روش استفاده از حافظه Flash به عنوان حافظه EEPROM

مهداد قاسمیان مشکانی

microc.blogfa.com

۱- مقدمه ۳
۲- تفاوت های اصلی حافظه های EEPROM با Flash داخلی ۳
۳- پیاده سازی EEPROM مجازی ۵
۳-۱- اصول ۵
۳-۱-۱- مثال ۶
۳-۱-۲- شرح نرم افزار EEPROM ۷
۳-۱-۲-۱- شرح توابع موجود در eeprom.c ۸
۳-۱-۲-۲- شرح فایل eeprom.h ۱۰
۳-۱-۲-۳- شرح فایل main.c ۱۱
۴- جنبه های برنامه کاربر ۱۱
۴-۱- مدیریت تک تک داده ها ۱۱
۴-۱-۱- برنامه ریزی بر مبنای word-by-word ۱۲
۴-۱-۲- برنامه ریزی بر مبنای byte-by-byte ۱۲
۴-۲- پوشش صحیح (بهبود استقامت حافظه Falsh) ۱۲
۴-۲-۱- مثال اجرای الگوریتم پوشش صحیح ۱۲
۴-۳- بازیابی هدر صفحه ها پس از قطع برق ۱۳
۴-۴- ظرفیت چرخه ۱۴
۵- منابع ۱۶

۱- مقدمه

اکثر دستگاه ها به EEPROM (حافظه فقط خواندنی با قابلیت برنامه ریزی و پاک کردن به صورت الکتریکال) برای نگه داری داده های غیر فرار نیاز دارند. برای کاهش قیمت، در برخی از میکرو کنترلر های خانواده ARM، از EEPROM استفاده نشده است. در عوض، در آن ها میتوان به کمک حافظه Flash، حافظه EEPROM را شبیه سازی کرد.

این متن تلاش می کند تا تفاوت های بین حافظه های EEPROM و Flash داخلی را بیان کرده و نحوه پیاده سازی حافظه EEPROM مجازی بر روی حافظه Flash داخلی را شرح دهد. در ادامه توضیحاتی پیرامون مثال کاربردی همراه این فایل و چگونگی کارکرد کتابخانه ای عرضه شده برای این منظور در خانواده ای میکرو های STM32F10x ارائه خواهد شد.

۲- تفاوت های اصلی حافظه های EEPROM با Flash داخلی

در دستگاه هایی که به ذخیره دائمی دیتا به صورت byte یا یک word نیاز است ، حافظه های EEPROM از انتخاب های اصلی می باشد.

از طرف دیگر ، میکرو کنترلر های استفاده شده در این دستگاه ها اکثرا بر پایه حافظه های Flash بنا شده اند. در اکثر میکرو های که قبلا موجود بودند و از هسته ARM استفاده نمی کردند ، حافظه EEPROM به صورت داخلی وجود داشت اما امروزه در اکثر میکرو کنترلرهای با هسته ARM ، برای کاهش قسمت های داخلی آی سی ، ذخیره فضای سیلیکون و کاهش هزینه سیستم ، از حافظه EEPROM داخلی استفاده نشده است و این میکرو ها دارای EEPROM نمی باشند.

در این گونه میکروکنترلر ها برای کاربرد هایی که نیاز به ذخیره داده در زمان اجرای برنامه است دو راه حل موجود می باشد.

- راه اول : استفاده از حافظه EEPROM خارجی که اکثرا به صورت سریال به میکرو متصل می شود.
- راه دوم : استفاده از حافظه Flash داخلی میکرو. منظور استفاده از قسمتی از حافظه فلش که مورد نیاز نیست و در آن کد قرار ندارد. این راه را اصطلاحا "EEPROM مجازی" می نامند.

در حافظه های FLASH برای نوشتن در حافظه باید حتما ابتدا همه حافظه و یا بلوکی که قصد نوشتن در آن را داریم را پاک کرده و بعد داده جدید در آن نوشته شود ، اما در حافظه های EEPROM اینچنین نیست. و می توان

بدون نیاز به پاک کردن اطلاعات قبلی ، اطلاعات جدید را روی آن نوشت. از این رو یک بسته نرم افزاری برای مدیریت نوشتن داده در حافظه های Flash داخلی نیاز است.

تفاوت های اصلی مابین حافظه Flash داخلی با حافظه های EEPROM خارجی که به صورت سریال به میکرو متصل می شوند در جدول زیر بیان شده است . توجه شود که این تفاوت ها برای همه میکروکنترلر های ARM است و مختص خانواده STM32F10xxx نمی باشد.

