در این مقاله به بررسی نحوه ی راه اندازی یک نمونه LCD گرافیکی رنگی که با نام LCD گوشی چینی یا LCD گوشی N96 یا LCD مدل ILI9325 در بازار شناخته می شود و نحوه ی راه اندازی واحد تایمر/کانتر میکروکنترلر های سری AT91SAM شرکت اتمل در کامپایلر KEIL پرداخته ایم .

توجه :

ویرایش قبلی این مقاله قبلا در مجله ی PMM (مجله میکروکنترلر فارسی – Persian Microcontroller Magazine) شماره 3 و5 منتشر بود . ویرایشی که همکنون مشاهده میکنید برای کتاب " مرجع کامل میکروکنترلرهای سری AT91SAM شرکت ATMEL" تهیه شده است .

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

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

جهت ارتباط با نویسنده به آدرس زیر مراجعه کنید :

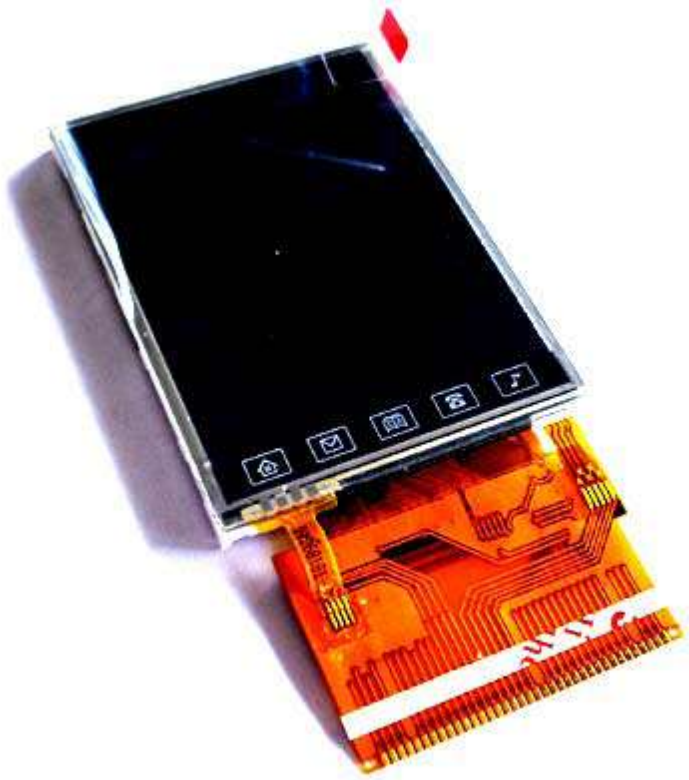
http://www.iranmicro.ir/forum/member.php?u=3640

قبل از مطالعه مطالب این بخش شما باید بخش های 1 و 2 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)

راه اندازی LCDN96 (معروف به N96 چینی)

در بازار LCD های مختلفی وجود دارد که میتوانند پیغام ها را به صورت خطی ، تک رنگ ، گرافیکی دو رنگ یا تمام رنگی به کاربر نمایش دهند .LCD های گرافیکی یکی از قطعات پرکاربرد در دستگاه های برقی/الکترونیک است که جهت ایجاد واسط کاربری مورد استفاده قرار می گیرد . در این بخش قصد داریم ، یکی از این نمایشگر ها که با نام LCD گوشی چینی یا LCD گوشی N96 یا LCD مدل ILI9325 در بازار شناخته می شود را بررسی کرده و بعد از تعریف و توضیح کاربرد پایه ، کتابخانه ی مورد نیاز برای راه اندازی آن با میکروکنترلر های سریAT91SAM اتمل را تشریح کنیم . این LCD در زمان تالیف این مقاله در بازار تهران با قیمت تقریبی 14000/12000 تومان به فروش می رسد.



از مهمترین خصوصیات این LCD در زیر به آن اشاره شده است :

- ✓ ابعاد حدود 3 اینچ
- ✓ تعداد پیکسل 240 در 320
- ✓ قابلیت تفکیک 262 هزار رنگ

✓ دارای دو مد دیتا باس 8 و 16 بیتی

✓ توان مصرفی بسیار پایین (بدون بک لایت) و کار کردن در رنج ولتاژ 3.3 ولت

✓ دارای یک تاچ اسکرین تعبیه شده روی ماژول

اولین مرحله ای که برای شروع به کار آن نیاز دارید ، همانطور که گفته شد ، شناسایی این LCD و انتخاب صحیح آن می باشد . بر اساس آماری که گرفته شده حدود 200 الی 300 نوع LCD گوشی های چینی در بازار ایران وجود دارد که فقط 5 تا 10 مورد آن ها ، قابل راه اندازی (تا الان) است .

اولین نکته اینکه به هیچ وجه به PCB منعطف ماژول و نحوه ی طراحی آن و سیم کشی های روی این LCD توجه نکنید . ماژول های مشابه دارای طراحی های مشابه نمی باشند . تنها و تنها موردی که باید برای پیدا کردن این LCD باید توجه کرد ، اندازه ی آن می باشد . در جدول زیر این اندازه و تعدادی از مشخصات آن را مشاهده می کنید .

Parameter	Value	Unit
LCD Mode	a-Si TFT/transmissive	-
Color	262K	-
Display Resolution	240*RGB*320	pixels
OUTLINE DIMEMSIONS	50.00(W) x69.20(H) x4.05(T)	mm
Active Area(A.A)	43.20 (W) x 57.60(H)	mm
Pixel Arrangement	RGB-stripe	-
Viewing Direction	12 O'clock	
Display Mode	Normally white	
LCD Controller/Driver	ILI9325	-
IC Package Type	COG	-
MPU interface	Standard 8080 system18-/16-bit paraller	-
Power Supply Voltage	2.5~3.3	V
Back-light	White LED*4	pcs

فروشنده های این ماژول ها هم این LCD را با نام LCD گوشی N96 چینی می شناسند . اما فاکتور اصلی انتخاب نمی باشد . بعد از در دست گرفتن ماژول مشخصات زیر باید از روی PCB منعطف مشهود باشد .

1 – تعداد پایه ها حتما 37 تا باشد .

2 – پایه های 12 تا 15 به Touch Panel ال سی دی رفته باشد .

3 – پایه های 16 تا 20 به قسمت بک لایت ماژول رفته باشد .

4 – پایه ی 21 ، NC (Not Connected) باشد .

5 – پایه های 5 و 34 به خطوط زخیم تر از پایه های دیگر متصل شده باشد .

این چند موارد تنها ویژگی ظاهری این ماژول می باشد که ملاک اصلی انتخاب شما برای خرید این LCD است . بعد از خرید این ماژول نوبت به تبدیل پایه های خروجی این ماژول به صورتیکه بتوان روی برد بورد از آن استفاده کرد ، است . نکته ی اول اینکه سوکتی برای این ماژول وجود ندارد . نکته ی دوم هم اینکه بهترین و ارزانترین راه لحیم پایه های این ماژول به سیم های حتی الامکان نازک است .

در مورد کنترلر این LCD هم باید گفت که معروفترین آن ها ILI9325 یا ILI9320 است که برگه ی اطلاعاتی این کنترلر در اینترنت به کثرت وجود دارد . کنترلر های دیگه ای هم وجود دارد که با دقت کردن در آن ها قابل مشاهده است که اینترفیس و کدهای دستورالعمل مشابهی نسبت به دو کنترلر ذکر شده دارند . در این مقاله تنها دو کنترلر گفته شده مورد بررسی قرار خواهند گرفت .

خود ماژول هم یک دیتاشیت مخصوص به خود دارد که به نام ELT240320 مشخص شده است که اطلاعاتی در مورد نام پایه ها ، ابعاد ماژول و .... در آن وجود دارد . در اینجا نیز در مورد پایه های این LCD بیشتر توضیح خواهیم داد . شکل صفحه ی بعد شماره و نام پایه ها را نشان می دهد .

No.	Symbol	Functional	Remark
1	DB1	Data bus	
2	DB2	Data bus	
3	DB3	Data bus	
4	DB4	Data bus	
5	GND	Ground	
6	VCC	Power	
7	CS	Chip select pin of serial inter face	
8	RS	Data or command	
9	WR	Write signal	
10	RD	Read signal	
11	IMO	Interface mode select	
12	X+	Touch panel X+	
13	Y+	Touch panel Y+	
14	X-	Touch panel X-	
15	Y-	Touch panel Y-	
16	LED-A	LED A	
17	LED-K4	LED K4	
18	LED-K3	LED K3	
19	LED-K2	LED K2	
20	LED-K1	LED K1	
21	NC	No connection	
22	DB5	Data bus	
23	DB10	Data bus	
24	DB11	Data bus	
25	DB12	Data bus	
26	DB13	Data bus	
27	DB14	Data bus	
28	DB15	Data bus	
29	DB16	Data bus	
30	DB17	Data bus	
31	REST	Reset dim	
32	VCC	Power	
33	VCC	Power	
34	GND	Ground	
35	DB6	Data bus	
36	DB7	Data bus	
37	DB8	Data bus	

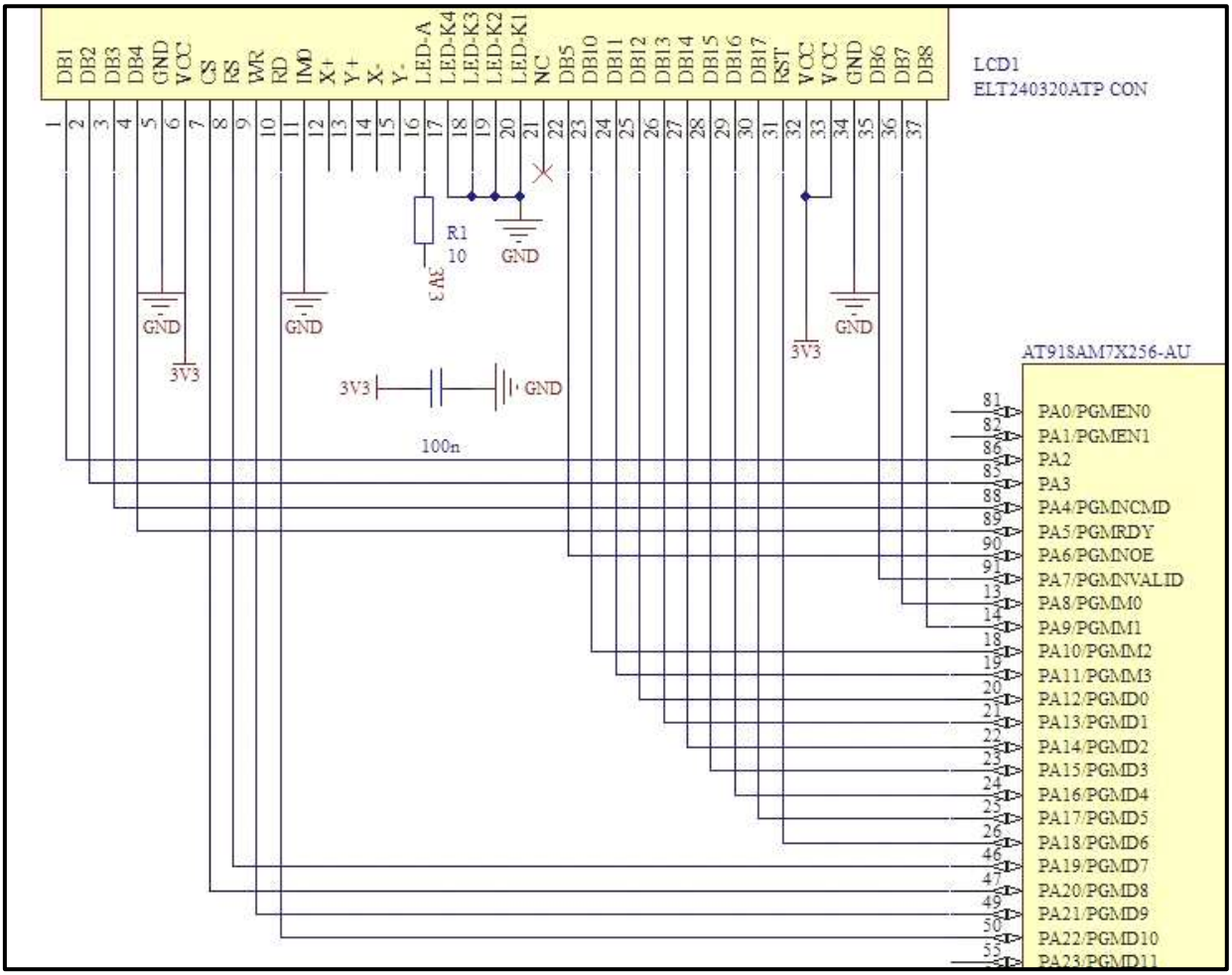
- پایه GND و VCC : این دو پایه که مجموعا 5 پایه از پایه های ماژول را شامل می شوند ، وظیفه ی تغذیه ی LCD را به عهده دارند که در برگ اطلاعاتی ماژول به ولتاژ حدود 2.5 الی 3.3 ولت اشاره شده است .
- پایه های DB1-DB8 و DB10-DB17 : این 16 پایه دیتا باس (گذرگاه داده) بین کنترلر LCD و میکروکنترلر می باشند . داده ها و کد دستورات از طریق این خطوط به LCD منتقل یا دریافت می شوند .

