صنایع تغذیه "Power" و پینهای میکروکنترلر ATSISAM7S64/128/256/512

نام پین	شماره پین	کاربرد	ولتاژ
V_{DDIN}	7	ولتاژ ورودی برای رگولاتور 1,85 ولتی داخلی	ولتاژ تغذیه ورودی کیپ → 3 تا 3,6 V
V_{DDOUT}	8	ولتاژ خروجی رگولاتور 1,85 ولتی	1,85 V ← ولتاژ کیپ (جهت استفاده برای موارد زیر)
$V_{DDFLASH}$	59	ولتاژ Flash	ورودی کیپ 3 تا 3,6 V
V_{DDIO}	18,45,58	ولتاژ پین‌های ورودی و خروجی	ورودی کیپ 3 تا 3,6 V 1,65 تا 1,95 V
V_{DDCORE}	12,24,54	ولتاژ هسته میکروکنترلر ARM7 core	ورودی کیپ 1,65 تا 1,95 V (ولتاژ خروجی رگولاتور 1,85 V را وصل می‌کنیم)
$V_{DD\ PLL}$	64	ولتاژ PLL	ورودی کیپ 1,65 تا 1,95 V
GND	2,17,46,60	زمین مدار	—

این میکروها فقط یک پورت ورودی و خروجی 32 بیتی بنام A دارند $PA_0 - PA_{31}$

منابع Reset در ATSISAM7S

1. ریست در زمان روشن شدن
2. ریست خارجی
3. ریست بوسیله watch dog
4. ریست توسط آشکارساز Brownout
5. ریست نرم‌افزاری

main clock → خروجی واحد PMC

MCK

① با وصل شدن ولتاژ هسته "VDDCORE" مدتی طول می‌کشد تا کلاک اسیلاتور داخلی (اصلی سیستم) به دامنه و فرکانس اصلی خود برسد، در این مدت که Start Up Time نام دارد، میکرو ریست می‌باشد

با سپری شدن این زمان، کلاک اصلی سیستم "MCK" (main clock) وصل می‌شود

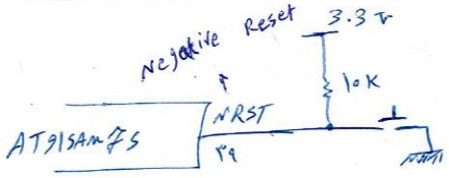
تذکر: زمان Start Up را بکمک 8 بیت رجیستر CKGR_MOR (main oscillator mode register) تحت عنوان OSCOUNT0-7 (oscillator counter) محاسبه می‌کنند.

Start Up Time = 8 * OSCOUNT / SLCK → 32,768 K

(oscillator counter, slow clock)

تذکر: بعد از سپری شدن ۳ سیکل از کلاک اصلی میکروکنترلر تنظیمات خود را انجام می‌دهد

۴



۲) مطابق مدار زیر می‌توان ریست خارجی انجام داد.

۳) سریز تایمر watch dog باعث ریست شدن میکرو می‌شود.

۴) ریست توسط آشکارساز Brown out

1.65 - 1.95
۲.۷
حدود
brownout threshold"

در صورتی که تغذیه هسته "CPU" یعنی ولتاژ پین VDD CORE از ولتاژ مرجع داخلی "بیشتر شود بعد از ۲ ماشین سیکل میکرو Reset می‌شود. "جهت جلوگیری از آسیب هسته"

تذکره: در حالت ۴ و ۳ پایه NRST برای مدت زمان مشخص بعنوان خروجی شده و در حال صفر قرار می‌گیرد.

۵) ریست توسط نرم افزار:

در برنامه هرکجا بخواهیم می‌توان میکرو و ادوات جانبی آنرا ریست کرد.

