OMAP35xx Applications Processor Interrupt Controller (INTC)

Texas Instruments OMAP™ Family of Products

Technical Reference Manual

Literature Number: SPRUFA8
February 2008

Texas Instruments

Contents

List of Figures

List of Tables

Interrupt Controller (INTC)

This chapter gives an overview of the interrupt controllers (INTCs) and describes in detail the MPU subsystem interrupt controller (MPU_INTC) module used in the OMAP35xx stand-alone Applications Processor.

Note: This chapter gives information about all modules and features in the high-tier device. See Chapter 1, *OMAP35xx Family* section, to check availability of modules and features. Ensure that interrupts of unavailable modules and features are masked in MPU/IVA subsystems.

Topic **Page**

1.1 Interrupt Controller Overview

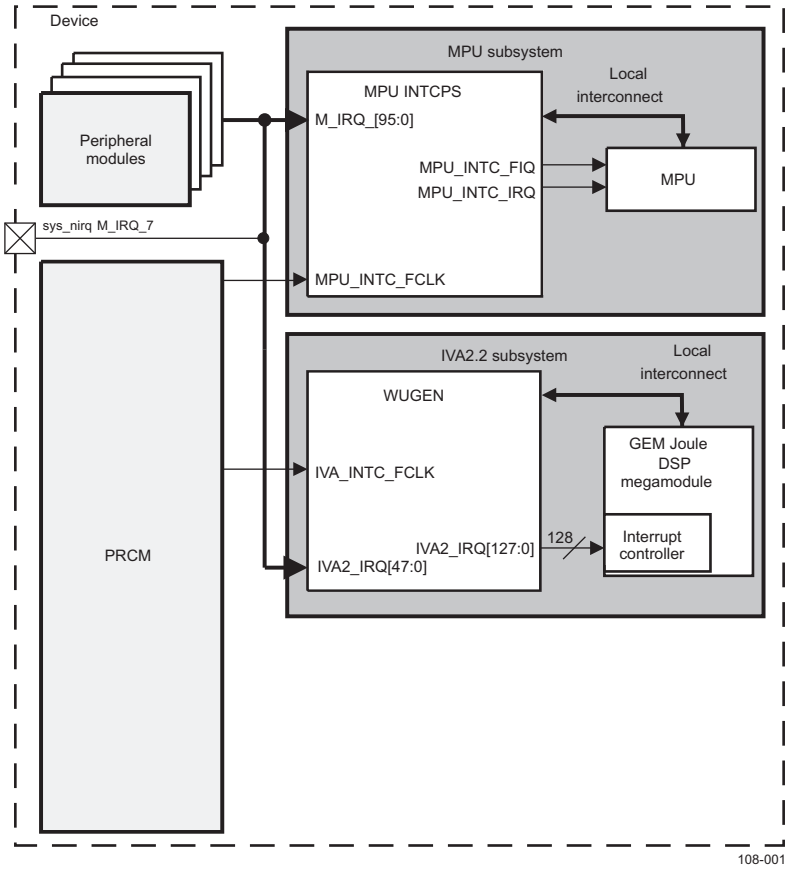
The device provides two interrupt controller (INTC) modules:

- **MPU subsystem INTC (INTCPS):** This module handles all MPU-related events, using Priority Threshold and Security Features. It communicates with the ARM Cortex-A8™ processor using a private local interconnect, and runs at half the speed of the processor.
- **IVA2.2 subsystem INTC:** This module is a specific combination of WUGEN (wake-up generator) and the GEM Joule DSP megamodule interrupt controller (IC). It is used in the device, but is not described in detail in this chapter. For detailed information about this INTC, see *IVA2.2 Subsystem* chapter.

Note: Some features may not be available or supported in your particular device. For more information, see Chapter 1, the *OMAP35xx Family* section, and your device-specific data manual.

Figure 1-1 shows the internal interrupt scheme.

Figure 1-1. Interrupt Controllers Highlight



1.2 Interrupt Controller Environment

The INTC can handle two types of interrupts originating from an external device:

- sys_nirq interrupt inputs:

  The MPU INTC handles external interrupts through a dedicated sys_nirq interrupt line that connects the INTC module with a TWL4030 power IC. An interrupt can generate a system wake-up event.

  If the system is idle and the external interrupt is masked, the interrupt cannot wake up the system. Like other interrupt lines, the external interrupt is active at low level and is acknowledged by the software according to the common programming model.

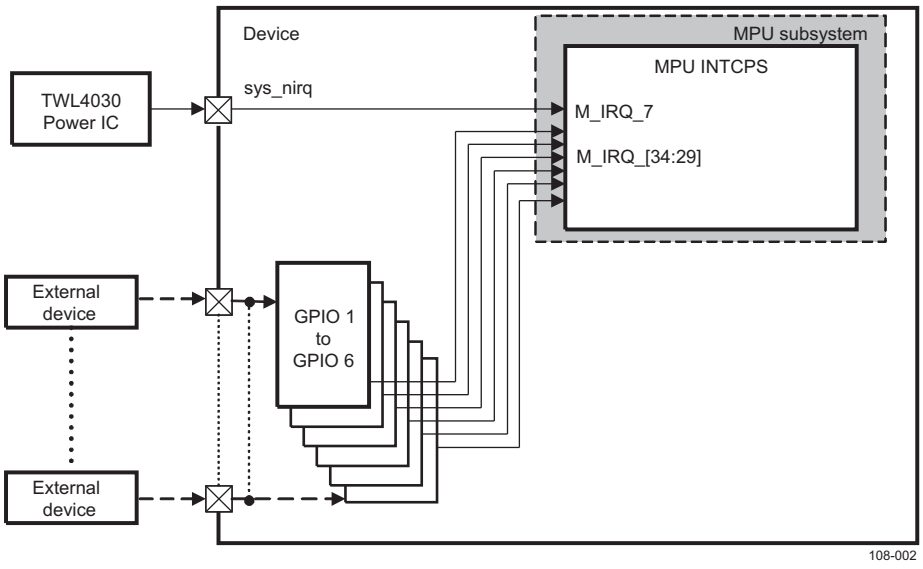
  Note: If the CORE power domain is in retention or off mode, both the interrupt requests (internal or external) and the MPU INTC have no effect. The CORE power domain does not wake up, and the interrupt is not signaled to the MPU.

- GPIO interrupt inputs:

  External devices can also use GPIO modules to generate interrupts to the MPU. There are six dedicated interrupt lines to the MPU INTC. One interrupt line is associated with each GPIO module. Each GPIO module can generate a single interrupt whenever there is at least one event in any one of the configured 32 GPIO inputs. For more information about GPIO features, see the *General-Purpose Interface* chapter.

Figure 1-2 shows the relationship between the device and external interrupts.

Figure 1-2. Interrupts from External Devices



The features specific to INTCPS are:

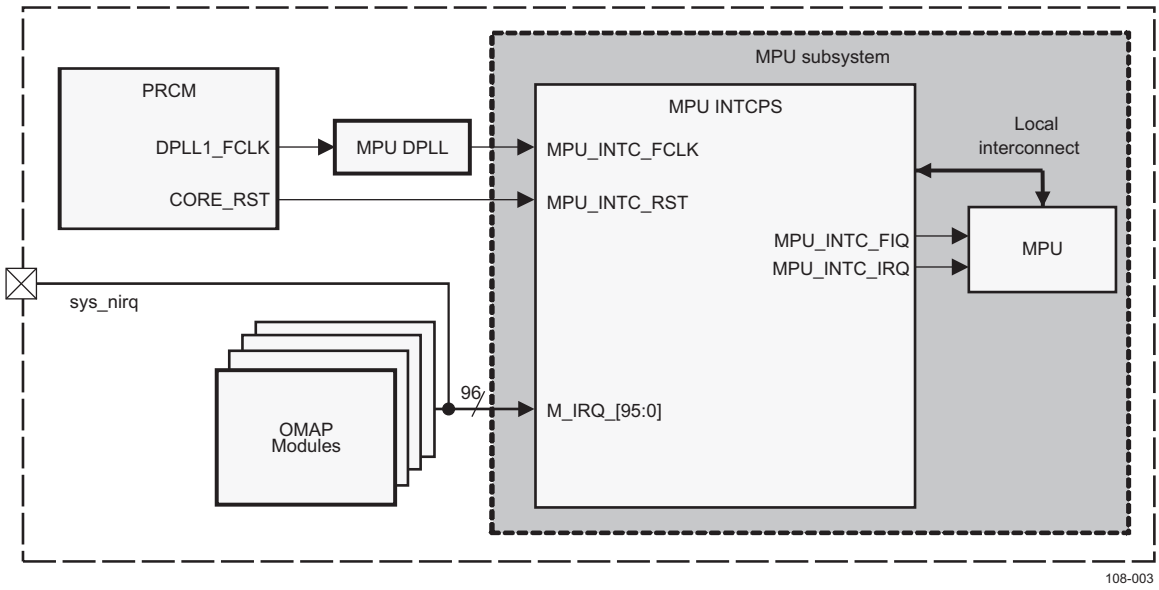
- Up to 96 level-sensitive interrupt inputs
- Individual priority (up to 64) for each interrupt input
- Interrupt lines connected to internal module interrupts
- One incoming interrupt line from an external device

1.3 MPU Subsystem INTCPS Integration

The INTCPS module is the interface between incoming interrupts and the two interrupt inputs of the MPU. It can handle up to 96 request inputs that can be configured as MPU FIQ or IRQ interrupt requests.

Figure 1-3 shows the integration of the INTCPS in the MPU subsystem.

Figure 1-3. MPU Subsystem INTCPS Integration



The MPU subsystem INTCPS is directly connected to the MPU by an MPU peripheral port. Consequently, the MPU subsystem INTCPS is accessible and visible only by the MPU.

1.3.1 Clocking, Reset, and Power Management Scheme

1.3.1.1 MPU Subsystem INTC Clocks

The MPU subsystem INTCPS runs at half the rate of the MPU functional clock (see the *MPU Subsystem* chapter).

The interface clock used for register access runs at the rate of the interconnect bus clock (equal to the rate of the MPU interface clock; see the *Power, Reset, and Clock Management* chapter).

The synchronizer clock allows external asynchronous interrupts to be resynchronized before they are masked.

Table 1-1 lists the MPU subsystem INTC clock rates.

Table 1-1. MPU Subsystem INTC Clock Rates

Clock	Frequency	Name	Comments
Functional	ARM_FCLK	MPU_INTC_FCLK	Source is the MPU DPLL.
Interface	ARM_FCLK	MPU_INTC_ICLK	Source is the PRCM module.
Synchronizer	MPU_INTC_FCLK	Synchronizer clock (module internal clock)	Source is the MPU_INTC_FCLK.