جدول ۱ : تفاوت های مابین FLASH داخلی و EEPROM خارجی

ویژگی	EEPROM خارجی	استفاده از Flash داخلی برای ایجاد یک EEPROM مجازی
زمان نوشتن	چند میلی ثانیه بایت های تصادفی : ۵ تا ۱۰ میلی ثانیه صفحه : صد میکرو ثانیه برای هر Word (۵ تا ۱۰ میلی ثانیه برای هر صفحه)	زمان نوشتن یک Word : ۲۰ میکرو ثانیه
زمان پاک کردن	-	صفحه/ جرم : ۲۰ میلی ثانیه
طریقه نوشتن	یک لحظه است، کاری به CPU ندارد فقط کافی است به طور صحیح تغذیه آن متصل شود	موقع شروع ، به وضعیت CPU بستگی دارد: اگر در زمان نوشتن ، CPU ریست شود ، عملیات نوشتن با شکست روبرو می شود، حتی اگر تغذیه هم متصل باشد.
دسترسی برای خواندن	سریال : صد میکرو ثانیه یک Word تصادفی : ۹۲ میکرو ثانیه صفحه : ۲۲,۵ میکرو ثانیه برای هر بایت	موازی : صد نانو ثانیه سیکل مصرفی CPU برای هر Word خیلی کم است زمان دسترسی : ۳۵ نانو ثانیه
چرخه برنامه ریزی/ پاک کردن	از ۱۰ تا ۱۰۰۰ کیلو سیکل	از ۱۰ تا ۱۰۰ کیلو سیکل (با استفاده از صفحات بیشتر می توان چرخه را بیشتر کرد)

۳- پیاده سازی EEPROM مجازی

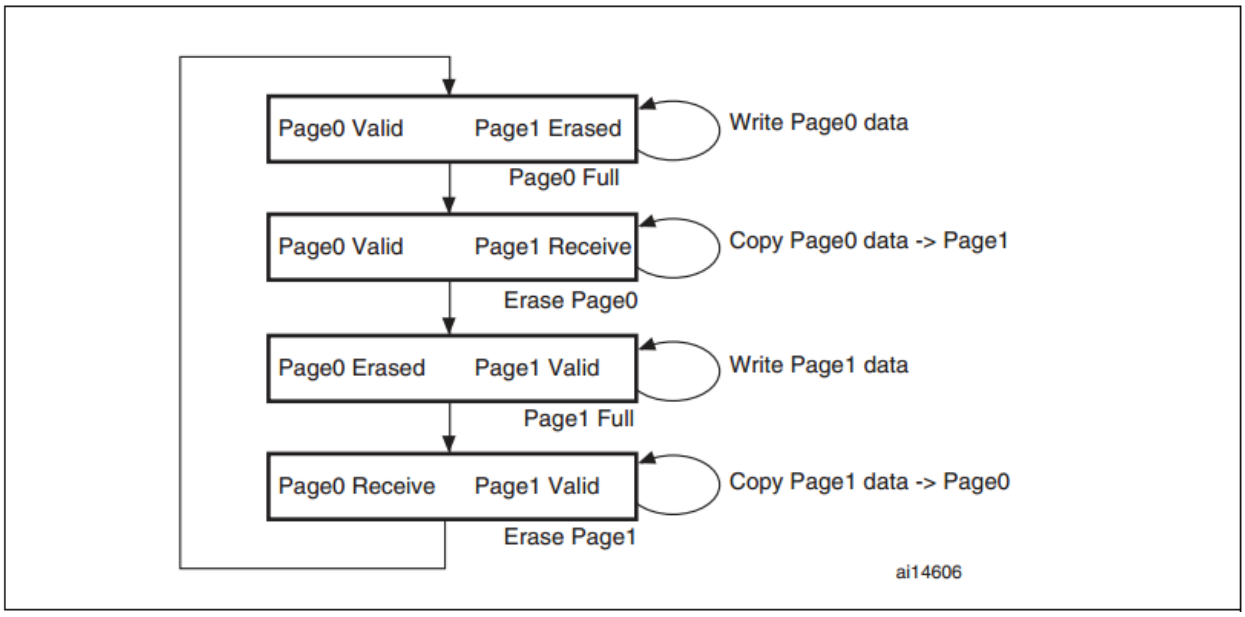
۳-۱- اصول

EEPROM مجازی را بر حسب محدودیت های حافظه FLASH و همچنین نیاز های محصول ، به روش های مختلف می توان اجرا کرد. روشی که در زیر جزئیات آن بررسی می شود حداقل به دو صفحه از حافظه فلش با اندازه مساوی نیاز دارد. یکی از صفحه ها به صورت پیش فرض پاک می شود ، و برنامه ریزی بایت به بایت را عرضه می کند. یک فیلد که ۱۶ بیت از هر صفحه را اشغال می کند برای مشخص کردن وضعیت صفحه در نظر گرفته شده است.

مکان این فیلد وضعیت در ابتدای صفحه ، یعنی همان آدرس پایه صفحه قرار دارد.

هر صفحه می تواند ۳ وضعیت داشته باشد :

- ERASED : صفحه خالی است
- RECEIVE_DATA : صفحه در حال دریافت داده از یک صفحه دیگر که پر شده می باشد
- VALID_PAGE : صفحه شامل داده های معتبر می باشد و این وضعیت تا زمانی که داده های معتبر به یک صفحه دیگری (که از قبل پاک شده باشد) منتقل نشود تغییری نخواهد کرد.

