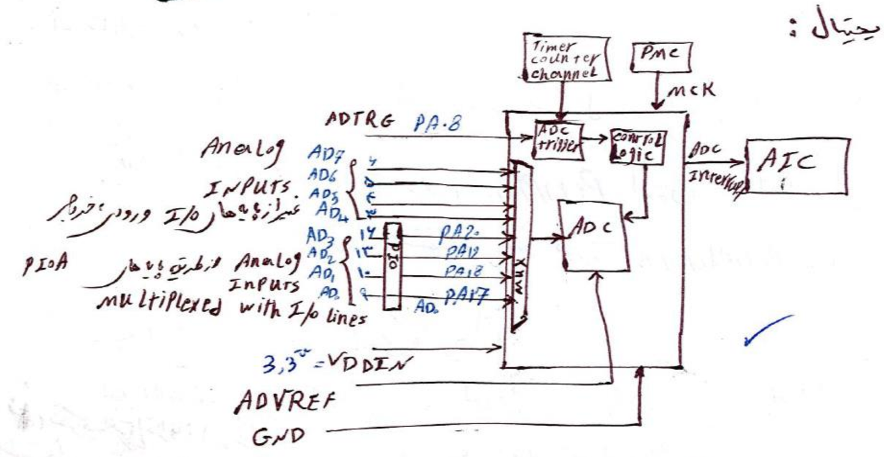
45 - ARM

مبدل آنالوگ به دیجیتال :



ADC controller

نکات مهم :

* یک مبدل ۱۰ بیتی با قابلیت ۸ بیتی شدن (توسط بیت در رجیستر مربوطه)
* انتخاب مکرر از ۸ ورودی آنالوگ که چهار ورودی از طریق PIOA $(AD_0 - AD_3)$ و چهار ورودی مستقل $(AD_4 - AD_7)$
* چهار ورودی ADC که از PIOA انتخاب می‌شوند نه حالت peripheral A هستند نه peripheral B بلکه حالتی اضافی هستند که در صورت فعال شدن توسط بیت در رجیستر مربوطه می‌توان از آنها استفاده کرد. یعنی کاربرد ویژه تعطیل می‌شود
* فعال سازی کلاک ADC: AT91C_BASE_PMC → PMC_PCER=(1<<AT91C_ID_ADC); ✓
* محدوده ولتاژ آنالوگ ورودی از صفر تا ADVREF می‌باشد.
* محدوده ADVREF برای حالت ۱۰ بیتی از 2,6V تا VDDINT (3,3V) و برای حالت ۸ بیتی از 2,5V تا VDDINT می‌باشد.
* نتیجه تبدیل هر کانال در رجیستر اختصاصی آن ریخته می‌شود.
* شروع عملیات تبدیل در ADC توسط تریگر
  1. نرم افزاری (با کد بیت مربوطه)
  2. سخت افزاری
     1. لبه بالا رونده سیگنال ورودی ADTRG
     2. لبه بالا رونده خروجی T/C ها در حالت waveform از TIOA ها
* واحد ADC مجهز به مد sleep و متوالی گر تبدیل (conversion sequencer) می‌باشد.
* مقاومت PULL UP پین‌های PIO بصورت پیش فرض بعد از ریست شدن فعال می‌شوند. بهتر است برای استفاده از این پینها بعنوان ورودی آنالوگ، ابتدا مقاومت‌های PULL UP غیر فعال شوند. لازم نیست گفته شود.

44

منابع تحریک :

۱ـ منابع تحریک سخت افزاری

الف) ADTRG : لبهٔ بالارونده‌ٔ سیگنال ورودی از این پایه باعث تحریک ADC می‌شود.

تذکر : این پین بصورت Peripheral B از پین PA.8 بر روی PIO قرار دارد
یعنی قبل از استفاده باید بصورت PeripheralB پیکره بندی می‌شود.

ب) خروجی یکی از تایمر/کانترها

لبه بالارونده‌ٔ یکی از سیگنالهای TIOA0 ، TIOA1 و TIOA2

۲) تحریک نرم افزاری: با نوشتن یک در بیت مربوطه تبدیل شروع می‌شود

رجیسترهای مربوط با ADC :

~~۲ـ منبع تحریک نرم افزاری~~

۱ـ ADC_CR :

با نوشتن یک در بیت START از رجیستر ADC_CR (control)، ADC شروع به تبدیل ولتاژ ورودی کانالهای فعال می‌کند.

