S12X_INT
Block Guide
V01.06

Original Release Date: 07 FEB 2003
Revised: 17 JUN 2004

TSPG, 8/16 Bit
Motorola, Inc.

Revision History

Version Number	Revision Date	Effective Date	Author	Description of Changes
01.00	22 JUL 2003			initial release
01.01	03 SEP 2003			reworked according to feedback from Design Review
01.02	29 OCT 2003			clarified wording according to feedback
01.03	05 NOV 2003			added XGATE wake-up feature
01.04	09 FEB 2004			format cleanups
01.05	06 APR 2004			clarified wording based on feedback from evaluation team
01.06	15 JUN 2004	17 JUN 2004		changed classification

Table of Contents

List of Figures

List of Tables

Preface

The following terms and abbreviations are used in the document.

Table 0-1 Terminology

Term	Meaning
CCR	Condition Code Register (in the S12X CPU)
DMA	Direct Memory Access
INT	Interrupt
IPL	Interrupt Processing Level
ISR	Interrupt Service Routine
MCU	Micro-Controller Unit
XGATE	please refer to the "XGATE Block Guide"
$\overline{\text{IRQ}}$	refers to the interrupt request associated with the $\overline{\text{IRQ}}$ pin
$\overline{\text{XIRQ}}$	refers to the interrupt request associated with the $\overline{\text{XIRQ}}$ pin

Section 1 Introduction

Figure 1-1 S12X_INT Block Diagram shows a block diagram of the S12X_INT module.

