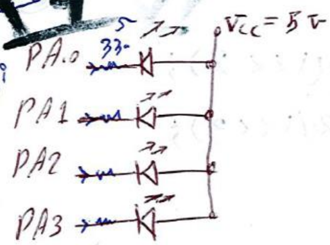
F4



پینها خروجی شده اند

چند مثال ARM :

(1) با توجه به شکل زیر برنامه‌ای بنویسید که ۴ LED مطابق شکل بصورت چرخشی و با یک تاخیر مناسب روشن شوند

کریستال ۱۸.۴۳۲ MHz فرکانس کلاک اصلی ۴۸ MHz باشد

```
#include <atmel/ioat91sam7s64.h>
void delay(void)
{
  unsigned int i,j;
  for(i=0;i<1000000;i++)
   for(j=0;j<1000000;j++);
}

void main (void)
{
     start_up
     speed_up
AT91C_BASE_PMC -> PMC_PCER=(1<<AT91C_ID_PIOA);     // کلاک ورودی و خروجی فعال
AT91C_BASE_PIOA -> PIO_PER=(1<<0|1<<1|1<<2|1<<3);   // چهار پایه بعنوان ورودی و خروجی
AT91C_BASE_PIOA -> PIO_OER=(1<<0|1<<1|1<<2|1<<3);   // چهار پایه بعنوان خروجی
AT91C_BASE_PIOA -> PIO_SODR(1<<0|1<<1|1<<2|1<<3);   // چهار پایه یک شوند همه LED ها خاموش
                                                    // همه خاموش
while(1)
{
  delay();
  AT91C_BASE_PIOA -> PIO_SODR=(1<<3);
  AT91C_BASE_PIOA -> PIO_CODR=(1<<0);
  delay();
  AT91C_BASE_PIOA -> PIO_SODR=(1<<0);
  AT91C_BASE_PIOA -> PIO_CODR=(1<<1);
  delay()
  AT91C_BASE_PIOA -> PIO_SODR=(1<<1);
  AT91C_BASE_PIOA -> PIO_CODR=(1<<2);
```

```
delay();
AT91C_BASE_PIOA -> PIO_SODR=(1<<2);
AT91C_BASE_PIOA -> PIO_CODR=(1<<3);
delay();
{
}
```

معرفی دو رجیستر مهم: برای مقدار دادن در خروجی

PIO_OWER (1)

PIO - output write Enable

با نوشتن یک در هرکدام از بیت‌ها می‌توان مقدار پین متناظر خروجی را طوری تعیین کرد که هر مقداری را در رجیستر PIO_ODSR بنویسیم در آن پین‌ها نوشته شود.

PIO_ODSR (2)

PIO_output Data status Register

در صورت مقابل اگر بیت صفر شود، پین متناظر خروجی صفر می‌شود و اگر یک شود، پین متناظر خروجی یک می‌شود.

رجیستر بالا

با اینکار هر ۳۲ پین یا تعدادی از آن‌ها را یکباره مقدار داد.

تذکر: می‌توا

مثال: مثال قبل را طوری طراحی کنید که با کمک مقادیر داده شده به رجیستر PIO_ODSR همان روند انجام شود:

```
#include <atmel/loat91sam7s64.h>
unsigned char K=1;
void delay(void)
{
  delay بزن
}
void main()
{
    start_up
    speed_up
  AT91C_BASE_PMC -> PMC_PCER= (1<<AT91C_ID_PIOA);
  AT91C_BASE_PIOA -> PIO_PER=(1<<0|1<<1|1<<2|1<<3);
  AT91C_BASE_PIOA -> PIO_OER=(1<<0|1<<1|1<<2|1<<3);
  AT91C_BASE_PIOA -> PIO_OWER=(1<<0|1<<1|1<<2|1<<3);
AT91C_BASE_PIOA-> PIO_ODSR=0xF;
  while(1)
    {
    delay();
    AT91C_BASE_PIOA -> PIOA_ODSR = ~K;
    K=K<<1;
    if(K==16)
     K=1;
    }
}
```

```
AT91C_BASE_PIOA
  -> PIO_ODSR=6;
delay();
N = 5;
delay();
N = 7
delay();
```

نحوهٔ خواندن یک مقدار:

