
AVR32718: AT32UC3 Series Software Framework DSPLib

1. Introduction

This application note describes the DSP Library from the AVR32[®] Software Framework. It details the main functions (prototype, algorithm and benchmark) of the DSP library: FFT, convolution, FIR and partial IIR using GCC compiler.

All the source code (C code and assembly), software example and GCC and IAR projects are released in the AVR32 UC3 Software Framework.

1.1 References

- **AVR32 UC3 Software Framework:** This framework provides software drivers, libraries and application examples to build any application for AVR32 UC3 devices. http://www.atmel.com/dyn/products/tools_card.asp?tool_id=4192
- **AVR32 Architecture Manual:** http://www.atmel.com/dyn/resources/prod_documents/doc32000.pdf



32-bit **AVR[®]**
Microcontrollers

Application Note



2. Radix-4 decimate in time complex FFT

2.1 Description

This function computes a complex FFT from an input signal. It uses the Radix-4 “Decimate In Time” algorithm and does not perform a calculation “in place” which means that the input buffer has to be different from the output buffer. Function prototype

```
void dspXX_trans_complexfft(
    dspXX_complex_t *vect1,
    dspXX_t *vect2,
    int nlog);
```

where XX corresponds to the number of bits of a basic data element (i.e. 16 or 32).

2.1.1 Arguments

This function takes three parameters: the output buffer, the input buffer and a value corresponding to the size of those buffers.

The output buffer (vect1) is a pointer on a complex vector of 2^{nlog} elements.

The input buffer (vect2) is a pointer on a real vector of 2^{nlog} elements.

The size argument (nlog) is in fact the base-2-logarithm of the size of the input vector. (nlog fits in [2, 4, 6, ..., 28])

2.1.2 Algorithm

Following is the algorithm used to implement the radix-4 DIT complex FFT. The optimized version is based on this algorithm but can differ in certain points due to the instruction set of the target:

```
size = 1 << nlog
FOR r FROM 0 TO size-1 STEP 4 DO
    Butterfly_zero_only_real_and_bit_reversing(vect1, vect2, r)
END
FOR stage FROM 1 TO nlog/2 DO
    m = 4 ^ stage
    FOR r FROM 0 TO size-1 STEP m DO
        Butterfly_zero(vect1, r)
    END
    FOR j FROM 1 TO m / 4 - 1 DO
        Comput_twiddle_factors(e, e2, e3, j / m)
    FOR r FROM 0 TO size-1 STEP m DO
        Butterfly(vect1, r, j, e, e2, e3)
    END
END
END
END
```

2.1.3 Notes

Interruptibility: the code is interruptible.

In-place computation is not allowed.

This function uses a static twiddle factors table raw-coded in the file “*BASIC/TRANSFORMS/dspXX_twiddle_factors.h*”. To generate those factors, you can use the script called “*tf_gen.sci*” and execute it with Scilab.

To avoid overflowing values, the resulting vector amplitude is scaled by $2^{\text{^}n\log}$.

All the vectors have to be 32-bit aligned.

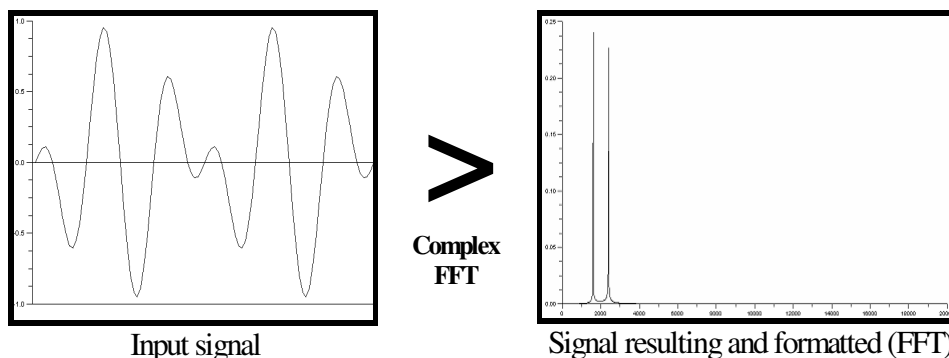
2.2 Benchmark

2.2.1 Benchmark routine

All these functions have been benchmarked on an avr32-uc3a0512 target. The programs have been compiled with avr32-gcc (4.0.2-atmel.1.0.0) with the $-O3$ optimization option and have been stored in FLASH memory. The fixed-point format used is the Q1.15 format for the 16-bit data and the Q1.31 format for the 32-bit data.

The benchmark process has been performed with the same input signal for all those functions and compared with a reference’s signal computed with a mathematic tool using floating point.

The input signal is a combination of one sine and one cosine. The sine oscillating at 400Hz and the cosine at 2KHz. Those signals have been multiplied and sampled at 40KHz.



2.2.2 Result

Here are tables of the main values of the benchmark results. All those values correspond to the best performances of the functions and are obtained with different compilation options. For more information, please refer to the complete benchmark result table in annexes.

2.2.2.1 16-bit radix-4 D.I.T. complex FFT: generic

Concerned file path: /BASIC/TRANSFORMS/dsp16_complex_fft_generic.c

	Lowest cycle count	Fastest computation at 60 MHz	Lowest Error		Lowest Algorithm's size in memory
			Amplitude average	Max. amplitude	
64-points	6,296	108.2us	1.58e-5	6.53e-5	1.1 Kbytes
256-points	33,723	578.0us	1.69e-5	8.80e-5	1.3 Kbytes
1024-points	169,006	2.90ms	1.67e-5	12.31e-4	2.0 Kbytes
4096-points	812,321	13.90ms	1.52e-5	14.60e-4	5.0 Kbytes

More details on Table 1.1.1 in annexes

2.2.2.2 16-bit radix-4 D.I.T. complex FFT: avr32-uc3 optimized

Concerned file path: /BASIC/TRANSFORMS/dsp16_complex_fft_avr32uc3.c

	Lowest cycle count	Fastest computation at 60 MHz	Lowest Error		Lowest Algorithm's size in memory
			Amplitude average	Max. amplitude	
64-points	2,611	44.4 us	1.63e-5	6.53e-5	710 bytes
256-points	13,661	232.2 us	1.68e-5	7.46e-5	902 bytes
1024-points	67,671	1.15 ms	1.69e-5	1.02e-4	1.6 Kbytes
4096-points	322,897	5.49 ms	1.58e-5	1.18e-4	4.6 Kbytes

Warning: this function is only compatible with Q1.15 numbers.

Note: this function needs 72 bytes of memory for the stack.

More details on Table 1.1.2 in annexes

2.2.2.3 32-bit radix-4 D.I.T. complex FFT: generic

Concerned file path: /BASIC/TRANSFORMS/dsp32_complex_fft_generic.c

	Lowest cycle count	Fastest computation at 60 MHz	Lowest Error		Lowest Algorithm's size in memory
			Amplitude average	Max. amplitude	
64-points	13,206	225.2us	6.0e-10	5.7e-9	2.0 Kbytes
256-points	74,297	1.27us	3.0e-10	4.8e-9	2.4 Kbytes
1024-points	383,212	6.53ms	3.0e-10	6.1e-9	3.9 Kbytes

More details on Table 1.2.1 in annexes

3. Convolution

3.1 Description

This function performs a linear convolution between two discrete sequences.

3.1.1 Function prototype

```
void dspXX_vect_conv(
    dspXX_t *vect1,
    dspXX_t *vect2,
    int vect2_size,
    dspXX_t *vect3,
    int vect3_size);
```

where XX corresponds to the number of bits of a basic data element (i.e. 16 or 32).

3.1.2 Arguments

This function takes five parameters: the output buffer, the two discrete sequences and their respective sizes.

- The output buffer (vect1) is a pointer on a real vector of **(vect2_size + vect3_size - 1)** elements.
- The first input buffer (vect2) is a pointer on a real vector of **vect2_size** elements.
- The first size argument (vect2_size) is the length of the first input buffer ($\text{vect2_size} \in [8, 9, 10, \dots]$).
- The second input buffer (vect3) is a pointer on a real vector of **vect3_size** elements.
- The second size argument (vect3_size) is the length of the second input buffer (vect3_size fits in $[8, 9, 10, \dots]$).

3.1.3 Requirements

This function requires 3 modules:

Module name	Function name	Concerned file path
Zero Padding	dspXX_vect_zeropad	/BASIC/VECTORS/zero_padding.c
Copy	dspXX_vect_copy	/BASIC/VECTORS/copy.c
Partial Convolution	dspXX_vect_convpart	/BASIC/VECTORS/convolution_partial.c

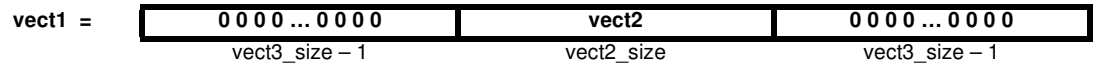


The output buffer of the function has to have at least a length of $\mathbf{N + 2 * M - 2}$ elements because of intern computations, where N is the length of the largest input buffer and M, the length of the smallest input buffer.

