



حافظه‌ها

در میکروکنترلرهای AVR

فهرست مطالب

• معماری AVR دارای حافظه‌های زیر است:

- حافظه برنامه ← *non volatile*
- حافظه داده ← *volatile*
- یک حافظه **EEPROM** برای ذخیره داده ← *non volatile*

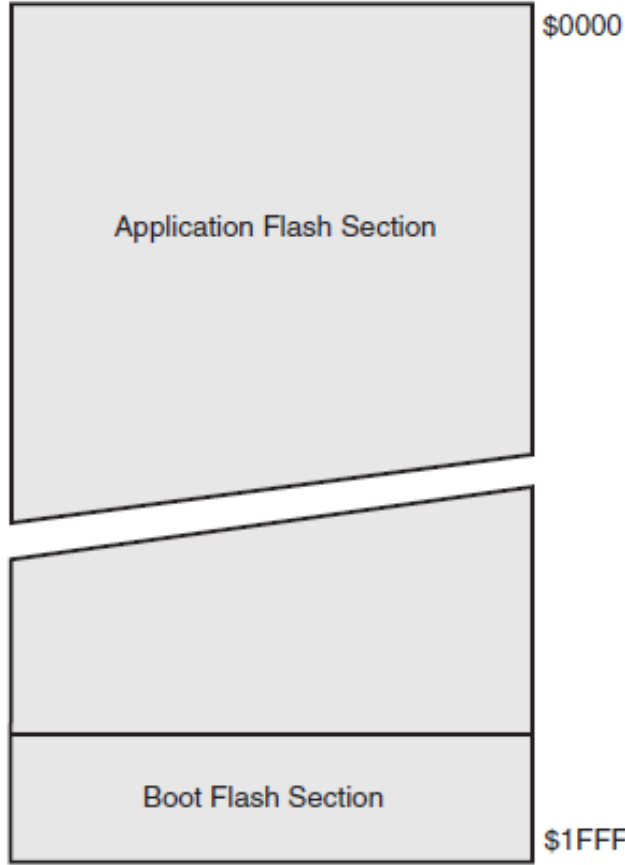
حافظه برنامه فلش قابل برنامه‌ریزی درون سیستمی

- ATmega16 دارای حافظه برنامه فلش از نوع قابل برنامه‌ریزی بصورت برنامه‌ریزی سوار بر تراشه درون سیستمی است.
- یعنی بدون نیاز به خارج کردن تراشه میکروکنترلر از مدار می‌توان برنامه را در حافظه فلش قرار داد.
- حجم این حافظه در ATmega16 برابر ۱۶ کیلو بایت می‌باشد.
- چون اکثر دستورالعمل‌های ATmega16 به تعداد ۱۶ یا ۳۲ بیت عرض دارند، لذا حافظه برنامه فلش به صورت 8K مکان حافظه ۱۶ بیتی است.
1 word (2byte)
2 word (4byte)

حافظه برنامه فلش قابل برنامه‌ریزی درون سیستمی

- حافظه فلش برنامه به دو قسمت بخش برنامه راه‌اندازی و بخش برنامه کاربردی تقسیم می‌شود.
- حافظه برنامه فلش را می‌توان تا ۱۰۰۰۰ بار برنامه‌ریزی نمود.
- برای برنامه‌ریزی حافظه برنامه فلش می‌توان از یک کابل برنامه‌ریزی استفاده نمود.

