Міністерство освіти і науки України

Центральноукраїнський національний технічний університет

**МЕХАНІКО-ТЕХНОЛОГІЧНИЙ ФАКУЛЬТЕТ**

Кафедра програмування та захисту інформації

**Звіт**

з виконаної лабораторної роботи № 7

дисципліни Програмування вбудованих систем”

на тему

«**Мікроконтролери ARM STM32F4xx,**

**використання диспетчера задач**»

Виконав :

студент академічної групи КІ-15

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

Перевірив :

Викладач

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

Кропивницкий- 2017

**Мета:** Навчитися освоювати диспетчер задач ARM STM32F4xx.

**Завдання:**

1)Створити програмне забезпечення для мікроконтролеру, яке виконує дві задачі в режимі почергового виконання.

2) Забезпечити незалежне керування світлодіодами в окремих задачах, щоб пересвідчитись в незалежності окремих задач.

Хід роботи

1. Програма:

/\*---------------------------- Include ---------------------------------------\*/

#include <stm32f4xx.h>

#include <stm32f4xx\_rcc.h>

#include <stm32f4xx\_gpio.h>

#include <stm32f4xx\_tim.h>

#include <stm32f4xx\_adc.h>

#include <misc.h>

#include <CoOS.h> /\*!< CoOS header file \*/

/\*---------------------------- Symbol Define -------------------------------\*/

#define STACK\_SIZE\_TASKA 128 /\*!< Define "taskA" task size \*/

#define STACK\_SIZE\_TASKB 128 /\*!< Define "taskA" task size \*/

#define STACK\_SIZE\_TASKC 128 /\*!< Define "taskA" task size \*/

#define PWM\_PERIOD 4096

#define V\_DIVISOR (3.3 / (double)PWM\_PERIOD)

/\*---------------------------- Variable Define -------------------------------\*/

OS\_STK taskA\_stk[STACK\_SIZE\_TASKA]; /\*!< Define "taskA" task stack \*/

OS\_STK taskB\_stk[STACK\_SIZE\_TASKB]; /\*!< Define "taskB" task stack \*/

OS\_STK taskC\_stk[STACK\_SIZE\_TASKC]; /\*!< Define "led" task stack \*/

/\*\*

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

\* @brief "taskA" task code

\* @param[in] None

\* @param[out] None

\* @retval None

\* @par Description

\* @details This task use to crate mutex and flags,print message "taskA running".

\* Indicate "taskA" had been executed.

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

\*/

void init\_leds();

void init\_ADC();

void init\_timer(int period);

void init\_ADC\_pin();

void init\_PWM\_channel();

u16 read\_ADC1(u8 channel);

void taskA (void\* pdata)

{

init\_leds();

while (1)

{

CoTickDelay (50);

init\_leds();

GPIO\_SetBits(GPIOD,GPIO\_Pin\_14);

CoTickDelay (50);

init\_leds();

GPIO\_ResetBits(GPIOD,GPIO\_Pin\_14);

}

}

/\*\*

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

\* @brief "taskB" task code

\* @param[in] None

\* @param[out] None

\* @retval None

\* @par Description

\* @details This task use to print message "taskB running". Indicate "taskB"

\* had been executed.

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

\*/

void taskB (void\* pdata)

{

while (1)

{

CoTickDelay (100);

init\_leds();

GPIO\_SetBits(GPIOD,GPIO\_Pin\_15);

CoTickDelay (100);

init\_leds();

GPIO\_ResetBits(GPIOD,GPIO\_Pin\_15);

}

}

/\*\*

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

\* @brief "taskB" task code

\* @param[in] None

\* @param[out] None

\* @retval None

\* @par Description

\* @details This task use to print message "taskB running". Indicate "taskB"

\* had been executed.

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

\*/

void taskC (void\* pdata)

{

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

GPIO\_InitTypeDef gpioStructure;

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

gpioStructure.GPIO\_Mode = GPIO\_Mode\_AIN;

gpioStructure.GPIO\_Speed = GPIO\_Speed\_50MHz;

GPIO\_Init(GPIOD, &gpioStructure);\*/

init\_ADC();

init\_timer(PWM\_PERIOD);

init\_PWM\_channel();

while(1)

{

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

GPIO\_InitTypeDef gpioStructure;

gpioStructure.GPIO\_Pin = GPIO\_Pin\_12;

gpioStructure.GPIO\_Mode = GPIO\_Mode\_AF;

gpioStructure.GPIO\_Speed = GPIO\_Speed\_50MHz;

GPIO\_Init(GPIOD, &gpioStructure);