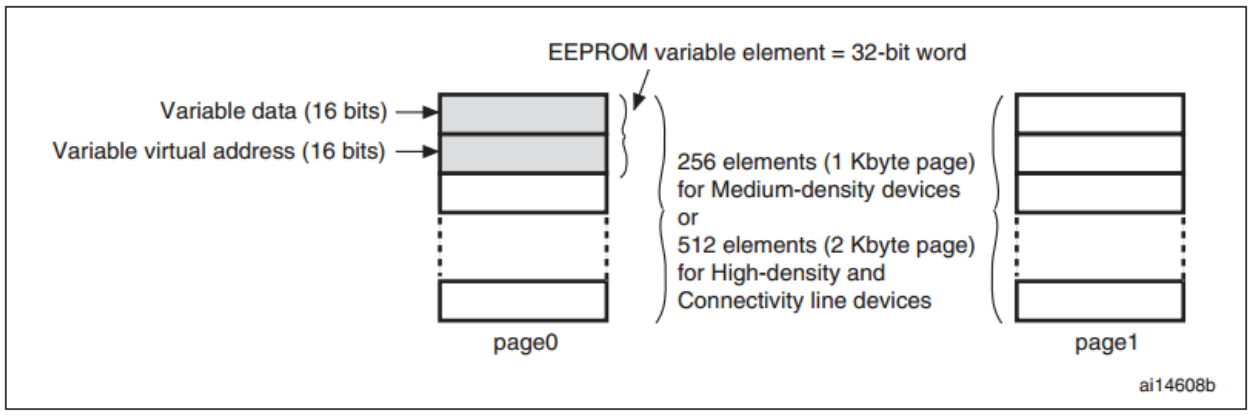


شکل ۱ سوئیچ مقدار هدر وضیعت مابین صفحه ۰ و صفحه ۱

در اصل ، وقتی که از این روش استفاده میکنیم ، کاربر اطلاعی از تغییر وضعیت صفحه ها ندارد.

همان طور که در این متن توضیح داده خواهد شد ، به کمک دو صفحه از حافظه FLASH میتوان حافظه EEPROM را شبیه سازی کرد.

برای هر متغییر یک مقدار و یک آدرس در نظر گرفته می شود. که هر دوتای آن ها در حافظه فلش ذخیره می شود و به کمک آدرس مجازی که برای آن متغییر در نظر گرفته شده است می توان به مقدار آن دسترسی پیدا کرد و آن را خواند یا تغییر داد. طول آدرس و مقدار متغییر ۱۶ بیت در نظر گرفته شده است. وقتی که دیتا تغییر داده می شود ، دیتای تغییر داده شده با نزدیکترین آدرس مربوطه اش در یک مکان جدید از حافظه فلش ذخیره می شوند. در هنگام بازیابی داده آخرین دیتایی که در فلش ذخیره شده است برگردانده می شود.

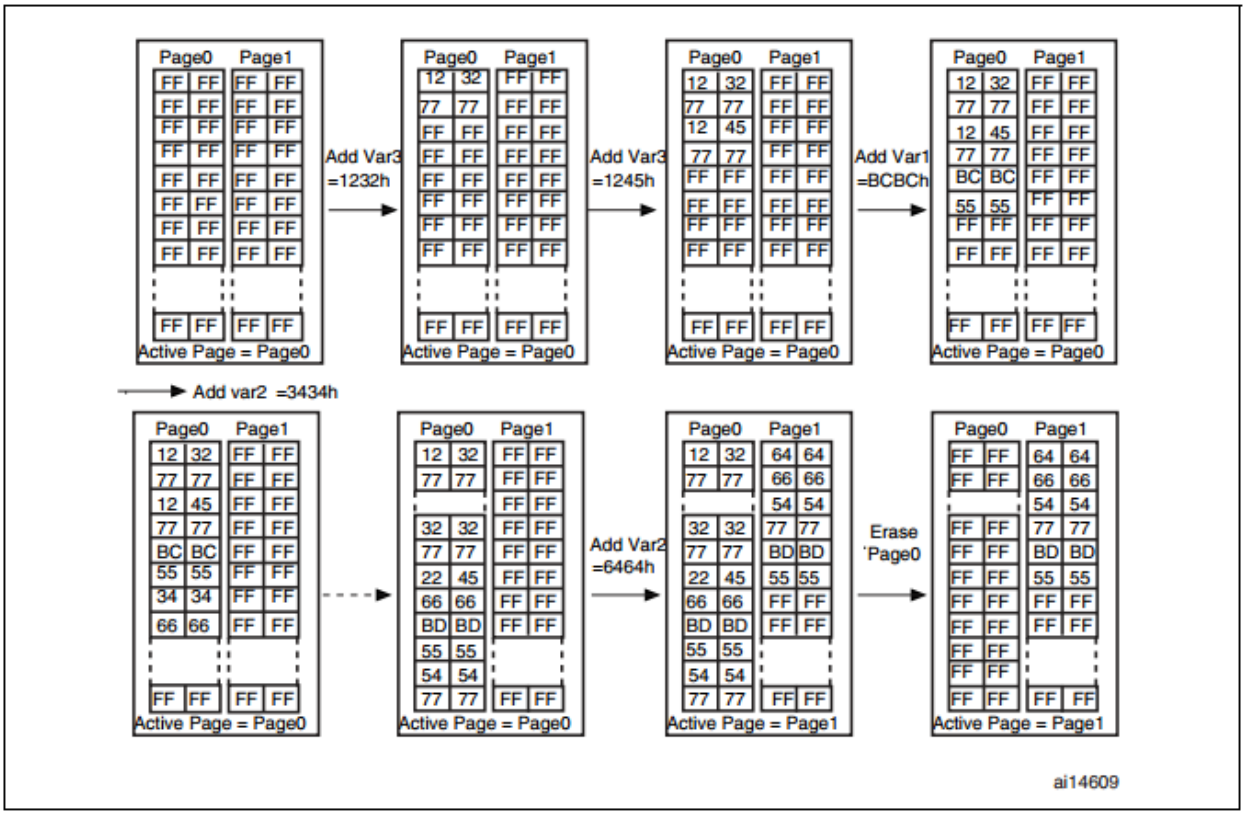


شکل ۲ مدل چیدمان متغییر های EEPROM

۳-۱-۱- مثال

در شکل ۳ طریقه مدیریت حافظه را برای ایجاد ۳ متغییر EEPROM نشان می دهد (Var1 ، Var2 ، Var3) که از آدرس های مجازی زیر استفاده میکند :

