

Overview

embARC is an open software platform designed to help accelerate the development and production of embedded systems based on DesignWare® ARC® processors.

This article provides a step by step guide to create your own simple timer and ISR example in embARC to becoming more familiar with ARC timer and interrupt handling in software.

An interrupt is a mechanism to respond to interrupt signals emitted by hardware or software in embedded systems. An Interrupt Service Routine (ISR) is the function to deal with the immediate event generated by a given interrupt. A timer is one of many peripherals that may provide interrupt signals.

More information on ARC Interrupts can be found from embARC Documentation under `\doc\embARC_Document.html`

embARC 2015.05

Main Page | Related Pages | Modules

▼ embARC
▶ embARC Introduction
▼ ARC
 Description
 ▶ Startup
 ▶ **Exception and Interrupt**
 Built-in Functions
 Resource Definitions
 ▶ Access to Resources
 Related files
▶ Device
▶ Board
▶ OS
▶ Middleware
▶ Example
▶ ChangeLog
 License
 ▶ Modules

Exception and Interrupt

ARC Exception and Interrupt

The ARCv2 processor is designed to allow exceptions to be taken and handled from user mode or kernel mode, and from interrupt service routines. All entry to an exception handler.

The exceptions can be divided into two parts: CPU exception and interrupt exception. CPU exceptions are triggered by errors such as wrong instruction exceptions are triggered by device interrupts and are always asynchronous.

Exception vectors are fetched in the instruction space (ICCM, main memory, but not DCCM). Every exception has the following information:

- **Vector Name** is a symbolic equivalent to the vector number.
- **Vector Number** is a unique 8-bit index into the table of exception or interrupt vectors.
- **Vector Offset** is a offset value calculated as four times the vector numbers (vector offset = 4*vector_number), which is used to determine the position of the exception handler.
- **Cause Code** is a 8-bit number to identify the exact cause of an exception.
- **Parameter** is a 8-bitfield to pass a single parameter from exception to the exception handler and identify exceptions with the same cause code.

For more details, please refer to the **ARCv2 ISA Programmer's Reference Manual (PRM)**.

Exception and Interrupt Processing in embARC

A basic exception and interrupt processing framework is implemented in embARC. Through this framework, users can handle specific exception and/or underlying details of saving and restoring registers or operating on aux registers.

For CPU exceptions and interrupts, **entry** is called first, then **handler** is called in entry. Users can define their own entry using **exc_entry_install**. A standard interrupt processing model is shown in the picture below.

NOTE: This article assumes the reader is already familiar with embARC. If this is your first project with embARC, please start by first reading our “Quick start” article to ensure your development environment is properly setup before you begin.

Development Environment

Development host operating system:

- **Windows 7**

Development Toolchain for Target Platform:

- **GNU Toolchain for DWC ARC Processors, version 2015.06**

Target platform:

- **ARC EM Starter Kit (EMSK), version 1.1.**

Getting Started

Creating the project “isr_timer” in embARC

- 1) Create the folder `\embARC\example\emsk\isr_timer`.
- 2) Add two files, makefile and main.c into the folder `\embARC\example\emsk\isr_timer`.
Modify the makefile to reflect appropriate configuration choices for BOARD, BD_VER and CUR_CORE as follows:

```
# Application name
APPL ?= isr_timer

# Optimization Level
# Please Refer to toolchain_XXX.mk for this option
OLEVEL ?= O2

##
# Current Board And Core
##
BOARD ?= emsk
BD_VER ?= 11
CUR_CORE ?= arcem6

##
# select debugging jtag
##
ITAG ?= usb
```

Edit main.c file in `\embARC\example\emsk\isr_timer` to add source code shown in the capture below.

```

/* embARC HAL */
#include "embARC.h"
#include "embARC_debug.h"

/** main entry for testing arc timer functions */
int main(void)
{
    uint32_t val;
    cpu_lock();

    board_init(); /* board init */

    cpu_unlock();

    EMBARC_PRINTF("Hello embARC! This is a example about how to use timer and isr.\n");

    while(1);
    return E_SYS;
}

