

To our customers,

Old Company Name in Catalogs and Other Documents

On April 1st, 2010, NEC Electronics Corporation merged with Renesas Technology Corporation, and Renesas Electronics Corporation took over all the business of both companies. Therefore, although the old company name remains in this document, it is a valid Renesas Electronics document. We appreciate your understanding.

Renesas Electronics website: <http://www.renesas.com>

April 1st, 2010
Renesas Electronics Corporation

Issued by: Renesas Electronics Corporation (<http://www.renesas.com>)

Send any inquiries to <http://www.renesas.com/inquiry>.

Notice

1. All information included in this document is current as of the date this document is issued. Such information, however, is subject to change without any prior notice. Before purchasing or using any Renesas Electronics products listed herein, please confirm the latest product information with a Renesas Electronics sales office. Also, please pay regular and careful attention to additional and different information to be disclosed by Renesas Electronics such as that disclosed through our website.
2. Renesas Electronics does not assume any liability for infringement of 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. No license, express, implied or otherwise, is granted hereby under any patents, copyrights or other intellectual property rights of Renesas Electronics or others.
3. You should not alter, modify, copy, or otherwise misappropriate any Renesas Electronics product, whether in whole or in part.
4. 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 of these circuits, software, and information in the design of your equipment. Renesas Electronics assumes no responsibility for any losses incurred by you or third parties arising from the use of these circuits, software, or information.
5. When exporting the products or technology described in this document, you should comply with the applicable export control laws and regulations and follow the procedures required by such laws and regulations. You should not use Renesas Electronics products or the technology described in this document for any purpose relating to military applications or use by the military, including but not limited to the development of weapons of mass destruction. Renesas Electronics products and technology may 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.
6. Renesas Electronics has used reasonable care in preparing the information included in this document, but Renesas Electronics does not warrant that such information is error free. Renesas Electronics assumes no liability whatsoever for any damages incurred by you resulting from errors in or omissions from the information included herein.
7. Renesas Electronics products are classified according to the following three quality grades: “Standard”, “High Quality”, and “Specific”. The recommended applications for each Renesas Electronics product depends on the product’s quality grade, as indicated below. You must check the quality grade of each Renesas Electronics product before using it in a particular application. You may not use any Renesas Electronics product for any application categorized as “Specific” without the prior written consent of Renesas Electronics. Further, you may not use any Renesas Electronics product for any application for which it is not intended without the prior written consent of Renesas Electronics. Renesas Electronics shall not be in any way liable for any damages or losses incurred by you or third parties arising from the use of any Renesas Electronics product for an application categorized as “Specific” or for which the product is not intended where you have failed to obtain the prior written consent of Renesas Electronics. The quality grade of each Renesas Electronics product is “Standard” unless otherwise expressly specified in a Renesas Electronics data sheets or data books, etc.
 - “Standard”: Computers; office equipment; communications equipment; test and measurement equipment; audio and visual equipment; home electronic appliances; machine tools; personal electronic equipment; and industrial robots.
 - “High Quality”: Transportation equipment (automobiles, trains, ships, etc.); traffic control systems; anti-disaster systems; anti-crime systems; safety equipment; and medical equipment not specifically designed for life support.
 - “Specific”: Aircraft; aerospace equipment; submersible repeaters; nuclear reactor control systems; medical equipment or systems for life support (e.g. artificial life support devices or systems), surgical implantations, or healthcare intervention (e.g. excision, etc.), and any other applications or purposes that pose a direct threat to human life.
8. You should use the Renesas Electronics products described in this document within the range specified by Renesas Electronics, especially with respect to the maximum rating, operating supply voltage range, movement power voltage range, heat radiation characteristics, installation and other product characteristics. Renesas Electronics shall have no liability for malfunctions or damages arising out of the use of Renesas Electronics products beyond such specified ranges.
9. Although Renesas Electronics endeavors to improve the quality and reliability of its products, semiconductor products have specific characteristics such as the occurrence of failure at a certain rate and malfunctions under certain use conditions. Further, Renesas Electronics products are not subject to radiation resistance design. Please be sure to implement safety measures to guard them against the possibility of physical injury, and injury or damage caused by fire in the event of the failure of a Renesas Electronics product, 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, please evaluate the safety of the final products or system 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. Please use Renesas Electronics products in compliance with all applicable laws and regulations that regulate the inclusion or use of controlled substances, including without limitation, the EU RoHS Directive. Renesas Electronics assumes no liability for damages or losses occurring as a result of your noncompliance with applicable laws and regulations.
11. This document may not be reproduced or duplicated, in any form, in whole or in part, without prior written consent of Renesas Electronics.
12. Please contact a Renesas Electronics sales office if you have any questions regarding the information contained in this document or Renesas Electronics products, or if you have any other inquiries.

(Note 1) “Renesas Electronics” as used in this document means Renesas Electronics Corporation and also includes its majority-owned subsidiaries.

(Note 2) “Renesas Electronics product(s)” means any product developed or manufactured by or for Renesas Electronics.