Var1 : 5555h ، Var2 : 6666h و Var3 : 7777h



شکل ۳ چگونگی بروزرسانی دیتا

۳-۱-۲- شرح نرم افزار EEPROM

همراه با این متن، یک مثال کاربردی در مورد چگونگی راه اندازی و استفاده از EEPROM مجازی ارائه شده است. این قسمت، فایل ها و توابع موجود در این مثال و نحوه استفاده از آن ها را توضیح می دهد.

در مثال ارائه شده از درایور کنترل حافظه Flash ارائه شده توسط شرکت STmicroelectronic استفاده شده است. برنامه در محیط Keil و برای میکروکنترولر های خانواده STM32F10xxx نوشته شده است. شما می توانید با تغییر درایور کنترل حافظه Flash و یا محیط نرم افزاری، از کتابخانه نوشته شده برای EEPROM مجازی ، در میکروکنترلر مورد نظر خود استفاده کنید.

همان طور که در قبل توضیح داده شد، برای ایجاد هر متغییر در EEPROM مجازی ، یک آدرس مجازی نیاز است. در مثال ارائه شده در فایل main.c ، سه متغییر در آرایه به نام VirADDVarTab تعریف شده است. این سه متغییر برای تست برنامه تعریف شده است.

پروژه شامل سه فایل اصلی است (به علاوه فایل های درایور حافظه FLASH و دیگر فایل های پروژه) :

- eeprom.c : شامل کد های C و توابع مورد نیاز جهت اجرای EEPROM است که شامل توابع زیر می باشد:

EE_Init()

EE_Format()

EE_FindValidPage()

EE_VerifyPageFullWriteVariable()

EE_ReadVariable()

EE_PageTransfer()

EE_WriteVariable()

- eeprom.h : این فایل شامل معرف های توابع موجود در فایل eeprom.c و تعدادی تعاریف می باشد.
- main.c : برنامه ای کاربردی در این فایل نوشته شده است و مثال هایی برای روش خواندن و نوشتن در حافظه EEPROM مجازی در این فایل می باشد.

۳-۱-۲-۱- شرح توابع موجود در eeprom.c

توابع موجود در فایل eeprom.c ، که برای ایجاد EEPROM مجازی نیاز است در زیر توضیح داده می شود :

- EE_Init()

در هنگام به روزرسانی ، پاک کردن و یا انتقال داده ها در صفحه ها اگر مشکل تغذیه ایجاد شود و افت ولتاژی رخ دهد ، ممکن است هدر فایلی که وضعیت هر صفحه را مشخص میکند خراب شود. در این مواقع ، تابع EE_Init() برای یافتن وضعیت صحیح صفحه ، دیتابیس را جسجتو میکند تا وضعیت صحیح را بازیابی کند. این تابع باید قبل از دسترسی به دیتابیس و بعد از هر افت ولتاژ (روشن شدن سیستم) اجرا شود. هیچ پارامتری برنمی گرداند. پروسه در جدول ۲ شرح داده شده است.

- EE_Format()

این تابع صفحه ۰ و صفحه ۱ را پاک میکند و در هدر صفحه ۰ ، مقدار VALID_PAGE می نویسد.

- EE_FindValidPage()

این تابع هدر هر دو صفه را میخواند و شماره هر کدام از صفحه ها که اون صفحه معتبر باشد را برمیگرداند.

- EE_VerifyPageFullWriteVariable()

این تابع پروسه نوشتن یک متغییر را پیاده سازی میکند. اگر متغییر از قبل وجود داشته باشد و حاوی مقداری در حافظه باشد، مقدار آن بروزرسانی می شود و در غیر این صورت برای اولین بار ایجاد می شود. این عملیات شامل پیدا کردن اولین مکان خالی در صفحه فعال، و پر کردن آن با آدرس مجازی و داده پاس شده به تابع می باشد. این تابع برای پیدا کردن اولین مکان خالی در صفحه، صفحه را از آخر شروع به جستجو میکند. در حالتی که صفحه فعال پر باشد ، مقدار PAGE_FULL برگردانده می شود. این تابع از پارامتر های زیر استفاده میکند :

آدرس مجازی : که در مثال ما می تواند یکی از مقادیر متغییر های Var1 ، Var2 ، Var3 باشد.

دیتا : مقدار متغییر که باید ذخیره شود.

این تابع در صورت موفقیت ، مقدار FLASH_COMPLETE را برمیگرداند. در صورتی که فضای کافی برای بروزرسانی متغییر نباشد و یا خطایی رخ دهد ، مقدار PAGE_FULL را برمیگرداند.

