سوال 1)

مزایای به کارگیری کتابخانه های استاندارد CMSIS

The benefits of the CMSIS are:

•CMSIS reduces the learning curve, development costs, and time-to-market. Developers can write software quicker through a variety of easy-to-use, standardized software interfaces.

•Consistent software interfaces improve the software portability and re-usability. Generic software libraries and interfaces provide consistent software framework.

•It provides interfaces for debug connectivity, debug peripheral views, software delivery, and device support to reduce time-to-market for new microcontroller deployment.

•It allows to use the compiler of your choice, as it is compiler independent.

•It enhances program debugging with peripheral information for debuggers and ITM channels for printf-style output.

•CMSIS is delivered in CMSIS-Pack format which enables fast software delivery, simplifies updates, and enables consistent integration into development tools.

•Continuous integration is common practice for most software developers nowadays. CMSIS-Build supports these workflows and makes continuous testing and validation easier.

مزایای استفاده از CMSIS :

منحنی یادگیری، هزینه های development و زمان عرضه به بازار را کاهش میدهد.

Developer ها میتوانند از طریق انواع رابط های نرم افزار استاندارد شده و با کاربری آسان، نرم افزار را سریع تر بنویسند پس یک زیرساخت که بتواند برای تولید و فهم کد در پروژه های مختلف استفاده شود دارد. همچنین یک سری تعاریف دارد که سریع تر و قابل فهم تر هستند و برنامه نویس میتواند از آن ها استفاده کند .رابط های نرم افزاری سازگار، قابلیت حمل و استفاده مجدد نرم افزار را بهبود می بخشد. کتابخانه ها و رابط های نرم افزار عمومی، چارچوب نرم افزاری سازگاری را ارائه می دهند. به این معنا که از طریق استاندارد سازی کردن کد میتوان استفاده مجدد از آن ها داشت.

Interface هایی برای دیباگ کردن اتصال ها، دیباگ اجزا های خارجی، تحویل نرم افزار و پشتیبانی دستگاه برای کاهش زمان عرضه به بازار برای استقرار میکروکنترلرهای جدید فراهم می کند.

به ما اجازه میدهد کامپایلری که میخواهیم استفده کنیم زیرا وابسته به کامپایلر خاصی نیست. یه این معنا که به chain tool خاصی مثل کامپایلر ارم یا کامپایلر GCCوابسته نیست و مستقل است و مبتنی بر استاندارد C برای chain tool های مختلف قابلیت به کار گیری دارد.

دیباگ کردن برنامه را با اطالعات جانبی برای دیباگرها و کانال های ITM برای خروجی های به سبک printf افزایش می دهد.

CMSIS در قالب Pack-CMSIS ارائه می شود که تحویل سریع نرم افزار را امکان پذیر می کند، به روزرسانی ها را ساده می کند و یکپارچگی مداوم با ابزارهای توسعه را امکان پذیر می سازد.

امروزه ادغام مداوم برای اکثر توسعه دهندگان نرم افزار رایج است Build-CMSIS .از این گردش های کاری پشتیبانی می کند و آزمایش و اعتبارسنجی مداوم را آسان تر می کند. زیرا source open است و مخفی نیست و میتوان آن را از گیت هاب آرم دانلود و استفاده کرد.

کاربرد CMSIS-DSP :

Cortex-M4 یک"کنترل کننده سیگنال دیجیتال"با تعدادی پیشرفت برای پشتتیبانی از الگوریتم های DSP است. توسعه یک سیستم DSP به بهترین وجه به عنوان یک "سرگرمی غیرمعمول" توصیف می شود و می تواند برای همه به جز ساده ترین سیستم ها بسیار دلهره آور باشد. برای کمک به انسان ها که الگوریتم های DSP را در پروژه هایCortex-M4 و Cortex-M3 بگنجانند، CMSIS شامل یک کتابخانه DSP است کته بیش از 60 مورد از متداول ترین توابع ریاضی DSP را ارائه می کند.

This user manual describes the CMSIS DSP software library, a suite of common signal processing functions for use on Cortex‑M and Cortex-A processor based devices.

The library is divided into a number of functions each covering a specific category:

- Basic math functions
- Fast math functions
- Complex math functions
- Filtering functions
- Matrix functions
- Transform functions
- Motor control functions
- Statistical functions
- Support functions
- Interpolation functions
- Support Vector Machine functions (SVM)
- Bayes classifier functions
- Distance functions
- Quaternion functions

The library has generally separate functions for operating on 8- bit integers, 16-bit integers, 32-bit integer and 32-bit floating‑point values.

The library is providing vectorized versions of most algorithms for Helium and of most f32 algorithms for Neon.

CMSIS-DSP: به طور خلاصه برای کاربرد های سیگنال پراسسینگ و اینستراکشن های ویژه و کاربرد های point fixed و point floatingاست.

