دانشگاه صنعتی اصفهان
دانشکده برق و کامپیوتر

دستورکار آزمایشگاه طراحی مدارهای واسط

آزمایش هشتم

تهیه کننده :مهران صفایانی

پاییز۹۵

آشنایی با شمارنده و زمانسنج

❖اهداف:

- **آشنایی با واحد شمارنده**
- **معرفی ثبات های شمارنده و زمانسنج و کاربا آنها**

❖مقدمه:

در این آزمایش با واحد شمارنده و زمانسنج میکروکنترلر LPC1768 و قابلیتهای آن آشنا می شویم . واحد زمانسنج یک بخش مهم در برنامه ریزی ارتباطات با مدارهای جانبی می باشد.

❖معرفی واحد زمانسنج میکروکنترلر LPC1768 :

میکروکنترلر LPC1768 دارای ۴ تایمر ۳۲ بیتی می باشند که به صورت تایمر و یا شمارنده مورد استفاده قرار می گیرند. به طور کلی تایمرها به شمارش کلاک می پردازند و براساس این شمارش به زمانی خاصی می توان رسید وسپس در زمان سرریز شدن تایمر کار مد نظر ما را انجام دهند. در حالت کانتر نیز به شمارش پالس های بیرونی می پردازند.

❖معرفی رجیسترهای تایمر و نحوه پیکر بندی تایمر:

رجیستر TC : ریجستر تایمر/کانتر .این ریجستر که وظیفه ذخیره شمارنده تایمر را بر عهده دارد یک ریجستر ۳۲ بیتی است . که قادر به شمارش از صفر تا 0xFFFFFFFF می باشد . این شمارنده هر PR+1 سیکل ، یک واحد اضافه خواهد شد .

رجیستر PR : ریجستر prescaler . هنگامی که مقدار ریجستر PC(Prescalre Counter) با این ریجستر برابر شود در کلاک بعدی یک واحد به این ریجستر TC اضافه خواهد شد .

رجیستر PC : این ریجستر با آمدن هر کلاک یک واحد اضافه می شود و تا زمانی که این مقدار با مقدار موجود در PR برابر نشود ، افزایش ادامه دارد . پس از برابری PC و PR یک واحد به TC اضافه شده و ریجستر PC صفر می شود .

رجیستر TCR : رجیستر کنترلی تایمر است .از این رجیستر جهت فعال و غیر فعال کردن تایمر و کانتر استفاده می شود. در ابتدای کار با تایمر باید ابتدا توسط این رجیستر باید تایمر ریست شود . بیت صفرم این رجیستر جهت فعال کردن تایمر و بیت اول جهت ریست کردن به کار می رود که با نوشتن دستورات زیر می توان این کار انجام داد. به جای x شماره تایمر را قرار می دهیم .

```
LPC_TIMx->TCR = 2;   // reset counter
LPC_TIMx->TCR = 1;   //enable counter
```

رجیستر MCR : زمانی که مقدار تایمر به رجیستر مقایسه می رسد با استفاده از این رجیستر می توان مشخص نمود که چه اتفاقی رخ دهد. در جدول زیر بیت های این رجیستر مشخص شده است . به عنوان مثال در صورتی که از رجیستر مقایسه MR0 استفاده کنیم با سه بیت اول این رجیستر می توان سه حالت راه اندازی وقفه ، ریست کردن شمارنده و متوقف کردن شمارنده را انجام داد . و با دستور روبرو آن را مقدار دهی می کنیم. در صورتی که به خواهیم از وقفه آن استفاده کنیم باید تنظیمات مربوط به وقفه ها نیز انجام شود.

Table 429. Match Control Register (T[0/1/2/3]MCR - addresses 0x4000 4014, 0x4000 8014, 0x4009 0014, 0x4009 4014) bit description

Bit	Symbol	Value	Description	Reset Value
0	MR0I	1	Interrupt on MR0: an interrupt is generated when MR0 matches the value in the TC.	0
		0	This interrupt is disabled	
1	MR0R	1	Reset on MR0: the TC will be reset if MR0 matches it.	0
		0	Feature disabled.	
2	MR0S	1	Stop on MR0: the TC and PC will be stopped and TCR[0] will be set to 0 if MR0 matches the TC.	0
		0	Feature disabled.	
3	MR1I	1	Interrupt on MR1: an interrupt is generated when MR1 matches the value in the TC.	0
		0	This interrupt is disabled	
4	MR1R	1	Reset on MR1: the TC will be reset if MR1 matches it.	0
		0	Feature disabled.	
5	MR1S	1	Stop on MR1: the TC and PC will be stopped and TCR[0] will be set to 0 if MR1 matches the TC.	0

فرض کنید بخواهیم در صورتی که مقدار تایمر با رجیستر MR0 برابر شد ، تایمر ریست و وقفه فعال شود با دستورات زیر این کار را انجام می دهیم.

```
LPC_TIM0->MCR = 3;
NVIC_SetPriority(TIMER0_IRQn,0);
NVIC_EnableIRQ(TIMER0_IRQn);
```

رجیستر تطابق MRx : شامل چهار رجیستر به شماره صفر تا ۳ است و مقدار تایمرها با این رجیستر مقایسه شده و براساس رجیستر MCR که گفته شد ، سه حالت مذکور رخ می دهد.

