12. Interrupts and Programmable Multilevel Interrupt Controller

12.1 Features

- Short and predictable interrupt response time
- Separate interrupt configuration and vector address for each interrupt
- Programmable multilevel interrupt controller
  - Interrupt prioritizing according to level and vector address
  - Three selectable interrupt levels for all interrupts: low, medium and high
  - Selectable, round-robin priority scheme within low-level interrupts
  - Non-maskable interrupts for critical functions
- Interrupt vectors optionally placed in the application section or the boot loader section

12.2 Overview

Interrupts signal a change of state in peripherals, and this can be used to alter program execution. Peripherals can have one or more interrupts, and all are individually enabled and configured. When an interrupt is enabled and configured, it will generate an interrupt request when the interrupt condition is present. The programmable multilevel interrupt controller (PMIC) controls the handling and prioritizing of interrupt requests. When an interrupt request is acknowledged by the PMIC, the program counter is set to point to the interrupt vector, and the interrupt handler can be executed.

All peripherals can select between three different priority levels for their interrupts: low, medium, and high. Interrupts are prioritized according to their level and their interrupt vector address. Medium-level interrupts will interrupt low-level interrupt handlers. High-level interrupts will interrupt both medium- and low-level interrupt handlers. Within each level, the interrupt priority is decided from the interrupt vector address, where the lowest interrupt vector address has the highest interrupt priority. Low-level interrupts have an optional round-robin scheduling scheme to ensure that all interrupts are serviced within a certain amount of time.

Non-maskable interrupts (NMI) are also supported, and can be used for system critical functions.

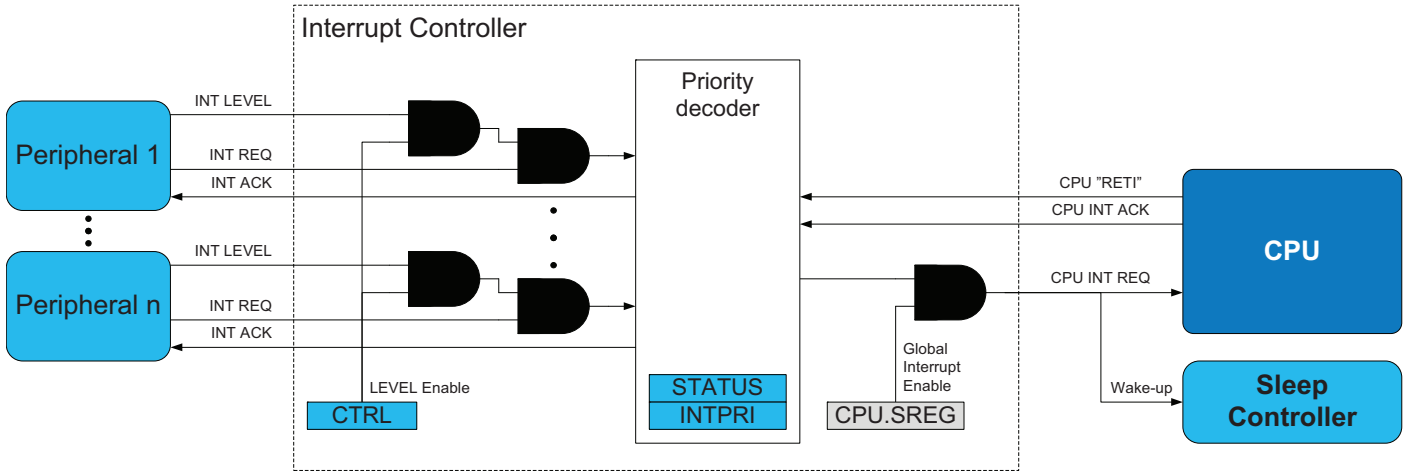
12.3 Operation

Interrupts must be globally enabled for any interrupts to be generated. This is done by setting the global interrupt enable (I) bit in the CPU status register. The I bit will not be cleared when an interrupt is acknowledged. Each interrupt level must also be enabled before interrupts with the corresponding level can be generated.

When an interrupt is enabled and the interrupt condition is present, the PMIC will receive the interrupt request. Based on the interrupt level and interrupt priority of any ongoing interrupts, the interrupt is either acknowledged or kept pending until it has priority. When the interrupt request is acknowledged, the program counter is updated to point to the interrupt vector. The interrupt vector is normally a jump to the interrupt handler; the software routine that handles the interrupt. After returning from the interrupt handler, program execution continues from where it was before the interrupt occurred. One instruction is always executed before any pending interrupt is served.

