مقدمه:

مطالبی که در حال مطالعه ی آن هستید بخش پنجم از کتاب" مرجع کامل میکروکنترلرهای سری AT91SAM شرکت ATMEL" است که در آن به بررسی سایر مباحث تکمیلی مربوط به میکروکنترلر های سری AT91SAM شرکت ATMEL ، مانند راه اندازی مبدل آنالوگ به دیجیتال ، کار با واحد دیباگ و... پرداخته شده است .

ویرایش قبلی این مطالب این بخش قبلا در مجله ی PMM (مجله میکروکنترلر فارسی – Persian Microcontroller Magazine) شماره 2 و 3 و4 و5 منتشر بود .

قبل از مطالعه مطالب این بخش شما باید بخش های 1 و 2 و 3 و 4 این کتاب که دارای مطالب زیر هستند را مطالعه کرده باشید:

1- مباحث مقدماتی : در این بخش شما با میکروکنترلر های مبتی بر هسته ی ARM آشنا شده و نحوه ی استفاده از آنها در کامپایلر KEIL را فرا گرفته و بعد از آشنایی با برخی از دستورات زبان C ، می آموزید که چطور از ورودی/خروجیهای این میکروکنترلرها استفاده کرده و چطور کتابخانه های مورد نیاز خود را ایجاد کنید . (نویسنده :1nafar)
2- راه اندازی منابع تامین کلاک : در این بخش شما با نحوه ی پیکربندی منابع کلاک میکروکنترلر آشنا می شوید .(نویسنده : آرمین غنی)
3- راه اندازی واحد تایمر/ کانتر و راه اندازی LCD گرافیکی رنگی : در فصل اول این بخش با نحوه ی راه اندازی راه اندازی واحد تایمر/ کانتر و در فصل بعدی با اصول راه اندازی LCD گرافیکی رنگی آشنا خواهید شد .(نویسنده : آرمین غنی)
4- پروتکل های ارتباطی : در این بخش با نحوه ی راه اندازی و استفاده از پروتکل های پروتکل USART ، پروتکل SPI ، پروتکل I2C و پروتکل I2S آشنا خواهید شد . (نویسنده :1nafar)

برای کسب اطلاعات بیشتر در مورد این کتاب پست شماره 11 تاپیک زیر را مطالعه کنید :

http://www.iranmicro.ir/forum/showthread.php?t=12189

توجه داشته باشید که مطالب این بخش ممکن است ناقص/اشتباه یا تاریخ گذشته باشند

مبدل آنالوگ به دیجیتال

مبدل آنالوگ به دیجیتال موجود در میکرو کنترلر های سری AT91SAM دارای ویژگی های زیر می باشند :

- 8-channel ADC
- 10-bit 384 K samples/sec. Successive Approximation Register ADC
- ±2 LSB Integral Non Linearity, ±1 LSB Differential Non Linearity
- Integrated 8-to-1 multiplexer, offering eight independent 3.3V analog inputs
- External voltage reference for better accuracy on low voltage inputs
- Individual enable and disable of each channel
- Multiple trigger sources
  - Hardware or software trigger
  - External trigger pin
  - Timer Counter 0 to 2 outputs TIOA0 to TIOA2 trigger
- Sleep Mode and conversion sequencer
  - Automatic wakeup on trigger and back to sleep mode after conversions of all enabled channels
- Four of eight analog inputs shared with digital signals

رجیستر های مربوط به ADC

در زبان C برای ADC نیز مانند سایر بخش های تعدادی رجیستر در نظر گرفته شده است که در زیر به بررسی تمامی آنها پرداخته ایم .

نام رجیستر	نام کامل	وضعیت	توضیح
ADC_CR	Control Register	Write-only	با مقدار دهی این رجیستر واحد ADC فعال میشود .
ADC_CHER	Channel Enable Register	Write-only	با مقدار دهی این رجیستر کانال دلخواه از ADC فعال میشود
ADC_CHDR	Channel Disable Register	Write-only	با مقدار دهی این رجیستر کانال دلخواه از ADC غیر فعال میشود
ADC_CHSR	Channel Status Register	Read-only	وضعیت کانال های ADC (فعال یا غیر فعال)در این رجیستر ذخیره میشود
ADC_MR	Mode Register	Read-write	این ریستر مد adc را مشخص می کند (single یا free) .
ADC_IER	Interrupt Enable Register	Write-only	با مقدار دهی این رجیستر وقفه ی ADC کانال دلخواه فعال میشود
ADC_IDR	Interrupt Disable Register	Write-only	با مقدار دهی این رجیستر وقفه ی ADC کانال دلخواه غیر فعال میشود
ADC_IMR	Interrupt Mask Register	Read-only	نوع پوشش وقفه ADC در این رجیستر ذخیره میشود .
ADC_LCDR	Last Converted Data Register	Read-only	مقدار آندازه گیری شده توسط adc در این ریسجتر ریخته می شود

ADC_CDRx	Channel Data Register X	Read-only	این ریجستر پرچم تبدیل داده adc کانال x می باشد .

*

برای کار با واحد ADC ، قبل از هر چیز باید ان را فعال کنید ، این کار با مقدار دهی رجیستر ADC_CR انجام میشود :

ADC Control Register
Register Name: ADC_CR
Access Type: Write-only

31	30	29	28	27	26	25	24
–	–	–	–	–	–	–	–
23	22	21	20	19	18	17	16
–	–	–	–	–	–	–	–
15	14	13	12	11	10	9	8
–	–	–	–	–	–	–	–
7	6	5	4	3	2	1	0
–	–	–	–	–	–	START	SWRST

همان طور که در تصویر بالا مشاهده می کنید ، برای این رجیستر فقط دو بیت start و swrst معتبر است ، این بیت ها میتوانند با صفر یا یک بودن خود حالت های زیر را اعمال نمایند :

• SWRST: Software Reset

0 = No effect.

1 = Resets the ADC simulating a hardware reset.

• START: Start Conversion

0 = No effect.

1 = Begins analog-to-digital conversion.

مثلا :

```
*AT91C_ADC_CR = 0x1;    // or  *AT91C_ADC_CR = AT91C_ADC_SWRST; // or *AT91C_ADC_CR =((unsigned int) 0x1 << 0);
*AT91C_ADC_CR = 0Xa;    // or *AT91C_ADC_CR =  AT91C_ADC_START;// or *AT91C_ADC_CR =((unsigned int) 0x1 << 1);
```

با دستورات بالا واحد adc ابتدا ریست شده و سپس راه اندازی میشود . در صورتی که فایل هدر مربوط به میکرو کنترلر دلخواه خود را برای AT91C_ADC_SWRST جستوجو کنید ، با دو خط زیر روبرو میشوید :

```
#define AT91C_ADC_SWRST      ((unsigned int) 0x1 <<  0) // (ADC) Software Reset
#define AT91C_ADC_START      ((unsigned int) 0x1 <<  1) // (ADC) Start Conversion
```

مشاهده میکنید که در این دو خط از واژه های مذکور به جای مقادیر عددی استفاده شده و مقدار دهی عددی یا استفاده از جملات ، عملا فرقی در نحوه ی راه اندازی adc ندارد .

رجیستر ADC_MR :

مقدار دهی رجیستر ADC_MR یا ADC Mode Register یکی از مهم ترین بخش های راه اندازی adc است ، در این رجیستر شما میتوانید تنظیمات مد های کم مصرفی (Sleep Mode) ، دقت adc (Resolution) و تریگر (Trigger Selection) را انجام دهید :

31	30	29	28	27	26	25	24
–	–	–	–	SHTIM			
23	22	21	20	19	18	17	16
–	STARTUP						
15	14	13	12	11	10	9	8
PRESCAL							
7	6	5	4	3	2	1	0
–	–	SLEEP	LOWRES	TRGSEL			TRGEN

بیت TRGEN(Trigger Enable) : با انتخاب مقدار صفر برای این بیت ، تبدیل adc به صورت عادی تبدیل خود را آغاز میکند . در صورتی که برای این بیت مقدار 1 انتخاب شود ، adc میتواند از طریق منابعی که در ادامه آورده میشود تحریک شده و تبدیل خود را آغاز نماید .

بیت ها ی TRGSEL(Trigger Selection) : در صورتی که برای بیت TRGEN مقدار یک انتخاب شود ، شما میتوانید با مقدار دهی این سه بیت مطابق جدول زیر منبع تحریک adc را مشخص نمایید :

TRGSEL			Selected TRGSEL
0	0	0	TIOA Ouput of the Timer Counter Channel 0
0	0	1	TIOA Ouput of the Timer Counter Channel 1
0	1	0	TIOA Ouput of the Timer Counter Channel 2
0	1	1	Reserved
1	0	0	Reserved
1	0	1	Reserved
1	1	0	External trigger
1	1	1	Reserved

بیت LOWRES (Resolution) : با انتخاب مقدار یک برای این بیت adc در مد 8 بیت و با انتخاب مقدار صفر adc در مد 10 بیت راه اندازی میشود .

بیت SLEEP (Sleep Mode) : با انتخاب مقدار 1 برای این بیت adc به مد کم مصرفی (Sleep Mode) میرود . برای بازگشت به حات عادی این بیت باید صفر شود .

بیت های PRESCAL و STARTUP و SHTIM به ترتیب برای تعیین کردن فرکانس کاری adc (ADCClock) و زمان استارتاپ (Startup Time) و نرخ نمونه برداری (Sample & Hold Time) به کار میروند ، موارد ذکر شده از فرمول های زیر محاسبه میشوند :

• PRESCAL: Prescaler Rate Selection

ADCClock = MCK / ((PRESCAL+1) * 2)

• STARTUP: Start Up Time

Startup Time = (STARTUP+1) * 8 / ADCClock

• SHTIM: Sample & Hold Time

Sample & Hold Time = SHTIM/ADCClock

سایر بیت های که در جدول بالا با علامت – مشخص شده اند ، برای مقاصد بعدی و تغییرات آینده رزرو شده اند ، ما باید به این بیت ها مقدار صفر دهیم .

رجیستر ADC_CHER و ADC_CHDR :

برای این دو رجیستر فقط 8 بیت اول معتبر میباشد :

31	30	29	28	27	26	25	24
–	–	–	–	–	–	–	–
23	22	21	20	19	18	17	16
–	–	–	–	–	–	–	–
15	14	13	12	11	10	9	8
–	–	–	–	–	–	–	–
7	6	5	4	3	2	1	0
CH7	CH6	CH5	CH4	CH3	CH2	CH1	CH0

در رجیستر ADC_CHER با قرار دادن chx=1 کانال x فعال می شود

در رجیستر ADC_CHDR با قرار دادن chx=1 کانال x غیر فعال می شود .