- پایه RESET : این پایه در صورتیکه صفر شود کنترلر LCD را بازنشانی می کند .
- پایه RS : این پایه وظیفه ی انتخاب رجیستر دستورالعمل و یا رجیستر داده را دارد .
- پایه CS : پایه ی انتخاب تراشه (Chip Select)
- پایه WR : در صورتیکه این پایه صفر شود ، قصد نوشتن داده یا دستور در رجیستر های دستورالعمل یا داده را داریم.
- پایه RD : در صورتیکه این پایه صفر شود ، قصد خواندن داده از روی LCD را داریم .
- IM0 : این پایه وظیفه ی انتخاب مد دیتاباس را دارد . در صورتی که یک شود مد 8 بیت و در غیر اینصورت مد 16 بیت انتخاب خواهد شد .
- پایه های X+ ، X- ، Y+ و Y- : این چهار پایه خطوط متصل به صفحه ی لمسی LCD می باشند .
- پایه های LED-A ، LED-K1 ، LED-K2 ، LED-K3 ، LED-K4 : پایه LED-A پایه ی آنود چهار دیود نورانی پشت صفحه (Back Light) و چهار پایه ی LED-K1 تا LED-K4 ، پایه های کاتد هر کدام از دیود های نورانی پشت صفحه هستند .
- پایه NC : پایه ی بدون اتصال

بعد از توضیحات تقریبا اجمالی در مورد ماژول LCD سراغ بررسی توابع نوشته شده در فایل سرآمد tftlcd_functions.h که جهت راه اندازی LCD مورد استفاده قرار می گیرد ، می پردازیم . نحوه ی اجرا و کارکرد توابع به عهده ی خواننده گذاشته می شود و در اینجا تنها به توضیح توابع و آرگومان های ارسالی به LCD پرداخته می شود .اولین مرحله تعیین و تعریف نحوه ی اتصال ماژول به میکروکنترلر می باشد . همانطور گفته شد ، ماژول LCD دارای 16 پایه ی دیتا باس و 5 پایه ی کنترل می باشد . برای تعریف نحوه ی اتصال از تعاریف پیش پردازنده های #define زیر کمک می گیریم .

```
/* LCD Pin Configuration */
#define TFTLCD_DATAPORT_x
#define TFTLCD_DATAPORT_OFFSET
#define TFTLCD_CONTROLPORT_x
#define TFTLCD_RST
#define TFTLCD_RS
#define TFTLCD_CS
#define TFTLCD_WR
#define TFTLCD_RD
```

تکه برنامه ی بالا پورت های اتصالی از میکروکنترلر را به دیتا پورت و کنترل پورت ماژول را مشخص می کند و حتما باید قبل از اتصال فایل سرآمد توابع LCD در برنامه قرار داده شود . به این صورت که خط اول (#define TFTLCD_DATAPORT_x) پورت اتصالی به دیتا باس LCD را تعریف می کند و باید به جای x ، دو گزینه ی A (پورت A میکروکنترلر) یا B (پورت B میکروکنترلر) قرار داده شود . خط دوم (#define TFTLCD_DATAPORT_OFFSET) فاصله ی شروع دیتا باس LCD به میکروکنترلر را نشان می دهد . برای مثال در صورتی که قصد دارید دیتا باس LCD از پایه ی PORTx.5 شروع به اتصال به میکرو کنترلر کند این تعریف را مساوی 5 قرار دهید . خط دوم پورت اتصالی میکروکنترلر به کنترل پورت LCD را مشخص می کند که در این صورت باید به جای x گزینه ی A یا B انتخاب شود . خطوط 4 تا 8 نیز شماره ی پایه ی اتصالی به هر کدام از خطوط کنترلی LCD (مانند RST ، CS و ...) را مشخص می کند . برای مثال ، شکل زیر که یک نمونه نحوه ی اتصال این LCD به میکروکنترلر را مشخص می کند را مشاهده می فرمایید .



لذا تعریف پایه های ارتباطی ماژول LCD و میکروکنترلر در ابتدای برنامه به صورت زیر می باشد .

```
/* LCD Pin Configuration */
#define TFTLCD_DATAPORT_A
#define TFTLCD_DATAPORT_OFFSET      2
#define TFTLCD_CONTROLPORT_A
#define TFTLCD_RST                  18
#define TFTLCD_RS                   19
#define TFTLCD_CS                   20
#define TFTLCD_WR                   21
#define TFTLCD_RD                   22
```

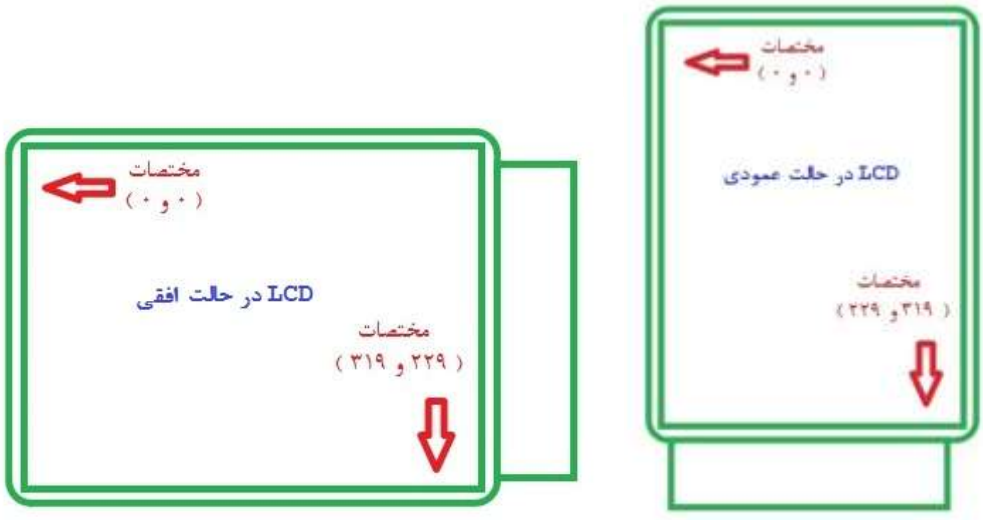
بعد از این مرحله باید نحوه ی قرار گیری و مختصات بند ی صفحه ی LCD را مشخص کنید . در صورتیکه قصد دارید LCD را به طور عمودی استفاده کنید ، تعریف زیر را در برنامه ی خود و قبل از اضافه کردن فایل سرآمد ، قرار دهید .

```
/* Declare LCD Rotation */
#define PORTRAIT
```

در غیر اینصورت دستور زیر را قرار دهید . دستور زیر نحوه ی مختصات بندی صفحه نمایش را به صورت افقی قرار می دهد .

```
/* Declare LCD Rotation */
#define LANDSCAPE
```

شکل زیر مختصات بندی صفحه نمایش در حالت های افقی و عمودی را نشان می دهد .



و در مرحله ی سوم توسط دستور زیر فایل سرآمد توابع راه اندازی LCD را به برنامه ملحق کنید .

```
/* LCD Functions */
#include "tftlcd_functions.h"
```

بعد از تعاریف اولیه ی برنامه باید در ابتدای تابع main و قبل از همه ی توابع دیگر LCD ، ماژول LCD مقدار دهی اولیه بشود . تابع زیر این کار را انجام می دهد و LCD را آماده ی پذیرفتن توابع دیگر می کند .

```
/* LCD Initialization */
tftlcd_init();
```

بعد از اجرای صحیح و کامل مرحله ی آخر شما می توانید توابعی که در زیر به آن ها پرداخته شده است را در برنامه استفاده کنید . به طور کلی توابع موجود به سه دسته تقسیم می شوند . دسته ی اول توابع پایه هستند که جزو ضروری ترین توابع فایل سرآمد می باشد و می توان توسط این سه تابع بقیه ی توابع را بوجود آورد و کار های مختلف پایه ای روی LCD انجام داد . دسته ی دوم توابعی هستند که معمولا برای رسم اشکال هندسی مختلف و اعمال مختلف روی پیکسل ها و صفحه ی نمایش مورد استفاده قرار می گیرند . و دسته ی سوم توابع مربوط به چاپ کاراکتر و ... می باشد . ابتدا به دسته ی اول توابع می پردازیم .

❖ تابع tftlcd_write_index_register() :

این تابع درون رجیستر دستورالعمل LCD ، عملیات نوشتن را انجام می دهد (کد دستورالعمل را به LCD ارسال می کند) و الگوی آن به صورت زیر است که آرگومان ارسالی آن همان کد دستوری است که قرار است به LCD ارسال شود .

```
void kslcd_write_index_register(int command)
```

❖ تابع tftlcd_write_wdr() :

این تابع همانطور که از نام آن پیداست ، درون رجیستر wdr ماژول LCD می نویسد . داده هایی که درون این رجیستر نوشته می شوند ، به طور معمول به حافظه ی RAM صفحه ی نمایش ارسال می شوند . این تابع نیز یک پارامتر ارسالی دارد که همان داده ای است که قرار است در رجیستر wdr نوشته شود .

```
void kslcd_write_wdr(int data)
```

❖ تابع tftlcd_read_rdr() :

این تابع عمل عکس تابع بالا را انجام می دهد . به این صورت که این رجیستر حاوی اطلاعات موجود روی صفحه ی نمایش LCD است که توسط این تابع از LCD خوانده می شود . پارامتر بازگشتی این تابع همان محتویات موجود در ثبات rdr ماژول LCD است .

```
void kslcd_write_wdr(int data)
```

دسته ی دوم توابع که در زیر به آن ها اشاره شده است .

