دانشگاه صنعتی امیرکبیر

دانشکده مهندسی پزشکی

گزارش آزمایشگاه میکروپروسسور

آزمایش ۳: منابع پالس ساعت و کنترل توان

گروه ۲:

حمیدرضا ابوئی مهریزی

Click here to enter text.

تاریخ اتمام آزمایش: ۱۴۰۱/۰۲/۲۲

آن چه از این آزمایش فرا گرفته‌اید:

مقدمه

پرسش‌ها و آزمایش‌ها

بخش اول

؟ سوأل ۱:

حلقه قفل فاز (Phase-Locked Loop) یا به اختصار PLL یک سیستم کنترلی است که کاربردهای فراوانی در الکترونیک و مخابرات دارد. در واقع، حلقه قفل فاز می‌تواند یک سیگنال خروجی را تولید کند که فاز آن به فاز سیگنال ورودی وابسته است. حلقه قفل فاز انواع مختلفی دارد، اما ساده‌ترین نوع آن، یک مدار الکترونیکی است که از یک اسیلاتور فرکانس متغیر (Variable Frequency Oscillator) و یک آشکارساز فاز (Phase Detector) در حلقه فیدبک تشکیل شده است. اسیلاتور در این مدار، یک سیگنال متناوب تولید می‌کند و آشکارساز فاز، مسئول مقایسه فاز آن سیگنال با فاز سیگنال متناوب ورودی است و در نهایت، باید اسیلاتور برای حفظ تطبیق فاز تنظیم شود.

؟ سوأل ۲:

با فرض این که مقادیر پیش‌فرض، روی رجیسترها مقدار 0x0000xx83 است، بنابراین ورودی فرکانس از HSI می آید و فعال است. تمام prescaler ها خاموش است و فرکانس آن برابر است با فرکانس ۱۶MHz.

7.3.1 RCC clock control register (RCC_CR)

Address offset: 0x00

Reset value: 0x0000 XX83 where X is undefined.

Access: no wait state, word, half-word and byte access

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Reserved				PLLI2SRDY	PLLI2SON	PLLRDY	PLLON	Reserved				CSSON	HSEBYP	HSERDY	HSEON
				r	rw	r	rw					rw	rw	r	rw

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
HSICAL[7:0]								HSITRIM[4:0]					Res.	HSIRDY	HSION
r	r	r	r	r	r	r	r	rw	rw	rw	rw	rw		r	rw

Bits 31:28 Reserved, must be kept at reset value.

Bit 27 **PLLI2SRDY**: PLLI2S clock ready flag
Set by hardware to indicate that the PLLI2S is locked.
0: PLLI2S unlocked
1: PLLI2S locked

Bit 26 **PLLI2SON**: PLLI2S enable
Set and cleared by software to enable PLLI2S.
Cleared by hardware when entering Stop or Standby mode.
0: PLLI2S OFF
1: PLLI2S ON

Bit 25 **PLLRDY**: Main PLL (PLL) clock ready flag
Set by hardware to indicate that PLL is locked.
0: PLL unlocked
1: PLL locked

Bit 24 **PLLON**: Main PLL (PLL) enable
Set and cleared by software to enable PLL.
Cleared by hardware when entering Stop or Standby mode. This bit cannot be reset if PLL clock is used as the system clock.
0: PLL OFF
1: PLL ON

Bits 23:20 Reserved, must be kept at reset value.

Bit 19 **CSSON**: Clock security system enable
Set and cleared by software to enable the clock security system. When CSSON is set, the clock detector is enabled by hardware when the HSE oscillator is ready, and disabled by hardware if an oscillator failure is detected.
0: Clock security system OFF (Clock detector OFF)
1: Clock security system ON (Clock detector ON if HSE oscillator is stable, OFF if not)

Bit 18 **HSEBYP**: HSE clock bypass

Set and cleared by software to bypass the oscillator with an external clock. The external clock must be enabled with the HSEON bit, to be used by the device.
The HSEBYP bit can be written only if the HSE oscillator is disabled.
0: HSE oscillator not bypassed
1: HSE oscillator bypassed with an external clock