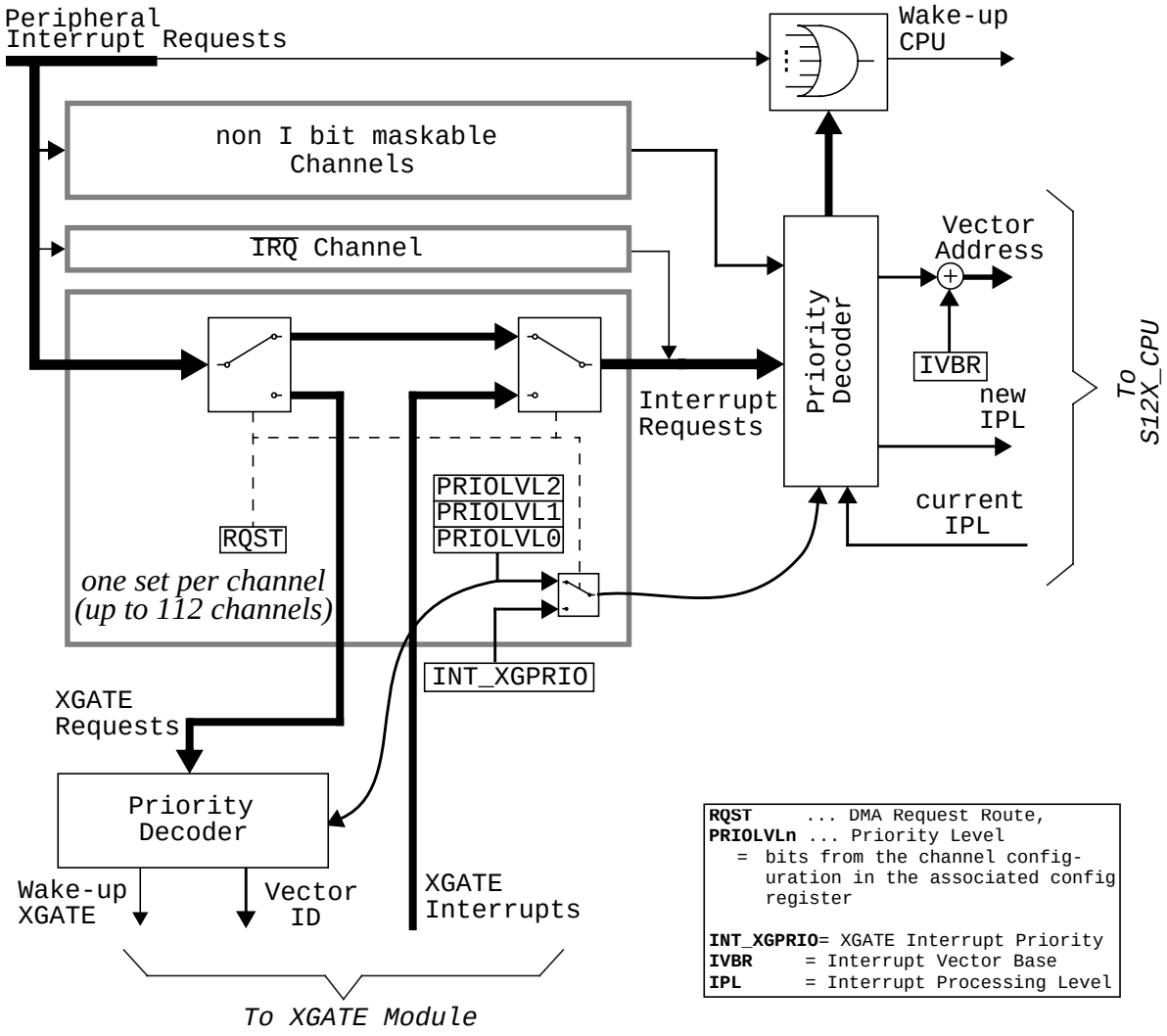

Figure 1-1 S12X_INT Block Diagram

1.1 Overview

The S12X_INT module decodes the priority of all system exception requests and provides the applicable vector for processing the exception to either the S12X_CPU or the XGATE module. The S12X_INT module supports I bit and X bit maskable interrupt requests, a non-maskable Unimplemented Opcode Trap, a non-maskable software interrupt (SWI) or Background Debug Mode request, a spurious interrupt

vector request and 3 system reset vector requests. Each of the I bit maskable interrupt request can be assigned to one of 7 priority levels supporting a flexible priority scheme. For interrupt requests that are configured to be handled by the S12X_CPU, the priority scheme can be used to implement nested interrupt capability where interrupts from a lower level are automatically blocked if a higher level interrupt is being processed. Interrupt requests configured to be handled by the XGATE module cannot be nested because the XGATE module cannot be interrupted while processing.

NOTE: *The HPRIO register and functionality of the S12_INT module is no longer supported, since it is superseded by the 7 level interrupt request priority scheme.*

1.2 Features

- Interrupt Vector Base Register (IVBR)
- One spurious interrupt vector (at address (Vector Base[1] + $10)).
- 2...113 I bit maskable interrupt vector requests (at addresses (Vector Base + $12...$F2)).
- each I bit maskable interrupt request has a configurable priority level and can be configured to be handled by either the S12X_CPU or the XGATE module.
- I bit maskable interrupts can be nested, depending on their priority levels.
- One X bit maskable interrupt vector request (at address (Vector Base + $F4)).
- One non maskable software interrupt request (SWI) or Background Debug Mode vector request (at address (Vector Base + $F6)).
- One non maskable Unimplemented Opcode Trap (TRAP) vector (at address (Vector Base + $F8)).
- 3 system reset vectors (at addresses $FFFA...$FFFE).
- Determines the highest priority DMA- and Interrupt vector requests and drives the vector to the XGATE module or to the bus on CPU request, respectively.
- Wakes-up the system from STOP- or WAIT-mode when an appropriate interrupt request occurs or whenever $\overline{\text{XIRQ}}$ is asserted, even if X interrupt is masked.
- XGATE can wake-up and execute code, even with CPU remaining in STOP- or WAIT-mode.

1.3 Modes of Operation

- Run Mode

  This is the basic mode of operation.

- Wait Mode

NOTES:

1. The Vector Base is a 16 bits address which is accumulated from the contents of the Interrupt Vector Base Register (IVBR, used as upper byte) and $00 (used as lower byte).

In wait mode the S12X_INT module is frozen. It is however capable of either waking-up the CPU if an interrupt occurs or waking-up the XGATE if an XGATE request occurs. Please refer to **5.3 Wake-up from STOP- or WAIT-Mode** for details.

