Volatile

کلمه کلیدی volatile صلاحیتی است که هنگام اعلام آن برای یک متغیر اعمال می شود. این کلمه به کامپایلر می گوید که مقدار متغیر ممکن است در هر زمان تغییر کند - بدون اینکه اقدامی توسط کدی انجام شود که کامپایلر در آن نزدیکی پیدا کند. پیامدهای این موضوع کاملا جدی است .

کلمه کلیدی volatile در تعریف متغیرهای سراسری -global- به کار میرود. این عبارت به کامپایلر اعلام میکند که :

" مقدار این متغیر ممکنه خارج از روال عادی برنامه -که تو ازش خبر داری- تغییر کنه، پس حواستو جمع کن و هر وقت هم قرار شد ازش استفاده کنی یک بار مقدارش رو بخون -فرض رو بر این نگیر که مقدارش تغییر نکرده".-

در این صورت کامپایلر هرجا این متغیر را میبیند بر رویش بهینه سازی انجام نمیدهد و اینطور در نظر میگیرد که، از جایی نامشخص ممکن است مقدارش را تغییر داده باشند. اما این جای نامشخص -برای کامپایلر- چیست؟

Volatile به معنای مقیم باعث می شه که از رجیستر های CPU که همون R0 تا R31 هستند استفاده نکنه و متغیر رو در فضای SRAM قرار بده.

در زبان اسمبلی اگر متغیری در فضای SRAM باشه کامپایلر بخواد از اون متغیر استفاده کنه ، اول باید اون متغیر رو به یکی از رجیستر های سی پی یو به عنوان مثال R2 انتقال بده و بعد از اون از متغیر برای انتقال ، مقایسه یا هر منظور دیگری استفاده کنه .
اگر از Volatile برای تعریف متغیر استفاده نکرده باشید و مجددا از این متغیر استفاده کنید ، کامپایلر دوباره مقدار متغیر را از SRAM نمیخونه و مستقیما از رجیستر CPU مجددا استفاده می کنه ، هر چند که ممکنه قبل از اون مقدار متغیر در SRAM تغییر کرده باشه ولی در رجیستر CPU مقدار همون مقدار قبلی باشه.

قرار دادن Volatile در هنگام تعریف متغیر باعث میشه با هر بار استفاده از متغیر ، کامپایلر مستقیما از فضای اختصاص داده شده از SRAM مقدار جدید رو بگیره و تغییرات مقدار متغیر در قسمتهای دیگر برنامه تشخیص داده بشه.

اگر حجم رم مورد نیاز برنامه کم باشه و عبارت Volatile رو قرار ندید ، کامپایلر از رجیستر های CPU برا اختصاص فضا به متغیر استفاده می کنه که مشکل فوق به وجود نمیاد ولی اگر برنامه حجیم تر بشه و فضای RAM بیشتری نیاز باشه ، کامپایلر مجبوره که از فضای SRAM برای بعضی از متغیر ها استفاده کنه ، که اگر هنگام تعریف این متغیر ها از volatile استفاده نکنید ، به مشکل فوق بر می خورید(مثلا گیر کردن برنامه توی یک While در این حالت برای نوشتن در متغیر از حافظه ی SRAM استفاده می شه و برای خواندن از متغیر از رجیستر CPU استفاده میشه).

When can we use volatile???
1.Code that works fine--until you enable compiler optimizations

2.Code that works fine--until interrupts are enabled

3.Flaky hardware drivers

4.RTOS tasks that work fine in isolation--until some other task is spawned

1.کدی که خوب کار کند- تا زمانی که بهینه سازی را روشن نکنید

2.کدی که تا زمانی که قطع می شود خوب کار می کند

3.درایور سخت افزاری پوسته پوسته

4.وظایف RTOS که به خوبی در انزوا کار می کنند - تا زمانی که انجام برخی کارها دیگر انجام نشود

Syntax of C's volatile Keyword
volatile uint16_t x;

uint16_t volatile y;

برای استفاده از این کلمه کلیدی می توانیم قبل یا بعد از نوع متغیر ان را قرار دهیم.

مورد اول و دوم مثل هم عمل میکنند . هردو یک متغیر اینتیجر که 16 بیت هست و بدون علامت هست را تعریف میکنند .

volatile uint8_t * p_reg;

uint8_t volatile * p_reg;

تعریف یک پوینتر به متغیری از این نوع خیلی مرسوم است .
استفاده از volatile-pointers به none-volatile pointers خیلی کم است .

uint16_t * volatile p_x;

استفاده از volatile-pointers به volatile-variables :

uint16_t volatile * volatile p_y;

اگر از volatile برای structs یا union استفاده کنیم کل متغیر های انها هم از این نوع میشوند میتوانیم هرکدام را که خواستیم volatile تعریف کنیم.

یک متغیر باید زمانی از نوع volatile تعریف شود که مقدارش به صورتی که انتظار نمی رود ممکن باشد تغییر کند.

Proper Use of C's volatile Keyword

سه نوع داده ای که ممکن است تغییر کنند:

1. Memory-mapped peripheral registers

2. Global variables modified by an interrupt service routine

.3 Global variables accessed by multiple tasks within a multi-threaded application