Eccelerators Library IP specification

Table of Contents

List of Figures

List of Tables

1. EventCatcherIfc

1.1. Interrupt Collector Interface (InterruptCollectorIfc)

Interface containing a basic Interrupt Collector block.

Table 1.1. Blocks of Interrupt Collector Interface

Blocks of Interrupt Collector Interface		
Block Address	ID	Block Name
`0x00`	InterruptCollectorBlk	Interrupt Collector Block

Table 1.2. Resets of Interrupt Collector Interface

Resets of Registers of Interrupt Collector Interface	
ID	Reset Name
Async	BusReset: Asynchronous Bus Reset

1.1.1. Interrupt Collector Block (InterruptCollectorBlk)

This block defines a basic interrupt collector for **level triggered** interrupt sources. Usually edge triggered sources e.g., timer pulses can be converted to level triggered ones by catching them in the user logic.

Constraints:

1. Allow interrupt processing by multiple CPUs without need for spinlocks.

2. Enable forwarding an interrupt to the CPU(s) by a mask for each source.

3. Provide control pulses to notify the user logic when a interrupt service for an interrupt request has been started and has been ended.

4. Use the control pulses to reset the interrupt request fo a source or do it by a write or read access directly to the user logic e.g. reading the receive data register of an UART.

Interrupt Collector details:

Figure 1.1. Interrupt Collector details

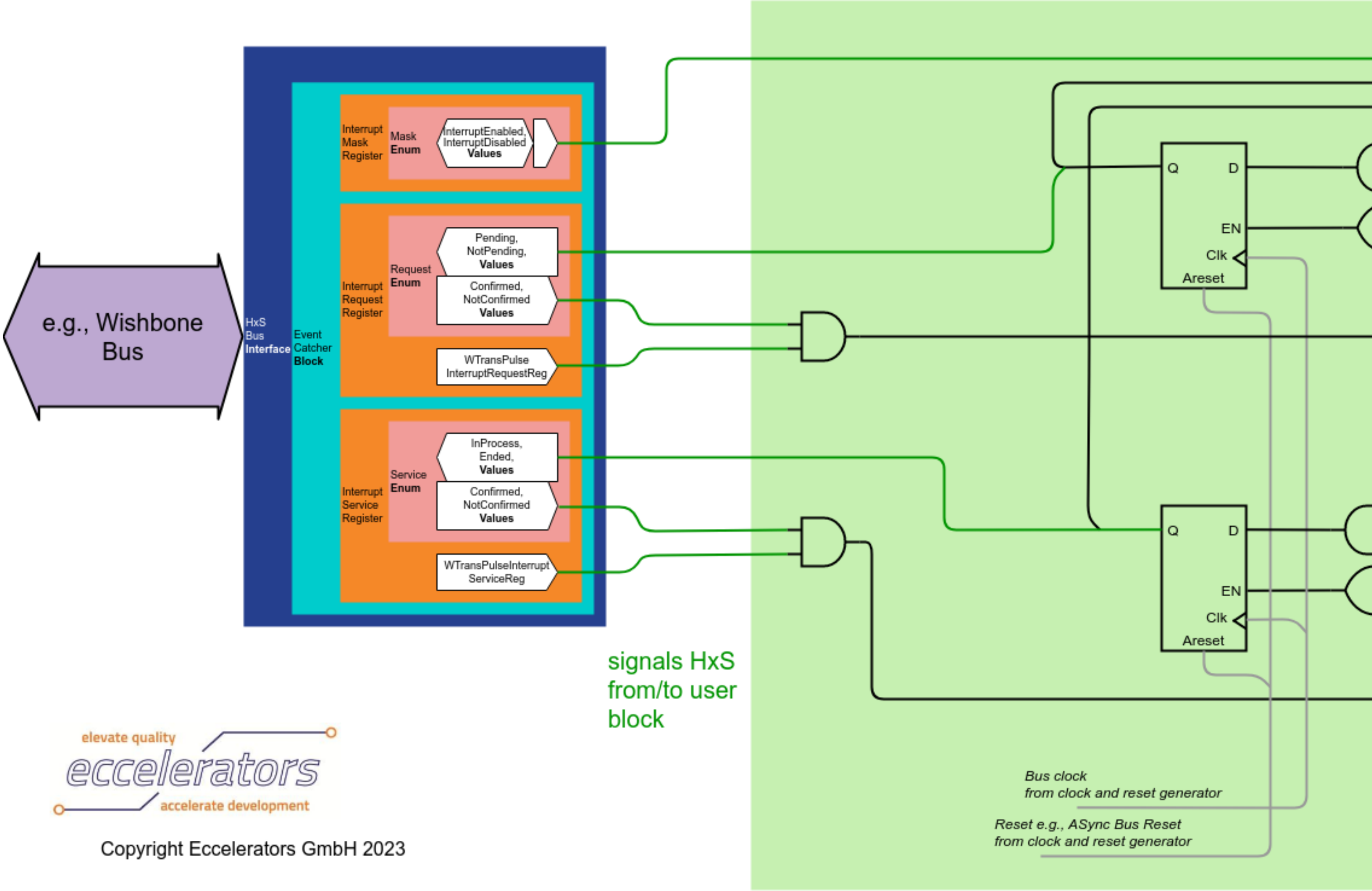
e.g., Wishbone Bus
HxS Bus Interface
Event Catcher Block
Interrupt Mask Register
Mask Enum
InterruptEnabled, InterruptDisabled Values
Interrupt Request Register
Request Enum
Pending, NotPending, Values
Confirmed, NotConfirmed Values
WTransPulse InterruptRequestReg
Interrupt Service Register
Service Enum
InProcess, Ended, Values
Confirmed, NotConfirmed Values
WTransPulseInterrupt ServiceReg
signals HxS from/to user block
Q
D
EN
Clk
Areset
Q
D
EN
Clk
Areset
Bus clock from clock and reset generator
Reset e.g., ASync Bus Reset from clock and reset generator
elevate quality
eccelerators
accelerate development
Copyright Eccelerators GmbH 2023

