Subject
Date

سیا صدامت ۹۴۳۱۰۴۴

لطفا اول درس دیر برگردانده!

۱- الف) مشخصات میکروکنترلر به نام STM32F3 ساخت شرکت STM Microelectronics

Features:

- ARM 32-bit Cortex-M3 CPU core
  - 72 MHz maximum frequency
  1.25 DMIPS/MHz performance at 0 wait state memory access
- Memories
  - 64 or 128 kbytes of Flash memory
  - 20 kbytes of SRAM
- clock, reset, supply management
  - 2.0 to 3.6 application supply and I/Os
  - POR, PDR, and programmable voltage detector (PVD)
  - 4-to-16 MHz crystal oscillator
  - internal 8 MHz factory-trimmed RC
  - internal 40kHz RC
  - PLL for cpu clock
  - 32 kHz oscillator for RTC
- low-power
  - Sleep, stop and standby modes
  - V_{BAT} supply for RTC and backup regs.

ب) ARM 32-bit Cortex M3 CPU

- 2x12-bit, 1MS A/D Convertors
  - Conversion range: 0 to 3.6 V
  - dual-sample and hold capability
  - temperature sensor
- DMA
  - 7 channel DMA Controller
  - peripheral supported timers, ADC, SPIs, I^2Cs and USARTs
- up to 80 fast I/O ports
  - 26/37/51/80 I/Os, all mappables on 16 external interrupt vectors and almost all 5 V-tolerant
- debug mode
  - serial wire debug (SWD) & JTAG interfaces
- 7 timers
  - three 16-bit timers, each with up to 4 IC/OC/PWM or pulse counter and quadrature encoder input
  - 16-bit, motor control PWM timer with dead-time generation
  - 2 watchdog timers
  - sysTick timer 24-bit downcounter

P4PCO

Subject
Date

- up to Communication interfaces
  - up to 2x I^2C interfaces (SMBus/PMBus)
  - up to 3 USARTs (ISO 7816 interfaces LIN, IrDA capability, modem control)
  - up to 2 SPIs (18 Mbit/s)
  - CAN interfaces (2.0B Active)
  - USB 2.0 full-speed inteface
- CRC calculation unit, 96-bit unique ID
- packages are ECOPACK

شکل داده شده بلوک دیاگرام عمومی این خانواده‌ی دستگاه است. از قسمت‌های مختلفی تشکیل شده است:

ARM Cortex-M3 از آخرین نسخه‌های پردازنده‌های ARM است (برای سیستم‌های embeded). یک پلتفرم با قیمت کم دارد که نیازهای MCU implementation را دارد. تعداد pin های کم و توان کم دارد.

محاسبات کامپیوتری قوی دارد و جواب دستگاه به وقفه‌ها پیشرفته است.

- embeded flash memory: 64 or 128 Kbytes of embeded Flash is available for storing programs and data.
- CRC (cyclic redundancy check) calculation unit: is used to get a CRC code from a 32-bit data word and a fixed generator polynominal. verifing data transmittion or storage integrity.
- embeded SRAM: 20 Kbytes of embeded SRAM accessed (read/write) at cpu clock speed with a wait states.

2.3 Overview

2.3.1 ARM® Cortex®-M3 core with embedded Flash and SRAM

The ARM® Cortex®-M3 processor is the latest generation of ARM processors for embedded systems. It has been developed to provide a low-cost platform that meets the needs of MCU implementation, with a reduced pin count and low-power consumption, while delivering outstanding computational performance and an advanced system response to interrupts.

The ARM® Cortex®-M3 32-bit RISC processor features exceptional code-efficiency, delivering the high-performance expected from an ARM core in the memory size usually associated with 8- and 16-bit devices.

The STM32F103xx performance line family having an embedded ARM core, is therefore compatible with all ARM tools and software.

Figure 1 shows the general block diagram of the device family.

2.3.2 Embedded Flash memory

64 or 128 Kbytes of embedded Flash is available for storing programs and data.

2.3.3 CRC (cyclic redundancy check) calculation unit

The CRC (cyclic redundancy check) calculation unit is used to get a CRC code from a 32-bit data word and a fixed generator polynomial.

