Altering Interrupt Source Priorities in the NET+ARM BSP.

This technical note describes how to change interrupt priorities in the Board Support Package for the NET+ARM® processor product line.

The note is broken into several sections.

- The first section reviews the overall interrupt handling strategy employed by the NET+OS™ products.
- The second section describes the specifics of how interrupt prioritization is implemented and how to alter source priorities.
- The third section provides some guidelines and precautions that must be observed in altering interrupt priorities.

NET+ARM Interrupt Processing

All interrupt requests for the ARM7TDMI core are routed to one of two lines, IRQ or FIRQ ("Fast Interrupt Request,") each of which causes a different exception. The IRQ exception inherently has a lower priority than FIRQ, and is masked at the core level while an FIRQ exception is being processed. In the NET+ARM, FIRQ is available to the watchdog timer and Timers 1 and 2. All other interrupt sources make use of IRQ.

Occurrence of an IRQ or FIRQ exception causes the "I" or "F" bit, respectively, of the ARM core's Current Program Status Register (CPSR) to be set; this disables the corresponding exception, preventing a second occurrence during critical portions of exception handling. These bits can also be set via processor instructions, allowing the exception handler software (in this case, the NET+ARM interrupt driver) to explicitly specify when exceptions can be tolerated and when they cannot. It is important to note that all interrupt sources can be masked in the GEN module, which essentially prevents IRQ or FIRQ from being asserted by one of the said sources.

Interrupts are handled by the Interrupt controller of the GEN module. Each interrupt can be individually enabled by setting the corresponding bit in the Interrupt Enable Register (IER - 0xFFB0 0030):

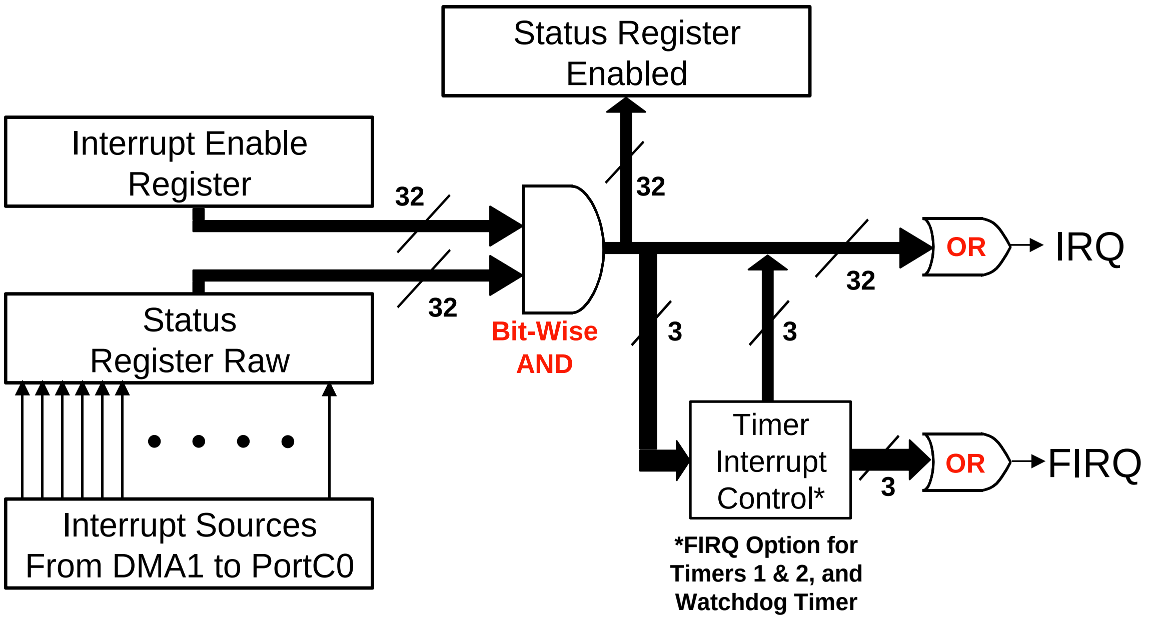
NET+ARM Interrupt Enable Register (0xFFB0 0030)

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
DMA1	DMA2	DMA3	DMA4	DMA5	DMA6	DMA7	DMA	DMA9	DMA10	ENI PORT 1	ENI PORT 2	ENI PORT 3	ENI PORT 4	ENET RX	ENET TX
RESET:															
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
SER 1 RX	SER 1 TX	SER2 RX	SER 2 TX	—	—	—	—	—	Watch Dog	Timer 1	Timer 2	PortC PC3	PortC PC2	PortC PC1	PortC PC0
RESET:															
0	0	0	0						0	0	0	0	0	0	0
R/W	R/W	R/W	R/W						R/W	R/W	R/W	R/W	R/W	R/W	R/W

Additional interrupt-related registers used in the GEN Module correspond on a bit-by-bit basis with the layout of the IER.