Reset controller control Register — رجیستر کنترل ریست

با نوشتن یک در بیت های رجیستر RSTC_CR می‌توان ریست انجام داد:

الف) با نوشتن یک در بیت PROCRST میکرو پردازنده "CPU" و watch dog ریست می‌شود.
(prossesor — رجیستهای داخلی CPU)

ب) " " " PERRST ادوات جانبی و حافظه سیستم، ریست می‌شود.
(Peripheral Reset)

ج) " " " EXTRST پایه NRST برای مدتی مشخص صفر می‌شود.
(External Reset)

~~ریست خارجی غیرفعال می‌گردد~~ ریست اتفاق می‌افتد. (میکرو کلا ریست می‌شود)

تذکره: تایمر نگهبان 12 بیتی می‌باشد که می‌توان توسط بیتهای WDT (watch dog value) در رجیستر WDT_MR watch dog Timer mode Register مقدار دهی کرد.

تذکره: WDT از کلاک اسیلاتور RC داخلی که بر 128 تقسیم شده است استفاده می‌کند. SLCK ۳۲ KHZ

نکته! با یک شدن بیت WDRSTT از رجیستر کنترلی WDT_CR watchdog Timer control Register تایمر نگهبان Reset می‌شود.
بیت صفرم

(۴) RSTC_RMR

(۶)

تمرین

بیتهای رجیستر های

رجیستر مد کنترلر ریست RSTC_MR ، رجیستر وضعیت کنترلر ریست RSTC_SR Reset controller status Register ، کنترلر ریست RSTC_CR Reset controller_ control Register

رجیستر وضعیت تایمر نگهبان WDT_SR ، WDT_MR ، رجیستر کنترل تایمر نگهبان WDT_CR

WDTC_WDMR watch dog Timer_Mode Register watch dog Timer_

را توضیح دهید.

برنامه بنویسم که

نحوه نوشتن درست است

WDTC_WDMR

مقدار = نام رجیستر <- نام ماژول بنویسید AT91C_BASE_

AT91C_BASE_RSTC -> RSTC_RMR = 0xA5000001;

Reset خارجی فعال شده است.

بیت اولی از رجیستر را یک کنید تا Reset خارجی فعال شود.

روشهای متعددی در این روشی همه بیتها تعیین تاثیر مقدار می‌گیرند

غیر فعال کردن (خاموش کردن) تایمر نگهبان

AT91C_BASE_WDTC -> WDTC_WDMR = (1<<15);

روشی شیفتی (بیت ۱۵ یک شود) در این حالت فقط بیت ۱۵ تغییر می‌شود و به بقیه بیتها کاری ندارم

نکته ۲: بیت WDDIS را یک کنید تا تایمر نگهبان خاموش شود (غیر فعال)

بیت شماره ۱۵

پیش فرض روشن است.

بجای این می‌توان نوشت روش شناسه

AT91C_WDTC_WDDIS

کلمه پرانتز

خاموش کردن بیت از طریق شناسه

یعنی این بیت از این رجیستر یک شود.

برای اینکه روی بقیه بیتها تاثیر نداشته باشد.

واحد "MAM" (واحد مدیریت حافظه)

میکروکنترلرهای ARM زمان دسترسی "Access Time" به حافظه فلش "Flash Rom" زیاد است. پردازنده ARM7 با 60MHz کار می‌کند، اما از آنجا که زمان دسترسی به حافظه Flash، 50ns می‌باشد یعنی سرعت (فرکانس) 20MHz (یک چهارم سرعت CPU) عملاً سرعت کار کاهش پیدا می‌کند.

$\frac{1}{50 \times 10^{-9}} = 20 \times 10^{6} = 20MHz$

روشهای غلبه بر این مشکل:

۱- ساده‌ترین روش آنست که بخشهایی از برنامه که حساس به زمان می‌باشد در RAM بارگذاری کنیم.
عیب این روش آنست که RAM محدود و گران است. (مانند ساختن سنتیسایزر با میکرو)
اول نغمه‌ها را سیازیم و در RAM بارگذاری کنیم
بعد بخوانیم و به DAC بدهیم.