رجستر ADC_CDRx:

حاصل تبدیل adc کانال x در 10 بیت اول این رجیستر ذخیره میشود :

31	30	29	28	27	26	25	24
–	–	–	–	–	–	–	–
23	22	21	20	19	18	17	16
–	–	–	–	–	–	–	–
15	14	13	12	11	10	9	8
–	–	–	–	–	–	DATA	
7	6	5	4	3	2	1	0
DATA							

رجیستر ADC_LCDR :

آخرین داده ی تبدیل شده ی adc در 10 بیت اول این رجیستر ذخیره میشود

31	30	29	28	27	26	25	24
–	–	–	–	–	–	–	–
23	22	21	20	19	18	17	16
–	–	–	–	–	–	–	–
15	14	13	12	11	10	9	8
–	–	–	–	–	–	DATA	
7	6	5	4	3	2	1	0
DATA							

- با استفاده از رجیستر های ADC_SR و ADC_CHSR میتوانید وضعیت کانال ها و بیت ها ی که در بالا توضیح دادیم را بخوانید ، برای کسب اطلاعات بیشتر به صفحه ی 497 دیتاشیت AT91SAM7X512/256/128 Preliminary مراجعه کنید .
- بیت های مربوط به وقفه ی adc بعد از تشریح واحد aic و در شماره های بعدس تشرح خواهد شد .
- رجیستر های بالا صرفا برای اطلاع شما از نحوه ی راه اندازی adc آورده شده است و ارزش دیگری ندارد (با استفاده از هدر adc.h میتوانید بدون نیاز به رجیستر های بالا و با دستورات ساده adc را راه اندازی نمایید .)
- هنگام کار ADC پایه ی VREF را به VCC متصل نمایید .

هدر ADC (مبدل آنالوگ به دیجیتال)

برای آسان نمودن عملیات راه اندازی adc ، اقدام به نوشتن یک هدر نموده ایم ، شما می توانید با مطالعه این فایل با نحوه راه اندازی adc بیشتر اشنا شوید . توسط این هدر شما می توانید به راحتی با adc های میکرو ها کار کنید و مقدار خوانده شده توسط آنها را در مکان های دیگر برنامه استفاده کنید

ابتدا فایل adc.h را در مسیر زیر کپی کنید (فایل در پوشه پیوست موجود می باشد) :

Program Files\Keil\ARM\INC\Atmel.

(در پوشه های SAM7Xو SAM7A3 و AT91SAM9G20 و سایر پوشه های موجود).

برای استفاده از این کتابخانه ، ابتدا باید ان را در برنامه فراخوانی کنید ، برای اینکار از دستور زیر استفاده می شود :

```
#include "adc.h"
```

با فرا خوانی این کتابخانه ، دستورات زیر به نرم افزار KEIL اضافه می شود که شرح آنها در زیر اورده شده است :

فعال سازی کانال های دلخواه از adc :

```
config_adc(x);
```

x آدرس کانال های دلخواه می باشد ، مثلا کانال های 6 و 7 adc را فعال کنید :

```
config_adc(0xc0);
```

0xc0 = c0 = 11000000 bin

شروع تبدیل :

```
start_adc();
```

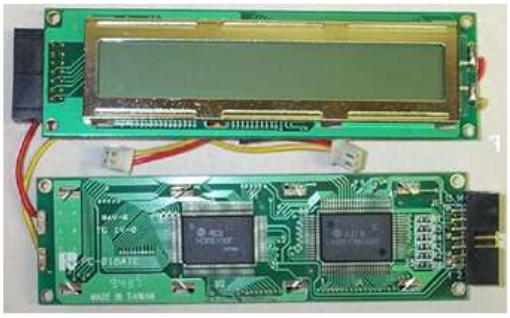
با دستور با adc شروع به کار می کند ، دستور بالا باید در حلقه اصلی قرار گیرد و قبل از خواندن adc اجرا شود .

```
x = read_adc(ch);
```

دستور بالا داده خوانده شده توسط adc را در متغیر x که از نوع int می باشد ، قرار میدهد . ch شماره کانال مورد نظر است .

راه اندازی lcd کاراکتری به صورت 4 و 8 بیت

نمایشگر های گرافیکی کار برد گسترده ای در دستگاه های الکترونیکی دارند ، از این نمایشگر ها برای نمایش دادن وضعیت های مختلف در دستگاه استفاده میشود .



راه اندازی ساده ، مصرف توان کم ، قیمت ارزان و عمر بالا از ویژگی بارز این نمایشگر ها میباشد . اغلب این نمایشگر از کنترل کننده ی hd44780 استفاده میکنند معمولا سازند ه برای کم کردن قیمت نهایی قطعه ، تراشه را بر روی برد آن پیدا سازی میکند و رویش را با لایه ای از چسب سخت میپوشاند .

دیتا شیت این lcd را میتوانید از ادرس زیر دانلود کنید (lcd و کنترل کننده ی آن) :

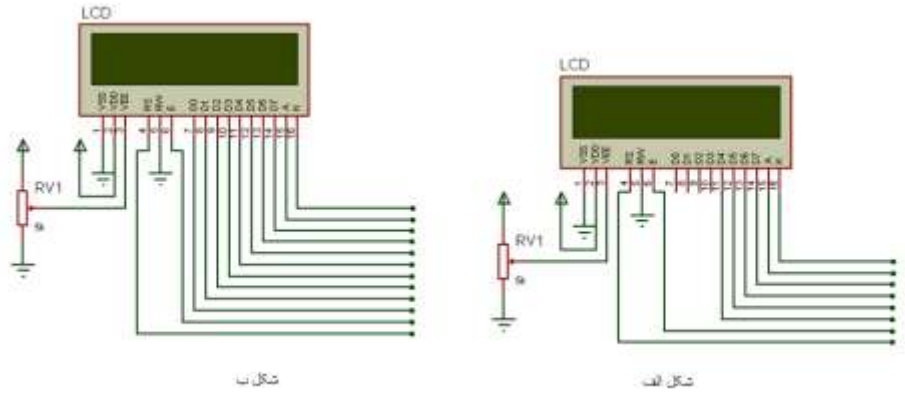
http://www.alldatasheet.com/datasheet-pdf/pdf/63673/HITACHI/HD44780.html

این lcd ها در نمونه های 16*1 و 16*2 و 20*2 و... ساخته میشوند ، راه اندازی تمامی آنها مشابه هم است و با کنترل 4 یا 8 بیتی انجام میشود ، تمامی این lcd ها دارای 16 پایه به شرح زیر میباشند :

محل اتصال	کاربرد	نام پایه	شماره
گراند مدار	گراند قطعه	vss	1
ولتاژ 5 ولت	تغذیه مثبت قطعه	vdd	2
اتصال به vdd = حداکثر کنتراست ، اتصال به گراند= کمترین کنتراست	کنترل کننده ی کنتراست	VO	3
یکی از پایه های میکرو که در برنامه مشخص می شود	انتخاب رجیستر	RS	4
یکی از پایه های میکرو که در برنامه مشخص می شود	نوشتن / خواندن	R/W	5
یکی از پایه های میکرو که در برنامه مشخص می شود	فعال ساز lcd	E	6
یکی از پایه های میکرو که در برنامه مشخص می شود	پایه دیتا	DB0	7
یکی از پایه های میکرو که در برنامه مشخص می شود	پایه دیتا	DB1	8
یکی از پایه های میکرو که در برنامه مشخص می شود	پایه دیتا	DB2	9
یکی از پایه های میکرو که در برنامه مشخص می شود	پایه دیتا	DB3	10

یکی از پایه های میکرو که در برنامه مشخص می شود	پایه دیتا	DB4	11
یکی از پایه های میکرو که در برنامه مشخص می شود	پایه دیتا	DB5	12
یکی از پایه های میکرو که در برنامه مشخص می شود	پایه دیتا	DB6	13
یکی از پایه های میکرو که در برنامه مشخص می شود	پایه دیتا	DB7	14
ولتاژ تغذیه مثبت	اند led بکلایت	LED/A	15
گراند مدار	کاتد led بکلایت	LED/K	16

برای ارتباط با این lcd میتوانید از مد های 4 و 8 بیتی استفاده نمایید ، در مد 4 بیت lcd به صورت شکل الف و در مد 48 بیتی lcd به صورت شکل ب به میکرو متصل میشود :



شکل ب شکل الف

در مد 8 بیتی علاوه بر نوشتن داده بر روی LCD ، توانایی خواندن اطلاعات آن را نیز دارید ، برای خواندن LCD پایه R/W باید در سطح منطقی یک قرار گیرد (در این حالت میتوان از LCD به عنوان یک حافظه ی موقت استفاده کرد ، خوشبختانه با وجود حافظه های متنوع در میکرو کنترلر ها از این قابلیت استفاده ی چندانی نمیشود) .

نحوه ی کار LCD:

در صورتی که پشت LCD را مشاهده کنید ، بر روی آن دو چیپ وجود دارد . اولین قطعه کنترل کننده ی hd44780 و دومین قطعه حافظه ی جانبی آن میباشد ، ممکن است این قطعات به صورت chip یا cob (chip on board) بر روی برد lcd نصب شده باشند .

شرکت سازنده ی اطلاعات مربوط به تمامی کاکتر های اسکی را در حافظه قرار میدهد ، با این کار راه اندازی LCD بسیار ساده شده و برای نمایش یک پیغام کافی است حرف یا عبارت خود را به کد اسکی تبدیل کرده و برای LCD ارسال کنید ، کنترل کننده بعد از دریافت کد ، آن را با مقادیر موجود در حافظه مقایسه کرده و اطلاعات متناظر را به نمایشگر

ارسال میکند . با روشن شدن پیکسل های متناظر با کد ارسال شده ، شما میتوانید آن را بر روی LCD مشاهده کنید . در صفحه ی 18 و 18 دیتاشیت قطعه ، لیست کلیه کاراکتر های که توسط LCD پشتیبانی میشود آورده است .

در صورتی که کاراکتر دلخواه شما در لیست فوق وجود ندارد میتوانید آن را توسط نرم افزار هایی که در ادامه معرفی میشوند ایجاد کرده و در حافظه ی موقت LCD ذخیره کنید ، LCD امکان ذخیره 7 کاراکتر را به شما میدهد .