1.3.1.2 Hardware and Software Reset

Table 1-2 lists the MPU subsystem INTC resets.

Table 1-2. Hardware and Software Reset

Type	Name	Source	Activation	Domain
Hardware	CORE_RST	PRCM	Active low	CORE
Software	SOFTRESET	MPU_INTC.INTCPS_SYSCONFIG[1] SOFTRESET bit	Active at 1	MPU INTC internal

1.3.1.3 Power Management

The MPU subsystem INTC belongs to the CORE power domain. As part of CORE power domain, it is sensitive to a CORE_RST issued by the PRCM. For more information about the CORE power domain implementation and CORE_RST signal, see the *Power, Reset, and Clock Management* chapter.

The MPU INTC clocks come from the MPU DPLL. For more information about these clocks control, see the *MPU Subsystem* chapter.

1.3.2 Interrupt Request Lines

Table 1-3 lists the incoming and outgoing interrupt lines of the INTCPS.

Table 1-3. Interrupt Lines Incoming and Outgoing

Type	Number	Name	Mapping	Comments
Interrupt request inputs	Up to 96	M_IRQ_[95:0]	See Table 1-4	Inputs to INTCPS module, source from various modules.
Interrupt request outputs	2	MPU_INTC_FIQ	MPU_INTC_FIQ	Outgoing to MPU Fast Interrupt
		MPU_INTC_IRQ	MPU_INTC_IRQ	Outgoing to MPU Normal Interrupt

Note: Interrupt request signals are active at low level.

CAUTION

A single interrupt source can be physically mapped to multiple INTCs (MPU subsystem, IVA2.2 subsystem). With multiple-mapped interrupts, it is strongly recommended each interrupt source be unmasked in only one INTC at a time.

This section gives information about all modules and features in the high-tier device. See Chapter 1, *OMAP35xx Family* section, to check availability of modules and features. Ensure that interrupts of unavailable modules and features are masked in MPU subsystem.

Table 1-4 lists interrupt mappings to the MPU subsystem.

Table 1-4. Interrupt Mapping to the MPU Subsystem(1)

IRQ	Source	Description
M_IRQ_0	EMUINT	MPU emulation(2)
M_IRQ_1	COMMTX	MPU emulation(2)
M_IRQ_2	COMMRX	MPU emulation(2)
M_IRQ_3	BENCH	MPU emulation(2)
M_IRQ_4	MCBSP2_ST_IRQ	Sidetone MCBSP2 overflow
M_IRQ_5	MCBSP3_ST_IRQ	Sidetone MCBSP3 overflow
M_IRQ_6	SSM_ABORT_IRQ	MPU subsystem secure state-machine abort (2)
M_IRQ_7	sys_nirq	External source (active low)

(1) All the IRQ signals are active at low level.
(2) These interrupts are internally generated within the MPU subsystem.

Table 1-4. Interrupt Mapping to the MPU Subsystem (continued)

IRQ	Source	Description
M_IRQ_8	RESERVED	RESERVED
M_IRQ_9	SMX_DBG_IRQ	SMX error for debug
M_IRQ_10	SMX_APP_IRQ	SMX error for application
M_IRQ_11	PRCM_MPU_IRQ	PRCM module IRQ
M_IRQ_12	SDMA_IRQ0	System DMA request 0 (3)
M_IRQ_13	SDMA_IRQ1	System DMA request 1 (3)
M_IRQ_14	SDMA_IRQ2	System DMA request 2
M_IRQ_15	SDMA_IRQ3	System DMA request 3
M_IRQ_16	MCBSP1_IRQ	McBSP module 1 IRQ (3)
M_IRQ_17	MCBSP2_IRQ	McBSP module 2 IRQ (3)
M_IRQ_18	SR1_IRQ	SmartReflex™ 1
M_IRQ_19	SR2_IRQ	SmartReflex™ 2
M_IRQ_20	GPMC_IRQ	General-purpose memory controller module
M_IRQ_21	SGX_IRQ	2D/3D graphics module
M_IRQ_22	MCBSP3_IRQ	McBSP module 3 (3)
M_IRQ_23	MCBSP4_IRQ	McBSP module 4 (3)
M_IRQ_24	CAM_IRQ0	Camera interface request 0
M_IRQ_25	DSS_IRQ	Display subsystem module (3)
M_IRQ_26	MAIL_U0_MPU_IRQ	Mailbox user 0 request
M_IRQ_27	MCBSP5_IRQ	McBSP module 5 (3)
M_IRQ_28	IVA2_MMU_IRQ	IVA2 MMU
M_IRQ_29	GPIO1_MPU_IRQ	GPIO module 1 (3)
M_IRQ_30	GPIO2_MPU_IRQ	GPIO module 2 (3)
M_IRQ_31	GPIO3_MPU_IRQ	GPIO module 3 (3)
M_IRQ_32	GPIO4_MPU_IRQ	GPIO module 4 (3)
M_IRQ_33	GPIO5_MPU_IRQ	GPIO module 5 (3)
M_IRQ_34	GPIO6_MPU_IRQ	GPIO module 6 (3)
M_IRQ_35	USIM_IRQ	USIM interrupt (HS devices only) (4)
M_IRQ_36	WDT3_IRQ	Watchdog timer module 3 overflow
M_IRQ_37	GPT1_IRQ	General-purpose timer module 1
M_IRQ_38	GPT2_IRQ	General-purpose timer module 2
M_IRQ_39	GPT3_IRQ	General-purpose timer module 3
M_IRQ_40	GPT4_IRQ	General-purpose timer module 4
M_IRQ_41	GPT5_IRQ	General-purpose timer module 5 (3)
M_IRQ_42	GPT6_IRQ	General-purpose timer module 6 (3)
M_IRQ_43	GPT7_IRQ	General-purpose timer module 7 (3)
M_IRQ_44	GPT8_IRQ	General-purpose timer module 8 (3)
M_IRQ_45	GPT9_IRQ	General-purpose timer module 9
M_IRQ_46	GPT10_IRQ	General-purpose timer module 10
M_IRQ_47	GPT11_IRQ	General-purpose timer module 11
M_IRQ_48	SPI4_IRQ	McSPI module 4
M_IRQ_49	SHA1MD5_IRQ2	SHA-1/MD5 crypto-accelerator 2 (HS devices only) (4)
M_IRQ_50	FPKA_IRQREADY_N	PKA crypto-accelerator (HS devices only) (4)
M_IRQ_51	SHA2MD5_IRQ	SHA-2/MD5 crypto-accelerator 1 (HS devices only) (4)

(3) Shared with the IVA2.2 interrupt controller.
(4) To determine if a HS version of your device is available and for more information on HS devices, please refer to your device-specific data manual.

Table 1-4. Interrupt Mapping to the MPU Subsystem (continued)

IRQ	Source	Description
M_IRQ_52	RNG_IRQ	RNG module (HS devices only) (4)
M_IRQ_53	MG_IRQ	MG function (5)
M_IRQ_54	MCBSP4_IRQ_TX	McBSP module 4 transmit(5)
M_IRQ_55	MCBSP4_IRQ_RX	McBSP module 4 receive(5)
M_IRQ_56	I2C1_IRQ	I^2C module 1
M_IRQ_57	I2C2_IRQ	I^2C module 2
M_IRQ_58	HDQ_IRQ	HDQ™/One-wire™
M_IRQ_59	McBSP1_IRQ_TX	McBSP module 1 transmit(5)
M_IRQ_60	McBSP1_IRQ_RX	McBSP module 1 receive(5)
M_IRQ_61	I2C3_IRQ	I^2C module 3
M_IRQ_62	McBSP2_IRQ_TX	McBSP module 2 transmit(5)
M_IRQ_63	McBSP2_IRQ_RX	McBSP module 2 receive(5)
M_IRQ_64	FPKA_IRQRERROR_N	PKA crypto-accelerator (HS devices only) (6)
M_IRQ_65	SPI1_IRQ	McSPI module 1
M_IRQ_66	SPI2_IRQ	McSPI module 2
M_IRQ_67	RESERVED	RESERVED
M_IRQ_68	RESERVED	RESERVED
M_IRQ_69	RESERVED	RESERVED
M_IRQ_70	RESERVED	RESERVED
M_IRQ_71	RESERVED	RESERVED
M_IRQ_72	UART1_IRQ	UART module 1
M_IRQ_73	UART2_IRQ	UART module 2
M_IRQ_74	UART3_IRQ	UART module 3 (also infrared)(5)
M_IRQ_75	PBIAS_IRQ	Merged interrupt for PBIASlite1 and 2
M_IRQ_76	OHCI_IRQ	OHCI controller HSUSB MP Host Interrupt
M_IRQ_77	EHCI_IRQ	EHCI controller HSUSB MP Host Interrupt
M_IRQ_78	TLL_IRQ	HSUSB MP TLL Interrupt
M_IRQ_79	PARTHASH_IRQ	SHA2/MD5 crypto-accelerator 1 (HS devices only) (6)
M_IRQ_80	Reserved	Reserved
M_IRQ_81	MCBSP5_IRQ_TX	McBSP module 5 transmit(5)
M_IRQ_82	MCBSP5_IRQ_RX	McBSP module 5 receive(5)
M_IRQ_83	MMC1_IRQ	MMC/SD module 1
M_IRQ_84	MS_IRQ	MS-PRO™ module
M_IRQ_85	Reserved	Reserved
M_IRQ_86	MMC2_IRQ	MMC/SD module 2
M_IRQ_87	MPU_ICR_IRQ	MPU ICR
M_IRQ_88	RESERVED	RESERVED
M_IRQ_89	MCBSP3_IRQ_TX	McBSP module 3 transmit(5)
M_IRQ_90	MCBSP3_IRQ_RX	McBSP module 3 receive(5)
M_IRQ_91	SPI3_IRQ	McSPI module 3
M_IRQ_92	HSUSB_MC_NINT	High-Speed USB OTG controller
M_IRQ_93	HSUSB_DMA_NINT	High-Speed USB OTG DMA controller
M_IRQ_94	MMC3_IRQ	MMC/SD module 3
M_IRQ_95	GPT12_IRQ	General-purpose timer module 12

(5) Shared with the IVA2.2 interrupt controller.
(6) To determine if a HS version of your device is available and for more information on HS devices, please refer to your device-specific data manual.