۲- استفاده از حافظه نهان "cashe":

حافظه نهان "کش"، حافظه کوچکی است بین CPU و حافظه Flash Rom که مانند آخرین خانه‌های حافظه را که به آنها دسترسی داشته‌ایم در خود نگهداری می‌کند.
عیب این روش آنست که cashe کلاً سخت افزار پیچیده‌ای دارد که بخش عظیمی از سطح تراشه (سیلیکون) را اشغال می‌کند که هم قیمت و هم توان مصرفی را افزایش می‌دهد.

۳- حالت MAM "Memory Acceleration module" حالت میانی بین حالت دسترسی مستقیم به فلش و حالت حافظه cashe، در این حالت این حافظه مانند cashe دستورالعمل بعدی را که قرار است CPU آن را اجرا کند، در حافظه سریع خود نگهداری می‌کند.
این حافظه ۱۲۸ بیت پهنا دارد (Data Bus) و در یک دسترسی به حافظه فلش "Rom" امکان بارگذاری ۴ دستورالعمل ۳۲ بیتی ARM یا هشت دستورالعمل ۱۶ بیتی Thumb در حافظه MAM را دارد. یعنی همواره چند دستورالعمل بعدی از حافظه فلش "Rom" به واحد MAM منتقل شده و CPU دستورالعمل را سریعاً از MAM برمی‌دارد.

نکته: واحد MAM بصورت پیش فرض غیرفعال است و دستورالعمل ها بطور مستقیم از Flash Rom خوانده می‌شوند. این واحد می‌تواند در ۳ مُد فعال، غیرفعال و نیمه فعال کار کند.
(بعضی زمانها فعال و بعضی زمانها غیرفعال)

در LPC2000: صرفاً جهت اطلاع

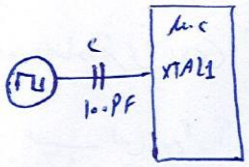
"mode control"

بیت های MODECTRL ... ۱ از رجیستر MAMCR (MAM control Register) جهت فعال ۱۰، غیرفعال، نیمه‌فعال کردن MAM استفاده می‌شود.

منابع کلاک در LPC2000: PLL

"External oscilator" کلاک

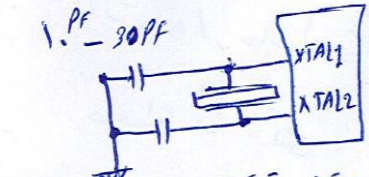
۱- اعمال پالس از طریق منبع کلاک خارجی به پایهٔ XTAL1



معمولاً 1MHZ – 50MHZ

این مُد را slave mode نیز می‌گویند. دامنهٔ سیگنال حداقل ۲۰۰mV باید باشد.

۲- استفاده از کریستال خارجی "External crystal"



معمولاً 1MHZ – 30MHZ

مقدار دقیق لازم دیتاشیت

این مُد را oscillation mode نیز می‌گویند.

"INT. PLL"

۳- استفاده از PLL داخلی

در این روش PLL داخلی فعال می‌شود و ضربی از فرکانس کلاک خارجی استفاده می‌شود.

تذکر: در اینحالت کلاک اعمالی حدوداً بین 10MHZ تا 25MHZ می‌باشد.

مانند ATOS که یکسری مدارات مقسم فرکانسی کلاک CPU تقسیمی از کلاک اعمالی بود (تقسیم بر ۱۲) و یا AVR که کلاک ورودی های کلاک CPU بود در ARM نیز می‌توان یکی PLL کلاک CPU را ضربی از کلاک ورودی انتخاب کرد.

PLL : مزیت

Electromagnetic compatibility "EMC" کاهش تأثیرات فرکانسی و افزایش کیفیت

۱- با فرکانس پایین کار می‌کنیم که باعث بهبود کریستال

۲- فرکانس CPU در حین کار قابلیت تغییر دارد.

تذکر: معمولاً PLL می‌تواند فرکانس ورودی را در عدد M ضرب کند

$$CCLK = M \times F_{osc}$$

CPU clock

M: عدد ۱-۴

حاصل باید در محدوده مجاز باشد.

