**Chapter 7: Interrupts**

**7.1 Environment . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 224**

7.1.1 Private, Shared and Software Interrupts . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . .225

7.1.2 Generic Interrupt Controller (GIC). . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . .225

7.1.3 Resets and Clocks . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . .225

7.1.4 Block Diagram . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . .225

7.1.5 CPU Interrupt Signal Pass-through . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . .226

**7.2 Functional Description . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 227**

7.2.1 Software Generated Interrupts (SGI) . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . .227

7.2.2 CPU Private Peripheral Interrupts (PPI) . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . .228

7.2.3 Shared Peripheral Interrupts (SPI) . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . .228

7.2.4 Interrupt Sensitivity, Targeting and Handling. . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . .2307.2.5 Wait for Interrupt Event Signal (WFI) . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . .232

**7.3 Register Overview. . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 232**

7.3.1 Write Protection Lock . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . .233

**7.4 Programming Model. . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 234**

7.4.1 Interrupt Prioritization . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . .234

7.4.2 Interrupt Handling . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . .234

7.4.3 ARM Programming Topics . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . .235

7.4.4 Legacy Interrupts and Security Extensions . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . .235

**7.1 Environment**

This chapter describes the system-level interrupt environment and the functions of the interrupt

controller (see Figure 7-1).

이 장에서는 시스템-레벨 인터럽트 환경과 인터럽트 제어장치의 기능에 대해 설명

The PS is based on ARM architecture, utilizing one or two Cortex-A9 processors (CPUs) and the GIC

pl390 interrupt controller

PS는 ARM아키텍쳐 기반으로 하며 이것은 하나 또는 두 개의 Cortex-A9프로세스와 the GIC pl390인터럽트 제어장치를 활용한다.

Note that single-core devices contain one Cortex-A9 processor (CPU), dual-core devices contain two.

단일코어장치에는 Cortex-A9프로세서 한 개, 듀얼 코얼 장치에는 Cortex-A9프로세서 두 개.

This chapter discusses the dual-core configuration.

이 장에는 듀얼 코어 구성을 토의한다.

The interrupt structure is closely associated with the CPU(s) and accepts interrupts from the I/O

peripherals (IOP) and the programmable logic (PL).

인터럽트구조는 CPU와 자세히 연관되어 있고 주변장치 I/O(IOP)와 프로그램 작동이 가능한 로직(PL)으로부터 인터럽트도 연관되어있다.

This chapter includes these key topics:

이 장의 중요한 주제

• Private, shared and software interrupts (개인,공유 및 소프트웨어 인터럽트)

• GIC functionality (GIC기능)->GIC는 Zynq안에 인터럽트 제어장치

• Interrupt prioritization and handling (인터럽트 우선순위 지정 및 처리)

**7.1.1 Private, Shared and Software Interrupts**

Each CPU has a set of private peripheral interrupts (PPIs) with private access using banked registers.

각 CPU에는 뱅킹된 레지스터를 사용하는 개인 액세스가 있는 개인 주변 장치 인터럽트(PPI) 세트가 있습니다.

The PPIs include the global timer, private watchdog timer, private timer, and FIQ/IRQ from the PL.

개인 주변 장치 인터럽트(PPI)은 global timer, watchdog timer(컴퓨터 내에 있는 중앙 처리 장치(CPU)의 고장을 검출하는 타이머이다), private timer 및 PL의 FIQ/IRQ가 포함 됩니다.

Software generated interrupts (SGIs) are routed to one or both CPUs.

인터럽트 생성된 소프트웨어(SGI)는 하나 또는 양쪽 CPU로 라우팅 된다.

The SGIs are generated by writing to the registers in the generic interrupt controller (GIC), refer to section 7.3

인터럽트 생성된 소프트웨어(SGI)는 범용 인터럽트 컨트롤러(GIC) 안 레지스터에 쓰는 것에 의해 생성된다.(7.3 절 참조)

The shared peripheral interrupts (SPIs) are generated by the various I/O and memory controllers in the PS and PL.