3.1.4 Algorithm

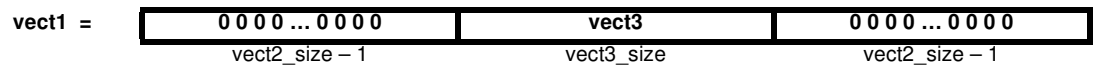
Following is the algorithm used to implement the convolution product. The optimized version is based on this algorithm but can differ in certain points due to the instruction set of the target:

IF vect2_size >= vect3_size **THEN**



Partial_convolution(vect1, vect1, vect2_size + 2*(vect3_size - 1), vect3, vect3_size)

ELSE



Partial_convolution(vect1, vect1, vect3_size + 2*(vect2_size - 1), vect2, vect2_size)

END

3.1.5 Notes

- **Interruptibility:** the code is interruptible.
- Due to its implementation, the dsp16-avr32-uc3 optimized version of the FIR requires a length of 4*m elements for the largest input discrete sequence and the output buffer (vect1) has to have a length of 4*n elements to avoid overflows.
- The input discrete sequences have to be scaled to avoid overflowing values.
- All the vectors have to be 32-bit aligned.

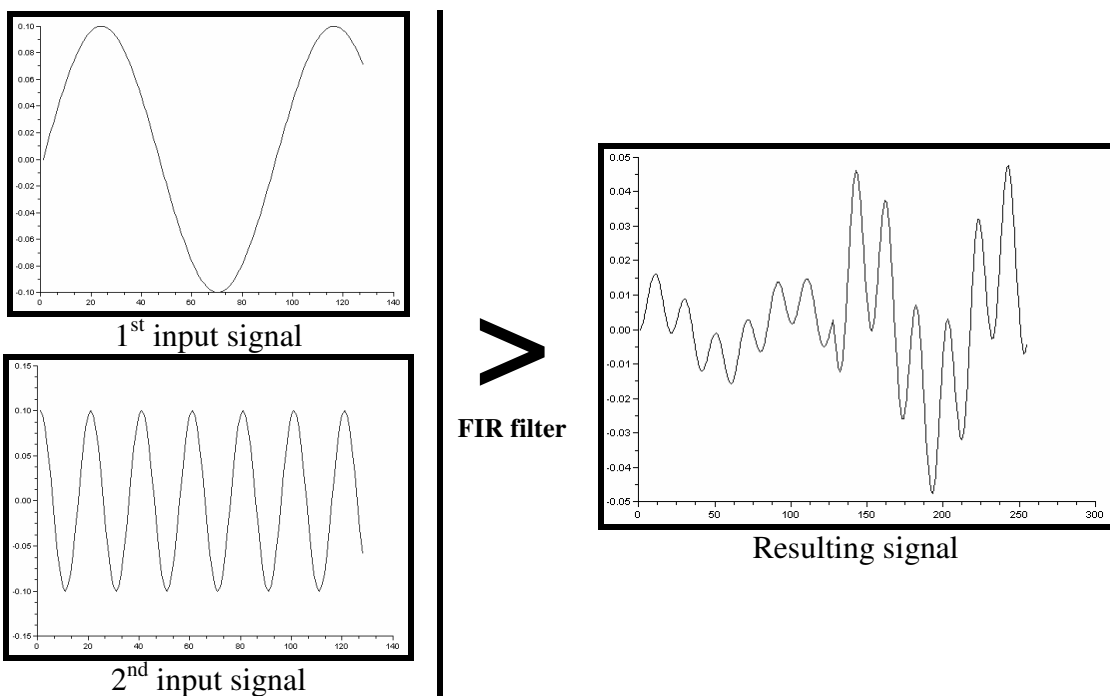
3.2 Benchmark

3.2.1 Benchmark routine

All these functions have been benchmarked on an avr32-uc3a0512 target. The programs have been compiled with avr32-gcc (4.1.2-atmel.1.0.0) with the -O3 optimization option and have been stored in FLASH memory. The fixed-point format used is the Q1.15 format for the 16-bit data and the Q1.31 format for the 32-bit data.

The benchmark process has been performed with the same input signal and impulse response for all those functions and compared with a reference's signal computed with a mathematic tool using floating point.

The first input signal is a sine oscillating at 433Hz and the second input signal is a cosine oscillating at 2KHz. Those signals are sampled at 40KHz.



3.2.2 Result

Here are tables of the main values of the benchmark results. All those values correspond to the best performances of the functions and are obtained with different compilation options. For more information, please refer to the complete benchmark result table in annexes.

Concerned file path: */BASIC/VECTORS/convolution.c*

3.2.2.1 16-bit Convolution: generic

Algorithm's size in memory: 2.2 Kbytes.

Length of the first input signal: 64 elements.

	Lowest cycle count	Fastest computation at 60 MHz	Lowest Error	
			Amplitude average	Max. amplitude
32-points	23,524	408.5us	2.0e-5	4.5e-5
64-points	57,757	1.00ms	1.8e-5	4.7e-5
128-points	86,752	1.50ms	1.8e-5	4.4e-5
256-points	144,736	2.51ms	1.5e-5	4.8e-5

More details on Table 2.1.1 in annexes

3.2.2.2 16-bit Convolution: avr32-uc3 optimized

Algorithm's size in memory: 950 bytes.

Length of the first input signal: 64 elements.

	Lowest cycle count	Fastest computation at 60 MHz	Lowest Error	
			Amplitude average	Max. amplitude
32-points	8,248	151.2us	2.0e-5	4.5e-5
64-points	19,087	349.1us	1.8e-5	4.7e-5
128-points	28,532	521.8us	1.8e-5	4.4e-5
256-points	47,412	866.9us	1.5e-5	4.8e-5

More details on Table 2.1.2 in annexes

3.2.2.3 32-bit Convolution: generic

Algorithm's size in memory: 3.3 Kbytes.

Length of the first input signal: 64 elements.

	Lowest cycle count	Fastest computation at 60 MHz	Lowest Error	
			Amplitude average	Max. amplitude
32-points	42,572	729.8us	0.4e-9	2.1e-9
64-points	109,179	1.87ms	0.4e-9	1.7e-9
128-points	163,968	2.81ms	0.5e-9	1.6e-9
256-points	273,536	4.69ms	0.6e-9	2.7e-9

More details on Table 2.2.1 in annexes

3.2.2.4 32-bit Convolution: avr32-uc3 optimized

Algorithm's size in memory: 1.5 Kbytes.

Length of the first input signal: 64 elements.

	Lowest cycle count	Fastest computation at 60 MHz	Lowest Error	
			Amplitude average	Max. amplitude
32-points	19,958	340.2us	0.5e-9	2.1e-9
64-points	50,501	860.4us	0.5e-9	1.7e-9
128-points	75,722	1.29ms	0.6e-9	2.4e-9
256-points	126,154	2.15ms	0.7e-9	2.7e-9

More details on Table 2.2.2 in annexes

4. FIR Filter (*alias Partial Convolution*)

4.1 Description

This function computes a real FIR filter using the impulse response of the desire filter onto a fixed-length signal.

4.1.1 Function prototype

```
void dspXX_filt_fir(
    dspXX_t *vect1,
    dspXX_t *vect2,
    int size,
    dspXX_t *h,
    int h_size);

void dspXX_vect_convpart(
    dspXX_t *vect1,
    dspXX_t *vect2,
    int vect2_size,
    dspXX_t *vect3,
    int vect3_size);
```

where XX corresponds to the number of bits of a basic data element (i.e. 16 or 32).

4.1.2 Arguments

This function takes five parameters: the output buffer, the input buffer, its size, the impulse response of the filter and its size.

The output buffer (vect1) is a pointer on a real vector of **(size - h_size + 1)** elements.

The input buffer (vect2) is a pointer on a real vector of **size** elements.

The size argument (size) is the length of the input buffer (size fits in [4, 8, 12, ...]).

The impulse response of the filter (h) is a pointer on a real vector of **h_size** elements.

The size argument (h_size) is the length of the impulse response of the filter (h_size fits in [8, 9, 10, ...]).