R32C/100 Series

Assembler Optimised 64 Point FFT

Introduction

This Application Note describes a 64-point complex valued radix 2 FFT (and IFFT) for Q14 fixed point values. The algorithm is implemented such that a single, hand-optimized assembler routine can perform both, FFT and IFFT, only depending on the specified frequency vector w - without sacrificing performance. The gain of the described optimized assembler FFT leads into a reduction of 50% of the processing time. The assembler based FFT performs a 64-point complex radix 2 FFT within 0.3814ms @ 50MHz.

Target Device

The code for this Application Note was developed on the R32C/100 Series in combination with NC100 compiler.

Contents

1. FFT / IFFT	2
2. Use of FFT / IFFT	2
3. Code Listing	3
Website and Support	20

1. FFT / IFFT

The Fast Fourier Transform (FFT) is a fast realization of the Discrete Fourier Transform (DFT). There are several C-based FFT / IFFT implementation available which could be reused for MCUs. In several cases a straightforward port of such an implementation does not achieve the same performance regarding speed and accuracy as assembler based implementation.

The FFT / IFFT of this application note is Decimation in TIME (DIT) based, which means the FFT requires the input data in a specific order and returns the output data in sequential order. The specific arrangement of the input values is done by the function `PerformDigitReversal64(COMPLEX x[])`. Based on this, the function `PerformDigitReversal64(COMPLEX x[])` shall be performed before calling the FFT calculation. The 64-point FFT / IFFT is implemented in 2.14 fixed point.

2. Use of FFT / IFFT

The function `void FastFourierTransform64(COMPLEX x[64], const COMPLEX w[64]);` performs 64-point FFT on vector `x` with frequency vector `w`. Two frequency vectors are provided: `g_w64Q14` for FFT and its conjugate `g_w64Q14conj` for IFFT. The provided frequency vectors are generated by Matlab®. The command

```
k=0:63; int16(fi(1/2 * (cos(k * 2 * pi / 64) - i * sin(k * 2 * pi / 64)), 1, 16, 14))
```

is used to create the normalized frequency vector $w = \exp(-jk * 2\pi/N)$ for forward Fourier Transform.

For inverse Fourier Transform, the normalized frequency vector $w^* = \exp(jk * 2\pi/N)$ is created using the Matlab® command

```
k=0:63; int16(fi(1/2 * (cos(k * 2 * pi / 64) + i * sin(k * 2 * pi / 64)), 1, 16, 14)).
```

The function `void PerformDigitReversal64(COMPLEX x[64]);` performs digit reversal on vector `x`, which needs to be done before the actual FFT / IFFT.

3. Code Listing

```

/*****
* DISCLAIMER

* This software is supplied by Renesas Technology Corp. and is only
* intended for use with Renesas products. No other uses are authorized.

* This software is owned by Renesas Technology Corp. and is protected under
* all applicable laws, including copyright laws.

* THIS SOFTWARE IS PROVIDED "AS IS" AND RENESAS MAKES NO WARRANTIES
* REGARDING THIS SOFTWARE, WHETHER EXPRESS, IMPLIED OR STATUTORY,
* INCLUDING BUT NOT LIMITED TO WARRANTIES OF MERCHANTABILITY, FITNESS FOR A
* PARTICULAR PURPOSE AND NON-INFRINGEMENT. ALL SUCH WARRANTIES ARE EXPRESSLY
* DISCLAIMED.

* TO THE MAXIMUM EXTENT PERMITTED NOT PROHIBITED BY LAW, NEITHER RENESAS
* TECHNOLOGY COPR. NOR ANY OF ITS AFFILIATED COMPANIES SHALL BE LIABLE
* FOR ANY DIRECT, INDIRECT, SPECIAL, INCIDENTAL OR CONSEQUENTIAL DAMAGES
* FOR ANY REASON RELATED TO THE THIS SOFTWARE, EVEN IF RENESAS OR ITS
* AFFILIATES HAVE BEEN ADVISED OF TEH POSSIBILITY OF SUCH DAMAGES.

* Renesas reserves the right, without notice, to make changes to this
* software and to discontinue the availability of this software.
* By using this software, you agree to the additional terms and
* conditions found by accessing the following link:
* http://www.renesas.com/disclaimer
*****/

/* Copyright (C) 2008. Renesas Technology Corp., All Rights Reserved. */

/*****
* File Name      : FFT.c
* Version        : 1.0
* Device(s)     : R32C/118
* Tool-Chain    : NC100
* H/W Platform  :
* Description    : FFT/IFFT tutorial code
* Limitations   : None
*****/
* History : DD.MM.YYYY Version Description
*         : 23.10.2008 1.00    First Release
*
*****/

/*****
Includes <System Includes> , "Project Includes"
*****/
#include <stdlib.h>
#include <string.h>
#include "Types.h"
#include "FastFourierTransform.h"

/* Initialise R32C processor clock */
void ConfigureOperatingFrequency(void);

