فهرست دوره جامع آموزش
برنامه نویسی میکروکنترلر آرم

فصل اول : زبان برنامه نویسی سی و نرم افزارSTM32CubeIDE

منابعی که در این فصل بررسی میشن:

- **STM32F103x Datasheet**
- **The C Programming Language by Kernighan & Ritchie**
- **The GNU C Reference Manual**
- **STM32CubeIDE user guide_UM2609**

Datasheet میکروکنترلرstm32f103

- **نقشه ی حافظه) ساختار حافظه ، جایگاه flash ، sram، stm32 peripheral register، cortex-m3 peripheral، (system memory**
- **پریفرال ها و رجیسترهایSTM32**
- **باسها**
- **cpu، instruction set، رجیسترهایcpu**
- **بررسی block diagram و نحوه ی عملکرد میکروکنترلر با مثال ، add ldr str instructions**
- **خوانواده میکروکنترلر , medium density , low density , performance line , F103 : high density**
- **stm32 Ordering information scheme**
- **پین ها و شماتیک : blue pill تغذیه ، ریست ، کلاک ، بوت ، پورتها ، پروگرم و دیباگ**
- **Boot mode STM32F103**
- **جدول 5 : مشخصات پین ها**
- **Absolute maximum ratingجدول 6 و 7**
- **sink & source**
- **injected and leakage current**
- **Operating conditionجدول 9**
- **I/O port characteristic، بررسی حد پایین و بالای ولتاژ برای پین های ورودی و خروجی**
- **CMOS & TTL**

- Output driving current، بررسی جریان پین خروجی در ولتاژ های متفاوت ، جدول 36
- Relaxed Vol/Voh

زبان برنامه نویسی C ، برنامه نویسیSTM32

- کامنت های یک خطی و چند خطی
- تعریف متغیر
- انواع متغیر اولیهint , short , char :
- ساخت نوع جدید متغیر باtypedef
- fixed width data type
- Fixed width integer data types , 8 , 16 , 32 ,64 bits
- Float & double
- Declaration & declaration list & Variable initialization
- Literals and constants
- نوشتن integer literal ها در مبنای 8 و 16 و10 ، نوشتن float literal ها
- Literal postfix : U , UL , ULL , LL , F
- const qualifier
- متغیر های local & global و تفاوتinitialization
- Expressionها ، constant & variable
- expressionهای حاوی operator ها و operand ها
- arithmetic operator : + , – , * , / , % :نوع و مقدارarithmetic expression
- Implicit type conversion :بررسی مثال
- Wider types
- assignment : += , *= , /= , -= , %= , = :نوع و مقدارassignment expression
- عملگر assignment و integer های با عرض مختلف
- R value & L value
- ++و —— ، تاثیر ++ و – قبل و بعد ازexpression
- relational : == , != , < , <= , > , >= :نوع و مقدارrelational expression
- عملگر تغییر نوع وexplicit type conversion

- logical : ! , && , || :نوع و مقدار logical expression
- conditional؟: ، نوع و مقدار conditional expression
- اولویت عملگر ها precedence and associativity :
- شرط if – else if – else , if – else , if
- Compound statement
- while
- حلقه ی for : initialization , condition , increment
- مقایسه while & for
- فرار از حلقه return :
- مکانیزم فرار از حلقه break:
- فرار از حلقه goto :
- حلقه do while
- Switch case
- مقایسه if و switch و موارد استفاده هر کدام
- continue در حلقه for و while
- لیبل گذاری و goto
- کاربرد های goto و محدودیت ها
- معرفی پوینتر ، تعریف پوینتر
- Reference operator & , dereference operator
- پوینتر به پوینتر
- Dangling pointer , wild pointer
- void پوینتر و محدودیت ها
- NULL pointer و کاربرد
- Call by value , call by reference
- نوع void
- معرفی تابع، return type و بررسی فراخوانی تابع توسط cpu
- بررسی متغیر های داخل تابع و آرگومان های ورودی و مکان در stack

- پروتوتایپ تابع و implicit declaration
- نوشتن تابع delay و nested loops
- الگوریتم های iterative و recursive با بررسی مثال فاکتوریل
- بررسی stack و الگوریتم های recursive

نرم افزار STM32CubeIDE

- معرفی و ساخت Workspace
- لغو دانلود های غیر ضروری
- stm32 Target selection
- ساخت پروژه bare metal برای STM32
- تغییر فونت و اندازه
- دسترسی به pdf ها و ویدئو ها در target selection و information center STM32
- Compile , download and debug , elf file, edit launch config
- دیباگ خط به خط کد : step into , step over , step return
- نمای console
- نمای problem
- پرسپکتیو open , reset , close , show text :
- ساخت perspective save as : perspective
- شخصی سازی perspective : customize perspective
- تنظیم toolbars , menu & shortcuts در perspective ها
- Debug tools: resume , terminate , suspend
- مدیریت toolbars و customize perspective و hide/show toolbar
- Show view در perspective های مختلف و دسترسی به تمام view ها
- Whitespace ها و نمایش whitespace
- Clone کردن پنجره
- Toggle split editor, horizontal & vertical

