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.

M16C/26

Measuring Computation Time of a Function Call

1.0 Abstract

The following article discusses a technique for measuring computation time spent during a function call, which can be in C or Assembly, from a main C program for the M16C/26 MCU. The method for calling assembly functions in C has been discussed in a separate application note (see References section). A sample program, which will run on the MSV30262 SKP board, is available for evaluation. This program allows a user to invoke either an assembly or C function call and obtain the computation time in microseconds.

2.0 Introduction

The Renesas M30262 is a 16-bit MCU based on the M16C/60 series CPU core. The MCU features include up to 64K bytes of Flash ROM, 2K bytes of RAM, and 4K bytes of virtual EEPROM. The peripheral set includes 10-bit A/D, UARTs, Timers, DMA, and GPIO.

The basic technique behind measuring the computation times is to use a timer and determine how fast a target function call returns before the timer expires. If the timer expires in T microseconds and the function call returns N times during this interval, then the computation time behind a single function call is given by T/N microseconds.

3.0 Number of Function Call Returns

In order to determine the number of function calls (=N) before the timer expires, we will use an incremental addition function that starts with an initial result of 0 and adds 1 at each call. The target function is placed inside a “while” loop that repeats as long as the timer does not expire. The incremental addition function is inserted in the same “while” loop as the target function (see List 1). This ensures that the incremental addition function will be called along with the target function. Thus, as long as the timer does not expire, the target function (and the incremental addition function) will be called repeatedly and once the timer expires, the result of the incremental addition function will indicate how many times (=N) the target function was called. In the demonstration code that accompanies this application note, we have used an assembly function that adds two integers (addend and augend). The method to call assembly function from C program has been described in the separate application note (see References section). Of course the target function could be a C function as well.

4.0 Timer Set-up

In order to measure the time associated with timer expiration, Timer A0 is configured in timer mode. In this mode, the timer counts an internally generated count source. The highest possible value (0xFFFF) is loaded in the Timer Counter register to get a maximum expiration length. Since the CPU clock is set at “divide by one” mode and Xin from SKP board crystal is 20 MHz, the timer expiration time can be calculated as: 0xFFFF/20MHz=

0xFFFF X 50 ns = 3276.750 us (microseconds). The timer is started just before making the target function call and stopped just after returning from the function call as shown in the code segment in list 1. This prevents unwanted portions of the program code to get included into the timer expiration time. As soon as the timer expires, an interrupt is generated and the corresponding interrupt service routine (see list 2) stops the timer permanently by 'ta0s = 0' which also prevents further target function calls to take place by making 'repeat = 0'.

```
while( repeat ){
       ta0s = 1;     /* start TimerA0 */
       /* Calling target function (in assembly) */
       result = asm_add( addend, augend );
       ta0s = 0;     /* stop TimerA0 */

       /* determine number of target function calls */
       /* call incremental addition function */
       num_func_call = add_C( num_func_call, increment );
}
```

List 1 Program loop to determine Number of function calls before timer expires

```
void TimerA0_ISR( void ){
       repeat = 0;          /* exit from while(repeat) loop */
       ta0s = 0;                   /* stop TimerA0 */
}
```

List 2 Timer interrupt service routine

5.0 Limitations – Interrupt Latency

The main error in determining computation time for function call in the approach described here is the interrupt latency time gets included with the timer expiration time. For example, let us assume that the theoretical timer expiration time is T (=3276.75 us in our case), the interrupt latency is L, number of function calls during T is N, and the number of function calls during L is 'n' then what we can actually determine for a function call by above approach is: (T+L)/(N+n). However, since in most practical cases, T>>L and N>>n, therefore, the calculated target function time is close to the actual T/N.

6.0 Reference

Renesas Technology Corporation Semiconductor Home Page

http://www.renesas.com

E-mail Support

support_apl@renesas.com

Data Sheets

- M16C/26 datasheets, M30262eds_0.30.pdf

User's Manual

- M16C/20/60 C Language Programming Manual, 6020c.pdf
- M16C/20/60 Software Manual, 6020software.pdf
- Optimizing C Program by Assembly Function Call in M16C/26 Application Note
- Writing interrupt handlers in C for the M16C Application Note
- MSV30262-SKP Quick start guide, Quick_Start_Guide_MSB30262.pdf
- MSV30262-SKP Users Manual, Users_Manual_MSV30262.pdf
- MDECE30262 Schematics, Schematics_MDECE30262_RevA.pdf

7.0 Implementation Software

7.1 Application Code Outline

The example programs, written in C and assembly, and compiled using KNC30 compiler, will run on the MSV30262 SKP board. To run the program, perform the following steps:

1. Load target program executable in the MCU Flash memory using KD30 (or FoUSB).

2. Reset the loaded program by pressing RESET and then run the program by pressing GO on the KD30 GUI. In case of FoUSB, disconnect USB monitor board connector from the UART1 / DEBUG port of the SKP board and supply 5V to Vcc. Press Reset button on the SKP board. The following message appears on the LCD display:

Press S2 **to start**

3. Press S2 and as soon as the timer expires and the computation time for the function call has been determined, the following message appears on the LCD display. The first line indicates “time per computation” for a single function call. The “Af” in the second line indicates that it was an Assembly function call and “1.65u” is short for 1.65 microseconds.