4.1.3 Requirements

This function requires one module:

Module name	Function name	Concerned file path
Partial Convolution	dspXX_vect_convpart	/BASIC/VECTORS/convolution_partial.c

4.1.4 Algorithm

Following is the algorithm used to implement the FIR filter. The optimized version is based on this algorithm but can differ in certain points due to the instruction set of the target:

```

FOR j FROM 0 TO size - h_size + 1 DO
  sum = 0
  FOR i FROM 0 TO h_size DO
    sum += vect2[i] * h[h_size - i - 1]
  END
  vect1[j] = sum >> DSPXX_QB
END

```

4.1.5 Notes

- **Interruptibility:** the code is interruptible.
- Due to its implementation, for the dsp16-avr32-uc3 optimized version of the FIR, the output buffer (vect1) has to have a length of 4*n elements to avoid overflows.
- The impulse response of the filter has to be scaled to avoid overflowing values.
- All the vectors have to be 32-bit aligned.

4.2 Benchmark

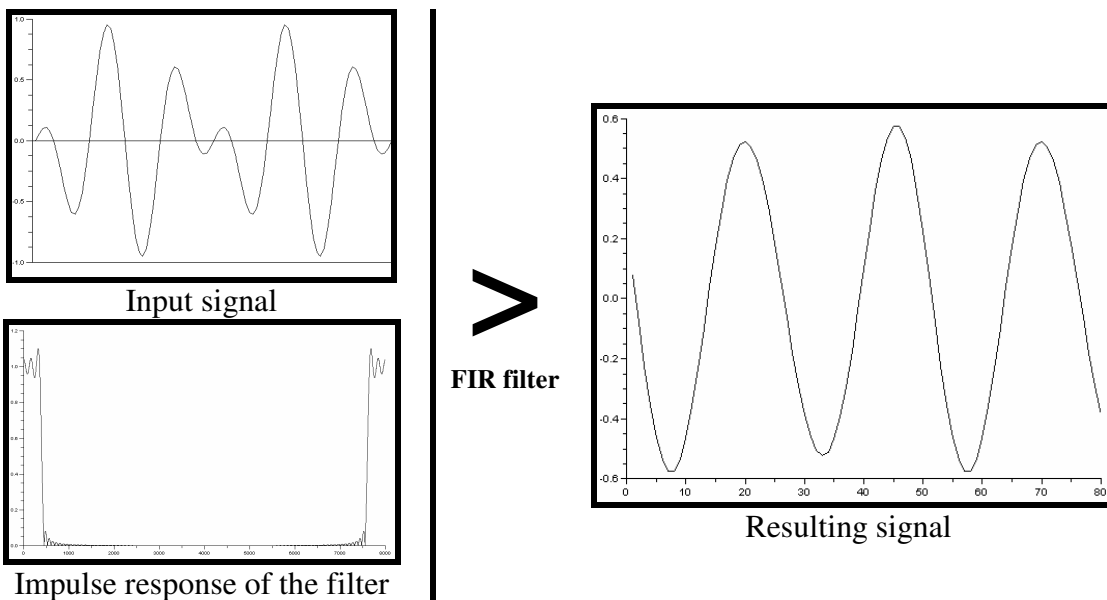
4.2.1 Benchmark routine

All these functions have been benchmarked on an avr32-uc3a0512 target. The programs have been compiled with avr32-gcc (4.1.2-atmel.1.0.0) with the `-O3` optimization option and have been stored in FLASH memory. The fixed-point format used is the Q1.15 format for the 16-bit data and the Q1.31 format for the 32-bit data.

The benchmark process has been performed with the same input signal and impulse response for all those functions and compared with a reference's signal computed with a mathematic tool using floating point.

The input signal is a combination of one sine and one cosine. The sine oscillating at 400Hz and the cosine at 2KHz. Those signals have been multiplied and sampled at 40KHz.

The impulse response describes a low-pass filter with a cutoff frequency equal to 400Hz.



4.2.2 Result

Here are tables of the main values of the benchmark results. All those values correspond to the best performances of the functions and are obtained with different compilation options. For more information, please refer to the complete benchmark result table in annexes.

4.2.2.1 16-bit FIR filter: generic

Concerned file path: `/BASIC/VECTORS/dsp16_convpart_generic.c`

Algorithm's size in memory: 2.0 Kbytes.

Number of Taps: 24.

	Lowest cycle count	Fastest computation at 60 MHz	Lowest Error	
			Amplitude average	Max. amplitude
64-points	7,424	128.0us	2.27e-5	9.46e-5
256-points	41,793	720.0us	2.22e-5	9.46e-5
512-points	87,617	1.51ms	2.23e-5	9.46e-5
1024-points	179,265	3.09ms	2.21e-5	9.46e-5

More details on Table 3.1.1 in annexes

4.2.2.2 16-bit FIR filter: avr32-uc3 optimized

Concerned file path: `/BASIC/VECTORS/dsp16_convpart_avr32uc3.c`

Algorithm's size in memory: 770 bytes.

Number of Taps: 24.

	Lowest cycle count	Fastest computation at 60 MHz	Lowest Error	
			Amplitude average	Max. amplitude
64-points	2,439	44.3us	2.27e-5	9.46e-5
256-points	12,712	230.7us	2.22e-5	9.46e-5
512-points	26,408	479.2us	2.23e-5	9.46e-5
1024-points	53,800	976.3us	2.21e-5	9.46e-5

More details on Table 3.1.2 in annexes

4.2.2.3 32-bit FIR filter: generic

Concerned file path: `/BASIC/VECTORS/dsp32_convpart_generic.c`

Algorithm's size in memory: 3.1 Kbytes.

Number of Taps: 24.

	Lowest cycle count	Fastest computation at 60 MHz	Lowest Error	
			Amplitude average	Max. amplitude
64-points	13,984	239.4us	2.1e-9	1.24e-8
256-points	79,073	1.35ms	2.3e-9	1.74e-8
512-points	165,857	2.84ms	2.6e-9	2.31e-8
1024-points	339,425	5.81ms	3.7e-9	2.84e-8

More details on Table 3.2.1 in annexes

4.2.2.4 32-bit FIR filter: avr32-uc3 optimized

Concerned file path: `/BASIC/VECTORS/dsp32_convpart_avr32uc3.c`

Algorithm's size in memory: 1.3 Kbytes.

Number of Taps: 24.

	Lowest cycle count	Fastest computation at 60 MHz	Lowest Error	
			Amplitude average	Max. amplitude
64-points	6,479	110.2us	2.1e-9	1.24e-8
256-points	36,432	619.0us	2.3e-9	1.24e-8
512-points	76,368	1.30ms	2.6e-9	2.31e-8
1024-points	156,240	2.65ms	3.7e-9	2.84e-8

More details on Table 3.2.2 in annexes

5. Partial IIR Filter

5.1 Description

This function computes a real IIR filter using the impulse response of the desire filter onto a fixed-length signal.

5.1.1 Function prototype

```
void dspXX_filt_iir(
    dspXX_t *vect1,
    dspXX_t *vect2,
    int size,
    dspXX_t *num,
    int num_size,
    dspXX_t *den,
    int den_size,
    int num_prediv,
    int den_prediv);
```

where XX corresponds to the number of bits of a basic data element (i.e. 16 or 32).

5.1.2 Arguments

This function takes five parameters: the output buffer, the input buffer, its size, the coefficients of the filter, theirs sizes and a coefficient's predivisor.

The output buffer (vect1) is a pointer on a real vector of **(size - num_size + 1)** elements.

The input buffer (vect2) is a pointer on a real vector of **size** elements.

The size argument (size) is the length of the input buffer (size fits in [4, 5, 6, 7, ...]).

The numerator's coefficients argument of the filter (num) is a pointer on a real vector of **num_size** elements.

The size argument (num_size) is the length of the numerator's coefficients of the filter (num_size fits in [1, 2, 3, ...]).

The denominator's coefficients argument of the filter (den) is a pointer on a real vector of **den_size** elements.

The size argument (den_size) is the length of the denominator's coefficients of the filter (den_size fits in [1, 2, 3, ...]).

The predivisors (num_prediv and den_prediv) are used to scale down the denominator/numerator's coefficients of the filter in order to avoid overflow values. So when you use this feature, you have to prescale manually the denominator/numerator's coefficients by 2^{prediv} else leave this field to 0.

5.1.3 Algorithm

Following is the algorithm used to implement the IIR filter. The optimized version is based on this algorithm but can differ in certain points due to the instruction set of the target:

```
// Initialization of the vect1 coefficients
FOR n FROM 0 TO den_size - 1 DO
sum1 = 0
FOR m FROM 0 TO num_size - 1 DO
sum1 += num[m] * vect2[n + num_size - m - 1]
END
sum2 = 0
FOR m FROM 1 TO n DO
sum2 += den[m] * vect1[n - m]
END
vect1[n] = (sum1 - (sum2 << prediv)) >> DSPXX_QB
END

FOR n FROM n TO size - num_size DO
sum1 = 0
FOR m FROM 0 TO num_size - 1 DO
sum1 += num[m] * vect2[n + num_size - m - 1]
END
sum2 = 0
FOR m FROM 1 TO den_size - 1 DO
sum2 += den[m] * vect1[n - m]
END
vect1[n] = (sum1 - (sum2 << prediv)) >> DSPXX_QB
END
```

5.1.4 Notes

- **Interruptibility:** the code is interruptible.
- Due to its implementation, for the dsp16-avr32-uc3 optimized version of the FIR, the output buffer (vect1) has to have a length of 4*n elements to avoid overflows.
- The impulse response of the filter has to be scaled to avoid overflowing values.
- All the vectors have to be 32-bit aligned.
- The first denominator's coefficient have to be equal to 1 / (2^prediv).
- The predivisor (prediv) must be lower or equals to the constant DSPXX_QB.

