Міністерство освіти і науки України

Центральноукраїнський національний технічний університет

**МЕХАНІКО-ТЕХНОЛОГІЧНИЙ ФАКУЛЬТЕТ**

Кафедра програмування та захисту інформації

**Звіт**

з виконаної лабораторної роботи № 4

дисципліни Програмування вбудованих систем”

на тему

«**Мікроконтролери ARM STM32F4xx, апаратне генерування ШІМ**

.»

Виконав :

студент академічної групи КІ-15

Аннаєв А. \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

Перевірив :

Викладач

Савеленко О.К.\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

Кропивницкий- 2017

**Мета:** Навчитися виконувати апаратне генерування ШІМ в мікроконтролерах ARM STM32F4xx.

**Завдання:**

1) Створити програмне забезпечення для мікроконтролеру, яке керуватиме світлодіодами на основі апаратного генерування ШІМ.

2) Досягти незалежну зміну яскравості кількох світлодіодів.

Хід роботи

1. Програма:

#include <stm32f4xx.h>

#include <stm32f4xx\_rcc.h>

#include <stm32f4xx\_gpio.h>

#include <stm32f4xx\_tim.h>

#include <misc.h>

////////////////////////////////////////////////////////////////////////////

void init\_gpio();

void init\_pwm(); // timer 4

void init\_blink(); // timer 6

TIM\_OCInitTypeDef oc\_init;

////////////////////////////////////////////////////////////////////////////

int main(void)

{

SystemCoreClockUpdate();

init\_gpio();

init\_pwm();

init\_blink();

while(1);

return 0;

}

//==========================================================================

void init\_gpio()

{

RCC\_AHB1PeriphClockCmd(RCC\_AHB1Periph\_GPIOD,ENABLE);

RCC\_APB1PeriphClockCmd(RCC\_APB1Periph\_TIM4, ENABLE);

GPIO\_InitTypeDef init;

init.GPIO\_Mode = GPIO\_Mode\_AF;

init.GPIO\_OType = GPIO\_OType\_PP;

init.GPIO\_Pin = GPIO\_Pin\_12 | GPIO\_Pin\_13 | GPIO\_Pin\_14 | GPIO\_Pin\_15;

init.GPIO\_Speed = GPIO\_Speed\_100MHz;

GPIO\_Init(GPIOD,&init);

GPIO\_PinAFConfig(GPIOD,GPIO\_PinSource12,GPIO\_AF\_TIM4);

GPIO\_PinAFConfig(GPIOD,GPIO\_PinSource13,GPIO\_AF\_TIM4);

GPIO\_PinAFConfig(GPIOD,GPIO\_PinSource14,GPIO\_AF\_TIM4);

GPIO\_PinAFConfig(GPIOD,GPIO\_PinSource15,GPIO\_AF\_TIM4);

}

//==========================================================================

void init\_pwm()

{

TIM\_TimeBaseInitTypeDef base\_timer;

TIM\_TimeBaseStructInit(&base\_timer);

base\_timer.TIM\_ClockDivision = TIM\_CKD\_DIV1;

base\_timer.TIM\_Prescaler = SystemCoreClock / 1000000 - 1;

base\_timer.TIM\_Period = 900;

base\_timer.TIM\_CounterMode = TIM\_CounterMode\_Up;

base\_timer.TIM\_RepetitionCounter = 0;

TIM\_TimeBaseInit(TIM4, &base\_timer);

TIM\_OCStructInit(&oc\_init);

oc\_init.TIM\_OCMode = TIM\_OCMode\_PWM1;

oc\_init.TIM\_OutputState = TIM\_OutputState\_Enable;

oc\_init.TIM\_OCPolarity = TIM\_OCPolarity\_High;

// D12

oc\_init.TIM\_Pulse = 0;

TIM\_OC1Init(TIM4,&oc\_init);

TIM\_OC1PreloadConfig(TIM4, TIM\_OCPreload\_Enable);

// D13

oc\_init.TIM\_Pulse = 0;

TIM\_OC2Init(TIM4,&oc\_init);

TIM\_OC2PreloadConfig(TIM4, TIM\_OCPreload\_Enable);

// D14

oc\_init.TIM\_Pulse = 0;

TIM\_OC3Init(TIM4,&oc\_init);

TIM\_OC3PreloadConfig(TIM4, TIM\_OCPreload\_Enable);

// D15

oc\_init.TIM\_Pulse = 0;

TIM\_OC4Init(TIM4,&oc\_init);

TIM\_OC4PreloadConfig(TIM4, TIM\_OCPreload\_Enable);

TIM\_ARRPreloadConfig(TIM4,ENABLE);

TIM\_Cmd(TIM4,ENABLE);

//NVIC\_EnableIRQ(TIM4\_IRQn);

}

//==========================================================================

void init\_blink()

{

RCC\_APB1PeriphClockCmd(RCC\_APB1Periph\_TIM6, ENABLE);

TIM\_TimeBaseInitTypeDef base\_timer;

TIM\_TimeBaseStructInit(&base\_timer);

base\_timer.TIM\_Prescaler = SystemCoreClock / 1000000 -1;

base\_timer.TIM\_Period = 9000;

TIM\_TimeBaseInit(TIM6, &base\_timer);

TIM\_ITConfig(TIM6, TIM\_IT\_Update, ENABLE);

TIM\_Cmd(TIM6, ENABLE); // Включаем таймер

NVIC\_EnableIRQ(TIM6\_DAC\_IRQn);

}

//==========================================================================

void TIM6\_DAC\_IRQHandler()

{

static int brightness = 0;

if (TIM\_GetITStatus(TIM6, TIM\_IT\_Update) != RESET)

{

TIM\_ClearITPendingBit(TIM6, TIM\_IT\_Update);

brightness = (brightness + 3) % 900;

// Led D12

oc\_init.TIM\_Pulse = (brightness + 0) % 900;

TIM\_OC1Init(TIM4,&oc\_init);

TIM\_OC1PreloadConfig(TIM4, TIM\_OCPreload\_Enable);

// Led D13

oc\_init.TIM\_Pulse = (brightness + 300) % 900;

TIM\_OC2Init(TIM4,&oc\_init);

TIM\_OC2PreloadConfig(TIM4, TIM\_OCPreload\_Enable);

// Led D14

oc\_init.TIM\_Pulse = (brightness + 500) % 900;

TIM\_OC3Init(TIM4,&oc\_init);

TIM\_OC3PreloadConfig(TIM4, TIM\_OCPreload\_Enable);

// Led D15

oc\_init.TIM\_Pulse = (brightness + 700) % 900;

TIM\_OC4Init(TIM4,&oc\_init);

TIM\_OC4PreloadConfig(TIM4, TIM\_OCPreload\_Enable);

}

}

////////////////////////////////////////////////////////////////////////////

**Висновки:**. Навчилися виконувати апаратне генерування ШІМ в мікроконтролерах ARM STM32F4xx.