Time/cmp
Af 1.65u

4. To return to the initial menu, start over from step 2 mentioned above.

7.2 Software Source Code

7.2.1 Source Code in C

```
/***********************************************************************
*
*       File Name: main.c
*
*       Content:   This program demonstrates the technique of measuring the
*                  computation time of a target function call. The target
*                  function call used in this program is an assembly function
*                  call. This target function is called (invoked)
*                  repeatedly until a preset timer expires. An incremental
*                  addition function in C is used to determine the number
*                  (=N) of function calls made during the duration (=T) of
*                  the timer expiration. The duration of the target function
*                  call is therefore given by: T/N.
*                  To run the program, load the program executable in the MCU
*                  on the MSV30262-SKP board. Then press switch S2. The
*                  duration of the target function call is displayed on the
*                  LCD display.
*
*       Date:  2-24-2003
*       This program was written to run on the MDECE30262 Board for MSV30262-SKP.
*
*       Copyright 2003 Renesas Technology America, Inc.
*       All rights reserved
*
*=======================================================================
*       $Log:$
*=======================================================================*/
```

```
#include "sfr262.h"
#include "skp26.h"
#include <stdio.h>

/* declare assembly function to be called from C */
extern unsigned int asm_add(unsigned int, unsigned int);

/* declare pragma to assign the variable transferring registers */
#pragma PARAMETER asm_add( R0, R1)

/* declare timerA0 interrupt service routine */
#pragma INTERRUPT /B TimerA0_ISR

#define TIMER_CONFIG 0x00          /* use timer mode */
#define CNTR_IPL  0x03             /* decide interrupt priority level */
#define TIMER_EXPIRATION 3276.750  /* FFFF/20MHz= FFFF X 50 ns = 3276.750 us */

/* define switch on skp board */
#define sw2          p10_5

#define UNLOCK_CM_REG       prc0 = 1; /* unlock clock mode registers cm0 and cm1 */
#define LOCK_CM_REG         prc0 = 0; /* lock clock mode registers cm0 and cm1 */

void display_compTime( unsigned int );   /* routine that displays computation time
*/
void init_timer( void );    /* routine that initializes timer */
void TimerA0_ISR( void );   /* Interrupt Service Routine for timer A0 */
void init_clock(void);      /* routine for initiaizing clock to divide/1 mode */
unsigned int add_C( unsigned int, unsigned int ); /* incremental addition function
(in C) */

int  repeat; /* variable used as a flag for looping */
char buf[9]; /* buffer required for displaying on 8 character line LCD */

/* declare variables used in increment addition function call (in C) */
unsigned int increment = 1;
unsigned int num_func_call; /* will hold incremented value */

/* declare variables used in target assembly function call */
unsigned int addend = 4;
unsigned int augend = 5;
unsigned int result;

/* declare variables for while loops */
int check_sw = 1;  /* used for checking switch press */
int loop_once = 1; /* used for displaying initial message on LCD */

/*****************************************************************************
Name:   main
Parameters: None
Returns: None
Description:  main program loop and initialization
*****************************************************************************/
```

```
main(){

       init_clock(); /* initialize CPU clock to run at divide / 1 mode = 20 MHz*/
       init_timer();        /* initialize timer */
       init_disp();  /* initialize and clear display. uses skp26.c file */

       /* display initial menu on LCD */
       while(loop_once == 1){
              sprintf( buf, "Press S2" );
              display(0,buf);
              sprintf(buf, "to start");
              display(1, buf);
              loop_once = 0;       /* exit to next while loop */
       }


       while (check_sw){           /* check if switch S2 is pressed */
              if( sw2 == 0 ){             /* switch S2 is pressed */
                     check_sw = 0;        /* enter while loop only once */
                     num_func_call = 0;
                     repeat = 1;
                     /* call target function repeatedly until timer expires */
                     while( repeat ){
                            ta0s = 1;     /* start TimerA0 */
                            /* Calling target function (in assembly) */
                            result = asm_add( addend, augend );
                            ta0s = 0;     /* stop TimerA0 */

                            /* determine number of target function calls */
                            num_func_call = add_C( num_func_call, increment );
                     }

                     display_compTime(num_func_call );  /* display computation time
*/
              }
       }
}
/*************************************************************************
Name:   init_timer
Parameters: None
Returns: None
Description: Initializes Timer A to timer mode
*************************************************************************/
void init_timer( void ){

       ta0 = 0xffff;               /* fill ta0 register with maximum value */
       asm(" fclr i ");
       ta0ic = CNTR_IPL;           /* assign interrupt priority level */
       ta0mr = TIMER_CONFIG;       /* configure timera0 as timer mode */
       asm( "fset i");

       return;
}
```

```
/***********************************************************************
Name:  TimerA0_ISR
Parameters: None
Returns: None
Description:  Interrupt Service Routine for Timer A0. stops repeated target
              function call by exiting from relevant while loop.
***********************************************************************/
void TimerA0_ISR( void ){

       repeat = 0;                 /* exit from while(repeat) loop */
       ta0s = 0;                         /* stop TimerA0 */
}

/***********************************************************************
Name:   display_compTime
Parameters: unsigned int
Returns: None
Description: Displays on LCD computation time of target function call.
***********************************************************************/
void display_compTime( unsigned int nfc ){

       float comp_time;
       comp_time = TIMER_EXPIRATION / nfc;
       /* initialize and clear display. uses skp26.c file */
       init_disp();

       /* display computation time */
       sprintf( buf, "Time/cmp");
       display( 0, buf);
       sprintf( buf, "Af %4.2fu", comp_time);
       display( 1, buf);

       return;
}

/***********************************************************************
Name: init_clock
Parameters: None
Returns: None
Description: Initializes CPU clock to divide/1 mode such that Xin=20MHz
***********************************************************************/
void init_clock( void ){
       UNLOCK_CM_REG        /* un-lock cm0 and cm1 */
       /* assign BCLK = Xin */
       cm07  = 0;                  /* select system clock Xin/Xout */
       cm06  = 0;           /* enable cm16 and cm17  */
       cm16  = 0;                  /* select no division clock mode */
       cm17  = 0;                  /* select no division clock mode */
       LOCK_CM_REG          /* lock the system clock control register */
       return;
}
```

```
/***********************************************************************
Name:  add_C
Parameters: unsigned int, unsigned int
Returns: unsigned int
Description: Addition function in C used as incremental addition function.
             Invoked along with calling the target function. Used to
             determine the number of function calls possible during the
             timer expiration.
***********************************************************************/
unsigned int add_C(unsigned int addend, unsigned int augend){
       unsigned int k;
       k = addend + augend;
       return k;
}
```