Among other applications, CRC-based techniques are used to verify data transmission or storage integrity. In the scope of the EN/IEC 60335-1 standard, they offer a means of verifying the Flash memory integrity. The CRC calculation unit helps compute a signature of the software during runtime, to be compared with a reference signature generated at link-time and stored at a given memory location.

2.3.4 Embedded SRAM

Twenty Kbytes of embedded SRAM accessed (read/write) at CPU clock speed with 0 wait states.

2.3.5 Nested vectored interrupt controller (NVIC)

The STM32F103xx performance line embeds a nested vectored interrupt controller able to handle up to 43 maskable interrupt channels (not including the 16 interrupt lines of Cortex®-M3) and 16 priority levels.

- Closely coupled NVIC gives low-latency interrupt processing
- Interrupt entry vector table address passed directly to the core
- Closely coupled NVIC core interface
- Allows early processing of interrupts
- Processing of *late arriving* higher priority interrupts
- Support for tail-chaining
- Processor state automatically saved
- Interrupt entry restored on interrupt exit with no instruction overhead

This hardware block provides flexible interrupt management features with minimal interrupt latency.

2.3.6 External interrupt/event controller (EXTI)

The external interrupt/event controller consists of 19 edge detector lines used to generate interrupt/event requests. Each line can be independently configured to select the trigger event (rising edge, falling edge, both) and can be masked independently. A pending register maintains the status of the interrupt requests. The EXTI can detect an external line with a pulse width shorter than the Internal APB2 clock period. Up to 80 GPIOs can be connected to the 16 external interrupt lines.

2.3.7 Clocks and startup

System clock selection is performed on startup, however the internal RC 8 MHz oscillator is selected as default CPU clock on reset. An external 4-16 MHz clock can be selected, in which case it is monitored for failure. If failure is detected, the system automatically switches back to the internal RC oscillator. A software interrupt is generated if enabled. Similarly, full interrupt management of the PLL clock entry is available when necessary (for example on failure of an indirectly used external crystal, resonator or oscillator).

Several prescalers allow the configuration of the AHB frequency, the high-speed APB (APB2) and the low-speed APB (APB1) domains. The maximum frequency of the AHB and the high-speed APB domains is 72 MHz. The maximum allowed frequency of the low-speed APB domain is 36 MHz. See *Figure 2* for details on the clock tree.

2.3.8 Boot modes

At startup, boot pins are used to select one of three boot options:

- Boot from User Flash
- Boot from System Memory
- Boot from embedded SRAM

The boot loader is located in System Memory. It is used to reprogram the Flash memory by using USART1. For further details please refer to AN2606.

2.3.9 Power supply schemes

- V_{DD} = 2.0 to 3.6 V: external power supply for I/Os and the internal regulator. Provided externally through V_{DD} pins.
- V_{SSA}, V_{DDA} = 2.0 to 3.6 V: external analog power supplies for ADC, reset blocks, RCs and PLL (minimum voltage to be applied to V_{DDA} is 2.4 V when the ADC is used). V_{DDA} and V_{SSA} must be connected to V_{DD} and V_{SS}, respectively.
- V_{BAT} = 1.8 to 3.6 V: power supply for RTC, external clock 32 kHz oscillator and backup registers (through power switch) when V_{DD} is not present.

For more details on how to connect power pins, refer to *Figure 14: Power supply scheme*.

2.3.10 Power supply supervisor

The device has an integrated power-on reset (POR)/power-down reset (PDR) circuitry. It is always active, and ensures proper operation starting from/down to 2 V. The device remains

in reset mode when V_{DD} is below a specified threshold, $V_{POR/PDR}$, without the need for an external reset circuit.

The device features an embedded programmable voltage detector (PVD) that monitors the V_{DD}/V_{DDA} power supply and compares it to the V_{PVD} threshold. An interrupt can be generated when V_{DD}/V_{DDA} drops below the V_{PVD} threshold and/or when V_{DD}/V_{DDA} is higher than the V_{PVD} threshold. The interrupt service routine can then generate a warning message and/or put the MCU into a safe state. The PVD is enabled by software.

Refer to *Table 11: Embedded reset and power control block characteristics* for the values of $V_{POR/PDR}$ and V_{PVD}.

2.3.11 Voltage regulator

The regulator has three operation modes: main (MR), low-power (LPR) and power down.

