

table is the precalculated trigonometric function (sinus and cosine) table

X is the FFT output vector in 1Q15 format

This function computes the N-point real forward radix-2 decimation-in-time Fast Fourier Transform on the given N point real input array. A separate bit reverse function is provided to convert the input samples to bit reversed order. The twiddle factors are stored in a lookup table [1Q15 format].The function is implemented as a complex FFT of size N/2 followed by a unpack stage to unpack the real FFT results.

Normally an FFT of a real sequence of size N produces a complex sequence of size N or 2N real numbers that will not fit in the input sequence. To accommodate all the results without requiring extra memory locations, the output reflects only half of the complex spectrum plus the spectrum at the nyquist point (N/2). This still provides the full information because an FFT of a real sequence has even symmetry around the nyquist point

3.3.1 DIT_FFT Examples

This example demonstrates the use of C166S V2 DSP Library Fast Fourier transform capabilities. First input data to the FFT is generated as follows

X[i] = K * [sin (2*pi*f1*i/Fs) + sin (2*pi*f2*i/Fs)] where 'K' is the scale factor which should be adjusted to prevent the overflow, f1 and f2 are the input frequencies, Fs is the sampling frequency. In this particular example 'K' is taken as 0.5, f1 is 370 Hz and f2 is 17500 Hz, It is sampled (Fs) at 44100 Hz.

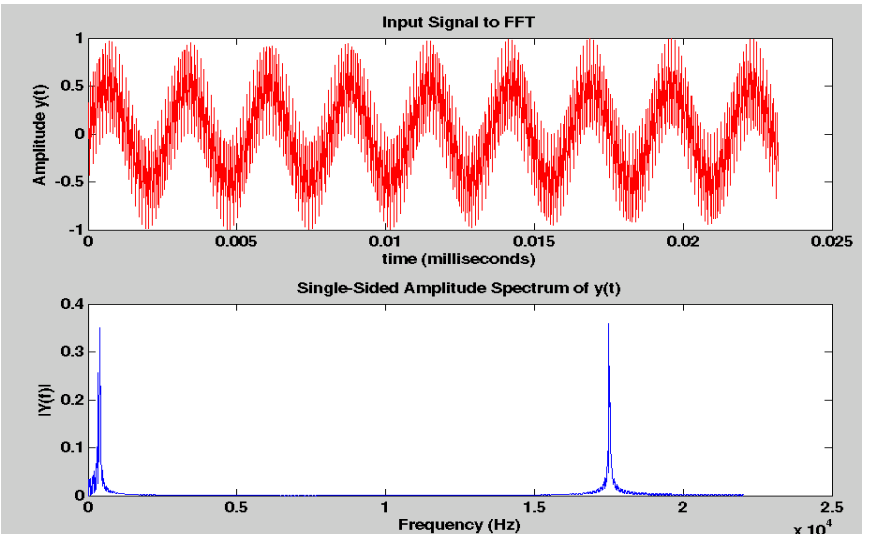

Figure 7 Input signal to FFT (above) and Magnitude of the output (below)

Figure 7 (top graph) shows the real part input of the FFT where N=1024 points. **Figure 7** (bottom graph) shows the magnitude of the output

In general the input signal has to be scaled down by 1+Sqrt(2) at each stage to get the desired output. However in real_DIT_FFT we are scaling the input by a factor of 2. Therefore there is no guarantee that overflow would never happen unless the input signal is sufficiently scaled down. For example if a Non-scaled DC signal is given as a input to the real_DIT_FFT we will not get the expected output as shown in **Figure 8**. However if we scale the input DC signal by a factor of 1.05 we can get the expected output as shown in **Figure 9**

Table 9 lists the performance of the FFT functions. The performance has to be calculated along with bit reversal function.