```

```

/* create input data*/
COMPLEX acmplxShortSequence[64] =
{
    { 0000, 0000 }, { 0000, 0000 }, { 0000, 0000 }, { 0000, 0000 },
    { -24117, -24117 }, { 0000, 0000 }, { 0000, 0000 }, { 0000, 0000 },
    { -24117, -24117 }, { 0000, 0000 }, { 0000, 0000 }, { 0000, 0000 },
    { 24117, 24117 }, { 0000, 0000 }, { 0000, 0000 }, { 0000, 0000 },
    { 24117, 24117 }, { 0000, 0000 }, { 0000, 0000 }, { 0000, 0000 },
    { 24117, 24117 }, { 0000, 0000 }, { 0000, 0000 }, { 0000, 0000 },
    { 24117, 24117 }, { 0000, 0000 }, { 0000, 0000 }, { 0000, 0000 },
    { 0000, 0000 }, { 0000, 0000 }, { 0000, 0000 }, { 0000, 0000 },
    { 0000, 0000 }, { 0000, 0000 }, { 0000, 0000 }, { 0000, 0000 },
    { 0000, 0000 }, { 0000, 0000 }, { 0000, 0000 }, { 0000, 0000 },
    { 24117, 24117 }, { 0000, 0000 }, { 0000, 0000 }, { 0000, 0000 },
    { -24117, -24117 }, { 0000, 0000 }, { 0000, 0000 }, { 0000, 0000 },
    { 24117, 24117 }, { 0000, 0000 }, { 0000, 0000 }, { 0000, 0000 },
    { -24117, -24117 }, { 0000, 0000 }, { 0000, 0000 }, { 0000, 0000 },
    { -24117, -24117 }, { 0000, 0000 }, { 0000, 0000 }, { 0000, 0000 },
    { 24117, 24117 }, { 0000, 0000 }, { 0000, 0000 }, { 0000, 0000 }
};

/*****
* Include      : none
* Declaration  : void main(void)
* Function Name: main
* Description   : Main function to demonstrate assembler based FFT / IFFT
* Argument     : none
* Return Value : none
*****/
void main(void)
{
    unsigned char ucCounter;

    ConfigureOperatingFrequency();

    /* Perform IFFT */
    PerformDigitReversal64(acmplxShortSequence);
    FastFourierTransform64(acmplxShortSequence, g_w64Q14conj);
    /* 1/N norming is not necessary due to unconditional scaling */
    /* and factor 1/N between FFT and IFFT */

    /* Perform FFT */

```

```
PerformDigitReversal64(acmplxShortSequence);
FastFourierTransform64(acmplxShortSequence, g_w64Q14);

/* Perform N norming due to unconditional scaling */
for(ucCounter = 0; ucCounter < 64; ucCounter++)
{
    acmplxShortSequence[ucCounter].real *= 64;
    acmplxShortSequence[ucCounter].imag *= 64;
}
}
```

```

;*****
; File Name      : FastFourierTransform_r32.a30
; Version       : 1.0
; Device(s)    : R32C/118
; Tool-Chain   : NC100
; H/W Platform :
; Description   : Implements a 64-point complex-valued FFT (and IFFT) for Q14
;                fixed point values. The algorithm is implemented such that
;                a single, hand-optimized assembler routine can perform both,
;                FFT and IFFT, only depending on the specified frequency
;                vector w - without sacrificing performance
; Limitations  : None
;*****
; History : DD.MM.YYYY Version Description
;         : 23.10.2008 1.00    First Release
;
;*****/

; Use this as your C prototype:
; void FastFourierTransform64(COMPLEX x[], const COMPLEX w[]);
; #pragma PARAMETER FastFourierTransform64(A0, A1)

        .section    program, code, align

;-----
; FastFourierTransform64()

        .glob      _FastFourierTransform64

; Registers on entry:
; A0    pointer to an array of 64 complex pairs of Q14 fixed point samples
; A1    pointer to the complex frequency vector (or its conjugate for IFFT)
;*****
; Include      : none
; Declaration  : void main(void)
; Function Name: main
; Description  : Main function to demonstrate assembler based FFT / IFFT
; Argument     :
;
;               Registers on entry:
;               A0 pointer to an array of 64 complex pairs of Q14 fixed
;               point samples
;               A1 pointer to the complex frequency vector (or its conjugate
;               for IFFT)
; Return Value : none
;*****/

_FastFourierTransform64:

        mov.l      a0, a0b      ; a0b = a0 = x
        mov.l      a1, a1b      ; a1b = a1 = w

        mov.b      #1, r2hb     ; r2h = n2 = 1
        mov.b      #128, r1hb   ; r1h = ie = 32 * 4

        mov.b      #0, r01b     ; r0l = i = 0

loop_i:  mov.b      r2hb, r31b   ; r3l = n1 = n2

