دانشگاه صنعتی امیرکبیر

دانشکده مهندسی پزشکی

گزارش آزمایشگاه میکروپروسسور

آزمایش ۲: بررسی پورت‌های ورودی – خروجی با کاربرد عمومی (GPIO)

گروه ۲:

حمیدرضا ابوئی مهریزی

Click here to enter text.

تاریخ اتمام آزمایش: ۱۴۰۱/۰۲/۱

آن چه از این آزمایش فرا گرفته‌اید:

نحوه استفاده از پورت‌ها، رجسیتر های آنها، ابزارهای آنها

مقدمه

در سری میکروکنترلرهای ARM، برخلاف AVR ، برای پورت‌ها تعداد زیادی رجیستر تعریف شده و قابلیت‌های بیشتری نسبت به آن دارد.

پرسش‌ها و آزمایش‌ها

بخش اول

؟ سوأل ۱:

این میکروکنترلر، دارای ۱۴۴ پایه است که ۱۱۴ تای آن‌ها پایه‌های ورودی خروجی همه‌منظوره هستند.

؟ سوأل ۲:

با توجه به این که در میکروکنترلرهای ARM، توجه ویژه‌ای به مصرف انرژی شده است، در اینجا هم برای کاهش مصرف انرژی برای پورت‌هایی که استفاده نمی‌شوند، یا نیاز به فرکانس کاری بالایی ندارند امکان این وجود دارد که مصرف را بهینه کند.

؟ سوأل ۳:

```
#include "stm32f4xx.h"
int main(void)
{
	unsigned char LED = 0x1;
	int i;
	RCC->AHB1ENR = 0x3FF;
	GPIOB->MODER = 0x5555<<8;
	GPIOB->ODR = 0xFF<<4;
	while(1)
	{
		GPIOB->BSRR = LED<<20;
		LED = LED<<1;
		if (LED==0x40)
		{
			LED = 0x1;
			GPIOB->ODR = 0xFF<<4;
		}
		for (i=0; i<0x50000; i++);
	}
}
```

در این کد ابتدا کلاک سیستم تعیین شده سپس در بیت‌های B4 تا B11، LED های سون سگمنت متصل شده اند و با یک وقفه کوتاه تک به تک روشن می‌شوند و دور میزند.

سوأل ۴:

برای قفل کردن یک موتور پله‌ای کافی است فقط یکی از کویل‌ها را روشن کنیم وخاموش نکنیم. بدین صورت جاذبه مغناطیسی بین روتور و استاتور از چرخش آن جلوگیری می‌کند و به اصطلاح قفل می‌کند.

6.5 LEDs

- LD1 COM:
  The LD1 default status is red. LD1 turns to green to indicate that communications are in progress between the PC and the ST-LINK/V2-B.
- LD2 PWR:
  The red LED indicates that the board is powered.
- User LD3:
  The green LED is a user LED connected to the I/O PG13 of the STM32F429ZIT6.
- User LD4:
  The red LED is a user LED connected to the I/O PG14 of the STM32F429ZIT6.
- User LD5:
  The green LED indicates when VBUS is present on CN6 and is connected to PB13 of the STM32F429ZIT6.
- User LD6:
  The red LED indicates an overcurrent from VBUS of CN6 and is connected to the I/O PC5 of the STM32F429ZIT6.

6.6 Push-buttons

- B1 USER:
  User and Wake-Up button connected to the I/O PA0 of the STM32F429ZIT6.
- B2 RESET:
  The push-button connected to NRST is used to RESET the STM32F429ZIT6.

تحقیق ۱:

MODERy[1:0]: Port x configuration bits (y = 0..15)
These bits are written by software to configure the I/O direction mode.
00: Input (reset state)
01: General purpose output mode
10: Alternate function mode
11: Analog mode

GPIO port output type register (GPIOx_OTYPER)
(x = A..I/J/K)

Bits 15:0 **OTy**: Port x configuration bits (y = 0..15)
These bits are written by software to configure the output type of the I/O port.
0: Output push-pull (reset state)
1: Output open-drain

GPIO port output speed register (GPIOx_OSPEEDR)
(x = A..I/J/K)

OSPEEDRy[1:0]: Port x configuration bits (y = 0..15)

These bits are written by software to configure the I/O output speed.

00: Low speed
01: Medium speed
10: High speed
11: Very high speed

Note: Refer to the product datasheets for the values of OSPEEDRy bits versus V_{DD} range and external load.

GPIO port pull-up/pull-down register (GPIOx_PUPDR) (x = A..I/J/K)

PUPDRy[1:0]: Port x configuration bits (y = 0..15)

These bits are written by software to configure the I/O pull-up or pull-down

00: No pull-up, pull-down
01: Pull-up
10: Pull-down
11: Reserved

IDRy: Port input data (y = 0..15)

These bits are read-only and can be accessed in word mode only. They contain the input value of the corresponding I/O port.