مجموعه ای از عملکردهای رایج پردازش سیگنال برای استفاده در دستگاه های مبتنی بر پردازنده Cortex-A و Cortex-M

کتابخانه به تعدادی عملکرد تقسیم می شود که هر کدام یک دسته بندی خاص را پوشش می دهند:

توابع ریاضی پایه

توابع سریع ریاضی

توابع پیچیده ریاضی

توابع فیلتر کردن

توابع ماتریسی تبدیل توابع

توابع کنترل موتور

توابع آماری

توابع پشتیبانی

توابع ماشین بُردار پشتیبانی(SVM)

توابع طبقه بندی کننده

و...

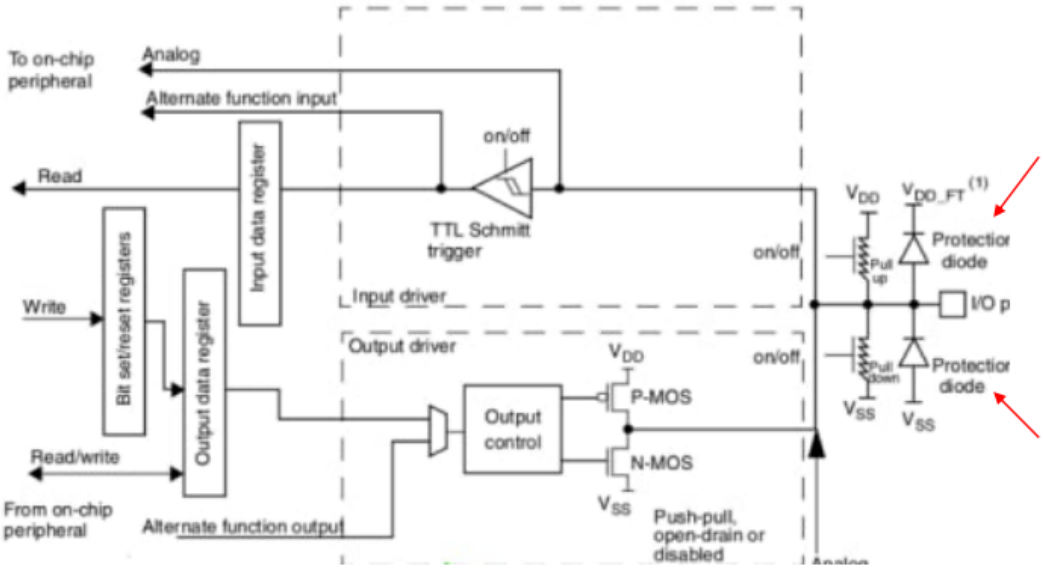
این کتابخانه به طور کلی دارای توابع مجزایی برای کار بر روی اعداد صحیح 8 بیتی، اعداد صحیح 16 بیتی، اعداد صحیح 32 بیتی و مقادیر ممیز شناور 32 بیتی است. این کتابخانه نسخه های برداری از اکثر الگوریتم را ارائه می دهد.

سوال 2)

معمولا برای آنکه پایه های ریزپردازنده به دلیل جریان کشی های غیر قابل پیش بینی سیستم آسیب نبیند، مستقیما به المان های مصرف کننده مانند سون سگمنت متصل نمی شوند.

بسته به شرایط و هدف ما در محافظت از میکروکنترلر، اعمال متفاوتی انجام میشود.

به عنوان مثال، برای محافظت میکروکنترلر از جریانهای زیاد ورودی یا خروجی، یک یا چند مقاومت با ضرایب مناسب را به صورت سری در مدار ورودی و خروجی این قطعات قرار میدهیم تا جریان ورودی و خروجی پایه های میکروکنترلر، کنترل شده باشند.



میتوانیم از یک سری دیود های پروتکشن استفاده کنیم. این دیود های پروتکشن بار هایی که جمع شدند و یک جریان لحظه ای خیلی بالا و آسیب زننده به مدارات داخل چیپ میتونن داشته باشند را یک جوری هدایت میکنند که وارد چیپ نشوند و به زمین یا تغذیه وصل بشوند.

سوال 3)

بخش اول)

رنگ و کارایی نوری LED ها به مواد و فرآیندهای ساخت ال ای دی مربوط می شود. در حال حاضر، قرمز، سبز و آبی به طور گسترده ای استفاده می شود. مواد مورد استفاده در ساخت ال ای دی ها می توانند فوتون هایی با انرژی های مختلف تولید کنند و از این طریق طول موج نور ساطع شده از LED، یعنی طیف یا رنگ را کنترل کنند. در روزهای اولیه صنعت LED، ساختار مواد GaAs1-xPx به صورت تئوری برای تولید LED با هر طول موج از نور مادون قرمز تا نور سبز استفاده می شد. طول موج LED را می توان با افت ولتاژ اتصال PN تعیین کرد.

بین این ها ما سه مدل led با رنگ های : قرمز ، نارنجی و زرد داریم

علت وجود این رنگ ها این است که در ساخت انها سه عنصر : آرسنیک ، فسفر و گالیم استفاده میشود و این سه مدل led را سه عنصر luminous tube میگویند.