```

- 3) Add ISR callback functions “*timer0_isr*” and “*timer1_isr*” in main.c. These ISR functions are called after timer0 or timer1 hardware raises an interrupt. In this example, the ISR is simply clearing the pending interrupt flag, increasing an interrupt count counter and printing a message to the console.

```

volatile static int t0 = 0;
volatile static int t1 = 0;

/** arc timer 0 interrupt routine */
static void timer0_isr(void *ptr)
{
    ptr = ptr;
    /* clear the interrupt pending flag */
    timer_int_clear(TIMER_0);
    t0++;
    EMBARC_PRINTF("timer0 interrupt: %d * 1s\n", t0);
}

/** arc timer 1 interrupt routine */
static void timer1_isr(void *ptr)
{
    ptr = ptr;
    /* clear the interrupt pending flag */
    timer_int_clear(TIMER_1);
    t1++;
    EMBARC_PRINTF("timer1 interrupt: %d * 3s\n", t1);
}

```

- 4) Initialize the timer0 and timer1 in the EMSK in the main.c file. “*BOARD_CPU_CLOCK*” is the timer frequency in the EMSK. The function “*int_handler_install*” is used to register ISR functions. The following configuration of “*timer_start*” means the timer0 interrupt will be triggered every second, and the timer1 interrupt will be triggered every 3 seconds.

```

/** main entry for testing arc timer functions */
int main(void)
{
    uint32_t val;
    cpu_lock();

    board_init(); /* board init */

    cpu_unlock();

    EMBARC_PRINTF("Hello emBARC! This is a example about how to use timer and isr\n");

    /* check whether timer 0 is present*/
    if (timer_present(TIMER_0)) {
        EMBARC_PRINTF("timer 0 is present\n");
        /* read the COUNT register */
        timer_current(TIMER_0, &val);
        EMBARC_PRINTF("cnt:%d\n", val);
        /* register the callback function */
        int_handler_install(INTNO_TIMER0, timer0_isr);
        /* enable the interrupt of timer 0 */
        int_enable(INTNO_TIMER0);
        /* set the timer 0 operation mode and Limit register*/
        timer_start(TIMER_0, TIMER_CTRL_IE, BOARD_CPU_CLOCK);
    }

    /* check whether timer 1 is present*/
    if (timer_present(TIMER_1)) {
        EMBARC_PRINTF("timer 1 is present\n");
        /* read the COUNT register */
        timer_current(TIMER_1, &val);
        EMBARC_PRINTF("cnt:%d\n", val);
        /* register the callback function */
        int_handler_install(INTNO_TIMER1, timer1_isr);
        /* enable the interrupt of timer 1 */
        int_enable(INTNO_TIMER1);
        /* set the timer 1 operation mode and Limit register*/
        timer_start(TIMER_1, TIMER_CTRL_IE, 3*BOARD_CPU_CLOCK);
    }

    while(1);
    return E_SYS;
}

```

- 5) From the command line, go to `\embARC\example\emsk\isr_timer` . Compile for EMSK 1.1 and ARC EM6 processor to generate `isr_timer_gnu_arcem6.elf`:

`make TOOLCHAIN=gnu BD_VER=11 CUR_CORE=arcem6`

```

Compiling      : " .././../arc/arc_cache.c
Compiling      : " .././../arc/arc_exception.c
Assembling     : " .././../arc/arc_startup.s
Assembling     : " .././../arc/arc_exc_asm.s
Archiving      : " obj_emsk_11/gnu_arcem6/libarc.a
arc-elf32-ar: creating obj_emsk_11/gnu_arcem6/libarc.a
Archiving      : " obj_emsk_11/gnu_arcem6/libembarc.a
arc-elf32-ar: creating obj_emsk_11/gnu_arcem6/libembarc.a
Linking        : " obj_emsk_11/gnu_arcem6/isr_timer_gnu_arcem6.elf