Bit 17 **HSERDY**: HSE clock ready flag

Set by hardware to indicate that the HSE oscillator is stable. After the HSEON bit is cleared, HSERDY goes low after 6 HSE oscillator clock cycles.
0: HSE oscillator not ready
1: HSE oscillator ready

Bit 16 **HSEON**: HSE clock enable

Set and cleared by software.
Cleared by hardware to stop the HSE oscillator when entering Stop or Standby mode. This bit cannot be reset if the HSE oscillator is used directly or indirectly as the system clock.
0: HSE oscillator OFF
1: HSE oscillator ON

Bits 15:8 **HSICAL[7:0]**: Internal high-speed clock calibration

These bits are initialized automatically at startup.

Bits 7:3 **HSITRIM[4:0]**: Internal high-speed clock trimming

These bits provide an additional user-programmable trimming value that is added to the HSICAL[7:0] bits. It can be programmed to adjust to variations in voltage and temperature that influence the frequency of the internal HSI RC.

Bit 2 Reserved, must be kept at reset value.

Bit 1 **HSIRDY**: Internal high-speed clock ready flag

Set by hardware to indicate that the HSI oscillator is stable. After the HSION bit is cleared, HSIRDY goes low after 6 HSI clock cycles.
0: HSI oscillator not ready
1: HSI oscillator ready

Bit 0 **HSION**: Internal high-speed clock enable

Set and cleared by software.
Set by hardware to force the HSI oscillator ON when leaving the Stop or Standby mode or in case of a failure of the HSE oscillator used directly or indirectly as the system clock. This bit cannot be cleared if the HSI is used directly or indirectly as the system clock.
0: HSI oscillator OFF
1: HSI oscillator ON

7.3.3 RCC clock configuration register (RCC_CFGR)

Address offset: 0x08

Reset value: 0x0000 0000

Access: 0 ≤ wait state ≤ 2, word, half-word and byte access

1 or 2 wait states inserted only if the access occurs during a clock source switch.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
MCO2		MCO2 PRE[2:0]			MCO1 PRE[2:0]			I2SSCR	MCO1		RTCPRE[4:0]				
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
PPRE2[2:0]			PPRE1[2:0]			Reserved		HPRE[3:0]				SWS1	SWS0	SW1	SW0
rw	rw	rw	rw	rw	rw			rw	rw	rw	rw	r	r	rw	rw

سوأل ۳:

سوأل ۴:

تحقیق ۱:

6.2.7 Clock security system (CSS)

The clock security system can be activated by software. In this case, the clock detector is enabled after the HSE oscillator startup delay, and disabled when this oscillator is stopped.

If a failure is detected on the HSE clock, this oscillator is automatically disabled, a clock failure event is sent to the break inputs of advanced-control timers TIM1 and TIM8, and an interrupt is generated to inform the software about the failure (clock security system interrupt CSSI), allowing the MCU to perform rescue operations. The CSSI is linked to the Cortex®-M4 with FPU NMI (non-maskable interrupt) exception vector.

Note: *When the CSS is enabled, if the HSE clock happens to fail, the CSS generates an interrupt, which causes the automatic generation of an NMI. The NMI is executed indefinitely unless the CSS interrupt pending bit is cleared. As a consequence, the application has to clear the CSS interrupt in the NMI ISR by setting the CSSC bit in the Clock interrupt register (RCC_CIR).*

If the HSE oscillator is used directly or indirectly as the system clock (indirectly meaning that it is directly used as PLL input clock, and that PLL clock is the system clock) and a failure is detected, then the system clock switches to the HSI oscillator and the HSE oscillator is disabled.

If the HSE oscillator clock was the clock source of PLL used as the system clock when the failure occurred, PLL is also disabled. In this case, if the PLLI2S was enabled, it is also disabled when the HSE fails.

