Operating Systems for Embedded Systems

Student: Piergiovanni Ferrara - ID: 242549

January 20, 2017

Target of Assignment #2
A signal analyzer shall be developed that acquires a digital square waveform and computes its period and frequency.

Implementation

Using the STM32F746DISCO board with uc-linux embedded OS, it has been chosen to use the TIM12 general-purpose timer and the input-capture-mode available on it.
The goal was to get the digital square waveform as input and therefore the GPIO on PORTH.6 was configured to enable alternate-function-mode.

Thanks to the initialization of the kernel module, the clocks of TIM12 and GPIO are enabled correctly. After that, TIM12 registers are configured to enable the counter, set the prescaler and configure the channel in input mode on TI1. Also, this timer is setup to capture interrupts and the update event is the overflow condition. Obviously, pointers to addresses are used in order to update or read bits inside many registers as it is showed in the following code lines:

```
/* Pointers to RCC registers */
unsigned int volatile * const rcc_ah = (unsigned int *)(RCC_BASEADDRESS+RCC_AHB1ENR);
unsigned int volatile * const rcc_ap = (unsigned int *)(RCC_BASEADDRESS+RCC_APB1ENR);

/* Pointers to GPIO registers */
unsigned int volatile * const portH_MODER = (unsigned int *)(PORTH_BASEADDRESS+
    PORTH6_MODER);
unsigned int volatile * const portH_AFRL = (unsigned int *)(PORTH_BASEADDRESS+
    PORTH6_AFRL);

/* Pointers to TIM12 registers */
unsigned short volatile * const tim12_PSC = (unsigned short *)(TIM12_BASEADDRESS+
    TIM12_PSC);
unsigned short volatile * const tim12_CR1 = (unsigned short *)(TIM12_BASEADDRESS+
    TIM12_CR1);
unsigned short volatile * const tim12_SMCR = (unsigned short *)(TIM12_BASEADDRESS+
    TIM12_SMCR);
```

```
unsigned short volatile * const tim12_SR = (unsigned short *)(TIM12_BASEADDRESS+
    TIM12_SR);
unsigned short volatile * const tim12_DIER = (unsigned short *)(TIM12_BASEADDRESS+
    TIM12_DIER);
unsigned short volatile * const tim12_EGR = (unsigned short *)(TIM12_BASEADDRESS+
    TIM12_EGR);
unsigned short volatile * const tim12_CCMR1 = (unsigned short *)(TIM12_BASEADDRESS+
    TIM12_CCMR1);
unsigned short volatile * const tim12_CCER = (unsigned short *)(TIM12_BASEADDRESS+
    TIM12_CCER);
unsigned short volatile * const tim12_COUNTER_VAL = (unsigned short *)(
    TIM12_BASEADDRESS+TIM12_COUNTER_VAL);
```

Prescaler (TIM12_PSC) value is set to 99 just at the beginning of the configuration and in this way the clock frequency of the TIM12 becomes:

$$Clkf_{timer} = \frac{f_{clk}}{PSC[15:0] + 1} \tag{1}$$

where f_{clk} is equal to $100MHz$ that is the clock frequency of the peripherals and PSC is a 16-bits register. Doing this, the counter clock frequency is set to $1MHz$ and a corresponding resolution of $1\mu s$.

When a rising edge is detected, an interrupt is raised and subsequently managed by a handler function. Inside it, if an overflow is occurred, a bit flag is reset into the register and a variable is incremented. The latter is used to get the MSB value. In the other case, that variable is shifted by 16 and then added to the read counter value from TIM12_CCR1 register. The new period value is stored into a 32-bits variable after the above operation.

The communication between user space and kernel space is possible by using system calls.
It has been written an application with a while loop in which the period value is delivered by invoking the $read\ function$. Thus, the period and the frequency computed are showed constantly and measured in ms and Hz respectively.

To summarize, it has been preferred to use a hardware solution that is able to provide more accurate measurements than the ones achievable using a software way. However, this implementation can estimate square waveforms with a resolution not quite close to $1\mu s$ but you can determine hundreds of KHz.