
SpartanMC

Complex Interrupt Controller

(IRQ-Ctrlp)

Table of Contents

1. Function	1
2. Module parameters	2
3. Peripheral Registers	2
3.1. IRQ-Ctrl Register Description	2
3.2. IRQ-Ctrl Register Priority IRQ-Number	2
3.3. IRQ-Ctrl C-Header for Register Description	3

List of Figures

1 IRQ-Ctrl block diagram for IR_SOURCES=54 1

List of Tables

1 IRQ-Ctrl modul parameters	2
1 IRQ-Ctrl registers	2
1 Priority IRQ-Number register layout	2

Complex Interrupt Controller (IRQ-Ctrlp)

Depending on the requirements of the target application two types of interrupt controllers could be instantiated (IRQ-Ctrl and IRQ-Ctrlp). The complex interrupt controller (IRQ-Ctrlp) allows the interruption of a running Interrupt Service Routine (ISR) by an interrupt of higher priority. Therefore, the interrupt enable bit (function call of "interrupt_enable()") must be set within the ISR. The complex interrupt controller can handle a maximum of 8 nested interrupts. The 9'th nested interrupt invocation is executed after completion of the 8'th ISR. It should be noted, that IRQ-Ctrlp requires much more FPGA resources than the simple IRQ-Ctrl.

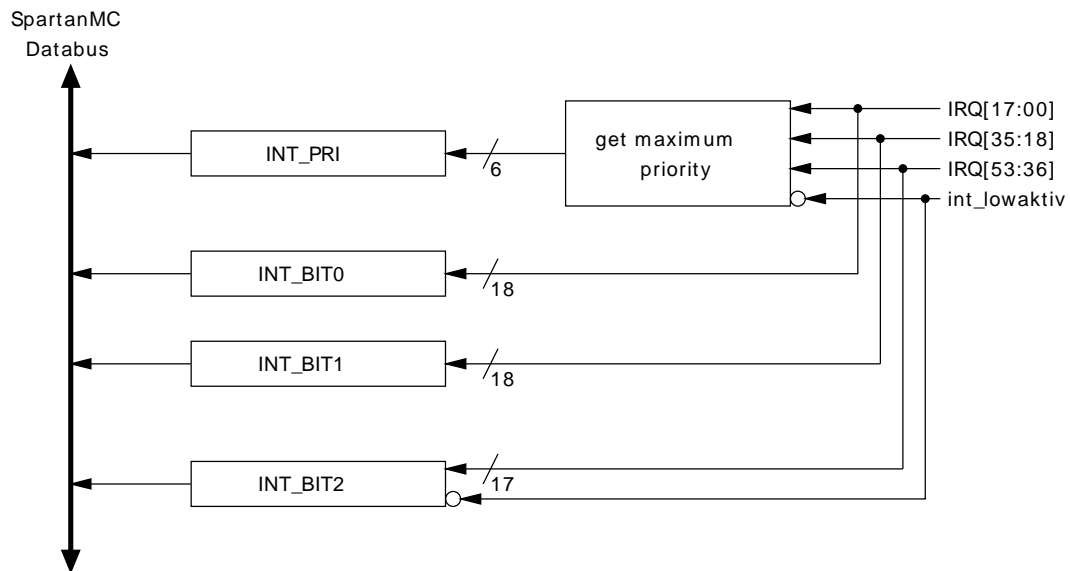


Figure 1: IRQ-Ctrl block diagram for IR_SOURCES=54

1. Function

The SpartanMC interrupt controller handles multiple interrupt sources as input sorted by their priority. Each interrupt capable peripheral must use a dedicated controller input signal for its interrupt request. If an interrupt occurs the controller sets the interrupt input signal of the pipeline. The interrupt number of the pending interrupt with the highest priority could be read from INT_PRI register. In order to check all interrupt sources, the controller provides a configurable number of 18 bit registers (INT_BIT0..2) each with 18 interrupt flags.

Pending interrupts are **not** stored within the interrupt controller. Thus, the logic to set/hold, reset and mask interrupts must be provided by the peripheral which is connected to the interrupt controller.

Note: The highest interrupt priority is reserved for the "int_lowactive" signal.

2. Module parameters

Table 1: IRQ-Ctrl modul parameters

Parameter	Default Value	Description
BASE_ADR	0x40	Start address of the memory mapped peripheral registers. The value is taken as offset to the start address of the peripheral memory space. This parameter is set by jConfig automatically.
IR_SOURCES	8	Number of IRQ Sources at SpartanMC Core.

3. Peripheral Registers

3.1. IRQ-Ctrl Register Description

The IRQ-Ctrl peripheral provides three or more 18 bit registers which are mapped to the SpartanMC address space located at $0x1A000 + \text{BASE_ADR} + \text{Offset}$.

Table 2: IRQ-Ctrl registers

Offset	Name	Access	Description
0	INT_PRI	read	Register for the current max. Priority IRQ-Number.
1	INT_BIT0	read	Contains the current IRQ-Sinals 0 to 17.
2	INT_BIT1	read	Contains the current IRQ-Sinals 18 to 35.
3	INT_BIT2	read	Contains the current IRQ-Sinals 36 to 54.

3.2. IRQ-Ctrl Register Priority IRQ-Number

Table 3: Priority IRQ-Number register layout

Bit	Name	Access	Reset	Description
0-7	INT_PRI	read	0	Contains the current max. priority.
8-11	INT_CNT	read	0	Contains the current Number of nested interrupts.
12-17	x	read	0	Not used.

3.3. IRQ-Ctrl C-Header for Register Description

```
#ifndef INTR_CTRL_H_
#define INTR_CTRL_H_

// Number of interrupts (i_bits)
// is set by jconfig.

typedef struct {
    volatile unsigned int int_pri;    // read
    volatile unsigned int int_bit0;  // read IRQ[17:00]
    volatile unsigned int int_bit1;  // read IRQ[35:18]
    volatile unsigned int int_bit2;  // read IRQ[53:36]
} intr_ctrl_t;

#endif /*INTR_CTRL_H_*/
```