7.2.2 Source code of Assembly function

```
;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
;Name: asm_add
;Parameters: addend, augend (passed through registers R0 and R1)
;Returns: int (passed through register R0)
;Description: Assembly addition function called from C program
          ;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
.SECTION      program_a
.GLB          _asm_add      ;declaring global assembly function

_asm_add:
       ADD.W  R1,R0  ;math operation (addition) on transferred variables
       RTS           ;R0 will be returned to the calling function
       .END
```

Keep safety first in your circuit designs!

- Renesas Technology Corporation puts the maximum effort into making semiconductor products better and more reliable, but there is always the possibility that trouble may occur with them. Trouble with semiconductors may lead to personal injury, fire or property damage.
  Remember to give due consideration to safety when making your circuit designs, with appropriate measures such as (i) placement of substitutive, auxiliary circuits, (ii) use of nonflammable material or (iii) prevention against any malfunction or mishap.

Notes regarding these materials

- These materials are intended as a reference to assist our customers in the selection of the Renesas Technology Corporation product best suited to the customer's application; they do not convey any license under any intellectual property rights, or any other rights, belonging to Renesas Technology Corporation or a third party.
- Renesas Technology Corporation assumes no responsibility for any damage, or infringement of any third-party's rights, originating in the use of any product data, diagrams, charts, programs, algorithms, or circuit application examples contained in these materials.
- All information contained in these materials, including product data, diagrams, charts, programs and algorithms represents information on products at the time of publication of these materials, and are subject to change by Renesas Technology Corporation without notice due to product improvements or other reasons. It is therefore recommended that customers contact Renesas Technology Corporation or an authorized Renesas Technology Corporation product distributor for the latest product information before purchasing a product listed herein.
  The information described here may contain technical inaccuracies or typographical errors.
  Renesas Technology Corporation assumes no responsibility for any damage, liability, or other loss rising from these inaccuracies or errors.
  Please also pay attention to information published by Renesas Technology Corporation by various means, including the Renesas Technology Corporation Semiconductor home page (http://www.renesas.com).
- When using any or all of the information contained in these materials, including product data, diagrams, charts, programs, and algorithms, please be sure to evaluate all information as a total system before making a final decision on the applicability of the information and products. Renesas Technology Corporation assumes no responsibility for any damage, liability or other loss resulting from the information contained herein.
- Renesas Technology Corporation semiconductors are not designed or manufactured for use in a device or system that is used under circumstances in which human life is potentially at stake. Please contact Renesas Technology Corporation or an authorized Renesas Technology Corporation product distributor when considering the use of a product contained herein for any specific purposes, such as apparatus or systems for transportation, vehicular, medical, aerospace, nuclear, or undersea repeater use.
- The prior written approval of Renesas Technology Corporation is necessary to reprint or reproduce in whole or in part these materials.
- If these products or technologies are subject to the Japanese export control restrictions, they must be exported under a license from the Japanese government and cannot be imported into a country other than the approved destination.
  Any diversion or reexport contrary to the export control laws and regulations of Japan and/or the country of destination is prohibited.
- Please contact Renesas Technology Corporation for further details on these materials or the products contained therein.