- Stop Mode

  In stop mode the S12X_INT module is frozen. It is however capable of either waking-up the CPU if an interrupt occurs or waking-up the XGATE if an XGATE request occurs. Please refer to **5.3 Wake-up from STOP- or WAIT-Mode** for details.

- Freeze Mode (BDM active)

  In freeze mode (BDM active) the Interrupt Vector Base Register is overridden internally. Please refer to **3.1.1 Interrupt Vector Base Register (IVBR)** for details.

Section 2 External Signal Description

The S12X_INT module has no external signals.

Section 3 Memory Map/Register Definition

Table 3-1 gives an overview over all S12X_INT registers. The base address (“Base”) of the S12X_INT module is determined at chip-level. Please refer to the Device User Guide for more information.

Table 3-1 Module Memory Map

Address	Use	Access
Base+$00	RESERVED	-
Base+$01	Interrupt Vector Base Register (IVBR)	R/W
Base + $02...$05	RESERVED	-
Base+$06	XGATE Interrupt Priority Configuration Register (INT_XGPRIO)	R/W
Base+$07	Interrupt Request Configuration Address Register (INT_CFADDR)	R/W
Base+$08	Interrupt Request Configuration Data Register 0 (INT_CFDATA0)	R/W
Base+$09	Interrupt Request Configuration Data Register 1 (INT_CFDATA1)	R/W
Base+$0A	Interrupt Request Configuration Data Register 2 (INT_CFDATA2	R/W
Base+$0B	Interrupt Request Configuration Data Register 3 (INT_CFDATA3)	R/W
Base+$0C	Interrupt Request Configuration Data Register 4 (INT_CFDATA4)	R/W
Base+$0D	Interrupt Request Configuration Data Register 5 (INT_CFDATA5)	R/W
Base+$0E	Interrupt Request Configuration Data Register 6 (INT_CFDATA6)	R/W
Base+$0F	Interrupt Request Configuration Data Register 7 (INT_CFDATA7)	R/W

3.1 Register Descriptions

3.1.1 Interrupt Vector Base Register (IVBR)

Register address Base + $01

	7	6	5	4	3	2	1	0
R W	IVB_ADDR[7:0]							
RESET:	1	1	1	1	1	1	1	1

= Unimplemented or Reserved

Figure 3-1 Interrupt Vector Base Register (IVBR)

Read: anytime

Write: anytime

IVB_ADDR[7:0] — Interrupt vector base address bits

These bits represent the upper byte of all vector addresses. Out of reset these bits are set to $FF (i.e. vectors are located at $FF10...$FFFE) to ensure compatibility to HCS12.

NOTE: *A system reset will initialize the Interrupt Vector Base Register with "$FF" before it is used to determine the reset vector address. Therefore changing the IVBR has no effect on the location of the 3 reset vectors ($FFFA-$FFFE).*

NOTE: *If the BDM is active (i.e. the CPU is in the process of executing BDM firmware code), the contents of IVBR are ignored and the upper byte of the vector address is fixed as "$FF".*

3.1.2 XGATE Interrupt Priority Configuration Register (INT_XGPRIO)

Register address Base + $06

	7	6	5	4	3	2	1	0
R	0	0	0	0	0	XILVL[2:0]		
W								
RESET:	0	0	0	0	0	0	0	1

= Unimplemented or Reserved

Figure 3-2 XGATE Interrupt Priority Configuration Register (INT_XGPRIO)

Read: anytime

Write: anytime

XILVL[2:0] — XGATE Interrupt Priority Level

Table 3-2 XGATE Interrupt Priority Levels

Priority	XILVL2	XILVL1	XILVL0	Meaning
	0	0	0	Interrupt request is disabled
low	0	0	1	Priority level 1
	0	1	0	Priority level 2
	0	1	1	Priority level 3
	1	0	0	Priority level 4
	1	0	1	Priority level 5
	1	1	0	Priority level 6
high	1	1	1	Priority level 7

The XILVL[2:0] bits configure the shared interrupt level of the DMA interrupts coming from the XGATE module. Out of reset the priority is set to the lowest active level (“1”).