کنترل LCD:

برای کار با LCD و کنترل آن تعدادی کد از طرف شرکت سازنده ایجاد شده است ، با ارسال این کد ها به LCD میتوانید عملیات های مختلفی نظیر پاک کردن ، انتقال مکان نما ، و.... را بر روی LCD انجام دهید ، لیست کامل کد ها + عمل کرد هر یک در جدول زیر آورده شده است :

Instruction	کد ارسالی	کاربرد
Clear display	0x01	با ارسال این کد بر روی خطوط دیتا ، صفحه ی نمایشگر پاک میشود
Return home	0x02	با ارسال این کد بر روی خطوط دیتا ، نمایش کاکتر ها از سطر وستون یک اغاز میشود .
cursor_on	0x0E	با ارسال این کد بر روی خطوط دیتا ، مکان نما روشن میشود
cursor_off	0x0C	با ارسال این کد بر روی خطوط دیتا ، مکان نما خاموش میشود
cursor_blink	0x0F	با ارسال این کد بر روی خطوط دیتا ، مکان نما چشمک میزند
Display off	0x0C	با ارسال این کد بر روی خطوط دیتا ، نمایشگر خاموش میشود
Display on	0x08	با ارسال این کد بر روی خطوط دیتا ، نمایشگر روشن میشود
Display shift left	0x1E	با ارسال این کد بر روی خطوط دیتا ، داده ی موجود بر روی نمایشگر یک خانه به چپ منتقل میشود .
Display shift right	0x18	با ارسال این کد بر روی خطوط دیتا ، داده ی موجود بر روی نمایشگر یک خانه به راست منتقل میشود .
Cursor-line one	0x00	با ارسال این کد بر روی خطوط دیتا ، مکان نما به سطر اول میرود . (نمایش از سطر اول شروع میشود)
Cursor-line two	0x40	با ارسال این کد بر روی خطوط دیتا ، مکان نما به سطر دوم میرود . (نمایش از سطر دوم شروع میشود)
Cursor-line tree	0x14	با ارسال این کد بر روی خطوط دیتا ، مکان نما به سطر سوم میرود . (نمایش از سطر سوم شروع میشود)
Cursor-line four	0x54	با ارسال این کد بر روی خطوط دیتا ، مکان نما به سطر چهارم میرود . (نمایش از سطر چهارم شروع میشود)

- برای نمایش متن در ستون دلخواه باید حافظه ی DDRAM را آدرس دهی کنید ، برای این کار کافی است شماره ی ستون دلخواه را با آدرس سطر که در بالا اشاره شد جمع کرده و برای lcd ارسال کنید .
- در هنگام ارسال کد های بالا باید پایه های RS و R/W در سطح منطقی صفر و پایه ی E در سطح منطقی یک قرار گیرد .

- در صورتی که LCD به صورت 8 بیت راه اندازی میشود ، کافی است داده های بالا را مستقیما بر روی خطوط داده قرار دهید .
- در صورتی که LCD به صورت 4 بیت راه اندازی شود ، باید کد های بالا را در دو نیم بایت برای LCD ارسال کنید .

هر بایت از 8 بیت تشکیل میشود ، مثلا کد متناظر با کاراکتر A برابر 0100 0001 میباشد که به چهار بیت اول (0100) بیت های بالایی (Upper Bits) و به چهار بیت دوم (0001) بیت هایی پایینی یا کم ارزش (Lower Bits) گفته میشود . در راه اندازی 4 بیتی ابتدا بیت های بالایی و بعد از یک تاخیر زمانی 4 بیت پایینی ارسال میشود . برای جداکردن 4 بیت اول میتوان از دستور & (اند) و برای 4 بیت دوم از دستور شیفت به راست (>>) استفاده کرد ، با مراجعه به هدر LCD و تابع lcd_command میتوانید نمونه ی عملی این موضوع را مشاهده کنید .

- مقدار متوسط زمانی که LCD برای پردازش دستور home Return نیاز دارد 1.52 میلی ثانیه میباشد، این زمان برای سایر دستورات 37 μs است .
- ارسال کاکتر به LCD کاملا مشابه ارسال دستورات میباشد ، فقط در این حالت باید پایه R/W در سطح منطقی صفر و پایه های RS و E در سطح منطقی یک قرار گیرد .

هدر LCD:

برای تسریع در راه اندازی LCD به صورت 4 و 8 بیت ، هدر lcd.h را ارائه میکنیم ، با استفاده از این هدر نیاز به راه اندازی قطعه به صورت دستی و پیکربندی آن نخواهید داشت و نمایش داده با دستوراتی که در ادامه معرفی میشوند ،میسر خواهد بود .

ابتدا فایل lcd.h را در مسیر زیر کپی کنید (فایل در پوشه پیوست موجود می باشد) :

Program Files\Keil\ARM\INC\Atmel.

(در پوشه های SAM7Xو SAM7A3 و AT91SAM9G20 و سایر پوشه های موجود).

برای استفاده از این کتابخانه ، باید بعد از معرفی پایه ها ، آن را در برنامه فراخوانی کنید ، برای اینکار از دستور زیر استفاده می شود :

```
#include <lcd.h>
```

با فرا خوانی این کتابخانه ، دستوراتی به نرم افزار KEIL اضافه می شود که شرح آنها در زیر آورده شده است :

مشخص کردن نوع ارتباط با lcd :

```
#define  LCD_bit x
```

X میتواند عدد 4 یا 8 باشد . هنگامی که از عدد 4 استفاده میکنید ، lcd به صورت 4 بیت پیکربندی میشود ، در این حالت نیازی به استفاده از پایه های db0 تا db3 نیست . با استفاده از عدد 8 قطعه به صورت 8 بیت راه اندازی شده و شما باید تمامی پایه های داده را به میکرو متصل کنید .

در صورتی که دستور بالا را در برنامه ی خود نیاورید ، lcd به صورت پیش فرض و در مد 4 بیت پیکربندی می شود.

مشخص کردن پایه های lcd :

```
#define  LCD_PORT  y
#define LCD_RS x
#define LCD_E x
#define LCD_DB0 x
#define LCD_DB1 x
#define LCD_DB2 x
#define LCD_DB3 x
#define LCD_DB4 x
#define LCD_DB5 x
#define LCD_DB6 x
#define LCD_DB7 x
```

در این دستورات y به نشانه ی پورت دلخواه از میکرو کنترلر و x پایه های دلخواه از پورت y میباشد . هنگامی که از lcd به صورت 4 بیت استفاده میشود نیازی به دستورات زیر نیست :

```
#define LCD_DB0 x
#define LCD_DB1 x
#define LCD_DB2 x
#define LCD_DB3 x
```

بعد از پیکربندی پایه ها ، با استفاده از دستورات زیر می توان با lcd کار کرد :

```
lcd_gotoxy(x,y);
```

دستور بالا مکان نما را به سطر x و ستون y می برد ، در واقع نوشتن از سطر x و ستون y شروع می شود .

```
lcd_putsf(x);
```

با دستور بالا می توان رشته x را بر روی lcd نمایش داد .

```
lcd_puts(x);
```

با دستور بالا عدد یا متغیر x را بر روی lcd نمایش داد میشود .

```
lcd_gotoxy(1,1);
lcd_putsf("abc");
lcd_gotoxy(2,1);
lcd_puts(a);
```

درمثال بالا عبارت abc در سطر و ستون اول و مقدار متغیر a در سطر دوم وستون دوم نمایش داده می شود .

```
Display_off( );
```

با دستور بالا lcd خاموش می شود .

```
Display_on();
```

دستور بالا lcd را روشن می کند .

```
cursor_off( );
```

دستور بالا مکان نمای lcd را خاموش می کند .

```
cursor_on ();
```

دستور بالا مکان نمای lcd را روشن می کند .

```
cursor_blink ( );
```

با دستور بالا مکان نما چشمک زن می شود .

```
shift_right (x) ;
```

دستور بالا موارد نوشته شده بر روی lcd را به اندازه ی x خانه به سمت راست جابجا می کند .

```
shift_ left (x) ;
```

دستور بالا موارد نوشته شده بر روی lcd را به اندازه ی x خانه به سمت چپ جابجا می کند.

```
lcd_clear();
```

دستور بالا lcd را پاک می کند .

تمامی دستورات کار با LCD به صورت 4 بیت مشابه با دستورات LCD 8 بیت می باشد و تنها تفاوت در دستورات پیکربندی است .

پروژه 1:نمایش قابلیت های کتابخانه

در این پروژه پایه های lcd مطابق زیر به میکرو کنترلر AT91SAM7x256 متصل شده اند :

پایه rs به پایه b.0 ----- پایه enable به پایه b.1 ، پایه های db4 تا db7 به ترتیب به پایه های b.2 تا b.5 .

```
#include "AT91SAM7X256.h"
#include "lcd.h"
#include "delay.h"
int main (void) {
int a=58;
#define  LCD_bit x
#define  LCD_PORT  B
#define LCD_RS 0
#define LCD_E 1
#define LCD_DB4 2
#define LCD_DB5 3
#define LCD_DB6 4
#define LCD_DB7 5
#include "lcd.h"
while(1)  {
lcd_gotoxy(2,5);
lcd_putsf("rthgdyehfn");
cursor_on();
 delay_s(5);
cursor_off();
 delay_s(5);
cursor_blink();
 delay_s(5);
 Display_off();
 delay_s(5);
  Display_on();
 delay_s(5);
 lcd_clear();
delay_s(5);
```

```
lcd_gotoxy(2,1);
lcd_putsf("xxxxxxxx");
delay_s(5);
lcd_gotoxy(1,2);
lcd_puts(a);
delay_s(5);
lcd_shift_right(5);
delay_s(5);
lcd_clear();
lcd_putsf("123654789");
delay_s(5);
lcd_shift_left(4);
delay_s(5);
 lcd_clear();
  }}
```

برای اجرا کردن پروژه بالا نیاز به انجام دادن تنظیمات خاصی نیست ، شما باید پایه های یک و پنج LCD را به گراند ، پایه 2 را به ولتاژ 5 ولت و پایه 3 را مطابق شکل که در تصاویر قبل وجود داشت تنظیم کنید .

پروژه 2 :ساعت و تقویم با استفاده از کتابخانه lcd :

```
#include "AT91SAM7X256.h"

#include "delay.h"

int main (void) {

int sec=55,min=12,hur=5,dey=23,mon=3,yer=89;

#define  LCD_PORT  B
#define LCD_RS 0
#define LCD_E 1
#define LCD_DB4 2
#define LCD_DB5 3
#define LCD_DB6 4
```

```
#define LCD_DB7 5
#include "lcd.h"

  while(1)  {
delay_ms(999);
sec++;
if (sec==60) {
min++,sec=0;}
if (min==60){
hur++,min=0;}
if (hur==24){
dey++,hur=0;}
if (dey==31){
mon++,dey=0;}
if (mon==12)
yer++;
lcd_gotoxy(1,1);
lcd_putsf("time:    :  :   ");
lcd_gotoxy(2,1);
lcd_putsf("date:    :  :   ");
lcd_gotoxy(1,8);
lcd_puts(hur);
lcd_gotoxy(1,11);
lcd_puts(min);
lcd_gotoxy(1,14);
lcd_puts(sec);
lcd_gotoxy(2,8);
lcd_puts(yer);
lcd_gotoxy(2,11);
lcd_puts(mon);
lcd_gotoxy(2,14);
lcd_puts(dey);
  }}
```

پروژه 3:نمایش دما روی LCD

در این پروژه adc های شماره 6 و 7 میکرو کنترلر at91sam7x256 راه اندازی شده اند ، ما مقادیر خوانده شده توسط آنها را بر روی lcd نمایش داده ایم :

```
#include "AT91SAM7X256.h"
#include "delay.h"
#include "adc.h"
#define  LCD_PORT_B
#define LCD_RS 8
#define LCD_E 10
#define LCD_DB4 12
#define LCD_DB5 13
#define LCD_DB6 14
#define LCD_DB7 15
#include "lcd.h"
int main (void){
unsigned int a , b;
lcd_init();
config_adc( 0xc0);
while(1){
start_adc();
lcd_gotoxy(1,1);
lcd_putsf( "adc=");
a = read_adc(6);
b = read_adc(7);
lcd_gotoxy(2,1);
lcd_puts(a);
lcd_gotoxy(2,7);
lcd_puts(b);
delay_ms(500);
}}
```

پروژه 4 :سیستم کنترل دما:

در این پروژه با استفاده از میکرو کنترلر at91sam7x256 ، lcd کارکتری و سنسور دما ی lm35 یک دستگاه کنترل تهویه ساخته شده است . برای این پروژه شرایط زیر وجود دارد :

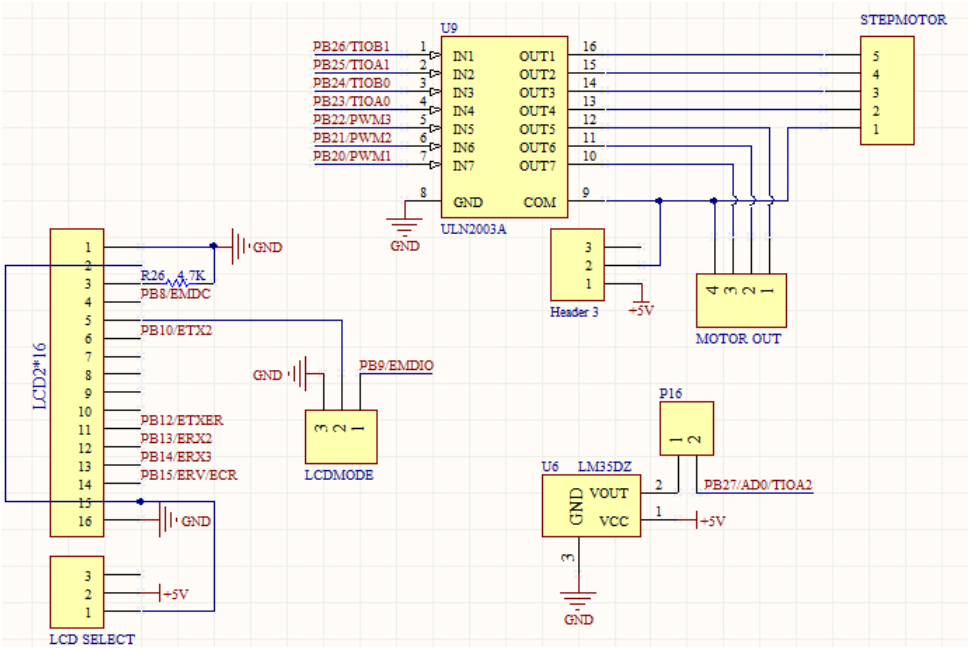
- دو موتور به پایه های 21 و 22 پورت B متصل شده است .
- هنگامی که دما از 30 درجه کمتر میشود ، باید هر دو موتور خاموش شود .
- هنگامی که دما از 50 درجه بیشتر میشود ، باید هر دو موتور روشن شود .
- هنگامی که دما بین 30 تا 50 درجه ی سانتیگراد است ، باید موتور یک خاموش و موتور 2 روشن شود .

- کلیه وضعیت ها شامل دمای محیط ، وضعیت موتور ها و... بر روی LCD کارکتری 2*16 نمایش داده میشود .

در ادامه شماتیک این پروژه آورده شده است :

- همانطور که مشاهده میکنید ما برای راه اندازی موتور ها از ایسی ULN2003 استفاده نموده ایم ، شما میتوانید در صورت نیاز از ترانزیستور یا ... برای راه اندازی موتور ها استفاده کنید .
- هنگام راه اندازی lcd به صورت 4 خط داده (4 بیت) در نرم افزار keil نیازی به اتصال پایه ی lcd 5 (R/W) نمیباشد ، شما میتوانید جامپر مربوط به این پایه که با نام LCDMODE بر روی برد آموزشی مشخص گردیده را به زمین متصل کنید . (این جامپر برای مصارف بعدی و برنامه نویسی با کامپایلر های دیگر تعبیه شده است ، ممکن است در کامپایلر های دیگر به پایه ی R/W نیاز داشته باشید) .
- به یاد داشته باشید که برای راه اندازی ADC حتما جامپر VREF (ولتاژ رفرنس) را بر روی برد ARM متصل نمایید .
- برای جلوگیری از تداخل خروجی سنسور دمای LM35 با دیگر قطعات متصل شده به ADC ، ما پایه ی Vout آن را توسط جامپر P16 از ورودی ADC (PB27/AD0/TIOA2) جدا کرده ایم ، برای راه اندازی سنسور این جامپر نیز باید متصل شود .

شماتیک پروژه :



برای نمایش داده بر روی LCD از کتابخانه ی LCD.h استفاده شده است ، در این کتابخانه نیازی به اتصال پایه های db0 تا lcd db3 نمی باشد . پایه هاس PB21 و PB22 به ترتیب به سیستم کنترل موتور یک و موتور دو متصل خواهد شد .

سورس پروژه :

```
#include <AT91SAM7X256.H>
#include "pio.h"
#include "adc.h"
#include "delay.h"
#define  LCD_PORT_B
#define LCD_RS 8
#define LCD_E 10
#define LCD_DB4 12
#define LCD_DB5 13
#define LCD_DB6 14
#define LCD_DB7 15
#include "lcd.h"
int main (void){
unsigned int a ;
lcd_init();
config_adc(0x01);
PORTA_OUTPUT= 0x07F80000;
RST_PORTB 0x07F80000;
while(1)
{
cursor_off();
start_adc();
lcd_gotoxy(1,1);
lcd_putsf("temp is:");
            a = read_adc(0);
            a=a/3;
```

```
if(a<30) {
RST_PORTB =PB22;
RST_PORTB=PB21;
lcd_gotoxy(2,1);
lcd_putsf("temp low");}
else if (a>50){
        SET_PORTB=PB21;
        SET_PORTB=PB22;
lcd_gotoxy(2,1);
lcd_putsf("temp high");}
else  {
        SET_PORTB=PB22;
RST_PORTB=PB21;
lcd_gotoxy(2,1);
lcd_putsf("temp normal"); }
lcd_gotoxy(1,10);
lcd_puts(a);
delay_ms(500);
lcd_clear();
}
}
```

توضیحات برنامه :

```
#include <AT91SAM7X256.H>
#include "pio.h"
#include "adc.h"
#include "delay.h"
#define  LCD_PORT_B
#define LCD_RS 8
#define LCD_E 10
#define LCD_DB4 12
```

```
#define LCD_DB5 13
#define LCD_DB6 14
#define LCD_DB7 15
#include "lcd.h"
```

در اولین بخش برنامه بعد از معرفی کردن میکرو و فراخوانی کتابخانه های مورد نیاز ، پایه های که lcd به آنها متصل شده ، برای کامپایلر معرفی گردیده است و در نهایت هدر lcd فراخوانی شده است.

```
int main (void){
unsigned int a ;
lcd_init();
config_adc(0x01);
PORTA_OUTPUT= 0x07F80000;
RST_PORTB 0x07F80000;
while(1)
```

در ابتدا ی حلقه ی اصلی ، lcd کارکتری و adc پیکربندی شده اند ، در خطوط بعدی کلیه پایه های متصل به 8 عدد led موجود بر روی برد اصلی به عنوان خروجی تعریف شده و سپس خاموش شده اند .

```
{
cursor_off();
start_adc();
lcd_gotoxy(1,1);
lcd_putsf("temp is:");
        a = read_adc(0);
        a=a/3;
if(a<30) {
RST_PORTB =PB22;
RST_PORTB=PB21;
lcd_gotoxy(2,1);
lcd_putsf("temp low");}
else if (a>50){
        SET_PORTB=PB21;
        SET_PORTB=PB22;
```

```
lcd_gotoxy(2,1);
lcd_putsf("temp high");}
else  {
        SET_PORTB=PB22;
RST_PORTB=PB21;
lcd_gotoxy(2,1);
lcd_putsf("temp normal"); }
lcd_gotoxy(1,10);
lcd_puts(a);
delay_ms(500);
lcd_clear();
}
}
```

در این بخش با استفاده از دستور while(1) یک حلقه ی بینهایت ایجاد گردیده است ، با دستور start_adc(); مبدل آنالوگ به دیجیتال میکرو تبدیل خود را آغاز کرده و با دستور a = read_adc(0); حاصل آن را در متغیر a میریزد .

در ادامه و با دستور lcd_puts(a); و lcd_gotoxy(1,10); مقدار این متغیر بر روی سطر دوم lcd به نمایش در می آید ، دستورات if موجود مقادیر این متغیر را با ارقام 30 و 50 مقایسه کرده و رشته ی مناسب را به lcd و توسط دستور lcd_putsf ارسال میکنند .

دستورات cursor_off(); و lcd_clear(); به ترتیب باعث خاموش شدن مکان نما و پاک شدن lcd میشوند .

تایمر Watchdog :

Watchdog یکی از تایمر های داخلی میکرو arm میباشد که میتوانید میکرو را بعد از یک زمان مشخص ریست کند .از این تایمر در هنگام بوجود آمدن خطا در سیستم و... استفاده میشود . با فعال شدن این تایمر بعد از گذشت زمان تعیین شده ، میکرو ریست میشود و برنامه از ابتدا اجرا میشود .

نحوه شبیه سازی تایمر Watchdog در نرم افزار keil :

Watchdog یکی از تایمر های داخلی میکرو ARM میباشد که می توانید میکرو را بعد از یک زمان مشخص ریست کند .از این تایمر در هنگام بوجود آمدن خطا در میکرو کنترلر یا سایر امکانات جانبی استفاده می شود . با فعال شدن این تایمر بعد از گذشت زمان تعیین شده که میتواند بین 0 تا 16 ثانیه تعیین شود ، میکرو ریست می شود و برنامه از ابتدا اجرا می شود .

توجه داشته باشید که تایمر Watchdog توسط یک شمارنده ی 12 بیتی راه اندازی میشود ، این شمارنده عملکردی کاملا مستقل از cpu و سایر بخش های جانبی دارد وکلاک آن از اسیلاتور مجزا ی 32.768 کیلو هرتزی تامین میشود ، خروجی این تایمر میتواند سیستم (مقادیر کلیه متغیر ها ، حافظه ها و...) یا فقط cpu را ریست کند .

برای واحد Watchdog نیز همچون سایر بخش های جانبی تعداد 3 رجیستر در نظر گرفته شده است که با مقدار دهی این رجیستر های میتوانید این تایمر را تنظیم کنید ، هر چند در ویزارد keil نیز امکان تنظیم کردن این تایمر به صورت گرافیکی وجود دارد (نحوه ی کار با ویزارد در ادامه آورده شده است) :

1- رجیستر WDTC_WDCR (Watchdog Timer Control Register):

31	30	29	28	27	26	25	24
KEY							
23	22	21	20	19	18	17	16
–	–	–	–	–	–	–	–
15	14	13	12	11	10	9	8
–	–	–	–	–	–	–	–
7	6	5	4	3	2	1	0
–	–	–	–	–	–	–	WDRSTT

بیت WDRSTT : با قرار دادن رقم یک به جای این بیت Watchdog ریست میشود .

