Using the fast IRQ in ARM Linux

(the official support and the fiq-engine external package)

ELC Europe 2009
Grenoble October 15th 2009

Alessandro Rubini
Independent consultant,
contract professor, Pavia University
<rubini@linux.it> <rubini@gnudd.com> <rubini@unipv.it>

What is the FIQ

The ARM core offers two different maskable interrupts

- The normal IRQ, used by all devices through a cascade of muxes
- The "fast" IRQ, a.k.a. FIQ, that nobody uses

We can thus consider the FIQ as a non-maskable interrupt, even though, if needed, it can be masked just like the irq

The FIQ as a input line can be connected to any peripheral

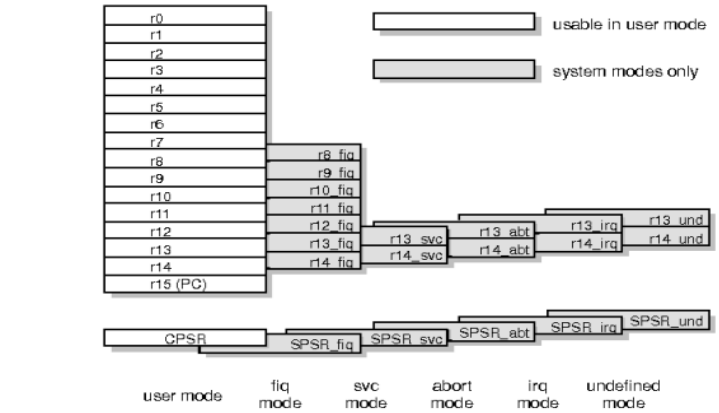
- All interrupt controllers (so far) allow any interrupt to be routed to either irq of fiq
- This is usually limited to the first level of multiplexers (32 irq sources)

With a timer and the FIQ, you can arrange for RT activities

- A few lines of assembly for critical tasks
- Or a real task, periodic or aperiodic, to do I/O or whatever
- Or you could connect a scheduler as well
  - This however is by no means a replacement for xenomai/rtai

ARM modes and registers

The ARM has a FIQ CPU mode, with specific registers



CONFIG_FIQ and set_fiq_handler()

The mainline kernel offers some FIQ support:

- CONFIG_FIQ can be activated in the configuration
  - By default only a few machines define CONFIG_FIQ
  - All other machines force CONFIG_FIQ to undef, for no real reason
- There are a few functions, in <arm/fiq.h>, for C code to use:

```
extern void set_fiq_handler(void *start, unsigned int length);
extern void set_fiq_regs(struct pt_regs *regs);
extern void get_fiq_regs(struct pt_regs *regs);
/* a few more */
```

Client code can define a small handler, which is usually written in assembly

- set_fiq_handler() copies the code to the proper place in RAM
- client code can pre-set the banked registers or read them

An example use of CONFIG_FIQ

In a recent project, I had to generate a 60Hz PWM

- It was used for the LCD backlight
- The hardware PWM couldn't run at such low frequencies
- All of the code, C and assembly, is shown in this page

```
.globl poi_fiq_init
.globl poi_fiq_fini
.globl poi_fiq_base

poi_fiq_init:
    ldr     r10, poi_fiq_base
    ldr     r10, [r10, #0x20]
    /* acked by reading the status register */
    ldr     r10, count
    add     r10, r10, #1
    and     r10, r10, #0xff
    str     r10, count
    ldr     r11, duty_addr
    ldr     r11, [r11]
    cmp     r10, r11
    ldr     r10, =0xfefff430 /* PB9: set at 0x30 */
    addgt   r10, r10, #4     /* or clear at 0x34 */
    mov     r11, #0x200
    str     r11, [r10]
    /* return to caller and mode */
    subs    pc, lr, #4
```

```
poi_fiq_base:
        .word 0
count:
        .word 0
duty_addr:
        .word poi_fiq_duty
.ltorg


        /* ioremap the timer block for asm */
        poi_fiq_base = ioremap_nocache(base, 0x40);

        /* Register the handler */
        ret = claim_fiq(&poi_fiq_handler);
        set_fiq_handler(&poi_fiq_init,
              &poi_fiq_fini - &poi_fiq_init);
```


fiq-engine: supporting external modules

A pair of years ago, I wrote the fiq-engine set of modules

- Supports a more complex task, written in C
- Supports modules for easier development
- Offers some diagnostics help