Table 1.3. Registers or Delegates of Interrupt Collector Block

Registers or Delegates of Interrupt Collector Block		

`0x00`
`..`
`0x0b`
		Interrupt Collector Block

Address	ID	Name
`0x00`	InterruptMaskReg	Interrupt Mask Register
`0x04`	InterruptRequestReg	Interrupt Request Register
`0x08`	InterruptServiceReg	Interrupt Service Register

1.1.1.1. Interrupt Mask Register (InterruptMaskReg)

Table 1.4. Bits of Interrupt Mask Register

<table>
<tr><td colspan="4">Bits of Interrupt Mask Register</td></tr>
<tr><td colspan="3"><code>0x00</code></td><td>Interrupt Mask Register (InterruptMaskReg)</td></tr>
<tr><td>Bits</td><td>ID</td><td>Type</td><td>Description</td></tr>
<tr><td>03</td><td>Mask3</td><td>RW</td><td><b>Table 1.5. Values of Mask3</b>
<table>
<tr><td>Value</td><td>ID</td><td>Type</td><td>Description</td></tr>
<tr><td><code>0x0</code></td><td>InterruptDisabled</td><td>RW</td><td>Interrupt is not forwarded to CPU(s).</td></tr>
<tr><td><code>0x1</code></td><td>InterruptEnabled</td><td>RW</td><td>Interrupt is forwarded to CPU(s).</td></tr>
</table></td></tr>
</table>

<table>
<tr><td></td><td></td><td></td><td><b>Table 1.6. Resets of Mask3</b><table><tr><td>0x0</td><td>BusReset</td><td>RW</td><td>Default Bus Reset</td></tr></table></td></tr>
<tr><td>02</td><td>Mask2</td><td>RW</td><td><b>Table 1.7. Values of Mask2</b><table><tr><td>Value</td><td>ID</td><td>Type</td><td>Description</td></tr><tr><td>0x0</td><td>InterruptDisabled</td><td>RW</td><td>Interrupt is not forwarded to CPU(s).</td></tr><tr><td>0x1</td><td>InterruptEnabled</td><td>RW</td><td>Interrupt is forwarded to CPU(s).</td></tr></table><b>Table 1.8. Resets of Mask2</b><table><tr><td>0x0</td><td>BusReset</td><td>RW</td><td>Default Bus Reset</td></tr></table></td></tr>
<tr><td>01</td><td>Mask1</td><td>RW</td><td><b>Table 1.9. Values of Mask1</b><table><tr><td>Value</td><td>ID</td><td>Type</td><td>Description</td></tr><tr><td>0x0</td><td>InterruptDisabled</td><td>RW</td><td>Interrupt is not forwarded to CPU(s).</td></tr><tr><td>0x1</td><td>InterruptEnabled</td><td>RW</td><td>Interrupt is forwarded to CPU(s).</td></tr></table><b>Table 1.10. Resets of Mask1</b><table><tr><td>0x0</td><td>BusReset</td><td>RW</td><td>Default Bus Reset</td></tr></table></td></tr>
<tr><td>00</td><td>Mask0</td><td>RW</td><td><b>Table 1.11. Values of Mask0</b><table><tr><td>Value</td><td>ID</td><td>Type</td><td>Description</td></tr><tr><td>0x0</td><td>InterruptDisabled</td><td>RW</td><td>Interrupt is not forwarded to CPU(s).</td></tr><tr><td>0x1</td><td>InterruptEnabled</td><td>RW</td><td>Interrupt is forwarded to CPU(s).</td></tr></table><b>Table 1.12. Resets of Mask0</b><table><tr><td>0x0</td><td>BusReset</td><td>RW</td><td>Default Bus Reset</td></tr></table></td></tr>
</table>