امنیت سیستم کلاک:

این سیستم می‌تواند به صورت نرم‌افزاری روشن می‌شود و در صورت فعال شدن، آشکار ساز کلاک بعد از گذر از تاخیر شروع HSE فعال می‌شود. این بیت با استفاده از

تحقیق ۲:

این بیت مربوط به peripheral clock enable register است و در صورتی که مقدار دهی نشود، کلاک تمام امکانات جانبی سیستم از جمله پورت‌های ورودی/خروجی و تایمرها خاموش خواهد بود و برای فعال کردن آن‌ها باید این رجیسترهای بخش RCC فعال شود.

همان‌طور که در شکل‌های زیر (که از refrence manual) برگرفته شده است ملاحظه می‌کنید، این دو سری رجیستر APB1END و AHB1ENR به ترتیب کلاک‌های قسمت تایمرها و پورت‌های ورودی/خروجی را مدیریت می‌کند.

6.3.13 RCC APB1 peripheral clock enable register (RCC_APB1ENR)

Address offset: 0x40

Reset value: 0x0000 0000

Access: no wait state, word, half-word and byte access.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
UART8 EN	UART7 EN	DAC EN	PWR EN	Reser-ved	CAN2 EN	CAN1 EN	Reser-ved	I2C3 EN	I2C2 EN	I2C1 EN	UART5 EN	UART4 EN	USART 3 EN	USART 2 EN	Reser-ved
rw	rw	rw	rw		rw	rw		rw	rw	rw	rw	rw	rw	rw	
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
SPI3 EN	SPI2 EN	Reserved		WWDG EN	Reserved		TIM14 EN	TIM13 EN	TIM12 EN	TIM7 EN	TIM6 EN	TIM5 EN	TIM4 EN	TIM3 EN	TIM2 EN
rw	rw			rw			rw	rw	rw	rw	rw	rw	rw	rw	rw