5.2 Benchmark

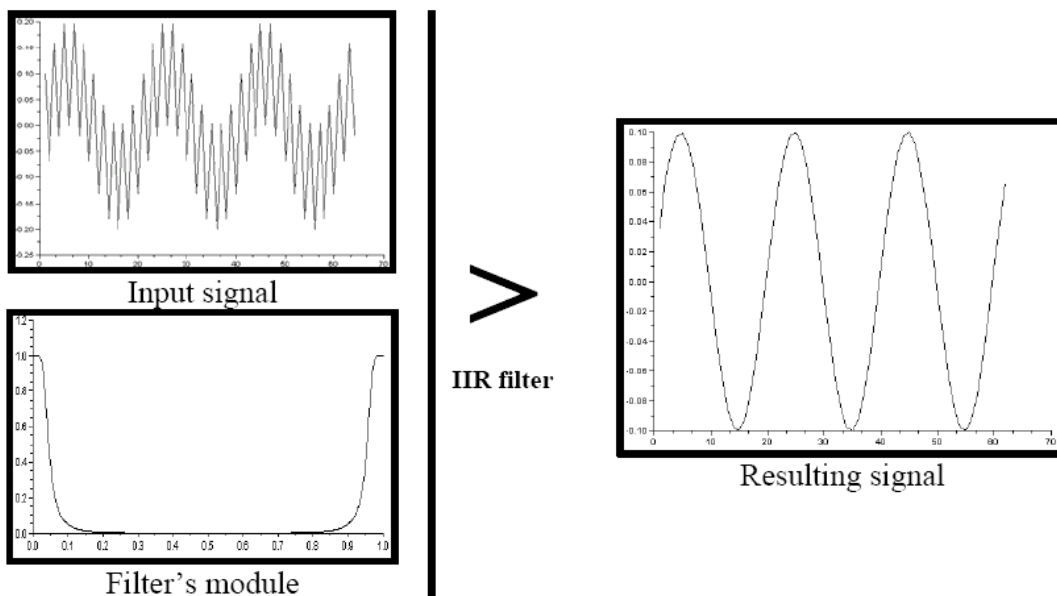
5.2.1 Benchmark routine

All these functions have been benchmarked on an avr32-uc3a0512 target. The programs have been compiled with avr32-gcc (4.1.2-atmel.1.0.0) with the `-O3` optimization option and have been stored in FLASH memory. The fixed-point format used is the Q1.15 format for the 16-bit data and the Q1.31 format for the 32-bit data.

The benchmark process has been performed with the same input signal and impulse response for all those functions and compared with a reference's signal computed with a mathematic tool using floating point.

The input signal is a combination of one sine and one cosine. The sine oscillating at 400Hz and the cosine at 4KHz. Those signals have been added together and sampled at 8KHz.

The filter used is a low-pass Butterworth filter with a cutoff frequency equal to 2KHz.



5.2.2 Result

Here are tables of the main values of the benchmark results. All those values correspond to the best performances of the functions and are obtained with different compilation options. For more information, please refer to the complete benchmark result table in annexes.

5.2.2.1 16-bit IIR filter: generic

Concerned file path: `/BASIC/FILTERING/dsp16_iir_generic.c`

Algorithm's size in memory: 266 bytes.

Order of the filter: 7

	Lowest cycle count	Fastest computation at 60 MHz	Lowest Error	
			Amplitude average	Max. amplitude
72-points	11,469	213.4us	1.40e-5	4.30e-5
256-points	44,591	829.8us	1.40e-5	4.30e-5
512-points	90,671	1.69ms	1.40e-5	4.30e-5
1024-points	182,831	3.40ms	1.40e-5	4.30e-5

More details on Table 4.1.1 in annexes

5.2.2.2 16-bit IIR filter: avr32-uc3 optimized

Concerned file path: `/BASIC/FILTERING/dsp16_iir_avr32uc3.c`

Algorithm's size in memory: 1.0 Kbytes (size optimization), 3.1 Kbytes (speed optimization).

Order of the filter: 7

	Lowest cycle count	Fastest computation at 60 MHz	Lowest Error	
			Amplitude average	Max. amplitude
72-points	4,332	78.0us	1.90e-5	6.90e-5
256-points	15,006	270.5us	1.70e-5	6.90e-5
512-points	29,854	538.2us	1.70e-5	6.90e-5
1024-points	59,550	1.07ms	1.70e-5	6.90e-5

More details on Table 4.1.2 in annexes

5.2.2.3 32-bit IIR filter: generic

Concerned file path: `/BASIC/FILTERING/dsp32_iir_generic.c`

Algorithm's size in memory: 400 bytes.

Order of the filter: 7

	Lowest cycle count	Fastest computation at 60 MHz	Lowest Error	
			Amplitude average	Max. amplitude
72-points	14,517	265.4us	1.50e-9	7.00e-9
256-points	56,471	1.03ms	1.50e-9	7.00e-9
512-points	114,839	2.10ms	1.50e-9	7.00e-9
1024-points	231,575	4.23ms	1.50e-9	7.00e-9

More details on Table 4.2.1 in annexes

5.2.2.4 32-bit IIR filter: avr32-uc3 optimized

Concerned file path: `/BASIC/FILTERING/dsp32_iir_avr32uc3.c`

Algorithm's size in memory: 3.0 Kbytes.

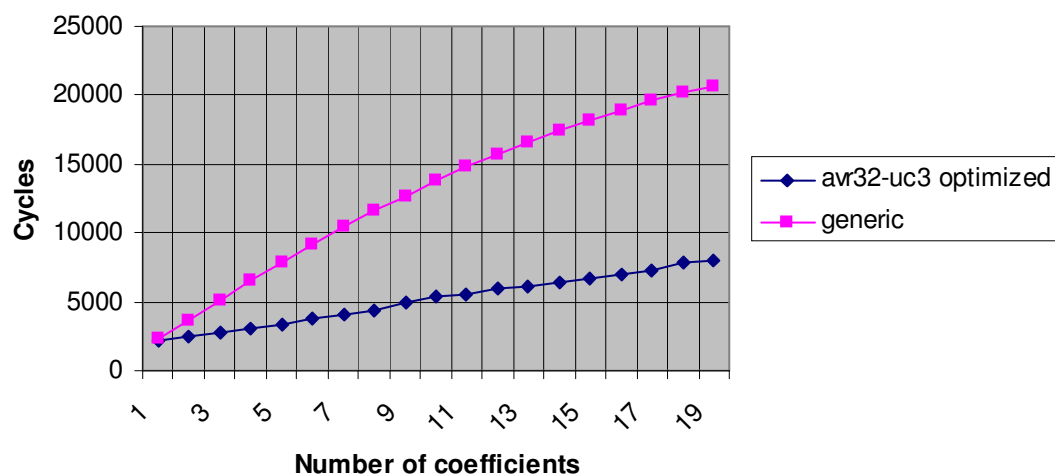
Order of the filter: 7

	Lowest cycle count	Fastest computation at 60 MHz	Lowest Error	
			Amplitude average	Max. amplitude
72-points	8,859	153.9us	1.50e-9	7.00e-9
256-points	33,333	577.1us	1.50e-9	7.00e-9
512-points	67,381	1.17ms	1.60e-9	7.00e-9
1024-points	135,477	2.34ms	1.60e-9	7.00e-9

More details on Table 4.2.2 in annexes

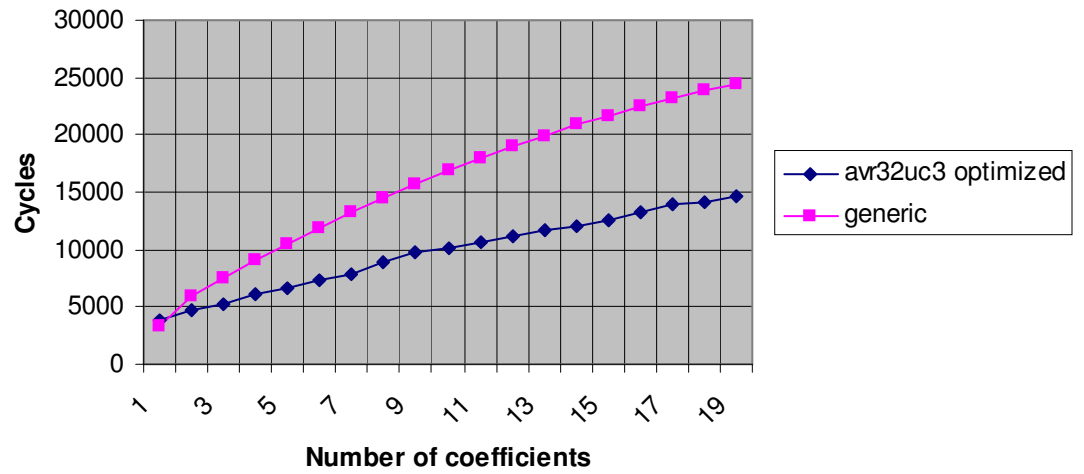
Benchmark results for the 16-bit version (with speed optimization)

The benchmark has been performed on a 72-element input signal.