```

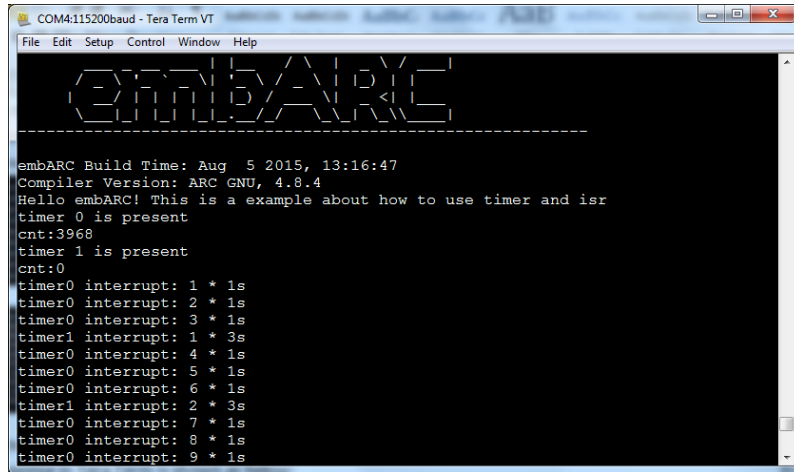
- 6) Enter **`make run TOOLCHAIN=gnu BD_VER=11 CUR_CORE=arcem6`** in command line to run the program on the target.

```

C:\Users\nnsong\workspace\embARC201505\embARC\example\emsk\isr_timer>make run TOOLCHAIN=gnu BD_VER=11 CUR_CORE=arcem6
"Download & Run obj_emsk_11/gnu_arcem6/isr_timer_gnu_arcem6.elf"
arc-elf32-gdb -ex "target remote | openocd --pipe -s C:/arc_gnu/share/openocd/scripts -f C:/arc_gnu/share/openocd/scripts/board/snps_em_sk_v1.cfg" -ex "load" -ex "c" obj_emsk_11/gnu_arcem6/isr_timer_gnu_arcem6.elf
GNU gdb (ARCCompact/ARCV2 ISA elf32 toolchain 2015.06) 7.9.1
Copyright (C) 2015 Free Software Foundation, Inc.
License GPLv3+: GNU GPL version 3 or later <http://gnu.org/licenses/gpl.html>
This is free software; you are free to change and redistribute it.
There is NO WARRANTY, to the extent permitted by law. Type "show copying"
and "show warranty" for details.
This GDB was configured as "--host=i686-w64-mingw32 --target=arc-elf32".
Type "show configuration" for configuration details.

```

The interrupt response will be shown on the terminal.



```
COM4:115200baud - Tera Term VT
File Edit Setup Control Window Help
embARC
-----
embARC Build Time: Aug 5 2015, 13:16:47
Compiler Version: ARC GNU, 4.8.4
Hello embARC! This is a example about how to use timer and isr
timer 0 is present
cnt:3968
timer 1 is present
cnt:0
timer0 interrupt: 1 * 1s
timer0 interrupt: 2 * 1s
timer0 interrupt: 3 * 1s
timer1 interrupt: 1 * 3s
timer0 interrupt: 4 * 1s
timer0 interrupt: 5 * 1s
timer0 interrupt: 6 * 1s
timer1 interrupt: 2 * 3s
timer0 interrupt: 7 * 1s
timer0 interrupt: 8 * 1s
timer0 interrupt: 9 * 1s
```

NOTE: Existing application `\embARC\example\emsk\timer` is very similar and available as part of the embARC software from <https://forums.embarc.org/categories/downloads>.

For any additional support on embARC, please post a question on embARC Forums at <https://forums.embarc.org/>