- Full screen , maximize and minimize active view
- help : st-link upgradeمنوی
- call hierarchy : show callees , show callers
- type hierarchy
- task
- minimap
- outline
- properties
- template
- bookmark
- Annotationها : error, warning, bookmark, task, search, breakpoint
- جابجایی بین annotation ها
- Search and replace
- نماهای expression & live expression & variable و تفاوت ها
- Number format in expression & live expression & variable
- expression : detail view
- expression : layout , vertical , horizontal , automatic , expression view only
- expression : show columns , select columns , address column
- expression : add expression group
- expression : number format for each expression , restore to preferences
- expression : find expression
- variable : select columns، location و تفاوت باaddress
- memory : مشاهده متغیرها در حافظه وrendering
- جابجایی نماها درperspective : detach , open new window
- نمای registers : مشاهده ی رجیستر های cpu در هنگام دیباگSTM32
- Break pointها و toggle break point وskip all break point
- Line break : توقف کد در بخش دیباگ بدلیل رسیدن به یک خط از پیش تایین شده.
- Method break point : توقف کد در بخش دیباگ بدلیل رسیدن به تابع
- نمای break point و غیر فعال کردن break point ها

- اضافه کردن watch point با استفاده از نمایexpression
- متوقف کردن debug در نتیجه خواندن یا نوشتن در یک آدرس
- Add dynamic printf : پرینت کردن مقدار متغیر در صورت عبور cpu از یک لاین مشخص کد.

فصل دوم : کامپایلر GCC

منابعی که در این فصل بررسی میشن:

- **Using the GNU Compiler Collection**

ورژن های زبان برنامه نویسی C ، کامپایلر GCC ، فایل ها و کلاس های ذخیره سازی

- معرفی نرم افزار keil و ساخت پروژه خالی در keil
- ورژن های c ، c80 ، c90 ، c99 ، c11و c17
- محدودویت های ورژن c89 در مورد for
- ورژن c در keil
- تغییر ورژن c در cubeide
- معرفی کامپایلر ARMCC در keil
- کامپایلر GCC در STM32CubeIDE
- معرفی GNU project
- فایلهای .c و .h و اضافه کردن به پروژه
- ساخت کتابخانه
- معرفی مکان فایل های .c و .h به نرم افزار STM32CubeIDE
- بررسی نحوه ی عملکرد و اجزای کامپایلر gcc و وظیفه ی هر بخش
- preprocessor , compiler , assembler , linker , locater
- فایل های .a , .o , .i , .s , .h , .c در مراحل مختلف کامپایل
- sectionها و ساختار فایل assembly
- .text , .rodata , .data , .bss
- symbol table , import table , export table با مثال
- فرایند relocation در مرحله linker
- بررسی symbol resolution در مرحله linker

- مرحله locater و فایل linker script
- sectionهای ram
- Sectionهای flash
- بررسی تمام storage class ها با مثال
- کلاس ذخیره سازی static برای متغیر های local
- staticبرای متغیر های global
- کلاس ذخیره سازی static برای متغیر های داخل header file
- staticبرای توابع محدود به فایل
- کلاس ذخیره سازی extern برای متغیر global
- autoبرای متغیر local
- کلاس ذخیره سازی register برای متغیر local

اکستنشن های GCC

- اعداد منفی در مبنای 2 : روش بیت علامت
- روش one’s compliment
- two’s compliment
- مقایسه و بررسی مزایا و معایب روش های مختلف نمایش اعداد منفی در مبنای 2
- مروری بر نمایش integer ها در مبنای 2
- نمایش اعداد اعشاری در مبنای 2
- تبدیل مبنا با اعشار
- استاندارد ieee 745 وfloating point
- معرفی اعداد Fixed point با مکان ممیز ثابت
- نوع های متغیر _Fract و _Accum و تمام مشتقات
- طراحی چند آزمایش در مورد ویژگی های متغیر های از نوع _Accum
- اعداد fixed point در بخش debug
- اجزای متغیر های fiexed point : integer part , fractional part , sign bit

- مقایسه متغیر های fixed point وfloating point
- منوالUsing the GNU Compiler Collection
- پسوند ها برای اعدادfixed point
- اعمال ریاضی رویfixed point
- معرفی extension ها و بررسی Extensions to the C Language Family در using the GCC
- بررسی : locally declared labels لیبل های داخل بلوک کد
- labels as values : labelمقدار
- بررسی : nested functions تابع داخل تابع ، توابعlocal
- typeof
- معرفیGCC Command Options
- Attributes
- roject propertiesو تنظیمات کامپایلر درstm32cubeide
- آپشن های نوشته شده در MCU GCC Compiler از داکیومنتUsing the GCC
- ARM options
- آپشن-mcpu=name
- option -mtune=name
- آپشن های کنترل استانداردC
- -std=gnu11 ، تنظیم آپشن از تبgeneral
- آپشن های-g3 , -g0 , debug ، تنظیم آپشن از تبdebugging
- کنترلpreprocessor
- آپشن -D ، تنظیم آپشن از تبpreprocessor
- معرفی مکان فایل های header از بخشinclude path
- مکان فایل های header از بخشpath and symbols
- -Iبرای معرفی مکان فایل های header به کامپایلر
- آپشن های کنترلoptimization

- تفاوت آپشن و فلگ
- آپشن O0- دسترسی از تبoptimization
- فلگ هایfdata-sections -ffunction-sections-
- آپشن Wall- درخواستwarning
- specs-و معرفی کتابخانه های استاندارد به کامپایلر
- آپشن mfloat-abi- و پیاده سازی floating point arithmetic با نرم افرار
- mthumb-وmarm-
- آپشن های کنترل خروجیGCC
- c-، عدم اجرایlinker
- آپشن S- ، توقف قبل ازassembler
- E-، فقط preprocessor اجرا شود
- آپشنassembler-with-cpp x language-
- مشخص کردن فایل های ورودی از تبpreprocessor

فصل سوم : دسترسی به حافظه در زبان C

منابعی که در این فصل بررسی میشن:

- **STM32F10 Reference manual_RM0008**
- **Using the GNU Compiler Collection**

رفرنس منوال و پریفرال GPIO میکروکنترلرSTM32F103

- **لیست کلمات اختصاری ویژگی بیت : rw, r, w, rc_w1, rc_w0, rc_r, rs, r_w, t**
- **Memory and bus architecture**
- **بررسی بلوک دیاگرام و نحوه ی عملکرد پریفرال GPIO میکروکنترلرSTM32F103**
- **دیود های محافظت**
- **Pull up & pull down**
- **Push pull & open drain**
- **بیت های MODE و CNF و تنظیمات پین در حالت های مختلف**
- **رجیستر هایBRR , BSRR , ORD , IDR**
- **مکانیزم قفل کردن تنظیمات GPIO و رجیستر LCKK**
- **وصل کردن کلاک پریفرال ها ، پریفرال RCC و رجیستر RCC_APB2ENR**
- **محاسبه آدرس رجیستر های و تنظیمات مورد نیاز برای ال ای دی چشمک زن**

ساخت پوینتر

- **نمای SFRs و تغییر مقدار رجیسترها**
- **Ledچشمک بدون کد نویسی با استفاده از قابلیت های debug درcubeide**
- **ساخت پوینتر برای هر رجیستر با کست کردن آدرس**
- **Ledچشمک زن بدونbitwise operator**
- **Bitwise operator : and , or , left shift , right shift , xor , not**
- **ساخت ماسک بیت باleft shift**
- **یک کردن بیت باbitwise or**

- صفر کردن بیت با bitwise and وnot
- تاگل کردن بیت باbitwise xor
- خواندن بیت با bitwise and وright shift
- ledچشمک زن باbitwise operator
- استفاده از پوینتر به عنوان آرایه
- ساخت ماشین حساب بیتی درlabview

استراکچر و CMSIS استاندارد

- structتعریف مقداردهی اولیه ، دسترسی به اعضاstruct tag , typedef struct
- استراکچر در حافظه
- پوینتر به struct ، کست کردن آدرس و دسترسی به حافظه در قالب استراکچر
- ساخت نوع استراکچر قالب رجیسترهای پریفرال،کست کردن آدرس پریفرال
- دسترسی به رجیسترها درقالب اعضای استراکچر
- Defineها در پنجره یdefine symbols
- اضافه کردن لایه ی نرم افزاریCMSIS
- بررسی فایلstm32f103xx.h
- فایل stm32f103xb.h (peripheral access layer) در لایه نرم افزاریCMSIS
- Peripheral structure , peripheral memory map , peripheral declaration bit definition
- Ledچشمک زن باCMSIS

بیت فیلد و ساخت فایل های CMSIS جدید

- بیت فیلد ، تعریف ، مقدار دهی اولیه و دسترسی
- one byte , 2 byte and 4 byte boundaries
- جایگیری بیت فیلد در حافظه وboundaries
- اندازه بیت فیلد

- کاربرد بیت فیلد در استفاده بهینه از حافظه
- کست کردن آدرس متغیر ها در قالب پوینتر به بیت فیلد
- بیت فیلد جایگزینbitwise operator
- ساخت بیت فیلد قالب رجیستر CRH ، دسترسی به رجیسترها با بیت فیلد
- نوشتن bit definition های جدید ، متناسب با بیت فیلد
- Ledچشمک زن با بیت فیلد
- یکی کردن MODE و CNF در بیت فیلد
- دانلون فایل STM32F103xx.svd با نرم افزارkeil
- ساخت فایل device دارای بیت فیلد باSVDConv.exe
- جایگزین کردن فایل هدر جدید
- و اصلاحdevice peripheral access layer header file

یونیون و کاربردها

- Unionتعریف مقداردهی اولیه ، دسترسی به اعضاunion , union tag , typedef union
- بررسی مثال خروجی سنسور dht22 وChecksum
- تفکیک integer های بزرگ به کوچک با bitwise operator ها
- اتصال integer های کوچک و ساخت integer های بزرگتر با bitwise operator ها
- کاربرد union برای پردازش داده های سنسور ، آرایه و استراکچر در union ، جایگزین bitwise operator
- نحوه ی ذخیره سازی اعداد در حافظهbig endian & little endian :
- ذخیره سازی اعضای union در حافظه و تاثیر little endian برunion
- unionبرای ساخت بافر های کوچک
- کاربرد union در آرایه های با اعضای متفاوت
- unionدر دسترسی به رجیسترهای پریفرال ها
- Underlying type
- اصلاح بیت فیلد برای پریفرال های باسAHB

Alignment & packing

- Structure padding
- Alignmentو محدودیت های دسترسی cpu به حافظه
- ldrb strb, Alignment =1 , byte boundaries
- lrdh strh , Alignment = 2 , half word
- ldr str , Alignment = 4 , word boundaries
- strd ldrd , Alignment = 8 , double word
- alignmentو اندازه استراکچر : 5 مثال
- عملگر__alignof__
- __attribute__((aligned()))__برای استراکچر
- اصلاح alignment برای بهینه سازی کپی استراکچر
- __attribute__((aligned()))__و typedef برای سایر انواع متغیر
- Packingبا اصلاح alignment نوع اعضای استراکچر
- __attribute__((aligned()))__برای متغیرها
- افزایش alignment اعضای استراکچر
- پک کل استراکچر با استفاده از__attribute__((__packed__))__
- Packingانتخابی اعضای استراکچر با__attribute__((packed))__
- pragma pack(1)
- pragma pack(1) pack()
- pragma pack(1) pack(push) pack(pop)

