ICU_I960

The ICU_I960 module contains the machine-dependent interrupt control unit code for Intel960-based systems.

Target File Definitions

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

CPU_xx	The CPU model, 'xx' is currently one of CX, HX or JX.
CPU_IMASK_ADDR	(Hx and Jx series) The address of the memory-mapped interrupt-mask register.
CPU_IPND_ADDR	(Hx and Jx series) The address of the memory-mapped interrupt-pending register.

Shared Library Macros and Routines

icu_check_ipend

int *icu_check_ipend*(
uint *bitpos*)

The *icu_check_ipend* routine returns TRUE if the bit at position *bitpos* is set in the hardware interrupt-pending register, and FALSE otherwise.

icu_clear_ipend

void *icu_clear_ipend*(
uint *bitpos*)

The *icu_clear_ipend* routine unsets the bit at position *bitpos* in the hardware interrupt-pending register. The update is performed as an atomic operation so no other bit settings in the register are affected.

icu_disable_interrupt

void *icu_disable_interrupt*(
int *ino*)

The *icu_disable_interrupt* routine masks out the interrupt numbered *ino* (0..31) 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* (0..31) 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 the 'vectored' form of the correspding interrupt (i.e. a value in the range 0..255).

rome_end_critical

void *rome_end_critical*(
uint *old*)

The *rome_end_critical* macroreturns 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.