- EE_ReadVariable()

این تابع مقدار مربوط به آدرسی که به عنوان پارامتر به تابع ارسال شده را بر میگرداند. فقط آخرین مقدار بروزرسانی شده خوانده می شود. تابع داخل یک حلقه می شود و وقتی که به آخرین مقدار آن متغییر مورد نظر رسید از حلقه خارج می شود. اگر هیچ موردی پیدا نکرد، متغییر ReadStatus با مقدار "۱" برگردانده می شود ، در غیر این صورت مقدار آن را صفر کرده و مقدار پیدا شده را در متغییر Read_Data قرار میدهد.

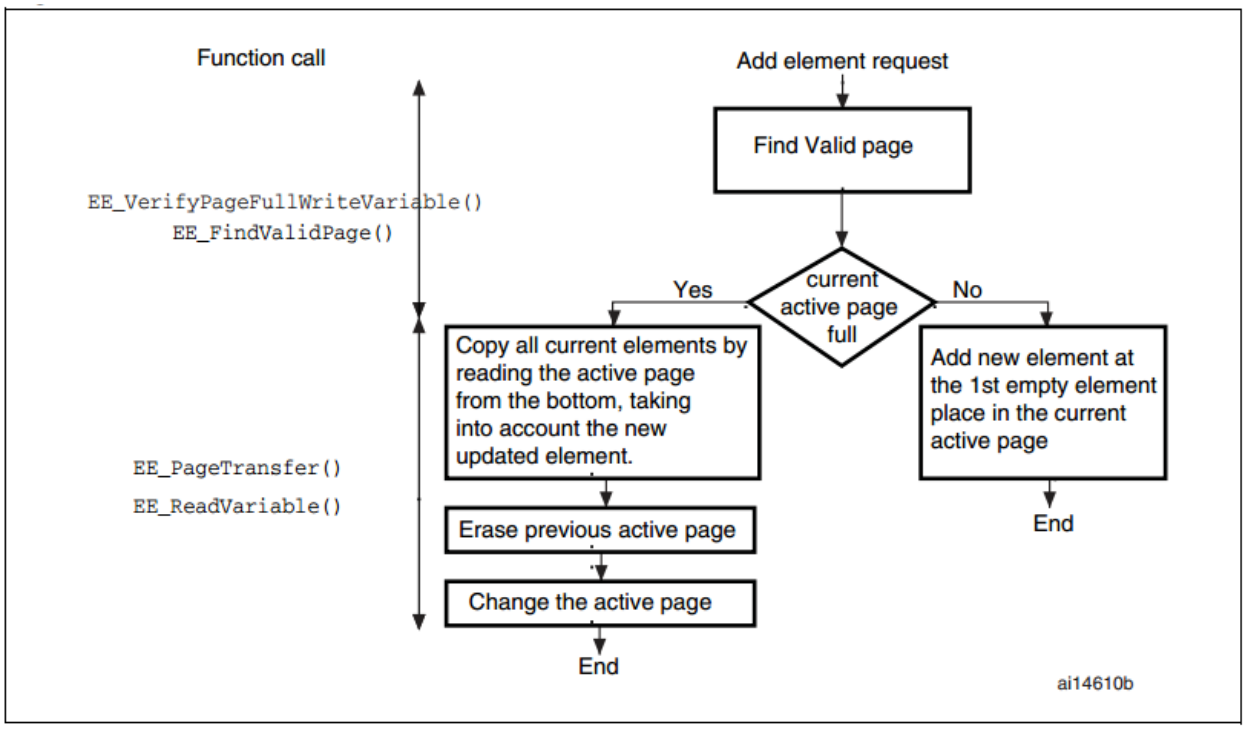
- EE_PageTransfer()

این تابع جدیدترین دیتا (آخرین بروزرسانی دیتا) را از صفحه ای که پر شده به یک صفحه خالی منتقل می کند. در ابتدا، یک صفحه فعال را پیدا میکند و سپس هدر این صفحه را به حالت RECEIVE_DATA تغییر میدهد. وقتی که ارسال دیتا کامل شد، هدر این صفحه جدید را به حالت VALID_PAGE تغییر میدهد، صفحه قدیمی پاک می شود و در هدرش ERASED نوشته می شود.

- EE_WriteVariable()

این تابع توسط برنامه کاربر، برای بروزرسانی یک متغییر صدا زده می شود. این تابع از توابع EE_VerifyPageFullWriteVariable() و EE_PageTransfer() استفاده می کند.

شکل ۴ پروسه بروزرسانی یک متغییر را در EEPROM نشان میدهد.



شکل ۴ فلوچارت نوشتن متغییر

۳-۱-۲-۲- شرح فایل eeprom.h

تعدادی از تعاریف موجود در این فایل ، بر حسب هر پروژه ، نیاز به تغییر و تنظیم دارند. این قسمت، این تعاریف و نحوه تغییر آن ها را شرح میدهد. توجه شود که مابقی تعاریف در این فایل تغییر داده نشود.

- EEPROM_START_ADDRESS

این تعریف بیان گر آدرس شروع ایجاد حافظه EEPROM مجازی در حافظه Flash می باشد. مقدار این آدرس باید حتما بعد از آخرین مکان مورد استفاده برای کد برنامه باشد.

- NumbOfVar

این تعریف بیان گر تعداد خانه های EEPROM مجازی می باشد.