ADC_CR		...	START	SWRST
31				

تذکر: این تبدیل متوالیاً ادامه دارد مگر آنکه کانالی غیرفعال گردد. و دوباره اگر بخواهیم تبدیل ادامه پیدا کند مجدداً آنرا (START) یک می‌کنیم. (زیرا با شروع تبدیل این بیت صفر می‌شود.) (همیشه داخل حلقهٔ while(1) نوشته می‌شود در نرم افزار)

یکبار برای همهٔ کانالها

یک شدن بیت SWRST (software Reset) باعث Reset ، ADC می‌شود.

```
AT91C_BASE_ADC->ADC_CR=0x2;
```

۲) معرفی رجیستر ADC_MR (mode) :

ADC_MR	...	SHTIM	START UP	PRESCAL	...	SLEEP	LOWERS	TRGSEL	TRGEN
	31 … 28	27 … 24	23 … 16	15 … 8		5	4	3 2 1	0

① TRGEN
- =0 → منبع تحریک نرم افزاری (پیش فرض)
- =1 → منبع تحریک سخت افزاری

② TRGSEL

- 0 0 0 → TIOA0
- 0 0 1 → TIOA1
- 0 1 0 → TIOA2
- 1 1 0 → EXTERNAL Trigger (ADTRG)

③ LOWERS → =0 → ۱۰ بیتی (پیش‌فرض) ADC
→ =1 → ۸ بیتی ADC

④ SLEEP → =0 → Normal mode (پیش‌فرض)
→ =1 → sleep mode
در لحظه تبدیل در مد sleep می‌رود و با اتمام تبدیل از حالت خواب بیدار شود (در بار SLEEP=1 ... شود.)

⑤ بیتهای PRESCAL

مقدار کلاک ADC بصورت زیر محاسبه می‌شود:

$$\text{ADC clock} = \frac{MCK}{(PRESCAL+1)\times 2}$$

محدوده کلاک ADC بین $\frac{MCK}{128}$ و $MCK/2$

بنابراین محدوده PRESCAL بین 0 تا 63 می‌باشد.

۶. بیتهای START UP

در هر ADC مدت زمانی طول می‌کشد تا عمل شروع به تبدیل ولتاژ ورودی کند (startup_Time) (قبل از آغاز تبدیل) بعد از start، ADC آماده بکار شود.

$$\text{start up\_time} = \frac{(STARTUP+1)\times 8}{\text{ADC clock}}$$

۷. بیتهای SHTIM

مدت چند سیکل ADC از ولتاژ ورودی نمونه برداری می‌کند و آن را نگه می‌دارد تا در حین انجام عملیات تبدیل، مقدار آن بدون تغییر باقی بماند. (نمونه برداری "قبل از شروع تبدیل")

$$\text{Sample \& Hold Time} = \frac{SHTIM}{\text{ADC clock}}$$

نکته: ADC در طول ۱۰ سیکل از سیگنال ADC clock، مقدار آنالوگ نمونه برداری شده را به مقدار دیجیتال متناظرش تبدیل می‌کند.

نتیجه: کل زمان نمونه برداری و تبدیل در یک ADC برابر است با:

سیکل‌های شروع + سیکل‌های نمونه‌برداری و نگهداری + ۱۰ سیکل تبدیل (کلاک ADC)

startUP Time — Sample & Hold Time

③ رجیستر ADC_CDR0-7 (channel Data)

نتایج تبدیل هر کانال بطور مجزا درون این رجیستر ریخته می‌شود.

```
x = AT91C_BASE_ADC->ADC_CDR2 & 0x3FF;
```

(۱۰ بیتی) — جهت آنکه مابقی بیتها صفر شود.

④ رجیستر ADC_LCDR → Last converted Data

این رجیستر برای همه کانالها مشترک است و آخرین (نتیجه) تبدیل (هر کانالی که باشد) درون این رجیستر نگهداری می‌شود.

⑤ رجیستر فعال ساز کانال ADC : ADC_CHER (channel Enable)

بترتیب با یک کردن هر بیت، ADC متناظر با آن بیت فعال می‌شود. بیتهای CH0 — CH7

```
AT91C_BASE_ADC-> ADC_CHER=(1<<1);
```

کانال 1 فعال شده است.