7.3.10 RCC AHB1 peripheral clock enable register (RCC_AHB1ENR)

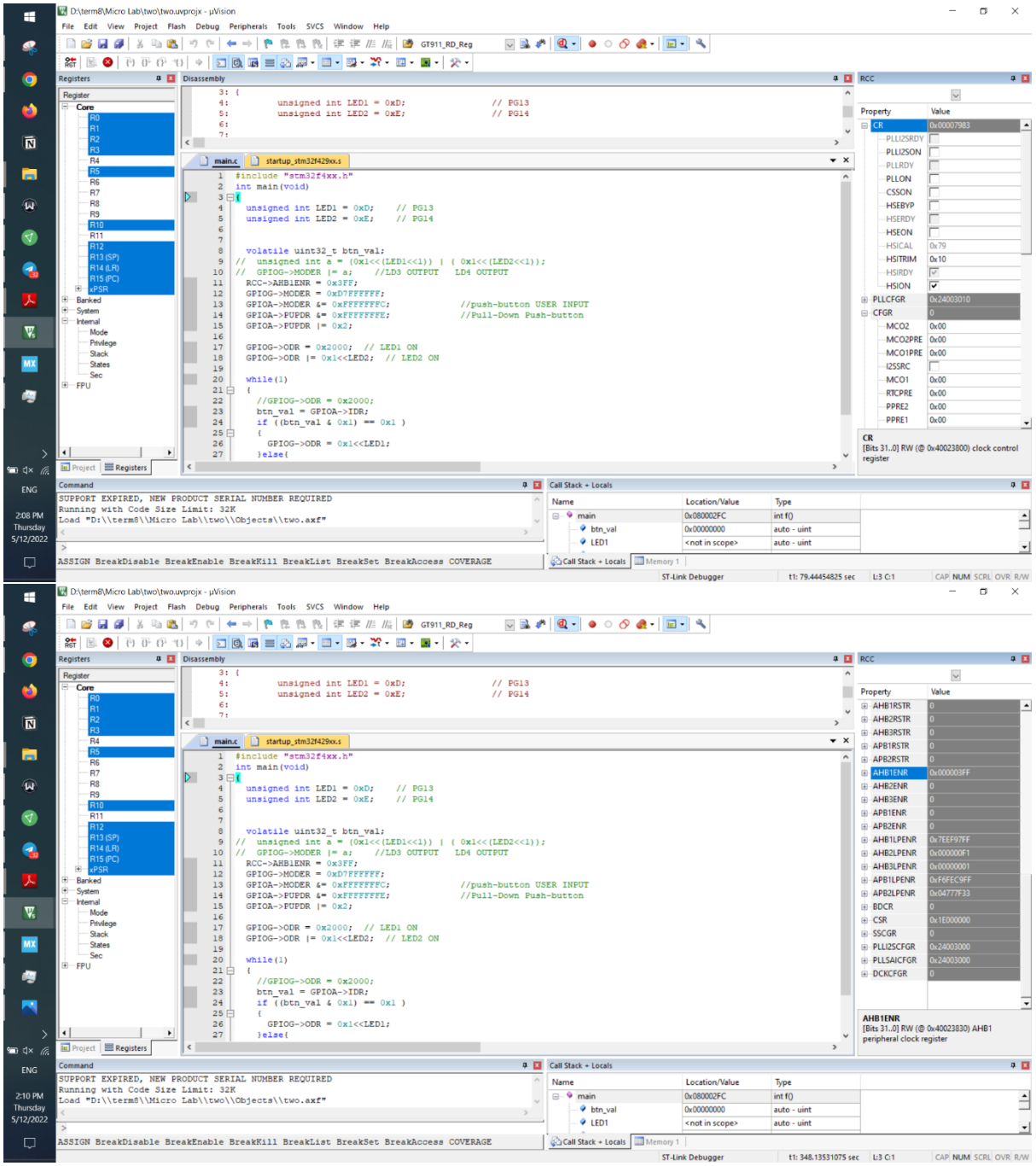
Address offset: 0x30

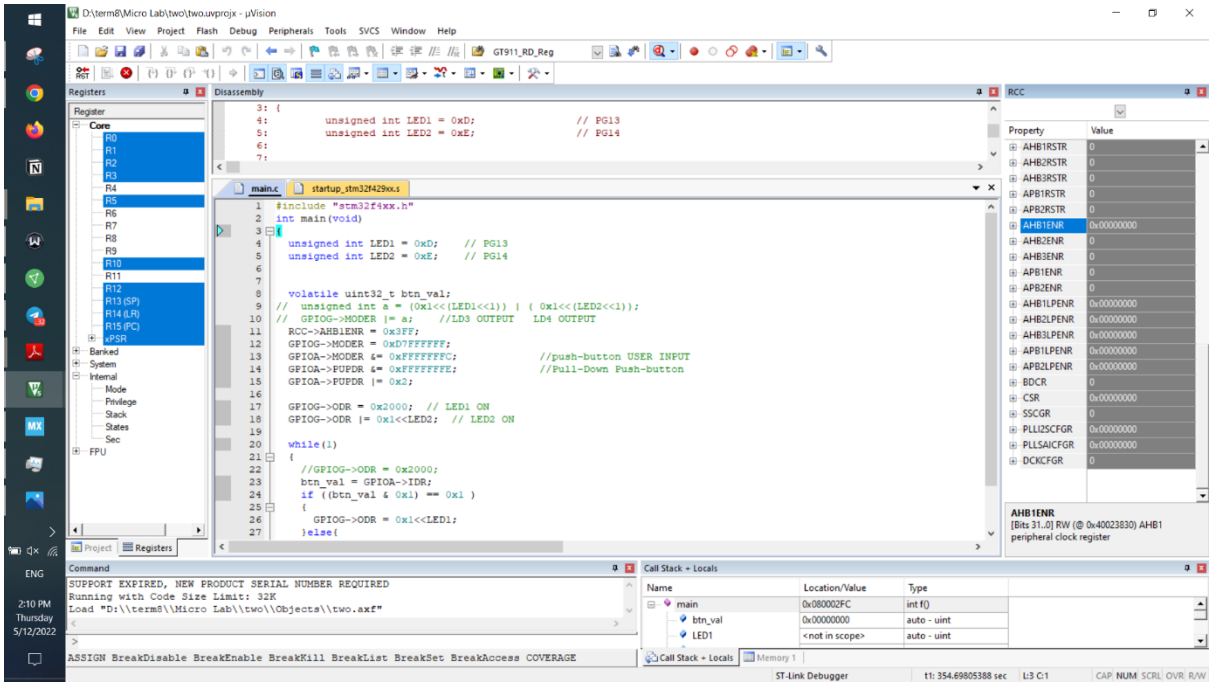
Reset value: 0x0010 0000

Access: no wait state, word, half-word and byte access.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Reser-ved	OTGH S ULPIE N	OTGH SEN	ETHM ACPTP EN	ETHM ACRXE N	ETHM ACTXE N	ETHMA CEN	Reserved		DMA2E N	DMA1E N	CCMDAT ARAMEN	Res.	BKPSR AMEN	Reserved	
	rw	rw	rw	rw	rw	rw			rw	rw			rw		
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved			CRCE N	Reserved			GPIOIE N	GPIOH EN	GPIOG EN	GPIOFE N	GPIOEEN	GPIOD EN	GPIOC EN	GPIO BEN	GPIO AEN
			rw				rw	rw	rw	rw	rw	rw	rw	rw	rw

آزمایش ۱:


D:\term8\Micro Lab\two\two.uvprojx - µVision
File Edit View Project Flash Debug Peripherals Tools SVCS Window Help
GT911_RD_Reg
Registers
Register
Core
R0
R1
R2
R3
R4
R5
R6
R7
R8
R9
R10
R11
R12
R13 (SP)
R14 (LR)
R15 (PC)
xPSR
Banked
System
Internal
Mode
Privilege
Stack
States
Sec
FPU
Disassembly
3: {
4: unsigned int LED1 = 0xD; // PG13
5: unsigned int LED2 = 0xE; // PG14
6:
7:
main.c
startup_stm32f429xx.s
1 #include "stm32f4xx.h"
2 int main(void)
3 {
4 unsigned int LED1 = 0xD; // PG13
5 unsigned int LED2 = 0xE; // PG14
6
7
8 volatile uint32_t btn_val;
9 // unsigned int a = (0x1<<(LED1<<1)) | (0x1<<(LED2<<1));
10 // GPIOG->MODER |= a; //LD3 OUTPUT LD4 OUTPUT
11 RCC->AHB1ENR = 0x3FF;
12 GPIOG->MODER = 0xD7FFFFFF;
13 GPIOA->MODER &= 0xFFFFFFFC; //push-button USER INPUT
14 GPIOA->PUPDR &= 0xFFFFFFFE; //Pull-Down Push-button
15 GPIOA->PUPDR |= 0x2;
16
17 GPIOG->ODR = 0x2000; // LED1 ON
18 GPIOG->ODR |= 0x1<<LED2; // LED2 ON
19
20 while(1)
21 {
22 //GPIOG->ODR = 0x2000;
23 btn_val = GPIOA->IDR;
24 if ((btn_val & 0x1) == 0x1)
25 {
26 GPIOG->ODR = 0x1<<LED1;
27 }else{
RCC
Property
Value
CR
0x00007983
PLLI2SRDY
PLLI2SON
PLLRDY
PLLON
CSSON
HSEBYP
HSERDY
HSEON
HSICAL
0x79
HSITRIM
0x10
HSIRDY
HSION
PLLCFGR
0x24003010
CFGR
0
MCO2
0x00
MCO2PRE
0x00
MCO1PRE
0x00
I2SSRC
MCO1
0x00
RTCPRE
0x00
PPRE2
0x00
PPRE1
0x00
CR
[Bits 31..0] RW (@ 0x40023800) clock control register
Project
Registers
Command
SUPPORT EXPIRED, NEW PRODUCT SERIAL NUMBER REQUIRED
Running with Code Size Limit: 32K
Load "D:\\term8\\Micro Lab\\two\\Objects\\two.axf"
ASSIGN BreakDisable BreakEnable BreakKill BreakList BreakSet BreakAccess COVERAGE
Call Stack + Locals
Name
Location/Value
Type
main
0x080002FC
int f()
btn_val
0x00000000
auto - uint
LED1
<not in scope>
auto - uint
Call Stack + Locals
Memory 1
ST-Link Debugger
t1: 79.44454825 sec
L:3 C:1
CAP NUM SCRL OVR R/W
ENG
2:08 PM
Thursday
5/12/2022
D:\term8\Micro Lab\two\two.uvprojx - µVision
File Edit View Project Flash Debug Peripherals Tools SVCS Window Help
GT911_RD_Reg
Registers
Disassembly
3: {
4: unsigned int LED1 = 0xD; // PG13
5: unsigned int LED2 = 0xE; // PG14
6:
7:
main.c
startup_stm32f429xx.s
1 #include "stm32f4xx.h"
2 int main(void)
3 {
4 unsigned int LED1 = 0xD; // PG13
5 unsigned int LED2 = 0xE; // PG14
6
7
8 volatile uint32_t btn_val;
9 // unsigned int a = (0x1<<(LED1<<1)) | (0x1<<(LED2<<1));
10 // GPIOG->MODER |= a; //LD3 OUTPUT LD4 OUTPUT
11 RCC->AHB1ENR = 0x3FF;
12 GPIOG->MODER = 0xD7FFFFFF;
13 GPIOA->MODER &= 0xFFFFFFFC; //push-button USER INPUT
14 GPIOA->PUPDR &= 0xFFFFFFFE; //Pull-Down Push-button
15 GPIOA->PUPDR |= 0x2;
16
17 GPIOG->ODR = 0x2000; // LED1 ON
18 GPIOG->ODR |= 0x1<<LED2; // LED2 ON
19
20 while(1)
21 {
22 //GPIOG->ODR = 0x2000;
23 btn_val = GPIOA->IDR;
24 if ((btn_val & 0x1) == 0x1)
25 {
26 GPIOG->ODR = 0x1<<LED1;
27 }else{
RCC
Property
Value
AHB1RSTR
0
AHB2RSTR
0
AHB3RSTR
0
APB1RSTR
0
APB2RSTR
0
AHB1ENR
0x000003FF
AHB2ENR
0
AHB3ENR
0
APB1ENR
0
APB2ENR
0
AHB1LPENR
0x7EEF97FF
AHB2LPENR
0x000000F1
AHB3LPENR
0x00000001
APB1LPENR
0xF6FEC9FF
APB2LPENR
0x04777F33
BDCR
0
CSR
0x1E000000
SSCGR
0
PLLI2SCFGR
0x24003000
PLLSAICFGR
0x24003000
DCKCFGR
0
AHB1ENR
[Bits 31..0] RW (@ 0x40023830) AHB1 peripheral clock register
Project
Registers
Command
SUPPORT EXPIRED, NEW PRODUCT SERIAL NUMBER REQUIRED
Running with Code Size Limit: 32K
Load "D:\\term8\\Micro Lab\\two\\Objects\\two.axf"
ASSIGN BreakDisable BreakEnable BreakKill BreakList BreakSet BreakAccess COVERAGE
Call Stack + Locals
Name
Location/Value
Type
main
0x080002FC
int f()
btn_val
0x00000000
auto - uint
LED1
<not in scope>
auto - uint
Call Stack + Locals
Memory 1
ST-Link Debugger
t1: 348.13531075 sec
L:3 C:1
CAP NUM SCRL OVR R/W
ENG
2:10 PM
Thursday
5/12/2022



آزمایش ۲:

توضیحات مقادیر تقسیم کننده‌ها و ضرب‌کننده‌های فرکانس، داخل کامنت‌های کد آمده است:

```
#include "stm32f4xx.h"
int main(void)
}
        unsigned int LED1 = 0xD;              // PG13
        unsigned int LED2 = 0xE;              // PG14
        unsigned char flag = 0;
        unsigned char status = 0;     //0:no pll 16 MHz      :\PLL 40Mhz
        unsigned long int i = 0;
        volatile uint32_t btn_val;
        RCC->AHB1ENR = 0x3FF;
        GPIOG->MODER = 0xD7FFFFFF;
        GPIOA->MODER &= 0xFFFFFFFC;                  //push-button USER INPUT
        GPIOA->PUPDR &= 0xFFFFFFFE;                  //Pull-Down Push-button
        GPIOA->PUPDR |= 0x2;
```

```
GPIOG->ODR = 0x2000;  // LED1 ON
GPIOG->ODR |= 0x1<<LED2;     // LED2 ON
RCC->PLLCFGR &= 0xF0BC8000;
RCC->PLLCFGR |= 0x04012810;
while(\)
}
        btn_val = GPIOA->IDR;
        if ((btn_val & 0x1) == 0x1 )
        }
                if (flag == 0)}
                        if (status == 0)}
                                status = 1;
                                RCC->CR |= 0x1000000 ;
                                RCC->CFGR &= 0xFFFFFFC;
                                RCC->CFGR |= 0X0000002;      //PLL
                        {else}
                                status = 0;
                                RCC->CR &= 0xFEFFFFFF ;
                                RCC->CFGR &= 0xFFFFFFC;
                                RCC->CFGR |= 0X0;     //HSI
                        {
                {
                flag = 1;
        {else}
                flag = 0;
        {
        GPIOG->ODR = 0x1<<LED2;
        for ( i=0; i<2000000;i++);
```

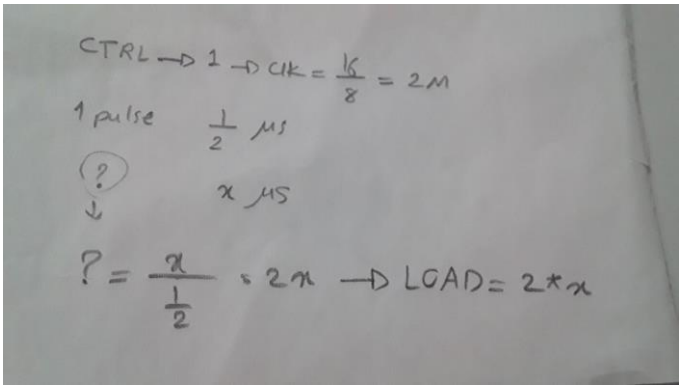
```
            GPIOG->ODR = 0x0;

            for ( i=0; i<2000000;i++);

        {

{
```

آزمایش ۳:



delay.h

```
#include "stm32f4xx.h"

void delay_ms (int x);
void delay_us (int x);
void delay_s (int x);

```

delay.h | delay.c | main.c*

```
#include "stm32f4xx.h"
#include "delay.h"

int main(void)
{
  int p;
  RCC->AHB1ENR = 0x3FF;
  GPIOA->MODER &=0XFFFFFFFC;
  GPIOG -> MODER = 0xD7FFFFFF; // PG13, PG14 -> OUTPUT
  RCC->PLLCFGR &= 0xF0BDA810;
  RCC->PLLCFGR |= 0x12810;

  RCC->CR |=0X00000003;
  RCC->CFGR = 0;
  while(1)
  {

    if (((GPIOA->IDR) & 1) == 1 ){
      p = 1;
    }
    if (p == 1){
//      GPIOG ->ODR = 0x2000; // PG13 IS ON
      RCC->CR |= 0x01000000;
      RCC->CFGR = 0XA;
}

    delay_s(1);
    GPIOG->ODR = 0X0000;
    delay_s(1);
    GPIOG->ODR = 0X4000;

  }
}

```

delay.h | delay.c

```
#include "delay.h"


void delay_us (int x){
  SysTick->CTRL |= 1;
  SysTick->LOAD = 2*x;
  while(SysTick->VAL);

}
void delay_ms (int x){
  SysTick->CTRL |= 1;
  SysTick->LOAD = 2000*x;
  while(SysTick->VAL);
}
void delay_s (int x){
  SysTick->CTRL |= 1;
  SysTick->LOAD = 2000000*x;
  while(SysTick->VAL);
}

```

مراجع

[1] https://www.linearmotiontips.com/microstepping-basics/#:~:text=Microstepping%20is%20achieved%20by%20using,decreases%20in%20the%20other%20winding.

LD293 Datasheet

Stm32f429zit datasheet

Stm32f429zit Refrence Manual

Stm32f429zit Discovery board

http://pnjunctionlab.com/l293-d/