فصل چهارم: پریپراسسور

منابعی که در این فصل بررسی میشن:

- **Using the GNU Compiler Collection**
- **The C Preprocessor**

پریپراسسور دیرکتیوها

- معرفی پریپراسسور**directives**
- **Object like macros**
- نوشتن**bit definition**
- نوشتن**peripheral declaration**
- **#include & #import**
- **#ifndef**و**header guard**
- **#pragma once**
- **ifdef**و فایل های**config**
- **#if , #else , #elif**
- **#error , #warning**
- عملگر**defined**
- تعریف و نحوه ی استفاده از**Function like macros**
- نحوه عملکرد پریپراسسور، اسم ماکرو و **expansion** ماکرو
- استفاده از ماکرو های **set , clear , toggle , read , modify** برای تغییر رجیسترها

مقایسه ماکرو و تابع و بررسی مدیریت استک در اسمبلی

- مقایسه تابع و ماکرو با بررسی اسمبلی کد در محیط دیباگ
- دریافت ورودی ها در ماکرو و تابع
- تفاوت ماکرو و تابع در دیباگ
- بررسی اهمیت پرانتز برای ورودی های ماکرو

- تاثیر پرانتز روی اسمبلی کد اکسپنشن ماکرو
- مقایسه عملگر ++ در ورودی ماکرو و تابع با بررسی کد اسمبلی
- بررسی خروجی تابع به عنوان ورودی ماکرو و تابع
- فراخوانی تابع و ماکرو با ماکرو، مثال ماکروcall
- کجا از ماکرو و کجا از تابع استفاده کنیم
- معرفی داکیومنتThe C Preprocessor
- بررسی دام های ماکرو از داکیومنتThe C Preprocessor
- Misnestingو دلایل پرهیز از آن
- بالانس نبودن پرانتز ها در ماکرو
- Operator precedence problem
- بررسیduplication of side effect
- self-referential macro
- بررسی اسکوپ متغیرهای داخل بلوک کد
- بلوک کد به عنوان ماکرو و نوشتن ماکرو در برکت
- ماکرو ، compound statementو سمی کالن اضافه
- مقایسه جایگیری متغیرهای ماکرو و تابع در استک با بررسی خط به خط اسمبلی و حافظه
- بررسی قدم به قدم استک و رجیستر استک پوینتر وr7
- اینستراکشنpush
- کاربرد رجیستر r7 در مدیریت استک
- آدرس دهی متغیرهای local در اسمبلی
- بررسی اسمبلی اینستراکشن های خروج متغیرهای local از استک
- نوشتن compound statement در چند خط
- ماکرو با do{}while(0) و مزایا
- اصلاح side effect های ماکرو ، دریافت متغیر در ماکرو
- بررسی امکان gcc ، statement and declarations in expressions
- ماکرو compound statement با مقدار، statementدر پرانتز

- تعریف توابع inline و مقایسه ماکرو و تابع inline
- بررسی ارور لینکر در مورد کلمه کلیدی inline و دو راه حل
- تنظیمات optimization و تاثیر بر inline شدن ماکرو
- معرفی attribute های توابع از داکیومنت using gcc
- نحوه معرفی ویژگی های توابع به کامپایلر و کلمه کلیدی __attribute__
- ویژگی always_inline برای inline کردن ماکرو به اجبار

فصل پنجم : آرایه ها، رشته ها، الگوریتم های سرچ و سورت ، SWVو printf

منابعی که در این فصل بررسی میشن:

- **Using the GNU Compiler Collection**
- **The Red Hat newlib C Library**
- **STM32cubeide user guide_UM2609**

مقدمه آرایه و رشته

- تعریف آرایه ، بررسی آرایه در حافظه ، بررسی کد اسمبلی دسترسی به اعضای آرایه
- تفاوت آرایه و پوینتر ، ایدنکس آرایه وpointer*
- ایندکس آرایه و pointer arithmetic در اسمبلی
- پوینتر و آرایه ورودی تابع
- عملگر sizeof و اندازه آرایه
- کاراکترها و جدول اسکی
- تعریف و مقداردهی رشته

پرینت اف

- معرفی پریفرالITM
- بررسی سخت افزار و نرم افزار مورد نیاز برایserial wire viewer STM32
- بررسی تابع write و تابعITM_SendChar
- تنظیمات مورد نیازprintf
- استفاده از تابع printf باSWV
- مقایسه اسکیپ سیکوئنس \n و\r
- اسکیپ سیکوئنس\\ , \? , \t , \’ , \” ,
- \ooo , \xhh

- معرفی منوال the red had newib C library
- ورودی ها و خروجی های printf
- بررسی کامل format specifier ها با مثال
- تایپ ها ٪٪ ، پرینت درصد
- %p , %u , %d , i، پرینت اینتیجر در مبنای 10 و پوینتر
- %g , %e , f، پرینت اعشاری
- %s , c، پرینت کاراکتر و رشته
- %o , x، پرینت در مبنای 8 و 16
- تاثیر سایز های l , ll , h , hh روی تایپ i و d
- سایز های l , ll , h , hh روی تایپ x
- روش های متفاوت نوشتن precision در format specifier
- مشخص کردن تعداد ارقام اعشار با precision برای تایپ های g , f , e
- تعداد کاراکتر ها در رشته با precision
- عرض در format specifier برای تایپ d
- استفاده از فلگ minus
- فلگ zero
- استفاده از فلگ پرانتز
- فلگ کاما
- plus
- استفاده از فلگ # برای نوع های x , o
- فلگ اسپیس
- تغییر ترتیب ورودی های تابع printf با پارامترها