⑥ رجیستر غیرفعال ساز کانال ADC : ADC_CHDR

با یک کردن هر بیت، ADC متناظر با آن بیت غیرفعال می‌شود. بیتهای CH0 — CH7

⑦ رجیستر وضعیت کانال ADC : ADC_CHSR (status)

با خواندن این رجیستر می‌توان فهمید کدام کانال فعال و کدام غیرفعال است. یک نشانه فعال بودن و صفر نشانه غیرفعال بودن کانال است. بیتهای CH0 — CH7

⑧ رجیستر وضعیت ADC : ADC_SR

31 …	19	18	17	16	15 … 8	7 … 0
...	~~RXBUFF~~	~~ENDRX~~	GOVER	DRDY	$OVER_{0-7}$	$EOC_{n=0,1,...,7}$

بیتهای EOC_n (End of conversion)

اگر یک شوند، یعنی عملیات تبدیل در کانال متناظر تمام شده است. (کامل شده است.)

تذکر: جهت اطمینان از اتمام تبدیل در هر کانال این بیتها را کنترل می‌کنیم مثلاً برای کانال ۲ و ۰ و ۱:
while ((AT91C_BASE_ADC->ADC_SR & 1<<2)==0);
(مقدار کانال ۲ را AND می‌کند تا بقیه صفر شوند.)
(حیثیت بیت ۱ از چپ صفر اضافه می‌شود)

بیت DRDY (Data Ready)

اگر یک شود: در حالت کلی بدون توجه به کانال خاص، اتمام تبدیل ADC را نشان می‌دهد. در هر بار scan کردن کانالهای ورودی، هرگاه آخرین کانال تبدیلش کامل می‌شود این بیت یک می‌شود و با تریگر مجدد این بیت همراه بیتهای EOC صفر می‌شود.

بیت $OVRE_n$ ($OVER_n$) (overrun Error)

اگر قبل از خواندن آخرین تبدیل مربوط به هر کانال، تبدیل دیگری انجام شود، نتایج تبدیل قبلی از دست می‌رود و بیت متناظر (OVER) یک می‌شود که نشانه این خطا می‌باشد. (خاص یک کانال) (هر بیت)

با Reset ADC یا پرش به تابع وقفه $OVER_n$ این بیت صفر می‌شود.

بیت GOVRE (General)

یک شدن این بیت در حالت کلی (بی‌ملاحظه کانالها) در هر کانالی خطای سرریز (OVRE) رخ دهد. (کلی است.)

با Reset ADC یا پرش به تابع وقفه GOVRE این بیت صفر می‌شود.

نکته: واحد کنترل ADC پس از تحریک از طریق هر یک از منابع (نرم افزاری یا سخت افزاری) ADC را فعال کرده و یکبار برای کانالهای فعال، تبدیل را انجام می‌دهد. (عملکرد تبدیل متوالی ADC)

و برای تبدیل مجدد لازم است، تحریک دوباره اتفاق افتد.

وقفه کانالهای ADC:

با یک شدن هر یک از بیتهای مذکور در رجیستر ADC_SR در صورتی که بیت متناظر در رجیستر ADC_IMR نیز یک باشد، وقفه ADC فعال می‌شود.

که توسط دو رجیستر ADC_IER و ADC_IDR صفر و یک می‌شود.

این قاعده کلی است

۷۰

به نام خدا

ARM

مبدل دیجیتال به آنالوگ:

مبدل دیجیتال به آنالوگ ADC در خانواده AT91SAM7S بصورت ۸ بیتی و ۱۰ بیتی می باشد

این واحد دارای ۸ ورودی آنالوگ می باشد که پینهای AD_0 ، AD_1 ، AD_2 و AD_3 بر روی پین های ورودی خروجی موازی (PA) و برای AD_4 ، AD_5 ، AD_6 و AD_7 پایه جداگانه در نظر گرفته شده است.

نکته: پایه ADVREF مربوط به ولتاژ مرجع می باشد که حداکثر 3,3v می باشد.