حافظہ برنامه فلش قابل برنامه ریزی درون سیستمی



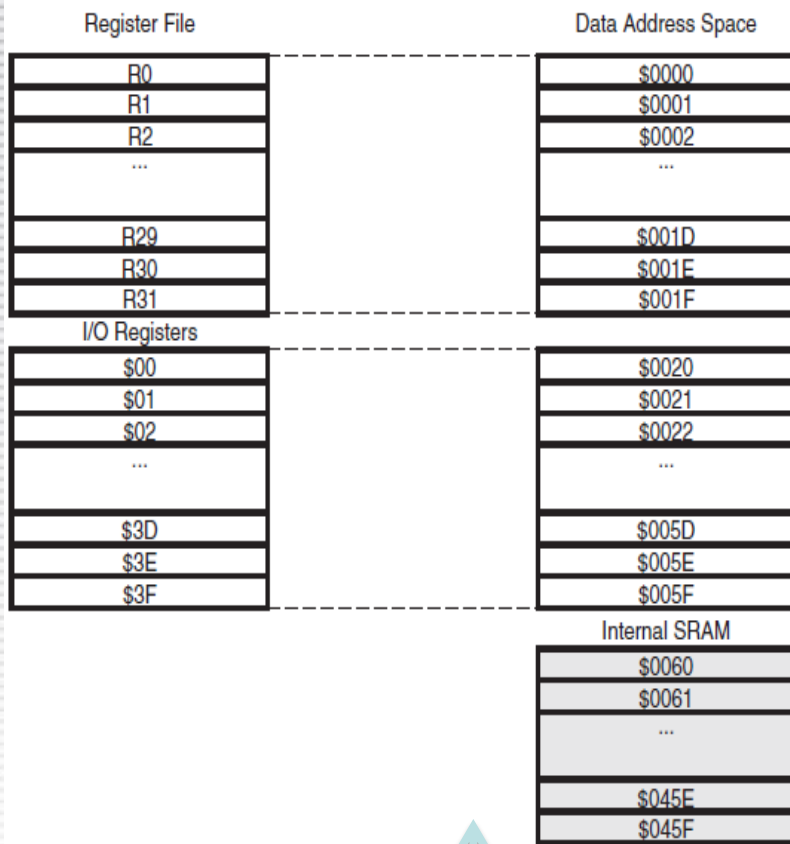
هر چه اندازه قسمت boot افزایش یابد، اندازه قسمت App کاهش می یابد

نقشه حافظه میکروکنترلر ATmega16

→ مجموع 16 KB

حافظه ها در ATmega16

حافظه داده SRAM



فضای حافظه داده و ثبات‌های عمومی همه منظوره

• شکل مقابل نشان‌دهنده سازمان حافظه **SRAM** میکروکنترلر ATmega16 می‌باشد.

• فایل ثبات، حافظه **I/O** و حافظه داده **SRAM** داخلی همگی در ۱۱۲۰ مکان پائین حافظه داده قرار دارند.

• اولین ۹۶ محل در حافظه داده فایل ثبات و حافظه **I/O** را آدرس‌دهی می‌کنند.

• ۱۰۲۴ مکان بعدی حافظه داده **SRAM** را آدرس‌دهی می‌کنند.

حافظه داده SRAM

۵ • حالت آدرس دهی متفاوت برای پوشش حافظه داده موجود است:

SRAM



(۱) حالت مستقیم

(۲) حالت غیرمستقیم با جابجایی



(۳) غیرمستقیم

Post Increment

(۴) غیرمستقیم با پیش افزایش



(۵) غیرمستقیم با پس کاهش

حافظه داده SRAM

- در فایل ثبات، ثبات های R26 تا R31 به عنوان ثبات های اشاره گر برای حالت آدرس دهی غیرمستقیم بکار می روند.
- حالت آدرس دهی مستقیم تمامی فضای حافظه داده را پوشش می دهد.
- هنگامی که از حالت های آدرس دهی غیرمستقیم با ثبات با پیش کاهش و پس افزایش اتوماتیک استفاده می کنیم، محتوای ثبات های آدرس X، Y و Z را می توان افزایش یا کاهش داد.
- ۳۲ ثبات کاری همه منظوره، ۶۴ ثبات I/O و ۱۰۲۴ بایت حافظه داده داخلی SRAM در ATmega16 همگی از طریق تمامی این حالت های آدرس دهی قابل دسترسی هستند.