공유 주변 장치 인터럽트(SPI)은 PS와 PL안에 다양한 I/O와 메모리 제어장치에 의해 생성된다.

They are routed to either or both CPUs.

공유 주변 장치 인터럽트(SPI)은 하나 또는 양쪽 CPU로 라우팅 된다.

The SPI interrupts from the PS peripherals are also routed to the PL.

공유 주변 장치 인터럽트(SPI)가 인터럽트 한다. 어떻게? PS주변장치가 PL에 라우팅되서~

**7.1.2 Generic Interrupt Controller (GIC)**

The generic interrupt controller (GIC) is a centralized resource for managing interrupts sent to the

CPUs from the PS and PL.

범용 인터럽트 제어장치(GIC)는 PS및 PL의 CPU(중앙처리장치)에 보내지는 인터럽트 들을 관리 하기 위한 중앙 집중화 된 리소스이다

The controller enables, disables, masks, and prioritizes the interrupt sources and sends them to the selected CPU (or CPUs) in a programmed manner as the CPU interface accepts the next interrupt.

제어장치는 인터럽트 소스에 활성화, 비활성화, 마스크 및 우선순위를 지정한다. 제어장치는 CPU 인터페이스가 다음 인터럽트를 허용할 때 프로그래밍 방식 안에서 인터럽트 소스를 선택하여 선택한 CPU로 보낸다.

In addition, the controller supports security extension for implementing a security-aware system.

또한 제어장치는 보안 인식 시스템을 위해 보안 확장을 지원한다.

The controller is based on the ARM Generic Interrupt Controller Architecture version 1.0 (GIC v1), non-vectored.

제어장치는 ARM 일반 인터럽트 제어장치 아키텍쳐 버전 1.0(GIC v1)과 비 벡터화 된 기반이다.

The registers are accessed via the CPU private bus for fast read/write response by avoiding temporary blockage or other bottlenecks in the interconnect.

레지스터는 빠른 읽기/쓰기 응답을 위해 CPU 개인 버스를 통해 접근한다. 왜 빠른 읽기/쓰기? 상호연결에서 다른 병목현상과 일시적인 막힘을 회피하기 위해

The interrupt distributor centralizes all interrupt sources before dispatching the one with the highest

priority to the individual CPUs.

인터럽트 분배기는 개별CPU에서 우선순위가 가장 높은 인터럽트를 처분하기 전에 모든 인터럽트 소스를 중앙 집중화한다.

The GIC ensures that an interrupt targeted to several CPUs can only be taken by one CPU at a time.

여러 CPU에 인터럽트 대상을 보장하는 GIC는 한번에 하나의 CPU에 의해 수행된다.

All interrupt sources are identified by a unique interrupt ID number.

모든 인터럽트 소스는 고유의 인터럽트 ID로 식별된다.

All interrupt sources have their own configurable priority and list of targeted CPUs.

모든 인터럽트 소스는 자체적 구성 가능한 우선순위와 대상CPU의 목록을 가지고 있다.

**7.1.3 Resets and Clocks**

The interrupt controller is reset by the reset subsystem by writing to the PERI\_RST bit of the A9\_CPU\_RST\_CTRL register in the SLCR.

인터럽트 제어장치는 SLCR안에 A9\_CPU\_RST\_CTRL레지스터의 PERI\_RST 비트에 기록하므로 리셋 서브 시스템 의해 리셋 된다.

The same reset signal also resets the CPU private timers and private watchdog timers (AWDT).

동일한 리셋 신호 또한 CPU private timers 와 private watchdog timers (AWDT)를 리셋한다.

Upon reset, all interrupts that are pending or being serviced are ignored.

리셋시(재설정시), 보류중이거나 서비스중이 모든 인터럽트는 무시된다.

The interrupt controller operates with the CPU\_3x2x clock (half the CPU frequency).