بیت های KEY : به این بیت ها مقدار 0xA5. داده میشود ، دیگر مقادیر باعث خاموش شدن تایمر میشود (نمیتوانید در تایمر بنویسید)

2- رجیستر WDTC_WDMR (Watchdog Timer Mode Register) :

31	30	29	28	27	26	25	24
–	–	WDIDLEHLT	WDDBGHLT	WDD			
23	22	21	20	19	18	17	16
WDD							
15	14	13	12	11	10	9	8
WDDIS	WDRPROC	WDRSTEN	WDFIEN	WDV			
7	6	5	4	3	2	1	0
WDV							

بیت های WDV : مقدار شمارنده ی تایمر 12 بیتی واحد Watchdog به جای این بیت ها قرار میگیرد . هنگامی که تایمر روشن میشود ، در هر پالس کلاک یک واحد از مقدار نسبت داده شده به این بیت ها کم میشود ، با صفر شدن عدد موجود کل میکرو یا CPU ریست خواهد شد .

مقدار تاخیر زمانی ایجاد شده از رابطه ی زیر محاسبه میشود :

$$TIME = \frac{\text{Clock divided value} * \text{WDT}}{\text{Watchdog Clock value}}$$

با توجه به فرمول بالا ، حداقل زمان قابل شمارش برای این تایمر برابر 3میلی ثانیه و حداکثر زمان برابر 16 ثانیه است .

بیت WDFIEN : با قرار دادن مقدار یک به جای این بیت ، پرچم وقفه ی Watchdog فعال میشود و شما میتوانید با فعال سازی واحد AIC زیر برنامه های وقفه را پیکربندی کنید .

بیت WDRSTEN : با قرار دادن مقدار یک به چای این بیت ، تایمر Watchdog میتواند عملیات ریست را انجام دهد .

بیت WDRPROC : با انتخاب مقدار صفر برای این بیت ، در هنگام صفر شدن شمارنده ی Watchdog کلیه بخش های میکروکنترلر ریست میشوند ، انتخاب مقدار یک باعث میشود تا در هنگام صفر شدن شمارنده فقط CPU ریست شود .

بیت های WDD : شما میتوانید با مقدار دهی این بیت ها ، مقدار مجاز برای شمارنده ی Watchdog را تعیین کنید . در صورتی که مقدار نسبت داده شده به بیت های WDV از این مقدار کمتر باشد با یک شدن بیت WDRSTT تایمر ریست شده و به صوت عادی کار خود را دنبال میکند . در صور تی که مقدار WDV از این مقدار بیشتر باشد ، یک شدن بیت WDRSTT باعث ایجاد خطا در تایمر میشود .

بیت WDDBGHLT : مقدار دهی یک برای این بیت باعث میشود که تایمر در حالت Debug که ممکن است CPU متوقف شود ، خاموش شود . مقدار دهی صفر باعث عمل کرد عادی تایمر میشود (یعنی اگر در حالت دیباگ CPU متوقف شده و تا صفر شدن شمارنده ی تایمر به حالت عادی بازنگردد میکرو ریست میشود) .

بیت WDIDLEHLT : مقدار دهی یک برای این بیت باعث میشود که تایمر در حالت Idle (مدهای کم مصرفی) که ممکن است CPU متوقف شود ، خاموش شود . مقدار دهی صفر باعث عمل کرد عادی تایمر میشود (یعنی اگر در یکی از مد های کم مصرفی خاموش شده و تا صفر شدن شمارنده ی تایمر به حالت عادی بازنگردد میکرو ریست میشود) .

بیت WDDIS : مقدار 1 = تایمر غیر فعال و مقدار صفر = تایمر فعال

3- رجیستر WDTC_WDSR (Watchdog Timer Status Register) :

31	30	29	28	27	26	25	24
–	–	–	–	–	–	–	–
23	22	21	20	19	18	17	16
–	–	–	–	–	–	–	–
15	14	13	12	11	10	9	8
–	–	–	–	–	–	–	–
7	6	5	4	3	2	1	0
–	–	–	–	–	–	WDERR	WDUNF

این رجیستر که وضعیت تایمر Watchdog را مشخص میکند دارای دو بیت به نام WDUNF و WDERR میباشد .

در صورتی که بیت WDUNF یک باشد ، به مفهوم این است که تایمر حداقل یک بار underflow (سر ریز) شده است ، صفر بودن به مفهوم عدم underflow است . یک بودن بیت WDERR به مفهمو رخ دادن یک خطا در هنگام عملکرد تایمر میباشد ، مقدار صفر برای این بیت به مفهوم عملکردن بدون اشکال تایمر است .

مثال :

```
#include <AT91SAM7X256.H>
#define KBD_PORT_A
#define KBD_DATAPORT_OFFSET 7
#define LCD_PORT_B
#define LCD_RS 8
#define LCD_E 10
#define LCD_DB4 12
#define LCD_DB5 13
#define LCD_DB6 14
#define LCD_DB7 15
#include "kbd.h"
#include "lcd.h"
#include "delay.h"
int main (void) {
int a;
*AT91C_WDTC_WDCR= (AT91C_WDTC_WDRSTT|AT91C_WDTC_KEY);
*AT91C_WDTC_WDMR=(AT91C_WDTC_WDRSTEN |AT91C_WDTC_WDDBGHLT
|AT91C_WDTC_WDIDLEHLT | 0X3FF) ; //WD Mode Register= WD Reset Enable , WD
Debug Halt , Watchdog Idle Halt ,WD Counter Value
lcd_init();
while (1) {
```

```
cursor_off();
a=get_kbd(300);
lcd_gotoxy(1,1);
lcd_putsf("KBD:");
lcd_gotoxy(1,7);
lcd_puts(a);
} }
```

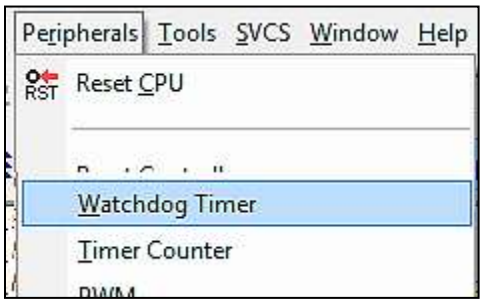
در این برنامه میکرو به صورت عادی خروجی کیپد 4*4 را میخواند و مقادیر دریافت شده از آن را بر روی LCD نمایش میدهد . در صورتی که اختلالی در عملکرد میکرو پیش آید و CPU یا کل سیستم هنگ کند ، تایمر Watchdog واردعمل شده و در صورتی که شرایط بعد از سپری شدن 16 ثانیه به حالت عادی برنگردد ، میکرو ریست میشود .

نحوه شبیه سازی تایمر WATCHDOGدر نرم افزار KEIL:

در شماره های قبل به بررسی بخش های مختلف سیمولاتور keil پرداختیم ، در ادامه به بررسی بخش Watchdog Timer می پردازیم :

ابتدا مثال بالا را کامپایل کنید و سپس وارد محیط شبیه سازی شوید :

از منوی peripheral گزینه ی Watchdog Timer را انتخاب کنید :



پنجره باز شده مکان نمایش اطلاعات مربوط به شبیه سازی بخش Watchdog میباشد ، در ادامه به توضیح بخش های مختلف این پنجره پرداخته ایم :

بخش Control

WDT_CR :در این بخش مقدار رجیستر WDTC_WDCR نمایش داده میشود .

WDRSTT :با فشار دادن این کلید ، تایمر واچ داگ باز نشانی می شود .

بخش Mode

WDT_MR: در ای بخش ، مقدار رجییستر WDTC_WDMR نمایش داده میشود .

WDV : مقدار شمارنده ی تایمر Watchdog که در برنامه تعیین شده است ، در این بخش نمایش داده می شود . .

WDD : مقدار مجاز تایمر برای ریست شدن (مدت زمان مجاز) تایمر در این بخش مشخص می شود .

Timer : مقدار شمارش شده توسط تایمر ، در این بخش نمایش داده می شود .

WDFIEN: این بخش ، وضعیت وقفه تایمر را نمایش میدهد ، تیک کردن این بخش به مفهوم فعال بودن وقفه میباشد .

WDRSTEN : وجود ☑ در این بخش به مفهوم فعال بودن تایمر برای ایجاد سیگنال ریست میباشد .

WDRPROC : وجود ☐ برای این بخش به این مفهموم است که در هنگام صفر شدن شمارنده ی Watchdog کلیه بخش های میکروکنترلر ریست میشوند ، وجود ☑ یعنی ،در هنگام صفر شدن شمارنده فقط CPU ریست شود .

WDDIS : فعال یا غیر فعال بودن تایمر در این بخش مشخص میشود .

:WDDBGHLT

☐ The Watchdog runs when the system is in idle mode.

☑ The Watchdog stops when the system is in idle state.

:WDIDLEHLT

☐ The Watchdog runs when the processor is in debug state.

☑ The Watchdog stops when the processor is in debug state.

بخش Status Group

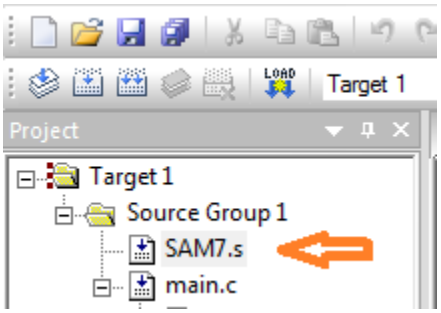
WDT_SR : ریسجتر های WDUNF and WDERR در این بخش نگهداری می شوند .

WDUNF : هنگامی که تایمر Watchdog سرریز می شود ، این بخش ☑ خواهد شد .

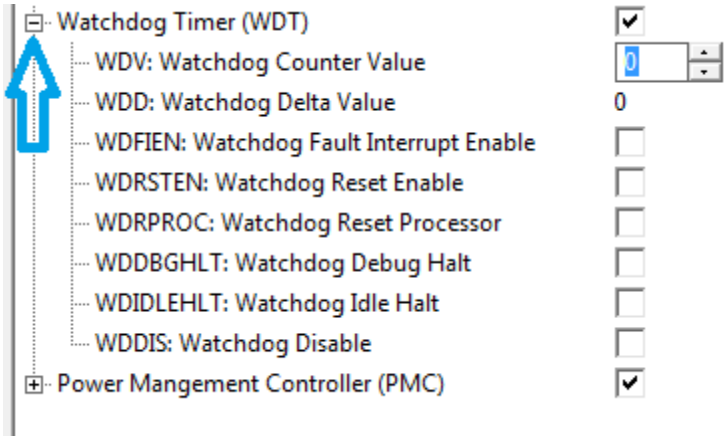
WDERR : هنگامی که در عملکرد Watchdog خطایی رخ دهد این بخش ☑ می شود .

- برنامه بالا را کامپایل کنید و طبق مطالب گفته شده ریست شدن برنامه را مشاهده نمایید .
- در شبیه ساز نرم افزار KEIL ، برای شبیه سازی تایمر Watchdog فرض بر این ایت که میکرو کنترلر هنگ کرده است .
- در ادامه ی نحوه ی کار با جادوگر کامپایلر KEIL برای انجام تنظیمات این تایمر آورده شده است .

در محیط نرم افزار و در پالت Project بر روی گزینه ی sam7.s دوبار کلیک کنید و در پایین پنجره ی باز شده گزینه ی Configuration Wizard را انتخاب نمایید .