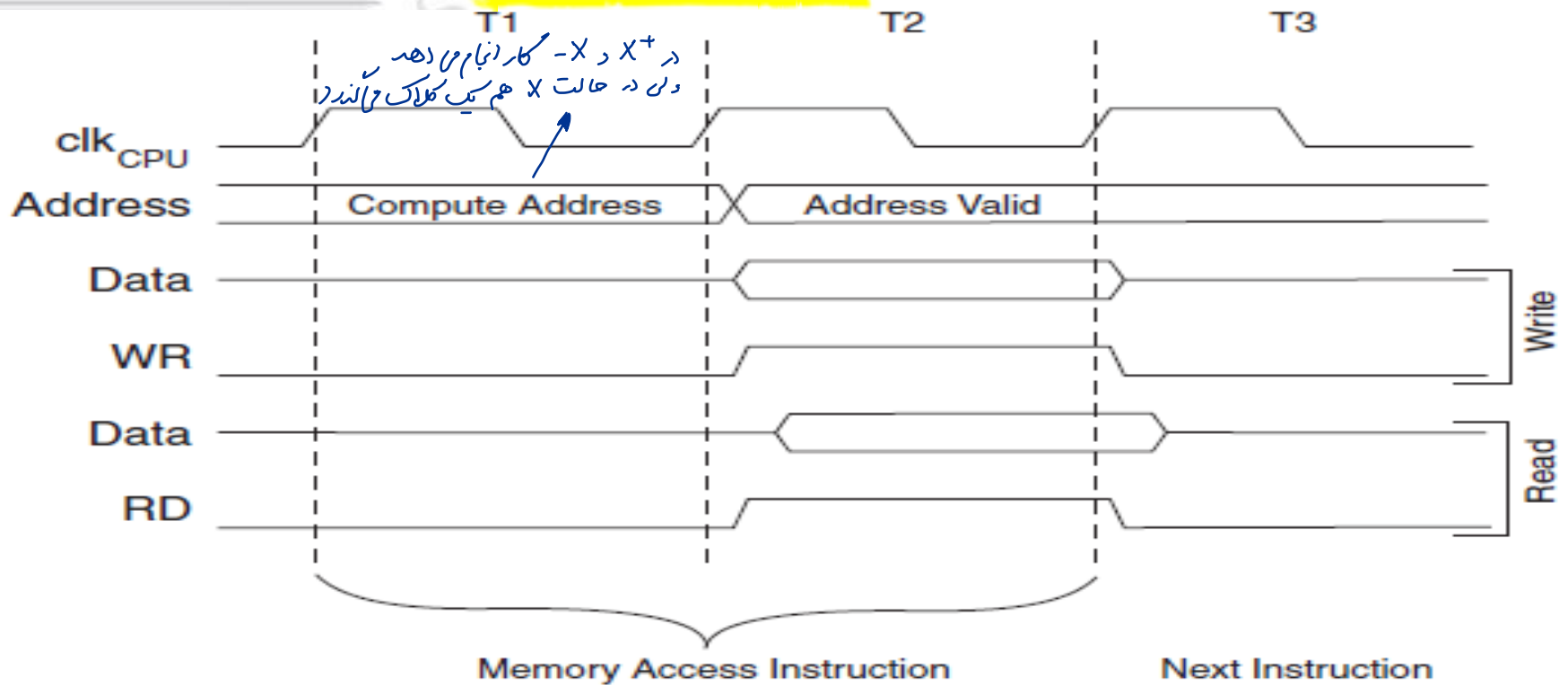
مجموعه دستورالعمل‌های میکروکنترلرهای ۸ بیتی AVR

