4. processor is also called the term CPU

Processor=processor core + surrounding processor peripherals.

20.operational mode of processor

Processor gives 2 operational modes: thread mode（线程模式） and handler mode（处理程序模式）

1. All application code is under thread mode also called user mode
2. All the exceptions handlers or interrupt handler will run under the Handler mode.
3. processor always starts with Thread mode
4. whenever the core meets the system exception or any interrupts then the core will change to handler mode.

22.access level of processor

Processor offers 2 access levels privileged access level(PAL by default) and Non- privileged access level(NPAL),PAL full access to restricted registers, NPAL may not have access to some of restricted registers.

Handler mode is PAL, thread mode can be changed to NPAL. Use the control register of the processor to switch the access level.

23.core register 16 page of CortexM4

①R0 to R12 registers are for general purpose.

②all the core registers are 32 bit wide

③the register R13 is called SP(堆栈指针)

④R14 is called LR 存储函数，子程序，exception的return信息。Used to hold the return information during function call and exception handling.

1. R15是程序计数器，called PC which holds the address of the next instruction to be executed.

Discussion about T bit of the EPSR. If T bit is 1, processor thinks the next instruction is from Thumb ISA. If T is 0, the instruction is from ARM ISA.

（1）PC是程序计数器，存储将要执行的指令地址

（2）LR是链接寄存器，是ARM处理器中一个有特殊用途的寄存器，当调用函数时，返回地址即PC的值被保存到LR中（mov lr,pc）。程序跳转（子程序调用，中断跳转）后，arm自动在该寄存器中存入原程序（未跳转）的下一条指令的地址。就是跳转去了一个函数以后，这个函数运行完，返回到哪在继续运行。

26. 第23条里的register都是non-memory mapped register.他们没有地址。

27.ARM GCC inline assembly coding 用于访问处理器的核心寄存器 core register

使用理由’ move the content of C variable data to ARM register R0

格式: \_\_asm volatile(“MOV R0,R1”);如果要一下写多个指令，那么格式

General format: \_\_asm volatile(code:output operand list: input operand list:clobber list);

\_\_asm volatile(

“LDR R0,[R1]\n\t”

“ADD R1,R0\n\t”); 中间没有逗号，都用双引号隔开

28.

内存读取，或者内存加载\_\_asm volatile(“LDR R0,[R1]”);把R1加载到R0里

相加\_\_asm volatile(“ADD R0,R0,R1”);把R1加R0的值加到R0上（第一个参数）

\_\_asm volatile(“STR R0,[R1]”);将结果存储回寄存器R1指向的存储位置

29.GCC-inline assembly coding3

Example1: move the content of ‘C’ variable ‘val’ to ARM register R0

\_\_asm volatile(“MOV R0,%0”::”r”(val)); 这里的”r” 是指使用寄存器工作

Operand indexing using%,sign followed by a digit %0 refers to the first operand, %1 refers to the second and so forth

Example2: move the content of CONTROL register to ‘C’ variable “control\_reg”

To read control register use MRS

Uint32\_t control\_reg;

\_\_asm volatile(“MRS %0,CONTROL”:”=r”(control\_reg)::);

这里的等号是 write-only operand,usually used for all output operands.

Example3:copy the content of ‘C’ variable var1 to var2

\_\_asm(“MOV %0,%1”:”=r”(var2):”r”(var1));

Example4: copy the contents of a pointer into another variable

Int p1, \*p2;

\_\_asm volatile(“LDR %0,[%1]”: “=r”(p1): “r”(p2)); //p1=\*p2

34. T bit of the EPSR

①various ARM processors support ARM-Thumb interworking, can switch between ARM and Thum state.

②If T bit is 1, processor thinks the next instruction is from Thumb ISA. If T is 0, the instruction is from ARM ISA.

③the cortex Mx processor does not support the ARM state. The value of T bit must always be 1. If not, result in usage fault.

④ The lsb(bit0) of the program counter(PC) is linked to T bit. When you load a value or an address in to PC the bit[0] of the value is loaded into the T-bit.

36.bus protocols and bus interface

AMBA specification supports several bus protocols.