The package is a kernel patch and a few modules

- We can't afford a page fault ("data abort") in fiq context
- The kernel patch ensures no data abort ever happens in FIQ.

The patch, not submitted to mainline, modifies vmalloc

- When vmalloc is called, maps are exported to all processes
- This is needed as I use vmalloc space in fiq-misc.ko
- The patch is small, but most likely not acceptable for mainline
  - It uses #ifdef ARM in mm/vmalloc.c
  - It is for a very uncommon use of the system

`http://gnudd.com/pub/samplecode/fiq-engine-1.3.tar.gz`

Bprintk and sysctl-stamp

fiq-engine first offers support for diagnostics

bprintk.ko is a buffered printk

- You can't call printk from the FIQ context
  - Actually you physically can, but it may explode
  - Linux may be in a critical sections when FIQ runs
- bprintk offers a printk-like function, with a local buffer
- The accumulated strings are sent to printk in a kernel timer

sysctl-stamp is a simple timestamping mechanism

- It uses the (somewhat deprecated) sysctl primitives
- The client module can record timestamping events
- The client can timestamp at any time from any context
- The user can read from and write to in /proc/sys/dev/
  - read: maximum, miminum, runing average
  - write: reset counters to start with fresh data

fiq-misc: communicating with user space

The fiq-misc module allocates a vmalloc area, exported in mmap

- The size is a parameter, defaults to 64kB
- No read or write is offered, as I love mmap (and I'm lazy)

FIQ context can't call Linux functions, but it can share memory

- Tipically the FIQ task either inputs or outputs data
- FIQ task and process must agree about a protocol to avoid races
- We have all the usual issues of concurrent access

fiq-misc.ko isn't really part of FIQ operation

- It just allocates and exports the buffer for fiq-task
- Being a vmalloc area accessed from FIQ, the vmalloc patch is required

fiq-engine and fiq-task: make a real-time task

fiq-engine is the main actor of the package

- It can be configured for IRQ (default) or FIQ
- It deals with all hardware registers, offering a C API

A few diffent ARM families are supported

- AT91SAM926x (used in production)
- PXA255/270 (used in production)
- STE Nomadik (beta stage, needs audit and publication in fiq-engine-1.4)
- iMX21 (work in progress)
- Samsung S3C440 (on request, not published as i can't currently test it)

fiq-task is the public-doman example of a user module

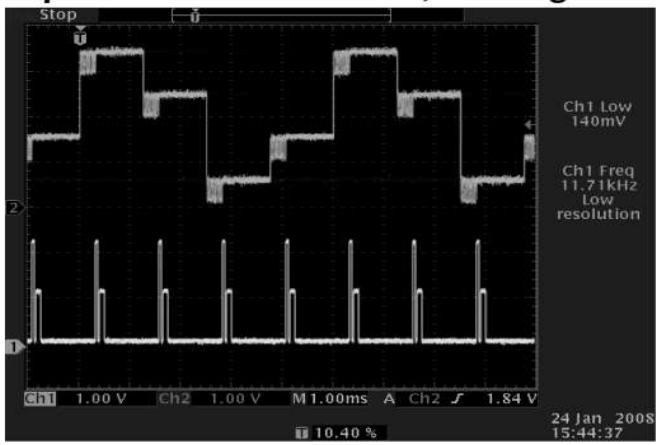
- The default implementation just toggles a GPIO pin
- It is public domain (all the rest is GPL) to allow proprieary users
  - It is only sample code, I let real programmers choose their license

fiq-empty is another task example

- It toggles the bit immediately and before exiting,
- Useful to time hw overhead in fiq acknowledge and timer programming