}

}

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

int main(void)

{

SystemCoreClockUpdate();

CoInitOS (); /\*!< Initial CooCox CoOS \*/

/\*!< Create three tasks \*/

CoCreateTask (taskA,0,0,&taskA\_stk[STACK\_SIZE\_TASKA-1],STACK\_SIZE\_TASKA);

CoCreateTask (taskB,0,1,&taskB\_stk[STACK\_SIZE\_TASKB-1],STACK\_SIZE\_TASKB);

CoCreateTask (taskC,0,2,&taskC\_stk[STACK\_SIZE\_TASKC-1],STACK\_SIZE\_TASKC);

CoStartOS (); /\*!< Start multitask \*/

while(1)

{

}

}

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

void init\_leds()

{

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

GPIO\_InitTypeDef gpio;

GPIO\_StructInit(&gpio);

//gpio.GPIO\_OType = GPIO\_OType\_PP;

gpio.GPIO\_Mode = GPIO\_Mode\_OUT;

gpio.GPIO\_Pin = GPIO\_Pin\_14 | GPIO\_Pin\_15;

GPIO\_Init(GPIOD, &gpio);

}

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

void init\_timer(int period)

{

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

TIM\_TimeBaseInitTypeDef timerInitStructure;

timerInitStructure.TIM\_Prescaler = SystemCoreClock / 1000000 - 1;

timerInitStructure.TIM\_CounterMode = TIM\_CounterMode\_Up;

timerInitStructure.TIM\_Period = period;

timerInitStructure.TIM\_ClockDivision = TIM\_CKD\_DIV1;

timerInitStructure.TIM\_RepetitionCounter = 0;

TIM\_TimeBaseInit(TIM4, &timerInitStructure);

TIM\_Cmd(TIM4, ENABLE);

}

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

void init\_PWM\_channel()

{

unsigned int binCode = read\_ADC1(ADC\_Channel\_1);

TIM\_OCInitTypeDef outputChannelInit = {0, };

outputChannelInit.TIM\_OCMode = TIM\_OCMode\_PWM1;

// 3.3V = 4096, PWM\_PERIOD must be 4096

outputChannelInit.TIM\_Pulse = (int)(((float)binCode /

V\_DIVISOR))%PWM\_PERIOD;

outputChannelInit.TIM\_OutputState = TIM\_OutputState\_Enable;

outputChannelInit.TIM\_OCPolarity = TIM\_OCPolarity\_High;

TIM\_OC1Init(TIM4, &outputChannelInit);

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

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

}

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

void init\_ADC\_pin()

{

GPIO\_InitTypeDef gpio;

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

GPIO\_StructInit(&gpio);

gpio.GPIO\_Mode = GPIO\_Mode\_AN;

gpio.GPIO\_Pin = GPIO\_Pin\_1 | GPIO\_Pin\_2;

GPIO\_Init(GPIOA, &gpio);

}

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

void init\_ADC()

{

init\_ADC\_pin();

ADC\_InitTypeDef ADC\_InitStructure;

ADC\_CommonInitTypeDef adc\_init;

RCC\_APB2PeriphClockCmd(RCC\_APB2Periph\_ADC1, ENABLE);

ADC\_DeInit();

ADC\_StructInit(&ADC\_InitStructure);

adc\_init.ADC\_Mode = ADC\_Mode\_Independent;

adc\_init.ADC\_Prescaler = ADC\_Prescaler\_Div2;

ADC\_InitStructure.ADC\_ScanConvMode = DISABLE;

ADC\_InitStructure.ADC\_ContinuousConvMode = DISABLE;

ADC\_InitStructure.ADC\_ExternalTrigConv = ADC\_ExternalTrigConvEdge\_None;

ADC\_InitStructure.ADC\_DataAlign = ADC\_DataAlign\_Right;

ADC\_InitStructure.ADC\_Resolution = ADC\_Resolution\_12b;

ADC\_CommonInit(&adc\_init);

ADC\_Init(ADC1, &ADC\_InitStructure);

ADC\_Cmd(ADC1, ENABLE);

}

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

u16 read\_ADC1(u8 channel)

{

ADC\_RegularChannelConfig(ADC1, channel, 1, ADC\_SampleTime\_3Cycles);

ADC\_SoftwareStartConv(ADC1);

while (ADC\_GetFlagStatus(ADC1, ADC\_FLAG\_EOC) == RESET);

return ADC\_GetConversionValue(ADC1);

}

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

**Висновки:**.Навчилися освоювати диспетчер задач ARM STM32F4xx.