Bits 15:0 **ODRy**: Port output data (y = 0..15)

These bits can be read and written by software.

Note: For atomic bit set/reset, the ODR bits can be individually set and reset by writing to the GPIOx_BSRR register (x = A..I/J/K).

GPIO port bit set/reset register (GPIOx_BSRR) (x = A..I/J/K)

Bits 31:16 **BRy:** Port x reset bit y (y = 0..15)

These bits are write-only and can be accessed in word, half-word or byte mode. A read to these bits returns the value 0x0000.

0: No action on the corresponding ODRx bit
1: Resets the corresponding ODRx bit

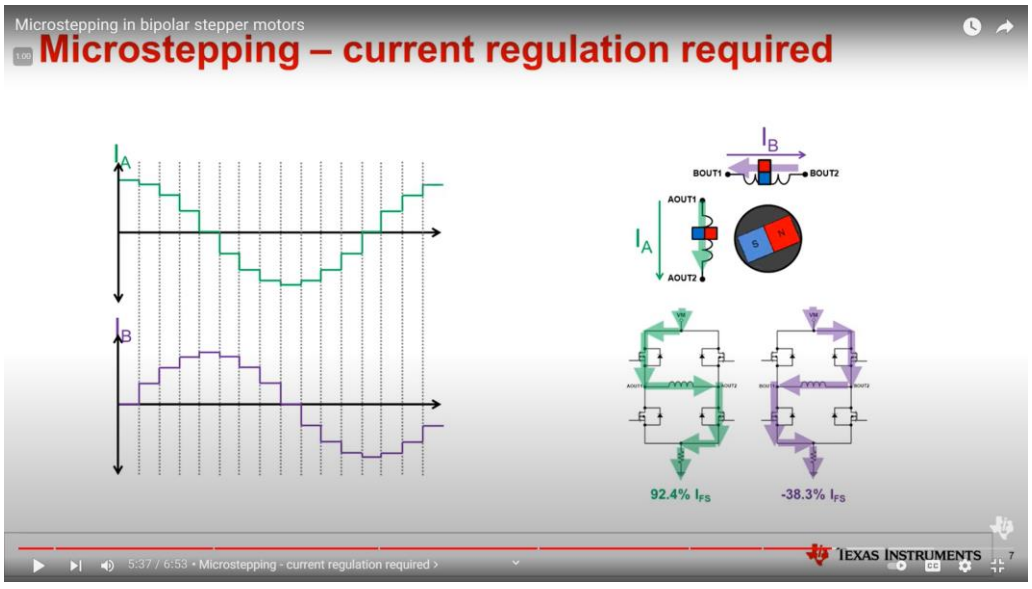
Note: If both BSx and BRx are set, BSx has priority.

Bits 15:0 **BSy:** Port x set bit y (y= 0..15)

These bits are write-only and can be accessed in word, half-word or byte mode. A read to these bits returns the value 0x0000.

0: No action on the corresponding ODRx bit
1: Sets the corresponding ODRx bit

تحقيق ٢:



Stepper motors move in discrete steps, or fractions of a revolution. For example, a stepper motor with a 1.8 degree step angle will make 200 steps for every full revolution of the motor (360 ÷ 1.8). This discrete motion means the motor's rotation isn't perfectly smooth, and the slower the rotation, the less smooth it is due to the relatively large step size. One way to alleviate this lack of smoothness at slow speeds is to reduce the size of the motor's steps. This is where microstepping comes in.

Microstepping control divides each full step into smaller steps to help smooth out the motor's rotation, especially at slow speeds. For example, a 1.8 degree step can be divided up to 256 times, providing a step angle of 0.007 degrees (1.8 ÷ 256), or 51,200 microsteps per revolution.

Microstepping is achieved by using pulse-width modulated (PWM) voltage to control current to the motor windings. The driver sends two voltage sine waves, 90 degrees out of phase, to the motor windings. While current increases in one winding, it decreases in the other winding. This gradual transfer of current results in smoother motion and more consistent torque production than full- or half-step control [1].