❖ تابع tftlcd_clear() :

این تابع کل صفحه نمایش LCD را پاک کرده و مکان نمای مجازی را به مختصات (0 و 0) می برد . این تابع هیچ مقدار ارسالی و بازگشتی ندارد .

```
void kslcd_clear(void)
```

❖ تابع tftlcd_write_pixel() : الگوی این تابع به صورت زیر است .

```
void kslcd_write_pixel(int x,int y,int color)
```

پارامتر x و y ، مختصات پیکسلی است که قرار است با رنگ color پر شود .

❖ تابع tftlcd_read_pixel() : این تابع رنگ پیکسلی به مختصات x و y را بر می گرداند .

```
int kslcd_read_pixel(int x,int y)
```

❖ تابع tftlcd_draw_line() : این تابع یک خط راست از مختصات شروع x0 و y0 تا مختصات انتهای x1 و y1 با رنگ مشخص شده توسط پارامتر color رسم می کند . فرم کلی این تابع به صورت زیر است .

```
void kslcd_draw_line(int x0,int y0,int x1,int y1,int color)
```

❖ تابع tftlcd_draw_rectangle() : فرم کلی این تابع به صورت زیر است .

```
void kslcd_draw_rectangle(int x0,int y0,int x1,int y1,char fill,int color)
```

این تابع وظیفه ی رسم یک چهارگوشه را دارد که می تواند توسط پارامتر fill ، تو خالی یا تـوپر باشـد . در صـورتیکه fill = 0 باشد ، چهار گوشه توخالی و اگر fill = 1 باشد چهار گوشه توپر خواهد شـد . پـارامتر هـای x0 و y0 ، مختصـات شـروع رسـم چهار گوشه و x1 و y1 ، مختصات انتهای این چهارگوشـه را تعیـین مـی کنـد . پـارامتر color نیـز تعیـین کننـده ی رنـگ ایـن چهارگوشه است .

- تابع tftlcd_draw_circle() : این تابع نیز به مختصات مرکز x و y و شعاع r و با رنگ color ، دایره ای رسم می کند . در صورتیکه آرگومان fill برابر صفر باشد دایره توخالی و اگر fill = 1 باشد دایره توپر خواهد بود .

```
void kslcd_draw_circle(int x0,int y0,int radius,char fill,int color)
```

- تابع tftlcd_write_pic() : این تابع یک عکسی که در حافظه ی flash میکروکنترلر ذخیره شـده اسـت را در مختصـات شروع x و y نمایش می دهد . الگوی این تابع به صورت زیر است و

```
void kslcd_write_pic(int x,int y,const short *pointer)
```

آرگومان سوم این تابع نام ثابتی است که به صورت آرایه در حافظه ی flash میکروکنترلر ذخیره شده است . باید به این نکتـه توجه داشته باشید که آرایه ی اول و دوم این ثابت را به ترتیب به طول و عرض همان عکس اضافه کنید . به صورت زیر

```
const short picture[] =
{
    length,width,
    0xBA62,0x9200,0xA200,0x9A00,0xA220,0xB282,0xC241,0xC240,
    0xA220,0xA220,0x9A20,0xA220,0xB261,0x9220,0x9200,0x9A20,
    0xA220,0x9A20,0xAA40,0x9A20,0xAA60,0xA220,0xAA40,0xA240,
    .
    .
    .
    .
    0x9A20,0x9A20,0x9A00,0xAA40,0xAA20,0xB240,0xAA60,0xAA00,
    0x9A00,0x9A00,0xAA42,0xAA62,0xAA01,0x89E0,0x99E0,0x9A20,
    0xAA21,0xA222,0x9A01,0x9A21,0x89E0,0x89C1,0x81C1,0x79C1
    };
```

که شما باید به جای length و width ، به ترتیب طول و عرض عکس را قرار دهید . دستور زیر نیز این عکس را در مختصات (0 و 0) قرار می دهد .

```
kslcd_write_pic(0,0,picture);
```

و در اینجا هم آخرین دسته از توابع را بررسی خواهیم کرد . قابل توجه اینکه یک فونت 8 در 16 نیز برای کار با توابع چاپ رشته در برنامه تعریف شده است .

- تابع tftlcd_gotoxy() : این تابع مکان نمای مجازی متن را به مختصات x و y می برد . باید توجه داشته باشید که حداکثر مقدار x ، 40 و y برابر 15 در حالت افقی و در حالت عمودی به ترتیب 15 برای x و 40 برای y می باشد . الگوی این تابع به صورت زیر است .

```
void kslcd_gotoxy(int x,int y)
```

- تابع tftlcd_putchar() : فرم کلی این تابع به صورت زیر است .

```
void kslcd_putchar(char character,int forground_color,int background_color,int transparent_mode)
```

آرگومان اول کاراکتری است که در مختصات فعلی مکان نما چاپ می شود . آرگومان دوم رنگ کاراکتر و آرگومان سوم رنگ پس زمینه ی کاراکتر را تعیین می کند . آرگومان چهارم نیز یک امکان جدید را برای کاربر قرار می دهد . در صورتیکه این پارامتر مقدار صفر داشته باشد این ویژگی غیر فعال می شود . اما در صورتیکه مقدار 1 داشته باشد ، رنگ پس زمینه غیر فعال می شود و به جای آن رنگ فعلی پیکسل در نظرگرفته می شود . شکل زیر این ویژگی را بهتر نمایش می دهد

- توابع tftlcd_puts() و tftlcd_putsf() : این دو تابع هر دو وظیفه ی نمایش یک رشته را روی صفحه ی نمایش دارند با این تفاوت که تابع tftlcd_puts() رشته ی ارسالی ذخیره شده در SRAM میکروکنترلر را می پذیرد و تابع tftlcd_putsf() رشته ای که در حافظه ی flash ذخیره شده است را نمایش می دهد . آرگومان های دوم تا چهارم نیز وظیفه ی مشابه تابع توضیح داده شده در قسمت قبل را دارند .

```
void kslcd_putsf(const char *string,int forground_color,int background_color,int transparent_mode)

void kslcd_puts(char *string,int forground_color,int background_color,int transparent_mode)
```

- مثال :برنامه ی زیر هم به صورت نمایشی نحوه ی استفاده از توابع توضیح داده شده را نشان می دهد

- نمونه مثال راه اندازی LcdN96

```
/* AT91SAMT7X256 Register definitions */
#include < AT91SAM7X256.H>
/* Delay Functions */
#define F_CPU 72000000
#include <delay.h>
/* Declare LCD Rotation */
#define PORTRAIT
/* LCD Pin Configuration */
```

```
#define TFTLCD_DATAPORT_A
#define TFTLCD_DATAPORT_OFFSET 0
#define TFTLCD_CONTROLPORT_A
#define TFTLCD_RST 16
#define TFTLCD_RS 17
#define TFTLCD_CS 18
#define TFTLCD_WR 19
#define TFTLCD_RD 20
/* LCD Functions */
#include "picture.h"
#include "tftlcd_functions.c"
int main(void)
{
delay_ms(100);
/* LCD Initialization */
tftlcd_init();
#ifdef LANDSCAPE
tftlcd_write_pic(0,0,picture);
delay_ms(5000);
tftlcd_draw_line(5,5,180,120,RED);
tftlcd_draw_line(10,5,185,120,GREEN);
tftlcd_draw_line(15,5,190,120,BLUE);
tftlcd_draw_rectangle(160,20,250,120,0,YELLOW);
tftlcd_draw_rectangle(165,25,245,115,0,PURPLE);
tftlcd_draw_rectangle(170,30,240,110,1,BLUE);
tftlcd_draw_circle(70,150,50,0,GREEN);
tftlcd_draw_circle(90,150,50,1,PURPLE);
tftlcd_gotoxy(20,12);
tftlcd_putsf( "...:::HELLO:::...",RED,WHITE,0);
tftlcd_gotoxy(2,9);
tftlcd_putsf( "without Transparent",GREEN,WHITE,0);
```

```
tftlcd_gotoxy(2,10);
tftlcd_putsf( "Transparent Mode",BLUE,WHITE,1);
delay_ms(10000);
while(1)
{
tftlcd_draw_rectangle(0,0,319,239,1,RED);
delay_ms(1000);
tftlcd_draw_rectangle(0,0,319,239,1,GREEN);
delay_ms(1000);
tftlcd_draw_rectangle(0,0,319,239,1,BLUE);
delay_ms(1000);
};
#endif

#ifdef PORTRAIT
tftlcd_write_pic(0,0,picture);
delay_ms(50000);
tftlcd_draw_line(5,5,180,120,RED);
tftlcd_draw_line(10,5,185,120,GREEN);
tftlcd_draw_line(15,5,190,120,BLUE);
tftlcd_draw_rectangle(160,20,200,120,0,YELLOW);
tftlcd_draw_rectangle(165,25,205,115,0,PURPLE);
tftlcd_draw_rectangle(170,30,205,110,1,BLUE);
tftlcd_draw_circle(70,150,50,0,GREEN);
tftlcd_draw_circle(90,150,50,1,PURPLE);
tftlcd_gotoxy(3,3);
tftlcd_putsf( "...:::HELLO:::...",RED,WHITE,0);
tftlcd_gotoxy(3,4);
tftlcd_putsf( "without Transparent",GREEN,WHITE,0);
tftlcd_gotoxy(3,6);
tftlcd_putsf( "Transparent Mode",BLUE,WHITE,1);
```

```
delay_ms(10000);
while(1)
{
tftlcd_draw_rectangle(0,0,239,319,1,RED);
delay_ms(1000);
tftlcd_draw_rectangle(0,0,239,319,1,GREEN);
delay_ms(1000);
tftlcd_draw_rectangle(0,0,239,319,1,BLUE);
delay_ms(1000);
};
#endif
}
```

تایمر و کانتر:

تایمر/کانتر ها یکی از بخش های مهم هر میکروکنترلر میباشد ، که توسط آن میتوانید مواردی همچون تولید پالس PWM ، شمارش تعداد پالس خارجی (Counter) ، اندازه گیری و تولید زمان (TIMER) و ... را انجام دهید .

در این بخش به بررسی واحد TC (Timer Counter) در سری AT91SAM پرداخته ایم . این خانواده دارای 1 تا 3 تایمر/کانتر 16 بیتی با عملکرد کاملا مجزا میباشند ، هر کانال می تواند به طور مستقل برنامه ریزی شود تا بتوان به وسیله ی آن کاربردهایی چون فرکانس متر ، شمارش وقایع خارجی ، تولید پالس ، ایجاد تاخیر های زمانی و تولید پالس های PWM ، ساخت .

همچنین هر کانال از این واحد ، دارای سه ورودی کلاک خارجی (XCx) ، پنج ورودی کلاک داخلی و دو ورودی/خروجی که قابل پیکره بندی توسط کاربر است ، می باشد. هر کانال تعدادی سیگنال وقفه را کنترل می کند که می تواند برای تولید وقفه استفاده شود .

در این قسمت می خواهیم با تایمر کانتر میکروکنترلر های AT91SAM7xxxx آشنا شویم . همانطور که مشخص است نحوه ی کار با تایمر کانتر توضیح داده شده در این قسمت مشابه کار با تایمر های دیگر محصولات ARM شرکت ATMEL می باشد .

این میکروکنترلر ها حاوی 3 تایمر کانتر مجزای 16 بیتی می باشند که در داخل دیتاشیت میکرو با نام واحد TC شناخته می شوند . اما عملکرد این سه تایمر کانتر کاملا مجزا بوده ، با این تفاوت که امکان انتخاب کلاک هر تایمر پیشرفته تر شده و به عنوان مثال می توان از ورودی کلاک خارجی هر تایمر به تایمر دیگر داد . به همین دلیل است که با یک نام TC شناخته شده است .

در جدول زیر هم پایه های مورد استفاده TC بر روی میکروکنترلر نشان داده شده است .