Benchmark results for the 32-bit version (with speed optimization)

The benchmark has been performed on a 72-element input signal.



Remark: the number of coefficients corresponds to a filter which order is equal to “Number of coefficients” – 1.

6. Annexes

Table 1.1.1 - Benchmark of 16-bit Radix-4 D.I.T. complex FFT: generic

	Optimization	T.F.T.* in SRAM (cycles)		T.F.T.* in FLASH (cycles)		Error		Algorithm's size in memory (bytes) (code size + T.F.T.* size)
		wait-state		wait-state		Amplitude average (x10 ⁻⁵)	Max. amplitude (x10 ⁻⁵)	
		0 at 30MHz	1 at 60MHz	0 at 30MHz	1 at 60MHz			
64-point	(0)	6,296 (209.9us)	6,489 (108.2us)	6,640 (221.3us)	7,012 (116.9us)	2.98	15.63	1K (840 + 194)
	(1)	6,343 (211.4us)	6,538 (109.0us)	6,777 (225.9us)	7,167 (119.5us)	1.58	6.53	1K (844 + 194)
	(2)	6,666 (222.2us)	6,959 (116.0us)	7,092 (236.4us)	7,546 (125.8us)	2.98	15.63	1.1K (1032 + 66)
	(3)	6,747 (224.9us)	6,996 (116.6us)	7,101 (236.7us)	7,536 (125.6us)	1.58	6.53	1K (984 + 66)
256-point	(0)	33,723 (1124.1us)	34,682 (578.0us)	35,265 (1175.5us)	37,033 (617.2us)	3.02	21.92	1.6K (840 + 770)
	(1)	34,041 (1134.7us)	35,003 (583.4us)	35,988 (1199.6us)	37,836 (630.6us)	1.69	8.80	1.6K (844 + 770)
	(2)	35,304 (1176.8us)	36,675 (611.3us)	37,194 (1239.8us)	39,294 (654.9us)	3.02	21.92	1.3K (1032 + 258)
	(3)	35,826 (1194.2us)	37,032 (617.2us)	37,419 (1247.3us)	39,453 (657.6us)	1.69	8.80	1.2K (984 + 258)
1024-point	(0)	169,006 (5.63ms)	173,611 (2.90ms)	175,394 (5.85ms)	183,358 (3.06ms)	3.04	25.01	3.8K (840 + 3K)
	(1)	170,795 (5.69ms)	175,404 (2.92ms)	178,863 (5.96ms)	187,161 (3.12ms)	1.67	12.31	3.8K (844 + 3K)
	(2)	175,446 (5.85ms)	181,703 (3.03ms)	183,248 (6.11ms)	192,530 (3.21ms)	3.04	25.01	2K (1032 + 1K)
	(3)	178,153 (5.94ms)	183,772 (3.06ms)	184,761 (6.16ms)	193,802 (3.23ms)	1.67	12.31	2K (984 + 1K)
4096-point	(0)	812,321 (27.08ms)	833,820 (13.90ms)	838,147 (27.94ms)	873,235 (14.55ms)	3.09	32.90	12.8K (840 + 12K)
	(1)	821,533 (27.38ms)	843,037 (14.05ms)	854,154 (28.47ms)	890,598 (14.84ms)	1.52	14.60	12.8K (844 + 12K)
	(2)	838,212 (27.94ms)	866,315 (14.44ms)	869,718 (28.99ms)	910,054 (15.17ms)	3.09	32.90	5K (1032 + 4K)
	(3)	851,232 (28.37ms)	876,816 (14.61ms)	877,959 (29.27ms)	917,367 (15.29ms)	1.52	14.60	5K (984 + 4K)

*: Twiddle Factors Table.

(0): Algorithmic optimized for **speed**.

(1): Algorithmic optimized for **accuracy**.

(2): Algorithmic optimized for **size**.

(3): Algorithmic optimized for **size** and **accuracy**.

Table 1.1.2 - Benchmark of 16-bit Radix-4 D.I.T. complex FFT: avr32-uc3 optimized

	Optimization	T.F.T.* in SRAM (cycles)		T.F.T.* in FLASH (cycles)		Error		Algorithm's size in memory (bytes) (code size + T.F.T.* size)
		wait-state		wait-state		Amplitude average (x10 ⁻⁵)	Max. amplitude (x10 ⁻⁵)	
		0 at 30MHz	1 at 60MHz	0 at 30MHz	1 at 60MHz			
64-point	(0)	2,611 (87.0us)	2,661 (44.4us)	2,753 (91.8us)	2,877 (48.0us)	3.00	10.04	894 (700 + 194)
	(1)	2,951 (98.4us)	2,999 (50.0us)	3,097 (103.2us)	3,265 (54.4us)	1.63	6.53	774 (580 + 194)
	(2)	2,833 (94.4us)	2,912 (48.5us)	3,027 (100.9us)	3,221 (53.7us)	2.93	10.04	710 (644 + 66)
	(3)	3,206 (106.9us)	3,311 (55.2us)	3,458 (115.3us)	3,683 (61.4us)	1.63	6.53	766 (700 + 66)
256-point	(0)	13,661 (455.4us)	13,932 (232.2us)	14,306 (476.9us)	14,904 (248.4us)	2.78	13.86	1.4K (700 + 770)
	(1)	15,777 (525.9us)	16,033 (267.2us)	16,428 (547.6us)	17,242 (287.4us)	1.68	7.46	1.3K (580 + 770)
	(2)	14,651 (488.4us)	15,056 (250.9us)	15,527 (517.6us)	16,442 (274.0us)	2.87	15.82	902 (644 + 258)
	(3)	16,916 (563.9us)	17,469 (291.2us)	18,041 (601.4us)	19,152 (319.2us)	1.68	7.46	958 (700 + 258)
1024-point	(0)	67,671 (2.26ms)	69,027 (1.15ms)	70,355 (2.35ms)	73,059 (1.22ms)	2.84	19.33	3.7K (700 + 3K)
	(1)	79,195 (2.64ms)	80,475 (1.34ms)	81,887 (2.73ms)	85,507 (1.43ms)	1.69	10.23	3.6K (580 + 3K)
	(2)	71,765 (2.39ms)	73,680 (1.23ms)	75,403 (2.51ms)	79,423 (1.32ms)	2.86	19.33	1.6K (644 + 1K)
	(3)	83,906 (2.80ms)	86,635 (1.44ms)	88,560 (2.95ms)	93,629 (1.56ms)	1.69	10.23	1.7K (700 + 1K)
4096-point	(0)	322,897 (10.76ms)	329,370 (5.49ms)	333,764 (11.13ms)	345,678 (5.76ms)	2.80	23.69	12.7K (700 + 12K)
	(1)	381,269 (12.71ms)	387,413 (6.46ms)	392,146 (13.07ms)	407,788 (6.80ms)	1.58	11.84	12.6K (580 + 12K)
	(2)	339,439 (11.31ms)	348,176 (5.80ms)	354,159 (11.81ms)	371,396 (6.19ms)	2.81	23.69	4.6K (644 + 4K)
	(3)	400,304 (13.34ms)	413,273 (6.89ms)	419,111 (13.97ms)	441,578 (7.36ms)	1.58	11.84	4.7K (700 + 4K)

*: Twiddle Factors Table.

(0): Algorithmic optimized for **speed**.

(1): Algorithmic optimized for **accuracy**.

(2): Algorithmic optimized for **size**.

(3): Algorithmic optimized for **size and accuracy**.