AHB Lite(AMBA High-performance bus) used for the main bus interface, very fast

and APB(AMBA Peripheral Bus), low speed. Used for PPB access

タイムライン

低い精度で自動的に生成された説明

ARM M4有4个总线接口，所有接口都基于AHB协议，有I-BUS,D-bus, S-bus and PPB.

D和I是用于接受代码区数据和指令的，

S是与SRAM以及peripheral, Ext

PPB is private peripheral bus region. Includes the NVIC, system timer and system control block.

39.introduction to stack memory

①stack memory is part of main memory(RAM) for temporary storage of data.

②last in first out. The stack can be accessed using PUSH and POP or memory manipulation (LD, STR)

③stack pointer register (SP, R13)

テキスト

自動的に生成された説明

タイムライン

自動的に生成された説明

Malloc之类的用Heap

In ARM Cortex Mx processor stack consumption model is Full Descending(FD)

41.

グラフィカル ユーザー インターフェイス

自動的に生成された説明

42.グラフィカル ユーザー インターフェイス, テキスト, アプリケーション, Word

自動的に生成された説明

グラフィカル ユーザー インターフェイス, テキスト, アプリケーション

自動的に生成された説明

グラフィカル ユーザー インターフェイス, テキスト, アプリケーション, 手紙, メール

自動的に生成された説明

43.

テキスト, アプリケーション

自動的に生成された説明

47.exception model

グラフィカル ユーザー インターフェイス, テキスト, アプリケーション, Word

自動的に生成された説明

51.NVIC

グラフィカル ユーザー インターフェイス, テキスト, アプリケーション, Word

自動的に生成された説明

NVIC是用于处理外部中断的（比如SPI中断）

ダイアグラム

自動的に生成された説明

52.NVIC register

ISER是中断使能寄存器，PDF的第P220页，用于设置或者启用中断

ICER PDF的第P221页，用于禁用中断

ISPR force interrupts into the pending state.

IABR,223页，哪一位为1，就表示处理器当前正在处理该中断

53.

テキスト, 手紙

自動的に生成された説明

グラフィカル ユーザー インターフェイス, アプリケーション, Teams

自動的に生成された説明

54.peripheral interrupt exercise.

①manually pend the pending bit for the USART3 IRQ number in NVIC 地址在219页

Uint32\_t \*pISPR1 =(uint32\_t\*)0XE000E204;

\*pISPR1 |=(1<<7)

②Enable the USART3 IRQ number in　NVIC

Uint32\_t \*pISER1 =(uint32\_t\*)0XE000E104;

\*pISER1 |=(1<<7)

③USART3 ISR

Void USART3\_IRQHandler(void){printf(in USART3 isr)}

在startup文件里找到USART3\_IRQHandler

55.interrupt priority

For ARM cortex Mx processor, lesser the priority value, higher the priority(urgency)