TC pin list

Pin Name	Description	Type
TCLK0-TCLK2	External Clock Input	Input
TIOA0-TIOA2	I/O Line A	I/O
TIOB0-TIOB2	I/O Line B	I/O

در این قسمت حداقل و مهمترین شرایط لازم برای راه اندازی TC آورده شده است . لذا اجرای دستورات زیر جهت کار با این قسمت الزامی است .

- خطوط I/O :

پایه ی I/O مورد استفاده برای ارتباط با تایمر کانتر (نظیر پایه های نشان داده شده در جدول بالا مانند TCLK0 یا TIOA2) ممکن است با بعضی از پایه های قسمت های جانبی دیگری مالتی پلکس شده باشد . کاربر بایستی ابتدا تنظیمات مربوط به قسمت PIO را لحاظ کند تا پایه های فوق به قسمت TC متصل شود . (جهت اطلاعات بیشتر به رجیستر های PIOx_PDR یا PIOx_PER ، PIOx_ASR یا PIOx_BSR مراجعه کنید .)

- قسمت PMC :

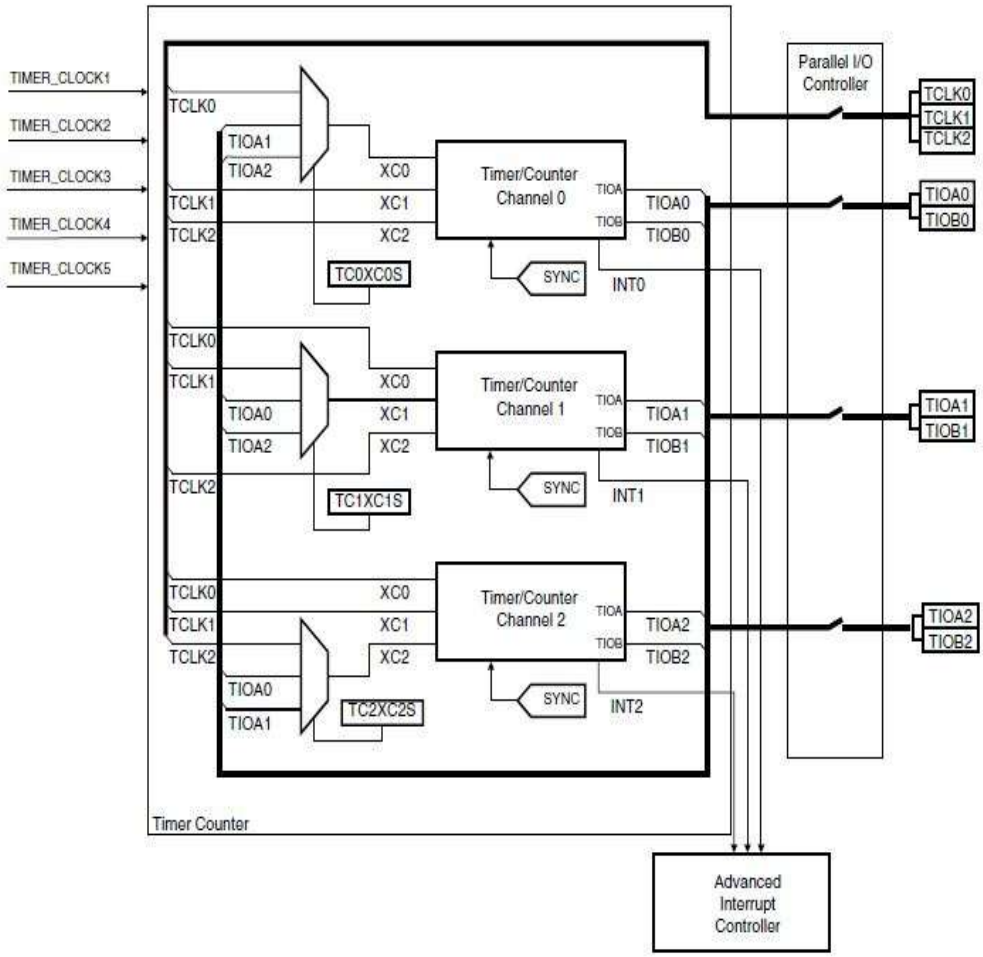
قسمت کلاک تایمر کانتر توسط PMC (Power Management Controller) کنترل می شود ، لذا ضروری است که برنامه نویس در ابتدا کلاک تایمرکانتر را فعال کند . (رجیستر های PMC_PCER و PMC_PCDR)

- وقفه :

تایمرکانتر دارای خطوط وقفه ی متصل به AIC (Advanced Interrupt Controller) می باشد . جهت استفاده از وقفه های تایمر کانتر تنظیمات مربوط به این قسمت نیز باید انجام شود .

توضیحات کامل مجموعه ی تایمرکانتر :

بعد از ایجاد شرایط اولیه برای تایمر نوبت به توضیح کامل این قسمت می رسد . توصیه می شود برای درک بهتر و کاملتر این قسمت تا پایان را چند بار تا یادگیری کامل بخوانید .در شکل زیر بلاک دیاگرام کلی TC نشان داده شده است . ابتدا خوب به قسمت های مختلف تایمر کانتر نگاه کنید .



همانطور که در شکل مشاهده می فرمایید سه کانال تایمر کانتر با نام TimerCounte Chanel x وجود دارد . در قسمت قبل نیز توضیح داده شد که هر کانال سه ورودی کلاک خارجی XCx دارد . این سه ورودی کلاک خارجی طوری طراحی شده اند که امکان ورود کلاک از خارج از میکروکنترلر به شمارنده در 5 راه وجود دارد . به عنوان مثال برای کانال 2 ، شما می توانید کلاک مورد نظر خود را به پایه های TCLK0 ، TCLK1 ، TCLK2 ، TIOA0 و TIOA1 (روی پایه های میکروکنترلر) قرار دهید . در این حالت تایمر شما در حالت کانتر (یا Capture Mode) پیکر بندی می شود . (مشابه حالت Counter در میکروکنترلر AVR) . در صورتیکه تایمر کانتر در حالت تایمر (یا Wave) پیکره بندی شده باشد ، کلاک های ورودی به رجیستر تایمر کانتر می تواند یکی از کلاک های زیر باشد . (لازم به ذکر است که در قسمت توضیحات رجیستر های TC به نحوه ی تنظیم حالات Capture یا Wave و همچنین نحوه ی انتخاب یکی از کلاک های زیر به طور کامل اشاره شده است .)

Timer Counter Clock Assignment

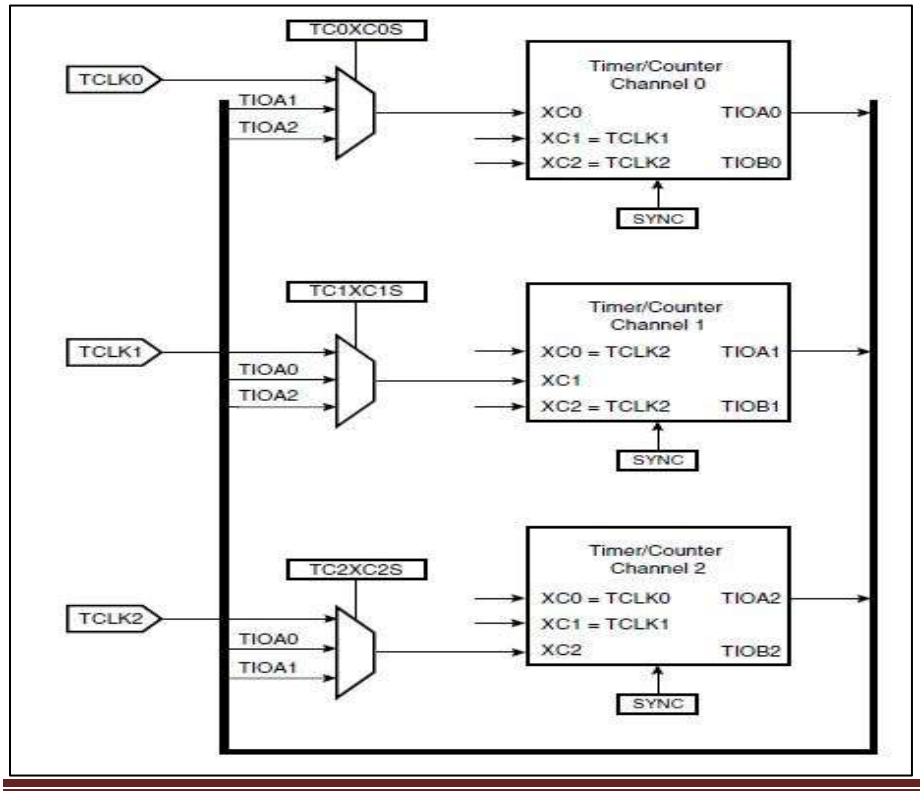
Name	Definition
TIMER_CLOCK1	MCK/2
TIMER_CLOCK2	MCK/8
TIMER_CLOCK3	MCK/32
TIMER_CLOCK4	MCK/128
TIMER_CLOCK5	MCK/1024

- نکته : کلاک های مورد انتخاب در شکل بالا به هر سه کانال متصل شده است و تنها کافی است که شما توسط رجیسترهای کنترلی تایمرکانتر اقدام به انتخاب یکی از آنها کنید .

یا به عبارتی دیگر هر کانال از رجیستر های شمارنده ، می تواند از منبع کلاک داخلی یا خارجی استفاده کند :

- ✓ منابع کلاک داخلی : TIMER_CLOCK0 , TIMER_CLOCK1 , TIMER_CLOCK2 , TIMER_CLOCK3 , TIMER_CLOCK4 , TIMER_CLOCK5
- ✓ منابع کلاک خارجی : XC0 , XC1 , XC2

به شکل زیر نگاه کنید :



حتما متوجه شدید که یک انتخاب مجدد برای ورودی های کلاک خارجی XC0 برای کانال صفر ، XC1 برای کانال یک و XC2 برای کانال دو وجود دارد . با کمک مالتی پلکسر های انتخاب که برای هر کدام از ورودی های کلاک بالا را در نظر گرفته شده است می توانید ، انتخاب ورودی های کلاک خارجی بیشتری داشته باشید .

با وجود این انعطاف پذیری در قسمت TC قادر به انتخاب ورودی کلاک خارجی زیادی برای هر کدام از کانال ها می شوید . برای درک بهتر این موضوع یک مثال می زنیم .

فرض کنید شما مجبور به استفاده از تایمر کانتر کانال 2 شده آید در صورتیکه تنها پایه ی TCLK0 از میکروکنترلر شما آزاد است . در این صورت با توجه به ویژگی جالب TC می توان به راحتی پایه ی TCLK0 را به تایمر کانتر کانال 2 داد و از آن بهره برد . به این ترتیب که (این موضوع را با نگاه کردن به بلاک دیاگرام کلی TC پیگیری کنید) با انجام تغییرات جزئی در رجیستر های کنترلی تایمر کانتر ، فیلد انتخاب کلاک خارجی را روی XC0 قرار می دهیم و از آنجا ، همانطور که در شکل مشخص است ، این ورودی کلاک خارجی به پایه ی TCLK0 متصل است و می توانید به این پایه کلاک خود را اعمال کنید . و یا اینکه در حالتی دیگر و با فرض مثال قبل ، می خواهیم از پایه ی TIOA0 کلاک خود را دریافت کنیم . در این صورت فیلد انتخاب کلاک خارجی را روی XC2 قرار داده و در مالتی پلکسر انتخابی XC2 ، ورودی دوم یعنی TIOA0 را انتخاب می کنیم . (در مورد چگونگی این انتخاب ها در قسمت های توضیحات رجیستری TC ، توضیحات بیشتری خواهیم داد .)