Table 1.2.1 - Benchmark of 32-bit Radix-4 D.I.T. complex FFT: generic

	Optimization	T.F.T.* in SRAM (cycles)		T.F.T.* in FLASH (cycles)		Error		Algorithm's size in memory (bytes) (code size + T.F.T.* size)
		wait-state		wait-state		Amplitude average (x10 ⁻⁹)	Max. amplitude (x10 ⁻⁹)	
		0 at 30MHz	1 at 60MHz	0 at 30MHz	1 at 60MHz			
64-point	(0)	13,206 (440.2us)	13,509 (225.2us)	13,580 (452.7us)	14,063 (234.4us)	0.70	5.70	2.2K (1816 + 392)
	(1)	15,323 (510.8us)	15,659 (261.0us)	15,627 (520.9us)	16,113 (268.6us)	0.60	5.70	2.4K (2112 + 392)
	(2)	13,622 (454.1us)	14,011 (233.5us)	13,938 (464.6us)	14,497 (241.6us)	0.70	5.70	2.0K (1952 + 136)
	(3)	15,714 (523.8us)	16,245 (270.8us)	16,058 (535.3us)	16,805 (280.1us)	0.60	5.70	2.3K (2248 + 136)
256-point	(0)	74,297 (2.48ms)	75,940 (1.27ms)	75,992 (2.53ms)	78,445 (1.31ms)	0.50	4.80	3.3K (1816 + 1.5K)
	(1)	87,791 (2.93ms)	89,605 (1.49ms)	89,165 (2.97ms)	91,636 (1.53ms)	0.30	4.80	3.6K (2112 + 1.5K)
	(2)	76,136 (2.54ms)	78,160 (1.30ms)	77,537 (2.58ms)	80,329 (1.34ms)	0.50	4.80	2.4K (1952 + 520)
	(3)	89,543 (2.98ms)	92,446 (1.54ms)	91,058 (3.04ms)	94,978 (1.59ms)	0.30	4.80	2.7K (2248 + 520)
1024-point	(0)	383,212 (12.77ms)	391,715 (6.53ms)	390,260 (13.01ms)	402,123 (6.70ms)	0.50	8.80	7.8K (1816 + 6K)
	(1)	457,571 (15.25ms)	466,815 (7.78ms)	463,279 (15.44ms)	475,223 (7.92ms)	0.30	6.10	8.1K (2112 + 6K)
	(2)	390,794 (13.03ms)	400,869 (6.68ms)	396,576 (13.22ms)	409,841 (6.83ms)	0.50	8.80	3.9K (1952 + 2K)
	(3)	464,988 (15.50ms)	479,847 (8.00ms)	471,226 (15.71ms)	490,367 (8.17ms)	0.30	6.10	4.2K (2248 + 2K)

*: Twiddle Factors Table.

(0): Algorithmic optimized for **speed**.

(1): Algorithmic optimized for **accuracy**.

(2): Algorithmic optimized for **size**.

(3): Algorithmic optimized for **size and accuracy**.

Table 2.1.1 - Benchmark of 16-bit Convolution: generic

1 st input signal	2 nd input signal	Execution time (cycles)		Error	
		wait-state		Amplitude average (x10 ⁻⁵)	Max. amplitude (x10 ⁻⁵)
		0 at 30MHz	1 at 60MHz		
32-point	32-points	15,681 (522.7us)	16,344 (272.4us)	1.60	3.90
	64-points	23,524 (784.1us)	24,508 (408.5us)	2.00	4.50
	128-points	39,204 (1.31ms)	40,830 (680.5us)	1.90	4.50
	256-points	70,564 (2.35ms)	73,470 (1.22ms)	1.70	4.10
64-point	64-points	57,757 (1.93ms)	60,084 (1.00ms)	1.80	4.70
	128-points	86,752 (2.89ms)	90,234 (1.50ms)	1.80	4.40
	256-points	144,736 (4.82ms)	150,522 (2.51ms)	1.50	4.80
128-point	128-points	221,780 (7.39ms)	230,512 (3.84ms)	1.80	5.30
	256-points	333,018 (11.10ms)	346,097 (5.77ms)	1.70	5.00
256-point	256-points	869,316 (28.98ms)	903,136 (15.05ms)	1.70	5.40

Table 2.1.2 - Benchmark of 16-bit Convolution: avr32-uc3 optimized

1 st input signal	2 nd input signal	Execution time (cycles)		Error	
		wait-state		Amplitude average (x10 ⁻⁵)	Max. amplitude (x10 ⁻⁵)
		0 at 30MHz	1 at 60MHz		
32-point	32-points	5,571 (185.7us)	6,127 (102.1us)	1.60	3.90
	64-points	8,248 (274.9us)	9,070 (151.2us)	2.00	4.50
	128-points	13,592 (453.1us)	14,944 (249.1us)	1.90	4.50
	256-points	24,280 (809.3us)	26,688 (444.8us)	1.70	4.10
64-point	64-points	19,087 (636.2us)	20,947 (349.1us)	1.80	4.70
	128-points	28,532 (951.1us)	31,308 (521.8us)	1.80	4.40
	256-points	47,412 (1.58ms)	52,012 (866.9us)	1.50	4.80
128-point	128-points	70,694 (2.36ms)	77,471 (1.29ms)	1.80	5.30
	256-points	105,966 (3.53ms)	116,099 (1.93ms)	1.70	5.00
256-point	256-points	272,214 (9.07ms)	298,031 (4.97ms)	1.70	5.40

Table 2.2.1 - Benchmark of 32-bit Convolution: generic

1 st input signal	2 nd input signal	Execution time (cycles)		Error	
		wait-state		Amplitude average (x10 ⁻⁹)	Max. amplitude (x10 ⁻⁹)
		0 at 30MHz	1 at 60MHz		
32-point	32-points	28,359 (945.3us)	29,182 (486.4us)	0.40	1.60
	64-points	42,572 (1.42ms)	43,788 (729.8us)	0.40	2.10
	128-points	70,988 (2.37ms)	72,990 (1.22ms)	0.30	1.90
	256-points	127,820 (4.26ms)	131,390 (2.19ms)	0.40	1.70
64-point	64-points	109,179 (3.64ms)	112,334 (1.87ms)	0.40	1.70
	128-points	163,968 (5.47ms)	168,678 (2.81ms)	0.50	1.60
	256-points	273,536 (9.12ms)	281,350 (4.69ms)	0.60	2.70
128-point	128-points	429,026 (14.30ms)	441,458 (7.36ms)	0.40	2.30
	256-points	644,074 (21.47ms)	662,677 (11.04ms)	0.40	2.00
256-point	256-points	1,701,554 (56.72ms)	1,750,962 (29.18ms)	0.50	3.10

Table 2.2.2 - Benchmark of 32-bit Convolution: avr32-uc3 optimized

1 st input signal	2 nd input signal	Execution time (cycles)		Error	
		wait-state		Amplitude average (x10 ⁻⁹)	Max. amplitude (x10 ⁻⁹)
		0 at 30MHz	1 at 60MHz		
32-point	32-points	13,361 (445.4us)	13,680 (228.0us)	0.60	2.30
	64-points	19,958 (665.3us)	20,414 (340.2us)	0.50	2.10
	128-points	33,142 (1.10ms)	33,872 (564.5us)	0.50	1.90
	256-points	59,510 (1.98ms)	60,784 (1.01ms)	0.50	2.10
64-point	64-points	50,501 (1.68ms)	51,624 (860.4us)	0.50	1.70
	128-points	75,722 (2.52ms)	77,376 (1.29ms)	0.60	2.40
	256-points	126,154 (4.21ms)	128,864 (2.15ms)	0.70	2.70
128-point	128-points	196,972 (6.57ms)	201,244 (3.35ms)	0.50	2.30
	256-points	295,540 (9.85ms)	301,887 (5.03ms)	0.60	2.30
256-point	256-points	778,684 (25.96ms)	795,388 (13.26ms)	0.60	3.10