When an interrupt of a specific type occurs, its corresponding bit is set in the Interrupt Status Register Raw (0xFFB0 0038). The logical AND of this register and the IER is available in the Interrupt Status Register Enabled (ISRE - 0xFFB0 0034). It is the latter register that is inspected by the NET+ARM interrupt driver to ascertain the highest-priority source of the pending interrupts when an IRQ or FIRQ exception occurs. This is illustrated by the following diagram:


Status Register Enabled
Interrupt Enable Register
32
32
32
Status Register Raw
Bit-Wise AND
3
3
32
OR
IRQ
Timer Interrupt Control*
OR
FIRQ
3
Interrupt Sources From DMA1 to PortC0
*FIRQ Option for Timers 1 & 2, and Watchdog Timer

On recognizing the highest-priority interrupt source, the driver temporarily masks out (disables) interrupts from lower-priority sources, ensuring that interrupt service for a high-priority source is not itself interrupted by a request from a lower-priority source. It then re-enables the appropriate exception via the “I” bit in the CPSR to permit higher-priority interrupts to occur and transfers control to the user’s interrupt handler (ISR).

When the user’s interrupt handler returns, the driver masks off the exception via the CPSR and re-enables interrupts from lower-priority sources via the IER. Exiting from the exception handling routine automatically re-enables the IRQ exception..

Interrupt Prioritization

The relative priority of each interrupt in the system is determined by the NAInterruptPriority array in module NETOSx\src\bsp\Na_isr.c. The default setting of this array is as follows:

```
static int unsigned NAInterruptPriority[ ] = {
 0, 1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12, 14, 13, 15,
 16, 17, 18, 19, 20, 21, 22, 23, 24, 25, 26, 27, 28, 29, 30, 31
};
```

The index of each entry in this array corresponds to the bit number of the corresponding interrupt in the interrupt registers (such as the Interrupt Enable Register or Interrupt Status Registers.) Thus, entry NAInterruptPriority[0] corresponds to Port C, bit 0, and NAInterruptPriority[31] corresponds to DMA Channel 1.

The NAInterruptPriority array must contain the priority from 0 to 31 (31 being the highest) for the interrupt corresponding to every bit in the Interrupt Registers whether enabled or not, and no duplicates are permitted. The reason for these restrictions is that when an interrupt request is received, all lower-priority interrupts are masked off until the highest-priority pending interrupt has been processed. Clearly, omissions or duplications in the NAInterruptPriority table would produce ambiguities, so they are not allowed by the NAInitIsr function that performs setup for interrupt service. If these guidelines are not followed, the BSP will enter an infinite loop, “blinking” Port C bit 1 (connected on the development board to the red or yellow LED) in groups of 3 flashes.

To reset the priorities of interrupts for the NET+ARM, you must alter the NAInterruptPriority array and rebuild the BSP. For example, to swap the priority of interrupts from Port C bits 0 and 1, change the array as follows:

```
static int unsigned NAInterruptPriority[ ] = {
 1, 0, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12, 14, 13, 15,
 16, 17, 18, 19, 20, 21, 22, 23, 24, 25, 26, 27, 28, 29, 30, 31
};
```

Precautions and Guidelines for Application

Some caveats must be observed:

- Priorities in the NAInterruptPriority array obviously affect the manner in which the NET+ARM chip and NET+OS respond to interrupt events, and should not be altered without undertaking a thorough analysis of the possible results of reprioritizing. To validate your analysis, it is suggested that you test your application first without resetting the interrupt priorities before checking it with interrupts reprioritized.
- The NAInterruptPriority table cannot be altered "on the fly" since it drives the setup of a number of other tables in the interrupt service routine. The only way in which to alter the routine safely is to disable all interrupts, make the alteration, and then re-execute the NAInitIsr function (this function is NOT documented in the BSP API manual since its use in customer code is ordinarily not recommended. Consult file Na-isr.c for details.) Relevant interrupts will then need to be re-enabled, just as when your initialization was first performed.
- Since the NAInterruptPriority array has the "static" qualifier, it is limited to file scope within the NA_isr.c file. Because this file is normally supplied as part of a library, you must include any revised copy explicitly when building. The linker will include your separate object file in preference to the copy in the library.