۳-۱-۲-۲-۱- بلوک بندی حافظه Flash

مقدار حافظه Flash در میکروکنترلر های خانواده STM32F10xxx ، از ۲۵۶ تا ۵۱۲ کیلو بایت، بر حسب نوع میکروکنترلر می باشد. آدرس تخصیص داده شده به حافظه Flash در باس آدرس این خانواده از آدرس 0x08000000 شروع و به 0x0807FFFF ختم می شود. حافظه Flash این خانواده دارای ۲۵۶ صفحه می باشد که مقدار صفحه ها ۱ کیلو بایت (برای قطعات تراکم متوسط) یا ۲ کیلو بایت (برای قطعات تراکم بالا) می باشد.

۳-۱-۲-۳- شرح فایل main.c

برنامه کاربردی در این فایل نوشته شده است. این قسمت خطوط برنامه موجود در این فایل را شرح می دهد.

- VirtAddVarTab[NumbOfVar] ={0x5555, 0x6666, 0x7777}

متغییر VirtAddVarTab یک آرایه به تعداد خانه های مورد نیاز در EEPROM مجازی می باشد. این آرایه نگه دارنده آدرس های مجازی در نظر گرفته شده برای خانه های EEPROM مجازی می باشد. در این برنامه سه متغییر به آدرس های مجازی 0x5555 و 0x6666 و 0x7777 تعریف شده اند. شما می توانید برای پروژه خود آدرس ها را تغییر دهید.

- FLASH_Unlock()

این تابع، از توابع درایور حافظه Flash می باشد که برای باز کردن قفل حافظه استفاده می شود. برای برنامه ریزی و پاک کردن حافظه Flash نیاز است که قبلا قفل آن باز شود. در صورتی که شما در پروژه خود از خانواده دیگری استفاده می کنید، نیاز است که تابع باز کننده قفل حافظه Flash آن خانواده را بکار گیرید.

- EE_Init()

پس از روشن شدن و بالا آمدن سیستم نیاز است که این تابع صدا زده شود. شرح این تابع در قبل موجود است.

- EE_WriteVariable(VirtAddVarTab[1], 1500)

این تابع از توابع اصلی کاربر می باشد. کاربر می تواند به کمک این تابع در حافظه EEPROM مجازی مقدار دلخواه خود را بنویسد. در این خط از برنامه مقدار ۱۵۰۰ را در خانه ۱ حافظه EEPROM مجازی می نویسد(آدرس مجازی این خانه 0x5555 است).

- EE_ReadVariable(VirtAddVarTab[1], &c)

این تابع از توابع اصلی کاربر می باشد. این تابع برای خواندن از EEPROM مجازی بکار می رود. در این خط از برنامه، خانه ۱ حافظه EEPROM مجازی خوانده و در متغییر c قرار داده شده است.

۴- جنبه های برنامه کاربر

این قسمت سعی دارد توضیح دهد که چگونه می توان بر محدودیت های نرم افزاری این روش غلبه کرد، تا بتوان برای کاربرد های مختلف از این تکنیک استفاده کرد.

۴-۱- مدیریت تک تک داده ها

EEPROM مجازی را می توان در کاربرد های که نیاز به ذخیره دیتا به صورت غیر فرار باشد مورد استفاده قرار داد. دیتا می تواند در قالب های byte ، half-word و یا word باشد. در اصل به دو عامل نیاز کاربر و ساختار حافظه Flash بستگی دارد. در مورد محدودیت های ساختار حافظه می توان به طول ذخیره شده ، دسترسی نوشتن و ... اشاره کرد.

حافظه Flash در خانواده چیپ STM32F10xxx اجازه دسترسی و نوشتن داده در حافظه را به صورت half-word، ۱۶ بیتی را میدهد. که به کمک تکنیک های برنامه نویسی میتوان دیتا را به صورت byte یا word نیز در در حافظه ذخیره کرد.

۴-۱-۱- برنامه ریزی بر مبنای word-by-word

درایور حافظه Flash یک تابعی در اختیار ما قرار می دهد که به کمک آن می توان یک دیتا ۳۲ بیتی "VarData" را در آدرس "VarAddress" از حافظه نوشت: FLASH_ProgramWord(VarAddress,Vardata)

که این تابع، تمام یک word را میتواند در آدرسی مشخصی از حافظه Falsh داخلی بنویسد.

۴-۱-۲- برنامه ریزی بر مبنای byte-by-byte

نوشتن بر مبنای بایت، امکان ایجاد تعداد بیشتری متغییر را به کاربر می دهد. در حالی که از کارایی سیستم می کاهد. توسط تابع FLASH_ProgramHalfWord() ، می توان آدرس مجازی و مقدار دیتا را در کنار یکدیگر در یک مکان از half word نوشت.

۴-۲- پوشش صحیح (بهبود استقامت حافظه Falsh)

حافظه Flash چیپ های سری STM32F10xxx ، نوشتن و پاک کردن هر صفحه را تا ۱۰۰۰۰ بار تضمین میکند.

برای کاربرد های با نوشتن زیاد نیاز است بیش از دو صفحه (۳ یا ۴) از حافظه Flash را برای کار EEPROM مجازی اختصاص داد. پس نیاز به الگوریتمی برای نظارت و توزیع چرخه نوشتن در صفحات داریم، این الگوریتم را "پوشش صحیح" می نامیم. وقتی که از این الگوریتم استفاده نشود، صفحه ها در یک نرخ مشابه استفاده نخواهند شد.