الگوریتم های سرچ و سورت آرایه

- تابع rand و تولید اعداد رندوم
- تابع برای پر کردن آرایه ، کپی آرایه ، پیدا کردن مینیمم و ماکزیمم

- الگوریتم selection sort و پیاده سازی
- bubble sortو پیاده سازی
- بهبود الگوریتم bubble sort و پیاده سازی
- الگوریتم insertion sort و پیاده سازی
- گپ و اضافه کردن گپ به الگوریتم selection sort و پیاده سازی
- الگوریتم shell sort و پیاده سازی
- quick sortو پارتیشن بندی آرایه
- پیاده سازی الگوریتم پارتیشن بندی آرایه
- quick sortبا دو روش
- معرفی تابع qsort (gcc sort) وپیاده سازی
- سورت کردن آرایه ی استراکچری
- مقایسه عملکرد الگوریتم های selection و bubble و insertion و shell و quick و qsort با مقایسهcpu cycle

فصل ششم: پریفرال های RCC ، EXTI، NVIC، SYSTICK

منابعی که در این فصل بررسی میشن:

- **cortex-m3 programming manual**
- **STM32F1 reference manuaul**

پریفرال RCC در میکروکنترلر هایSTM32F1

- مدار **reset** در میکروکنترلر هایSTM32F1
- معرفی**Clock tree**
- منابع کلاک، کریستال ها و مدارهای**RC**
- تفاوت ماژول منبع کلاک و کریستال
- کلاک **HSI** و بیت های **HSION** و **HSIRDY** در رجیستر**RCC_CR**
- فعال کردن کلاک **HSE** و بیت های **HSEON** و **HSERDY** در رجیستر**RCC_CR**
- استفاده از منبع کلاک بجای کریستال و بیت **HSEBYP** در رجیستر**RCC_CR**
- تقسیم کلاک **HSE** بر دو در ورودی **PLL** و بیت **PLLXTPRE** در رجیستر**RCC_CFGR**
- معرفی**PLL**
- مشخص کردن ورودی **PLL** با بیت **PLLSRC** در رجیستر**RCC_CFGR**
- ضریب **PLL** در بیت **PLLMUL** در رجیستر**RCC_CFGR**
- تنظیم **AHB Prescaler** با بیت **HPRE** در رجیستر**RCC_CFGR**
- کلاک پریفرال های باس **APB1** با بیت **PPRE1** در رجیستر**RCC_CFGR**
- تنظیم کلاک پریفرال های باس **APB2** با بیت **PPRE2** در رجیستر**RCC_CFGR**
- کلاک **ADC** با بیت **ADCPRE** در رجیستر**RCC_CFGR**
- تنظیم کلاک **USB** با بیت **USBPRE** در رجیستر**RCC_CFGR**
- روشن کردن **PLL** با بیت های **PLLON** و **PLLRDY** در رجیستر**RCC_CR**
- منبع **SYSCLK** با بیت های **SW** و **SWS** در رجیستر**RCC_CFGR**
- فعال کردن **Prefetch** در بیت **PRFTBE** در رجیستر**FLASH_ACR**

- تنظیم two wait state در بیت های LATENCY رجیستر FLASH_ACR
- نوشتن تابع تنظیم کلاک میکروکنترلر با رجیسترها
- پین MCO و خروجی کلاک ، بیت های MCO در رجیستر RCC_CFGR
- تنظیم پین MCO و اندازه گیری کلاک PLL/2 با اسیلوسکوپ
- بررسی نحوه ی تنظیم کلاک توسط کتابخانه HAL
- ماکروهای فعالسازی کلاک در کتابخانه HAL
- نوشتن تابع فعالسازی کلاک پریفرال ها با پوینتر void

پریفرال EXTI ، شمردن تعداد دفعات فشرده شدن کلید با Polling

- سون سگمنت کاتد مشترک و انتخاب مقاومت ها
- مشخص کردن اعدادی که در رجیستر BSRR باید نوشته بشن برای نمایش اعداد 0 تا 9 در سون سگمنت
- انجام پروژه ی نمایش اعداد در سون سگمنت
- تنظیم پین در حالت ورودی دیجیتال ، راه اندازی Push button و مدار Debouncing
- بررسی interrupt و polling
- لاین های اینتراپت خارجی و ساختار پریفرال EXTI
- اینتراپت در لبه ی بالارونده، رجیستر EXTI_RTSR
- اینتراپت در لبه ی پایین روند، رجیستر EXTI_FTSR
- فعال کردن نرم افزاری اینتراپت های خارجی با رجیستر EXTI_SWIER
- فعال سازی اینتراپت های خارجی در رجیستر EXTI_IMR
- Pending request register
- تفاوت اینتراپت و ایونت
- فعال سازی event در رجیستر EXTI_EMR
- اتصال پین ها به لاین های اینتراپت با رجیستر های EXTICR در پریفرال AFIO
- پروژه شمردن تعداد دفعات فشرده شدن Push button و نمایش در Seven segment با برنامه نویسی رجیستری

پوینتر به تابع در زبانC

- معرفی ، تعریف و مقدار دهی پوینتر به تابع
- فراخوانی تابع با استفاده از پوینتر به تابع
- کاربرد های پوینتر به تابع
- پوینتر به تابع عضوی از استراکچر
- Object oriented programmingدر زبان سی با پوینتر به تابع
- تعریف و مقدار دهی اولیه ی آرایه ای از پوینتر ها به تابع
- آرایه از پوینترها به تابع برای انتخاب توابع ، جایگزینif