$\frac{Vref}{2^n-1}$ = سایز پله

تریگر:

برای شروع تبدیل از یک تریگر نرم افزاری یا سخت افزاری که توسط تریگر تایمر کانتر (سخت افزاری ①) و یا پایه ADTRG خارجی (پین ورودی تریگر) (سخت افزاری ②) استفاده کرد.

~~برای تریگر نرم افزاری بیت START در رجیستر ADC_CR را یک می کنیم~~

و برای تریگر سخت افزاری از پایه های TIOA کانال های تایمر کانتر در حالت waveform یا پایه ADTRG خارجی استفاده می کنیم و برای (فعال کردن) تریگر سخت افزاری ابتدا باید بیت TRGEN در رجیستر ADC_MR را یک نماییم و توسط بیتهای TRGSEL از همین رجیستر مرجع تریگر سخت افزاری را مشخص می کنیم. (کد ای اسکی در بالا مشخص فوق است.)

تذکر: در تریگر سخت افزاری تبدیل در لبه بالارونده سیگنال انتخاب شروع می شود.

نکته: برای انجام تبدیل ها کانال های فعال، تنها یک دستور شروع کافی می باشد (بیشتر در نرم افزار مطرح است).

نکته: بیت TRGEN اگر یک شود تحت تریگر سخت افزاری فعال است و اگر صفر باشد تریگر نرم افزاری فعال است.

71

(از رجیستر ADC_MR بیت صفرم)

معرفی رجیسترهای مرتبط با ADC:

۱- رجیسترهای ADC_CHER (Channel - Enable) و ADC_CHDR (Disable)

جهت فعال و غیرفعال کردن کانالهای ADC که هر بیت متناظر با یک کانال

۲- رجیستر ADC_MR (mode) برای انجام تنظیمات لازم برای ADC

الف) مثلاً بیت LOWRES ← صفر باشد، ADC ده بیتی
← یک باشد، ADC هشت بیتی

ب) بیت‌های PRESCAL برای تنظیم کلاک ADC بکمک رابطه زیر:

ADC clock = MCK / ((PRESCAL+1)*2)   تعداد نمونه‌هایی که در یک ثانیه برمی‌دارد

ج) بیتهای STARTUP مدت زمان راه‌اندازی ADC بعد از تریگر و فعال کردن ADC و یا ریست ADC

STARTUP Time = (STARTUP+1)*8/ADC clock

د) بیتهای SHTIM این بیتها برای تنظیم زمان sample&Hold (زمان نمونه برداری)

sample&Hold = SHTIM/ADC clock

۳- رجیستر ADC_CHSR
رجیستر وضعیت کانال ADC که نشان دهندهٔ فعال یا غیرفعال بودن کانالهای ADC را معین می‌کند.
صفر نشانه غیرفعال بودن کانال مربوطه متناظر با بیت؛
یک " " فعال " " " " ؛

۴- رجیستر ADC_SR رجیستر وضعیت ADC

وضعیت ADC را در کلیه شرایط نشان می‌دهد. ۸ بیت اول EOC_n (n=0-7) اگر یک باشند یعنی کانال متناظر فعال بوده و عملیات تبدیل تمام شده است.

۵- رجیستر ADC_LCDR (Last Converted Data)

بیت‌های LDATA آخرین داده تبدیل شده دیجیتال را می‌توان از این بیتها و نتیجتاً رجیستر فوق خواند.
← هر کانالی که آخر تبدیل شده است

۶- رجیستر ADC_CDR_n (Converted Data) n=0,7

می‌توان داده تبدیل شده دیجیتال کانال ADC متناظر را به شرط فعال بودن کانال از بیتهای این رجیستر (بیتهای DATA) خواند
← یک کانال مشخص
(بیت EOC_n متناظر یک باشد از رجیستر ADC_SR)

(۳) ۷۲

Channel Reset

۱۷) رجیستر کنترل ADC_CR

Soft ware Reset

جهت کنترل ADC

بیتادوه الف) بیت SWRST با یک نوشتن در این بیت واحد ADC ریست می‌شود.

بیتدوم ب) بیت START با نوشتن یک در این بیت تبدیل آنالوگ به دیجیتال شروع می‌شود و اگر صفر شود ADC خاموش است (البته این شروع و خاموش شدن برای حالت نرم افزاری می‌باشد) یعنی در حالت سخت افزاری کاری با بیت START نداریم.