3.1.3 Interrupt Request Configuration Address Register (INT_CFADDR)

Register address Base + $07

	7	6	5	4	3	2	1	0
R	INT_CFADDR[7:4]				0	0	0	0
W								
RESET:	0	0	0	1	0	0	0	0

= Unimplemented or Reserved

Figure 3-3 Interrupt Configuration Address Register (INT_CFADDR)

Read: anytime

Write: anytime

INT_CFADDR[7:4] — Interrupt Request Configuration Data Register select bits

These bits determine which of the 128 configuration data registers are accessible in the 8 register window at INT_CFDATA0...7. The hexadecimal value written to this register corresponds to the upper nibble of the lower byte of the interrupt vector, i.e. writing $E0 to this register selects the configuration data register block for the 8 interrupt vector requests starting with vector (Vector Base +$E0) to be accessible as INT_CFDATA0...7.

NOTE: *Writing all zero selects non-existing configuration registers. In this case write accesses to INT_CFDATA0...7 will be ignored and read accesses will return all zero.*

3.1.4 Interrupt Request Configuration Data Registers (INT_CFDATA0...7)

The 8 register window visible at addresses INT_CFDATA0...7 contains the configuration data for the block of 8 interrupt requests (out of 128) selected by the Interrupt Configuration Address Register (INT_CFADDR) in ascending order. INT_CFDATA0 represents the Interrupt Configuration Data Register of the vector with the lowest address in this block, while INT_CFDATA7 represents the Interrupt Configuration Data Register of the vector with the highest address, respectively.

Register address Base + $08...$0F

	7	6	5	4	3	2	1	0
R	RQST	0	0	0	0	PRIOLVL[2:0]		
W								
RESET:	0	0	0	0	0	0	0	1[1]

= Unimplemented or Reserved

NOTES:
1. Please refer to the notes following the PRIOLVL[2:0] description below.

Figure 3-4 Interrupt Configuration Data Registers (INT_CFDATA0...7)

Read: anytime

Write: anytime

RQST — XGATE Request Enable

This bit determines if the associated interrupt request is handled by the S12X_CPU or by the XGATE module.

1 = interrupt request is handled by the XGATE module
0 = interrupt request is handled by the S12X_CPU

NOTE: *The $\overline{IRQ}$ interrupt can not be handled by the XGATE module. For this reason the configuration register for vector ((Vector Base + $F2) =$\overline{IRQ}$ vector address) does not contain a "RQST" bit. Writing a one to the location of the "RQST" bit in this register will be ignored and a read access will return zero.*

PRIOLVL[2:0] — Interrupt Request Priority Level Bits

The PRIOLVL[2:0] bits configure the interrupt request priority level of the associated interrupt request. Out of reset all interrupt requests are enabled at the lowest active level ("1") to provide backwards compatibility with previous HCS12 interrupt controllers. Please also refer to **Table 3-3** for available interrupt request priority levels.

Table 3-3 Interrupt Priority Levels

Priority	PRIOLVL2	PRIOLVL1	PRIOLVL0	Meaning
	0	0	0	Interrupt request is disabled
low	0	0	1	Priority level 1

Table 3-3 Interrupt Priority Levels

Priority	PRIOLVL2	PRIOLVL1	PRIOLVL0	Meaning
	0	1	0	Priority level 2
	0	1	1	Priority level 3
	1	0	0	Priority level 4
	1	0	1	Priority level 5
	1	1	0	Priority level 6
high	1	1	1	Priority level 7

NOTE: *Write accesses to configuration data registers of unused interrupt channels will be ignored and read accesses will return all zero. For information about what interrupt channels are used in a specific MCU, please refer to the Device User Guide of that MCU.*

NOTE: *When vectors (Vector Base + $F0...$FE) are selected by writing $F0 to INT_CFADDR, writes to INT_CFDATA2...7 ($F4...$FE) will be ignored and read accesses will return all zero. The corresponding vectors do not have configuration data registers associated with them.*

NOTE: *Write accesses to the configuration register for the spurious interrupt vector request (Vector Base + $10) will be ignored and read accesses will return "$07" (request is handled by the S12X_CPU, PRIOLVL = "7").*

Section 4 Functional Description

The S12X_INT module processes all exception requests to be serviced by the S12X_CPU module. These exceptions include interrupt vector requests and reset vector requests. Each of these exception types and their overall priority level is discussed in the subsections below.