The PMIC status register contains state information that ensures that the PMIC returns to the correct interrupt level when the RETI (interrupt return) instruction is executed at the end of an interrupt handler. Returning from an interrupt will return the PMIC to the state it had before entering the interrupt. The status register (SREG) is not saved automatically upon an interrupt request. The RET (subroutine return) instruction cannot be used when returning from the interrupt handler routine, as this will not return the PMIC to its correct state.

Figure 12-1. Interrupt controller overview



12.4 Interrupts

All interrupts and the reset vector each have a separate program vector address in the program memory space. The lowest address in the program memory space is the reset vector. All interrupts are assigned individual control bits for enabling and setting the interrupt level, and this is set in the control registers for each peripheral that can generate interrupts. Details on each interrupt are described in the peripheral where the interrupt is available.

All interrupts have an interrupt flag associated with it. When the interrupt condition is present, the interrupt flag will be set, even if the corresponding interrupt is not enabled. For most interrupts, the interrupt flag is automatically cleared when executing the interrupt vector. Writing a logical one to the interrupt flag will also clear the flag. Some interrupt flags are not cleared when executing the interrupt vector, and some are cleared automatically when an associated register is accessed (read or written). This is described for each individual interrupt flag.

If an interrupt condition occurs while another, higher priority interrupt is executing or pending, the interrupt flag will be set and remembered until the interrupt has priority. If an interrupt condition occurs while the corresponding interrupt is not enabled, the interrupt flag will be set and remembered until the interrupt is enabled or the flag is cleared by software. Similarly, if one or more interrupt conditions occur while global interrupts are disabled, the corresponding interrupt flag will be set and remembered until global interrupts are enabled. All pending interrupts are then executed according to their order of priority.

Interrupts can be blocked when executing code from a locked section; e.g., when the boot lock bits are programmed. This feature improves software security. Refer to “Memory Programming” on page 407 for details on lock bit settings.

Interrupts are automatically disabled for up to four CPU clock cycles when the configuration change protection register is written with the correct signature. Refer to “Configuration Change Protection” on page 13 for more details.

12.4.1 NMI – Non-Maskable Interrupts

Which interrupts represent NMI and which represent regular interrupts cannot be selected. Non-maskable interrupts must be enabled before they can be used. Refer to the device datasheet for NMI present on each device.