کنترل بیشتر بر روی کلاک هر کانال :

همچنین می توان کنترل کلاک تایمر کانتر را (چه در حالت Capture و چه در حالت Wave) تنظیم کرد . یعنی می توان کلاک را فعال یا غیر فعال ، متوقف و یا شروع کرد .

امکان فعال یا غیر فعال کردن کلاک توسط بیت CLKEN و CLKDIS و یا توسط رجیستر های مقایسه ای RA ، RB و RC ، وجود دارد .

کلاک تایمر کانتر هر کانال می تواند شروع یا متوقف شود . اعمالی همچون تریگر کردن (نرم افزاری ، خارجی و یا توسط رجیستر های مقایسه ای) ، همیشه کلاک تایمر را شروع می کنند. و یابه وسیله ی رجیستر های مقایسه ای RB و RC ، کلاک تایمر کانتر متوقف می شود.

- نکته : اعمال شروع و متوقف کردن کلاک تایمر کانتر وقتی معتبر است که کلاک تایمر فعال باشد .

تریگر کردن تایمر کانتر :

منظور از تریگر کردن ، بازنشانی شمارنده ی تایمر کانتر به صفر و شروع مجدد آن است . به طور کلی چهار راه برای تریگر کردن تایمر در حین کارش وجود دارد که یک حالت از چهار حالت آن بستگی به مد عملکرد تایمر کانتر (Capture یا Wave) ، که در بخش بعد توضیح خواهیم داد ، دارد . موارد زیر راه های تریگر کردن تایمر کانتر را نشان می دهد . توجه داشته باشید که موارد زیر به نوع عملکرد تایمر کانتر بستگی ندارد .

- تریگر نرم افزاری : هر کانال توسط بیت های موجود در رجیستر TC_CCRx و به طور جداگانه می تواند تریگر شود .
- هر کانال دارای یک ورودی SYNC هستند و به بیت SYNC در رجیستر TC_BCR متصل هستند . هنگامی که این بیت یک شود ، یک تریگر نرم افزاری برای هر سه کانال اتفاق می افتد .
- تریگرتوسط ثبات مقایسه ای RC : این رجیستر (RC) برای هر کانال جداگانه می باشد و در صورت برابری با مقدار تایمر کانتر کانال خود یک تریگر اتفاق می افتد . البته این نکته حائز اهمیت است که باید تنظیمات لازم برای چنین کاری در رجیستر های کنترلی تایمر کانتر انجام شود .

مورد چهارم تریگر ، که قبلا گفته شد برای هر مد عملکرد تایمر کانتر متفاوت است ، به عنوان تریگر خارجی شناخته می شود . در مد Capture ، تریگر خارجی می تواند یکی از پایه های TIOA و یا TIOB انتخاب شود . در مد Wave ، تریگر خارجی یکی از پایه های TIOB ، XC0 ، XC1 و XC2 است .

حالت های عملکرد TC :

هر کانال از تایمر کانتر می تواند در یکی از مدهای زیر کار کند :

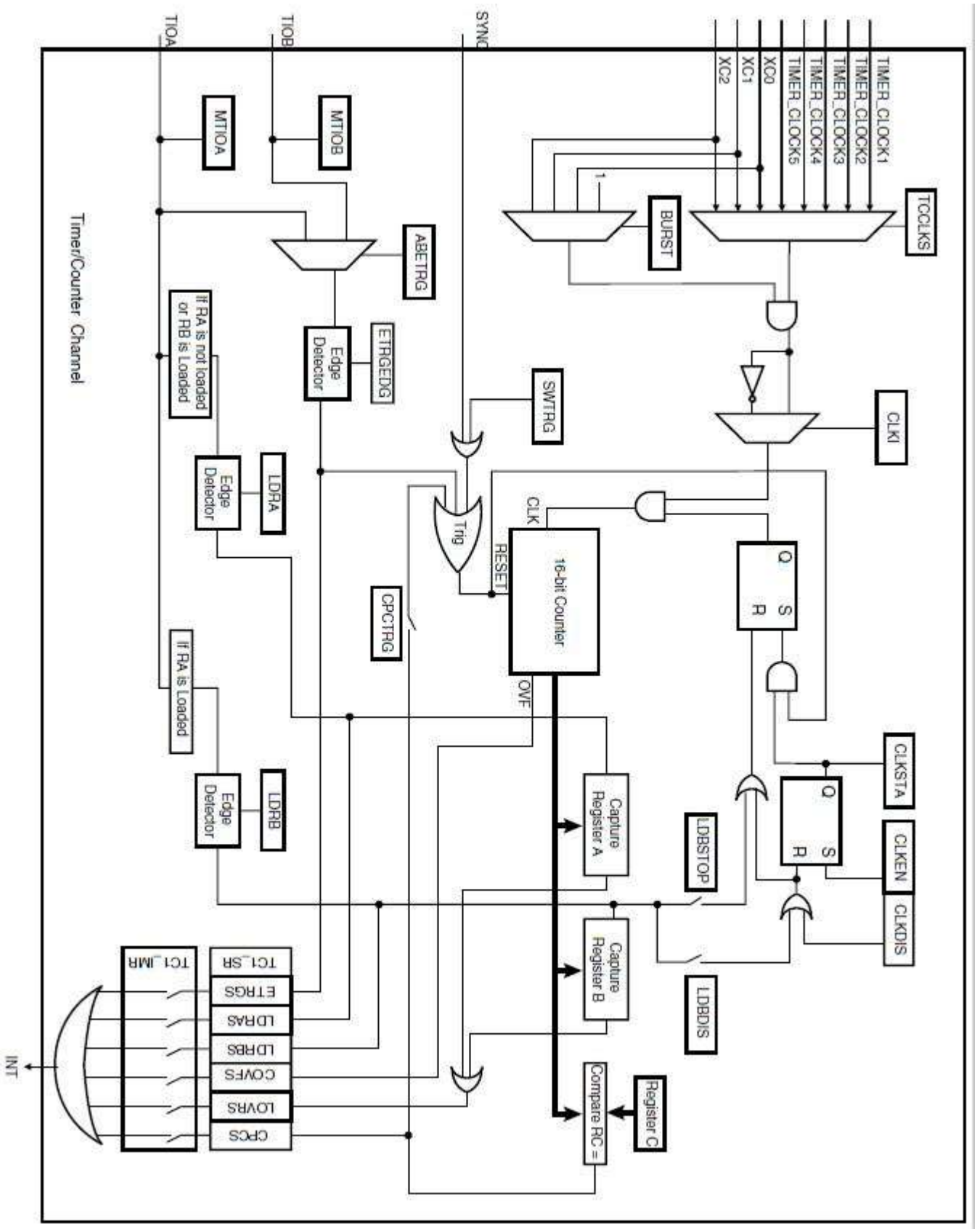
- مد Capture برای اندازه گیری سیگنال های خارجی (مشابه مد Counter در AVR)
- مد Wave جهت تولید سیگنال های مختلف (مشابه مد Timer در AVR)

در این شماره از مجله مد Capture را آموزش می دهیم . و آموزش مد Wave به دلیل گستردگی تنظیمات به شماره ی بعد موکول می شود . از این جا به بعد توضیحات هر قسمت به همراه نام پارامتر و یا رجیستر کنترلی همان قسمت آورده می شود . لذا تاکید می شود برای درک بهتر تنظیمات تایمر کانتر در مد Capture ، این قسمت را چند بار بخوانبد .

✓ مد Capture تایمر کانتر :

این مد با صفر کردن بیت Wave (بیت شماره 15) در رجیستر TC_CMR (Timer Counter Chanel Mode Register) فعال می شود . مد Capture به تایمر کانتر اجازه می دهد که اندازه گیری های روی سیگنال از جمله اندازه گیری فرکانس ، اندازه گیری عرض پالس ، پریود ، چرخه ی کار پالس و یا حتی اختلاف فاز بین سیگنال های پایه های TIOA و TIOB را انجام دهد .

شکل صفحه ی بعد اجزای داخلی تایمر کانتر در مد Capture را نشان می دهد . توصیه می شود به همراه توضیحات ، اجزای داخلی آن را نیز بررسی کنید .


TIMER_CLOCK1
TIMER_CLOCK2
TIMER_CLOCK3
TIMER_CLOCK4
TIMER_CLOCK5
XC0
XC1
XC2
TCCLKS
BURST
1
CLKI
SYNC
TIOB
TIOA
MTIOA
MTIOB
ABETRG
ETRGEDG
Edge Detector
If RA is not loaded or RB is Loaded
SWTRG
Edge Detector
LDRA
Trig
CLK
RESET
16-bit Counter
OVF
Q
S
R
CPCTRG
If RA is Loaded
Edge Detector
LDRB
Capture Register A
Capture Register B
Compare RC =
Register C
LDBSTOP
LDBDIS
CLKSTA
CLKEN
CLKDIS
TC1_IMR
TC1_SR
ETRGS
LDRAS
LDRBS
COVFS
LOVRS
CPCS
INT
Timer/Counter Channel

رجیستر های RA و RB :

رجیستر های A و B (RA و RB) به عنوان رجیسترهای Capture استفاده می شوند . (اگر به خاطر داشته باشید چنین رجیستری در تایمر کانتر میکروکنترلرهای AVR وجود داشت) . این به این معنی است که هنگام اتفاق افتادن یه واقعه رو پایه های TIOA ، می توان مقدار فعلی رجیستر شمارنده ی تایمر کانتر را در این دو رجیستر ذخیره (Capture) کرد . پارامتر LDRA در رجیستر کنترلی TC_CMR ، مشخص کننده ی لبه ی حساس پایه TIOA جهت بارگذاری RA می باشد . به همین ترتیب پارامتر LDRB مشخص کننده ی لبه ی حساس TIOA برای بارگذاری رجیستر RB می باشد .

تریگر در مد Capture :

همانطور که اشاره شد یکی از حالت های تریگر نرم افزاری ، تریگر توسط SYNC ، مقایسه رجیستر RC و تریگر خارجی می تواند شمارنده ی تایمر کانتر را بازنشانی کند . پارامتر ABETRG در TC_CMR ، مشخص کننده ی ورودی تریگر خارجی بین پایه های TIOA یاTIOB می باشد . بعد از انتخاب یکی از این دو حالت نوبت به تعیین لبه ی حساس برای تریگر می باشد . پارامتر ETRGEDG برای همین منظور تعبیه شده است . (حساس به لبه ی مثبت یا منفی و یا هردو) وقتی که این پارامتر مقدار صفر داشته باشد ، خاصیت تریگر خارجی غیر فعال می شود . (ETRGEDG = 0)

توضیحات اجمالی رجیستر های کنترلی TC در مد Capture :

مرحله ی آخر مربوط به شناسایی رجیستر ها و فیلد های آنان می باشد . در این قسمت خواهیم دید که چطور امکاناتی را که در قسمت های قبل گفته شد را به کار ببریم و تایمر کانتر را بسته به نیاز خود پیکره بندی کنیم .

شکل زیر کلیه ی رجیستر های واحد TC را نشان می دهد . توجه داشته باشید که رجیستر های TC_BCR و TC_BMR برای همه ی کانال های تایمر کانتر مشترک می باشد . بقیه ی رجیستر هایی که در شکل می بینید برای هر کانال به طور مستقل در دسترس است .