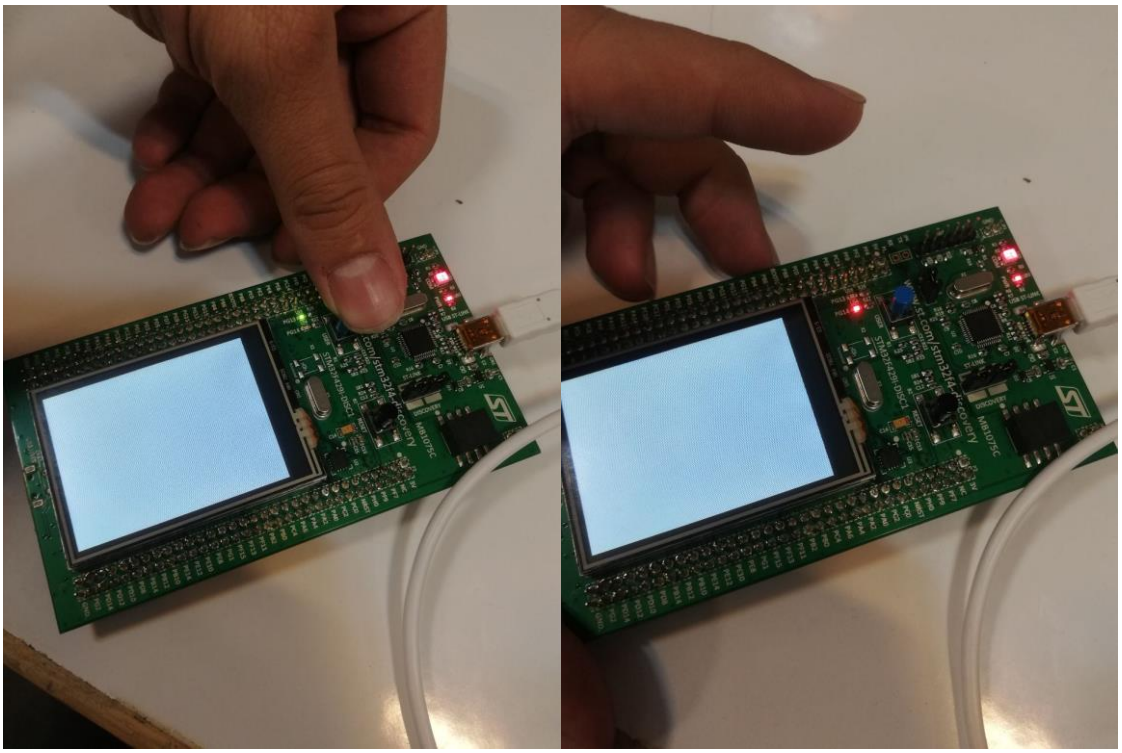
آزمایش ۱:

```
#include "stm32f4xx.h"                  // Device header
int i = 0;
int main(void)
{
	RCC->AHB1ENR = 0x3FF;


	GPIOG->MODER = (1<<26) | (1<<28);


	while(1)
	{
		GPIOB->ODR = (1<<13);
		for(i=0; i<99999; i++);
		GPIOB->ODR = (1<<14);
		for(i=0; i<99999; i++);
	}
```

}



آزمایش ۲:

15: in4:B

14: in3:B'

13: in2:A'

12: in1:A

1001 0000 0000 0000 :0x00009000

1010 0000 0000 0000 :0x0000A000

0110 0000 0000 0000 :0x00006000

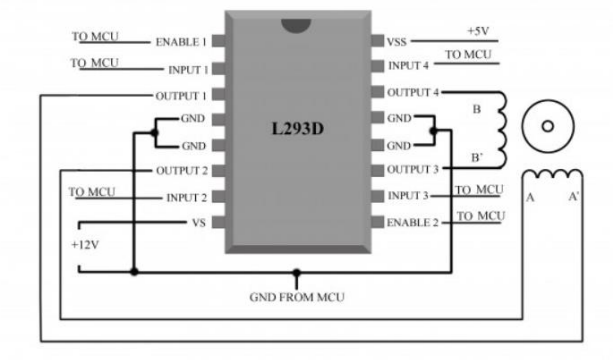
0101 0000 0000 0000 :0x00005000

```
#include "stm32f4xx.h"
int i;
int main(void)
{
	unsigned int step[4] = {0x00009000,0x0000A000,0x000006000,0x00005000};
	RCC->AHB1ENR = 0x3FF;
```

```
// MOTOR B12-B14
GPIOB->MODER |= 0x55000000;
int order = 0;
while(1)
{
        GPIOB->ODR =step[order];
        order++;
        if (order==4){
                order = 0;
        }
        for (i = 0;i<5000;i++);   //delay
}
}
```

درایور استفاده شده در این آزمایش L293 می‌باشد. 12EN و34EN به VCC متصل می‌شود و ورودی‌ها به پایه‌ی B12 تا B15 متصل می‌شوند.

1. Bi polar stepper motor



Logic Table

Stepper Table	A	B	A'	B'
STEP 1	1	1	0	0
STEP 2	0	1	1	0
STEP 3	0	0	1	1
STEP 4	1	0	0	1

آزمایش به سختی با موفقیت نسبی به پایان رسید.

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/