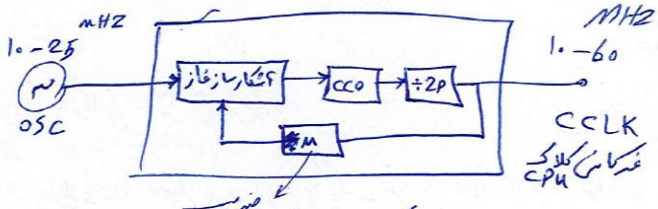
تذکر: در مدار PLL یک مقسم برای CCO "current controlled oscillator" داریم بطوریکه

(اسیلاتور کنترل جریان)

$$F_{CCO} = CCLK \times 2P$$

F_{CCO}: در بازه $156\,MHz - 320\,MHz$

$2P$: مقسم فرکانس



مثال: فرض کنید میکروکنترلر به کریستال 12 MHZ وصل شده تا CPU در فرکانس 60 MHZ کار کند مقادیر P و M را مشخص کنید

$$M = CCLK / F_{osc} = 60/12 = 5$$

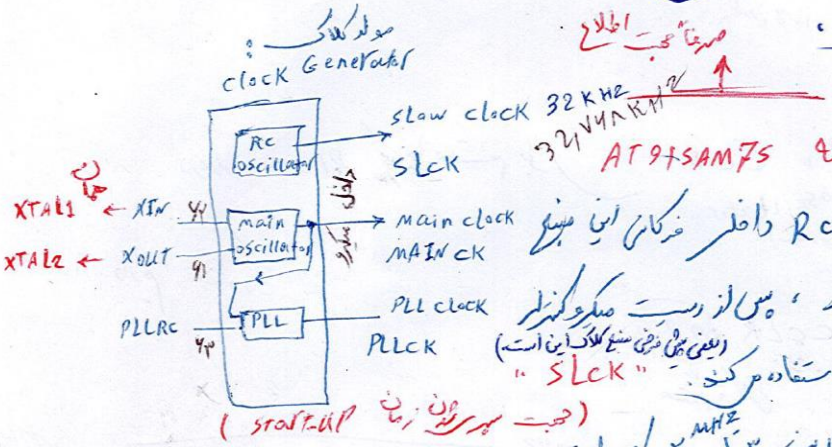
از آنجا که F_{CCO} باید بین 156 تا 320 MHZ باشد.

$$156 < 60 \times 2P < 320 \Rightarrow 1,3 < P < 2,66 \Rightarrow P = 2$$

عدد صحیح

نکته: PLL فقط بصورت نرم افزاری فعال می‌شود و برای فعال سازی آن از رجیستر PLLCON (PLL control) و برای تنظیم مقادیر M و P از طریق رجیستر PLLCFG "PLL configuration" استفاده می‌کنیم.

تذکر: جهت آگاهی از وضعیت PLL از طریق رجیستر PLLSTAT "PLL Status" استفاده و درضمن رجیستری تحت عنوان PLL FEED از تغییرات ناخواسته تنظیمات PLL جلوگیری می‌کند.



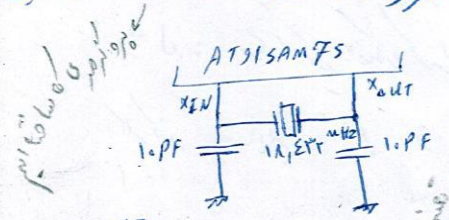
منابع کلاک در تراشه AT91SAM7S

۱- منبع اسیلاتور RC داخلی فرکانس این منبع تقریباً ۳۲KHz می‌باشد، پس از ریست میکروکنترلر مداراتش از این منبع کلاک استفاده می‌کند. (یعنی پیش فرض منبع کلاک این است) "SLCK"

۲- اسیلاتور اصلی، کریستالی بین ۳ تا ۲۰ MHz که مطابق شکل زیر بسته می‌شود و با استفاده از PLL می‌توان فرکانس آنرا چندبرابر کرد.