1.4 Interrupt Controller Functional Description

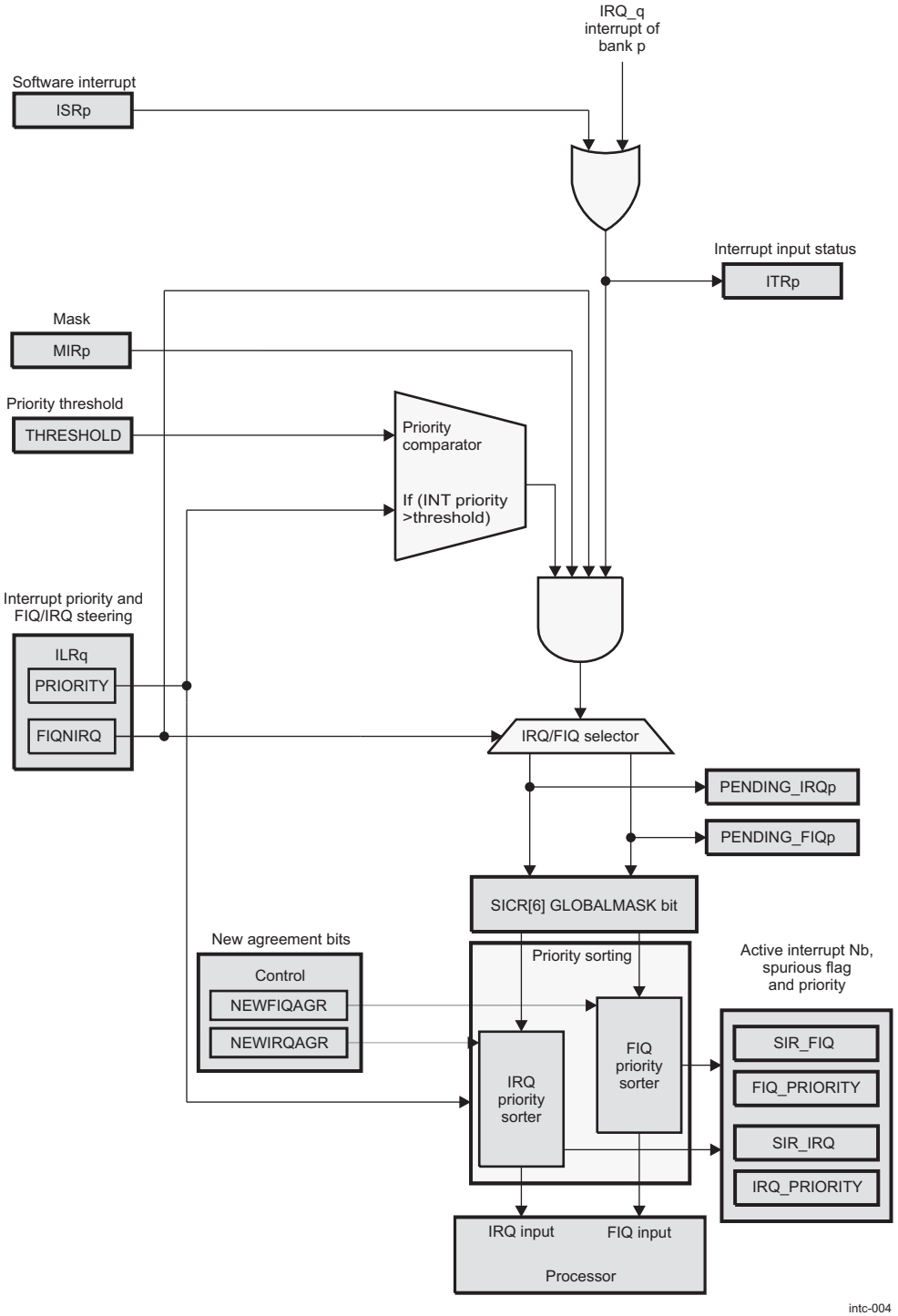
The main features of the INTCPS are:

- Individual priority (up to 64 levels) for each interrupt input
- Ability for each interrupt to be steered to either FIQ or IRQ
- Independent priority sorting for FIQ and IRQ; FIQ sorting is processed concurrently with IRQ sorting.
- Priority masking: Interrupts can be masked based on the priority threshold register.
- Atomic bit set and clear capability for interrupt mask and software interrupt registers
- For high-security (HS) devices only:
  - Global interrupt mask
  - Protection for secure interrupts
- Power-management and wake-up support
- Auto-idle power-saving support

The INTCPS processes incoming interrupts by masking and priority sorting, then it generates the interrupt requests to the MPU.

Figure 1-4 shows the top-level view of the interrupt processing.

Figure 1-4. Top-Level Block Diagram



(1) Public Mask = SICR[3] PUBLICINHIBIT bit OR (SICR[2] AUTOINHIBIT bit AND SICR[0] SSMFIQSTATUS bit)

(2) The SCRn forces the interrupt priority to 0 (highest).

(3) The SCRn forces the interrupt to FIQ.

(4) The comparator output is TRUE only if the interrupt priority is higher than the threshold register priority. The highest priority is 0x0; the lowest priority is 0x3F (63).

1.4.1 Interrupt Processing

1.4.1.1 Input Selection

The INTCPS supports only level-sensitive incoming interrupt detection. A peripheral asserting an interrupt maintains it until software has handled the interrupt and instructed the peripheral to de-assert the interrupt.

A software interrupt is generated if the corresponding bit in the MPU_INTC.INTCPS_ISR_SETn register is set (register bank number: n = [0,2] for the MPU subsystem INTCPS, 96 incoming interrupt lines are supported). The software interrupt clears when the corresponding bit in the MPU_INTC.INTCPS_ISR_CLEARn register is written. Typical use of this feature is software debugging.

1.4.1.2 Masking

1.4.1.2.1 Individual Masking

Detection of interrupts on each incoming interrupt line can be enabled or disabled independently by the MPU_INTC.INTCPS_MIRn interrupt mask register. In response to an unmasked incoming interrupt, the INTCPS can generate one of two types of interrupt requests to the processor:

- IRQ: low-priority interrupt request
- FIQ: fast interrupt request

The type of interrupt request is determined by the MPU_INTC.INTCPS_ILRm[0] FIQNIRQ bit (m= [0,95]).

The current incoming interrupt status before masking is readable from the MPU_INTC.INTCPS_ITRn register. After masking and IRQ/FIQ selection, and before priority sorting is done, the interrupt status is readable from the MPU_INTC.INTCPS_PENDING_IRQn and MPU_INTC.INTCPS_PENDING_FIQn registers.

1.4.1.2.2 Global Masking (HS Devices Only)

Software may use global masking for security purposes, but this feature must be used with care. To determine if a HS version of your device is available and for more information on HS devices, please refer to your device-specific data manual.

1.4.1.2.3 Priority Masking

To enable faster processing of high-priority interrupts, a programmable priority masking threshold is provided (the MPU_INTC.INTCPS_THRESHOLD[7:0] PRIORITYTHRESHOLD field). This priority threshold allows preemption by higher priority interrupts; all interrupts of lower or equal priority than the threshold are masked. However, priority 0 can never be masked by this threshold; a priority threshold of 0 is treated the same way as priority 1.

PRIORITY and PRIORITYTHRESHOLD fields values can be set between 0x0 and 0x3F; 0x0 is the highest priority and 0x3F is the lowest priority.

When priority masking is not necessary, a priority threshold value of 0xFF disables the priority threshold mechanism. This value is also the reset default for backward compatibility with previous versions of the INTCPS.

1.4.1.3 Priority Sorting

A priority level (0 being the highest) is assigned to each incoming interrupt line. Both the priority level and the interrupt request type are configured by the MPU_INTC.INTCPS_ILRm register. If more than one incoming interrupt with the same priority level and interrupt request type occur simultaneously, the highest-numbered interrupt is serviced first.

When one or more unmasked incoming interrupts are detected, the INTCPS separates between IRQ and FIQ using the corresponding MPU_INTC.INTCPS_ILRm[0] FIQNIRQ bit. The result is placed in INTCPS_PENDING_IRQn or INTCPS_PENDING_FIQn

If no other interrupts are currently being processed, INTCPS asserts IRQ/FIQ and starts the priority computation. Priority sorting for IRQ and FIQ can execute in parallel.

Each IRQ/FIQ priority sorter determines the highest priority interrupt number. Each priority number is placed in the corresponding MPU_INTC.INTCPS_SIR_IRQ[6:0] ACTIVEIRQ field or MPU_INTC.INTCPS_SIR_FIQ[6:0] ACTIVEFIQ field. The value is preserved until the corresponding MPU_INTC.INTCPS_CONTROL NEWIRQAGR or NEWFIQAGR bit is set.

Once the interrupting peripheral device has been serviced and the incoming interrupt de-asserted, the user must write to the appropriate NEWIRQAGR or NEWFIQAGR bit to indicate to the INTCPS the interrupt has been handled. If there are any pending unmasked incoming interrupts for this interrupt request type, the INTCPS restarts the appropriate priority sorter; otherwise, the IRQ or FIQ interrupt line is de-asserted.

1.4.2 Secure Interrupts (HS Devices Only)

To determine if a HS version of your device is available and for more information on HS devices, please refer to your device-specific data manual.

1.4.3 Register Protection

If the MPU_INTC.INTCPS_PROTECTION[0] PROTECTION bit is set, access to the INTCPS registers is restricted to the supervisor mode. Access to the MPU_INTC.INTCPS_PROTECTION register is always restricted to privileged mode.

1.4.4 Module Power Saving

The INTCPS provides an auto-idle function in its three clock domains:

- Interface clock
- Functional clock
- Synchronizer clock

The interface clock auto-idle power-saving mode is enabled if the MPU_INTC.INTCPS_SYSCONFIG[0] AUTOIDLE bit is set to 1. When this mode is enabled and there is no activity on the bus interface, the interface clock is disabled internally to the module, thus reducing power consumption. When there is new activity on the bus interface, the interface clock restarts without any latency penalty. After reset, this mode is disabled, by default.

The functional clock auto-idle power-saving mode is enabled if the MPU_INTC.INTCPS_IDLE[0] FUNCIDLE bit is set to 0. When this mode is enabled and there is no active interrupt (IRQ or FIQ interrupt being processed or generated) or no pending incoming interrupt, the functional clock is disabled internally to the module, thus reducing power consumption. When a new unmasked incoming interrupt is detected, the functional clock restarts and the INTCPS processes the interrupt. If this mode is disabled, the interrupt latency is reduced by one cycle. After reset, this mode is enabled, by default.

The synchronizer clock allows external asynchronous interrupts to be resynchronized before they are masked. The synchronizer input clock has an auto-idle power-saving mode enabled if the MPU_INTC.INTCPS_IDLE[1] TURBO bit is set to 1. If the auto-idle mode is enabled, the standby power is reduced, but the IRQ or FIQ interrupt latency increases from four to six functional clock cycles. This feature can be enabled dynamically according to the requirements of the device. After reset, this mode is disabled, by default.