Use case: a cash register with S3C440

This prints tickets at 20cm/s, running the whole printer

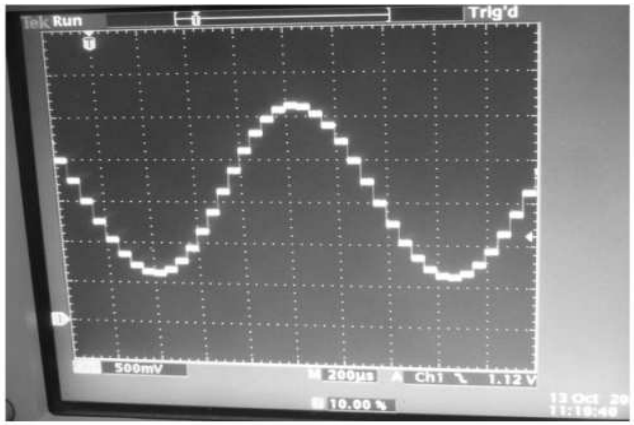


- The top track shows 2 motor coils and spi transfer, driven in fiq context
- The bottom track shows two heaters used in the printer, driven in fiq context

Use case: ADC/DAC without FIFO (PXA270)

This project is meant to test bluetooth amplifiers for safety helmets, 24 of them at a time

- The application must feed 1 DAC and read 2 ADCs every 50 usec



Other uses: PBX, motor control

fiq-engine has been used in a PBX (code written by client)

- Several input lines must leave the CPU in a TDM channel
- The peripheral-to-RAM DMA saves linear buffers
- The RAM-to-peripheral DMA needs interleaved channels
- The FIQ is fired every 64us to shuffle the bytes in memory

In another case, it is used in a motor-control application

- 128us cycle time, to communicate with axis control
- Previously the client used RTAI on x86 and a PCI digital-io board
- Now it's done with FIQ on AT91SAM9263 with GPIO signals

Hard figures: fiq sample code on PXA270

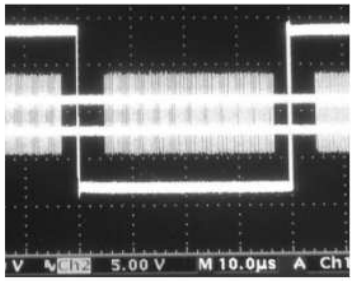
The fiq-busy module has been introduced for diagnostics

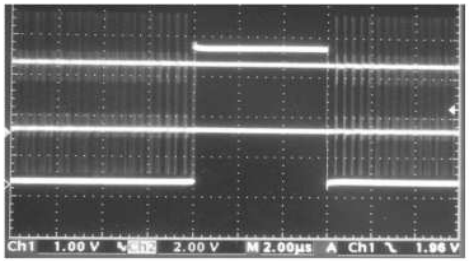
- It continuously toggles a gpio pin from process context ("insmod" process)
- Running fiq-empty on another pin allows to see how things mix

This way, I discovered the PXA270 is horribly slow in its I/O

- It takes 6 usec just to acknoledge fiq and reprogram the timer
- It takes 0.8 usec to move a gpio pin

The figures show fiq-task and fiq-empty with busy.ko running





Live example: STE Nomadik

**At the conference a denostration has been run,
showing fiq-task running on both normal IRQ and FIQ
on the nhk8815 evaluation board for the Nomadik ARM9 cpu**

Hard figures: fiq running on AT91SAM9263

On AT@200MHz it runs at higer rates than on PXA@400MHz

- The sample code at 20 usec is pretty stable
- fiq-empty takes only 0.7 usec to keep things running
- There is, however, a delay on switching modes

The figures show fiq-task at 19usec and fiq-empty

- This also shows a 0.1 usec jitter in duration, for cache effects