인터럽트 컨트롤러는 CPU\_3x2x클럭(CPU 주파수 절반)으로 작동한다.

**7.1.4 Block Diagram**

The shared peripheral interrupts are generated from various subsystems that include the I/O peripherals in the PS and logic in the PL.

PS의 I/O 주변 장치와 PL의 로직을 포함하는 서브시스템으로부터 공유 주변 장치 인터럽트는 생성된다.

The interrupt sources are illustrated in Figure 7-2.

인터럽트 소스는 그림 7-2에 설명되어있다.

**7.1.5 CPU Interrupt Signal Pass-through**

The IRQ/FIQ from the PL can be routed through the GIC as PPI#4 and #1, or bypass the GIC using the pass-through multiplexer shown in Figure 7-3.

PL의 IRQ/FIQ는 GIC를 통해 PPI#4와#1로 라우팅 되거나 pass-through 멀티플렉서를 사용하여 GIC를 우회 할 수 있다. 그림 7-3이 보여준다.

This logic is instantiated for both CPUs.

이 로직은 양쪽의 CPU에 대해 인스턴스화됩니다.

The pass-through mode is enabled through the mpcore.ICCICR register, according to Table 7-1.

이 pass-through 모드는 mpcoreICCICR레지스터를 통해 활성화 된다. 따라서 7-1 표를 봐라

**7.2 Functional Description**

**7.2.1 Software Generated Interrupts (SGI)**

Each CPU can interrupt itself, the other CPU, or both CPUs using a software generated interrupt (SGI).

각각의 CPU는 인터럽트 생성된 소프트웨어를 사용하여 자신이나 다른 CPU또는 양쪽 CPU를 인터럽트 할 수 있다.

There are 16 software generated interrupts (see Table 7-2).

16개 인터럽트 생성된 소프트웨어가 있다.( 표 7-2봐라)

An SGI is generated by writing the SGI interrupt number to the ICDSGIR register and specifying the target CPU(s).

인터럽트 번호를 ICDSGIR레지스터에 기록하고 대상 CPU를 지정하면 SGI가 생성된다.

This write occurs via the CPU's own private bus.

이 기록은 CPU의 개인 버스를 통해 발생한다.

Each CPU has its own set of SGI registers to generate one or more of the 16 software generated interrupts.

각 CPU는 16개의 생성된 인터럽트 소프트웨어 중 하나 이상을 생성하는 고유한 SGI레지스터 세트가 있다.

The interrupts are cleared by reading the ICCIAR (Interrupt Acknowledge) register or writing a 1 to the corresponding bits of the ICDICPR (Interrupt Clear-Pending) register.

ICCIAR레지스터를 읽거나 ICDICPR레지스터의 해당되는 비트에 1을 쓰면 인터럽트가 클리어된다.

All SGIs are edge triggered.

모든 SGI는 에지 트리거된다.

The sensitivity types for SGIs are fixed and cannot be changed; the ICDICFR0 register is read-only, since it specifies the sensitivity types of all the 16 SGIs.

SGI의 민감도 유형은 고정되어있으며 변경 할 수 없다. ICDICFR0레지스터는 모든 16개 SGI의 민감도 유형을 지정하기 때문에 읽기 전용이다.

**7.2.2 CPU Private Peripheral Interrupts (PPI)**

Each CPU connects to a private set of five peripheral interrupts. The PPIs are listed in Table 7-3.

각 CPU는 5개 주변장치 인터럽트의 개인 세트에 연결된다. PPI은 표 7-3에 나열되어 있다.

The sensitivity types for PPIs are fixed and cannot be changed; therefore, the ICDICFR1 register is read-only, since it specifies the sensitivity types of all the 5 PPIs

PPI의 민감도 유현은 고정되어있으며 변경 할 수 없다. 따라서 ICDICFR1레지스터는 모든 5개PPL의 민감도 유형이 지정되어 있기 때문에 읽기 전용이다.