AT91C_BASE_PIOA->PIO_PER=(1<<0|1<<1|1<<2|1<<3);

بعد از آنکه پین بعنوان ورودی معرفی شد

AT91C_BASE_PIOA->PIO_ODR=(//);

AT91C_BASE_PIOA->PIO_IFER=(//);

بعنوان مثال A_0, A_1, A_2, A_3

از رجیستر PIO_PDSR استفاده می‌کنیم مثال:

K= AT91C_BASE_PIOA->PIO_PDSR;

هرچه در ورودیهای $A_0 - A_3$ باشد یکجا بعنوان یک مقدار در K می‌ریزد.

تذکر: بجهت غیرفعال کردن بقیه پینها بهتر است حاصل با یک مقدار AND شود تا وضعیت بقیه پینها صفر گردد.

ورودی انتخاب

فرض کنید پین A_4 و A_5 ورودی باشد

K=(AT91C_BASE_PIOA->PIO_PDSR & 0xB0);

0 0 1 1 0 0 0 0

تذکر: مقدار فوق ~~را می‌توان هر کاری انجام داد بعنوان مثال~~ شیفت به راست بدهیم

باید مقدار را به ابتدای بیاوریم یعنی یک مقدار مجزا لازم است

K=(AT91C_BASE_PIOA->PIO_PDSR & 0xB0) >> 4);

شیفت به راست

عدد حاصل یکی از مقادیر روبروی می‌باشد

~~فرقی ندارد با K' = K >> 2~~

S_1	S_0	
0	0	→ 0
0	1	→ 1
1	0	→ 2
1	1	→ 3

تمرین: برنامه‌ای بنویسید که از 8 پین اول میکرو عدد وارد شود، اگر این عدد مضرب 2 است است یک LED روی پین نهم روشن شود، در غیر اینصورت روشن نگردد.

تمرین: پینهای $A_{10} - A_{18}$ را بعنوان یک پورت ۸ بیتی مجزا برای ورودی تنظیم کنید مقدار ورودی را بخوانید اگر حاصل مضربی از 2 بود، LED روی پین A_0 روشن شود در غیر اینصورت LED روی پین A_1 روشن شود

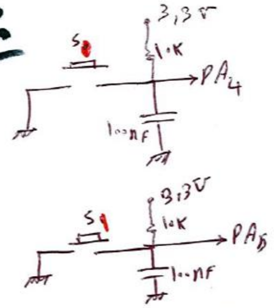
تمرین: برنامه ساعت را بنویسید بطوریکه از ۴ پین اول بعنوا خروجی استفاده شود.

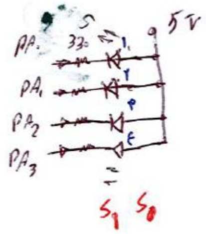
تذکر: بکمک دستور #define می‌توان از نوشتن مکرر و طولانی عبارات جلوگیری کرد

مثال) #define LED ON AT91C_BASE_PIOA->PIO_CODR=(1<<0);

در برنامه بجای عبارت دوم همیشه می‌نویسیم LED ON (پین صفرم، صفر شود).

۳۰





مثال) برنامه ای بنویسید مطابق شکل زیر LED ها چرخشی روشن شوند و کلید S1 باعث افزایش delay و S0 باعث کاهش delay گردد.

```
#include <Atmel/ioat91sam7s64.h>
# define   LED1ON    AT91C_BASE_PIOA->PIO_CODR=(1<<0);
# define   LED1off   "   "   "   "   _SODR=  "  ;
# define   LED2ON    "   "   "   "  _CODR=(1<<1);
"  "       LED2off   "   "   "   "   " SODR=(1<<1);
"  "       LED3ON    "   "   "   "   CODR=(1<<2);
"  "       LED3off   "   "   "   "   SODR=(1<<2);
"  "       LED4ON    "   "   "   "   CODR=(1<<3);
"  "       LED4off   "   "   "   "   SODR=(1<<3);

unsigned int L=10; i,j,k,n;
void delay (void)
{
K=(AT91C_BASE_PIOA->PIO_PDSR & 0x30) >> 4;
if(K==2)
  L=L+10;
if((K==1)&&(L!=0))
  L=L-10;
for(n=0; n<L; n++)
for(i=0; i<10000; i++)
for(j=0; j<10000; j++);
}
```