پریفرال NVIC ، فایل core_cm3.h و تنظیم اینتراپت ها

- حالت کاری thread mode وhandler mode
- اینتراپت های CPU و اینتراپت هایdevice
- جدول vector table و آدرس توابعIRQ
- فراخوانی تابع روتین اینتراپت
- بررسی وضعیت pending وactive
- معرفیcortex-m3 technical reference manual
- معرفیcortex-m3 programming manual
- اولویت و شماره اینتراپت ها
- فعال کردن اینتراپت های در رجیستر های ISERx در پریفرالNVIC
- غیر فعال کردن اینتراپت ها در رجیسترهای ICERx در پریفرالNVIC
- فعال کردن Pending در رجیسترهای ISPRx در پریفرالNVIC
- غیر فعال کردن Pending در رجیستر های ICPRx در پریفرالNVIC
- خوندن وضعیت Active در IABRx در پریفرالNVIC
- اولویت بندی اینتراپت ها در رجیسترهای IPRx در پریفرالNVIC
- بیت های Preemption priority وSub-priority

- تنظیم گروه بندی اولویت در بیت های PRIGROP
- بررسی IRQn_Type ، شماره گزاری اینتراپت ها در لایه ی نرم افزاری CMSIS
- معرفی فایل core_cm3.h
- بررسی استراکچر قالب رجیسترهای پریفرال NVIC در فایل core_cm3.h
- توابع فایل core_cm3.h برای تنظیم پریفرال NVIC
- بررسی تابع NVIC_EnbleIRQ
- محاسبه ی باقیمانده تقسیم بر 32 با AND کردن شماره اینتراپت با x1F0
- خارج قسمت تقسیم بر 32 با شیفت راست شماره اینتراپت
- پروژه تنظیم اینتراپت خارجی با استفاده از توابع فایل core_cm3.h
- گروه بندی اولیت با تابع NVIC_SetPriorityGrouping
- اولویت اینتراپت با تابع NVIC_SetPriority
- انکود کردن گروه بندی با تابع NVIC_EncodePriority
- خوندن Priority group با تابع NVIC_GetPriorityGrouping
- صفر کردن Flag interrupt
- vector table در فایل startup
- نوشتن توابع IRQHandler در فایل it.c
- تعریف آرایه ی از پوینترها به توابع برای مدیریت توابع callback اینتراپت های EXTI
- نوشتن تابع callback اینتراپت پین PB6 و مقدار دهی پوینتر به تابع
- تابع برای مدیریت اینتراپت های پریفرال EXTI
- ماکرو برای فراخوانی پوینترها به توابع callback
- پروژه اینتراپت خارجی پین PB6 و افزایش عدد نمایش داده شده در سون سگمنت با فشردن کلید

فصل هفتم: فلش، رم و لینکر اسکریپت

منابعی که در این فصل بررسی میشن:

- **stm32 flash memory programming manual**
- **Using ld**

نوشتن در فلش با رجیسترها

- معرفی منوال**stm32 flash memory programming manual**
- روش های **ICP** و **IAP** در پروگرم کردن فلش
- بررسی ساختار فلش **main memory , information block** و صفحه ها
- خوندن**flash**
- پریفرال**flash program and erase controller**
- مقدار رجیستر های کلید
- باز کردن قفل فلش ، رجیستر **FLASH_KEYR** و مقادیر کلید **1** و کلید **2**
- پیاده سازی الگوریتم پاک کردن یک صفحه در فلش
- یک کردن بیت **PER** ، نوشتن آدرس در **FAR** ، بیت**STRT**
- چک کردن بیت **lock** ، یک کردن بیت **PG** ، نوشتن داده ها با دسترسی **16** بیتی در فلش
- بیت**bsy**
- فرایند باز کردن قفل **option byte** و رجیستر **FLASH_OPTKEYR** ، چک کردن بیت**OPTWRE**
- الگوریتم پاک کردن **option byte** ، بیت**OPTER**
- نوشتن در **option bytes** و بیت**OPTPG**
- آپشن بایت **RDP** و محافظت فلش در برابر خوانده شدن
- مقدار **0 xff**و **RDPRT** و سایر مقادیر در آپشن بایت**RDP**
- بررسی نقشه رجیسترهای**flash**
- بایت کم ارزش و پر ارزش**option byte**
- تنظیمات آپشن بایت**USER**

- آپشن بایت های Data0 وData1
- WRP0تا WRP4 برای محافظت هر 4 صفحه در برابر نوشتن
- اصلاح بیت فیلد های فلش
- نوشتن کد باز کردن قفل فلش ، پاک کردن فلش و نوشتن در فلش با رجیسترها
- پاک کردن صفحه آخر فلش و ذخیره اعداد دلخواه ، بررسی حافظه در دیباگ
- ساخت آرایه در نمای expression برای مشاهده فلش در دیباگ
- پوینتر و استراکچر برای دسترسی به آپشن بایت ها
- نوشتن کد باز کردن قفل آپشن بایت ها ، پاک کردن آپشن بایت ها و نوشتن با رجیسترها
- ساخت متغیر کمکی برای دسترسی به بیت های آپشن بایت ها
- تنظیمات محافظت در برابر خواندن باRDP
- محافظت در برابر نوشتن باWRP
- تغییر آپشن بایت های RDP و WRP و باز کردن قفل فلش با نرم افزار st link utility