```



```

mov.b      #0, r1lb      ; r1l = ia = 0
shl.b      #1, r2hb      ; n2 <=<= 1;

mov.b      #0, r0hb      ; r0h = j = 0

loop_j:    extz.bl      r1lb, a1      ; a1 = ia
add.l      alb, a1      ; a1 += &w[0]
mov.l      [a1], r7r5    ; r7 = w[ia].imag, r5 = w[ia].real

mov.b      r0hb, r2lb    ; r2lb = k = j

loop_k:    extz.bl      r2lb, a2      ; a2 = k
shl.l      #2, a2      ; a2 *= 4 (sizeof COMPLEX)
add.l      a0b, a2      ; a2 += &x[0], i.e. a2 = &x[k]

extz.bl    r2lb, a0      ; a0 = k
extz.bl    r3lb, r2r0    ; r2r0 = n1
add.l      r2r0, a0      ; a0 = k + n1
shl.l      #2, a0      ; a0 *= 4 (sizeof COMPLEX)
add.l      a0b, a0      ; a0 += &x[0], i.e. a0 = &x[m = k + n1]

mov.l      [a0], r6r4    ; load x[m] into r6(x[m].imag), r4(x[m].real)

mov.w      r4, r0
emul.w     r5, r0      ; multiply x[m].real and w[i,j].real,
                    ; result in r2r0

mov.w      r6, r1
emul.w     r7, r1      ; multiply x[m].imag and w[i,j].imag,
                    ; result in r3r1

sub.l      r3r1, r2r0    ; subtract r3r1 from r2r0, result in r2r0
sha.l      #-13, r2r0    ; fixed point scaling; r0 now holds t.real
mov.l      r2r0, a3      ; save r2r0 in a3

mov.w      r4, r0
emul.w     r7, r0      ; multiply x[m].real and w[i,j].imag,
                    ; result in r2r0

mov.w      r6, r1
emul.w     r5, r1      ; multiply x[m].imag and w[i,j].real,
                    ; result in r3r1

add.l      r2r0, r3r1    ; add r2r0 to r3r1, result in r3r1
sha.l      #-13, r3r1    ; fixed point scaling; r1 now holds t.imag

mov.l      a3, r2r0      ; restore r2r0 (r0 holds t.real now)
mov.w      r1, r2      ; t.real in r0, t.imag in r2, r3r1 becomes
                    ; available

mov.l      [a2], a1      ; load x[k] into a1

mov.l      a1, r6r4      ; r6 = x[k].imag, r4 = x[k].real
exts.wl    r4, r6r4      ; r6r4 = x[k].real
exts.wl    r0, r3r1      ; r3r1 = t.real
sub.l      r3r1, r6r4    ; r6r4 = x[k].real - t.real
sha.l      #-1, r6r4     ; fixed point scaling; r4 now holds x[m].real

mov.l      a1, r3r1      ; r3r1 = x[k]
exts.wl    r3, r3r1      ; r3r1 = x[k].imag
exts.wl    r2, a3        ; a3 = t.imag
sub.l      a3, r3r1      ; r3r1 = x[k].imag - t.imag
sha.l      #-1, r3r1     ; fixed point scaling; r1 now holds x[m].real

```

```

mov.w      r1, r6      ; let r6r4 hold the complex pair x[m]
mov.l      r6r4, [a0]  ; store r6r4 as x[m]

mov.l      a1, r6r4    ; restore x[k] from a1
exts.wl    r4, r6r4    ; r6r4 = real part of x[k]
exts.wl    r0, r3r1    ; r0 -> r3r1
add.l      r3r1, r6r4  ; r6r4 = x[k].real + t.real
sha.l      #-1, r6r4   ; fixed point scaling; r4 now holds x[m].real

mov.l      a1, r3r1    ; restore x[k] from a1
exts.wl    r3, r3r1    ; r3r1 = imaginary part of x[k]
exts.wl    r2, a3      ; move imaginary part of t to a3
add.l      a3, r3r1    ; add imaginary parts, result int r3r1
sha.l      #-1, r3r1   ; fixed point scaling r3r1 = x[k].imag + t.imag
mov.w      r1, r6      ; let register r6r4 hold the complex pair x[k]
mov.l      r6r4, [a2]  ; store r6r4 as x[k]

add.b      r2hb, r2lb  ; k = k + n2
cmp.b      #64, r2lb   ; loop while k < 64
jltu

inc.b      r0hb        ; j++
add.b      r1hb, r1lb  ; ia += ie
cmp.b      r3lb, r0hb  ; loop while j < n1
jltu

inc.b      r0lb        ; i++
shl.b      #-1, r1hb   ; ie >= 1
cmp.b      #6, r0lb    ; loop while i < 6
jltu

rts

.end