Note that the fast interrupt (FIQ) signal and the interrupt (IRQ) signal from the PL are inverted and then sent to the interrupt controller.

PL에 인터럽트 신호와 빠른 인터럽트 신호는 반전되고 인터럽트 제어장치에 보내지게 된다.

Therefore, they are active High at the PS-PL interface, although the ICDICFR1 register reflects them as active Low level.

따라서 ICDICFR1레지스터 액티브가 로우 레벨로 반영되더라도 PPI은 PS-PL인터페이스에서 액티브 하이 이다.

**7.2.3 Shared Peripheral Interrupts (SPI)**

A group of approximately 60 interrupts from various modules can be routed to one or both of the CPUs or the PL.

다양한 모듈로부터 약 60개 인터럽트 그룹이 PL나 CPU중 하나 또는 둘 다 라우팅 될 수 있다.

The interrupt controller manages the prioritization and reception of these interrupts for the CPUs

인터럽트 컨트롤러는 CPU에 대한 이러한 인터럽트의 수신과 우선순위를 관리한다.

Except for IRQ #61 through #68 and #84 through #91, all interrupt sensitivity types are fixed by the requesting sources and cannot be changed. The GIC must be programmed to accommodate this. The boot ROM does not program these registers; therefore the SDK device drivers must program the GIC to accommodate these sensitivity types.

IRQ #61~#68 및 #84~#91을 제외하고 모든 인터럽트 민감도 유형은 요청하는 소스에 의해 고정되고 변경 할 수 없다. GIC는 이를 수용하도록 프로그래밍 되야 한다. 부트 ROM은 이 레지스터를 프로그램 하지 않습니다. SDK 장치 드라이버는 이러한 민감도 유형을 수용 할 수 있도록 GIC를 프로그래밍해야 한다.

For an interrupt of level sensitivity type, the requesting source must provide a mechanism for the

interrupt handler to clear the interrupt after the interrupt has been acknowledged.

레벨 민감도 유형의 인터럽트 경우, 요청 소스는 인터럽트가 확인된 후 인터럽트 처리기가 인터럽트를 지우는 메커니즘을 제공 해야합니다.

This requirement applies to any IRQF2P[n] (from PL) with a high level sensitivity type.

이 요구사항은 민감도가 높은 유형의 IRQF2P[n](PL로부터)에 적용된다.

For an interrupt of rising edge sensitivity, the requesting source must provide a pulse wide enough for the GIC to catch.

상승 엣지 민감도의 인터럽트의 경우, 요청 소스는 GIC가 포착 할 수 있을 만큼 충분히 넣ㅂ은 펄스를 제공해야한다.

The ICDICFR2 through ICDICFR5 registers configure the interrupt types of all the SPIs.

ICDICFR2~ICDICFR5레지스터는 모든 SPI의 인터럽트 유형을 구성한다.

Each interrupt has a 2-bit field, which specifies sensitivity type and handling model.

각 인터럽트 민감도 유형과 처리 모델을 지정하는 2비트 필드를 가진다.

The SPI interrupts are listed in Table 7-4.

SPI인터럽트는 표7-4에 나열되어 있다.

**7.2.4 Interrupt Sensitivity, Targeting and Handling**

There are three types of interrupts that come into the GIC as explained in section : SPI, PPI and SGI.

GIC에는 세가지 유형의 인터럽트가 있다. SPI, PPI, SGI섹션에서 설명한다.

In a general sense, the interrupt signals includes a sensitivity setting, whether one or both CPUs handle the interrupt, and which CPU or CPUs are targeted: zero, one, or both.

일반적인 의미에서 인터럽트 신호는 민감도 설정이 포함되며, 하나 또는 모두 CPU가 인터럽트를 처리하는지 여부, 대상 CPU가 0또는1 둘 다 인지 여부가 포함된다

However, the functionality of most interrupt signals include fixed settings, while others are partially programmable.

그러나 대부분의 인터럽트 신호의 기능에는 고정 된 설정이 포함되지만 나머지는 부분적으로 프로그래밍 가능하다.