خطای OVERRUN

اگر رجیستر ADC_CDR را قبل از تبدیل جدید نخوانیم بیت OVER (۸ بیت، 0-7) متناظر کانال در رجیستر ADC_SR یک می‌شود و بطور مشابه

اگر " " ADC_LCDR را هم " " " " " GOVER (یک بیت، General) در رجیستر ADC-SR یک می‌شود و خطای OVERRUN ایجاد می‌شود.

تمرین

رجیستر ماسک وقفه ADC ADC_IMR / رجیستر فعال کننده وقفه ADC_IER

رجیستر غیرفعال کننده ADC ADC_IDR و سایر رجیسترهای مرتبط با ADC را معرفی و بیتهای آنها را توضیح دهید.

مثال) برنامه ای بنویسید که ADC را بصورت زیر تنظیم نماید و ولتاژ کانال یک (PA18) را پس از تبدیل بصورت ۸ بیتی روی پین های PA0 تا PA7 نشان دهد. فرکانس کلاک اصلی 48 MHz درنظر بگیریم

۷۳

ADC clock = MCK/2 → بیت های PRESCAL صفر

Startup Time = 8/ADC clock → بیت های STARTUP صفر

Sample & Hold Time = 1/ADC clock → بیت های SHTIM یک

```
#include <Atmel/ioat91sam7s64.h>
void main()
{
    start_up
    speed_up
  AT91C_BASE_PMC->PMC_PCER = (1<< AT91C_ID_PIOA);   // کلاک ورودی خروجی ها فعال (→ 2)
  AT91C_BASE_PMC->PMC_PCER = (1<< AT91C_ID_ADC);    // کلاک ADC فعال (→ 4)

AT91C_BASE_PIOA->PIO_PER = (1<<0| 1<<1| --- |1<<7);
AT91C_BASE_PIOA->PIO_OER = (        "        );
AT91C_BASE_PIOA->PIO_OWER = (       "        );
AT91C_BASE_PIOA->PIO_ODSR = 0x0;

AT91C_BASE_ADC->ADC_MR = 0x100010;   // با توجه به تنظیمات داده های بالا و تنظیم ۸ بیتی ADC فعال
                                     // PRESCAL = 0, STARTUP = 0, SHTIM = 1
AT91C_BASE_ADC->ADC_CHER = (1<<1);   // کانال ۱ ADC فعال

while(1)
  {
   AT91C_BASE_ADC->ADC_CR = 0x2;     // تبدیل شروع شود.

   while ((AT91C_BASE_ADC->ADC_SR & (1<<1)) == 0);
   // بیت ۱ ام ADC_SR ، اگر صفر است (تبدیل تمام نشده) همینجا بماند
   // می توانست با 2 (۰۰۰۰۰۱۰) And شود. (در واقع عبارت داخل پرانتز 32 بیتی می شود.)
   // ۱ یعنی کانال ۱
   AT91C_BASE_PIOA->PIO_ODSR = AT91C_BASE_ADC->ADC_CDR1 & 0xFF;
   // ۸ بیت اول بماند و بقیه صفر شود.
  }
}
```

مثال ۲: برنامه‌ای برای ADC ۱۰ بیتی بنویسید که ورودی کانال ۵ را پس از تبدیل روی PA_0 تا PA_9 ببرد.

ADC clock = MCK/2

start up Time = 8 / ADC clock

sample & Hold = 1 / ADC clock

۷۴

```
#include<Atmel/ioat91sam7s64.h>
void main()
{
    start_up
    speed_up
```

فعال سازی کلاک PIOA و ADC مانند مثال قبل

تنظیم پینها مانند مثال قبل PA_0 تا PA_9 بصورت خروجی

```
AT91C_BASE_PIOA->PIO_ODSR=0;
AT91C_BASE_ADC->ADC_MR=0x01000000;   // تنظیمات بالا و ۱۰ بیتی ADC
AT91C_BASE_ADC->ADC_CHER=(1<<5);     // کانال پنجم فعال
while(1)
{
  AT91C_BASE_ADC->ADC_CR=0x2;        // تبدیل شروع شود
  while((AT91C_BASE_ADC->ADC_SR & (1<<5))==0);
  AT91C_BASE_PIOA->PIO_ODSR = AT91C_BASE_ADC->ADC_CDR5 & 0x3FF;
}
```