بوت لودرSTM32

- معرفی منوالAN2606
- بررسی سخت افزار مورد نیاز برای استفاده از bootloader ، پینهای PA9 وPA10
- اتصالات مبدل usart to usb برای استفاده ازstm32 bootloader
- تنظیمات نرم افزار STM32CubeProgrammer برای استفاده ازbootloader
- بررسی تنظیمات پین های bootmode برای استفاده ازbootloader

داینامیک مموری آلوکیشن

- dynamic memory allocationو منطقهheap
- مقایسه دینامیک و استاتیکmemory allocation
- کتابخانه stdlib.h از فایلthe red hat newlib C library
- ورودی و خروجی و نحوه عملکرد تابعmalloc

- نوع متغیر size_t
- مثال کلاس دانشجویان (آرایه ای از آدرس های استراکچرهای مربوط به دانشجو)
- ساخت آرایه ای از پوینتر ها برای ذخیره سازی دانشجویان
- آرایه ای از پوینترها ورودی به تابع ، دو روش
- پرینت کلاس دانشجویان
- نحوه صحیح مقدار دهی به رشته با تابع strcpy
- محاسبه تعداد اعضای آرایه با sizeof
- کاربرد null pointer در مثال دانشجویان
- استفاده از تابع malloc برای ذخیره دانشجویان جدید
- چک کردن خروجی تابع malloc
- حافظه heap در دیباگ
- ورودی ها و خروجی های تابع calloc
- تفاوت کاربرد calloc و malloc
- صفر کردن حافظه توسط تابع calloc
- اهمیت تابع calloc برای آرایه ای از پوینترها
- ساخت آرایه با طول متغیر
- ذخیره سازی اندازه آرایه با طول متغیر
- ورودی ها ، خروجی ها و نحوه عملکرد realloc
- تابع realloc و آرایه با طول متغیر
- نوشتن تابع برای صفر کردن حافظه بعد از realloc
- ورودی ها و خروجی ها تابع free و نحوه ی عملکرد
- استفاده از تابع free برای آزاد کردن حافظه

لینکر اسکریپت

- فایل لینکر اسکریپت و زبان برنامه نویسی linker command language

- کامنت در لینکر اسکریپت
- مروری بر وظایف ، ورودی های لینکر و خروجی لینکر
- وظایف فایل لینکر اسکریپت
- معرفی منوال using ld
- داکیومنت : using ld فصل اول مقدمه
- command line optionهای لینکر اسکریپت در تنظیمات پروژه درSTM32CubeIDE
- فراخوانی نرم افزار لینکر با کامندld
- بررسی آپشن o- برای تغییر نام فایل o. خروجی
- آپشن Wl- برای آپشن های لینکر در زمان فراخوانی غیر مستقیم
- ورودی های فراخوانی نرم افزارlinker
- آپشن T- برای معرفی فایل linker script به لینکر
- gc-sections–برای پاک کردن سکشن های استفاده نشده
- آپشن start-group– و end-group– برای معرفی فایل های آرشیو کتابخانه های استاندارد
- lc-وlm-
- ویژگی های executable , readable , allocatable , loadable برای سکشن ها
- آدرس های VMA و LMA برای هر سکشن
- بررسی آدرس های VMA و LMA برای سکشنdata.
- فرمت لینکر اسکریپت ، ساختار کلی فایل لینکر اسکریپت
- کامند ها در لینکر اسکریپت ، کلمات کلیدی و مقدار دهی به سمبل
- بررسی یک فایل linker script ساده
- حداقل ها در کامندSECTIONS
- کاربرد سمبل location counter در کامندSECTIONS
- سمبل ها در لینکر اسکریپت ، تعریف و مقدار دهی سمبل
- نوع اکسپرشن ها در لینکر اسکریپت
- ثابت ها در لینکر اسکریپت

- پسوند های k و m
- استفاده از توابع لینکر اسکریپت داخل اکسپرشن
- ORIGIN
- تابعLENGTH
- تعریف مناطق حافظه با کامندmemory
- مشخص کردن اسم ، مبدا و حجم منطقه های حافظه
- ویژگی های حافظه در کامندmemory : executable , read-only , writable
- اصلاح حجم flash در پنجره build analyzer با اصلاح کامندmemory
- کامند ENTRY و مشخص کردن تابع Reset_Handler به عنوان نقطه شروع برنامه
- سایر روش های مشخص کردن نقطه ی شروع برنامه
- کامند SECTIONS و وظایف
- sections-commandها
- توضیحات سکشن خروجی
- output-section-command
- توضیحات سکشن ورودی
- الگوهای رشته ها ، wildcard patterns
- کاراکتر استریسک وwildcard patterns
- علامت سوال وwildcard patterns
- استفاده از wildcard در توضیحات سکشن ورودی
- مثال اول تغییر لینکر اسکریپت: قرار دادن تابع در منطقه جدید حافظه فلش
- پاتیشن بندی فلش و اضافه کردن سکشن جدید در کامند مموری
- تعریف سکشن خروجی جدید در کامدن سکشن برای تابع
- قرار دادن تابع در سکشن ورودی دلخواه با اتریبوت سکشن
- مثال دوم تغییر لینکر اسکریپت: قرار دادن ثابت در منطقه جدید حافظه فلش
- پاتیشن بندی فلش و اضافه کردن سکشن جدید در کامند مموری
- تعریف سکشن خروجی جدید در کامند سکشن برای ثابت

- قرار دادن ثابت در سکشن ورودی دلخواه با اتریبیوت سکشن

فصل هشتم: اسمبلی ARM

منابعی که در این فصل بررسی میشن:

- **ARM®v7-M Architecture Reference Manual**
- **Using as**
- **Cortex-m3 programming manual_PM0056**

