ICU_I386

The ICU_I386 module contains the machine-dependent interrupt control unit code for Intel-Architecture PC-based systems.

Target File Definitions

The values required in the target file depend on the model of CPU on the board.

ICU_IRQ*n*	(*n* = 0..15) The interrupt vector numbers returned by the first-level interrupt handlers for the CPU interrupts IRQ*n* .

Shared Library Macros and Routines

icu_clear_interrupt

void *icu_clear_interrupt*(
    int *ino*)

The *icu_clear_interrupt* routine clears the interrupt-pending state for the interrupt numbered *ino.*

icu_disable_interrupt

void *icu_disable_interrupt*(
    int *ino*)

The *icu_disable_interrupt* routine masks out the interrupt numbered *ino* so that it will not generate a machine interrupt.

icu_enable_interrupt

void *icu_enable_interrupt*(
    int *ino*)

The *icu_enable_interrupt* routine allows the device(s) connected to interrupt number *ino* to generate a machine interrupt.

icu_setup_default_handlers

void *icu_setup_default_handlers*(**void**)

The *icu_setup_default_handlers* routine is called during system initialisation to populate the interrupt handler table with the default ’unhandled-interrupt’ handler, before any driver-specific handlers are installed.

rome_add_handler

(void) *rome_add_handler*(
  int *where*,
  void (*rtn***)(int)**)

The *rome_add_handler* routine adds the routine *rtn* as a handler for the interrupt specified by *where*. The value of *where* should be one of the 16 *ICU_IRQn* values corresponding to the requested IRQ interrupt.

rome_end_critical

void *rome_end_critical*(
  uint *old*)

The *rome_end_critical* macro returns the criticality level to the state specified by the *old* parameter, which should be passed (unchanged) from the corresponding call to *rome_start_critical*. This macro generates inlined assembler code for maximum execution speed.

rome_start_critical

uint *rome_start_critical*(**void**)

The *rome_start_critical* macro enters a new critical section, disabling all external interrupts. It returns an opaque token representing the previous criticality level, to be passed to *rome_end_critical* to restore the state. This macro generates inlined assembler code for maximum execution speed.