1.1.1.2. Interrupt Request Register (InterruptRequestReg)

Table 1.13. Bits of Interrupt Request Register

<table>
<tr><td colspan="4">Bits of Interrupt Request Register</td></tr>
<tr><td colspan="3">0x04</td><td>Interrupt Request Register (InterruptRequestReg)</td></tr>
<tr><td>Bits</td><td>ID</td><td>Type</td><td>Description</td></tr>
<tr><td>03</td><td>Request3</td><td>R/W</td><td><b>Table 1.14. Values of Request3</b><table><tr><td>Value</td><td>ID</td><td>Type</td><td>Description</td></tr><tr><td>0x0</td><td>NotPending</td><td>R</td><td>An Interrupt is not pending.</td></tr><tr><td>0x1</td><td>Pending</td><td>R</td><td>An Interrupt is pending.</td></tr><tr><td>0x0</td><td>NotConfirmed</td><td>W</td><td>Writing this value has no effect.</td></tr><tr><td>0x1</td><td>Confirmed</td><td>W</td><td>Notifies HW that a pending interrupt has been recognized by SW.
SW confirms that a respective interrupt service routine has been entered. The value isn’t stored, thus there is no need to reset it to ‘0’ again. Solely the write action is sufficient.</td></tr></table><b>Table 1.15. Resets of Request3</b><table><tr><td>Init Value</td><td>ID</td><td>Impact</td><td>Description</td></tr><tr><td>0b0</td><td>Async</td><td>R</td><td>BusReset: Asynchronous Bus Reset
The bit Request is set to NotPending after reset in the usually attached InterruptCollector HW block.</td></tr></table></td></tr>
<tr><td>02</td><td>Request2</td><td>R/W</td><td><b>Table 1.16. Values of Request2</b><table><tr><td>Value</td><td>ID</td><td>Type</td><td>Description</td></tr><tr><td>0x0</td><td>NotPending</td><td>R</td><td>An Interrupt is not pending.</td></tr><tr><td>0x1</td><td>Pending</td><td>R</td><td>An Interrupt is pending.</td></tr><tr><td>0x0</td><td>NotConfirmed</td><td>W</td><td>Writing this value has no effect.</td></tr><tr><td>0x1</td><td>Confirmed</td><td>W</td><td>Notifies HW that a pending interrupt has been recognized by SW.
SW confirms that a respective interrupt service routine has been entered. The value isn’t stored, thus there is no need to reset it to ‘0’ again. Solely the write action is sufficient.</td></tr></table><b>Table 1.17. Resets of Request2</b><table><tr><td>Init Value</td><td>ID</td><td>Impact</td><td>Description</td></tr><tr><td>0b0</td><td>Async</td><td>R</td><td>BusReset: Asynchronous Bus Reset
The bit Request is set to NotPending after reset in the usually attached InterruptCollector HW block.</td></tr></table></td></tr>
<tr><td>01</td><td>Request1</td><td>R/W</td><td><b>Table 1.18. Values of Request1</b><table><tr><td>Value</td><td>ID</td><td>Type</td><td>Description</td></tr></table></td></tr>
</table>

Value	ID	Type	Description
`0x0`	NotPending	R	An Interrupt is not pending.
`0x1`	Pending	R	An Interrupt is pending.
`0x0`	NotConfirmed	W	Writing this value has no effect.
`0x1`	Confirmed	W	Notifies HW that a pending interrupt has been recognized by SW. SW confirms that a respective interrupt service routine has been entered. The value isn’t stored, thus there is no need to reset it to ‘0’ again. Solely the write action is sufficient.

Table 1.19. Resets of Request1

Init Value	ID	Impact	Description
`0b0`	Async	R	BusReset: Asynchronous Bus Reset The bit Request is set to NotPending after reset in the usually attached InterruptCollector HW block.

Bits	ID	Type	Description
00	Request0	R/W	Table 1.20. Values of Request0

Table 1.20. Values of Request0

Value	ID	Type	Description
`0x0`	NotPending	R	An Interrupt is not pending.
`0x1`	Pending	R	An Interrupt is pending.
`0x0`	NotConfirmed	W	Writing this value has no effect.
`0x1`	Confirmed	W	Notifies HW that a pending interrupt has been recognized by SW. SW confirms that a respective interrupt service routine has been entered. The value isn’t stored, thus there is no need to reset it to ‘0’ again. Solely the write action is sufficient.