۱۰ بیتی (۱۰ بیت اول مهم نیست صفر شود)

۷۵

تمرین ۳

برنامه‌ای بنویسید که ADC را بصورت زیر تنظیم اولیه نموده و ولتاژ کانال چهارم (ADC4) را بکمک تایمر/کانتر دو هر ۵۰۰ms بصورت ۸ بیتی تبدیل نموده و نتیجه را در بیت‌های PA0 تا PA7 نمایش دهد. (کلاک ۴۸MHZ)

ADC Clock = MCK/2

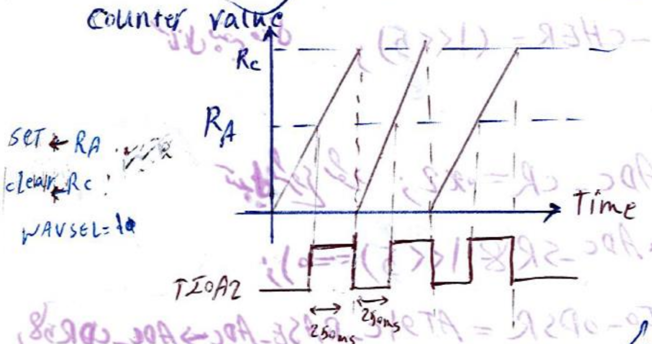
Start UP Time = 8/ADC clock

Sample & Hold Time = 1/ADC clock

WAVSEL=10

لازم است T/C2 هر ۵۰۰ms بتواند ADC را فعال کند، یعنی T/C در مُد waveform تنظیم شده و هر ۵۰۰ms لبه بالارونده بر روی TIOA2 تولید شود.

با مقایسه
RC ← clear
RA ← set



با فرض Timer clock 5

$$\text{فرکانس تایمر} = \frac{MCK}{1024} = \frac{48M}{1024} = 46928\ HZ$$

$$\text{مدت زمان یک کلاک} = \frac{1}{f_T} = \frac{1}{46928} = 21,3\ \mu s$$

هر یک پالس مدت پریود تایمر/کانتر تا RC طول می‌کشد.

$$R_C = \frac{500\ ms}{21,3\ \mu s} = 23474$$

$$R_A \text{ مقدار جبری} = R_C \times (1 - \text{TIOA2 دیوتی سایکل } 0/5) = 11737$$

برنامه [illegible]

۷۴

```
#include <Atmel/ioat91sam7s64.h>
void main()
{
      start_up
      speed_up
AT91C_BASE_PMC->PMC_PCER=(1<<AT91C_ID_PIOA);
      〃              〃      =(1<<AT91C_ID_ADC);
      〃              〃      =(1<<AT91C_ID_TC2);

AT91C_BASE_PIOA->PIO_PER=(1<<0|1<<1|---1<<7);

      〃        ->PIO_OER=      〃    ;
      〃        ->PIO_OWER=     〃    ;
      〃        ->PIO_ODSR=0;
AT91C_BASE_ADC->ADC_MR=0x1000015;
      〃        ->ADC_CHER=(1<<4);
AT91C_BASE_TC2->TC_CMR=0x9C004;
AT91C_BASE_TC2->TC_RA=11737;
AT91C_BASE_TC2->TC_RC=23474;
AT91C_BASE_TC2->TC_CCR=0x5;
while(1)
  {
   while((AT91C_BASE_ADC->ADC_SR&1<<4)==0);
   AT91C_BASE_PIOA->PIO_ODSR=AT91C_BASE_ADC->
                 ADC_CDR4&0xFF;
  }
}
```

- ADC_MR: رجیستر سخت افزار TIOA2 T/C2 فعال و تنظیمات کلاک مطابق بالا
- ADC_CHER: کانال چهارم فعال
- TC_CMR: ACPC 10, ACPA 01, WAVE 1, WAVSEL 110, TCCLKS 100 — تایمر کانتر 2 در حالت waveform و TCCLKS فعال
- TC_RA: set و clear به ترتیب با مقایسه RC و RA
- TC_CCR: تایمر کانتر 2 روشن
- ADC_SR: EOC4
- 0xFF: ۸ بیتی