To ensure optimal power consumption, INTC_INIT_REGISTER1[0] INIT1 and INTC_INIT_REGISTER2[1] INIT2 bits must be set to 1 during initialization.

1.4.5 Interrupt Latency

The IRQ/FIQ interrupt generation takes four INTCPS functional clock cycles (plus or minus one cycle) if the MPU_INTC.INTCPS_IDLE[1] TURBO bit is set to 0. If the TURBO bit is set to 1, the interrupt generation takes six cycles, but power consumption is reduced while waiting for an interrupt.

These latencies can be reduced by one cycle by disabling functional clock auto-idle (MPU_INTC.INTCPS_IDLE[0] FUNCIDLE bit set to 1), but power consumption is increased, so the benefit is minimal. For information about power saving, see Section 1.4.4.

To minimize interrupt latency when an unmasked interrupt occurs, the IRQ or FIQ interrupt is generated before priority sorting completion. The priority sorting takes 10 functional clock cycles, which is less than the minimum number of cycles required for the MPU to switch to the interrupt context after reception of the IRQ or FIQ event.

Any read of the MPU_INTC.INTCPS_SIR_IRQ or MPU_INTC.INTCPS_SIR_FIQ register during the priority sorting process stalls until priority sorting is complete and the relevant register is updated. However, the delay between the interrupt request being generated and the interrupt service routine being executed is such that priority sorting always completes before the MPU_INTC.INTCPS_SIR_IRQ or MPU_INTC.INTCPS_SIR_FIQ register is read.

1.5 Interrupt Basic Programming Model

1.5.1 Initialization Sequence

1. Program the MPU_INTC.INTCPS_SYSCONFIG register: If necessary, enable the interface clock autogating by setting the AUTOIDLE bit.
2. Program the MPU_INTC.INTCPS_IDLE register: If necessary, disable functional clock autogating or enable synchronizer autogating by setting the FUNCIDLE bit or TURBO bit accordingly.
3. Program the MPU_INTC.INTCPS_ILRm register for each interrupt line: Assign a priority level and set the FIQNFIQ bit for an FIQ interrupt (by default, interrupts are mapped to IRQ and priority is 0x0 [highest]).
4. Program the MPU_INTC.INTCPS_MIRn register: Enable interrupts (by default, all interrupt lines are masked).

Note: To program the MPU_INTC.INTCPS_MIRn register, the MPU_INTC.INTCPS_MIR_SETn and MPU_INTC.INTCPS_MIR_CLEARn registers are provided to facilitate the masking, even if it is possible for backward-compatibility to write directly to the MPU_INTC.INTCPS_MIRn register.

1.5.2 MPU INTC Processing Sequence

After the MPU_INTC.INTCPS_MIRn and MPU_INTC.INTCPS_ILRm registers are configured to enable and assign priorities to incoming interrupts, the interrupt is processed as explained in the following subsections.

IRQ and FIQ processing sequences are quite similar, the differences for the FIQ sequence are shown after a '/' character in **bold** characters in the text or the code below.

1. One or more unmasked incoming interrupts (M_IRQ_n signals) are received and IRQ or FIQ outputs (MPU_INTC_IRQ/**FIQ**) are not currently asserted.
2. If the MPU_INTC.INTCPS_ILRm[0] FIQNIRQ bit is set to 0, the MPU_INTC_IRQ output signal is generated. If the FIQNIRQ bit is set to 1, the MPU_INTC_FIQ output signal is generated.
3. The INTC performs the priority sorting and updates the MPU_INTC.INTCPS_SIR_IRQ[6:0] ACTIVEIRQ /**MPU_INTC.INTCPS_SIR_FIQ[6:0] ACTIVEFIQ** field with the current interrupt number.
4. During priority sorting, if the IRQ/**FIQ** is enabled at the host processor side, the host processor automatically saves the current context and executes the ISR as follows:

Note: The ARM host processor automatically performs the following actions in pseudo code.

```
LR = PC + 4                    /* return link                    */
SPSR = CPSR                    /* Save CPSR before execution     */
CPSR[5] = 0                    /* Execute in ARM state           */
CPSR[7] = 1                    /* Disable IRQ                    */
```

```
CPSR[8] = 1                     /* Disable Imprecise Data Aborts    */
CPSR[9] = CP15_reg1_EEbit       /* Endianness on exception entry    */
if interrupt == IRQ then
    CPSR[4:0] = 0b10010             /* Enter IRQ mode                   */
    if high vectors configured then
        PC = 0xFFFF0018
    else
        PC = 0x00000018             /* execute interrupt vector         */
else if interrupt == FIQ then
    CPSR[4:0] = 0b10001             /* Enter FIQ mode                   */
    CPSR[6] = 1                     /* Disable FIQ                      */
    if high vectors configured then
        PC = 0xFFFF001C
    else
        PC = 0x0000001C             /* execute interrupt vector         */
end if
```

5. The ISR saves the remaining context, identifies the interrupt source by reading the ACTIVEIRQ/**ACTIVEFIQ** field, and jumps to the relevant subroutine handler as follows:

> **CAUTION**
>
> The code in steps 5 and 7 is an assembly code compatible with ARM architecture V6 and V7. This code is developed for the Texas Instruments Code Composer Studio tool set. It is a draft version, only tested on an emulated environment.

```
; INTCPS_SIR_IRQ/INTCPS_SIR_FIQ register address
INTCPS_SIR_IRQ_ADDR/INTCPS_SIR_FIQ_ADDR .word 0x48200040/0x48200044

; ACTIVEIRQ bit field mask to get only the bit field
ACTIVEIRQ_MASK .equ 0x7F

_IRQ_ISR/_FIQ_ISR:

    ; Save the critical context
    STMFD SP!, {R0-R12, LR}             ; Save working registers and the Link register
    MRS R11, SPSR                   ; Save the SPSR into R11

    ; Get the number of the highest priority active IRQ/FIQ
    LDR R10, INTCPS_SIR_IRQ_ADDR/INTCPS_SIR_FIQ_ADDR
    LDR R10, [R10]                  ; Get the INTCPS_SIR_IRQ/INTCPS_SIR_FIQ register
    AND R10, R10, #ACTIVEIRQ_MASK   ; Apply the mask to get the active IRQ number

    ; Jump to relevant subroutine handler
    LDR PC, [PC, R10, lsl #2]        ; PC base address points this instruction + 8
    NOP                             ; To index the table by the PC

    ; Table of handler start addresses
    .word IRQ0handler ;For IRQ0 of BANK0
    .word IRQ1handler
    .word IRQ2handler
```

6. The subroutine handler executes code specific to the peripheral generating the interrupt by handling the event and de-asserting the interrupt condition at the peripheral side.

```
; IRQ0 subroutine
IRQ0handler:

    ; Save working registers
    STMFD SP!, {R0-R1}

    ; Now read-modify-write the peripheral module status register
    ; to de-assert the M_IRQ_0 interrupt signal

    ; De-Assert the peripheral interrupt
```

```
    MOV R0, #0x7                    ; Mask for 3 flags
    LDR R1, MODULE0_STATUS_REG_ADDR ; Get the address of the module Status Register
    STR R0, [R1]                    ; Clear the 3 flags

    ; Restore working registers
    LDMFD SP!, {R0-R1}

    ; Jump to the end part of the ISR
    B IRQ_ISR_end/FIQ_ISR_end
```

7. After the return of the subroutine, the ISR sets the NEWIRQAGR/**NEWFIQAGR** bit to enable the processing of subsequent pending IRQs/**FIQs** and to restore ARM context in the following code. Because the writes are posted on an Interconnect bus, to be sure that the preceding writes are done before enabling IRQs/**FIQs**, a Data Synchronization Barrier is used. This operation ensure that the IRQ/**FIQ** line is de-asserted before IRQ/**FIQ** enabling. After that, the INTC processes any other pending interrupts or de-asserts the MPU_INTC_IRQ/**MPU_INTC_FIQ** signal if there is no interrupt.

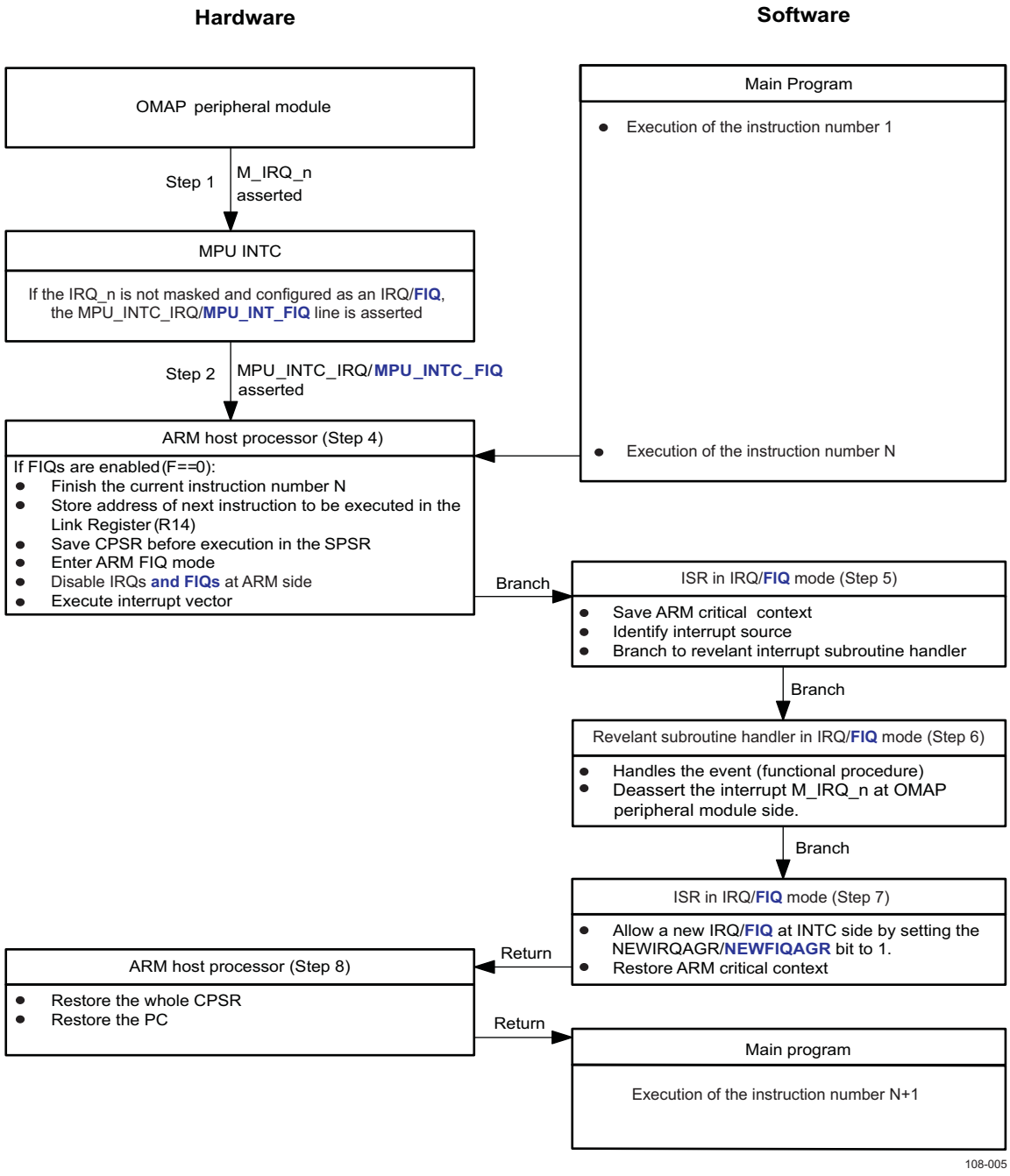
```
; INTCPS_CONTROL register address
INTCPS_CONTROL_ADDR .word 0x48200048

; NEWIRQAGR/NEWFIQAGR bit mask to set only the NEWIRQAGR/NEWFIQAGR bit
NEWIRQAGR_MASK/NEWFIQAGR_MASK .equ 0x01/0x02

IRQ_ISR_end/FIQ_ISR_end:

    ; Allow new IRQs/FIQs at INTC side
    ; The INTCPS_CONTROL register is a write only register so no need to write back others bits
    MOV R0, #NEWIRQAGR_MASK/NEWFIQAGR_MASK ; Get the NEWIRQAGR/NEWFIQAGR bit position
    LDR R1, INTCPS_CONTROL_ADDR
    STR R0, [R1]                    ; Write the NEWIRQAGR/NEWFIQAGR bit to allow new IRQs/FIQs

    ; Data Synchronization Barrier
    MOV R0, #0
    MCR P15, #0, R0, C7, C10, #4

    ; restore critical context
    MSR SPSR, R11                   ; Restore the SPSR from R11
    LDMFD SP!, {R0-R12, LR}         ; Restore working registers and Link register

    ; Return after handling the interrupt
    SUBS PC, LR, #4
```