```

```

/*****
* File Name      : FastFourierTransform.c
* Version        : 1.0
* Device(s)     : R32C/118
* Tool-Chain    : NC100
* H/W Platform  :
* Description    : Implements a 64-point complex-valued FFT (and IFFT) for Q14
*                 fixed point values. The algorithm is implemented such that
*                 a single, hand-optimized assembler routine can perform both,
*                 FFT and IFFT, only depending on the specified frequency vector
*                 w - without sacrificing performance
* Limitations   : None
*****
* History : DD.MM.YYYY Version Description
*         : 23.10.2008 1.00   First Release
*
*****/

/*****
Includes <System Includes> , "Project Includes"
*****/
#include "types.h"
#include "FastFourierTransform.h"

/*****
* Normalized frequency vector w = exp(-jk * 2pi/N)
*
* To be used for forward Fourier transform
*
* MATLAB® command to generate the complex frequency vector:
* k=0:63; int16(fi(1/2*(cos(k*2*pi/64)-i*sin(k*2*pi/64)),1,16,14))
*****/
const COMPLEX g_w64Q14[64] =
{
    { 8192, 0 }, { 8153, -803 }, { 8035, -1598 }, { 7839, -2378 },
    { 7568, -3135 }, { 7225, -3862 }, { 6811, -4551 }, { 6333, -5197 },
    { 5793, -5793 }, { 5197, -6333 }, { 4551, -6811 }, { 3862, -7225 },
    { 3135, -7568 }, { 2378, -7839 }, { 1598, -8035 }, { 803, -8153 },
    { 0, -8192 }, { -803, -8153 }, { -1598, -8035 }, { -2378, -7839 },
    { -3135, -7568 }, { -3862, -7225 }, { -4551, -6811 }, { -5197, -6333 },
    { -5793, -5793 }, { -6333, -5197 }, { -6811, -4551 }, { -7225, -3862 },
    { -7568, -3135 }, { -7839, -2378 }, { -8035, -1598 }, { -8153, -803 },
    { -8192, 0 }, { -8153, 803 }, { -8035, 1598 }, { -7839, 2378 },
    { -7568, 3135 }, { -7225, 3862 }, { -6811, 4551 }, { -6333, 5197 },
    { -5793, 5793 }, { -5197, 6333 }, { -4551, 6811 }, { -3862, 7225 },
    { -3135, 7568 }, { -2378, 7839 }, { -1598, 8035 }, { -803, 8153 },
    { 0, 8192 }, { 803, 8153 }, { 1598, 8035 }, { 2378, 7839 },
    { 3135, 7568 }, { 3862, 7225 }, { 4551, 6811 }, { 5197, 6333 },
    { 5793, 5793 }, { 6333, 5197 }, { 6811, 4551 }, { 7225, 3862 },
    { 7568, 3135 }, { 7839, 2378 }, { 8035, 1598 }, { 8153, 803 }
};

/*****
* Conjugate of normalized frequency vector w* = exp(jk * 2pi/N)
*
* To be used for inverse Fourier transform

```

```

*
* MATLAB® command to generate the complex frequency vector (conjugate):
* k=0:63; int16(fi(1/2*(cos(k*2*pi/64)+i*sin(k*2*pi/64)),1,16,14))
*****/
const COMPLEX g_w64Q14conj[64] =
{
    { 8192, 0 }, { 8153, 803 }, { 8035, 1598 }, { 7839, 2378 },
    { 7568, 3135 }, { 7225, 3862 }, { 6811, 4551 }, { 6333, 5197 },
    { 5793, 5793 }, { 5197, 6333 }, { 4551, 6811 }, { 3862, 7225 },
    { 3135, 7568 }, { 2378, 7839 }, { 1598, 8035 }, { 803, 8153 },
    { 0, 8192 }, { -803, 8153 }, { -1598, 8035 }, { -2378, 7839 },
    { -3135, 7568 }, { -3862, 7225 }, { -4551, 6811 }, { -5197, 6333 },
    { -5793, 5793 }, { -6333, 5197 }, { -6811, 4551 }, { -7225, 3862 },
    { -7568, 3135 }, { -7839, 2378 }, { -8035, 1598 }, { -8153, 803 },
    { -8192, 0 }, { -8153, -803 }, { -8035, -1598 }, { -7839, -2378 },
    { -7568, -3135 }, { -7225, -3862 }, { -6811, -4551 }, { -6333, -5197 },
    { -5793, -5793 }, { -5197, -6333 }, { -4551, -6811 }, { -3862, -7225 },
    { -3135, -7568 }, { -2378, -7839 }, { -1598, -8035 }, { -803, -8153 },
    { 0, -8192 }, { 803, -8153 }, { 1598, -8035 }, { 2378, -7839 },
    { 3135, -7568 }, { 3862, -7225 }, { 4551, -6811 }, { 5197, -6333 },
    { 5793, -5793 }, { 6333, -5197 }, { 6811, -4551 }, { 7225, -3862 },
    { 7568, -3135 }, { 7839, -2378 }, { 8035, -1598 }, { 8153, -803 }
};

/*****
* Include      : FastFourierTransform.h
* Declaration  : static void swap(COMPLEX x[], const int i, const int j)
* Function Name: swap
* Description  : Swap, exchanges two pairs of complex values i, j
* Argument     : COMPLEX x[], const int i, const int j
* Return Value : none
*****/
static void swap(COMPLEX x[], const int i, const int j)
{
    const COMPLEX t = x[i];

    x[i]= x[j];
    x[j] = t;
}

/*****
* Include      : FastFourierTransform.h
* Declaration  : void PerformDigitReversal64(COMPLEX x[])
* Function Name: PerformDigitReversal64
* Description  : PerformDigitReversal64, digital reversal for 64-point
*              radix-2 FFT
* Argument     : COMPLEX x[]
* Return Value : none
*****/
void PerformDigitReversal64(COMPLEX x[])
{
    swap(x, 1, 32);
    swap(x, 2, 16);
    swap(x, 3, 48);
    swap(x, 4, 8);
    swap(x, 5, 40);
    swap(x, 6, 24);
}