Register Mapping[1]

نام در کامپایلر KEIL	Register	Name	Access	Reset
*AT91C_TCx_CCR	Channel Control Register	TC_CCR	Write-only	–
*AT91C_TCx_CCR	Channel Mode Register	TC_CMR	Read-write	0
	Reserve			
	Reserved			
*AT91C_TCx_CV	Counter Value	TC_CV	Read-only	0
*AT91C_TCx_RA	Register A	TC_RA	Read-write[2]	0
*AT91C_TCx_RB	Register B	TC_RB	Read-write[2]	0
*AT91C_TCx_RC	Register C	TC_RC	Read-write	0
*AT91C_TCx_SR	Status Register	TC_SR	Read-only	0
*AT91C_TCx_IER	Interrupt Enable Register	TC_IER	Write-only	–
*AT91C_TCx_IDR	Interrupt Disable Register	TC_IDR	Write-only	–
*AT91C_TCx_IMR	Interrupt Mask Register	TC_IMR	Read-only	0
*AT91C_TCB_BCR	Block Control Register	TC_BCR	Write-only	–
*AT91C_TCB_BMR	Block Mode Register	TC_BMR	Read-write	0
	Reserved	–	–	–

Notes: 1. Channel index ranges from 0 to 2.
2. Read-only if WAVE = 0

برای دستیابی به این رجیستر ها و خواندن یا تغییر مقادیر آنها در کامپایلر KEIL ، از ردیف اول جدول کمک بگیرید.

- نکته : به دلیل اینکه بعضی از رجیستر ها برای هر کانال متفاوت است به جای x در نام رجیستر ها در کامپایلر KEIL شماره ی کانال مورد نظر قرار دهید .

برای مثال برای دسترسی به رجیستر TC_CCV تایمر کانتر کانال 1 به صورت زیر در کامپایلر قرار دهید :

```
*AT91C_TC1_CCR = 0x00000000;
```

- نکته : به نوع رجیستر (منظور Read only یا write only یا read write) توجه کنید . برای مثال شما نمی توانید در رجیستر شمارنده ی تایمر کانتر (TC_CV) چیزی بنویسید .

ثبات TC Block Control Register :

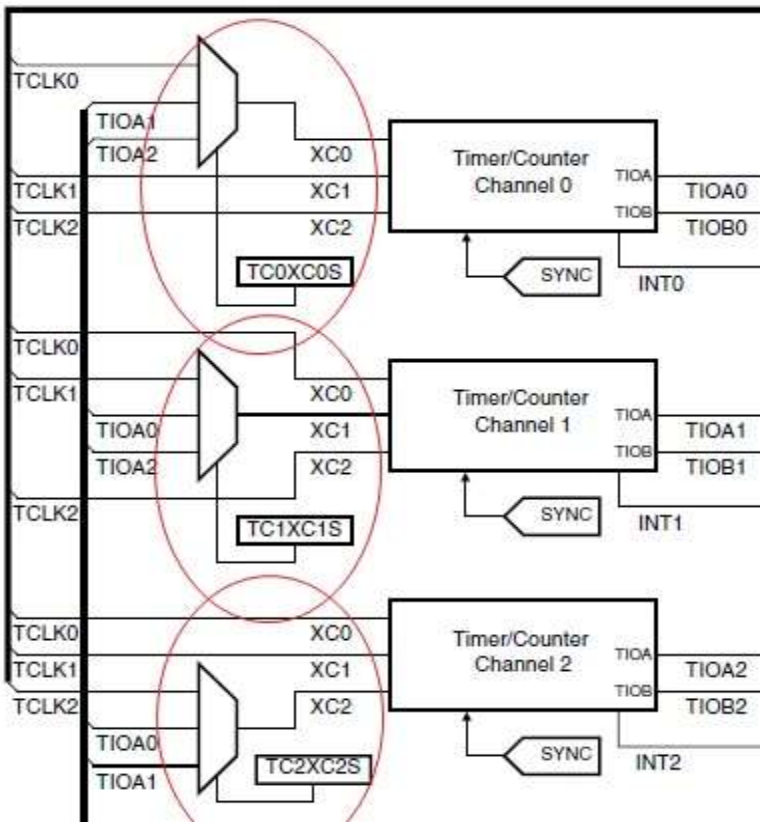
31	30	29	28	27	26	25	24
–	–	–	–	–	–	–	–
23	22	21	20	19	18	17	16
–	–	–	–	–	–	–	–
15	14	13	12	11	10	9	8
–	–	–	–	–	–	–	–
7	6	5	4	3	2	1	0
–	–	–	–	–	–	–	SYNC

- بیت SYNC : هنگامی که این بیت یک می شود یک تریگر نرم افزاری برای هر کانال اتفاق می افتد و مقدار شمارنده ی تایمر کانتر بازنشانی می شود .

ثبات TC Block Mode 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
–	–	TC2XC2S		TC1XC1S		TC0XC0S	

این سه فیلد در واقع همان انتخاب دوم ورودی کلاک خارجی XCx می باشد که در شکل زیر نشان داده شده است .



- فیلد TC0XC0S : که بسته به مقداری که به آن می دهیم می تواند یکی از ورودی های خودش را انتخاب کند

TC0XC0S		Signal Connected to XC0
0	0	TCLK0
0	1	none
1	0	TIOA1
1	1	TIOA2

- فیلد TC1XC1S :

TC1XC1S		Signal Connected to XC1
0	0	TCLK1
0	1	none
1	0	TIOA0
1	1	TIOA2

- فیلد TC2XC2S :

TC2XC2S		Signal Connected to XC2
0	0	TCLK2
0	1	none
1	0	TIOA0
1	1	TIOA1

ثبات TC Chanel Control 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
–	–	–	–	–	SWTRG	CLKDIS	CLKEN

- بیت CLKEN : این بیت کلاک تایمر کانتر را فعال می کند . می توان با استفاده از این بیت ، به صورت نرم افزاری ، کلاک تایمر کانتر را کنترل کرد .

نکته : این رجیستر از جمله رجیستر هایی است که برای هر کانال جداگانه می باشد .

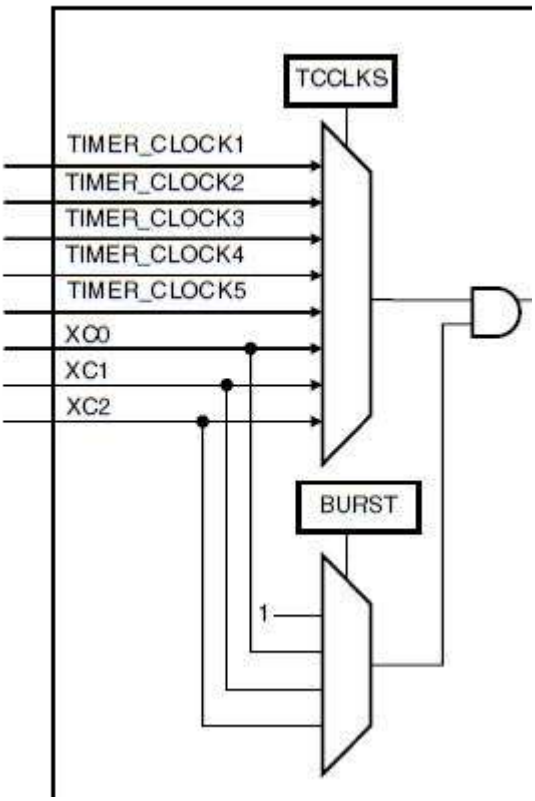
- بیت CLKDIS : این بیت کلاک تایمر کانتر کانال مربوط به خود را غیر فعال می کند .
- بیت SWTRG : این بیت برای تریگر نرم افزاری کانال مربوط به خود استفاده می شود . زمانی که این بیت یک می شود تریگر اتفاق می افتد .

ثبات TC Chanel Mode Register در مد Capture :

قبل از توضیح این قسمت لازم به ذکر است که این ثبات بسته به مد عملکرد خودش ، دارای فیلد ها و بیت های مخصوص خودش می باشد . آنچه که در اینجا توضیح داده می شود ، قسمت های مختلف این رجیستر در مد Capture می باشد . در شماره ی بعدی مجله مد Wave این ثبات توضیح داده خواهد شد .

31	30	29	28	27	26	25	24
–	–	–	–	–	–	–	–
23	22	21	20	19	18	17	16
–	–	–	–	LDRB		LDRA	
15	14	13	12	11	10	9	8
WAVE	CPCTRG	–	–	–	ABETRG	ETRGEDG	
7	6	5	4	3	2	1	0
LDBDIS	LDBSTOP	BURST		CLKI	TCCLKS		

- فیلد TCCLKS : این فیلد تعیین کننده ی کلاک ورودی به رجیستر شمارنده ی تایمر کانتر است . در واقع این فیلد موقعیت زیر را دارد . همانطور که در شکل پیداست می توان از بین سیگنال های TIMER_CLOCKx و XCx یکی را به ورودی کلاک تایمر کانتر هدایت کرد .



TCCLKS			Clock Selected
0	0	0	TIMER_CLOCK1
0	0	1	TIMER_CLOCK2
0	1	0	TIMER_CLOCK3
0	1	1	TIMER_CLOCK4
1	0	0	TIMER_CLOCK5
1	0	1	XC0
1	1	0	XC1
1	1	1	XC2

- بیت CLKI : هنگامی که این بیت صفر باشد با لبه ی بالا رونده ی کلاک تایمر کانتر افزایش پیدا می کند . و در صورتی که مقدار یک داشته باشد با لبه ی صعودی سیگنال ساعت افزایش پیدا می کند .
- فیلد BURST : موقعیت این فیلد در شکل بالا نشان داده شد . در واقع وظیفه ی این قسمت ، همان شروع و متوقف سازی عملکرد تایمر کانتر می باشد . همانطور که در شکل می بینید بسته به مقدار این فیلد یکی از ورودی های آن فعال و به خروجی مالتی پلکسر هدایت می شود و سپس خروجی مالتی پلکسر به یک گیت AND داده شده است . در صورتی که ورودی انتخاب شده توسط این فیلد مقدار یک داشته باشد ، یکی از ورودی های گیت AND نیز مقدار یک دارد و در نتیجه کلاک تایمر فعال می شود و به قسمت های دیگر تایمر کانتر می رسد . در غیر اینصورت در صورتی که سیگنال انتخاب شده مقدار صفر داشته باشد ، یکی از ورودی گیت AND صفر شده و کلاک تایمر کانتر جلوتر نمی رود .

BURST		
0	0	The clock is not gated by an external signal.
0	1	XC0 is ANDed with the selected clock.
1	0	XC1 is ANDed with the selected clock.
1	1	XC2 is ANDed with the selected clock.

- بیت LDBSTOP : در صورتی که این بیت یک باشد با بار شدن ثبات B (RB) کلاک تایمر کانتر متوقف می شود
- بیت LDBDIS : در صورتیکه این بیت یک شود با بارشدن رجیستر RB کلاک تایمر کانتر غیر فعال می شود .
- فیلد ETRGEDG : بسته به مقداری که به این فیلد می دهید می توانید لبه ی حساس جهت تریگر کردن خارجی تایمر کانتر را تعیین کنید . همانطور که در قسمت های قبل گفته شد تایمر منابع مختلفی برای تریگرکردن خود دارد که یکی از آنها تریگرکردن خارجی بود . این فیلد لبه ی حساس به تریگر خارجی را تعیین می کند .