LD ⁽²⁾	Rd, Z	Load Indirect	$Rd \leftarrow (Z)$	None	1 ⁽⁵⁾ /2 ⁽³⁾	1 ⁽³⁾ (4)
LD ⁽²⁾	Rd, Z+	Load Indirect and Post-Increment	$Rd \leftarrow (Z)$ $Z \leftarrow Z+1$	None	2 ⁽³⁾	1 ⁽³⁾ (4)
LD ⁽²⁾	Rd, -Z	Load Indirect and Pre-Decrement	$Z \leftarrow Z-1$ $Rd \leftarrow (Z)$	None	2 ⁽³⁾ /3 ⁽⁵⁾	2 ⁽³⁾ (4)
LDD ⁽¹⁾	Rd, Z+q	Load Indirect with Displacement	$Rd \leftarrow (Z + q)$	None	2 ⁽³⁾	2 ⁽³⁾ (4)
STS ⁽¹⁾	k, Rr	Store Direct to Data Space	$(k) \leftarrow Rr$	None	1 ⁽⁵⁾ /2 ⁽³⁾	2 ⁽³⁾
ST ⁽²⁾	X, Rr	Store Indirect	$(X) \leftarrow Rr$	None	1 ⁽⁵⁾ /2 ⁽³⁾	1 ⁽³⁾
ST ⁽²⁾	X+, Rr <i>Post Increment</i>	Store Indirect and Post-Increment	$(X) \leftarrow Rr$ $X \leftarrow X+1$	None	1 ⁽⁵⁾ /2 ⁽³⁾	1 ⁽³⁾
ST ⁽²⁾	-X, Rr <i>Pre decrement</i>	Store Indirect and Pre-Decrement	$X \leftarrow X-1$ $(X) \leftarrow Rr$	None	2 ⁽³⁾	2 ⁽³⁾
ST ⁽²⁾	Y, Rr	Store Indirect	$(Y) \leftarrow Rr$	None	1 ⁽⁵⁾ /2 ⁽³⁾	1 ⁽³⁾
ST ⁽²⁾	Y+, Rr	Store Indirect and Post-Increment	$(Y) \leftarrow Rr$ $Y \leftarrow Y+1$	None	1 ⁽⁵⁾ /2 ⁽³⁾	1 ⁽³⁾
ST ⁽²⁾	-Y, Rr	Store Indirect and Pre-Decrement	$Y \leftarrow Y-1$ $(Y) \leftarrow Rr$	None	2 ⁽³⁾	2 ⁽³⁾
STD ⁽¹⁾	Y+q, Rr	Store Indirect with Displacement	$(Y + q) \leftarrow Rr$	None	2 ⁽³⁾	2 ⁽³⁾
ST ⁽²⁾	Z, Rr	Store Indirect	$(Z) \leftarrow Rr$	None	1 ⁽⁵⁾ /2 ⁽³⁾	1 ⁽³⁾

حافظه داده SRAM

زمان دسترسی به حافظه داده:

- دسترسی به SRAM داده داخلی در دو چرخه پالس ساعت انجام می شود.




زمانبندی مربوط به نحوه دسترسی به حافظه SRAM

حافظه ها در ATmega16

حافظه داده EEPROM

• میکروکنترلر ATmega16 شامل ۵۱۲ بایت حافظه داده از نوع EEPROM است که بصورت یک فضای داده جداگانه سازماندهی شده است که در آن بایت های مجزا را می توان نوشت یا خواند.

• حافظه EEPROM دارای امکان نوشتن و پاک کردن به تعداد ۱۰۰۰۰ بار می باشد. 

Electric Erasable

حافظه داده EEPROM

دسترسی خواندن و نوشتن به EEPROM

- اگر کد کاربر شامل دستورالعمل‌هایی باشد که این دستورالعمل‌ها در EEPROM می‌نویسند، چندین اقدام احتیاطی باید مورد توجه قرار گیرد.
- در زمانی که منبع تغذیه شدیداً فیلتر نشده باشد، احتمال دارد که ولتاژ تغذیه VCC با روشن و خاموش شدن تغذیه به آرامی بالا و پائین برود.
- این نکته باعث می‌شود که برای لحظاتی میکروکنترلر در ولتاژی کمتر از حداقل ولتاژ تعریف شده برای سیگنال ساعت کار کند.
EEPROM برای نوشتن داخلی نیاز به منبع کلاک دارد - اینم داخلی RC کالیبره شده

حافظه داده EEPROM

دسترسی خواندن و نوشتن به EEPROM

- برای جلوگیری از نوشتن ناخواسته اطلاعات در EEPROM در اثر اختلالات ناشی از تغذیه، یک فرآیند خاص برای نوشتن باید دنبال شود.