- MR is used in the nominal regulation mode (Run)
- LPR is used in the Stop mode
- Power down is used in Standby mode: the regulator output is in high impedance: the kernel circuitry is powered down, inducing zero consumption (but the contents of the registers and SRAM are lost)

This regulator is always enabled after reset. It is disabled in Standby mode, providing high impedance output.

2.3.12 Low-power modes

The STM32F103xx performance line supports three low-power modes to achieve the best compromise between low-power consumption, short startup time and available wakeup sources:

- **Sleep** mode

  In Sleep mode, only the CPU is stopped. All peripherals continue to operate and can wake up the CPU when an interrupt/event occurs.

- **Stop** mode

  The Stop mode achieves the lowest power consumption while retaining the content of SRAM and registers. All clocks in the 1.8 V domain are stopped, the PLL, the HSI RC and the HSE crystal oscillators are disabled. The voltage regulator can also be put either in normal or in low-power mode.
  The device can be woken up from Stop mode by any of the EXTI line. The EXTI line source can be one of the 16 external lines, the PVD output, the RTC alarm or the USB wakeup.

- **Standby** mode

  The Standby mode is used to achieve the lowest power consumption. The internal voltage regulator is switched off so that the entire 1.8 V domain is powered off. The PLL, the HSI RC and the HSE crystal oscillators are also switched off. After entering Standby mode, SRAM and register contents are lost except for registers in the Backup domain and Standby circuitry.

  The device exits Standby mode when an external reset (NRST pin), an IWDG reset, a rising edge on the WKUP pin, or an RTC alarm occurs.

Note: The RTC, the IWDG, and the corresponding clock sources are not stopped by entering Stop or Standby mode.

2.3.13 DMA

The flexible 7-channel general-purpose DMA is able to manage memory-to-memory, peripheral-to-memory and memory-to-peripheral transfers. The DMA controller supports circular buffer management avoiding the generation of interrupts when the controller reaches the end of the buffer.

Each channel is connected to dedicated hardware DMA requests, with support for software trigger on each channel. Configuration is made by software and transfer sizes between source and destination are independent.

The DMA can be used with the main peripherals: SPI, I^2C, USART, general-purpose and advanced-control timers TIMx and ADC.

2.3.14 RTC (real-time clock) and backup registers

The RTC and the backup registers are supplied through a switch that takes power either on V_{DD} supply when present or through the V_{BAT} pin. The backup registers are ten 16-bit registers used to store 20 bytes of user application data when V_{DD} power is not present.

The real-time clock provides a set of continuously running counters which can be used with suitable software to provide a clock calendar function, and provides an alarm interrupt and a periodic interrupt. It is clocked by a 32.768 kHz external crystal, resonator or oscillator, the internal low-power RC oscillator or the high-speed external clock divided by 128. The internal low-power RC has a typical frequency of 40 kHz. The RTC can be calibrated using an external 512 Hz output to compensate for any natural crystal deviation. The RTC features a 32-bit programmable counter for long-term measurement using the Compare register to generate an alarm. A 20-bit prescaler is used for the time base clock and is by default configured to generate a time base of 1 second from a clock at 32.768 kHz.

2.3.15 Timers and watchdogs

The medium-density STM32F103xx performance line devices include an advanced-control timer, three general-purpose timers, two watchdog timers and a SysTick timer.

Table 4 compares the features of the advanced-control and general-purpose timers.

Table 4. Timer feature comparison

Timer	Counter resolution	Counter type	Prescaler factor	DMA request generation	Capture/compare channels	Complementary outputs
TIM1	16-bit	Up, down, up/down	Any integer between 1 and 65536	Yes	4	Yes
TIM2, TIM3, TIM4	16-bit	Up, down, up/down	Any integer between 1 and 65536	Yes	4	No

Advanced-control timer (TIM1)

The advanced-control timer (TIM1) can be seen as a three-phase PWM multiplexed on 6 channels. It has complementary PWM outputs with programmable inserted dead-times. It can also be seen as a complete general-purpose timer. The 4 independent channels can be used for

- Input capture
- Output compare
- PWM generation (edge- or center-aligned modes)
- One-pulse mode output

If configured as a general-purpose 16-bit timer, it has the same features as the TIMx timer. If configured as the 16-bit PWM generator, it has full modulation capability (0-100%).