Table 3.1.1 - Benchmark of 16-bit FIR Filter: generic

	Number of Taps	I.R.* in SRAM (cycles)		I.R.* in FLASH (cycles)		Error	
		wait-state		wait-state		Amplitude average (x10 ⁻⁵)	Max. amplitude (x10 ⁻⁵)
		0 at 30MHz	1 at 60MHz	0 at 30MHz	1 at 60MHz		
64-point	24	7,424 (247.5us)	7,682 (128.0us)	10,704 (356.8us)	12,352 (205.9us)	2.27	9.46
	48	5,780 (192.7us)	5,996 (99.9us)	8,517 (283.9us)	9,868 (164.5us)	3.09	15.57
256-point	24	41,793 (1.39ms)	43,202 (720.0us)	60,433 (2.01ms)	69,760 (1.16ms)	2.22	9.46
	48	70,101 (2.34ms)	72,620 (1.21ms)	103,750 (3.46ms)	120,268 (2.00ms)	5.66	27.20
	72	90,921 (3.03ms)	94,262 (1.57ms)	135,691 (4.52ms)	157,528 (2.63ms)	11.25	52.65
	100	105,127 (3.50ms)	108,903 (1.82ms)	156,466 (5.22ms)	185,989 (3.10ms)	14.19	63.71
512-point	24	87,617 (2.92ms)	90,562 (1.51ms)	126,737 (4.22ms)	146,304 (2.44ms)	2.23	9.46
	48	155,861 (5.20ms)	161,452 (2.69ms)	230,726 (7.69ms)	267,468 (4.46ms)	5.78	27.20
	72	216,617 (7.22ms)	224,566 (3.74ms)	323,339 (10.78ms)	375,384 (6.26ms)	10.79	52.65
	100	276,391 (9.21ms)	286,311 (4.77ms)	411,442 (13.71ms)	489,093 (8.15ms)	14.05	63.71
1024-point	24	179,265 (5.98ms)	185,282 (3.09ms)	259,345 (8.64ms)	299,392 (4.99ms)	2.21	9.46
	48	327,381 (10.91ms)	339,116 (5.65ms)	484,678 (16.16ms)	561,868 (9.36ms)	5.85	27.20
	72	468,009 (15.60ms)	485,174 (8.09ms)	698,635 (23.29ms)	811,096 (13.52ms)	10.69	52.65
	100	618,919 (20.63ms)	641,127 (10.69ms)	921,394 (30.71ms)	1,095,301 (18.26ms)	13.99	63.71

*: Impulse Response.

Table 3.1.2 - Benchmark of 16-bit FIR Filter: avr32-uc3 optimized

	Number of Taps	I.R.* in SRAM (cycles)				Error	
		wait-state		I.R.* in FLASH (cycles)		Amplitude average (x10 ⁻⁵)	Max. amplitude (x10 ⁻⁵)
		0 at 30MHz		1 at 60MHz			
		0 at 30MHz	1 at 60MHz	0 at 30MHz	1 at 60MHz		
64-point	24	2,439 (81.3us)	2,657 (44.3us)	2,703 (90.1us)	3,054 (50.9us)	2.27	9.46
	48	2,115 (70.5us)	2,309 (38.5us)	2,355 (78.5us)	2,670 (44.5us)	3.09	15.57
256-point	24	12,712 (423.7us)	13,841 (230.7us)	14,128 (470.9us)	15,966 (266.1us)	2.22	9.46
	48	21,604 (720.1us)	23,573 (392.9us)	24,148 (804.9us)	27,390 (456.5us)	5.66	27.20
	72	28,192 (939.7us)	30,785 (513.1us)	31,576 (1.05ms)	35,862 (597.7us)	11.25	52.65
	100	32,966 (1.10ms)	36,014 (600.2us)	36,966 (1.23ms)	42,015 (700.3us)	14.19	63.71
512-point	24	26,408 (880.3us)	28,753 (479.2us)	29,360 (978.7us)	33,182 (553.0us)	2.23	9.46
	48	47,588 (1.59ms)	51,925 (865.4us)	53,204 (1.77ms)	60,350 (1.01ms)	5.78	27.20
	72	66,464 (2.22ms)	72,577 (1.21ms)	74,456 (2.48ms)	84,566 (1.41ms)	10.79	52.65
	100	85,574 (2.85ms)	93,486 (1.56ms)	95,974 (3.20ms)	109,087 (1.82ms)	14.05	63.71
1024-point	24	53,800 (1.79ms)	58,577 (976.3us)	59,824 (1.99ms)	67,614 (1.13ms)	2.21	9.46
	48	99,556 (3.32ms)	108,629 (1.81ms)	111,316 (3.71ms)	126,270 (2.10ms)	5.85	27.20
	72	143,008 (4.77ms)	156,161 (2.60ms)	160,216 (5.34ms)	181,974 (3.03ms)	10.69	52.65
	100	190,790 (6.36ms)	208,430 (3.47ms)	213,990 (7.13ms)	243,231 (4.05ms)	13.99	63.71

*: Impulse Response.

Table 3.2.1 - Benchmark of 32-bit FIR Filter: generic

	Number of Taps	I.R.* in SRAM (cycles)				Error	
		wait-state		I.R.* in FLASH (cycles)		Amplitude average (x10 ⁻⁹)	Max. amplitude (x10 ⁻⁹)
		0 at 30MHz		1 at 60MHz			
		0 at 30MHz	1 at 60MHz	0 at 30MHz	1 at 60MHz		
64-point	24	13,984 (466.1us)	14,365 (239.4us)	16,608 (553.6us)	18,624 (310.4us)	2.10	12.40
	48	11,101 (370.0us)	11,419 (190.3us)	13,311 (443.7us)	14,865 (247.8us)	2.50	14.40
256-point	24	79,073 (2.64ms)	81,181 (1.35ms)	93,985 (3.13ms)	105,408 (1.76ms)	2.30	17.40
	48	135,518 (4.52ms)	139,291 (2.32ms)	162,688 (5.42ms)	181,713 (3.03ms)	2.70	16.60
	72	177,131 (5.90ms)	182,137 (3.04ms)	213,391 (7.11ms)	238,002 (3.97ms)	4.60	31.50
	100	187,238 (6.24ms)	194,313 (3.24ms)	241,717 (8.06ms)	264,808 (4.41ms)	4.60	31.90
512-point	24	165,857 (5.53ms)	170,269 (2.84ms)	197,153 (6.57ms)	221,120 (3.69ms)	2.60	23.10
	48	301,406 (10.05ms)	309,787 (5.16ms)	361,856 (12.06ms)	404,177 (6.74ms)	3.20	23.90
	72	422,123 (14.07ms)	434,041 (7.23ms)	508,559 (16.95ms)	567,218 (9.45ms)	5.10	35.30
	100	492,390 (16.41ms)	510,985 (8.52ms)	635,701 (21.19ms)	696,424 (11.61ms)	5.10	31.90
1024-point	24	339,425 (11.31ms)	348,445 (5.81ms)	403,489 (13.45ms)	452,544 (7.54ms)	3.70	28.40
	48	633,182 (21.11ms)	650,779 (10.85ms)	760,192 (25.34ms)	849,105 (14.15ms)	4.60	29.30
	72	912,107 (30.40ms)	937,849 (15.63ms)	1,098,895 (36.63ms)	1,225,650 (20.43ms)	6.50	35.30
	100	1,102,694 (36.76ms)	1,144,329 (19.07ms)	1,423,669 (47.46ms)	1,559,656 (26.00ms)	6.40	33.30

*: Impulse Response.

Table 3.2.2 - Benchmark of 32-bit FIR Filter: avr32-uc3 optimized

	Number of Taps	I.R.* in SRAM (cycles)		I.R.* in FLASH (cycles)		Error	
		wait-state		wait-state		Amplitude average (x10 ⁻⁹)	Max. amplitude (x10 ⁻⁹)
		0 at 30MHz	1 at 60MHz	0 at 30MHz	1 at 60MHz		
64-point	24	6,479 (216.0us)	6,613 (110.2us)	9,141 (304.7us)	10,918 (182.0us)	2.10	12.40
	48	5,132 (171.1us)	5,245 (87.4us)	7,305 (243.5us)	8,866 (147.8us)	2.50	14.40
256-point	24	36,432 (1.21ms)	37,140 (619.0us)	51,574 (1.72ms)	61,605 (1.03ms)	2.30	12.40
	48	62,157 (2.07ms)	63,420 (1.06ms)	88,906 (2.96ms)	107,937 (1.80ms)	2.70	16.60
	72	81,114 (2.70ms)	82,788 (1.38ms)	116,446 (3.88ms)	142,173 (2.37ms)	4.60	31.50
	100	94,441 (3.15ms)	96,490 (1.61ms)	140,910 (4.70ms)	166,671 (2.78ms)	4.60	31.90
512-point	24	76,368 (2.55ms)	77,844 (1.30ms)	108,150 (3.61ms)	129,189 (2.15ms)	2.60	23.10
	48	138,189 (4.61ms)	140,988 (2.35ms)	197,706 (6.59ms)	240,033 (4.00ms)	3.20	23.90
	72	193,242 (6.44ms)	197,220 (3.29ms)	277,470 (9.25ms)	338,781 (5.65ms)	5.10	35.30
	100	248,297 (8.28ms)	253,674 (4.23ms)	370,542 (12.35ms)	438,287 (7.30ms)	5.10	31.90
1024-point	24	156,240 (5.21ms)	159,252 (2.65ms)	221,302 (7.38ms)	264,357 (4.41ms)	3.70	28.40
	48	290,253 (9.68ms)	296,124 (4.94ms)	415,306 (13.84ms)	504,225 (8.40ms)	4.50	29.30
	72	417,498 (13.92ms)	426,084 (7.10ms)	599,518 (19.99ms)	731,997 (12.20ms)	6.50	35.30
	100	556,009 (18.53ms)	568,042 (9.47ms)	829,806 (27.66ms)	981,519 (16.36ms)	6.40	33.30

*: Impulse Response.