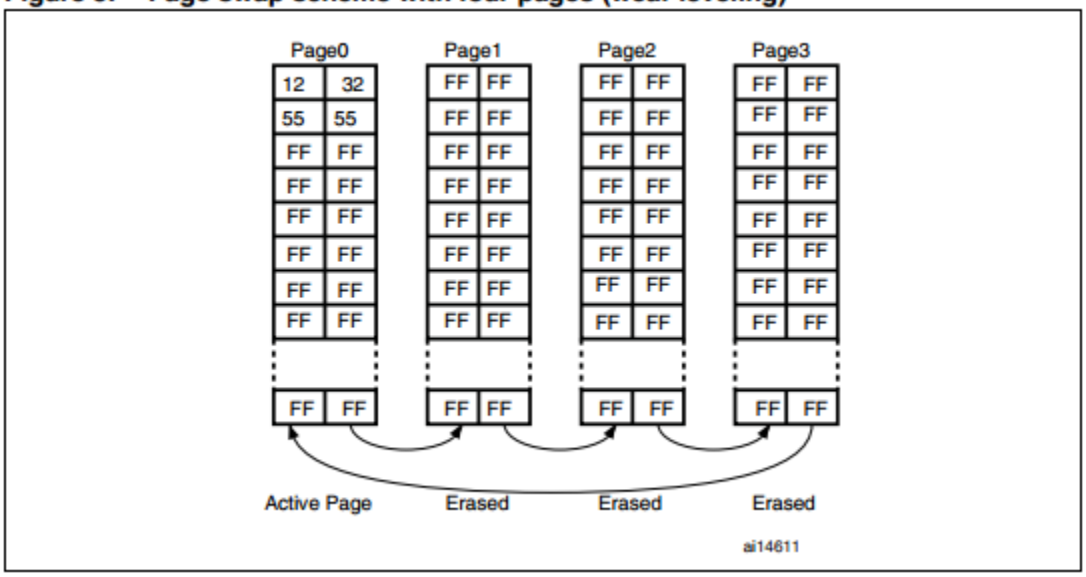
۴-۲-۱- مثال اجرای الگوریتم پوشش صحیح

در این مثال، به منظور افزایش ظرفیت EEPROM مجازی، تعداد ۴ صفحه (Page0 , Page1 , Page2 , Page3) استفاده شده است.

الگوریتم به صورت زیر عمل میکند :

وقتی که صفحه n پر است، برنامه به صفحه n+1 سوئیچ می کند. صفحه n حالا یک صفحه بدرد نخور است و پاک می شود. وقتی که صفحه ۳ پر می شود، برنامه بر میگردد به صفحه ۰، و صفحه ۳ که بدرد نخور شده است را پاک میکند و همین روال ادامه دارد. به شکل ۵ توجه نمایید.

Figure 5. Page swap scheme with four pages (wear leveling)



شکل ۵ روند جابجایی صفحه با ۴ صفحه (پوشش صحیح)

الگوریتم پوشش صحیح توسط تابع EE_FindValidPage() ، عملیاتی و اجرا می شود.

۴-۳- بازیابی هدر صفحه ها پس از قطع برق

در زمان های بروز رسانی متغییر، پاک سازی و انتقال صفحه ، در صورتی که برق قطع شود امکان دارد که هدر صفحه یا دیتا خراب شود.

برای آشار سازی این خطا و بازیابی آن، تابع EE_Init() به کار برده می شود. این تابع باید بلافاصله از روشن شدن صدا زده شود. اصل این تابع در ابتدای یادداشت توضیح داده شد. این تابع وضعیت صفحه ها را بررسی می کند و در صورت لزوم بازیابی می کند.

بعد از قطع برق ، تابع EE_Init() برای بررسی وضعیت هدر صفحه به کار می رود. ۹ حالت ممکن وجود دارد، ۳ حالت این وضعیت ها غیر معتبر هستند. جدول ۲ عملکردی که باید در هر وضعیت هدر انجام شود را توضیح می دهد.

جدول ۲ حالت های ممکن هدر وضعیت و عملکرد متناسب با آن

صفحه ۱	صفحه ۰		
	ERASED	RECEIVE_DATA	VALID_PAGE
ERASED	وضعیت غیر معتبر هدر دو صفحه را پاک میکند و سپس صفحه ۰ فرمت می شود	صفحه ۱ پاک می شود و هدر صفحه ۰ مقدار VALID_PAGE میگیرد	استفاده از صفحه ۰ به عنوان صفحه معتبر و پاک کردن صفحه ۱
RECEIVE_DATA	پاک کردن صفحه ۰ و تعیین مقدار VALID_PAGE در هدر صفحه ۱	وضعیت غیر معتبر هر دو صفحه را پاک میکند و صفحه ۰ را فرمت میکند	استفاده از صفحه ۰ به عنوان صفحه معتبر و ارسال آخرین بروزرسانی متغییر ها از صفحه ۰ به صفحه ۱ و سپس تعیین مقدار VALID_PAGE در هدر صفحه ۱ و پاک کردن صفحه ۰
VALID_PAGE	استفاده از صفحه ۱ به عنوان صفحه معتبر و پاک کردن صفحه ۰	استفاده از صفحه ۱ به عنوان صفحه معتبر و ارسال آخرین بروزرسانی متغییر ها از صفحه ۰ به صفحه ۰ و سپس تعیین مقدار VALID_PAGE در هدر صفحه ۰ و پاک کردن صفحه ۱	وضعیت غیر معتبر هر دو صفحه پاک می شود و سپس صفحه ۰ فرمت می شود