グラフィカル ユーザー インターフェイス が含まれている画像

自動的に生成された説明

用IPR来配置中断优先级

ダイアグラム

自動的に生成された説明

60个IPR，每一个IPR里面每8bits控制一个中断，然后8bit里只有implemented可以控制中断

56.pre-empt and sub priority

グラフィカル ユーザー インターフェイス, テキスト, アプリケーション

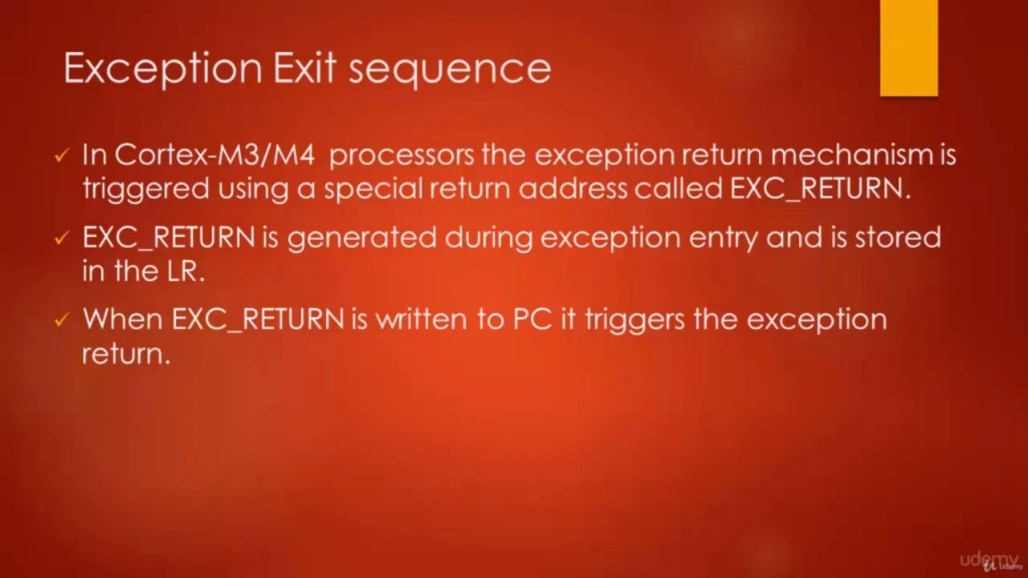
自動的に生成された説明

57.interrupt priority exercise

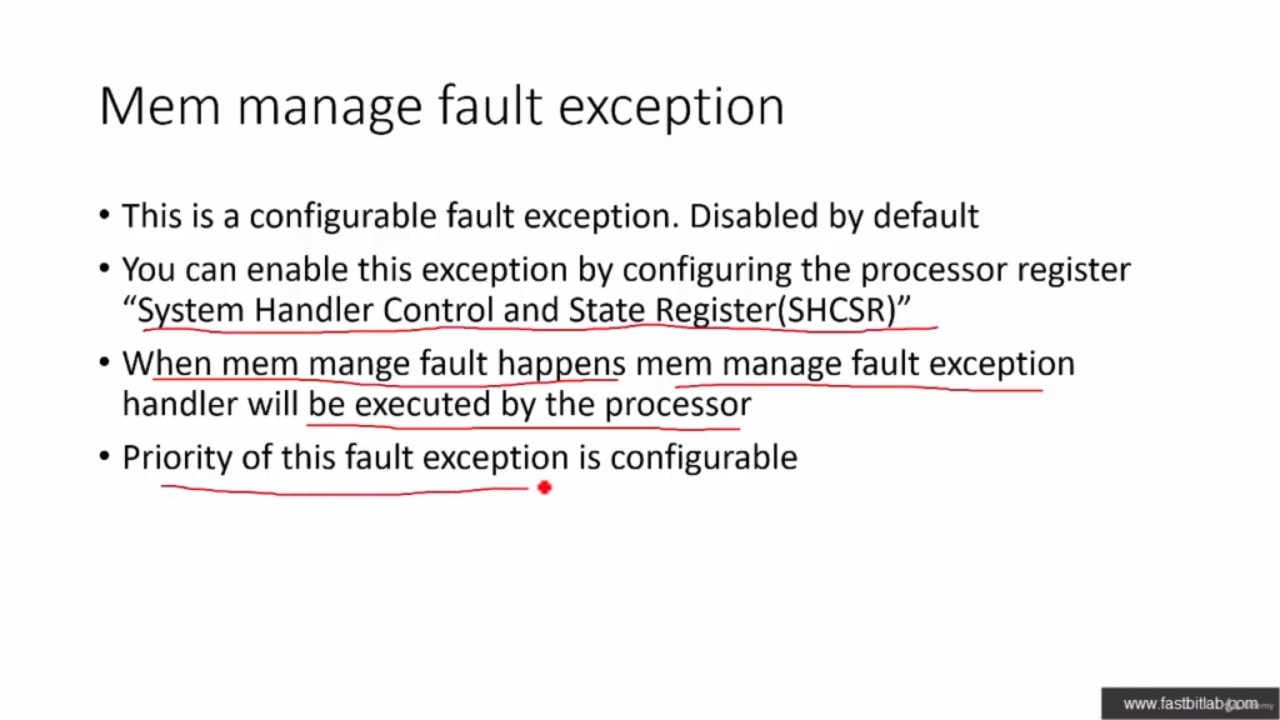
グラフィカル ユーザー インターフェイス, テキスト, アプリケーション

自動的に生成された説明

先设置中断优先级，在进行中断

59.

63.other configurable faults



ダイアグラム

自動的に生成された説明

启用第16位去enable这个memory exception，第17位是bus exception