4.1 S12X Exception Requests

The S12X_CPU handles both reset requests and interrupt requests. The S12X_INT contains registers to configure the priority level of each I bit maskable interrupt request which can be used to implement an interrupt priority scheme. This also includes the possibility to nest interrupt requests. A priority decoder is used to evaluate the priority of a pending interrupt request.

4.1.1 Interrupt prioritization

After system reset all interrupt requests with a vector address lower than or equal to (Vector Base + $F2) are enabled, are set up to be handled by the S12X_CPU and have a pre-configured priority level of “1”. The exception to this rule is the spurious interrupt vector request at (Vector Base + $10) which cannot be disabled, is always handled by the S12X_CPU and has a fixed priority level of “7”. A priority level of “0” effectively disables the associated interrupt request.

If more than one interrupt request is configured to the same interrupt priority level the interrupt request with the higher vector address wins the prioritization.

The following conditions must be met for an I bit maskable interrupt request to be processed.

1. The local “interrupt enabled” bit in the peripheral module must be set.
2. The setup in the configuration register associated with the interrupt request channel must meet the following conditions:
   a. The XGATE Request Enable bit must be zero to have the S12X_CPU handle the interrupt request.
   b. The Priority Level must be set to non zero.
   c. The Priority Level must be greater than the current Interrupt Processing Level in the Condition Code Register (CCR) of the S12X_CPU (PRIOLVL[2:0] > IPL[2:0]).
3. The I-Bit in the Condition Code Register (CCR) of the S12X_CPU must be cleared.
4. There is no SWI, TRAP or $\overline{\text{XIRQ}}$ request pending.

NOTE: *All non I bit maskable interrupt requests always have higher priority than I bit maskable interrupt requests. If an I bit maskable interrupt request is interrupted by a non I bit maskable interrupt request, the currently active Interrupt Processing Level (IPL) remains unaffected. It is possible to nest non I bit maskable interrupt requests, e.g. by nesting SWI or TRAP calls.*

4.1.1.1 Interrupt Priority Stack

The current Interrupt Processing Level (IPL) is stored in the Condition Code Register (CCR) of the S12X_CPU. This way the current IPL is automatically pushed to the stack by the standard interrupt stacking procedure. The new IPL is copied to the CCR from the Priority Level of the highest priority active interrupt request channel which is configured to be handled by the S12X_CPU. The copying takes place when the interrupt vector is fetched. The previous IPL is automatically restored by executing the “RTI” instruction.

4.2 XGATE Requests

The S12X_INT module processes all exception requests to be serviced by the XGATE module. The overall priority level of those exceptions is discussed in the subsections below.

4.2.1 XGATE request prioritization

An interrupt request channel is configured to be handled by the XGATE module, if the RQST bit of the associated configuration register is set to “1” (please refer to **3.1.4 Interrupt Request Configuration Data Registers (INT_CFDATA0...7)**). The priority level setting (PRIOLVL) for this channel becomes the DMA priority which will be used to determine the highest priority DMA request to be serviced next by the XGATE module. Additionally, DMA interrupts may be raised by the XGATE module by setting one or more if the XGATE Channel Interrupt flags (using the “SIF” instruction). This will result in a S12X_CPU interrupt with vector address Vector Base + (2 * Channel ID number), where the Channel ID number corresponds to the highest set Channel Interrupt Flag, if the XGIE bit is set.

The shared interrupt priority for the DMA interrupt requests is taken from the XGATE Interrupt Priority Configuration Register (please refer to **3.1.2 XGATE Interrupt Priority Configuration Register (INT_XGPRIO)**). If more than one DMA interrupt request channel becomes active at the same time, the channel with the highest vector address wins the prioritization.