In debug mode, the advanced-control timer counter can be frozen and the PWM outputs disabled to turn off any power switch driven by these outputs.

Many features are shared with those of the general-purpose TIM timers which have the same architecture. The advanced-control timer can therefore work together with the TIM timers via the Timer Link feature for synchronization or event chaining.

General-purpose timers (TIMx)

There are up to three synchronizable general-purpose timers embedded in the STM32F103xx performance line devices. These timers are based on a 16-bit auto-reload up/down counter, a 16-bit prescaler and feature 4 independent channels each for input capture/output compare, PWM or one-pulse mode output. This gives up to 12 input captures/output compares/PWMs on the largest packages.
The general-purpose timers can work together with the advanced-control timer via the Timer Link feature for synchronization or event chaining. Their counter can be frozen in debug mode. Any of the general-purpose timers can be used to generate PWM outputs. They all have independent DMA request generation.

These timers are capable of handling quadrature (incremental) encoder signals and the digital outputs from 1 to 3 hall-effect sensors.

Independent watchdog

The independent watchdog is based on a 12-bit downcounter and 8-bit prescaler. It is clocked from an independent 40 kHz internal RC and as it operates independently of the main clock, it can operate in Stop and Standby modes. It can be used either as a watchdog to reset the device when a problem occurs, or as a free-running timer for application timeout management. It is hardware- or software-configurable through the option bytes. The counter can be frozen in debug mode.

Window watchdog

The window watchdog is based on a 7-bit downcounter that can be set as free-running. It can be used as a watchdog to reset the device when a problem occurs. It is clocked from the main clock. It has an early warning interrupt capability and the counter can be frozen in debug mode.

SysTick timer

This timer is dedicated for OS, but could also be used as a standard downcounter. It features:

- A 24-bit downcounter
- Autoreload capability
- Maskable system interrupt generation when the counter reaches 0
- Programmable clock source

2.3.16 I²C bus

Up to two I²C bus interfaces can operate in multimaster and slave modes. They can support standard and fast modes.

They support dual slave addressing (7-bit only) and both 7/10-bit addressing in master mode. A hardware CRC generation/verification is embedded.

They can be served by DMA and they support SM Bus 2.0/PM Bus.

2.3.17 Universal synchronous/asynchronous receiver transmitter (USART)

One of the USART interfaces is able to communicate at speeds of up to 4.5 Mbit/s. The other available interfaces communicate at up to 2.25 Mbit/s. They provide hardware management of the CTS and RTS signals, IrDA SIR ENDEC support, are ISO 7816 compliant and have LIN Master/Slave capability.

All USART interfaces can be served by the DMA controller.

2.3.18 Serial peripheral interface (SPI)

Up to two SPIs are able to communicate up to 18 Mbits/s in slave and master modes in full-duplex and simplex communication modes. The 3-bit prescaler gives 8 master mode frequencies and the frame is configurable to 8 bits or 16 bits. The hardware CRC generation/verification supports basic SD Card/MMC modes.

Both SPIs can be served by the DMA controller.

2.3.19 Controller area network (CAN)

The CAN is compliant with specifications 2.0A and B (active) with a bit rate up to 1 Mbit/s. It can receive and transmit standard frames with 11-bit identifiers as well as extended frames with 29-bit identifiers. It has three transmit mailboxes, two receive FIFOs with 3 stages and 14 scalable filter banks.

2.3.20 Universal serial bus (USB)

The STM32F103xx performance line embeds a USB device peripheral compatible with the USB full-speed 12 Mbs. The USB interface implements a full-speed (12 Mbit/s) function interface. It has software-configurable endpoint setting and suspend/resume support. The dedicated 48 MHz clock is generated from the internal main PLL (the clock source must use a HSE crystal oscillator).

2.3.21 GPIOs (general-purpose inputs/outputs)

Each of the GPIO pins can be configured by software as output (push-pull or open-drain), as input (with or without pull-up or pull-down) or as peripheral alternate function. Most of the GPIO pins are shared with digital or analog alternate functions. All GPIOs are high current-capable.

The I/Os alternate function configuration can be locked if needed following a specific sequence in order to avoid spurious writing to the I/Os registers.

I/Os on APB2 with up to 18 MHz toggling speed.