بر روی زبانه ی Watchdog Timer کلیک کنید تا بتوانید گزینه های موجود در آن را مشاهده نمایید :



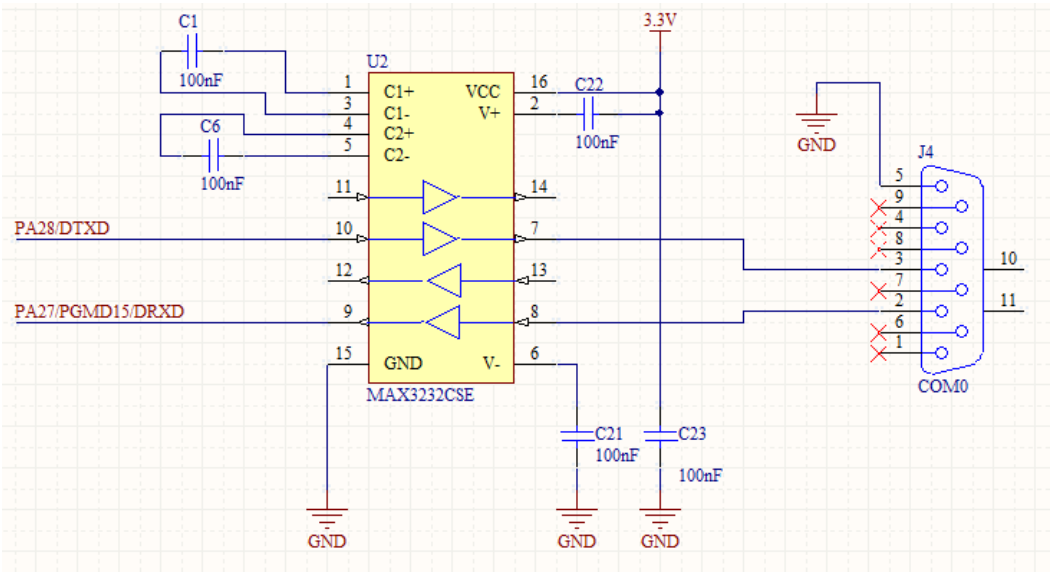
ما در صفحه ی 19 کلیه موارد موجود در این بخش را توضیح دادیم ، گزینه های دلخواه را انتخاب نمایید و سپس برنامه را کامپایل کنید تا کد متناظر به فایل sam7.s و کد هگز پروژه افزوده شود .

واحد Debug Unit در میکرو کنترلر های ARM :

در سری AT91SAM واحدی به نام Debug Unit وجود دارد که توسط آن میتوان موارد زیر را انجام داد :

- ارسال متغیر یا رشته ی دلخواه به میکروکنترلر
- دریافت متغیر یا رشته ی دلخواه از میکرو میکروکنترلر
- دریافت مقادیر رجیستر ها ، متغیر ها و مقادیر خانه های حافظه
- دریافت ورژن ، نوع هسته (میکرو کنترلر از چه هسته ای استفاده میکند (ARM946ES یا ARM7TDMI یا ...)) و خانواده ی میکرو کنترلر (میکرو جزء کدام گروه از تولیدات اتمل میباشد (AT91SAM7Sxx Series یا AT91x55 Series یا ...))
- دریافت مقدار حجم برنامه ی موجود در حافظه ی فلش ، مقدار فضا ی حافظه ی sram داخلی .
- نوشتن مقادیر دلخواه در خانه های حافظه یا متغیر های به کار رفته در برنامه
- و..

این واحد عمل کردی مشابه با واحد usart دارد ، برای کار با آن شما باید پایه های DTXD و DRXD را مطابق شماتیک زیر به پورت com کامپیوتر خود متصل نمایید .



در کامپایلر keil تعداد 12 رجیستر برای راه اندازی و کار با این واحد در نظر گرفته شده است که نام و کاربرد آنها را در جدول زیر مشاهده میکنید :

نام رجیستر	نام کامل	کاربرد
DBGU_CR	Debug Unit Control Register	تنظیمات ارسال/دریافت و بازنشانی داده توسط این رجیستر انجام میشود
DBGU_MR	Debug Unit Mode Register	تنظیمات مربوط به بیت خطا و مد ارسال داده توسط این رجیستر انجام میشود
DBGU_IER	Debug Unit Interrupt Enable Register	وقفه بخش های مختلف واحد Debug Unit توسط این رجیستر فعال میشود
DBGU_IDR	Debug Unit Interrupt Disable Register	وقفه بخش های مختلف واحد Debug Unit توسط این رجیستر غیر فعال میشود
DBGU_IMR	Debug Unit Interrupt Mask Register	نوع پوشش وقفه های فعال شده توسط این رجیستر تعیین میشود
DBGU_SR	Debug Unit Status Register	وضعیت بخش های که توسط رجیستر های قبلی فعال شده در این رجیستر ذخیره میشود
DBGU_RHR	Debug Unit Receive Holding Register	داده ی دریافت شده در 8 بیت اول این رجیستر ذخیره می شود
DBGU_THR	Debug Unit Transmit Holding Register	داده ای که قرار است ارسال شود باید در 8 بیت اول این رجیستر قرار گیرد
DBGU_BRGR	Debug Unit Baud Rate Generator Register	مقدار نرخ انتقال داده توسط این رجیستر تعیین میشود .
DBGU_CIDR	Debug Unit Chip ID Register	ورژن ، نوع هسته ، مقدار فضای حافظه ی فلش و sram و خانواده ی میکرو کنترلر در این رجیستر ذخیره میشود .
DBGU_EXID	Debug Unit Chip ID Extension Register	سایر مشخصات قطعه در این رجیستر ذخیره میشود .
DBGU_FNR	Debug Unit Force NTRST Register	نحوه ی ریست شدن میکرو با مقدار دهی این رجیستر انجام میشود

توضیحات :

- ✓ برای دست یابی به کلیه اطلاعات شما باید آنها را در رجیستر DBGU_THR قرار داده و به کامپیوتر ارسال کنید .

- ✓ در صورتی که اطلاعات شما بیشتر از 8 بیت میباشد ، باید آن را به بخش های 8 بیتی تقسیم کنید (با استفاده از دستور شیفت)
- ✓ برای ارسال متغیر ها و مقادیر خانه های حافظه به کامپیوتر ، باید آدرسشان را در رجیستر DBGU_THR قرار دهید .
- ✓ با مراجعه به صفحات 215 تا 228 دیتا شیت AT91SAM7X512/256/128 Preliminary میتوانید اطلاعات بیشتر در مورد مقدار دهی رجیستر های فوق بدست آورید .
- ✓ به دلیل کاربرد عمومی این بخش در تبادل داده با نمایشگر (کامپیوتر) از توضیحات اضافه در مورد رجیستر ها خوداری کرده و به تشریح کتابخانه ی dbgu.h میپردازیم .

بررسی کتابخانه ی dbgu.h :

ابتدا فایل dbgu.h را در مسیر زیر کپی کنید (فایل در پوشه پیوست موجود میباشد) :

```
\keil\ARM\INC\Atmel\SAM7X
```

برای استفاده از این کتابخانه ، ابتدا باید آن را در برنامه فراخوانی کنید ، برای اینکار از دستور زیر استفاده میشود :

```
#include " dbgu.h"

d_Init(Baudrate);
```

با استفاده از دستور بالا واحد دیباگ راه اندازی شده و آماده ی استفاده میباشد ، شما باید به جای Baudrate نرخ انتقال داده ی دلخواه را درج نمایید .

نرخ انتقال داده ی درج شده در بالا باید با نرخ انتقال داده ی درج شده در برنامه کامپوتر یکی باشد ، شما میتوانید برای مشاهده ی اطلاعات دیباگ از نرم افزار های هایپر ترمینال ویندوز ، ترمینال ایمولاتور بسکام ، دیباگ پورت keil یا هر نرم افزار دیگری که قادر به ارتباط با پورت سریال است استفاده کنید .

با اجرا شدن دستور بالا موارد زیر در محیط ترمینال نمایش داده میشود :

```
device has been installed! press a key to continue ...
```

نمایش این پیغام به مفهوم درست بودن اتصالات و اجرای صحیح برنامه میباشد ، با فشردن یکی از کلید ها به مرحله ی بعد میروید :

this library is provide by 1NAFAR (at91sam7x645@gmail.com) for Atmel's at91sam series to access Debug Unit(DBGU) port. The Debug Unit provide a single entry point from the processor for access to all the debugca_ pabilitie of Atmel's ARM-based systems. for more information visit WWW.IRANMICRO.IR press E key to Exit and run your programs or press I for get chip ID

این متن اطلاعاتی را در مورد این واحد به شما میدهد ، شما میتوانید با فشردن کلید E از این بخش صرف نظر کرده وبرنامه ی خود را اجرا نمایید ، یا کلید I را فشار دهید تا به مرحله ی بعد بروید ، در مرحله ی بعد اطلاعات زیر به نمایش در می آید :

chip name=XXXXXXXXX

XXXXXX نام میکرو کنترلر موجود است.

Embedded Processor=XXXXX

نام هسته ی استفاده شده در میکرو کنترلر است .

Nonvolatile Program Memory Size=XXXXX

میزان حافظه ی قطعه را نمایش میدهد .

Internal SRAM Size=XXXXX

مقدار حافظه ی SRAM داخلی قطعه را نمایش میدهد .

Architecture Identifier=XXXXX

نام خانواده ی قطعه در گروه بندی اتمل را نمایش میدهد .

Nonvolatile Program Memory Type is=XX

نام حافظه های که در این برنامه مورد استفاده قرار گرفته است را نمایش میدهد .

ping chip finished. press any key to Exit and run your programs

و در نهایت این بخش با جمله ی بالا به پایان میرسد ، شما میتوانید با فشردن یکی از کلید ها برنامه ی خود را دنبال کنید .

d_ printf ("string");

توسط این دستور میتوانید رشته ی دلخواه را به پورت سریال کامپیوتر ارسال کنید ، رشته ی string میتواند مجموعه ای از اعداد وحروف که در میان دو عدد " قرار گرفته اند باشد .

ارسال کارکتر های \n و \t و \r بعد از رشته ی string به ترتیب باعث رفتن به خط بعد ، ایجاد فاصله به اندازی 5 حرف و پاک شدن صفحه میشود (مثال را ببنید) .

d_print(var);

توسط این دستور میتوانید یک متغیر یا عدد علامت دار را به پورت سریال کامپیوتر ارسال کنید

d_scanf(x);

از دستور scanf برای دریافت داده از پورت سریال کامپیوتر استفاده می شود . داده ی ارسال شده از طرف کامپیوتر در متغیر x قرار میگیرد .

d_ dump();

با قرار دادن دستور بالا در برنامه ، با فشردن کلید M میتوانید برنامه را به حالت دیباگ برده و وضعیت متغیر ها و رجیستر های مختلف را مشاهده کنید ، شما همچنین میتوانید مقدار و وضعیت آنها را تغییر دهید .