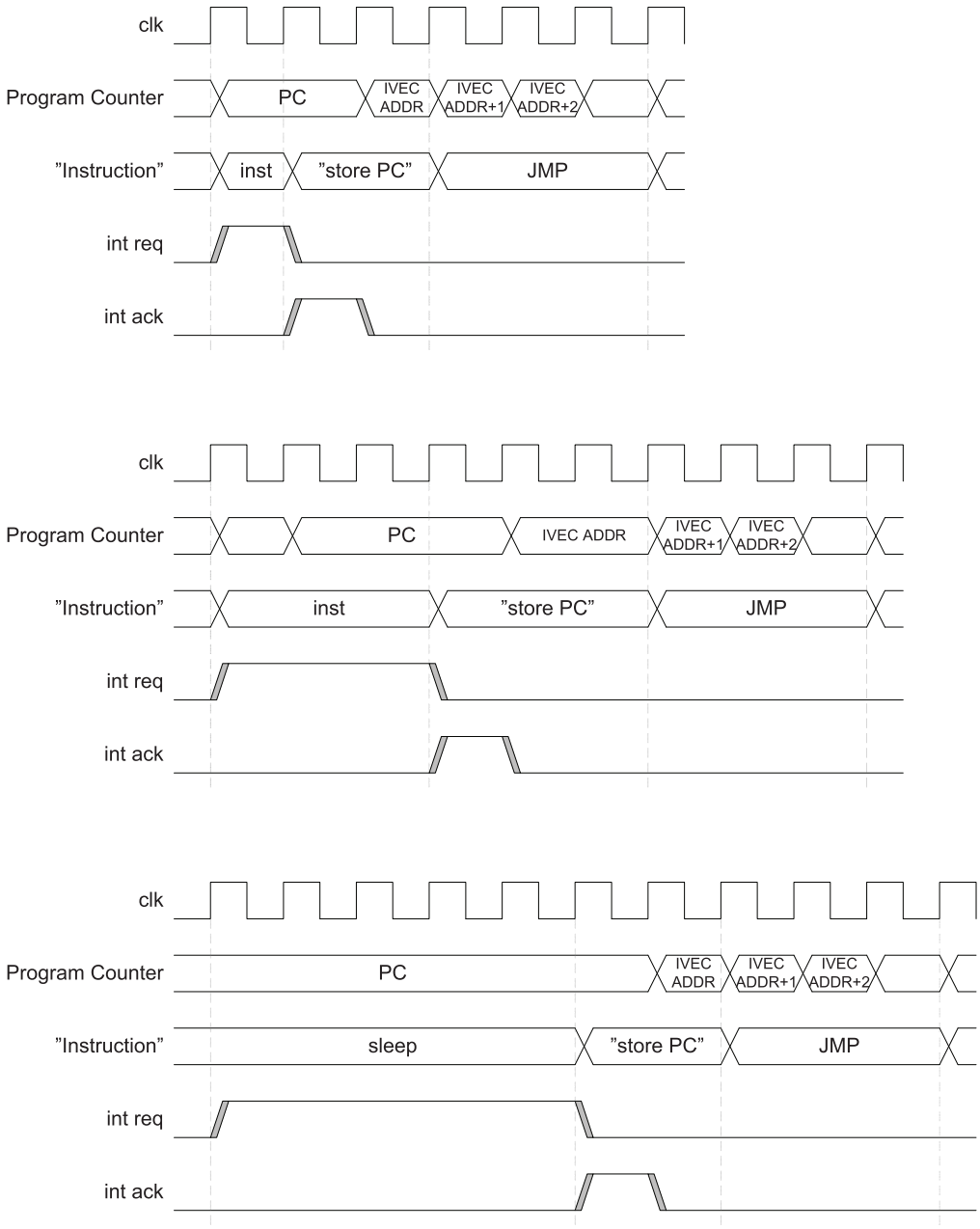
An NMI will be executed regardless of the setting of the I bit, and it will never change the I bit. No other interrupts can interrupt a NMI handler. If more than one NMI is requested at the same time, priority is static according to the interrupt vector address, where the lowest address has highest priority.

12.4.2 Interrupt Response Time

The interrupt response time for all the enabled interrupts is three CPU clock cycles, minimum; one cycle to finish the ongoing instruction and two cycles to store the program counter to the stack. After the program counter is pushed on the stack, the program vector for the interrupt is executed. The jump to the interrupt handler takes three clock cycles.

If an interrupt occurs during execution of a multicycle instruction, this instruction is completed before the interrupt is served. See Figure 12-2 on page 133 for more details.

Figure 12-2. Interrupt execution of a multi cycle instruction.



If an interrupt occurs when the device is in sleep mode, the interrupt execution response time is increased by five clock cycles. In addition, the response time is increased by the start-up time from the selected sleep mode.

A return from an interrupt handling routine takes four to five clock cycles, depending on the size of the program counter. During these clock cycles, the program counter is popped from the stack and the stack pointer is incremented.

12.5 Interrupt level

The interrupt level is independently selected for each interrupt source. For any interrupt request, the PMIC also receives the interrupt level for the interrupt. The interrupt levels and their corresponding bit values for the interrupt level configuration of all interrupts is shown in Table 12-1.

Table 12-1. Interrupt levels.

Interrupt level configuration	Group configuration	Description
00	OFF	Interrupt disabled.
01	LO	Low-level interrupt
10	MED	Medium-level interrupt
11	HI	High-level interrupt

The interrupt level of an interrupt request is compared against the current level and status of the interrupt controller. An interrupt request of a higher level will interrupt any ongoing interrupt handler from a lower level interrupt. When returning from the higher level interrupt handler, the execution of the lower level interrupt handler will continue.