2.3.22 ADC (analog-to-digital converter)

Two 12-bit analog-to-digital converters are embedded into STM32F103xx performance line devices and each ADC shares up to 16 external channels, performing conversions in single-shot or scan modes. In scan mode, automatic conversion is performed on a selected group of analog inputs.

Additional logic functions embedded in the ADC interface allow:

- Simultaneous sample and hold
- Interleaved sample and hold
- Single shunt

The ADC can be served by the DMA controller.

An analog watchdog feature allows very precise monitoring of the converted voltage of one, some or all selected channels. An interrupt is generated when the converted voltage is outside the programmed thresholds.

The events generated by the general-purpose timers (TIMx) and the advanced-control timer (TIM1) can be internally connected to the ADC start trigger, injection trigger, and DMA trigger respectively, to allow the application to synchronize A/D conversion and timers.

2.3.23 Temperature sensor

The temperature sensor has to generate a voltage that varies linearly with temperature. The conversion range is between 2 V < V_{DDA} < 3.6 V. The temperature sensor is internally connected to the ADC12_IN16 input channel which is used to convert the sensor output voltage into a digital value.

2.3.24 Serial wire JTAG debug port (SWJ-DP)

The ARM SWJ-DP Interface is embedded. and is a combined JTAG and serial wire debug port that enables either a serial wire debug or a JTAG probe to be connected to the target. The JTAG TMS and TCK pins are shared with SWDIO and SWCLK, respectively, and a specific sequence on the TMS pin is used to switch between JTAG-DP and SW-DP.

Subject:
Date

ب) 3 دستور العمل:

• یک دستور برای انجام عملیات ریاضی: فرض کنید می خواهیم چند عدد را با هم جمع کنیم می توانیم carry

را هم در نظر بگیریم (ADC) یا در نظر نگیریم (ADD). مثال) ADD R_2, R_1

محتوای R_2 را به R_1 اضافه کن و نتیجه را در R_2 قرار بده.

این عملیات معمولا روی ثبات ها انجام می گیرند به همین دلیل عملوندهای دستورات ریاضی، ثبات یا عدد هستند نه عنصر حافظه. SP و PC مستثنی

می توان شرط هایی را برای جمع در نظر گرفت مثلا ADC با carry جمع می کند.

• یک دستور برای جابجایی: عملوند R_1 را در R_2 کپی می کند. مثال) MOV R2, R1

MVN, operand2 را می گیرد و not آن را به جای RD جایگزین می کند. MOV {S} {cond} Rd, operand2

در MOV می توانیم از SP و PC استفاده کنیم. op2 باید رجیستری ثابت باشد و نباید تغییر کند. MVN {S} {cond} Rd, operand2

• یک دستور برای صدا زدن زیرروال: ① مانع از به ترتیب اجرا شدن برنامه می شود. DSB {cond}

به این صورت که دستورهایی که بعد از DSB می آیند تا وقتی که دستور DSB اجرا نشود (زیرروال متوقف شده) اجرا نمی شوند. flag ها را

تغییر نمی دهد. DSB; data synchronisation barrier

مثال ② یک label را به عنوان عملوند می گیرد که متعلق به زیرروال است. BL: branch with link

آدرس دستور بعدی در LR می نویسد. (the link reg, R14)

پایپ لاین پردازنده را خالی می کند و همه ی دستورهایی که به دنبال ISB را از حافظه ISB {cond}

یا cache دوباره fetch می شوند (بعد از اینکه دستور ISB کامل شود) ISB; instruction synchronisation barrier

③ SVC: supervisor call

④ WFE ← برای یک event صبر می کند ⑤ WFI ← برای یک interrupt صبر می کند.

ج) FPU (floating point unit): FPU شامل add، subtract،

divide، multiply & accumulate، square root operations است، همین طور تبدیل

بین fixed xxxxx-point format و floating-point data، floating point constant را انجام می دهد.

P4PCO

Subject:
Date

شامل 32 تا رجیستر است که می‌توانند به صورت 16 تا دوطه‌ای برای load و store و move استفاده شوند.

