M16C/26

Writing Interrupt Handlers in C

1.0 Abstract

This application note describes how to write hardware and software interrupt handlers using NC30 C (version 4.XX and earlier) compiler for M16C microcontrollers. Interrupt mechanisms are processor (hardware) specific, the ANSI C specification for writing interrupt functions is, at best, vague.

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. In order to process interrupts, your program must do two things:

1. Properly declare the function handling the interrupt.
2. Set the appropriate interrupt vector to point to the function.

3.0 Hardware Interrupts

By declaring the function as an interrupt handler, the compiled output ends in a "reit" (return from interrupt) instruction rather than the standard "rts" (return from subroutine). For hardware interrupts, use the declaration,

```
#pragma INTERRUPT function_name /B
```

Note: "INTERRUPT" must be uppercase.

and the prototype,

```
void  function_name  (void);
```

Obviously, a hardware interrupt can neither be passed a value nor return a value.

Section 5.0 gives an example hardware interrupt processing program. Note the "/B" option in the #pragma statement. This option speeds up interrupt response by switching to CPU register Bank 1. Without this switch, the compiler generates code to stack all registers used by the interrupt functions. Be careful when using this option if using an RTOS or nested interrupts.

4.0 Reference

Renesas Technology Corporation Semiconductor Home Page

http://www.renesas.com

E-mail Support

support_apl@renesas.com

Data Sheets

- M16C/26 datasheets, M30262eds.pdf

User's Manual

- M16C/20/60 C Language Programming Manual, 6020c.pdf
- M16C/20/60 Software Manual, 6020software.pdf
- MSV30262-SKP Quick start guide, Quick_Start_Guide_MSB30262.pdf
- MSV30262-SKP Users Manual, Users_Manual_MSV30262.pdf

5.0 Software Code

The example program was written to run on the MSV30262 Starter Kit but could be modified to implement in a user application. The program was compiled using the KNC30 compiler. The program demonstrates using the interrupt to invoke relevant service routine. Interrupt signals are generated by pressing any of the user pushbutton switches S2, S3 or S4 on the SKP board. The program accepts incoming interrupt signals in 3 of the 4 Key Input Interrupt pins. Relevant interrupt service routine is invoked to toggle the red LED (D3) of the SKP board. If the red LED was ON, then pressing any of the 3 switches will turn it OFF and vise-versa.

```
/******************************************************************************
*
*       File Name: main.c
*
*       Content:   This program toggles the red LED (D3) with successive pressing
*                  of any of the three switches -2, -3 or -4. With each pressing
*                  of a switch, a key input interrupt is generated and toggles
*                  the red LED from ON to OFF or vise-versa.

*       Date:  11-13-2002
*       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"         // M16C/26 special function register definitions
#pragma INTERRUPT  /B       KeyInput_ISR

/* LEDs */
#define       red_led       p7_0
#define       yellow_led    p7_1
#define       green_led     p7_2


void KeyInput_ISR(void);    /* declare interrupt service routine */
void mcu_init(void);        /* declare mcu initialization routine */

/*****************************************************************************
Name:   main
Parameters: None
Returns: None
Description:  main program loop

*****************************************************************************/

main() {
       mcu_init();                 /* initialize MCU */

       kupic = 2;                  /* set key input interrupt level = 2 */

       _asm( "fset i");            /* enable interrupts */

       while(1);                   /* wait for interrupt */
}

/*****************************************************************************
Name:         KeyInput_ISR
Parameters:   None
Returns:      None
Description: This is the service routine for Key Input Interrupt. This routine
             will be called whenever any of the switches S2, S3, or S4 is pressed.
*****************************************************************************/

void KeyInput_ISR(void){                  /* Toggle red LED */

       _asm( "fset i");                   /* enable interrupt for the ROM monitor */

       if( red_led == 1)                  /* check if red LED is OFF */
              red_led = 0;                /* Turn red LED ON */
       else                               /* Otherwise red LED is ON */
              red_led = 1;                /* Turn red LED OFF */
}
```

```
/*****************************************************************************
Name:          mcu_init
Parameters:    None
Returns:       None
Description:   Routine for initializing the LEDs of the SKP board (MSV30262)
               And the port pins of Key Input Interrupt
*****************************************************************************/
void mcu_init(void) {

   /* LED initialization */
       pd7_0 = 1;           /* set LED ports to outputs (connected to LEDs) */
       pd7_1 = 1;
       pd7_2 = 1;

       red_led = 1;         /* turn off LEDs */
       green_led = 1;
       yellow_led = 1;

   /* Configure port pins for Key Input Interrupt */

       pd10_5 = 0;    /* set Key Input port pins of switches -2, -3 and -4 to inputs
*/
       pd10_6 = 0;
       pd10_7 = 0;

       pd10_4 = 1;/* set the remaining (unused) Key Input port pin to output for avoiding*/
              /* undesirable (low) input signal that may cause unwanted interference
*/

       return;
}
```

Now the second part of the process is to set up the interrupt vector. Included with the compiler is the startup file 'sect30.inc'. Open this file and near the end, the interrupt vectors are declared. The assembler function label is the C function name preceded by an underscore "_". First define the label as global using the '.glb' pseudo-instruction, then replace the 'dummy_int' label at the appropriate table entry (see sample code below). For our example program given above, we need to edit the Key Input Interrupt vector commented as "key-on wakeup (for user)"

```
;----------------------------------------------------------------
; variable vector section
;----------------------------------------------------------------
          :
          :
       .lword dummy_int            ; DMA0 (for user)
       .lword dummy_int            ; DMA1 (for user)
       .glb    _KeyInput_ISR
       .lword _KeyInput_ISR        ; Key-on wakeup (for user)
       .lword dummy_int            ; AD Convert (for user)
          :
          :
```

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.