There are two sets of control registers for sensitivity, handling, and targeting:

민감도, 처리 및 대상 지정을 위한 두세트의 제어 레지스터가 있다.

• mpcore.ICDICFR[5:0] registers: sensitivity and handling. See Figure 7-4. -> 민감도,처리

• mpcore.ICDIPTR[23:0] registers: targeting CPU(s). See Figure 7-5. -> 대상지정

**Shared Peripheral Interrupts (SPI)**

The SPI interrupts can be targeted to any number of CPUs, but only one CPU handles the interrupt.

SPI인터럽트는 여러 CPU를 대상으로 할 수 있지만 하나의 CPU만 인터럽트를 처리합니다.

If an interrupt is targeted to both CPUs and they respond to the GIC at the same time, the MPcore ensures that only one of the CPUs reads the active interrupt ID#.

만약 인터럽트가 두 CPU를 대상으로 하고 동시에 GIC에 응답하는 경우 MPcoreCPU중 하나만 활성 인터럽트 ID#를 읽도록 한다.

The other CPU receives the Spurious ID# 1023 interrupt or the next pending interrupt, depending on the timing.

다른 CPU는 타이밍에 따라 Spurious(거짓스럼)한 ID #1023인터럽트 또는 다음 보류중인 인터럽트를 수신한다.

This removes the requirement for a lock in the interrupt service routine. Targeting the CPU is done by the ICDIPTR [23:8] registers.

이렇게 하면 인터럽트 서비스 루틴에 대한 잠금 요구 사항이 제거된다. CPU를 대상으로 하는 작업은 ICDIPTR[23:8]레지스터에 의해 수행된다.

The sensitivity of each SPI interrupt must be programmed to match those listed in Table 7-4, PS and PL Shared Peripheral Interrupts (SPI). The sensitivity is programmed using the ICDICFR [5:2] registers.

각 SPI 인터럽트의 민감도는 표7-4, PS및PL 공유 주변장치 인터럽트(SPI)에 나열된 것과 일치하도록 프로그래밍 해야한다. 민감도는 ICDICFR[5:2]레지스터를 사용하여 프로그래밍된다.

**Private Peripheral Interrupts (PPI)**

Each CPU has its own separate PPI interrupts with fixed functionality; the sensitivity, handling, and targeting of these interrupts are not programmable.

각 CPU에는 고정 기능이 있는 별도의 PPI 인터럽트가 있다. 이러한 인터럽트 민감도, 처리 및 대상 지정은 프로그래밍 할 수 없다.

Each interrupt only goes to its own CPU and is handled by that CPU.

각 인터럽트는 CPU로만 이동하며 해당 CPU가 처리한다.

The ICDICFR [1] register is read-only and the ICDIPTR [5:2] registers are essentially reserved.

ICDICFR[1]레지스터는 읽기 전용이며 ICDIPTR[5:2]레지스터는 기본적으로 예약 되어있다.

**Software Generated Interrupts (SGI)**

The SGI interrupts are always edge sensitive and are generated when software writes the interrupt number to ICDSGIR register.

SGI 인터럽트는 항상 엣지에 민감하며 소프트웨어가 인터럽트 번호를 ICDSGIR레지스터에 쓸 때 생성된다.

All of the targeted CPUs defined in the ICDIPTR [23:8] must handle the interrupt in order to clear it. See Figure 7-4 and Figure 7-5.

ICDIPTR[23:8]에 정의 된 모든 대상CPU는 이를 지우려면 인터럽트를 처리 해야합니다. 그림 7-4와7-5를 참고하라

**7.2.5 Wait for Interrupt Event Signal (WFI)**

The CPU can go into a wait state where it waits for an interrupt (or event) signal to be generated.

CPU는 인터럽트(또는 이벤트) 신호가 생성되기를 기다리는 대기 상태가 될 수 있다.

The wait for interrupt signal that is sent to the PL is described in Chapter 3, Application Processing Unit.