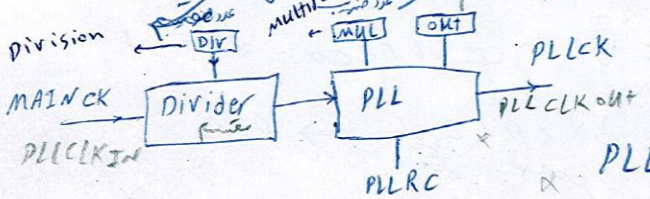
تذکر: البته می‌توان از پایه XIN، از یک منبع کلاک خارجی نیز استفاده کرد. (XOUT بی‌استفاده می‌ماند)

تذکر: برای آنکه بتوا از پورت USB، میکروکنترلر را به کمک برنامه SAM-BA برنامه‌نویسی و debug نمود، باید کریستال خارجی حتماً ۱۸,۴۳۲ MHz باشد.



حلقه قفل‌فاز "Phase Locked Loop"

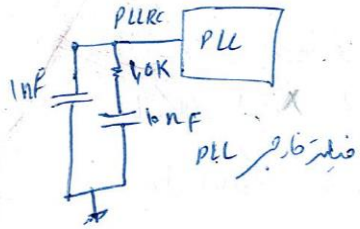
PLL: بمنظور ضرب یا تقسیم نمودن فرکانس اسیلاتور اصلی بوده و فرکانس ورودی آن بین ۲ تا ۳۲ MHz و خروجی آن می‌تواند تا ۲۲۰ MHz باشد.



$$PLLCLKOUT = PLLCLKIN * ((MUL+1)/DIV)$$

(درجه ۲ با تمرین)

تذکر: این PLL احتیاج به یک فیلتر خارجی مطابق شکل دارد و مقادیر R_2 و C مناسب با کریستال خارجی انتخاب می‌شود و برای کریستال ۱۸٫۴۳ MHz شرح ذیل است:



$PLLCLKOUT = \frac{PLLCLKIN}{DIV}$

تذکر: اگر MUL را صفر در نظر بگیریم PLL غیرفعال می‌شود.

و برای فعال‌کردن PLL کافیست MUL را غیر صفر در نظر بگیریم.

PMC:

نکته: همانطور که از نامش پیداست وظیفه آن مدیریت توان مصرفی میکرو است و با Power management clock یا Main clock میکرو می‌باشد.

هر کدام بتنهایی تمام منابع کلاک هستند، PMC فقط مدیریت توان مصرفی میکرو است.

PMC: PMC کلاک اسیلاتور اصلی و اسیلاتور RC و PLL را می‌گیرد و توسط آنها کلاک اصلی MCK (main clock) یا کلاک میکرو پردازنده «CPU» و کلاک پردازنده «periPh_eral clk» و کلاک USB و کلاک دستگاههای جانبی PCK (پردازنده کلاک) و کلاک قابل برنامه‌ریزی خروجی «PCKx» را تولید و هرکدام را جداگانه مدیریت می‌کند و یا آنها را فعال و غیرفعال می‌کند.

(ورودی PMC که می‌تواند از منابع کلاک مختلف باشد → PLL clock)

(خروجی‌های PMC می‌باشند)

programing clock

رجیسترهای مرتبط با PMC:

۱- رجیستر فعال‌کننده کلاک سیستم PMC_SCER
"PMC_system clock Enable Register"
با یک نوشتن در بیتهای این رجیستر کلاک‌های متناظر فعال می‌شود مثلاً با یک نوشتن در بیت صفرم PCK کلاک پردازنده فعال می‌شود و یا با یک شدن بیت UDP، کلاک USB برروی 48 MHz تنظیم می‌شود یا با یک شدن بیتهای PCK_0 و PCK_1 و PCK_2 کلاک قابل برنامه‌ریزی خروجی فعال می‌شود.