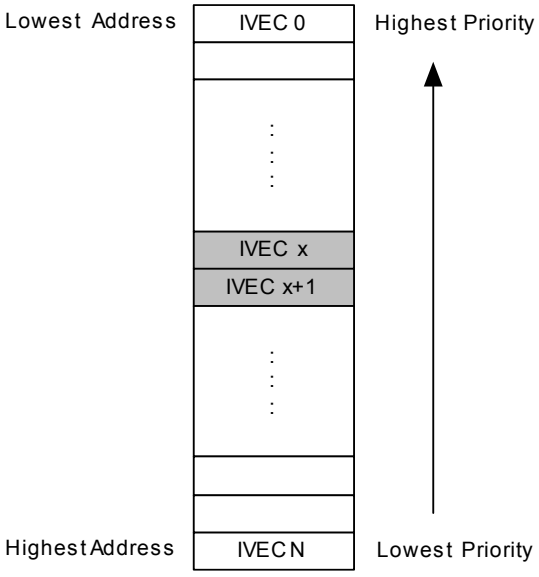
12.6 Interrupt priority

Within each interrupt level, all interrupts have a priority. When several interrupt requests are pending, the order in which interrupts are acknowledged is decided both by the level and the priority of the interrupt request. Interrupts can be organized in a static or dynamic (round-robin) priority scheme. High- and medium-level interrupts and the NMI will always have static priority. For low-level interrupts, static or dynamic priority scheduling can be selected.

12.6.1 Static priority

Interrupt vectors (IVEC) are located at fixed addresses. For static priority, the interrupt vector address decides the priority within one interrupt level, where the lowest interrupt vector address has the highest priority. Refer to the device datasheet for the interrupt vector table with the base address for all modules and peripherals with interrupt capability. Refer to the interrupt vector summary of each module and peripheral in this manual for a list of interrupts and their corresponding offset address within the different modules and peripherals.

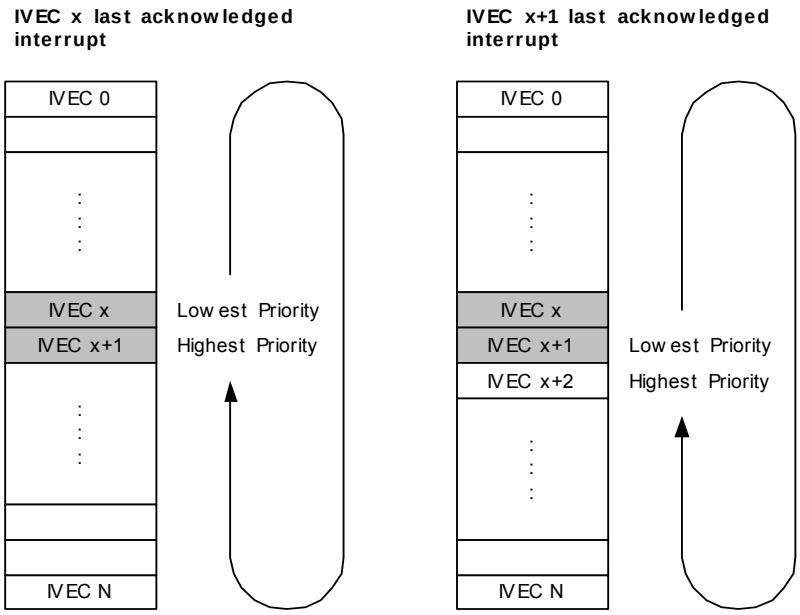
Figure 12-3. Static priority.



12.6.2 Round-robin Scheduling

To avoid the possible starvation problem for low-level interrupts with static priority, where some interrupts might never be served, the PMIC offers round-robin scheduling for low-level interrupts. When round-robin scheduling is enabled, the interrupt vector address for the last acknowledged low-level interrupt will have the lowest priority the next time one or more interrupts from the low level is requested.

Figure 12-4. Round-robin scheduling.



12.7 Interrupt vector locations

Table 12-2 on page 135 shows reset and Interrupt vectors placement for the various combinations of BOOTRST and IVSEL settings. If the program never enables an interrupt source, the Interrupt Vectors are not used, and regular program code can be placed at these locations. This is also the case if the Reset Vector is in the Application section while the Interrupt Vectors are in the Boot section or vice versa.

Table 12-2. Reset and interrupt vectors placement.