8. After the ISR return, the ARM automatically restores its context as follows:

```
CPSR = SPSR
PC = LR
```

Figure 1-5 shows the IRQ/**FIQ** processing sequence from the originating device peripheral module to the main program interruption.

Figure 1-5. IRQ/FIQ Processing Sequence



Note: The differences between the IRQ and the FIQ sequence are highlighted in blue and bold characters.

The priority sorting mechanism is frozen during an interrupt processing sequence. If an interrupt condition occurs during this time, the interrupt is not lost. It is sorted when the NEWIRQAGR/**NEWFIQAGR** bit is set (priority sorting is reactivated).

1.5.3 MPU INTC Preemptive Processing Sequence

Preemptive interrupts, also called nested interrupts, can reduce the latencies for higher priority interrupts. A preemptive ISR can be suspended by a higher priority interrupt. Thus, the higher priority interrupt can be served immediately.

Nested interrupts must be used carefully to avoid using corrupted data. Programmers must save corruptible registers and enable IRQ or FIQ at ARM side.

IRQ and FIQ processing sequences are quite similar, the differences for the FIQ sequence are shown after a '/' character in **bold** characters in the text or the code below.

To enable IRQ/**FIQ** preemption by higher priority IRQs/**FIQs**, programers can follow this procedure to write the ISR:

At the beginning of an IRQ/**FIQ** ISR:

1. Save the ARM critical context registers.
2. Save the MPU_INTC.INTCPS_THRESHOLD PRIORITYTHRESHOLD field before modifying it.
3. Read the active interrupt priority in the MPU_INTC.INTCPS_IRQ_PRIORITY IRQPRIORITY / **MPU_INTC.INTCPS_FIQ_PRIORITY FIQPRIORITY** field and write it to the PRIORITYTHRESHOLD(1) field.
4. Read the active interrupt number in the MPU_INTC.INTCPS_SIR_IRQ[6:0] ACTIVEIRQ / **MPU_INTC.INTCPS_SIR_FIQ[6:0] ACTIVEFIQ** field to identify the interrupt source.
5. Write 1 to the appropriate MPU_INTC.INTCPS_CONTROL NEWIRQAGR **and**(2) **NEWFIQAGR** bit while an interrupt is still processing to allow only higher priority interrupts to preempt.
6. Because the writes are posted on an Interconnect bus, to be sure that the preceding writes are done before enabling IRQs/**FIQs**, a Data Synchronization Barrier is used. This operation ensure that the IRQ line is de-asserted before IRQ/**FIQ** enabling.
7. Enable IRQ/**FIQ** at ARM side.
8. Jump to the relevant subroutine handler.

The sample code below shows the previous steps:

CAUTION

The code below is an assembly code compatible with ARM architecture V6 and V7. This code is developed for the Texas Instruments Code Composer Studio tool set. It is a draft version, only tested on an emulated environment.

```
;  bit field mask to get only the bit field
ACTIVEPRIO_MASK .equ 0x3F

_IRQ_ISR:
    ; Step 1 : Save the critical context
    STMFD SP!, {R0-R12, LR}            ; Save working registers
    MRS R11, SPSR                      ; Save the SPSR into R11

    ; Step 2 : Save the INTCPS_THRESHOLD register into R12
    LDR R0, INTCPS_THRESHOLD_ADDR
    LDR R12, [R0]

    ; Step 3 : Get the priority of the highest priority active IRQ
```

(1) The priority-threshold mechanism is enabled automatically when writing a priority in the range of 0x00 to 0x3F while reading it from the IRQPRIORITY and FIQPRIORITY fields. Writing a value of 0xFF (reset default) disables the priority-threshold mechanism. Values between 0x3F and 0xFF must not be used.
When the hardware-priority threshold is in use, the priorities of interrupts selected as FIQ or IRQ become linked; otherwise, they are independent. When they are linked, all FIQ priorities must be set higher than all IRQ priorities to maintain the relative priority of FIQ over IRQ.

(2) When handling FIQs using the priority-threshold mechanism, both NEWFIQAGR and NEWIRQAGR bits must be written at the same time to ensure that the new priority threshold is applied while an IRQ sort is in progress. This IRQ will not have been seen by the ARM, as it will have been masked on entry to the FIQ ISR. However, the source of the IRQ remains active and it is finally processed when the priority threshold falls to a priority sufficiently low to allow it to be processed. The precaution of writing to New FIQ Agreement is not required during an IRQ ISR, as FIQ sorting is not affected (providing all FIQ priorities are higher than all IRQ priorities).

```
    LDR R1, INTCPS_IRQ_PRIORITY_ADDR/INTCPS_FIQ_PRIORITY_ADDR
    LDR R1, [R1]                        ; Get the INTCPS_IRQ_PRIORITY/INTCPS_FIQ_PRIORITY register
    AND R1, R1, #ACTIVEPRIO_MASK        ; Apply the mask to get the priority of the IRQ
    STR R1, [R0]                        ; Write it to the INTCPS_THRESHOLD register

    ; Step 4 : Get the number of the highest priority active IRQ
    LDR R10, INTCPS_SIR_IRQ_ADDR/INTCPS_SIR_FIQ_ADDR
    LDR R10, [R10]                      ; Get the INTCPS_SIR_IRQ/INTCPS_SIR_FIQ register
    AND R10, R10, #ACTIVEIRQ_MASK       ; Apply the mask to get the active IRQ number

    ; Step 5 : Allow new IRQs and FIQs at INTC side
    MOV R0, #0x1/0x3                    ; Get the NEWIRQAGR and NEWFIQAGR bit position
    LDR R1, INTCPS_CONTROL_ADDR
    STR R0, [R1]                        ; Write the NEWIRQAGR and NEWFIQAGR bit

    ; Step 6 : Data Synchronization Barrier
    MOV R0, #0
    MCR P15, #0, R0, C7, C10, #4

    ; Step 7 : Read-modify-write the CPSR to enable IRQs/FIQs at ARM side
    MRS R0, CPSR                        ; Read the status register
    BIC R0, R0, #0x80/0x40              ; Clear the I/F bit
    MSR CPSR, R0                        ; Write it back to enable IRQs

    ; Step 8 : Jump to relevant subroutine handler
    LDR PC, [PC, R10, lsl #2]           ; PC base address points this instruction + 8
    NOP                                 ; To index the table by the PC

    ; Table of handler start addresses
    .word IRQ0handler ;IRQ0 BANK0
    .word IRQ1handler
    .word IRQ2handler
```

After the return of the relevant IRQ/**FIQ** subroutine handle :

1. Disable IRQs/**FIQs** at ARM side.
2. Restore the MPU_INTC.INTCPS_THRESHOLD PRIORITYTHRESHOLD field.
3. Restore the ARM critical context registers.