CP ACR
- offset : 0X88
- reset value : 0X0000000
- Coprocessor access control register

FPCCR
- offset : 0X04
- reset value : 0XC000000
- returns Fpu control data

FpCAR , FPSCR , FP D SCR ← در ادامه بیشتر

برای فعال کردن FPU : باید یکی از دستورات مقابل را اجرا کنید که دسترسی را در ادامه می‌بندد.

4.6 Floating point unit (FPU)

The Cortex-M4F FPU implements the FPv4-SP floating-point extension.

The FPU fully supports single-precision add, subtract, multiply, divide, multiply and accumulate, and square root operations. It also provides conversions between fixed xxxxx-point and floating-point data formats, and floating-point constant instructions.

The FPU provides floating-point computation functionality that is compliant with the ANSI/IEEE standard 754-2008, IEEE standard for Binary Floating-Point Arithmetic, referred to as the IEEE 754 standard.

The FPU contains 32 single-precision extension registers, which you can also access as 16 doubleword registers for load, store, and move operations.

Table 55 shows the floating-point system registers in the Cortex-M4F system control block (SCB). The base address of the additional registers for the FP extension is 0xE000 ED00.

Table 55. Cortex-M4F floating-point system registers

Address	Name	Type	Reset	Description
0xE000ED88	CPACR	RW	0x00000000	*Section 4.6.1: Coprocessor access control register (CPACR) on page 252*
0xE000EF34	FPCCR	RW	0xC0000000	*Section 4.6.2: Floating-point context control register (FPCCR) on page 252*
0xE000EF38	FPCAR	RW	-	*Section 4.6.3: Floating-point context address register (FPCAR) on page 254*
0xE000EF3C	FPDSCR	RW	0x00000000	*Section 4.6.5: Floating-point default status control register (FPDSCR) on page 256*
-	FPSCR	RW	-	*Section 4.6.4: Floating-point status control register (FPSCR) on page 254*

The following sections describe the floating-point system registers whose implementation is specific to this processor.

Note: For more details on the IEEE standard and floating-point arithmetic (IEEE 754), refer to the AN4044 Application note. Available from website www.st.com.

4.6.1 Coprocessor access control register (CPACR)

Address offset (from SCB): 0x88

Reset value: 0x0000000

Required privilege: Privileged

The CPACR register specifies the access privileges for coprocessors.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Reserved								CP11		CP10		Reserved			
								rw		rw					
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved															

Bits 31:24 Reserved. Read as Zero, Write Ignore.

Bits 23:20 **CPn:** [2n+1:2n] for n values 10 and 11. Access privileges for coprocessor n. The possible values of each field are:

- 0b00: Access denied. Any attempted access generates a NOCP UsageFault.
- 0b01: Privileged access only. An unprivileged access generates a NOCP fault.
- 0b10: Reserved. The result of any access is Unpredictable.
- 0b11: Full access.

Bits 19:0 Reserved. Read as Zero, Write Ignore.

4.6.2 Floating-point context control register (FPCCR)

Address offset: 0x04

Reset value: 0xC000000

Required privilege: Privileged

The FPCCR register sets or returns FPU control data.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
ASPEN	LSPEN	Reserved													
rw	rw														
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved							MONRDY	Reserved	BFRDY	MMRDY	HFRDY	THREAD	Reserved	USER	LSPACT
							rw		rw	rw	rw	rw		rw	rw

Bit 31 **ASPEN**: Enables CONTROL<2> setting on execution of a floating-point instruction. This results in automatic hardware state preservation and restoration, for floating-point context, on exception entry and exit.
0: Disable CONTROL<2> setting on execution of a floating-point instruction.
1: Enable CONTROL<2> setting on execution of a floating-point instruction.

Bit 30 **LSPEN**:
0: Disable automatic lazy state preservation for floating-point context.
1: Enable automatic lazy state preservation for floating-point context.

Bits 29:9 Reserved.

Bit 8 **MONRDY**:
0: DebugMonitor is disabled or priority did not permit setting MON_PEND when the floating-point stack frame was allocated.
1: DebugMonitor is enabled and priority permits setting MON_PEND when the floating-point stack frame was allocated.

Bit 7 Reserved.

Bit 6 **BFRDY**:
0: BusFault is disabled or priority did not permit setting the BusFault handler to the pending state when the floating-point stack frame was allocated.
1: BusFault is enabled and priority permitted setting the BusFault handler to the pending state when the floating-point stack frame was allocated.