اسمبلی آرم

- معرفی **inline assembly** ، نوشتن اسمبلی اینستراکشن ها در فایل**c**.
- **Basic asm**و**extended asm**
- معرفی و نوشتن اینستراکشن ، نمانیک و آپراند ها
- منوال**architecture arm v7**
- آرکیتکچر پروفایل ها**A , R , M**
- اینستراکشن ست**arm , thumb**
- انواع متغیر برای**cpu**
- بررسی مقدار اینستراکشن **mov** و انکودینگ
- محاسبه ی کد باینری اینستراکشن**mov**
- اینستراکشن **immediate** و**register**
- بررسی تفاوت انکودینگ ها در اینستراکشن**mov**
- عملیات اینستراکشن ، سودو کد و سودو فانکشن
- سودو کد اینستراکشن**mov**
- **imm32**و**zeroextend**
- رجیستر **Rd** در انکودینگ های**mov**
- فیلد**qualifier**
- رجیستر **APSR** و وضعیت اجرای برنامه

- تعریف و کاربرد فلگ ها
- n، شناسایی عدد منفی
- فلگ z ، شناسایی صفر
- آپدیت کردن فلگ با S ، بررسی setflag در انکودینگ های mov
- تست آپدیت کردن فلگ های N و Z با اینستراکشن mov
- نوشتن عدد منفی در رجیستر cpu
- اینستراکشن mvn
- اجرای شرطی اینستراکشن ها
- کاربرد اینستراکشن if then و it block
- شرایط اینستراکشن های it block ، x , y , z
- بررسی انکودینگ اینستراکشن it
- کاندیشن ها و نمانیک اکستنشن ، کاندیشن اول در it block
- T و e در it instruction
- شرطی اجرا کردن اینستراکشن mov
- فیلد کاندیشنال
- کاندیشن ها در it block
- اینستراکشن cmp و انکودینگ ها
- cmp و فلگ z
- بررسی عملکرد و انکودینگ اینستراکشن add(immediate)
- عملکرد و انکودینگ اینستراکشن add(register)
- شیفت در اینستراکشن add
- انکودینگ شیفت ، نوع و مقدار شیفت
- انواع شیفت ها با مثال lsl , lsr , asr
- carry در add
- فلگ over flow در add

اینستراکشن های Load وStore

- معرفی اینستراکشنldr , ldrh , ldrb
- عملوند های اینستراکشنldr (immediate)
- اینستراکشنldr (literal)
- حالت آدرس دادنPC relative
- labelدر اینستراکشنldr (literal)
- معرفی رفرنس منوالUsing as
- معرفی دیرکتیو ها
- سینتکس و کاربرد دیرکتیوequ.
- لود کردن اعداد 32 بیتی در رجیسترهای CPU
- بخش literal pool برای ذخیره سازی ثابت ها بعد از توابع
- بررسی کد نوشته شده توسط کامپایلر برای مقدار دهی متغیر ها
- اینستراکشنstr (immediate)
- بررسی اینستراکشنstr (register)
- دسترسی به اعضای آرایه و استراکچر و محاسبه آدرس در اسمبلی
- اینستراکشن ldr و str از نوع immediate برای دسترسی به اعضای آرایه و استراکچر در زبان اسمبلی

کد اسمبلی در فایل های c. همراه با ورودی و خروجی ، extended asm

- بررسی سینتکس extended asm در منوال using the gnu compiler
- عملوند های خروجی درextended asm
- عملوند های ورودی درextended asm
- نوشتن مثال هایextended asm
- دسترسی به متغیر های زبان سی از اسمبلی با آدرس
- asm symbolic name
- قید های ورودی هایextended asm

- قید های خروجی های extended asm
- قید رجیستر r
- constraint modifier های = و+
- expression در عملوند های ورودی
- مفهوم constraint
- قید whitespace
- قید m ، memory operand
- o constraint
- دسترسی به متغیر های زبان سی از اسمبلی بصورت مستقیم
- قید های اختصاصی خانواده ARM
- constraint I برای ثابت های هشت بیتی

اینستراکشن های branch و push ، pop

- اینستراکشن B ، جامپ به label
- بررسی اسمبلی کد if else if نوشته شده توسط کامپایلر
- پیاده سازی if else if در اسمبلی با cmp و b
- conditional branch و beq
- اینستراکشن شرطی bpl
- تاثیر اینستراکشن های cmp و subs روی فلگ carry
- تفریق در اینستراکشن های cmp و subs
- شرط hi و اینستراکشن شرطی bhi
- بررسی شرط ls و اینستراکشن شرطی bls
- do{}while در زبان اسمبلی آرم
- while{} در زبان اسمبلی آرم
- switch case در زبان اسمبلی آرم
- اینستراکشن branch with link ، bl

- بررسی کاربرد اینستراکشن bl در فراخوانی توابع
- عملکرد اینستراکشن bl وlink register
- اینستراکشن branch and exchange ، bx
- تغییرinstruction execution state
- کاربرد bx در برگشت از توابع
- بررسی مدیریت استک در اسمبلی
- اینستراکشنpush
- رجیسترstack pointer
- نقش رجیستر r7 در مدیریت استک
- اینستراکشنpop
- جامپ با تغییر رجیستر program counter با استفاده از اینستراکشنpop
- بررسی ترتیب قرار گیری و خروج مقدار رجیستر های cpu در حافظه استک با اینستراکشن های push و pop
- bitcountدر عملیات push وpop
- MemAدر عملیات push وpop
- link registerدر استک