شیفت به راست

خواندن پین 4 و 5

کلید S1 فشار داده شده

کلید S0 فشار داده شده

اگر شیفت به راست ندهیم
if(K==0x20) می نوشتیم

یا بجای این می نوشتیم
if(K==0x10 && L!=0)

⑥

```
void main()
{
        Start_up
        Speed_up
```

PIOA تنظیم AT91C_BASE_PMC -> Pmc_PCER=(1<<AT91C_ID_PIOA);

تنظیم AT91C_BASE_PIOA -> PIO_PER = (1<<0|1<<1|1<<2|1<<3|1<<4|1<<5);

تنظیم AT91C_BASE_PIOA -> PIO_OER=(1<<0|1<<1|1<<2|1<<3);

تنظیم AT91C_BASE_PIOA -> PIO_ODR=(1<<4|1<<5);

تنظیم AT91C_BASE_PIOA -> PIO_IFER (1<<4|1<<5);

```
LED1OFF;
LED2off;
LED3off;
LED4off;
 while(1)
    {
    delay();
    LED4off;
    LED1oN;
    delay();
    LED1off;
    LED2oN;
    delay();
    LED2off
    LED3oN;
    delay();
    LED3OFF;
    LED4oN;
    }
}
```

اول چیستم پایینی را توضیح دهیم بعد مثال را

مثال) برنامه‌ای بنویسید که کلاک قابل برنامه‌ریزی خروجی ۲ را بر روی پین ۳۲ PA31 با فرکانس 12MHz فعال کند. تذکر: فرکانس کلاک اصلی 48MHz باشد.

(با PCK2 روی پین B در حالت دوم، مجهول است)

```
#include <atmel/ioat91sam7s64.h>

void main()
{
    start_up
    speed_up
    AT91C_BASE_PMC->PMC_PCKR[2]=0x0B;
    AT91C_BASE_PMC->PMC_SCER=0x401;

    AT91C_BASE_PIOA->PIO_PDR=(1<<31);
    AT91C_BASE_PIOA->PIO_MDER=(1<<31);
    AT91C_BASE_PIOA->PIO_BSR=(1<<31);

    while(1);
}
```

speed_up ← بجای فقط آخر که فقط کلاک CPU فعال می‌شود می‌نویسیم:

PMC_PCKR[2]=0x0B ← کلاک قابل برنامه‌ریزی همان کلاک خروجی PLL تقسیم بر ۴ (از صفحه پایینی برگه)

PMC_SCER=0x401 ← هم کلاک CPU فعال هم کلاک قابل برنامه‌ریزی ۲

PIO_PDR ← استفاده از پین برای دستگاه‌های جانبی

PIO_MDER ← چند منظوره

PIO_BSR ← حالت B، PCK2 برای پین 31 میکرو می‌باشد.

رجیستر کلاک قابل برنامه‌ریزی PMC

Programmable clock Register

PMC_PCKR[n]

↑ از ۰ تا ۲ ← ۳ کلاک قابل برنامه‌ریزی داریم

صحبت انجام تنظیمات لازم برای کلاک قابل برنامه‌ریزی

۲ بیت اول CSS (select) ← انتخاب منبع کلاک: clock source select

CSS	انتخاب منبع کلاک
0 0	انتخاب کلاک آهسته
0 1	انتخاب خروجی کلاک اسیلاتور اصلی
1 0	
1 1	انتخاب خروجی کلاک PLL

prescaller

۳ بیت بعدی PRES

PRES ← تقسیم از کلاک انتخابی بالا به خروجی کلاک قابل برنامه‌ریزی

PRES	
0 0 0	تقسیم بر ۱
0 0 1	تقسیم بر ۲
0 1 0	۴
0 1 1	۸
1 0 0	۱۶
1 0 1	۳۲
1 1 0	۶۴
1 1 1	

AT91C_BASE_PMC->PMC_PCKR[2]=0x0B; برنامه‌ریزی

PMC_PCKR[2] = 0 010 11 ← انتخاب خروجی PLL

$\frac{48M}{4} = 12M$ ← تقسیم بر ۴