AN183

Application Note

WRITING INTERRUPTS USING THE ANGEL DEBUGGER

Note: Cirrus Logic assumes no responsibility for the attached information which is provided “AS IS” without warranty of any kind (expressed or implied).

P.O. Box 17847, Austin, Texas 78760
(512) 445 7222 FAX: (512) 445 7581
http://www.cirrus.com

1. INTRODUCTION

This application note explains how to install a interrupt service routine (ISR) in the Angel debugging environment. This application note assumes you are using a Cirrus Logic 72xx development board that is running Angel, Version 1.20, and the ARM® Tools, Version 2.5.

The ARM project file *(*LED_ISR.APJ) and two source files *(*ISRSHELL.S and LED_ISR.C) are the example files for this application note. Treat the source code in the example files as an important reference source for understanding the concepts presented in this application note.

2. BACKGROUND

There are two traditional methods for installing an interrupt service routine:

- Application software installs the ISR
- Use Angel Debugger to Install ISR

2.1 Method One: Application Software Implementation of ISR

The first method assumes that the interrupt vectors reside in RAM starting at location `0x0`. In this case, the application software saves off the existing interrupt vector and replaces it with a branch to the new ISR. The new ISR then exits by chaining to the previous ISR. This approach is explained in detail in the Chapter 9 of the *ARM Software Development Toolkit User Guide*.

This method is useful when Angel is compiled to run entirely in upper memory and RAM exists at the bottom of memory where the exception vectors reside (including IRQ). This method can also be used if you are using a JTAG-based debugger (such as Multi-ICE™, JEENI™, or Wiggler) and have programmed the Memory Management Unit (MMU) to map DRAM to the first 16M bytes of memory (see the application note, AN177, *Using JTAG for Debugging EP72xx Microcontrollers*).

2.2 Method Two: Use Angel Functionality

The Angel debug monitor resides in ROM (FLASH) memory starting at 0x0. Angel is used to call a function to install, save, and restore the existing ISR vectors.

These functions are typical of a BIOS or an operating system, and are also found in target debuggers such as Angel. The Angel debugger contains two special SWI (software interrupt) calls that allow for the installation of an ISR. This is done using the Angel semi-hosting SWI (by default this is `0x123456` in ARM state).

3. SOFTWARE

The example program installs an interrupt that flashes the heartbeat LED once a second. It also uses the printf semi-hosting functions of Angel to print the status to the console screen. The C code defines two special Angel SWI's:

- Angel_GetIRQ, which gets the existing interrupt vector
- Angel_SetIRQ, which installs the location for the new ISR

The procedure for installing the ISR follows these steps:

1) Get the old vector using Angel_GetIREQ, and stash it away in a global location.
2) Install the ISR routine vector by calling Angel_SetIRQ.
3) Program the peripherals and the interrupt mask registers.

4. SAMPLE EXERCISE

Change the interrupt rate from 1 second to 10ms. (Hint: See source code comments.) Then change the program so that the LED blinks at 1 second intervals. (Hint: you will need to modify the interrupt routine to blink only on every 100th occurrence.)

• Notes •


Maverick™
from
CIRRUS LOGIC