Lab 6 – Musical Keyboard
Interrupts Using RTAI

ECE 4220/7220
Real Time Embedded Computing
Spring 2016
University of Missouri
Interrupt Handler

• Piece of code that runs when an interrupt is triggered.

• Some interrupts can be programmed to go off in four different ways
  – Edge sensitive: Rising/Falling
  – Level sensitive: High/Low

• The TS-7250 has 64 interrupt request lines
  – One line can have multiple handlers
RTAI and the Auxiliary board

• All of the buttons can be programmed to cause an interrupt.

• However, these signals all cause the same interrupt request line to trigger.

• This means that the handler must be able to determine which button triggered the interrupt.
Installing an Interrupt Handler

• We will use a kernel module to install interrupt handlers.

• Installing a handler requires two components:
  – Software: providing code to run when interrupt line is triggered
  – Hardware: configuring I/O lines to be able to accept interrupts
Hardware Side

When configuring an I/O line as an interrupt the following must be configured:

The line must be set as an input

• The type of the interrupt must be set
  – edge/level rising/falling high/low

• The line can be configured for de-bounce

• The line **MUST** be designated as an interrupt (there is a register that you must modify)
Software Side

• To attach a handler to a interrupt line we will use the following command from the prelab:

```c
int rt_request_irq(unsigned irq_num, int (*handler)(unsigned irq_num, void *cookie), void *cookie, int retmode)
```

- irq_num is the request line number to attach the handler to
- handler is a pointer to the handler to install
- cookie is a pointer to a cookie structure (we will provide NULL)
- retmode is the return mode which determines how the interrupt is handled (we use 1)

• To enable/disable interrupt handling we will use:

```c
rt_enable_irq(unsigned irq)
rt_disable_irq(unsigned irq)
```
Handler Responsibilities

• There are two things a handler must do when handling an interrupt request:

  – Disable the request line so no new interrupts can trigger

  – Clear the End Of Interrupt register

• The handler will also execute the desired code.
Example

static void my_handler(unsigned irq_num, void *cookie) {
    // disable interrupt handling
    // do stuff
    // clear interrupt
    // re-enable interrupt handling
}

rt_request_irq(irq_num, my_handler, 0, 1);  // actual code

// clear interrupt

// enable interrupt (register level)

// enable interrupt handling