ETRGEDG		Edge
0	0	none
0	1	rising edge
1	0	falling edge
1	1	each edge

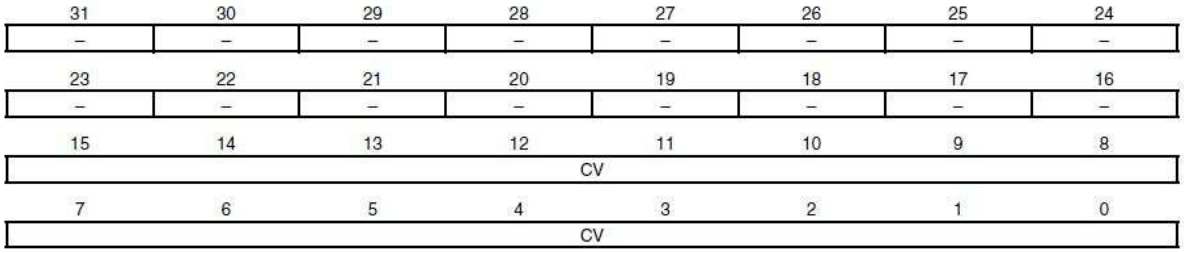
- بیت ABETRG : این بیت تعیین کننده ی ورودی تریگر خارجی تایمر کانتر می باشد . در صورتیکه مقدار صفر داشته باشد پایه ی TIOB ورودی تریگر خارجی است و در غیر اینصورت اگر یک شود پایه ی TIOA وظیفه ی تریگر کردن خارجی تایمر را بر عهده دارد . (لبه ی حساس جهت تریگر خارجی روی این پایه در فیلد قبل توضیح داده شد)
- بیت CPCTRG : این بیت اگر یک شود موجب فعال کردن تریگر مقایسه ی ثبات RC می شود . بعد از یک شدن این بیت ، اگر مقدار تایمر کانتر با مقدار موجود در ثبات RC برابر شد یک تریگر اتفاق می افتد .
- بیت Wave : این بیت اگر صفر باشد تایمر کانتر در مد Capture است . در غیر اینصورت (اگر یک شود) تایمر در مد Wave شروع به کار می کند .
- فیلد LDRA : این فیلد تعیین کننده ی لبه ی حساس برای Capture کردن روی رجیستر RA می باشد .

LDRA		Edge
0	0	none
0	1	rising edge of TIOA
1	0	falling edge of TIOA
1	1	each edge of TIOA

- فیلد LDRB : این فیلد تعیین کننده ی لبه ی حساس برای Capture کردن روی رجیستر RB می باشد .

LDRB		Edge
0	0	none
0	1	rising edge of TIOA
1	0	falling edge of TIOA
1	1	each edge of TIOA

ثبات TC Counter Value Register :

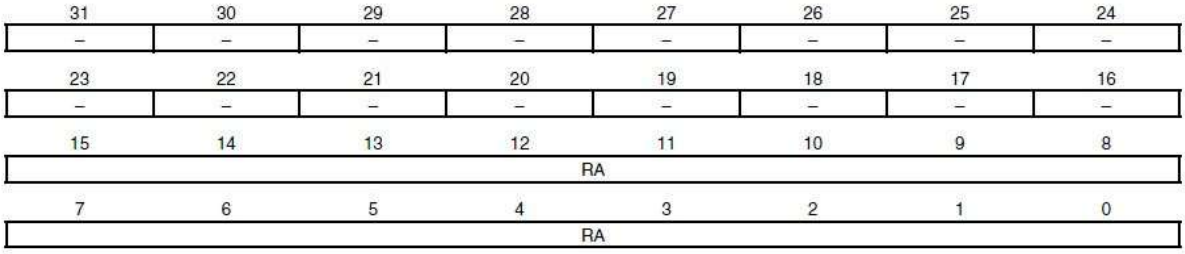


- فیلد CV : این ثبات مقدار فعلی رجیستر شمارنده ی تایمر کانتر را در بر دارد . توجه داشته باشید که این ثبات Read Only می باشد و امکان بارگذاری در این ثبات وجود ندارد . برای خواندن مقدار این رجیستر به صورت زیر در کامپایلر KEIL عمل کنید . (فرض کنید که a یک متغیر از نوع int وجود دارد .)

```
a = *AT91C_TC0_CV;
```

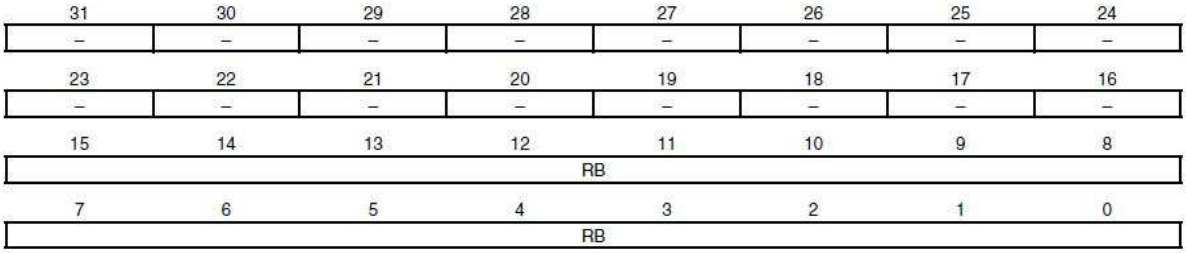
دستور بالا مقدار ثبات شمارنده ی تایمرکانتر شماره صفر را در متغیر a قرار می دهد .

ثبات TC Register A :



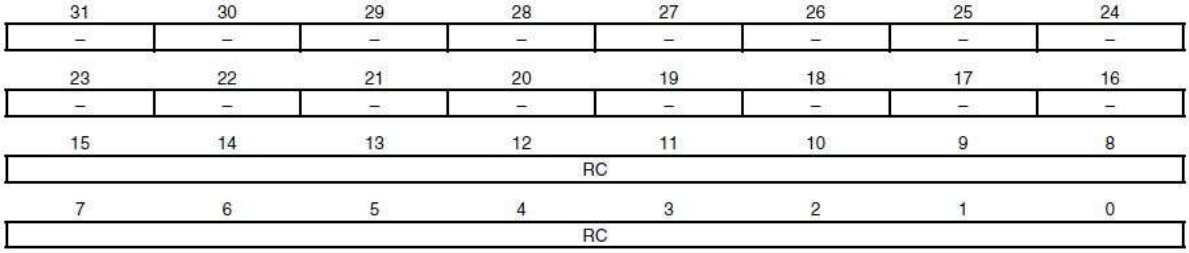
- فیلد RA : این فیلد و رجیستر ، مقدار مقایسه RA را در خود ذخیره می کند یا حاصل Capture در این رجیستر می باشد .

ثبات TC Register B :



- فیلد RB : این فیلد و رجیستر ، مقدار مقایسه RB را در خود ذخیره می کند یا حاصل Capture در این رجیستر می باشد .

ثبات TC Register C :



- فیلد RC : این فیلد و رجیستر ، مقدار مقایسه RC را در خود ذخیره می کند یا حاصل Capture در این رجیستر می باشد .

ثبات وضعیت TC Status Register :

31	30	29	28	27	26	25	24
–	–	–	–	–	–	–	–
23	22	21	20	19	18	17	16
–	–	–	–	–	MTIOB	MTIOA	CLKSTA
15	14	13	12	11	10	9	8
–	–	–	–	–	–	–	–
7	6	5	4	3	2	1	0
ETRGS	LDRBS	LDRAS	CPCS	CPBS	CPAS	LOVRS	COVFS

- بیت COVFS : اگر مقدار یک داشته باشد ، حداقل یک سرریز از آخرین خواندن از رجیستر TC_CV انجام شده است .
- بیت LOVRS : حداقل دوبار در درون رجیستر های RA ویا RB بارگذاری شده است ، بدون خواندن از رجیستر مربوطه و یا خواندن از ثبات وضعیت تایمر کانتر . (اگر Wave = 0 باشد)
- بیت های CPAS , CPBS , CPCS : این بیت ها برای مد Wave کاربرد دارند که توضیح داده خواهند شد .
- بیت LDRAS : در صورتیکه مقدار یک داشته باشد یک بارگذاری در درون ثبات RA ، از آخرین خواندن از ثبات وضعیت تایمر کانتر اتفاق افتاده است .
- بیت LDRBS : در صورتیکه مقدار یک داشته باشد یک بارگذاری در درون ثبات RB ، از آخرین خواندن از ثبات وضعیت تایمر کانتر اتفاق افتاده است .
- بیت ETRGS : اگر مقدار یک داشته باشد یک تریگر خارجی از آخرین خواندن از ثبات وضعیت تایمر کانتر اتفاق افتاده است .
- بیت CLKSTA : اگر صفر باشد دلالت بر غیر فعال بودن کلاک تایمر کانتر است . و در صورتیکه یک باشد کلاک تایمر کانتر همچنان فعال است .
- بیت MTIOA : این بیت همان سطح منطقی پایه ی TIOA می باشد . توجه داشته باشید که این بیت در صورتیکه تایمر کانتر در مد Wave باشد نشان دهنده ی چیز دیگری است .
- بیت MTIOB : این بیت همان سطح منطقی پایه ی TIOB می باشد . توجه داشته باشید که این بیت در صورتیکه تایمر کانتر در مد Wave باشد نشان دهنده ی چیز دیگری است .

ثبات های TC_IER , TC_IDR , TC_IMR :

در این قسمت ثبات های مربوط به وقفه را توضیح خواهیم داد . نکته ی اول اینکه برای رجیسترهای فعال سازی و غیر فعال سازی هر وقفه ، یک ثبات مستقل وجود دارد (TC_IER و TC_IDR) و یک ثبات جداگانه برای آگاهی از وضعیت فعال یا غیر فعال بودن وقفه ای مشخص (TC_IMR) در دسترس است . (نمی توان رجیستر های فعال یا غیر فعال سازی وقفه ها را خواند .)

ابتدا موقعیت بیت های مربوط به هر نوع وقفه ی تایمرکانتر را توضیح می دهیم . سپس با استفاده از همین توضیحات می توان با سه ثبات فعال سازی وقفه (TC_IER) ، غیر فعال سازی وقفه (TC_IDR) و ثبات وضعیت فعال یا غیر فعال بودن وقفه (TC_IMR) ، کار کرد .

به عنوان مثال برای فعال سازی وقفه ی سرریز تایمر کانتر بیت صفر ثبات TC_IER را یک می کنیم و برای غیر فعال سازی همین وقفه سراغ همان موقعیت بیتی می رویم (یعنی همان موقعیت بیت صفرم) ، با این تفاوت که باید مقدار بیت صفرم ثبات TC_IDR یک شود . و به همین ترتیب برای آگاهی از وضعیت فعال بودن یا غیر فعال بودن وقفه بیت صفر ثبات TC_IMR را بررسی می کنیم ، در صورتیکه این بیت صفر باشد ، وقفه غیر فعال و اگر یک باشد ، وقفه ی مربوطه فعال می باشد .

ترتیب موقعیت بیت های وقفه های مختلف تایمر کانتر به صورت زیر است .

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
ETRGS	LDRBS	LDRAS	CPCS	CPBS	CPAS	LOVRS	COVFS

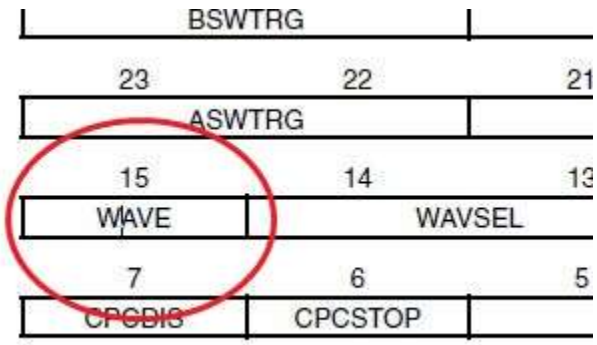
- بیت COVFS : مربوط یه وقفه ی سرریز تایمرکانتر می باشد .
- بیت LOVRS : وقفه ی مربوط به بارگذاری بیش از دو بار از آخرین خواندن از رجیستر های RA و RB و یا ثبات وضعیت تایمر کانتر می باشد .
- بیت های CPAS , CPBS , CPCS : این وقفه ها در مد Wave توضیح داده خواهند شد .