- هنگامی که EEPROM خوانده می شود، برای مدت ۴ پالس ساعت و هنگامی که در EEPROM نوشته می شود، برای مدت ۲ پالس ساعت قبل از اجرای دستورالعمل بعدی CPU در حالت ایست قرار می گیرد.

زمان لازم برای نوشتن یک بایت در EEPROM در جدول زیر ارائه شده است:

Symbol	Number of Calibrated RC Oscillator Cycles ⁽¹⁾	Typ Programming Time
EEPROM write (from CPU)	8448	8.5 ms

حافظه داده EEPROM

ثبات‌های مورد نیاز برای کار با EEPROM:

- (1) ثبات آدرس EEPROM شامل دو بخش **EEARL** و **EEARH** 
 - (2) ثبات داده EEPROM: **EEDR** 
 - (3) ثبات کنترل EEPROM: **EECR** 
- ← مشخص می‌کند که من خواهیم بنویسیم یا بخوانیم

ثبات های EEPROM

(۱) ثبات آدرس EEPROM شامل دو بخش EEARL و EEARH

Bit	15	14	13	12	11	10	9	8	
	-	-	-	-	-	-	-	EEAR8	EEARH
	EEAR7	EEAR6	EEAR5	EEAR4	EEAR3	EEAR2	EEAR1	EEAR0	EEARL
	7	6	5	4	3	2	1	0	
Read/Write	R	R	R	R	R	R	R	R/W	
	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	
Initial Value	0	0	0	0	0	0	0	X	
	X	X	X	X	X	X	X	X	

چون 512 بایت رام خواهیم آدرس دهی کنیم به ۹ بیت نیاز داریم

ثبات های EEPROM

٢) ثبات داده EEPROM : EEDR

Bit	7	6	5	4	3	2	1	0	
	MSB							LSB	EEDR
Read/Write	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	
Initial Value	0	0	0	0	0	0	0	0	

ثبات های EEPROM

- در عملیات نوشتن در EEPROM، ثبات EEDR حاوی مقدار داده‌ای که باید در EEPROM نوشته شود می باشد.
- در این حالت، ثبات EEAR مشخص کننده آدرس مکان مورد نظر در حافظه EEPROM است.
- در عملیات خواندن از EEPROM، ثبات EEDR حاوی مقدار داده خوانده شده از EEPROM می باشد.
- در این حالت، ثبات EEAR مشخص کننده آدرس مکان مورد نظر در حافظه EEPROM است.

برای اینکه دقت بیاید باید بیت [I] یک باشد و همچنین EERIE را هم بیت حرکت
 * دقت زمان انجام خواهد شد که EEWB صفر شود

ثبات های EEPROM

(۳) ثبات کنترل EEPROM: EECR


Bit	7	6	5	4	3	2	1	0	
	-	-	-	-	EERIE	EEMWE	EEWE	EERE	EECR
Read/Write	R	R	R	R	RW	RW	RW	RW	
Initial Value	0	0	0	0	0	0	X	0	

EE Ready Interrupt Enable

Master write enable

write enable

* لازم می آید که EEWB یک شدن EEMWB است ؟ چرا برای نوشتن باید در تابلت یک بشه؟ برای مقابله با نویز

هنگامی که write تمام بشه EEWB بصورت خودکار صفر میشه  با کنترل این می توانیم بفهمیم write قبل تمام شده یا نه

* برای خواندن مقدار EEPROM بیت EERE را بیت کرده (یک می کند) و اطلاعات داخل رجیستر EEDR میاد؛ چون ALW فقط بار رجیسترهای 32

تا بی اول و ۹۶ تا می بعدیش سرکار دارد

* اگر خواندن قبلی در جریان باشد می توانیم بنویسیم؛ ولی اگر نوشتن قبلی در جریان باشد نمی توانیم دوباره بنویسیم
 * چون خواندن در یک کلاک باس تمام می شود؛ ولی نوشتن زیاد طول می کشد

ثبات های EEPROM

بیت ۲ بنام **EEMWE**: بیت راهبر نوشتن در EEPROM

- این بیت، تعیین می کند که یک کردن بیت **EEWE** موجب نوشته شدن در EEPROM گردد.

