Operating systems for embedded systems

Danilo Moceri (ID: 242550)

January 20, 2017

Assignment #2

A signal analyzer shall be developed that acquires a digital square waveform and computes its period and frequency.

Solution

To solve the assignment, as asked, I have developed a solution that uses the STM32F746DISCO board with uc-linux embedded. In order to reach the goal I have used the timer (TIM12) peripheral which offers the input capture function. To do this, firstly in the developed kernel module, called "sample", I have mapped some peripheral registers using their addresses in memory, an example of this operation it is showed below.

```
struct GpioRegs{

        unsigned int moder;
        unsigned int otyper;
        unsigned int ospeedr;
        unsigned int pupdr;
        unsigned int idr;
        unsigned int odr;
        unsigned int bsrr;
        unsigned int lckr;
        unsigned int afrl;
        unsigned int afrh;
};

volatile struct GpioRegs * const GpioH = (struct GpioRegs *) GPIOH_BASE_ADDR;
volatile struct TimerRegs * const Timer12 = (struct TimerRegs *) TIMER12_BASE_ADDR;
```

To measure the square wave period I have chosen to use a resolution of $1\mu s$, therefore I have set the TIM12 prescaler register to 99, in fact the counter clock frequency can be calculate as:

$$CNT_{ck} = \frac{F_{ck}}{PSC + 1} \tag{1}$$

Where F_{ck} is the timer clock frequency and it is set to $100 \cdot 10^6 Hz$, while PCS is the prescaler register, thus with this configuration the TIM12 has a frequency of

$1MHz$.
The TIM12 is used in interrupt mode through the configuration of some bits in the register DIER, I have chosen to enable the capture interrupt and the update interrupt. In particular the update event has become the overflow event thanks to the configuration of URS bit in TIM12_CR1 reg. In fact considering that the TIM12 has a counter of 16-bits I have managed the overflow event to allow the measuring of slower signal.
To provide this functionality I have used the overflow interrupt event to increment a variable which contains the MSB value, on the other hand when a capture interrupt event occurs the waveform period is stored into a 32-bits variable as Period = (MSB « 16) | TIM12_CCR1 .
The TIM12 interrupt event is managed by the function called timer_handler which is registered through the system call *request_irq()*. Below it is showed the timer interrupt handler body.

```
static irq_handler_t timer_handler( unsigned int irq, struct pt_regs *regs ){

/*      If is an update interrupt (Overflow if UDIS=0 in TIM12_CR1)      */
        if(Timer12->sr & UIF_MASK){

/*              Increments the MSB variables    */
                h_period++;
/*              Resets the interrupt flag               */
                Timer12->sr &= ~UIF_MASK;
        }else{

/*              If is a capture interrupt (Rise edge)    */
                if(Timer12->sr & CC1IF_MASK){

/*                      Saves the period read (MBR<<16 + LSB)   */
                        period = (h_period << 16) | Timer12->ccr1;
/*                      Resets the MBR because it is a new rising edge  */
                        h_period = 0;
                }
        }
        return (irq_handler_t)IRQ_HANDLED;
}
```

In order to allow the communication between the kernel module and the application I have implemented the *ioctl()* and *read()* functions, in particular the *ioctl()* function is used to give start and stop commands to the TIM12 input capture, while the *read()* function provide the last read period measured in μs through a 32-bits variable.
To test the kernel module I have developed a simple application which starts the input capture through the *ioctl()* function, after this operation an infinite loop prints the square wave period obtained by the function *read()*. In this case I have chosen to update period and frequency value every $500\mu s$. Finally the ctrl-c click allows the input capture stopping through the function *ioctl()*.

Conclusion

Theoretically with the adopted solution it is possible to analyze square wave with a frequency that goes from few Hertz to $1MHz$ with a precision of $1\mu s$, however it is not really possible.
In fact during the tests it has been seen that the highest frequency measurable is around $200KHz$, anyhow this bond can be overcome decreasing the prescaler but this choose has been discarded in this work.