```

```
swap(x, 7, 56);  
swap(x, 9, 36);  
swap(x, 10, 20);  
swap(x, 11, 52);  
swap(x, 13, 44);  
swap(x, 14, 28);  
swap(x, 15, 60);  
swap(x, 17, 34);  
swap(x, 19, 50);  
swap(x, 21, 42);  
swap(x, 22, 26);  
swap(x, 23, 58);  
swap(x, 25, 38);  
swap(x, 27, 54);  
swap(x, 29, 46);  
swap(x, 31, 62);  
swap(x, 35, 49);  
swap(x, 37, 41);  
swap(x, 39, 57);  
swap(x, 43, 53);  
swap(x, 47, 61);  
swap(x, 55, 59);  
}
```

```

/*****
* File Name      : FastFourierTransform.h
* Version        : 1.0
* Device(s)     : R32C/118
* Tool-Chain    : NC100
* H/W Platform  :
* Description    : Implements a 64-point complex-valued FFT (and IFFT) for Q14
*                 fixed point values. The algorithm is implemented such that
*                 a single, hand-optimized assembler routine can perform both,
*                 FFT and IFFT, only depending on the specified frequency vector
*                 w - without sacrificing performance
* Limitations   : None
*****/
* History : DD.MM.YYYY Version Description
*         : 23.10.2008 1.00    First Release
*
*****/

/*****
Imported global variables and functions (from other files)
*****/

/* Normalized frequency vector w, for forward Fourier transform */
extern const COMPLEX g_w64Q14[64];

/* Conjugate of normalized frequency vector w, for inverse Fourier transform */
extern const COMPLEX g_w64Q14conj[64];

/* FastFourierTransform64, 64-point radix-2 FFT of Q14 fixed point values */
void FastFourierTransform64(COMPLEX x[], const COMPLEX w[64]);

#ifdef R32C100
#pragma PARAMETER FastFourierTransform64(a0, a1)
#endif

/* PerformDigitReversal64, digital reversal for 64-point radix-2 FFT */
void PerformDigitReversal64(COMPLEX x[]);

```

```
/******  
* File Name      : types.h  
* Version        : 1.0  
* Device(s)     : R32C/118  
* Tool-Chain    : NC100  
* H/W Platform  :  
* Description    : Declaration of principal types such as COMPLEX  
* Limitations   : None  
*****  
* History : DD.MM.YYYY Version Description  
*       : 23.10.2008 1.00      First Release  
*  
*****/  
  
/* COMPLEX - pair of complex values, 16-bit fixed point */  
  
typedef struct tagCOMPLEX  
{  
    signed short real; /*16-bit fixed point*/  
    signed short imag; /*16-bit fixed point*/  
} COMPLEX;
```

```

/*****
FILE NAME      hwsetup.c
DESCRIPTION    Hardware Setup functions for RSKR32C111

Copyright     : 2005 Renesas Technology Europe Ltd.
Copyright     : 2005 Renesas Technology Corporation.
All Rights Reserved

*****/
/*****
Revision History
DD.MM.YYYY OSO-UID Description
14.09.2005 RTE-DDE First Release
*****/

/*****
System Includes
*****/
// define settings for clock circuit
// select external clock
//#define fXin  6
//#define fXin  8
//#define fXin 10
#define fXin 16

// select PLL frequency
//#define PLL    96
//#define PLL   100
//#define PLL   120
#define PLL    128
//#define PLL   144
//#define PLL   160
//#define PLL   180
//#define PLL   192

// select divider for Base clock
// Base clock = PLL frequency / BCD
#define BCD    2
//#define BCD   3
//#define BCD   4
//#define BCD   6

// select divider for CPU clock
// CPU clock = Base clock / CCD
#define CCD    1
//#define CCD   2
//#define CCD   3
//#define CCD   4

// select divider for peripheral bus clock
// Peripheral bus clock = Base clock / PCD
//#define PCD    1
#define PCD    2
//#define PCD   3
//#define PCD   4

// select divider for peripheral clock
// Peripheral clock = PLL frequency / PCL
//#define PCL    2