- هنگامی که **EEMWE** قبلا 0 و هم اکنون آنرا یک کنیم، یک کردن **EEWE** در ۴ پالس ساعت، موجب نوشته شدن داده در آدرس انتخاب شده می شود.

- هنگامی که بیت **EEMWE** توسط نرم افزار یک شده باشد، سخت افزار بعد از ۴ پالس ساعت آنرا صفر می کند

ثبات‌های EEPROM

بیت ۱ بنام **EEWE**: بیت فعال‌ساز نوشتن در EEPROM

- سیگنال فعال‌ساز نوشتن در EEPROM یعنی سیگنال EEWE، نقش استروب برای نوشتن در EEPROM را برعهده دارد.
- هنگامی که داده و آدرس بدرستی مشخص شوند، بیت EEWE باید یک شود تا مقداری در EEPROM نوشته شود.
- بیت EEMWE باید قبل از یک کردن EEWE مقدارش یک شود، در غیر اینصورت مقداری در EEPROM نوشته نمی‌شود.

بعد CPU میوه تو halt → EEWG یک بیت → 4 کلاک و وقت داریم → EEMWE یک بیت
به مدت 4 کلاک پالس

ثبات های EEPROM

هنگام نوشتن در EEPROM، روال زیر باید دنبال شود (رعایت ترتیب مراحل ۳ و ۴ ضروری نیست):

(۱) منتظر شوید تا بیت EEWG صفر شود

(۲) منتظر شوید تا بیت SPMEN در SPMCR صفر شود 

(۳) آدرس جدید EEPROM را در EEAR بنویسید

(۴) داده جدید EEPROM را در EEDR بنویسید

(۵) مقدار 1 منطقی را در بیت EEMWE و مقدار 0 را در بیت EEWG ثبات EECR بنویسید

(۶) در ۴ چرخه ساعت بعد از یک کردن EEMWE، بیت EEWG را یک کنید.

* EEMWE بعد از 4 کلاک پالس صفر شود (در هر حالت)

Store Program Memory Control Register (SPMCR)

Store Program Memory Enable (SPMEN)

ثبات های EEPROM

- در طی زمان نوشتن CPU در حافظه فلش، نمی توان EEPROM را برنامه ریزی نمود. ← علت مرحله ۲ صفر تپ
- نرم افزار باید قبل از پایه گذاری یک عمل نوشتن جدید در EEPROM، بررسی کند که برنامه ریزی فلش کامل شده است. ✓
- مرحله ۲ زمانی موضوعیت دارد که که نرم افزار شامل Boot loader ی است که این امکان را برای CPU فراهم می سازد که فلش را برنامه ریزی کند، در غیر این صورت مرحله ۲ می تواند حذف شود.
- هنگامی که زمان دسترسی برای نوشتن سپری شده باشد، بیت EEW E توسط سخت افزار صفر می شود.
- نرم افزار کاربر می تواند این بیت را سرکشی کند و نوشتن بایت بعدی را بعد از صفر شدن آن انجام دهد.
- هنگامی که بیت EEW E یک شده باشد، CPU برای مدت ۴ چرخه ساعت قبل از اجرای دستورالعمل بعدی دچار ایست می شود.

