Documentation about GPIO Interrupts in PACSAT and Golf software

(Origininally an email sent to Chris VE2TCP)
By Burns Fisher WB1FJ

First let me talk about pin and peripheral numbers.

The code uses port and pin numbers. So for example the current interrupt is on gioPORTB, pin 0. Looking on the Launch Pad document, it calls that GIOB[0], which is on AX5043 pin 126, and Booster Pack J8 pin 8. However, if you ever want to track that on Jim's board, he has named the pins in two ways. What the LaunchPad calls J8 and J9, he also calls both together J5, AND he numbers the pins as though it was a single jack. So to match the LaunchPad docs, you find the matching jack number, and count down. So J8 pin 8 is Jim's J5 pin 15. That was not what you asked, but I wanted to double check everything, so I just documented what I did.

To come close to answering your question, let me trace gioPORTB pin 0, which is Jim's net AX5043_VHF_IRQ back through the software. In drivers/hardwareConfig.h is where the connection is made from the device peripheral names (and those are named by the HAL Code Generator, btw) to the code names. You will see

```
#define GPIO_DCTInterruptPort gioPORTB
```
and
```
#define GPIO_DCTInterruptPin 0
```

So the simple answer for your immediate question is that If you just change GPIO_DCTInterruptPin to 3 that should do what you want immediately (i.e. make the interrupt come from the UHF 5043 rather than the VHF.

But let me trace that through the software so you can see how it works. And I'll post this somewhere too.

==========

Note that DCTInterrupt is the name of a GPIO. So it is used as a prefix for a lot of things like you see above. Each GPIO has a structure called xxxInfo where xxx is the name of the GPIO (or at least the function of the GPIO--I was not always consistent here, I'm sorry to say). in gpioDriver, you will see AX5043InterruptInfo which is initialized with GPIO_DCTInterruptPort and GPIO_DCTInterruptPin to say which actual peripheral we are using. The 1 is how many bits at a time are being used, the next is the default on/off setting if the GPIO is an output (it is not), the next says the GPIO is used for input, the next 2 or the interrupt edge, the next two are opencoollector or tristate (again for output).

NOW, these structures a pointed to by an array of pointers named GPIOInfoStructures, which is indexed by the GPIO name (in this case DCTInterrupt indexes the pointer to

AX5043InterruptInfo (yes they should have the same prefix names!) DCT comes from Golf Digital Command Tranceiver.

When you call GPIOInit specifying DCTInterrupt, we get the info structure and init a bunch of stuff via registers and/or HAL calls. But now we get to
    if(thisGPIO->CanInterrupt && (task != NO_TASK)){
where it puts the message destination and the message name in an array indexed by GPIO number. The aux is in case you want to get a gpio value when the interrupt happens. There is also an array to indicate which GPIOs can accept an interrupt. AND FINALLY we set up the hardware (gioReg->INTDET) and the HAL routine (gioEnableNotification).

So how does this all get an interrupt turned into a message? The interrupt (via a HAL routine) calls gioNotification with the hardware port and pin number. gioNotification searches the list of interrupt possiblities and gets the GPIO name/index and calls GPIOIntRoutine, uses the GPIO index to index the array holding the message name, destination and argument (if there is one) and sends an inter-task message. Note that this is all in an interrupt routine (interrupts block, privs fully on) so you must call the "FromISR" version of NotifyInterTask. Essentially that prevents trying to put a task into a wait state...a FreeRTOS limitation.

==========

So after all that, adding a second interrupt is pretty easy :-) In fact, I can do it most easily probably, although I can't test it, so why don't you tell me when you are ready. But here are the steps.

1) Probably change the GPIO names from DCTInterrupt to DCTRxInterrupt and add DCTTxInterrupt (or UHF or whatever). That will go in the enum in gpioDriver.h
2) Add them to drivers/inc/hardwareConfig.h like

~~#define GPIO_DCTInterruptPort gioPORTB~~
~~#define GPIO_DCTInterruptPin 0~~
#define GPIO_DCTRxInterruptPort gioPORTB
#define GPIO_DCTRxInterruptPin 0
#define GPIO_DCTTxInterruptPort gioPORTB
#define GPIO_DCTTxInterruptPin 3

Add a GPIOInfoStructure in GPIO.c (and change the names in the original one):

```
static const GPIOInfo DCTTxInterruptInfo = {
                    GPIO_DCTTxInterruptPort
                    ,GPIO_DCTTxInterruptPin
                    ,1
```

```
                    ,GPIO_UNUSED // Default off
                    ,GPIO_IN
                    ,true,false //Interrupts one one edge only
                    ,false,false // Not Open collector nor tristate
};
```

And set up to pointer array to match the GPIO name with the GPIO info structure; that is the order matches the GPIO name order in the enum.

```
static const GPIOInfo *GPIOInfoStructures[
{
 &LED1Info,&LED2Info,&,&
};
```