README on the Vectored Interrupt Controller of the LH7A404
==========

The 404 revision of the LH7A40X series comes with two vectored interrupts controllers. While the kernel does use some of the features of these devices, it is far from the purpose for which they were designed.

When this README was written, the implementation of the VICs was in flux. It is possible that some details, especially with priorities, will change.

The VIC support code is inspired by routines written by Sharp.

Priority Control

The significant reason for using the VIC's vectoring is to control interrupt priorities. There are two tables in arch/arm/mach-lh7a40x/irq-lh7a404.c that look something like this.

```
  static unsigned char irq_pri_vic1[] = { IRQ_GPIO3INTR, };
  static unsigned char irq_pri_vic2[] = {
        IRQ_T3UI, IRQ_GPIO7INTR,
        IRQ_UART1INTR, IRQ_UART2INTR, IRQ_UART3INTR, };
```

The initialization code reads these tables and inserts a vector address and enable for each indicated IRQ. Vectored interrupts have higher priority than non-vectored interrupts. So, on VIC1, IRQ_GPIO3INTR will be served before any other non-FIQ interrupt. Due to the way that the vectoring works, IRQ_T3UI is the next highest priority followed by the other vectored interrupts on VIC2. After that, the non-vectored interrupts are scanned in VIC1 then in VIC2.

ISR

The interrupt service routine macro get_irqnr() in arch/arm/kernel/entry-armv.S scans the VICs for the next active interrupt. The vectoring makes this code somewhat larger than it was before using vectoring (refer to the LH7A400 implementation). In the case where an interrupt is vectored, the implementation will tend to be faster than the non-vectored version. However, the worst-case path is longer.

It is worth noting that at present, there is no need to read VIC2_VECTADDR because the register appears to be shared between the controllers. The code is written such that if this changes, it ought to still work properly.

Vector Addresses

The proper use of the vectoring hardware would jump to the ISR specified by the vectoring address. Linux isn't structured to take advantage of this feature, though it might be possible to change things to support it.

In this implementation, the vectoring address is used to speed the search for the active IRQ. The address is coded such that the lowest 6 bits store the IRQ number for vectored interrupts. These numbers correspond to the bits in the interrupt status registers. IRQ zero is the lowest interrupt bit in VIC1. IRQ 32 is the lowest interrupt bit in VIC2. Because zero is a valid IRQ number and because we cannot detect whether or not there is a valid vectoring address if that address is zero, the eigth bit (0x100) is set for vectored interrupts. The address for IRQ 0x18 (VIC2) is 0x118. Only the ninth bit is set for the default handler on VIC1 and only the tenth bit is set for the default handler on VIC2.

In other words.

0x000	- no active interrupt
0x1ii	- vectored interrupt 0xii
0x2xx	- unvectored interrupt on VIC1 (xx is don't care)
0x4xx	- unvectored interrupt on VIC2 (xx is don't care)