برنامه نوشتن در EEPROM

Assembly Code Example

```
EEPROM_write: label
; Wait for completion of previous write
sbic EECR, EEWE موجب منشر در از حلقه خارج شویم
rjmp EEPROM_write skip bit if clear
relative jump
; Set up address (r18:r17) in address register
out EEARH, r18 آدرس که در r18 است را در EEARH قرار می دهیم
out EEARL, r17
; Write data (r16) to data register
out EEDR, r16
; Write logical one to EEMWE
sbi EECR, EEMWE اگر من این در دستور به دستورانی قرار دهیم باید بیش تر از 4 بایتک با پس نشود
; Start eeprom write by setting EEWE
sbi EECR, EEWE
ret
```

C Code Example

```
void EEPROM_write(unsigned int uiAddress,
unsigned char ucData)
{
/* Wait for completion of previous write */
while((EECR & (1<<EEWE))
;
/* Set up address and data registers */
EEAR = uiAddress;
EEDR = ucData;
/* Write logical one to EEMWE */
EECR |= (1<<EEMWE);
/* Start eeprom write by setting EEWE */
EECR |= (1<<EEWE);
}
```

ثبات های EEPROM

بیت ۳ بنام **EERIE**: بیت فعال ساز آماده بودن EEPROM برای خواندن

یک کردن بیت **EERIE** موجب فعال سازی وقفه خواندن از EEPROM و نوشتن مقدار 0 آنرا غیر فعال می نماید.

ثبات های EEPROM

بیت ۰ بنام EERE: بیت فعال ساز خواندن EEPROM

- بیت EERE سیگنال استروب (فعال ساز) خواندن از EEPROM است.
- هنگامی که آدرس صحیح در ثبات EEAR مشخص شود، بیت EERE باید برای انجام عمل خواندن از EEPROM یک شود.
- دسترسی برای خواندن EEPROM، به اندازه یک دستورالعمل بطول می انجامد و داده مورد تقاضا بلافاصله فراهم خواهد شد. ✓

ثبات‌های EEPROM

- هنگامی که EEPROM خوانده می‌شود، CPU قبل از اجرای دستورالعمل بعدی به مدت ۴ پالس ساعت halt می‌شود.
- کاربر باید بیت EEWL را قبل از شروع عملیات خواندن سرکشی و آنرا صفر کند.
- اگر یک عملیات نوشتن در حال انجام باشد، هیچ یک از عملیات خواندن EEPROM و تغییر ثبات EEAR امکان‌پذیر نیست.
- نوسان‌ساز کالیبره شده برای تنظیم زمان دسترسی‌ها به کار می‌رود

EEPROM برنامه خواندن از حافظه

Assembly Code Example

EEPROM_read:

; Wait for completion of previous write

sbic EECR, EEWE

rjmp EEPROM_read

; Set up address (r18:r17) in address register

out EEARH, r18

out EEARL, r17

; Start eeprom read by writing EERE

sbi EECR, EERE

; Read data from data register

in r16, EEDR

ret

C Code Example

unsigned char EEPROM_read(**unsigned int** uiAddress)

{

/ Wait for completion of previous write */*

while(EECR & (1<<EEWE))

;

/ Set up address register */*

EEAR = uiAddress;

/ Start eeprom read by writing EERE */*

EECR |= (1<<EERE);

/ Return data from data register */*

return EEDR;