The sample code below shows the three previous steps:

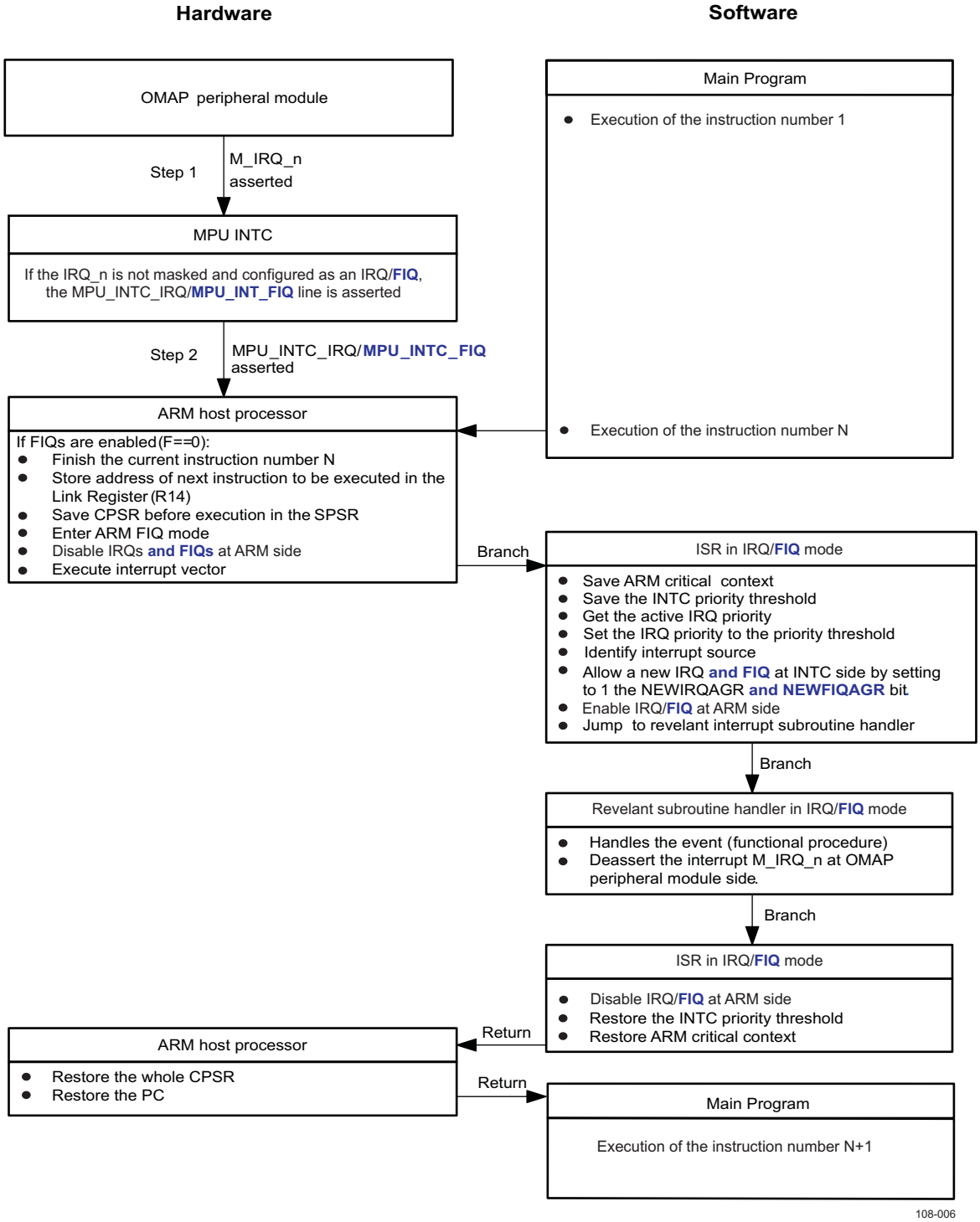
CAUTION

The code below is an assembly code compatible with ARM architecture V6 and V7. This code is developed for the Texas Instruments Code Composer Studio tool set. It is a draft version, only tested on an emulated environment.

```
IRQ_ISR_end:

    ; Step 1 : Read-modify-write the CPSR to disable IRQs/FIQs at ARM side
    MRS R0, CPSR                        ; Read the CPSR
    ORR R0, R0, #0x80/0x40              ; Set the I/F bit
    MSR CPSR, R0                        ; Write it back to disable IRQs

    ; Step 2 : Restore the INTCPS_THRESHOLD register from R12
    LDR R0, INTCPS_THRESHOLD_ADDR
    STR R12, [R0]

    ; Step 3 : Restore critical context
    MSR SPSR, R11                       ; Restore the SPSR from R11
    LDMFD SP!, {R0-R12, LR}             ; Restore working registers and Link register

    ; Return after handling the interrupt
    SUBS PC, LR, #4
```

Figure 1-6 shows the nested IRQ/**FIQ** processing sequence from the originating device peripheral module to the main program interruption.

Figure 1-6. Nested IRQ/FIQ Sequence



Note: The differences between the IRQ and the FIQ sequence are highlighted in blue and bold characters.

1.5.4 MPU INTC Spurious Interrupt Handling

The values in the MPU_INTC.INTCPS_MIRn or MPU_INTC.INTCPS_ILRm registers must not be changed while the corresponding interrupt is asserted. If these registers are changed within the 10-cycle (INTC functional clock cycles) window after the interrupt assertion, only the active interrupt input that triggered the sort could be masked before its turn in the sort. The resulting MPU_INTC.INTCPS_SIR_IRQ, MPU_INTC.INTCPS_SIR_FIQ, MPU_INTC.INTCPS_IRQ_PRIORITY and MPU_INTC.INTCPS_FIQ_PRIORITY register values become invalid.

This condition is detected for both IRQ and FIQ, and the invalid status is flagged across the SIR and PRIORITY registers SPURIOUSIRQFLAG (see Note 1) and SPURIOUSFIQFLAG (see Note 2) fields. A 0 indicates valid and a 1 indicates invalid interrupt number and priority. The invalid indication can be tested in software as a false register value.

Notes:

1. The MPU_INTC.INTCPS_SIR_IRQ[31:7] SPURIOUSIRQFLAG field is a copy of the MPU_INTC.INTCPS_IRQ_PRIORITY[31:7] SPURIOUSIRQFLAG field.
2. The MPU_INTC.INTCPS_SIR_FIQ[31:7] SPURIOUSFIQFLAG field is a copy of the MPU_INTC.INTCPS_FIQ_PRIORITY[31:7] SPURIOUSFIQFLAG field.

1.6 Interrupt Controller Registers

Table 1-5 lists the base address and address space for the INTC instances.

Table 1-5. INTC Instance Summary

Module Name	Base Address	Size
MPU INTC	0x4820 0000	4K bytes

1.6.1 Register Mapping Summary

CAUTION

MPU INTC registers are limited to 32-bit and 16-bit data accesses. 8-bit is not allowed and can corrupt register content.

In Section 1.6.2, each register from MPU_INTC.INTCPS_ITRn to MPU_INTC.INTCPS_PENDING_FIQn contains 32 bits, 1 bit for each interrupt (in ascending order: bit 0 of the MPU_INTC.INTCPS_ITR0 register applies to interrupt line 0; bit 0 of the MPU_INTC.INTCPS_ITR1 register applies to interrupt line 32).

Table 1-6. MPU INTC Register Summary

Register Name	Type	Register Width (Bits)	Address Offset	MPU INTC Physical Address
INTCPS_SYSCONFIG	RW	32	0x0000 0010	0x4820 0010
INTCPS_SYSSTATUS	R	32	0x0000 0014	0x4820 0014
INTCPS_SIR_IRQ	R	32	0x0000 0040	0x4820 0040
INTCPS_SIR_FIQ	R	32	0x0000 0044	0x4820 0044
INTCPS_CONTROL	RW	32	0x0000 0048	0x4820 0048
INTCPS_PROTECTION	RW	32	0x0000 004C	0x4820 004C
INTCPS_IDLE	RW	32	0x0000 0050	0x4820 0050
INTCPS_IRQ_PRIORITY	RW	32	0x0000 0060	0x4820 0060
INTCPS_FIQ_PRIORITY	RW	32	0x0000 0064	0x4820 0064
INTCPS_THRESHOLD	RW	32	0x0000 0068	0x4820 0068
INTCPS_ITRn (1)	R	32	0x0000 0080 + (0x20 * n)	0x4820 0080 + (0x20 * n)
INTCPS_MIRn (1)	RW	32	0x0000 0084 + (0x20 * n)	0x4820 0084 + (0x20 * n)
INTCPS_MIR_CLEARn (1)	W	32	0x0000 0088 + (0x20 * n)	0x4820 0088 + (0x20 * n)
INTCPS_MIR_SETn (1)	W	32	0x0000 008C + (0x20 * n)	0x4820 008C + (0x20 * n)
INTCPS_ISR_SETn (1)	RW	32	0x0000 0090 + (0x20 * n)	0x4820 0090 + (0x20 * n)
INTCPS_ISR_CLEARn (1)	W	32	0x0000 0094 + (0x20 * n)	0x4820 0094 + (0x20 * n)
INTCPS_PENDING_IRQn (1)	R	32	0x0000 0098 + (0x20 * n)	0x4820 0098 + (0x20 * n)
INTCPS_PENDING_FIQn (1)	R	32	0x0000 009C + (0x20 * n)	0x4820 009C + (0x20 * n)
INTCPS_ILRm (2)	RW	32	0x0000 0100 + (0x4 * m)	0x4820 0100 + (0x4 * m)

(1) n = 0 to 2
(2) m = 0 to 95

Table 1-7. Device INTC Initialization Register Summary

Register Name	Type	Register Width (Bits)	Physical Address
INTC_INIT_REGISTER1	RW	32	0x480C 7010
INTC_INIT_REGISTER2	RW	32	0x480C 7050

1.6.2 MPU INTC Register Descriptions

Table 1-8 through Table 1-44 describe the MPU INTC registers.

1.6.2.1 INTCPS_SYSCONFIG

Table 1-8. INTCPS_SYSCONFIG

Address Offset	0x010		
Physical Address	0x4820 0010	**Instance**	MPU INTC
Description	This register controls various parameters of the module interface.		
Type	RW		

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved																														SOFTRESET	AUTOIDLE

Bits	Field Name	Description	Type	Reset
31:2	Reserved	Write 0s for future compatibility. Read returns reset value.	R	0x00000000
1	SOFTRESET	Software reset. Set this bit to trigger a module reset. The bit is automatically reset by the hardware. Read returns 0. Write 0x0: No functional effect Write 0x1: The module is reset.	RW	0
0	AUTOIDLE	Internal interface clock gating strategy 0x0: Interface clock is free-running. 0x1: Automatic interface clock gating strategy is applied, based on the interface bus activity.	RW	0

Table 1-9. Register Call Summary for Register INTCPS_SYSCONFIG

MPU Subsystem INTCPS Integration • Hardware and Software Reset: [0]
Interrupt Controller Functional Description • Module Power Saving: [1]
Basic Programming Model • Initialization Sequence: [3]
Interrupt Controller Registers • Register Summary: [4]

1.6.2.2 INTCPS_SYSSTATUS

Table 1-10. INTCPS_SYSSTATUS

Address Offset	0x014		
Physical Address	0x4820 0014	**Instance**	MPU INTC
Description	This register provides status information about the module.		
Type	R		

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved																															RESETDONE

Bits	Field Name	Description	Type	Reset
31:1	Reserved	Read returns reset value.	R	0x00000000
0	RESETDONE	Internal reset monitoring	R	-
		Read 0x0: Internal module reset is ongoing.		
		Read 0x1: Reset complete		

Table 1-11. Register Call Summary for Register INTCPS_SYSSTATUS

Interrupt Controller Registers
- Register Summary: [0]

1.6.2.3 INTCPS_SIR_IRQ

Table 1-12. INTCPS_SIR_IRQ

Address Offset	0x040		
Physical Address	0x4820 0040	**Instance**	MPU INTC
Description	This register supplies the currently active IRQ interrupt number.		
Type	R		

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
SPURIOUSIRQFLAG																								ACTIVEIRQ							