PL로 전송되는 인터럽트 신호 대기는 3장 응용 프로그램 처리 장치에 설명 되어 있다.

**7.3 Register Overview**

The ICC and ICD registers are part of the pl390 GIC register set.

ICC 및 ICD레지스터는 pl390 GIC레지스터 세트의 일부이다.

There are 60 SPI interrupts. This is far fewer than what the pl390 can support, so there are far fewer interrupt enable, status, prioritization and processor target registers in the ICD than is possible for the pl390.

60개의 SPI인터럽트가 있다. 이것은 pl390가 지원할 수 있는 것보다 훨씬 적기 때문에 ICD에서 pl390보다 가능한 인터럽트 활성화, 상태, 우선순위, 지정 및 프로세서 대상 레지스터가 훨씬적다.

A summary of the ICC and ICD registers are listed in Table 7-5

ICC 및 ICD레지스터의 요약은 표7-5에 나열되어 있다.

**7.3.1 Write Protection Lock**

The interrupt controller provides the facility to prevent write accesses to critical configuration registers.

인터럽트 제어 장치는 중요한 구성 레지스터에 대한 쓰기 접근을 방지하는 기능을 제공한다.

This is done by writing a one to the APU\_CTRL[CFGSDISABLE] bit. The APU\_CTRL register is part of the AP SoC’s System Level Control register set, SLCR.

이는 APU\_CTRL[CFGSDISABLE]비트에 1을 쓰면 된다. APU\_CTRL레지스터는 AP SoC의 시스템 레벨 제어 레지스터 세트인 SLCR의 일부이다.

This controls the write behavior for the secure interrupt control registers.

이것은 보안 인터럽트 제어 레지스터에 대한 쓰기 동작을 제어한다.

**7.4 Programming Model**

**7.4.1 Interrupt Prioritization**

All of the interrupt requests (PPI, SGI and SPI) are assigned a unique ID number.

모든 인터럽트 요청(PPI,SGI,SPI)에는 고유한 ID번호가 할당된다.

The controller uses the ID number to arbitrate.

제어 장치는 ID번호를 사용하여 중재한다.

The interrupt distributor holds the list of pending interrupts for each CPU, and then selects the highest priority interrupt before issuing it to the CPU interface

인터럽트 분배기는 각CPU에 대해 보류중인 인터럽트 목록을 보유한 다음 CPU 인터페이스에 발생하기 전에 우선순위가 가장 높은 인터럽트를 선택한다.

Interrupts of equal priority are resolved by selecting the lowest ID.

동일한 우선순위의 인터럽트는 가장 낮은 ID를 선택하여 해결한다.

The prioritization logic is physically duplicated to enable the simultaneous selection of the highest priority interrupt for each CPU.

우선 순위 지정 로직은 물리적으로 복제되어 각CPU에 대해 최우선 순위의 인터럽트를 동시에 선택할 수 있습니다.

The interrupt distributor holds the central list of interrupts, processors and activation information, and is responsible for triggering software interrupts to the CPUs.

인터럽트 분배기는 인터럽트, 프로세서 및 활성화 정보의 중앙 목록을 보유하며 CPU에 대한 소프트웨어 인터럽트를 트리거한다.

SGI and PPI distributor registers are banked to provide a separate copy for each connected processor.

SGI 및 PPI배포자 레지스터는 연결된 각프로세서에 대해 별도의 복사본을 제공하기 위해 뱅킹된다.

Hardware ensures that an interrupt targeting several CPUs can only be taken by one CPU at a time.

하드웨어를 사용하면 여러 CPU를 대상으로 하는 인터럽트를 한번에 하나 CPU에서만 처리할 수 있다.

The interrupt distributor transmits to the CPU interfaces the highest pending interrupt.

인터럽트 분배기는 가장 높은 보류중인 인터럽트를 CPU인터페이스에 전송한다.