```



```

//#define PCL    4
//#define PCL    6
#define PCL     8

/*****
User Includes
*****/
/* sfr118.h provides a structure to access all of the device registers. */
#include "sfr118.h"
#include "clk_setup.h"

/*****
User Program Code Prototypes
*****/
/* These functions are private so their prototypes are defined locally */
void ConfigureOperatingFrequency(void);

/*****
Function Name:    ConfigureOperatingFrequency
Description:     Sets up operating speed and start sub-clock
Parameters:     none
Return value:    none
*****/
void ConfigureOperatingFrequency(void)
{
    unsigned short wait=0;
    prcr = 0xFF;    // enable write to control registers
    prcr2 = 0x80;   // enable write to CM3 register
    prr = 0xAA;     // enable write to CCR, FMCR, PBC register
                    // value depends on setting of CCR register
    pm2 |= 0x44;    // processor mode register 2: enable clock change
    cm0 = 0x02;     // system clock control register 0: output f8 on CLKout
    cm1 = 0x20;     // system clock control register 1
    cm2 = 0x00;     // oscillation stop detect register
    cm3 = 0x02;     // low speed mode clock control register
    tcspr = 0x00;   // count source prescaler register
    tcspr = 0x80;   // count source prescaler register
    cpsrf = 0x00;   // clock prescaler reset register
    pbc = _PBC;     // Peripheral bus clock
    ccr = CCR1;     // clock control register
    ccr = CCR2;     // clock control register
    prcr = 0xFF;
    plc0 = _plc0;   // pll control register 0
    prcr = 0xFF;
    plc1 = _plc1;   // pll control register 1
    seo = 0;        // pll mode
    while (wait<0x8000)
        wait++;
    bcs = 0;        // base clock source is PLL

    prcr = 0xFF;
    pm3 = _pm3;     // peripheral clock = PLL clock / 4
    pm2 &= ~0x02;   // processor mode register 2: disable clock change
    prcr = 0x00;
}
/*****
End of function ConfigureOperatingFrequency
*****/

```

/******

FILE NAME clk_setup.h

DESCRIPTION Clock Setup

Copyright : 2008 Renesas Technology Europe Ltd.

Copyright : 2008 Renesas Technology Corporation.

All Rights Reserved

*****/

/******

Revision History

DD.MM.YYYY OSO-UID Description

01.10.2008 RTE-FLA First Release

*****/

```

#if CCD==3            // set value for CPU clock
#define _CCR1         0x04
#elif CCD==2
#define _CCR1         0x08
#elif CCD==1
#define _CCR1         0x0C
#elif CCD==4
#define _CCR1         0x00
#else
#error CCD not defined or invalid value
#endif

#if PCD==1            // set value for peripheral bus clock
#define CCR1           _CCR1
#define _PBC           0x0000
#elif PCD==2
#define CCR1           _CCR1+0x10
#define _PBC           0x0504
#elif PCD==3
#define CCR1           _CCR1+0x20
#define _PBC           0x0A0D
#elif PCD==4
#define CCR1           _CCR1+0x30
#define _PBC           0x0F0F
#else
#error PCD not defined or invalid value
#endif

#if BCD==6            // set value for bus clock
#define CCR2           CCR1
#elif BCD==4
#define CCR2           CCR1+0x01
#elif BCD==3
#define CCR2           CCR1+0x02
#elif BCD==2
#define CCR2           CCR1+0x03
#else
#error BCD not defined or invalid value
#endif

#if fXin==6           // set PLL values

```

```

#if PLL==96
#define _plc0      0x45
#define _plc1      0x01
#elif PLL==100
#define _plc0      0x09
#define _plc1      0x02
#elif PLL==120
#define _plc0      0x07
#define _plc1      0x01
#elif PLL==128
#define _plc0      0x8B
#define _plc1      0x02
#elif PLL==144
#define _plc0      0x68
#define _plc1      0x01
#elif PLL==160
#define _plc0      0x0F
#define _plc1      0x02
#elif PLL==180
#define _plc0      0x11
#define _plc1      0x02
#elif PLL==192
#define _plc0      0x8B
#define _plc1      0x01
#else
#error Invalid PLL frequency for f(Xin)=6MHz
#endif
#elif fXin==8
#if PLL==96
#define _plc0      0x68
#define _plc1      0x03
#elif PLL==100
#define _plc0      0x04
#define _plc1      0x01
#elif PLL==120
#define _plc0      0x05
#define _plc1      0x01
#elif PLL==128
#define _plc0      0x45
#define _plc1      0x01
#elif PLL==144
#define _plc0      0x26
#define _plc1      0x01
#elif PLL==160
#define _plc0      0x07
#define _plc1      0x01
#elif PLL==180
#define _plc0      0x08
#define _plc1      0x01
#elif PLL==192
#define _plc0      0x68
#define _plc1      0x01
#else
#error Invalid PLL frequency for f(Xin)=8MHz
#endif
#elif fXin==10
#if PLL==96
#define _plc0      0x68

```

```

#define _plc1      0x04
#elif PLL==100
#define _plc0      0xA6
#define _plc1      0x03
#elif PLL==120
#define _plc0      0x26
#define _plc1      0x02
#elif PLL==128
#define _plc0      0x8B
#define _plc1      0x04
#elif PLL==144
#define _plc0      0x4D
#define _plc1      0x04
#elif PLL==160
#define _plc0      0x45
#define _plc1      0x01
#elif PLL==180
#define _plc0      0x26
#define _plc1      0x02
#elif PLL==192
#define _plc0      0x32
#define _plc1      0x04
#else
#error Invalid PLL frequency for f(Xin)=10MHz
#endif
#elif fXin==16
#if PLL==96
#define _plc0      0x68
#define _plc1      0x07
#elif PLL==100
#define _plc0      0x04
#define _plc1      0x03
#elif PLL==120
#define _plc0      0x05
#define _plc1      0x03
#elif PLL==128
#define _plc0      0x45
#define _plc1      0x03
#elif PLL==144
#define _plc0      0x26
#define _plc1      0x03
#elif PLL==160
#define _plc0      0x07
#define _plc1      0x03
#elif PLL==180
#define _plc0      0x08
#define _plc1      0x03
#elif PLL==192
#define _plc0      0x68
#define _plc1      0x03
#else
#error Invalid PLL frequency for f(Xin)=16MHz
#endif
#else
#error Invalid f(Xin) value
#endif

#if PCL==2          // set value for pm3 to define peripheral clock