و گروه دیگری از LED ها هستند که ترکیب های استفاده شده در آنها به صورت زیر است :

Blue LED : GaN = Gallium Nitride

Red LED : GaA = Gallium Arsenic

Green LED : GaP = Gallium Phosphorus

که به این سه رنگ LED ما دو عنصری میگوییم

بخش دوم)

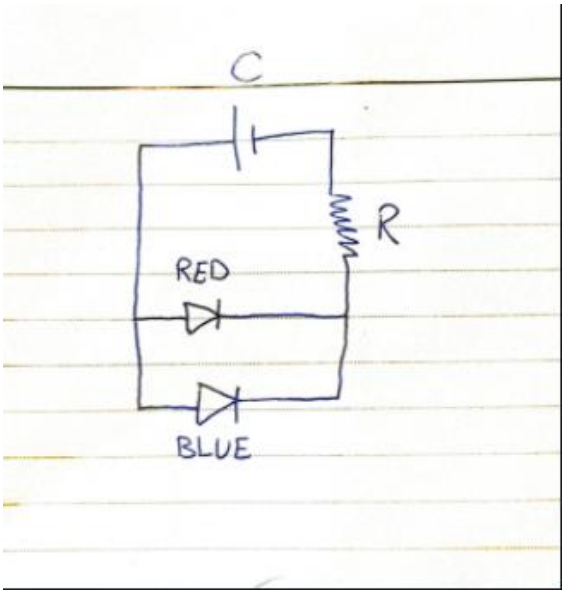
بهتر است این کار انجام نشود زیرا با توجه به اینکه هر LED از ترکیبات متفاوتی تشکیل شده threshold ولتاژ آنها باهم متفاوت است.

به عنوان مثال با اطلاعاتی که بدست اوردیم افت ولتاژ LED قرمز کمتر از LED آبی میباشد و اعداد به صورت زیر هستند :

Red : 1.8v voltage drop

Blue : 3v voltage drop

و به عنوان مثال در مدار زیر :



اگر مقادیر خازن و مقاومت به گونه ای باشند که مقدار آنها مینیمم برای روشن کردن LED قرمز که مقدار افت ولتاژ کمتری دارد کافی باشد، در نتیجه LED آبی روشن نخواهد شد و اگر این LED هارا جایگزین کنیم ممکن است در مداری با مقادیر dynamic مشکلاتی از قبیل روشن شدن یا نشدن LED مورد نظر را شاهد باشیم.

برای حل این موضوع نیز راه حلی که ارائه شده این است که از یک لیمیت کننده جریان برای LED ها استفاده کنیم.

سوال 4)

وقتی دکمه فشاری یا سوئیچ ضامن یا میکرو سوئیچ را فشار می دهیم، دو قسمت فلزی با هم تماس پیدا می کنند تا منبع تغذیه را کوتاه کنند. اما آنها فوراً به هم وصل نمی شوند بلکه قطعات فلزی چندین بار قبل از برقراری اتصال پایدار واقعی متصل و جدا می شوند. هنگام رها کردن دکمه نیز همین اتفاق می افتد. این باعث می‌شود که راه‌اندازی کاذب یا راه‌اندازی چندگانه مانند دکمه چند بار فشار داده شود. مانند افتادن یک توپ پرنده از بلندی است و همچنان بر روی سطح می پرد، تا زمانی که استراحت کند.

به سادگی می توان گفت که switch bouncing رفتار غیر ایده آل هر سوئیچ است که چندین انتقال از یک ورودی واحد ایجاد می کند. هنگامی که ما با مدارهای برق سروکار داریم، switch bouncing مشکل عمده ای نیست، اما در زمانی که با مدارهای منطقی یا دیجیتال سروکار داریم، مشکلاتی ایجاد می کند. از این رو، برای حذف جهش از مدار، از Switch Debouncing Circuit استفاده می شود.

Software Debouncing:

Debouncing در نرم افزار نیز اتفاق می افتد، در حالی که برنامه نویسان تاخیرهایی را برای خلاص شدن از Software Debouncing اضافه می کنند. اضافه کردن تاخیر کنترل کننده را مجبور می کند تا برای یک دوره زمانی خاص متوقف شود، اما اضافه کردن تاخیرها گزینه خوبی در برنامه نیست، زیرا برنامه را متوقف می کند و زمان پردازش را افزایش می دهد. بهترین راه استفاده از interupt در کد برای Software Debouncing است.

Hardware Debouncing:

در تکنیک hardware debouncing افزاری از فلیپ فلاپ S-R برای جلوگیری از switch debouncing مدار استفاده می کنیم. این بهترین روش انحرافی در بین همه است.

منبع:

https://circuitdigest.com/electronic-circuits/what-is-switch-bouncing-and-how-to-prevent-it-using-debounce-circuit#:~:text=Simply%2C%20we%20can%20say%20that,the%20logic%20or%20digital%20circuits