Bits	Field Name	Description	Type	Reset
31:7	SPURIOUSIRQFLAG	Spurious IRQ flag	R	0x1FFFFFF
6:0	ACTIVEIRQ	Active IRQ number	R	0x00

Table 1-13. Register Call Summary for Register INTCPS_SIR_IRQ

Interrupt Controller Functional Description
- Priority Sorting: [0]
- Interrupt Latency: [1] [2]

Basic Programming Model
- MPU INTC Processing Sequence: [3]
- MPU INTC Preemptive Processing Sequence: [4]
- MPU INTC Spurious Interrupt Handling: [5] [6]

Interrupt Controller Registers
- Register Summary: [7]

1.6.2.4 INTCPS_SIR_FIQ

Table 1-14. INTCPS_SIR_FIQ

Address Offset	0x044		
Physical Address	0x4820 0044	**Instance**	MPU INTC
Description	This register supplies the currently active FIQ interrupt number.		
Type	R		

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
SPURIOUSFIQFLAG																								ACTIVEFIQ							

Bits	Field Name	Description	Type	Reset
31:7	SPURIOUSFIQFLAG	Spurious FIQ flag	R	0x1FFFFFF
6:0	ACTIVEFIQ	Active FIQ number	R	0x00

Table 1-15. Register Call Summary for Register INTCPS_SIR_FIQ

Interrupt Controller Functional Description
- Priority Sorting: [0]
- Interrupt Latency: [1] [2]

Basic Programming Model
- MPU INTC Processing Sequence: [3]
- MPU INTC Preemptive Processing Sequence: [4]
- MPU INTC Spurious Interrupt Handling: [5] [6]

Interrupt Controller Registers
- Register Summary: [7]

1.6.2.5 INTCPS_CONTROL

Table 1-16. INTCPS_CONTROL

Address Offset	0x048		
Physical Address	0x4820 0048	**Instance**	MPU INTC
Description	This register contains the new interrupt agreement bits.		
Type	RW		

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved																														NEWFIQAGR	NEWIRQAGR

Bits	Field Name	Description	Type	Reset
31:2	Reserved	Write 0s for future compatibility. Read returns reset value.	R	0x00000000
1	NEWFIQAGR	Reset FIQ output and enable new FIQ generation.	W	-
		Write 0x0: No functional effect		
		Write 0x1: Reset FIQ output and enable new FIQ generation.		
0	NEWIRQAGR	New IRQ generation	W	-
		Write 0x0: No functional effect		
		Write 0x1: Reset IRQ output and enable new IRQ generation.		

Table 1-17. Register Call Summary for Register INTCPS_CONTROL

Interrupt Controller Functional Description
• Priority Sorting: [0]
Basic Programming Model
• MPU INTC Preemptive Processing Sequence: [1]
Interrupt Controller Registers
• Register Summary: [2]

1.6.2.6 INTCPS_PROTECTION

Table 1-18. INTCPS_PROTECTION

Address Offset	0x04C		
Physical Address	0x4820 004C	**Instance**	MPU INTC
Description	This register controls protection of the other registers. It can be accessed only in supervisor mode, regardless of the current value of the protection bit.		
Type	RW		

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved																															PROTECTION
R-0																															R/W-0

Bits	Field Name	Description	Type	Reset
31:1	Reserved	Write 0s for future compatibility. Read returns reset value.	R	0x00000000
0	PROTECTION	Protection mode 0x0: Protection mode is disabled (default). 0x1: Protection mode is enabled. When enabled, all the MPU INTC registers are accessible only in privileged mode.	RW	0

Table 1-19. Register Call Summary for Register INTCPS_PROTECTION

Interrupt Controller Functional Description
• Register Protection: [0] [1]
Interrupt Controller Registers
• Register Summary: [2]

1.6.2.7 INTCPS_IDLE

Table 1-20. INTCPS_IDLE

Address Offset	0x050		
Physical Address	0x4820 0050	**Instance**	MPU INTC
Description	This register controls the functional clock auto-idle and the synchronizer clock auto-gating.		
Type	RW		

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved																														TURBO	FUNCIDLE
R-0																														R/W-0	R/W-0

Bits	Field Name	Description	Type	Reset
31:2	Reserved	Write 0s for future compatibility. Read returns reset value.	R	0x00000000
1	TURBO	Input synchronizer clock auto-gating	RW	0
		0x0: Input synchronizer clock is free-running (default).		
		0x1: Input synchronizer clock is auto-gated based on interrupt input activity.		
0	FUNCIDLE	Functional clock idle mode	RW	0
		0x0: Functional clock gating strategy is applied (default).		
		0x1: Functional clock is free-running.		

Table 1-21. Register Call Summary for Register INTCPS_IDLE

Interrupt Controller Functional Description • Module Power Saving: [0] [1] • Interrupt Latency: [3] [4]
Basic Programming Model • Initialization Sequence: [5]
Interrupt Controller Registers • Register Summary: [6]

1.6.2.8 INTCPS_IRQ_PRIORITY

Table 1-22. INTCPS_IRQ_PRIORITY

Address Offset	0x060		
Physical Address	0x4820 0060	**Instance**	MPU INTC
Description	This register supplies the currently active IRQ priority level.		
Type	R		

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
SPURIOUSIRQFLAG																								IRQPRIORITY							

Bits	Field Name	Description	Type	Reset
31:6	SPURIOUSIRQFLAG	Spurious IRQ flag	R	0x3FFFFFF
5:0	IRQPRIORITY	Current IRQ priority	R	0x00

Table 1-23. Register Call Summary for Register INTCPS_IRQ_PRIORITY

Basic Programming Model • MPU INTC Preemptive Processing Sequence: [0] • MPU INTC Spurious Interrupt Handling: [1] [2]
Interrupt Controller Registers • Register Summary: [3]

1.6.2.9 INTCPS_FIQ_PRIORITY

Table 1-24. INTCPS_FIQ_PRIORITY

Address Offset	0x064		
Physical Address	0x4820 0064	**Instance**	MPU INTC
Description	This register supplies the currently active FIQ priority level.		
Type	R		

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
SPURIOUSFIQFLAG																								FIQPRIORITY							

Bits	Field Name	Description	Type	Reset
31:6	SPURIOUSFIQFLAG	Spurious FIQ flag	R	0x3FFFFFF
5:0	FIQPRIORITY	Current FIQ priority	R	0x00

Table 1-25. Register Call Summary for Register INTCPS_FIQ_PRIORITY

Basic Programming Model
- MPU INTC Preemptive Processing Sequence: [0]
- MPU INTC Spurious Interrupt Handling: [1] [2]

Interrupt Controller Registers
- Register Summary: [3]

1.6.2.10 INTCPS_THRESHOLD

Table 1-26. INTCPS_THRESHOLD

Address Offset	0x068		
Physical Address	0x4820 0068	**Instance**	MPU INTC
Description	This register sets the priority threshold.		
Type	RW		

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved																								PRIORITYTHRESHOLD							

Bits	Field Name	Description	Type	Reset
31:8	Reserved	Write 0s for future compatibility. Read returns reset value.	R	0x000000
7:0	PRIORITYTHRESHOLD	Priority threshold	RW	0xFF
		Write 0xFF: Priority threshold disabled		
		Write 0x0 to 0x3F: Priority threshold enabled		

Table 1-27. Register Call Summary for Register INTCPS_THRESHOLD

Interrupt Controller Functional Description
- Masking: [0]

Basic Programming Model
- MPU INTC Preemptive Processing Sequence: [1] [2]

Interrupt Controller Registers
- Register Summary: [3]

1.6.2.11 INTCPS_ITRn

Table 1-28. INTCPS_ITRn

Address Offset	0x080 + (0x20 * n)	**Index**	n = 0 to 2
Physical Address	0x4820 0080 + (0x20 * n)	**Instance**	MPU INTC
Description	This register shows the raw interrupt input status before masking.		
Type	R		

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
ITR																															

Bits	Field Name	Description	Type	Reset
31:0	ITR	Interrupt status before masking	R	Depends on interrupt inputs

Table 1-29. Register Call Summary for Register INTCPS_ITRn

Interrupt Controller Functional Description • Masking: [0]
Interrupt Controller Registers • Register Summary: [1] [2]

1.6.2.12 INTCPS_MIRn

Table 1-30. INTCPS_MIRn

Address Offset	0x084 + (0x20 * n)	**Index**	n = 0 to 2
Physical Address	0x4820 0084 + (0x20 * n)	**Instance**	MPU INTC
Description	This register contains the interrupt mask.		
Type	RW		

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
MIR																															

Bits	Field Name	Description	Type	Reset
31:0	MIR	Interrupt mask 0x1: The interrupt is masked 0x0: The interrupt is unmasked	RW	0xFFFFFFFF

Table 1-31. Register Call Summary for Register INTCPS_MIRn

Interrupt Controller Functional Description • Masking: [0]
Basic Programming Model • Initialization Sequence: [1] [2] [3] • MPU INTC Processing Sequence: [4] • MPU INTC Spurious Interrupt Handling: [5]
Interrupt Controller Registers • Register Summary: [6]

1.6.2.13 INTCPS_MIR_CLEARn

Table 1-32. INTCPS_MIR_CLEARn

Address Offset	0x088 + (0x20 * n)	**Index**	n = 0 to 2
Physical Address	0x4820 0088 + (0x20 * n)	**Instance**	MPU INTC
Description	This register is used to clear the interrupt mask bits.		
Type	W		

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
MIRCLEAR																															

Bits	Field Name	Description	Type	Reset
31:0	MIRCLEAR	Clear the interrupt mask bits. Read returns 0. Write 0x1: Clears the MIR mask bit to 0 Write 0x0: No functional effect	W	0x00000000

Table 1-33. Register Call Summary for Register INTCPS_MIR_CLEARn

Basic Programming Model • Initialization Sequence: [0]
Interrupt Controller Registers • Register Summary: [1]

1.6.2.14 INTCPS_MIR_SETn

Table 1-34. INTCPS_MIR_SETn

Address Offset	0x08C + (0x20 * n)	**Index**	n = 0 to 2
Physical Address	0x4820 008C + (0x20 * n)	**Instance**	MPU INTC
Description	This register is used to set the interrupt mask bits.		
Type	W		

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
MIRSET																															

Bits	Field Name	Description	Type	Reset
31:0	MIRSET	Mask the interrupt bits. Read returns 0. Write 0x0: No functional effect Write 0x1: Sets the MIR mask bit to 1.	W	0x00000000

Table 1-35. Register Call Summary for Register INTCPS_MIR_SETn

Basic Programming Model • Initialization Sequence: [0]
Interrupt Controller Registers • Register Summary: [1]

1.6.2.15 INTCPS_ISR_SETn

Table 1-36. INTCPS_ISR_SETn

Address Offset	0x090 + (0x20 * n)	**Index**	n = 0 to 2
Physical Address	0x4820 0090 + (0x20 * n)	**Instance**	MPU INTC
Description	This register is used to set the software interrupt bits. It is also used to read the currently active software interrupts.		
Type	RW		

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
ISRSET																															

Bits	Field Name	Description	Type	Reset
31:0	ISRSET	Set the software interrupt bits. Read returns the currently active software interrupts.	RW	0x00000000
		Write 0x0: No functional effect		
		Write 0x1: Sets the software interrupt bits to 1.		