```

```
#define _pm3      0x060
#elif PCL==4
#define _pm3      0x040
#elif PCL==6
#define _pm3      0x020
#elif PCL==8
#define _pm3      0x000
#else
#error Invalid divider for peripheral clock
#endif
```

Website and Support

Renesas Technology Website

<http://www.renesas.com/>

Inquiries

<http://www.renesas.com/inquiry>

All trademarks and registered trademarks are the property of their respective owners.

Notes regarding these materials

1. This document is provided for reference purposes only so that Renesas customers may select the appropriate Renesas products for their use. Renesas neither makes warranties or representations with respect to the accuracy or completeness of the information contained in this document nor grants any license to any intellectual property rights or any other rights of Renesas or any third party with respect to the information in this document.
2. Renesas shall have no liability for damages or infringement of any intellectual property or other rights arising out of the use of any information in this document, including, but not limited to, product data, diagrams, charts, programs, algorithms, and application circuit examples.
3. You should not use the products or the technology described in this document for the purpose of military applications such as the development of weapons of mass destruction or for the purpose of any other military use. When exporting the products or technology described herein, you should follow the applicable export control laws and regulations, and procedures required by such laws and regulations.
4. All information included in this document such as product data, diagrams, charts, programs, algorithms, and application circuit examples, is current as of the date this document is issued. Such information, however, is subject to change without any prior notice. Before purchasing or using any Renesas products listed in this document, please confirm the latest product information with a Renesas sales office. Also, please pay regular and careful attention to additional and different information to be disclosed by Renesas such as that disclosed through our website. (<http://www.renesas.com>)
5. Renesas has used reasonable care in compiling the information included in this document, but Renesas assumes no liability whatsoever for any damages incurred as a result of errors or omissions in the information included in this document.
6. When using or otherwise relying on the information in this document, you should evaluate the information in light of the total system before deciding about the applicability of such information to the intended application. Renesas makes no representations, warranties or guaranties regarding the suitability of its products for any particular application and specifically disclaims any liability arising out of the application and use of the information in this document or Renesas products.
7. With the exception of products specified by Renesas as suitable for automobile applications, Renesas products are not designed, manufactured or tested for applications or otherwise in systems the failure or malfunction of which may cause a direct threat to human life or create a risk of human injury or which require especially high quality and reliability such as safety systems, or equipment or systems for transportation and traffic, healthcare, combustion control, aerospace and aeronautics, nuclear power, or undersea communication transmission. If you are considering the use of our products for such purposes, please contact a Renesas sales office beforehand. Renesas shall have no liability for damages arising out of the uses set forth above.
8. Notwithstanding the preceding paragraph, you should not use Renesas products for the purposes listed below:
 - (1) artificial life support devices or systems
 - (2) surgical implantations
 - (3) healthcare intervention (e.g., excision, administration of medication, etc.)
 - (4) any other purposes that pose a direct threat to human life
 Renesas shall have no liability for damages arising out of the uses set forth in the above and purchasers who elect to use Renesas products in any of the foregoing applications shall indemnify and hold harmless Renesas Technology Corp., its affiliated companies and their officers, directors, and employees against any and all damages arising out of such applications.
9. You should use the products described herein within the range specified by Renesas, especially with respect to the maximum rating, operating supply voltage range, movement power voltage range, heat radiation characteristics, installation and other product characteristics. Renesas shall have no liability for malfunctions or damages arising out of the use of Renesas products beyond such specified ranges.
10. Although Renesas endeavors to improve the quality and reliability of its products, IC products have specific characteristics such as the occurrence of failure at a certain rate and malfunctions under certain use conditions. Please be sure to implement safety measures to guard against the possibility of physical injury, and injury or damage caused by fire in the event of the failure of a Renesas product, 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 applicable measures. Among others, since the evaluation of microcomputer software alone is very difficult, please evaluate the safety of the final products or system manufactured by you.
11. In case Renesas products listed in this document are detached from the products to which the Renesas products are attached or affixed, the risk of accident such as swallowing by infants and small children is very high. You should implement safety measures so that Renesas products may not be easily detached from your products. Renesas shall have no liability for damages arising out of such detachment.
12. This document may not be reproduced or duplicated, in any form, in whole or in part, without prior written approval from Renesas.
13. Please contact a Renesas sales office if you have any questions regarding the information contained in this document, Renesas semiconductor products, or if you have any other inquiries.