(نحوه مقداردهی رجیستر در AT91C_BASE_PMC: نام رجیستر → شروع رجیستر → AT91C_BASE_PMC->PMC_SCER = 0x1;)

۲- رجیستر غیرفعال‌کننده کلاک سیستم PMC_SCDR
"PMC_system clock Disable Register"
با یک شدن بیتهای متناظر (مانند بیتهای بالا) کلاک متناظر غیرفعال می‌شود.

۳- رجیستر وضعیت کلاک سیستم PMC_SCSR (صرفاً جهت اطلاع)
"PMC_system clock status Register"
نشان‌دهنده وضعیت کلاک‌ها می‌باشد، یک‌شدن بیتها (مانند بیتهای بالا) نشان‌دهنده فعال‌بودن کلاک و صفر بودنشان، نشان‌دهنده غیرفعال بودن کلاک متناظر است.

AT91C_BASE_PMC → PMC_PCER = (1<<2 | 1<<4); PIOA ADC

۴- رجیستر فعال کننده دستگاههای جانبی "PMC_PCER"

"Pmc Peripheral clock Enable Register"

با یک نوشتن در بیتهای این رجیستر کلاک دستگاه جانبی متناظر فعال می‌شود.

مانند کلاک انواع وقفه، تایمر، کانترها (صفر تا دو)، وقفه‌های خارجی (صفر و یک)، USART0 و USART1، پورت سریال، PWM، مبدل آنالوگ به دیجیتال و بیتهای

اسم دستگاههای جانبی در جدول ۱۱A

AT91C_BASE_PMC → PMC_PCER = ... (32) بین

کلاک بیتهای ورودی و خروجی دستگاههای جانبی متناظر فعال می‌شود.

۵- رجیستر غیرفعال کننده کلاک دستگاههای جانبی "PMC_PCDR"

(AT91C_ID_PIOA) (1<<2)

"Pmc Peripheral clock Disable Register"

با یک شدن بیتهای این رجیستر کلاک دستگاههای جانبی غیرفعال می‌شود.

۶- رجیستر وضعیت کلاک دستگاههای جانبی "PMC_PCSR"

"Pmc Peripheral clock Status Register"

یک بودن بیتهای این رجیستر نشانهٔ فعال بودن کلاک دستگاه جانبی متناظر می‌باشد.

صفر " " " " " " " غیرفعال " " " " " " "

۷- رجیستر اسیلاتور اصلی مولد کلاک pmc "CKGR_MOR" (انتخاب اسیلاتور)

جهت تعیین منبع کلاک اولیه (کریستال یا منبع خارجی)

"Pmc clock Generator Main oscillator Register"

AT91C_BASE_CKGR → CKGR_MOR = 0x ...

جهت تعیین منبع کلاک خارجی یا کریستال یا منبع خارجی

انتخاب کریستال

وقتی‌که MOSCEN = 1 شود، STARTUP شروع به شمارش می‌کند.

3 دسته بیت مهم این رجیستر:

MOSCEN: Main oscillator Enable

اگر این بیت صفر باشد، اسیلاتور اصلی غیرفعال است.

" " یک " " " فعال بوده و در این حالت بیت OSCBYPASS باید صفر باشد.

OSCBYPASS

اگر یک شود: در این حالت کلاک خارجی به منبع کلاک خارجی از پایهٔ XIN اعمال می‌شود و در این حالت بیت MOSCEN باید صفر باشد.

کریستال بین پایه‌های XTAL1 و XTAL2 یا XIN و XOUT

XTAL1 XTAL2 XIN XOUT

۱۳

ج) بیتهای OSCOUNT (۸ بیت حق) oscillator COUNTer

این بیتها برای تعیین Start_UP Time می‌باشد.

STartUP_Time = 8 * oscOUNT / SLCK

۸- رجیستر فرکانس کلاک اصلی مولد کلاک Pmc "CKGR_MCFR" می‌خواهد گفته شود

Pmc clock Generator Main clock Frequency Register

جهت تعیین وضعیت فرکانس خروجی

الف) بیتهای MAINF main Frequency

این بیتها تعداد کلاک خروجی اسیلاتور اصلی در مدت زمان ۱۶ کلاک آهسته "SLCK" را مشخص می‌کند.