}

حافظه ورودی/خروجی I/O

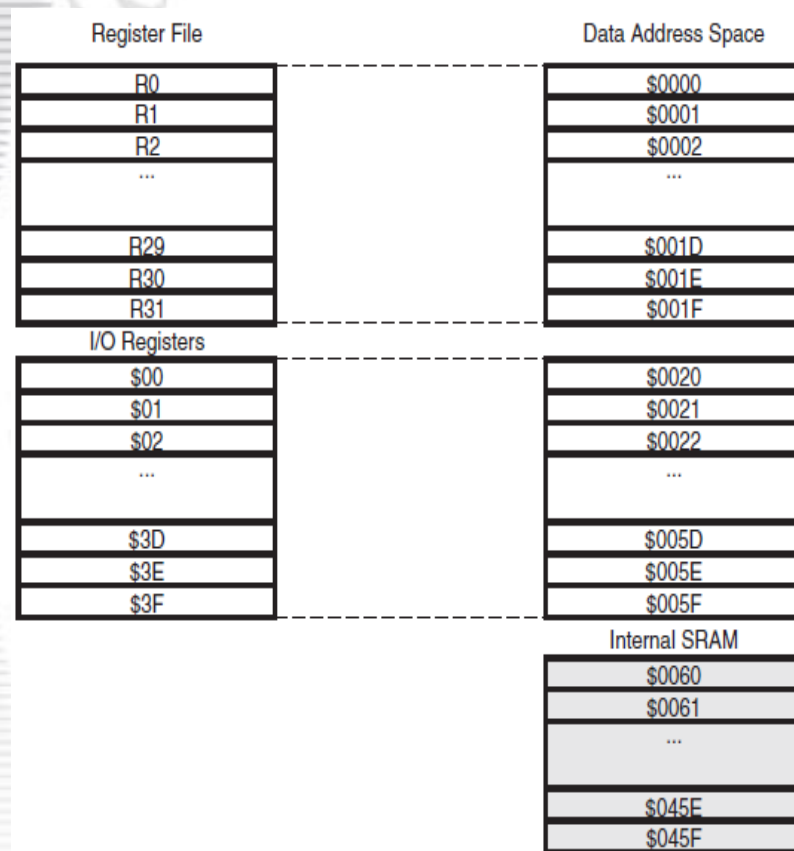
- ثبات های I/O در فاصله آدرس فضای حافظه I/O شامل ۶۴ بیت آدرس می شود که برای عملیات های مربوط به ماژول های داخلی (شامل عملیات های ورودی-خروجی، SPI، USART، زمان سنج، شمارنده، مبدل آنالوگ به رقمی و مانند آن) استفاده می شود.

- حافظه I/O به دو روش می تواند مورد دسترسی قرار گیرد:

SRAM: به عنوان SRAM، محدوده آدرس ها از \$20 شروع و تا \$5F ادامه می یابد

ثبات های I/O: به عنوان ثبات I/O، آدرس ها در محدوده \$00 تا \$3F قرار می گیرند.

حافظه ورودی/خروجی I/O



حافظه ورودی/خروجی I/O

- حافظه I/O می‌تواند مستقیماً یا به عنوان محل‌هایی از فضای داده که در ادامه مجموعه ثبات‌های عمومی همه منظوره قرار می‌گیرند و شامل آدرس‌های \$20 تا \$5F می‌شوند، آدرس‌دهی شود.
- تمامی ورودی/خروجی‌های ATmega16 در فضای I/O قرار دارند.
- مکان‌های I/O توسط دستورالعمل‌های **IN** و **OUT** قابل دسترسی هستند.
- این دستورالعمل‌ها داده را بین فضای **I/O** و ۳۲ ثبات عمومی کاری انتقال می‌دهند.

حافظه ورودی/خروجی I/O

• ثبات های I/O موجود در آدرس های \$00 تا \$1F توسط دستورالعمل های SBI و CBI مستقیماً بصورت بیتی قابل دسترسی هستند.

• هنگامی که دستورات خاص I/O یعنی IN و OUT استفاده شوند، باید آدرس های I/O موجود در محدوده \$00 تا \$3F استفاده شوند.

بیت شماره ۱ (از آدرس ۰ تا ۳۱)

SBI	A, b	Set Bit in I/O Register	$I/O(A, b) \leftarrow 1$
CBI	A, b	Clear Bit in I/O Register	$I/O(A, b) \leftarrow 0$

• در این ثبات ها، مقدار بیت ها بطور جداگانه می توانند توسط دستورالعمل های SBIS و SBIC بررسی شوند.

بیشتر دارد که دستور بعدی ۲ بایت است یا ۳ بایت

SBIC	A, b	Skip if Bit in I/O Register Cleared	if $(I/O(A, b) = 0)$ PC \leftarrow PC + 2 or 3
SBIS	A, b	Skip if Bit in I/O Register Set	If $(I/O(A, b) = 1)$ PC \leftarrow PC + 2 or 3

• لیکن اگر ثبات های I/O به عنوان فضای داده استفاده شوند، باید مقدار \$20 را به محدوده آدرس های \$00 تا \$3F اضافه و از دستورالعمل های LD و ST استفاده نمود.