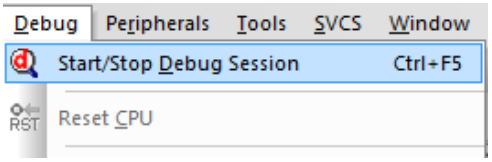
میکرو را روشن کرده و بعد از انجام داده موراد بالا کلید Hرا فشار دهید ، اطلاعات نمایش داده شده در محیط ترمینال شما را در هنگام کار یاری میدهد .

مشاهده ی اطلاعات در نرم افزار keil :

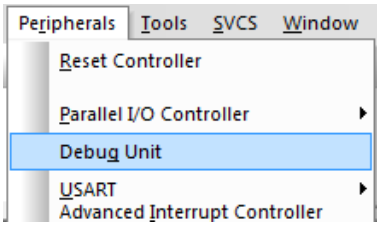
```
#include "AT91SAM7x256.h"
#include "debug.h"
#include "delay.h"
int main(void){
d_init(9600);
```

```
while (1){
delay_s(1);
d_printf("TEST\n");
 delay_s(1);
d_dump();
}}
```

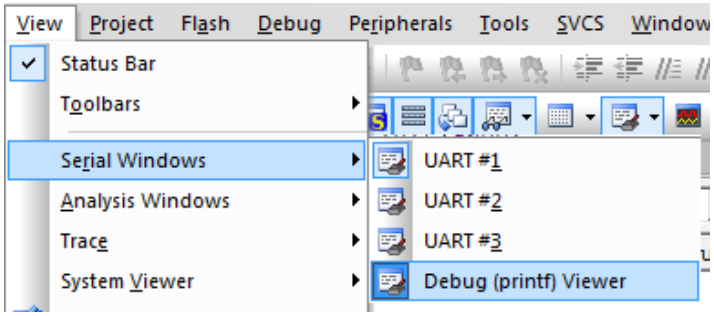
بعد از کامپایلر کردن برنامه و انتقال آن به میکرو کنترلر ، به محیط شبیه سازی نرم افزار وارد شوید :



از منوی peripheral گزینه ی Debug Unit را انتخاب کنید :



از مسیر VIEW > Serial windows گزینه ی debug (printf) viewer را انتخاب نمایید :



عملیات شبیه سازی برنامه را آغاز کنید ، شما میتوانید در بخش اول مقدار رجیستر ها و بیت های مخصوص این واحد و در بخش دوم داده ی دریافتی و ارسالی آن را مشاهده نمایید . شما میتوانید واحد دیباگ را همانند بخش usart در نرم افزار keil شبیه سازی کنید ، برای کسب اطلاعات بیشتر به مجله شماره ی 6–7 (ویرایش دوم) مراجعه کنید .

برای مشاهده ی اطلاعات به صورت زنده ، علاوه بر محیط بالا ، میتوانید از سایر برنامه های پورت سریال نظیر ترمینال ایمولاتور بسکام ، هایپر ترمینال ویندوز و .... استفاده نمایید . برای این نرم افزار ها تنظیمات No parity,Channel Mode=Normal Mode را انجام دهید .

واحد Real-time Timer

Real-time Timer یا زمان شمار واقعی یکی از بخش های کم نظیر میکرو کنترلر های سری At91sam میباشد . این تایمر که از یک کانتر 32 بیتی و یک واحد مقایسه گر (برای ایجاد سیگنال هشدار(Alarm)) تشکیل شده است ، با استفاده از کلاک فرکانس پایین (کریستال 32768 هرتز) میتواند تا مدت 2^{32} ثانیه یا 136 سال ، زمان را بدون هیچ گونه خطا شمارش کند .

این تایمر میتواند در هر شمارش ، سیگنال هشدار یا تریگر بخش های جانبی را ایجاد کند ، شما همچنین میتوانید از آن برای تولید زمان های دلخواه استفاده نمایید . کار با این تایمر بسیار ساده بوده و با چهار رجیستر زیر میتوان آن را راه اندازی نمود :

رجیستر RTTC_RTMR (Real-time Timer Mode Register)

31	30	29	28	27	26	25	24
–	–	–	–	–	–	–	–
23	22	21	20	19	18	17	16
–	–	–	–	–	RTTRST	RTTINCIEN	ALMIEN
15	14	13	12	11	10	9	8
RTPRES							
7	6	5	4	3	2	1	0
RTPRES							

بیت های RTPRES : واحد RTT برای شمارش زمان منقضی استفاده میشود ، این تایمر از یک کانتر 32 بیتی که پالس کلاک آن از تقسیم شدن کلاک فرکانس پایین بر مقدار بیت های RTPRES تایمن میشود ، برای شمارش زمان استفاده میکند ، برای مثال اگر مقدار RTPRES برابر با 0x00008000 (32768 در مبنای دسیمال) باشد ، تایمر با کلاک 1 هرتز کار میکند ، در این حالت شمارش هر پالس یک ثانیه به طول خواهد انجامید .

بیت ALMIEN : با مقدار دهی این بیت با رقم یک ، هنگامی که مقدار شمرده شده توسط تایمر با مقدار رجیستر RTT_AR (که در ادامه به بررسی آن خواهیم پرداخت) برابر شد ، بیت ALMS در رجیستر RTT_SR (که در ادامه به بررسی آن خواهیم پرداخت) یک میشود . مقدار دهی این رجیستر با رقم صفر باعث غیر فعال شدن هشدار میشود .

بیت RTTINCIEN : مقدار دهی این رجیستر با رقم یک باعث فعال شدن وقفه ی تایمر میشود ، رقم صفر باعث غیر فعال شدن وقفه خواهد شد ، توضیحات بیشتر در ادامه آورده شده است .

بیت RTTRST : مقدار دهی این بیت با مقدار 1 باعث ریست شدن کانتر و محتوای بیت های RTPRES میشود .

رجیستر RTTC_RTAR (Real-time Timer Alarm Register) :

31	30	29	28	27	26	25	24
ALMV							

23	22	21	20	19	18	17	16
ALMV							

15	14	13	12	11	10	9	8
ALMV							

7	6	5	4	3	2	1	0
ALMV							

بیت های ALMV : رجیستر RTTC_RTVR ، رجیستر مقایسه ای واحد RTT میباشد . با فعال شدن این واحد در هر بار شمارش ، مقدار شمرده شده با مقدار این رجیستر مقایسه می شود و در صورتی که برابری و یک بودن بیت ALMIEN در رجیستر RTTC_RTMR بیت بیت ALMS در رجیستر RTTC_RTSR (که در ادامه به بررسی آن خواهیم پرداخت) یک میشود . در حالت پیش فرض مقدار بیت های ALMV برابر با 0Xffffffff میباشد .

رجیستر RTTC_RTVR (Real-time Timer Value Register) :

31	30	29	28	27	26	25	24
CRTV							

23	22	21	20	19	18	17	16
CRTV							

15	14	13	12	11	10	9	8
CRTV							

7	6	5	4	3	2	1	0
CRTV							

بیت های CRTV : مقدار شمرده شده توسط کانتر ، در این بیت ها ذخیره میشود و شما میتوانید با خواندن این رجیستر از زمان سپری شده با خبر شوید .

رجیستر RTTC_RTSR (Real-time Timer Status Register)

31	30	29	28	27	26	25	24
–	–	–	–	–	–	–	–

23	22	21	20	19	18	17	16
–	–	–	–	–	–	–	–

15	14	13	12	11	10	9	8
–	–	–	–	–	–	–	–

7	6	5	4	3	2	1	0
–	–	–	–	–	–	RTTINC	ALMS

رجیستر RTT_SR وضعیت واحد RTT را در خود ذخیره میکند و شما میتوانید با خواندن آن از رویداد های رخ داده با خبر شوید .

بیت ALMS :

0 : مقدار شمرده شده توسط کانتر با مقدار رجیستر RTTC_RTAR برابر نشده است (توجه داشته باشید که برای استفاده از این بیت باید بیت ALMIEN در رجیستر RTT_MR یک باشد)

1 : مقدار شمرده شده توسط کانتر با مقدار رجیستر RTTC_RTAR برابر شده است .

بیت RTTINC : این بیت هنگامی که مقدار کانتر یک واحد افزایش میباشد ، یک میشود ، از این بیت میتوان برای ایجاد یک پالس متناوب یا تحریک بخش های داخلی و خارجی یا ... استفاده کرد .

0: مقدار کانتر هنوز افزایش نیافته است

1 : مقدار کانتر افزایش یافته است .

توجه داشته باشید که بیت RTTINC بعد از هر بار خواندن رجیستر RTT_SR فعال میشود ، مثلا اگر شما مقدار رجستر RTT_SR را بخوانید و به زیر برنامه ای دیگر بروید و مجددا مقدار RTT_SR را بخوانید ، در صورتی که محتوای کانتر افزایش یافته باشد این بیت یک میشود .

مثال :

```
#include <AT91SAM7X256.H>
#include "pio.h"
int main (void) {
*AT91C_RTTC_RTMR = (0x1 << 16)|(0x2000 << 0)|(0x1 << 18);
//RTT Mode Register= Alarm Enable, counter fed by 32.768K divided by 0x2000 , RTT Restart
*AT91C_RTTC_RTAR=0XFFFF;
  PORTB_OUTPUT= 0xFFFF;
            while(1){
SET_PORTB= AT91C_RTTC_CRTV;
RST_PORTB=~AT91C_RTTC_CRTV;
}
}
```

در پروژه ی بالا مقدار شمرده شده توسط تایمر بر روی پورت B قرار داده شده است ، توجه داشته باشید که برای اجرای عملی این مثال باید مقدار کریستال میکرو را به 32.768 کیلوهرتز تغییر دهید .

ضمیمه چهار : دستورات ریاضی کامپایلر keil:

برای استفاده از این دستورات باید کتابخانه ی <math.h> را با دستور زیر در برنامه خود فراخوانی کنید :

```
#include <math.h>
```

توابع مثلثاتی :

ما از قبل با این توابع آشنا شدیم و از آنها برای استفاده های زیادی همچون ، بدست آوردن ضریب قدرت ، حجم های برداری توان ، زوایای کشش و.... استفاده می کردیم ، شما می توانید این توابع را با ARM و توسط نوشتن دستورات زیر در کامپایلر keil محاسبه کنید .

دستور cos :

این دستور به فرم کلی زیر است :

```
k=cos(x);
```

در این دستور xمقدار متغیر ورودی و k حاصل عملیات است ، مثال :

```
float k,x=45;

k=cos(x);
```

در مثال بالا مقدار x برابر با 45 است ، با دستور cos مقدار کسینوس x محاسبه شده و در متغیر k ریخته می شود ، بعد از انجام دستور مقدار k برابر با .707 می شود .