4.3 Priority Decoders

The S12X_INT module contains priority decoders to determine the priority for all interrupt requests pending for the respective target.

There are two priority decoders, one for each interrupt request target (S12X_CPU, XGATE module). The function of both priority decoders is basically the same with one exception: the priority decoder for the XGATE module does not take the current Interrupt Processing Level into account because XGATE requests cannot be nested.

Because the vector is not supplied until the CPU requests it, it is possible that a higher priority interrupt request could override the original exception that caused the CPU to request the vector. In this case, the CPU will receive the highest priority vector and the system will process this exception instead of the original request.

If the interrupt source is unknown (for example, in the case where an interrupt request becomes inactive after the interrupt has been recognized, but prior to the vector request), the vector address supplied to the CPU will default to that of the spurious interrupt vector.

NOTE: *Care must be taken to ensure that all exception requests remain active until the system begins execution of the applicable service routine; otherwise, the exception request may not get processed at all or the result may be a spurious interrupt request (vector at address (Vector Base + $10)).*

4.4 Reset Exception Requests

The S12X_INT supports 3 system reset exception request types (please refer to CRG for details):

- Pin-Reset, Power-On-Reset, low voltage Reset or illegal address Reset
- Clock Monitor Reset request
- COP Watchdog Reset request

4.5 Exception Priority

The priority (from highest to lowest) and address of all exception vectors issued by the S12X_INT upon request by the S12X_CPU is shown in **Table 4-1**.

Table 4-1 Exception Vector Map and Priority

Vector Address[1]	Source
$FFFE	Pin-Reset, Power-On-Reset, low voltage Reset, illegal address Reset
$FFFC	Clock Monitor reset
$FFFA	COP Watchdog reset
(Vector Base + $F8)	Unimplemented opcode trap
(Vector Base + $F6)	Software interrupt instruction (SWI) or BDM vector request
(Vector Base + $F4)	$\overline{\text{XIRQ}}$ interrupt request
(Vector Base + $F2)	$\overline{\text{IRQ}}$ interrupt request
(Vector Base + $F0...$12)	Device-specific I bit maskable interrupt sources (priority determined by the associated configuration registers, in descending order)
(Vector Base + $10)	Spurious Interrupt

NOTES:
1. 16 bits Vector Address based

Section 5 Initialization/Application Information

5.1 Initialization

After system reset, software should:

- Initialize the Interrupt Vector Base Register if the interrupt vector table is not located at the default location ($FF10-$FFF8).
- Initialize the Interrupt Processing Level Configuration Data Registers (INT_CFADDR, INT_CFDATA0...7) for all interrupt vector requests with the desired priority levels and the request target (S12X_CPU or XGATE module). It might be a good idea to disable unused interrupt requests.
- If the XGATE module is used, setup the XGATE Interrupt Priority Register (INT_XGPRIO) and configure the XGATE module (please refer the XGATE Block Guide for details).
- Enable I maskable interrupts by clearing the I bit in the CCR.
- Enable the X maskable interrupt by clearing the X bit in the CCR (if required).

5.2 Interrupt Nesting

The Interrupt Request Priority Level scheme makes it possible to implement priority based interrupt request nesting for the I bit maskable interrupt requests handled by the S12X_CPU.

- I bit maskable interrupt requests can be interrupted by an interrupt request with a higher priority, so that there can be up to 7 nested I bit maskable interrupt requests at a time (refer to **Figure 5-1 Interrupt Processing example** for an example using up to 3 nested interrupt requests).

I -bit maskable interrupt requests cannot be interrupted by other I bit maskable interrupt requests per default. In order to make an Interrupt Service Routine (ISR) interruptible, the ISR must explicitly clear the I bit in the CCR (“CLI”). After clearing the I bit, I bit maskable interrupt requests with higher priority can interrupt the current ISR.