طبق رابطه زیر

$$MAINCK = \frac{SLCK \times MAINF}{16}$$

کاربرد آنها را نمی‌دانم

ب) ۱ بیت MAINRDY جهت اطلاع

اگر این بیت صفر باشد مقادیر MAINF غیرمعتبر و اسیلاتور اصلی غیرفعال می‌باشد.

" " یک " " معتبر " " فعال "

۹- رجیستر PLL مولد کلاک Pmc "CKGR_PLLR"

" Pmc clock Generator PLL Register "

جهت تعیین و تنظیم مقادیر کلاک خروجی PLL

AT91C_BASE_CKGR->CKGR_PLLR = 0x 2903FFC

الف) بیت‌های DIV ۸ بیت

اگر مقدار این بیتها صفر باشد خروجی مقسم صفر است (خروجی خروجی)

" " یک " مقسم کنار گذاشته می‌شود.

کریستال ۱۸,۴۳۲ MHz کلاک ۴۸MHz = مقدار کلاک PLL

$$\text{مقدار کلاک PLL} = \frac{\text{کریستال}}{DIV} \times (MULL + 1)$$

تذکر: این بیتها می‌توانند مقادیری بین ۲ تا ۲۵۵ داشته باشند که در خروجی PLL مؤثرند

۱۴

ب) بیتهای PLLACOUNT OUT (2 بیت)

برای تعیین محدوده فرکانسی خروجی PLL

00 دوبیت OUT برابر 14.MHZ تا

10 دوبیت OUT برابر 200MHZ تا 14.MHZ

تذکر: در این سری میکروکنترلرها که ما گزینه فرکانسی را 48MHZ انتخاب میکنیم، این بیتها برابر 00 عمل میکنند.

ج) بیتهای MUL (11 بیتی)

$$PLL\ clockOut = PLL\ clkIN \times \left(\frac{MUL+1}{DIV}\right)$$

اگر این بیتها صفر باشند، PLL غیرفعال میشود.

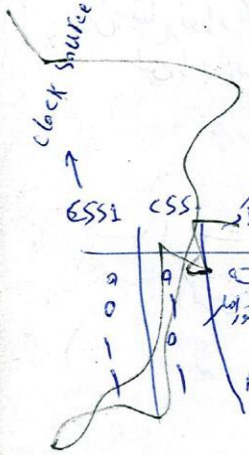
مقادیر MUL میتوانند بین 1 تا 2047 باشد که در فرکانس خروجی PLL مؤثر است.

۱۱ بیت

۹- رجیستر کلاک اصلی Pmc "PMC_MCKR"

Pmc master clock Register

جهت انتخاب منبع کلاک اصلی "MCK" (MCK های)



الف) بیت CSS (دوبیت اول)

طبق جدول منبع کلاک اصلی انتخاب میشود

انتخاب خروجی PLC بعنوان کلاک اصلی

AT91C_BASE_PMC->PMC_MCKR = 0x3;

ب) بیتهای PRES (3 بیت) press called

با انتخاب یکی از حالات این 3 بیت تقسیمی از کلاک انتخابی، به عنوان کلاک MCK های میشود

پیش فرض 0 هستند و کلاک های CPU کلاک انتخابی عبور داده

تمرین: رجیسترهای زیر را بررسی کنید و وظیفه هر کدام را بنویسید

۱) PMC_SR رجیستر وضعیت PMC status Register مدت اطلاع رسانی

الف) PMC_PCK رجیستر کلاک قابل برنامه ریزی Pmc Programmable clock Register

PMC_IDR

ب) PMC_IER* رجیستر فعال کننده وقفه Pmc Interrupt Enable Register

۱۲ PMC_IMR رجیستر ماسک وقفه Pmc interrupt mask Register