Table 1.21. Resets of Request0

Init Value	ID	Impact	Description
`0b0`	Async	R	BusReset: Asynchronous Bus Reset The bit Request is set to NotPending after reset in the usually attached InterruptCollector HW block.

1.1.1.3. Interrupt Service Register (InterruptServiceReg)

Table 1.22. Bits of Interrupt Service Register

Bits of Interrupt Service Register			
`0x08`			Interrupt Service Register (InterruptServiceReg)
Bits	ID	Type	Description
03	Service3	R/W	Table 1.23. Values of Service3

Table 1.23. Values of Service3

Value	ID	Type	Description
`0x0`	Ended	R	The Interrupt service has ended.
`0x1`	InProcess	R	The Interrupt is in service.
`0x0`	NotConfirmed	W	Writing this value has no effect.
`0x1`	Confirmed	W	Notifies HW that a pending interrupt has been recognized by SW. SW confirms that a respective interrupt service routine has been processed completely. The value isn’t stored, thus there is no need to reset it to ‘0’ again. Solely the write action is sufficient.

Table 1.24. Resets of Service3

Init Value	ID	Impact	Description
`0b0`	Async	R	BusReset: Asynchronous Bus Reset The bit Service is set to Ended after reset in the usually attached InterruptCollector HW block.

Bits	ID	Type	Description
02	Service2	R/W	Table 1.25. Values of Service2

Table 1.25. Values of Service2

Value	ID	Type	Description
`0x0`	Ended	R	The Interrupt service has ended.
`0x1`	InProcess	R	The Interrupt is in service.
`0x0`	NotConfirmed	W	Writing this value has no effect.
`0x1`	Confirmed	W	Notifies HW that a pending interrupt has been recognized by SW. SW confirms that a respective interrupt service routine has been processed completely. The value isn’t stored, thus there is no need to reset it to ‘0’ again. Solely the write action is sufficient.

Table 1.26. Resets of Service2

Init Value	ID	Impact	Description
`0b0`	Async	R	BusReset: Asynchronous Bus Reset The bit Service is set to Ended after reset in the usually attached InterruptCollector HW block.

Bits	ID	Type	Description
01	Service1	R/W	Table 1.27. Values of Service1

Table 1.27. Values of Service1

Value	ID	Type	Description

<table>
<tr><td></td><td></td><td></td><td>
<table>
<tr><td>0x0</td><td>Ended</td><td>R</td><td>The Interrupt service has ended.</td></tr>
<tr><td>0x1</td><td>InProcess</td><td>R</td><td>The Interrupt is in service.</td></tr>
<tr><td>0x0</td><td>NotConfirmed</td><td>W</td><td>Writing this value has no effect.</td></tr>
<tr><td>0x1</td><td>Confirmed</td><td>W</td><td>Notifies HW that a pending interrupt has been recognized by SW.
SW confirms that a respective interrupt service routine has been processed completely. The value isn’t stored, thus there is no need to reset it to ‘0’ again. Solely the write action is sufficient.</td></tr>
</table>
<b>Table 1.28. Resets of Service1</b>
<table>
<tr><th>Init Value</th><th>ID</th><th>Impact</th><th>Description</th></tr>
<tr><td>0b0</td><td>Async</td><td>R</td><td>BusReset: Asynchronous Bus Reset
The bit Service is set to Ended after reset in the usually attached InterruptCollector HW block.</td></tr>
</table>
</td></tr>
<tr><td>00</td><td>Service0</td><td>R/W</td><td>
<b>Table 1.29. Values of Service0</b>
<table>
<tr><th>Value</th><th>ID</th><th>Type</th><th>Description</th></tr>
<tr><td>0x0</td><td>Ended</td><td>R</td><td>The Interrupt service has ended.</td></tr>
<tr><td>0x1</td><td>InProcess</td><td>R</td><td>The Interrupt is in service.</td></tr>
<tr><td>0x0</td><td>NotConfirmed</td><td>W</td><td>Writing this value has no effect.</td></tr>
<tr><td>0x1</td><td>Confirmed</td><td>W</td><td>Notifies HW that a pending interrupt has been recognized by SW.
SW confirms that a respective interrupt service routine has been processed completely. The value isn’t stored, thus there is no need to reset it to ‘0’ again. Solely the write action is sufficient.</td></tr>
</table>
<b>Table 1.30. Resets of Service0</b>
<table>
<tr><th>Init Value</th><th>ID</th><th>Impact</th><th>Description</th></tr>
<tr><td>0b0</td><td>Async</td><td>R</td><td>BusReset: Asynchronous Bus Reset
The bit Service is set to Ended after reset in the usually attached InterruptCollector HW block.</td></tr>
</table>
</td></tr>
</table>