Table 4.1.1 - Benchmark of 16-bit FIR Filter: generic

	Order of the filter	Execution time (cycles)		Error	
		wait-state		Amplitude average (x10 ⁻⁵)	Max. amplitude (x10 ⁻⁵)
		0 at 30MHz	1 at 60MHz		
72-point	2	5,294 (176.5us)	5,717 (95.3us)	1.50	4.00
	7	11,469 (382.3us)	12,802 (213.4us)	1.40	4.30
	12	16,319 (544.0us)	18,387 (306.5us)	2.50	7.60
256-point	2	19,096 (636.5us)	20,621 (343.7us)	1.50	4.00
	7	44,591 (1.49ms)	49,786 (829.8us)	1.40	4.30
	12	68,761 (2.29ms)	77,451 (1.29ms)	3.50	8.90
512-point	2	38,296 (1.28ms)	41,357 (689.3us)	1.50	4.00
	7	90,671 (3.02ms)	101,242 (1.69ms)	1.40	4.30
	12	141,721 (4.72ms)	159,627 (2.66ms)	3.70	8.90
1024-point	2	76,696 (2.56ms)	82,829 (1.38ms)	1.50	4.00
	7	182,831 (6.09ms)	204,154 (3.40ms)	1.40	4.30
	12	287,641 (9.59ms)	323,979 (5.40ms)	3.90	8.90

Table 4.1.2 - Benchmark of 16-bit FIR Filter: avr32-uc3 optimized

	Order of the filter	Execution time (cycles)		Error	
		wait-state		Amplitude average (x10 ⁻⁵)	Max. amplitude (x10 ⁻⁵)
		0 at 30MHz	1 at 60MHz		
72-point	2	2,725 (90.8us)	2,871 (47.9us)	1.20	4.00
	7	4,332 (144.4us)	4,682 (78.0us)	1.90	6.90
	12	6,089 (203.0us)	6,534 (108.9us)	3.30	8.40
256-point	2	9,167 (305.6us)	9,633 (160.6us)	1.10	4.00
	7	15,006 (500.2us)	16,228 (270.5us)	1.70	6.90
	12	22,283 (742.8us)	23,876 (397.9us)	5.10	12.40
512-point	2	18,127 (604.2us)	19,041 (317.4us)	1.10	4.00
	7	29,854 (995.1us)	32,292 (538.2us)	1.70	6.90
	12	44,811 (1.49ms)	48,004 (800.1us)	5.60	12.40
1024-point	2	36,047 (1.20ms)	37,857 (631.0us)	1.10	4.00
	7	59,550 (1.99ms)	64,420 (1.07ms)	1.70	6.90
	12	89,867 (3.00ms)	96,260 (1.60ms)	5.80	12.40

Table 4.2.2 - Benchmark of 32-bit FIR Filter: generic

	Order of the filter	Execution time (cycles)		Error	
		wait-state		Amplitude average (x10 ⁻⁹)	Max. amplitude (x10 ⁻⁹)
		0 at 30MHz	1 at 60MHz		
72-point	2	7,582 (252.7us)	8,072 (134.5us)	1.00	3.80
	7	14,517 (483.9us)	15,922 (265.4us)	1.50	7.00
	12	19,952 (665.1us)	22,097 (368.3us)	2.60	10.30
256-point	2	27,456 (915.2us)	29,232 (487.2us)	0.80	3.80
	7	56,471 (1.88ms)	61,922 (1.03ms)	1.50	7.00
	12	83,986 (2.80ms)	92,937 (1.55ms)	1.50	10.30
512-point	2	55,104 (1.84ms)	58,672 (977.9us)	0.80	3.80
	7	114,839 (3.83ms)	125,922 (2.10ms)	1.50	7.00
	12	173,074 (5.77ms)	191,497 (3.19ms)	1.40	10.30
1024-point	2	110,400 (3.68ms)	117,552 (1.96ms)	0.80	3.80
	7	231,575 (7.72ms)	253,922 (4.23ms)	1.50	7.00
	12	351,250 (11.71ms)	388,617 (6.48ms)	1.30	10.30

Table 4.2.1 - Benchmark of 32-bit FIR Filter: avr32-uc3 optimized

	Order of the filter	Execution time (cycles)		Error	
		wait-state		Amplitude average (x10 ⁻⁹)	Max. amplitude (x10 ⁻⁹)
		0 at 30MHz	1 at 60MHz		
72-point	2	5,297 (176.6us)	5,662 (94.4us)	1.00	3.80
	7	8,859 (295.3us)	9,233 (153.9us)	1.50	7.00
	12	11,670 (389.0us)	12,239 (204.0us)	3.00	10.30
256-point	2	18,731 (624.4us)	20,014 (333.6us)	0.80	3.80
	7	33,333 (1.11ms)	34,625 (577.1us)	1.50	7.00
	12	46,816 (1.56ms)	48,855 (814.3us)	1.80	10.30
512-point	2	37,419 (1.25ms)	39,982 (666.4us)	0.80	3.80
	7	67,381 (2.25ms)	69,953 (1.17ms)	1.60	7.00
	12	95,712 (3.19ms)	99,799 (1.66ms)	1.70	10.30
1024-point	2	74,795 (2.49ms)	79,918 (1.33ms)	0.80	3.80
	7	135,477 (4.52ms)	140,609 (2.34ms)	1.60	7.00
	12	193,504 (6.45ms)	201,687 (3.36ms)	1.60	10.30



Headquarters

Atmel Corporation
2325 Orchard Parkway
San Jose, CA 95131
USA
Tel: 1(408) 441-0311
Fax: 1(408) 487-2600

International

Atmel Asia
Room 1219
Chinachem Golden Plaza
77 Mody Road Tsimshatsui
East Kowloon
Hong Kong
Tel: (852) 2721-9778
Fax: (852) 2722-1369

Atmel Europe
Le Krebs
8, Rue Jean-Pierre Timbaud
BP 309
78054 Saint-Quentin-en-
Yvelines Cedex
France
Tel: (33) 1-30-60-70-00
Fax: (33) 1-30-60-71-11

Atmel Japan
9F, Tonetsu Shinkawa Bldg.
1-24-8 Shinkawa
Chuo-ku, Tokyo 104-0033
Japan
Tel: (81) 3-3523-3551
Fax: (81) 3-3523-7581

Product Contact

Web Site
www.atmel.com

Technical Support
Enter Product Line E-mail

Sales Contact
www.atmel.com/contacts

Literature Requests
www.atmel.com/literature

Disclaimer: The information in this document is provided in connection with Atmel products. No license, express or implied, by estoppel or otherwise, to any intellectual property right is granted by this document or in connection with the sale of Atmel products. **EXCEPT AS SET FORTH IN ATMEL'S TERMS AND CONDITIONS OF SALE LOCATED ON ATMEL'S WEB SITE, ATMEL ASSUMES NO LIABILITY WHATSOEVER AND DISCLAIMS ANY EXPRESS, IMPLIED OR STATUTORY WARRANTY RELATING TO ITS PRODUCTS INCLUDING, BUT NOT LIMITED TO, THE IMPLIED WARRANTY OF MERCHANTABILITY, FITNESS FOR A PARTICULAR PURPOSE, OR NON-INFRINGEMENT. IN NO EVENT SHALL ATMEL BE LIABLE FOR ANY DIRECT, INDIRECT, CONSEQUENTIAL, PUNITIVE, SPECIAL OR INCIDENTAL DAMAGES (INCLUDING, WITHOUT LIMITATION, DAMAGES FOR LOSS OF PROFITS, BUSINESS INTERRUPTION, OR LOSS OF INFORMATION) ARISING OUT OF THE USE OR INABILITY TO USE THIS DOCUMENT, EVEN IF ATMEL HAS BEEN ADVISED OF THE POSSIBILITY OF SUCH DAMAGES.** Atmel makes no representations or warranties with respect to the accuracy or completeness of the contents of this document and reserves the right to make changes to specifications and product descriptions at any time without notice. Atmel does not make any commitment to update the information contained herein. Unless specifically provided otherwise, Atmel products are not suitable for, and shall not be used in, automotive applications. Atmel's products are not intended, authorized, or warranted for use as components in applications intended to support or sustain life.

© 2007 Atmel Corporation. All rights reserved. Atmel®, logo and combinations thereof, AVR® and others are registered trademarks or trademarks of Atmel Corporation or its subsidiaries. Other terms and product names may be trademarks of others.