- بیت LDRAS : وقفه ی مربوط به بارگذاری در درون رجیستر RA
- بیت LDRBS : وقفه ی مربوط به بارگذاری در درون ثبات RB
- بیت ETRGS : وقفه ی مربوط به اتفاق افتادن یک تریگر خارجی

یک مثال ساده !

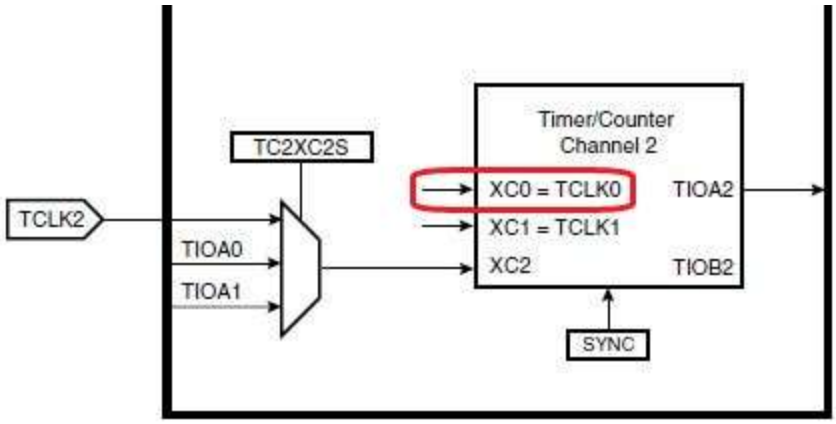
برای درک بهتر کار با واحد تایمر کانتر (مد Capture) یک برنامه ی ساده آورده شده است تا نمونه ی پیکره بندی و کار با ثبات های تایمرکانتر آموزش داده شود . مرحله به مرحله ی انجام تنظیمات به همراه شکل نشان داده شده است .

در این مثال تایمر کانتر 2 را در مد Capture قرار داده ایم . لذا باید بیت 15 رجیستر TC_CMR2 حتما صفر باشد .



می خواهیم یک شمارنده با تایمر کانتر شماره 2 بسازیم که ورودی کلاک (یا همان میکرو سوئیچ) پایه ی TCLK0 (PB.12) باشد . پس شروع به تصمیم گیری در مورد ورودی کلاک تایمرکانتر می کنیم . در وحله ی اول باید کلاک تایمرخود را مشخص کنیم و به دلیل اینکه کلاک ما خارجی (خارج از میکروکنترلر می باشد) باید از یکی از ورودی های کلاک خارجی XCx تایمر کانتر 2 که به پایه ی TCLK0 متصل است را انتخاب کنیم .

شکل زیر می تواند درک ما را بهتر کند . به شکل نگاه کنید .

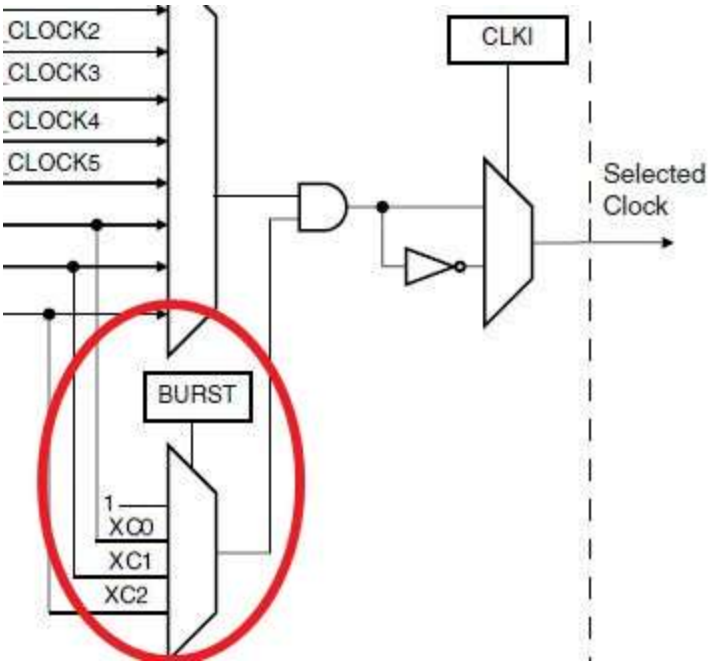


همانطورکه در شکل می بینید تنها پایه ی XC0 به TCLK0 متصل است . در صورتیکه شما بخواهید (به دلایل مختلفی همچون اشغال بودن پایه ی TCLK0) از پایه ی TCLK1 به عنوان ورودی کلاک بهره ببرید باید ورودی کلاک XC1 را انتخاب کنید . سپس سراغ فیلد TCCLKS در رجیستر TC_CMR2 می رویم . در شکل سمت چپ نیز ورودی های انتخاب شده بسته به مقدار این فیلد را نشان می دهد .

TCCLKS			Clock Selected
0	0	0	TIMER_CLOCK1
0	0	1	TIMER_CLOCK2
0	1	0	TIMER_CLOCK3
0	1	1	TIMER_CLOCK4
1	0	0	TIMER_CLOCK5
1	0	1	XC0
1	1	0	XC1
1	1	1	XC2

پس مقدار این فیلد باید 0x5 (به صورت باینری 0101) باشد .

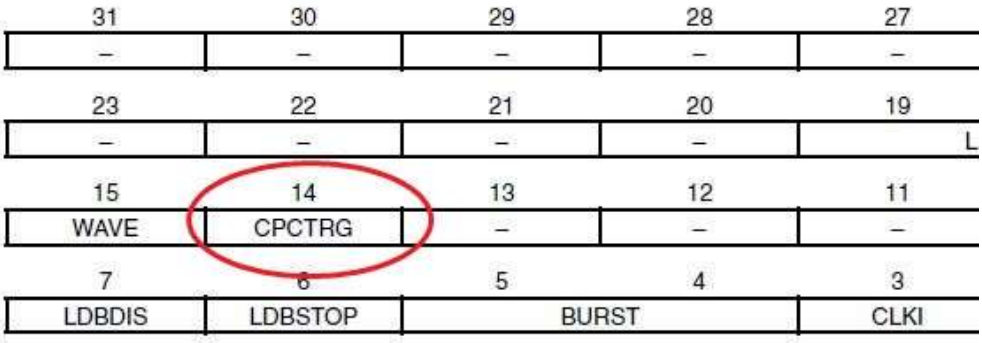
برنامه هر بار در درون حلقه ی بینهایت خودش مقدار شمارنده ی تایمر کانتر خود را روی پورت A قرار می دهد . همچنین به وسیله ی ورودی خارجی XC1 که به پایه ی TCLK1 متصل است ، عملیات توقف و ادامه شمارش را کنترل کنیم . برای این منظور از فیلدBURST کمک می گیریم .



به کمک این قسمت ورودی خارجی XC1 که به TCLK1 میکروکنترلر متصل است را انتخاب می کنیم که به ورودی گیت AND اعمال شده است . اگر پایه ی TCLK1 مقدار یک داشته باشد ، کلاک انتخاب شده توسط فیلد TCCLKS به داخل تایمر کانتر هدایت می شود (یعنی همان ادامه ی شمارش تایمر کانتر) و اگر سطح منطقی این پایه صفر شد ، ورودی گیت AND نیز صفر شده و در نتیجه کلاک به داخل تایمر کانتر هدایت نمی شود و شمارش متوقف می شود . این همان چیزی است که در طراحی اولیه مسئله لازم داشتیم . فیلد BURST در ثبات TC_CMR2 به همراه جدول عملکرد آن در زیر نشان داده شده است .

BURST		
0	0	The clock is not gated by an external signal.
0	1	XC0 is ANDed with the selected clock.
1	0	XC1 is ANDed with the selected clock.
1	1	XC2 is ANDed with the selected clock.

و در ادامه تایمر کانتر را طوری تنظیم می کنیم که با رسیدن به عدد 20 ، صفر شود (تریگر شود) این کار به راحتی و با تنظیم تریگر کردن توسط رجیستر RC امکان پذیر است . به این صورت که به رجیستر RC مقدار 20 می دهیم و بیت CPCTRG را تنظیم می کنیم . شکل زیر موقعیت این بیت را نشان می دهد .



اگر مقدار این بیت صفر باشد ، مقدار RC هیچ تاثیری روی سیگنال تریگر ندارد .(یا به عبارتی دیگر تریگر مقایسه ی RC غیر فعال می باشد) اما اگر یک باشد با مساوی شدن رجیستر RC با مقدار شمارنده ی تایمر کانتر 2 یک تریگر اتفاق می افتد .

به عنوان یک تنظیم اضافی ، بیت CLKI را نیز یک می کنیم . با این کار سیگنال روی پایه ی TCLK0 (کلاک تایمر کانتر) با لبه ی بالا رونده ، موجب افزایش شمارنده ی تایمر کانتر می شود . موقعیت این بیت را درشکل زیر می بینید .

❖ برنامه ی نهایی :

و در انتها برنامه ای که پیش فرض های مسئله ی ما را اجرا کند را در زیر می بینید . نکته ی مهم اینکه لازم نیست که به دلیل اینکه پایه ی TCLK0 جزو Pheripheral دوم پورت B می باشد ، تنظیمات Pheripheral ها را در PIO اعمال کنیم . زیرا Pheripheral هایی که به عنوان ورودی هستند نیازی به تنظیم ندارند .

نکته ی دوم اینکه برای داشتن مقدار اولیه ی صفر برای رجیستر TC_CV2 در ابتدای برنامه یک تریگر نرم افزاری (بیت SWTRG) انجام شده است .

نکته سوم اینکه بیت CLKEN در رجیستر TC_CCR2 ،برای فعال شدن کلاک تایمرکانتر یک شده است.(CLKDIS = 0)

و نکته چهارم اینکه مقاومت بالاکش درونی میکروکنترلر برای پایه های TCLK0 و TCLK1 نیز فعال شده است .

```
// AT91SAM7X256 Register Name Library
#include < AT91SAM7X256.h>
int main(void)
{
// Enable TC Clock
* AT91C_PMC_PCER = (1<< AT91C_ID_TC2);
// Timer Counter Configuration : Capture Mode
// TC_CCR2 Configuration : (CLKEN = 1) (CLKDIS = 0) (SWTRG = 1)
* AT91C_TC2_CCR = 0x00000005;
// TC_CMR2 Configuration : (TCCLKS = 101) (CLKI = 1) (BURST = 10) (CPCTRG = 1) (WAVE =0)
* AT91C_TC2_CMR = 0x0000402D;
// Put 20 In The Timer Counter Register C (RC)
* AT91C_TC2_RC = 20;
// Enable PB.12 And PB.19 Embedded Pullup Resistor
* AT91C_PIOB_PPUER = (1<<12) | (1<<19);
// Make PortA Output
* AT91C_PIOA_OER = 0xFFFFFFFF;
while(1)
{
// show the TC2 value On PortA
* AT91C_PIOA_CODR = 0xFFFFFFFF;
* AT91C_PIOA_SODR = * AT91C_TC2_CV;
};
}
```