An ISR of an interruptible I bit maskable interrupt request could basically look like this:

- service interrupt, e.g. clear interrupt flags, copy data, etc.
- clear I bit in the CCR by executing the instruction “CLI” (thus allowing interrupt requests with higher priority)
- process data
- return from interrupt by executing the instruction “RTI”

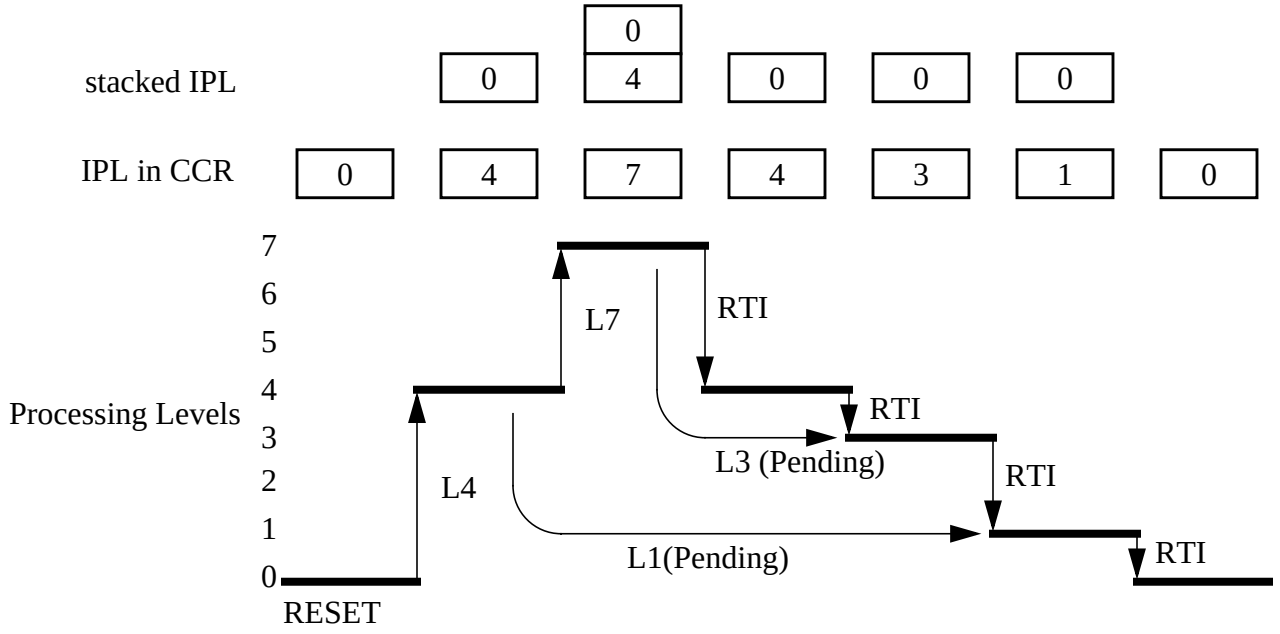

Figure 5-1 Interrupt Processing example

5.3 Wake-up from STOP- or WAIT-Mode

5.3.1 CPU wake-up from STOP- or WAIT-Mode

Every I bit maskable interrupt request which is configured to be handled by the S12X_CPU is capable of waking the MCU from STOP- or WAIT-mode. To determine whether an I bit maskable interrupts is qualified to wake-up the CPU or not, the same settings as in normal run mode are applied during STOP- or WAIT-mode:

- If the I bit in the CCR is set, all I bit maskable interrupts are masked from waking-up the MCU.
- An I bit maskable interrupt is ignored if it is configured to a priority level below or equal to the current IPL in CCR.
- I bit maskable interrupt requests which are configured to be handled by the XGATE are not capable of waking up the CPU.

An $\overline{\text{XIRQ}}$ request can wake-up the MCU from STOP- or WAIT-Mode at anytime, even if the X bit in CCR is set.

5.3.2 XGATE wake-up from STOP- or WAIT-Mode

Interrupt request channels which are configured to be handled by the XGATE are capable of waking-up the XGATE. Interrupt request channels handled by the XGATE do not affect the state of the CPU.

Block Guide End Sheet

FINAL PAGE OF
26
PAGES