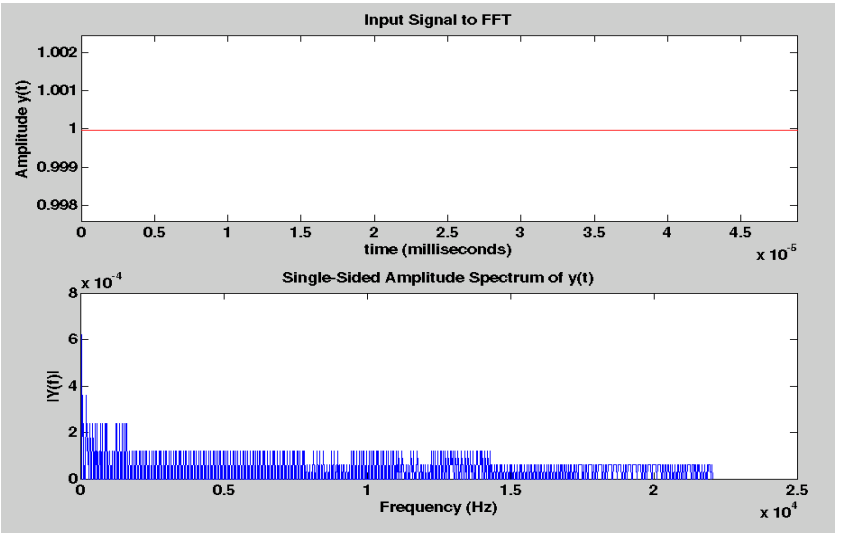

Figure 8 Input signal to FFT (Non-Scaled DC signal) and the Output

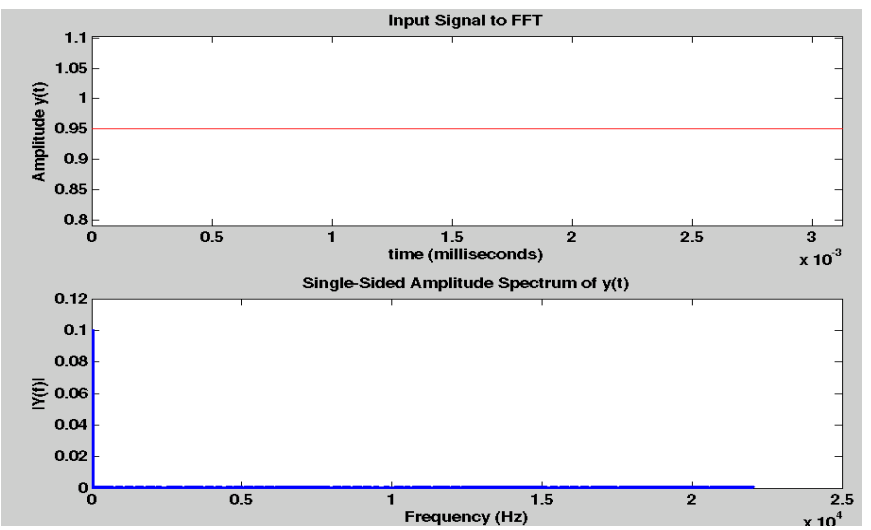

Figure 9 Input Signal to FFT(Scaled DC signal) and the Output

Table 9 FFT Benchmark (Tasking Classic)

Functions	No of cycles (1024 Input samples)	
	Flash	**PSRAM**
Real_DIT_FFT	267236	168060
Bit_reverse	27852	18024

Table 10 FFT Benchmark (Tasking VX)

Functions	No of cycles (1024 Input samples)	
	Flash	**PSRAM**
Real_DIT_FFT	265196	163444
Bit_reverse	24272	14444

Note: The cycle count is measured by including the function call overhead.

4 Conclusion

In this Application Note we have described the usage and performance of the key C166S V2 Lib signal processing routines. The provided information can help users to better utilize the library for their system development

5 Appendix

Table 11 FIR Coefficients for Table 2 Design spec

	Unquantized Coefficients	Quantized Coefficients [1Q15 Format]	
h(0)	-0.001007080078125	-34	h[18]
h(1)	-0.002899169921875	-96	h[17]
h(2)	0.00927734375	303	h[16]
h(3)	-0.0010986328125	-36	h[15]
h(4)	-0.025634765625	-841	h[14]
h(5)	0.030731201171875	1006	h[13]
h(6)	0.029876708984375	979	h[12]
h(7)	-0.1102294921875	-3613	h[11]
h(8)	0.051239013671875	1678	h[10]
h(9)	0.519744873046875	17030	h[9]

Table 12 IIR Direct Form -1 Coefficients for Table 4 design spec

Denominator(ak)		Numerator(bk)	
Unquantized Coefficients	Quantized Coefficients [2Q14 Format]	Unquantized Coefficients	Quantized Coefficients [2Q14 Format]
1.00	16384	0.5	8192
-0.1253662109375	-2054	1	16384
0.17437744140625	2857	0.5	8192

Table 13 IIR Biquad Direct form- 1(Section1) Coefficients for Table 5

Denominator(ak)		Numerator(bk)	
Unquantized Coefficients	Quantized Coefficients [1Q15 Format]	Unquantized Coefficients	Quantized Coefficients [1Q15 Format]
1.00	32767	0.283796874	9298
-0.393524096987	-12894	0.567593748	18599
0.461402860998	15120	0.283796874	9298

Table 14 IIR Biquad Direct form- 1(Section 2) Coefficients for Table 5 design

Denominator(ak)		Numerator(bk)	
Unquantized Coefficients	**Quantized Coefficients [1Q15 Format]**	**Unquantized Coefficients**	**Quantized Coefficients [1Q15 Format]**
1.00 -0.2849877434044 0.0583390108734	32767 -9338 1912	0.43066495102 0.86132990204 0.43066495102	14111 28224 1411

6 References

1. XC166Lib, A DSP Library for XC16x Family, Tasking compiler, V1.1 June 2003

2. XC2000 family DSP optimization guide for XC2000 and XC166 microcontroller families with MAC unit (AP16113)"

3. User's manual, XC2000 derivatives volume1 (of 2): System units

4.Tasking Application Note, Running code copied from ROM Memory to RAM Memory AN019-13 V8.0

5. SPRA 947, Signal processing examples using TMS320C67X Digital signal processing library

6. Ashok Ambardar, Analog and Digital Signal processing, 2nd edition, Thomson Brooks/cole publication

7. C.Britton, Rorabaugh, DSP Premier, Tata McGraw-Hill Edition

w w w . i n f i n e o n . c o m

Published by Infineon Technologies AG