It receives back the information that the interrupt has been acknowledged, and can then change the status of the corresponding interrupt.

인터럽트가 수신 확인 되었다는 정보를 수신한 다음 해당 인터럽트 상태를 변경 할 수 있다.

Only the CPU that acknowledges the interrupt can end that interrupt.

인터럽트를 인지한 CPU만 해당 인터럽트를 종료할 수 있습니다.

**7.4.2 Interrupt Handling**

The response of the GIC to a pending interrupt when an IRQ line de-asserts is described in the ARM document: *IHI0048B\_gic\_architecture\_specification.pdf* (see Appendix A, Additional Resources).

IRQ라인이 비활성화 될 때 보류중인 인터럽트에 대한 GIC의 응답은 ARM문서 : IHI0048B\_gic\_architecture\_specification.pdf(부록 A,추가 리소스 참조)에 설명되어 있다.

See the Note in Section 1.4.2 with additional information in Section 3.2.4.

3.2.4절의 추가 정보와 함께 1.4.2절의 참고사항을 참조하라

If the interrupt is pending in the GIC and IRQ is de-asserted, the interrupt in the GIC becomes inactive (and the CPU never sees it).

GIC에서 인터럽트가 보류상태이고 IRQ가 de-asserted되어 있으면 GIC는 비활성화된다.

If the interrupt is active in the GIC (because the CPU interface has acknowledged the interrupt), then the software ISR determines the cause by checking the GIC registers first and then polling the I/O

Peripheral interrupt status registers.

인터럽트가 GIC에서 활성화 된 경우(CPU 인터페이스가 인터럽트를 확인했기 때문에) 소프트웨어 ISR은 먼저 GIC레지스터를 확인한 다음 I/O 주변 장치 인터럽트 상태레지스터를 폴링(컴퓨터측에 주도권이 있는 방식)하여 원인을 확인한다.

**7.4.3 ARM Programming Topics**

The ARM GIC architecture specification includes these programming topics:

ARM GIC 아키텍처 사양에는 다음과 같은 프로그래밍 항목이 포함되어 있다.

• GIC register access -> GIC레지스터 액세스

• Distributor and CPU Interfaces -> 배포자 및 CPU인터페이스

• Affects of the GIC security extensions ->GIC보안 확장의 영향

• PU Interface registers ->PU 인터페이스 레지스터

• Preserving and restoring controller state ->컨트롤러 상태 유지 및 복원

**7.4.4 Legacy Interrupts and Security Extensions**

When the legacy interrupts (IRQ, FIQ) are used, and an interrupt handler accesses both IRQs and FIQs in secure mode (via ICCICR[AckCtl]=1), race conditions occasionally occur when reading the interrupt IDs.

레거시 인터럽트(IRQ,FIQ)가 사용되고 인터럽트 처리기가 ICCICR[AckCtrl]=1을 통해 보안 모드에서 IRQ및 FIQ에 모두 액세스하면 인터럽트 ID를 읽을 때 경쟁 조건이 가끔 발생한다.

There is also a risk of seeing FIQ IDs in the IRQ handler, as the GIC only knows what security state the handler is reading from, not which type of handler.

또한 GIC는 핸들러 유형이 아닌 핸들러에서 읽는 보안 상태만 알기 때문에 IRQ핸들러에서 FIQ ID를 볼 위험이 있다.

There are two workable solutions:

실제 가능한 솔루션은 두가지가 있다.

• Only signal IRQs to a re-entrant IRQ handler and use the preemption feature in the GIC.

재입력 IRQ처리기에만 IRQ신호를 보내고 GIC의 선점기능을 사용

• Use FIQ and IRQ with ICCICR[AckCtl]=0 and use the TLB tables to handle IRQ in non-secure mode, and handle FIQ in secure mode.

ICCICR[AckCtrl]=0인 FIQ및 IRQ를 사용하고 TLB테이블을 사용하여 비보안 모드에서 IRQ를 처리하고 보안모드에서 FIQ를 처리