BOOTRST	IVSEL	Reset address	Interrupt vectors start address
1	0	0x0000	0x0002
1	1	0x0000	Boot Reset Address + 0x0002
0	0	Boot Reset Address	0x0002
0	1	Boot Reset Address	Boot Reset Address + 0x0002

12.8 Register description

12.8.1 STATUS – Status register

Bit	7	6	5	4	3	2	1	0
+0x00	NMIEX	–	–	–	–	HILVLEX	MEDLVLEX	LOLVLEX
Read/Write	R	R	R	R	R	R	R	R
Initial Value	0	0	0	0	0	0	0	0

- **Bit 7 – NMIEX: Non-Maskable Interrupt Executing**
  This flag is set if a non-maskable interrupt is executing. The flag will be cleared when returning (RETI) from the interrupt handler.
- **Bit 6:3 – Reserved**
  These bits are unused and reserved for future use. For compatibility with future devices, always write these bits to zero when this register is written.
- **Bit 2 – HILVLEX: High-level Interrupt Executing**
  This flag is set when a high-level interrupt is executing or when the interrupt handler has been interrupted by an NMI. The flag will be cleared when returning (RETI) from the interrupt handler.
- **Bit 1 – MEDLVLEX: Medium-level Interrupt Executing**
  This flag is set when a medium-level interrupt is executing or when the interrupt handler has been interrupted by an interrupt from higher level or an NMI. The flag will be cleared when returning (RETI) from the interrupt handler.
- **Bit 0 – LOLVLEX: Low-level Interrupt Executing**
  This flag is set when a low-level interrupt is executing or when the interrupt handler has been interrupted by an interrupt from higher level or an NMI. The flag will be cleared when returning (RETI) from the interrupt handler.

12.8.2 INTPRI – Interrupt priority register

Bit	7	6	5	4	3	2	1	0
+0x01	INTPRI[7:0]							
Read/Write	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
Initial Value	0	0	0	0	0	0	0	0

- **Bit 7:0 – INTPRI: Interrupt Priority**
  When round-robin scheduling is enabled, this register stores the interrupt vector of the last acknowledged low-level interrupt. The stored interrupt vector will have the lowest priority the next time one or more low-level interrupts are pending. The register is accessible from software to change the priority queue. This register is not reinitialized to its initial value if round-robing scheduling is disabled, and so if default static priority is needed, the register must be written to zero.

12.8.3 CTRL – Control register

Bit	7	6	5	4	3	2	1	0
+0x02	**RREN**	**IVSEL**	**–**	**–**	**–**	**HILVLEN**	**MEDLVLEN**	**LOLVLEN**
Read/Write	R/W	R/W	R	R	R	R/W	R/W	R/W
Initial Value	0	0	0	0	0	0	0	0

- **Bit 7 – RREN: Round-robin Scheduling Enable**
  When the RREN bit is set, the round-robin scheduling scheme is enabled for low-level interrupts. When this bit is cleared, the priority is static according to interrupt vector address, where the lowest address has the highest priority.

- **Bit 6 – IVSEL: Interrupt Vector Select**
  When the IVSEL bit is cleared (zero), the interrupt vectors are placed at the start of the application section in flash. When this bit is set (one), the interrupt vectors are placed in the beginning of the boot section of the flash. Refer to the device datasheet for the absolute address.
  This bit is protected by the configuration change protection mechanism. Refer to “Configuration Change Protection” on page 13 for details.

- **Bit 5:3 – Reserved**
  These bits are unused and reserved for future use. For compatibility with future devices, always write these bits to zero when this register is written.

- **Bit 2 – HILVLEN: High-level Interrupt Enable**(1)
  When this bit is set, all high-level interrupts are enabled. If this bit is cleared, high-level interrupt requests will be ignored.

- **Bit 1 – MEDLVLEN: Medium-level Interrupt Enable**(1)
  When this bit is set, all medium-level interrupts are enabled. If this bit is cleared, medium-level interrupt requests will be ignored.

- **Bit 0 – LOLVLEN: Low-level Interrupt Enable**(1)
  When this bit is set, all low-level interrupts are enabled. If this bit is cleared, low-level interrupt requests will be ignored.

Note: 1. Ignoring interrupts will be effective one cycle after the bit is cleared.

12.9 Register summary

Address	Name	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0	Page
+0x00	STATUS	NMIEX	–	–	–	–	HILVLEX	MEDLVLEX	LOLVLEX	136
+0x01	INTPRI	INTPRI[7:0]								136
+0x02	CTRL	RREN	IVSEL	–	–	–	HILVLEN	MEDLVLEN	LOLVLEN	137