دستور sin :

این دستور به فرم کلی زیر است :

```
k= sin (x);
```

در این دستور xمقدار متغیر ورودی و k حاصل عملیات است ، مثال :

```
float k,x=45;

k= sin (x);
```

در مثال بالا مقدار x برابر با 45 است ، با دستور sin مقدار سینوس x محاسبه شده و در متغیر k ریخته می شود ، بعد از انجام دستور مقدار k برابر با .707 می شود

- دستور tan :

این دستور به فرم کلی زیر است :

```
k= tan (x);
```

در این دستور xمقدار متغیر ورودی و k حاصل عملیات است ، مثال :

```
float k,x=45;

k= tan (x);
```

در مثال بالا مقدار x برابر با 45 است ، با دستور tan مقدار تانژانت x محاسبه شده و در متغیر k ریخته می شود ، بعد از انجام دستور مقدار k برابر با 1 می شود

- دستور cot :

در کامپایلر keil دستوری برای بدست آوردن مقدار کتانژانت یک متغیر وجود ندارد ، اما شما می توانید با معکوس کردن مقدار تانژانت یک متغیر ، مقدار کتانژانت آن را بدست اورید ، مثال :

```
float k,x=45;

k= tan (x);

k=1/k;
```

توابع مثلثاتی هیپربولیک

از توابع هیپربولیک برای توصیف حرکت موج در اجسام کشسان، شکل خطوط انتقال نیروی برق، توزیع دما در پروژه های فلزی که لوله های داغ را سرد می کنند، خم های تعقیب و نظریه ی نسبیت و.... استفاده می شود . این توابع برای سینوس و کسینوس دارای تعریف زیر است :

سینوس هیپربولیک (sinh) (ex - e-x)/2

کسینوس هیپربولیک (cosh) (ex + e-x)/2

همانطور که می دانید تانژانت از تقسیم سینوس به کسینوس و کتانژانت از تقسیم کسینوس به سینوس هیپربولیک بدست می آید . در ادامه مجموعه توابع محاسبات ، مربوط به موارد بالا که در نرم افزار keil وجود دارد آورده شده است :

دستور cosh :

این دستور به فرم کلی زیر است :

```
k=cosh(x);
```

در این دستور xمقدار متغیر ورودی و k حاصل عملیات است ، مثال :

```
float k,x=45;

k=cosh(x);
```

در مثال بالا مقدار x برابر با 45 است ، با دستور cosh مقدار کسینوس هیپربولیک x محاسبه شده و در متغیر k ریخته می شود ، بعد از انجام دستور مقدار k برابر با 1746713552874254767 4.017398616703 می شود .

دستور sinh :

این دستور به فرم کلی زیر است :

```
k= sinh (x);
```

در این دستور xمقدار متغیر ورودی و k حاصل عملیات است ، مثال :

```
float k,x=45;

k= sinh (x);
```

در مثال بالا مقدار x برابر با 45 است ، با دستور sinh مقدار سینوس هیپربولیک x محاسبه شده و در متغیر k ریخته می شود ، بعد از انجام دستور مقدار k برابر با 1746713552874254 7.017398616703 می شود

دستور tanh :

این دستور به فرم کلی زیر است :

```
k= tanh (x);
```

در این دستور xمقدار متغیر ورودی و k حاصل عملیات است ، مثال :

```
float k,x=45;

k= tan (x);
```

در مثال بالا مقدار x برابر با 45 است ، با دستور tanh مقدار تانژانت هیپربولیک x محاسبه شده و در متغیر k ریخته می شود ، بعد از انجام دستور مقدار k برابر با 1 می شود

نکته :

-محدوده ی توابع هیپربولیک : sinh x : بین منفی و مثبت بینهایت

cosh x: از مثبت 1 تا مثبت بینهایت

tanh x: از −1 تا +1

-شکل تابع cosh x تقریبا مانند سهمی رو به بالاست و شکل sinh x تقریبا مثل یک تابع درجه 3 است ،

-همانطور که قبلا نیز گفته شد تانژانت از تقسیم سینوس به کسینوس و کتانژانت از تقسیم کسینوس به سینوس هیپربولیک بدست می آید ، مثال :

```
k= sinh (x);

k2=cosh(x);

tanh=k/k2;

coth=k2/k;
```

coth و tanh دو متغیر از نوی فلوت یا ... هستند .

آرک توابع مثلثاتی :

منشا توابع مثلثاتی معکوس، مسائلی است که در آن ها باید با استفاده از اندازه ی اضلاع یک مثلث، زوایای آن را به دست آورد. این توابع، پاد مشتق بسیاری از توابع دیگر هم هستند و لذا در جواب های تعدادی از معادلات دیفرانسیل مورد بحث در ریاضیات، مهندسی و فیزیک ظاهر می شوند ، در واقع با ارک شما مسیر بدست آوردن تابع را معکوس طی می کنید و از جواب به مقدار زاویه می رسید .

دستور acos :

این دستور به فرم کلی زیر است :

```
k=acos(x);
```

در این دستور xمقدار متغیر ورودی و k حاصل عملیات است ، مثال :

```
float k,x=.707;

k=acos(x);
```

در مثال بالا مقدار x برابر با .705 است ، با دستور acos مقدار ارک کسینوس x محاسبه شده و در متغیر k ریخته می شود ، بعد از انجام دستور مقدار k برابر با 45 می شود .

دستور asin :

این دستور به فرم کلی زیر است :

```
k= sin (x);
```

در این دستور xمقدار متغیر ورودی و k حاصل عملیات است ، مثال :

```
float k,x=45;

k= asin (x);
```

در مثال بالا مقدار x برابر با .707 است ، با دستور asin مقدار ارک سینوس x محاسبه شده و در متغیر k ریخته می شود ، بعد از انجام دستور مقدار k برابر با 45 می شود

- دستور atan:

این دستور به فرم کلی زیر است :

```
k= atan (x);
```

در این دستور xمقدار متغیر ورودی و k حاصل عملیات است ، مثال :

```
float k,x=45;

k= atan (x);
```

در مثال بالا مقدار x برابر با 1است ، با دستور atan مقدار ارک تانژانت x محاسبه شده و در متغیر k ریخته می شود ، بعد از انجام دستور مقدار k برابر با 45 می شود.

- دستور atan2 :

این دستور به فرم کلی زیر است :

```
k= atan2(x,y);
```

با دستور با مقدار تانژانت معکوس دو نقطه نسبت به محور مختصات بدست می آید ، مثال

```
float k,x=-3,y=-6;
```

```
k= atan (x,y);
```

در مثال بالا مقدار x برابر با 3- و مقدار y برابر با6- است ، با دستور atan مقدار ارک تانژانت x,y محاسبه شده و در متغیر k ریخته می شود ، بعد از انجام دستور مقدار k برابر با 243.5 می شود.

برای بدست آوردن اطلاعات بیشتر در مورد نسبت های مثلثاتی به کتاب ریاضیات سال های دوم و سوم دبیرستان مراجعه کنید .

- دستور exp(x) :

این دستور به فرم کلی زیر است :

```
k= exp(x) ;
```

با این دستور مقدار e (عدد نپر) به توانx را محاسبه می کند ، x مقدار ورودی است و می تواند یک عدد باشد .

- دستور log(x) :

این دستور به فرم کلی زیر است :

```
k= log (x) ;
```

با این دستور مقدار لگارتیم x در مبنای e محاسبه شده و در متغیر k ریخته می شود . این دستور معکوس دستور قبل است .

- دستور log10(x) :

این دستور به فرم کلی زیر است :

```
k= log10(x) ;
```

با این دستور مقدار لگارتیم x در مبنای 10 محاسبه شده و در متغیر k ریخته می شود.

- دستور pow(x,y):

این دستور به فرم کلی زیر است :

```
k= pow(x,y);
```

با این دستور متغیر x به توان متغیر y می رسد و در متغیر k ریخته می شود.

- دستور pow(x,y):

این دستور به فرم کلی زیر است :

```
k= pow(x,y);
```

با این دستور متغیر x به توان متغیر y می ر سد و در متغیر k ریخته می شود.

- دستور fmod(x,y) :

این دستور به فرم کلی زیر است :

```
k= fmod(x,y);
```

با این دستور باقیمانده حاصل x تقسیم بر y محاسبه شده و در متغیر k ریخته می شود .

- دستور sqrt(x) :

این دستور به فرم کلی زیر است :

```
k= sqrt(x);
```

با این دستور جذر عدد x محاسبه شده و در متغیر k ریخته می شود ، مقدار x باید مثبت باشد .

- دستور ceil(x):

این دستور به فرم کلی زیر است :

```
k= ceil(x);
```

با این دستور عدد اعشاری x به عدد رند بعدی ، تبدیل می شود ، مثال

```
float k,x3.1;
```

```
k= ceil(x);
```

در مثال بالا x برابر با 3.1 است ، با اجرا شدن دستور مقدار k برابر با 4 می شود . این دستور برای اعداد منفی نیز صادق است ، مثلا -10.3 به -10 تبدیل می شود .

دستور floor(x) :

این دستور به فرم کلی زیر است :

```
k= floor(x);
```

با این دستور عدد اعشاری x به عدد رند قبلی ، تبدیل می شود ، مثال

```
float k,x3.1;

k= floor(x);
```

در مثال بالا x برابر با 3.1 است ، با اجرا شدن دستور مقدار k برابر با 3 می شود . این دستور برای اعداد منفی نیز صادق است ، مثلا -10.3 به -11 تبدیل می شود .

دستور ldexp(x):

این دستور به فرم کلی زیر است :

```
k= ldexp(x);
```

با این دستور عدد 2 به توان x می رسد.

مثال

```
float k,x=10;

k= ldexp(x);
```

در مثال بالا x برابر با 10 است ، با اجرا شدن دستور مقدار k برابر با 1024 می شود . در واقع این دستور معکوس دستور pow (2,y) است و مقدار y را بر می گرداند .

دستور frexp(x):

این دستور به فرم کلی زیر است :

```
k= frexp(x);
```

با این دستور عدد x به توان های از عدد2 تبدیل می شود . مثال

```
float k,x=1024;

k= frexp(x);
```

در مثال بالا x برابر با 1024 است ، با اجرا شدن دستور مقدار k برابر با 10 می شود . در واقع این دستور معکوس دستور ldexp(x)است .

دستور (modf x,y):

این دستور به فرم کلی زیر است :

```
k= modf(x,&y);
```

با این دستور مقدار اعشاری یک متغیر اعشاری از قسمت صحیح آن جدا می شود ،

مثال :

```
double k,x, y;


 x = 3.14159265;

 k = modf (x , &y);
```

در مثال بالا x برابر با 3.14159265 است ، با اجرا شدن دستور مقدار k برابر با 3.000000 و مقدار y برابر با 0.141593 می شود .

دستور abs (x):

```
k= abs(x);
```

با این قدر مطلق x را محاسبه می کند (عدد منفی را مثبت می کند) .

دستورات مبدل داده و کار با رشته ها

- دستور atof (x):

```
n = atof ( szInput );
```

این دستور یک متغیر از نوع رشته را به دبل تبدیل می کند (Convert string to double) .

- دستور atol (x):

```
n = atol ( szInput );
```

این دستور یک متغیر از نوع رشته را به لانگ مثبت تبدیل می کند (Convert string to long integer) .

- دستور atoi (x):

```
n = atoi ( szInput );
```

این دستور یک متغیر از نوع رشته را به اینتجر تبدیل می کند (Convert string to integer) .

- دستور atoi (x):

```
n = atoi ( szInput );
```

این دستور یک متغیر از نوع رشته را به اینتجر تبدیل می کند (Convert string to integer) .