کلاک مربوط به بلوک TIMER توسط ریجستر های PCLKSEL0 و PCLKSEL1 به صورت پیش فرض برابر با $\frac{CPU_{clk}}{4}$ تنظیم شده است . به همین صورت کلاک متصل به ریجستر TC از رابطه زیر به دست می آید :

$$\frac{\frac{CPU_{clk}}{4}}{prescaler}$$

همچنین داریم :

$$MRx = \frac{t * PCLK\_TIMx}{prescaler}$$

برنامه نمونه: برنامه ای بنوسید که با تایمر صفر LED متصل به پورت 2.1 را دو ثانیه روشن و یک ثانیه خاموش شود.

```
#include <lpc17xx.h>
int i = 0;
int dummy = 0;
void delay(){
      dummy++;
}
void TIMER0_IRQHandler(void){
      if(i > 0){
            LPC_GPIO2 -> FIOPIN = 0xf;
            i++;
            if(i == 3)
                  i = 1;
      }
      else{
            i++;
            LPC_GPIO2 -> FIOPIN = 0x0;
      }
      LPC_TIM0 -> IR = (1 << 0);       //Clear interrupt flag
      delay();
}
int main(){
      LPC_TIM0 -> CTCR = 0;//Set timer mode and TC incremented when
the PC maches the PR
      LPC_TIM0 -> PR = 0 ;             //Set prescale 0
      LPC_TIM0 -> MR0 = 50000000;      //set MR0 for 1s
      LPC_TIM0 -> TCR = 2;             //Reset Timer 1
      LPC_TIM0 -> MCR = 3;             //The TC will be reset if MR0
      maches it and the TC and PC will be stopped
```

```
        NVIC_SetPriority(TIMER0_IRQn,0);
        NVIC_EnableIRQ(TIMER0_IRQn);
        LPC_GPIO2 -> FIODIR = 0xf;
        LPC_GPIO3 -> FIODIR = 1<<26;
        LPC_TIM0 -> TCR = 1;              //Enable Timer 1
        while(1){
        }
}
```

نکته مهم در این برنامه پاک کردن پرچم وقفه ها می باشد.

بررسی رجیستر EMR : چهار بیت اول آن در صورت تطابق می توانند toggle ، از سطح صفر به یک و بر عکس روند. بیتهای EMCx مشخص می کند که زمانی که تطابق رخ داد چه حالتی برای پایه MATx رخ دهد. نکته مهم این است که جهت استفاده از این پایه ها است که ابتدا باید آنها را با رجیستر PINSELL درمد MAT قرار دهیم.به عنوان مثال فرض کنید بخواهیم در صورت تطابق پایه MAT0.0 که مربوط به پورت ۱ و بیت شماره ۲۸ آن است Toggle شود.

```
LPC_TIM0->EMR = 0x30;
LPC_PINCON->PINSEL3 = 3<< 24;
```

❖کانتر

همانطور که می دانید تفاوت کانتر با تایمر ادر این است که کانتر کلاک های خارجی را می شمارد در حالی که تایمر کلاک داخلی خود را می شمارد. رجیسترهای کانتر همه مثل تایمر هستند و تنها با یک رجیستر کنترلی تایمر را در مد کانتر قرار می دهیم.

بررسی رجیستر CTCR : این رجیستر جهت کنترل کانتر می باشد در صورتی که به دوبیت اول آن مقدار 00 را دهیم در مد تایمر و در صورتی که مطابق جدول زیر مقادیر دیگری دهیم می توان در مد کانتر استفاده نمود. به عنوان مثال با قرار دادن مقدار 01 به این رجیستر در لبه های بالا رونده را می شمارد.

Table 428. Count Control Register (T[0/1/2/3]CTCR - addresses 0x4000 4070, 0x4000 8070, 0x4009 0070, 0x4009 4070) bit description

Bit	Symbol	Value	Description	Reset Value
3:2	Count Input Select		When bits 1:0 in this register are not 00, these bits select which CAP pin is sampled for clocking.	00
		00	CAPn.0 for TIMERn	
		01	CAPn.1 for TIMERn	
		10	Reserved	
		11	Reserved	
			Note: If Counter mode is selected for a particular CAPn input in the TnCTCR, the 3 bits for that input in the Capture Control Register (TnCCR) must be programmed as 000. However, capture and/or interrupt can be selected for the other 3 CAPn inputs in the same timer.	
31:4	-	-	Reserved, user software should not write ones to reserved bits. The value read from a reserved bit is not defined.	NA

برنامه نمونه: برنامه ای بنوسید که با فشردن کلید متصل به CAP1.0 را به کانتر یک واحد اضافه شود و نتیجه را بر روی LCD کارکتری نمایش دهد.

```
#include <lpc17xx.h>
#define LCD_LPC1768
#define LCD_PORT_2
#define LCD_RS      0
#define LCD_RW      1
#define LCD_E       2
#define LCD_DB4     4
#define LCD_DB5     5
#define LCD_DB6     6
#define LCD_DB7     7
#include "lcd.h"
void delay(){
      int i =0;
      for(i=0;i<50000 ; i++);
}
int main(){
      LPC_TIM1 -> CTCR = 1;                  //Set Counter mode and
      rising edges on the CAP
      LPC_TIM1 -> PR = 0 ;                   //Set prescale 0
      LPC_TIM1 -> MR0 = 0XFFF;
      LPC_TIM1 -> TCR = 2;                   //Reset Counter 1
      LPC_TIM1 -> MCR = 3;                   //The TC will be reset
      if MR0 mach and the TC and PC will be stopped
      LPC_PINCON -> PINSEL3 = 0x00000030;    //Set for CAP1.0
      LPC_GPIO2 -> FIODIR = 0xff;
      LPC_TIM1 -> TCR = 1;                   //Enable Counter 1
```

```
        lcd_gotoxy(1,0);
        lcd_clear();
        while(1){
              delay();
              lcd_puts(LPC_TIM1 -> TC);
        }
}
```

❖دستورکار

I. برنامه ای بنویسید که با تنظیم تایمرها هر دقیقه اطلاعات سنسور دما را خوانده و بروی نمایشگر نشان دهد.