Table 1-37. Register Call Summary for Register INTCPS_ISR_SETn

Interrupt Controller Functional Description • Input Selection: [0]
Interrupt Controller Registers • Register Summary: [1]

1.6.2.16 INTCPS_ISR_CLEARn

Table 1-38. INTCPS_ISR_CLEARn

Address Offset	0x094 + (0x20 * n)	**Index**	n = 0 to 2
Physical Address	0x4820 0094 + (0x20 * n)	**Instance**	MPU INTC
Description	This register is used to clear the software interrupt bits.		
Type	W		

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
ISRCLEAR																															

Bits	Field Name	Description	Type	Reset
31:0	ISRCLEAR	Clear the software interrupt bits. Read returns 0.	W	0x00000000
		Write 0x0: No functional effect		
		Write 0x1: Clears the software interrupt bits to 0.		

Table 1-39. Register Call Summary for Register INTCPS_ISR_CLEARn

Interrupt Controller Functional Description • Input Selection: [0]
Interrupt Controller Registers • Register Summary: [1]

1.6.2.17 INTCPS_PENDING_IRQn

Table 1-40. INTCPS_PENDING_IRQn

Address Offset	0x098 + (0x20 * n)	**Index**	n = 0 to 2
Physical Address	0x4820 0098 + (0x20 * n)	**Instance**	MPU INTC
Description	This register contains the IRQ status after masking.		
Type	R		

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
PENDINGIRQ																															

Bits	Field Name	Description	Type	Reset
31:0	PENDINGIRQ	IRQ status after masking.	R	0x00000000

Table 1-41. Register Call Summary for Register INTCPS_PENDING_IRQn

Interrupt Controller Functional Description
- Masking: [0]
- Priority Sorting: [1]

Interrupt Controller Registers
- Register Summary: [2]

1.6.2.18 INTCPS_PENDING_FIQn

Table 1-42. INTCPS_PENDING_FIQn

Address Offset	0x09C + (0x20 * n)	**Index**	n = 0 to 2
Physical Address	0x4820 009C + (0x20 * n)	**Instance**	MPU INTC
Description	This register contains the FIQ status after masking.		
Type	R		

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
PENDINGFIQ																															

Bits	Field Name	Description	Type	Reset
31:0	PENDINGFIQ	FIQ status after masking.	R	0x00000000

Table 1-43. Register Call Summary for Register INTCPS_PENDING_FIQn

Interrupt Controller Functional Description
- Masking: [0]
- Priority Sorting: [1]

Interrupt Controller Registers
- Register Summary: [2] [3]

1.6.2.19 INTCPS_ILRm

Table 1-44. INTCPS_ILRm

Address Offset	0x100 + (0x4 * m)	**Index**	m = 0 to 95
Physical Address	0x4820 0100 + (0x4 * m)	**Instance**	MPU INTC
Description	These registers contain the priority for the interrupts and the FIQ/IRQ steering.		
Type	RW		

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved																								PRIORITY						Reserved	FIQNIRQ

Bits	Field Name	Description	Type	Reset
31:8	Reserved	Write 0s for future compatibility. Read returns reset value.	R	0x000000
7:2	PRIORITY	Interrupt priority	RW	0x00
1	Reserved	Write 0 for future compatibility. Read returns reset value.	R	0
0	FIQNIRQ	Interrupt IRQ FIQ mapping. Read returns reset value. Write 0x0: Interrupt is routed to IRQ. Write 0x1: Interrupt is routed to FIQ.	RW	0

Table 1-45. Register Call Summary for Register INTCPS_ILRm

Interrupt Controller Functional Description
- Masking: [0]
- Priority Sorting: [1] [2]

Basic Programming Model
- Initialization Sequence: [3]
- MPU INTC Processing Sequence: [4] [5]
- MPU INTC Spurious Interrupt Handling: [6]

Interrupt Controller Registers
- Register Summary: [7]

1.6.3 Device INTC Initialization Register Descriptions

Table 1-46 and Table 1-48 describe device INTC registers that need to be programmed during initialization to ensure optimal power savings.

Table 1-46. INTC_INIT_REGISTER1

Physical Address	0x470C 8010	**Instance**	Device INTC Initialization
Description	This register enables power optimizations.		
Type	RW		

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved																															INIT1

Bits	Field Name	Description	Type	Reset
31:1	Reserved	Write 0s for future compatibility. Read returns reset value.	R	0x00000000
0	INIT1	To ensure lowest power configuration, this bit must be set to 1 during initialization.	RW	0

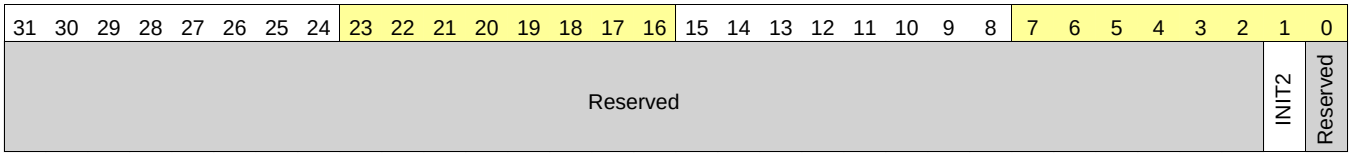
Table 1-47. Register Call Summary for Register INTC_INIT_REGISTER1

Interrupt Controller Registers

- Register Summary: [0]

Table 1-48. INTC_INIT_REGISTER2

Physical Address	0x470C 8050	**Instance**	Device INTC Initialization
Description	This register enables power optimizations.		
Type	RW		



Bits	Field Name	Description	Type	Reset
31:2	Reserved	Write 0s for future compatibility. Read returns reset value.	R	0x00000000
1	INIT2	For optimal power consumption, this bit must be set to 1 during initialization.	RW	0
0	Reserved	For optimal power consumption keep default value of 0 for this bit.	R	0

Table 1-49. Register Call Summary for Register INTC_INIT_REGISTER2

Interrupt Controller Registers

- Register Summary: [0]

IMPORTANT NOTICE

Texas Instruments Incorporated and its subsidiaries (TI) reserve the right to make corrections, modifications, enhancements, improvements, and other changes to its products and services at any time and to discontinue any product or service without notice. Customers should obtain the latest relevant information before placing orders and should verify that such information is current and complete. All products are sold subject to TI's terms and conditions of sale supplied at the time of order acknowledgment.

TI warrants performance of its hardware products to the specifications applicable at the time of sale in accordance with TI's standard warranty. Testing and other quality control techniques are used to the extent TI deems necessary to support this warranty. Except where mandated by government requirements, testing of all parameters of each product is not necessarily performed.

TI assumes no liability for applications assistance or customer product design. Customers are responsible for their products and applications using TI components. To minimize the risks associated with customer products and applications, customers should provide adequate design and operating safeguards.

TI does not warrant or represent that any license, either express or implied, is granted under any TI patent right, copyright, mask work right, or other TI intellectual property right relating to any combination, machine, or process in which TI products or services are used. Information published by TI regarding third-party products or services does not constitute a license from TI to use such products or services or a warranty or endorsement thereof. Use of such information may require a license from a third party under the patents or other intellectual property of the third party, or a license from TI under the patents or other intellectual property of TI.

Reproduction of TI information in TI data books or data sheets is permissible only if reproduction is without alteration and is accompanied by all associated warranties, conditions, limitations, and notices. Reproduction of this information with alteration is an unfair and deceptive business practice. TI is not responsible or liable for such altered documentation. Information of third parties may be subject to additional restrictions.

Resale of TI products or services with statements different from or beyond the parameters stated by TI for that product or service voids all express and any implied warranties for the associated TI product or service and is an unfair and deceptive business practice. TI is not responsible or liable for any such statements.

TI products are not authorized for use in safety-critical applications (such as life support) where a failure of the TI product would reasonably be expected to cause severe personal injury or death, unless officers of the parties have executed an agreement specifically governing such use. Buyers represent that they have all necessary expertise in the safety and regulatory ramifications of their applications, and acknowledge and agree that they are solely responsible for all legal, regulatory and safety-related requirements concerning their products and any use of TI products in such safety-critical applications, notwithstanding any applications-related information or support that may be provided by TI. Further, Buyers must fully indemnify TI and its representatives against any damages arising out of the use of TI products in such safety-critical applications.

TI products are neither designed nor intended for use in military/aerospace applications or environments unless the TI products are specifically designated by TI as military-grade or "enhanced plastic." Only products designated by TI as military-grade meet military specifications. Buyers acknowledge and agree that any such use of TI products which TI has not designated as military-grade is solely at the Buyer's risk, and that they are solely responsible for compliance with all legal and regulatory requirements in connection with such use.

TI products are neither designed nor intended for use in automotive applications or environments unless the specific TI products are designated by TI as compliant with ISO/TS 16949 requirements. Buyers acknowledge and agree that, if they use any non-designated products in automotive applications, TI will not be responsible for any failure to meet such requirements.

Following are URLs where you can obtain information on other Texas Instruments products and application solutions:

Products		**Applications**	
Amplifiers	amplifier.ti.com	Audio	www.ti.com/audio
Data Converters	dataconverter.ti.com	Automotive	www.ti.com/automotive
DSP	dsp.ti.com	Broadband	www.ti.com/broadband
Clocks and Timers	www.ti.com/clocks	Digital Control	www.ti.com/digitalcontrol
Interface	interface.ti.com	Medical	www.ti.com/medical
Logic	logic.ti.com	Military	www.ti.com/military
Power Mgmt	power.ti.com	Optical Networking	www.ti.com/opticalnetwork
Microcontrollers	microcontroller.ti.com	Security	www.ti.com/security
RFID	www.ti-rfid.com	Telephony	www.ti.com/telephony
RF/IF and ZigBee® Solutions	www.ti.com/lprf	Video & Imaging	www.ti.com/video
		Wireless	www.ti.com/wireless

Mailing Address: Texas Instruments, Post Office Box 655303, Dallas, Texas 75265