۴-۴- ظرفیت چرخه

منظور از یک چرخه برنامه ریزی/پاک کردن ، یک یا چند بار نوشتن و یک بار پاک کردن صفحه می باشد. یعنی هر بار که چیزی در صفحه نوشته شود و بعد صحفه پاک شود یک چرخه حساب می شود حالا خواه یک مورد نوشتن در صفحه داشته باشیم و یا چند مورد.

وقتی که تکنولوژی EEPROM استفاده شود ، هر بایت می تواند به مقدار محدودی نوشته و پاک شود، عموما بین ۱۰۰۰۰ تا ۱۰۰۰۰۰ دفعه.

هرچند، در حافظه های Flash توکار ، حداقل سایز پاک کردن، یک صفحه است و تعداد چرخه برنامه ریزی/پاک کردن اشاره به تعداد چرخه های پاک کردن یک صفحه دارد(نه اشاره به بایت). برای چیپ های STM321F10xxx تعداد ۱۰۰۰۰ بار برنامه ریزی/پاک کردن تضمین شده است. ماکزیمم مدت زمان زندگی EEPROM مجازی بدین وسیله محدود است.

ظرفیت چرخه به سایز و تعداد دیتایی که کاربر به کار می برد بستگی دارد.

در این مثال، دو صفحه (۱ کیلو در قطعات تراکم متوسط و ۲ کیلو در قطعات تراکم بالا) به صورت برنامه ریزی با دیتای ۱۶ بیتی استفاده شده است. به هر متغییر یک آدرس مجازی ۱۶ بیتی تخصیص داده شده است. این است که ، هر متغییر به اندازه یک word فضای حافظه را اشغال میکند(۱۶ بیت دیتا و ۱۶ بیت آدرس مجازی متغییر). یک صفحه می تواند مقدار ۱ کیلو بایت (برای قطعات تراکم متوسط) یا ۲ کیلو بایت (برای قطعات تراکم بالا) ذخیره کند که ضرب در پوشش چرخه حافظه Flash که ۱۰۰۰۰ است شده و مقدار نتیجه ۱۰۰۰۰ کیلو بایت (برای قطعات تراکم متوسط) یا ۲۰۰۰۰ کیلو بایت (برای قطعات تراکم بالا) می شود. این مقدار قابلیت ذخیره دیتا برای مدت زمان عمر یک صفحه در EEPROM مجازی است. در نتیجه ، اگر برای EEPROM مجازی ، دو صفحه قرار داده شود تعداد ۲۰۰۰۰ کیلو بایت یا ۴۰۰۰۰ کیلو بایت داده را می تواند در خود ذخیره کند. اگر بیش از دو صفحه استفاده شود ، نتیجتا تعداد بایت ها چند برابر می شود.

دانستن عرض داده ها از یک متغییر ذخیره شده، امکان محاسبه تعداد کل متغییر های که می تواند در حافظه EEPROM مجازی در طول عمر حافظه ذخیره شود را میسر میکند.

جدول ۳ ، بر اساس اندازه دیتا و آدرس های مجازی متتغییر ها ، ایده ای را برای محاسبه تعداد متغییر های که در حافظه EEPROM مجازی میتوان ذخیره کرد را ارائه میکند.

جدول ۳ ماکزیمم تعداد متغییر ذخیره شده در EEPROM مجازی (با چرخه ۱۰۰۰۰) (۱) (۲)

سایز متغییر	۲ × صفحه های ۱ کیلو بایت	۲ × صفحه های ۲ کیلو بایت	واحد

متغییر ۸ بیتی (با ۸ بیت برای آدرس مجازی)	10000 × (210 - 2)	10000 × (211 - 2)	متغییر
متغییر ۱۶ بیتی (با ۱۶ بیت برای آدرس مجازی)	5000 × (210 - 4)	5000 × (211 - 4)	متغییر
متغییر ۳۲ بیتی (با ۳۲ بیت برای آدرس مجازی)	2500 × (210 - 8)	2500 × (211 - 8)	متغییر

۱. ماکزیمم تعداد متغییر ها شامل آدرس مجازی مربوطه اش نمی باشد.

۲. مقدار کم شده از تعداد متغییر های قابل تعریف به خاطر فضایی است که برای هدر هر صفحه در بالای صفحه در نظر گرفته شده است. بسته به نوع متغییر برای حفظ آرایش برنامه ، مقداری بایت خالی به بایت های هدر صفحه اضافه می شود. این بایت ها نیز از تعداد ماکزیمم متغییر کاهش داده شده است.

۵- منابع

۱- اطلاعات قطعه خانواده STM32F10xxx , شرکت STmicroelectronic

۲- یادداشت کاربری به شماره AN2594 , شرکت STmicroelectronic

۳- فایل برنامه ریزی دستی حافظه Flash به شماره PM0075 , توسط شرکت STmicroelectronic