Bit 5 **MMRDY**:
0: MemManage is disabled or priority did not permit setting the MemManage handler to the pending state when the floating-point stack frame was allocated.
1: MemManage is enabled and priority permitted setting the MemManage handler to the pending state when the floating-point stack frame was allocated.

Bit 4 **HFRDY**:
0: Priority did not permit setting the HardFault handler to the pending state when the floating-point stack frame was allocated.
1: Priority permitted setting the HardFault handler to the pending state when the floating-point stack frame was allocated.

Bit 3 **THREAD**:
0: Mode was not Thread Mode when the floating-point stack frame was allocated.
1: Mode was Thread Mode when the floating-point stack frame was allocated.

Bit 2 Reserved.

Bit 1 **USER**:
0: Privilege level was not user when the floating-point stack frame was allocated.
1: Privilege level was user when the floating-point stack frame was allocated.

Bit 1 **LSPACT**:
0: Lazy state preservation is not active.
1: Lazy state preservation is active. floating-point stack frame is allocated but saving state to it is deferred.

4.6.3 Floating-point context address register (FPCAR)

Address offset: 0x08

Reset value: 0x0000000

Required privilege: Privileged

The FPCAR register holds the location of the unpopulated floating-point register space allocated on an exception stack frame.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
ADDRESS[31:16]															
rw															
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
ADDRESS[15:3]													Reserved		
rw															

Bits 31:3 ADDRESS: Location of unpopulated floating-point register space allocated on an exception stack frame.

Bits 2:0 Reserved. Read as Zero, Writes Ignored.

4.6.4 Floating-point status control register (FPSCR)

Address offset: Not mapped

Reset value: 0x0000000

Required privilege: Privileged

The FPSCR register provides all necessary user level control of the floating-point system.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
N	Z	C	V	Reserved	AHP	DN	FZ	RMode		Reserved					
rw	rw	rw	rw		rw	rw	rw	rw	rw						
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved								IDC	Reserved		IXC	UFC	OFC	DZC	IOC
								rw			rw	rw	rw	rw	rw

Bit 31 **N**: Negative condition code flag. Floating-point comparison operations update these flags. For more details on the result, refer to *Table 56*.

0: Operation result was positive, zero, greater than, or equal.

1: Operation result was negative or less than.

Bit 30 **Z**: Zero condition code flag. Floating-point comparison operations update these flags. For more details on the result, refer to *Table 56*.

0: Operation result was not zero.

1: Operation result was zero.

Bit 29 **C**: Carry condition code flag. Floating-point comparison operations update these flags. For more details on the result, refer to *Table 56*.

0: Add operation did not result in a carry bit or subtract operation resulted in a borrow bit.

1: Add operation resulted in a carry bit or subtract operation did not result in a borrow bit.

Bit 28 **V**: Overflow condition code flag. Floating-point comparison operations update this flag. For more details on the result, refer to *Table 56*.
0: Operation did not result in an overflow
1: Operation resulted in an overflow.

Bit 27 Reserved.

Bit 26 **AHP**: Alternative half-precision control bit:
0: IEEE half-precision format selected.
1: Alternative half-precision format selected.

Bit 25 **DN**: Default NaN mode control bit:
0: NaN operands propagate through to the output of a floating-point operation.
1: Any operation involving one or more NaNs returns the Default NaN.

Bit 24 **FZ**: Flush-to-zero mode control bit:
0: Flush-to-zero mode disabled. Behavior of the floating-point system is fully compliant with the IEEE 754 standard.
1: Flush-to-zero mode enabled.

Bits 23:22 **RMode**: Rounding Mode control field. The specified rounding mode is used by almost all floating-point instructions:
0b00: Round to nearest (RN) mode
0b01: Round towards plus infinity (RP) mode
0b10: Round towards minus infinity (RM) mode
0b11: Round towards zero (RZ) mode.

Bit 21:8 Reserved.

Bit 7 **IDC**: Input denormal cumulative exception bit. Cumulative exception bit for floating-point exception.
1: Indicates that the corresponding exception occurred since 0 was last written to it.

Bit 6:5 Reserved

Bit 4 **IXC**: Inexact cumulative exception bit. Cumulative exception bit for floating-point exception.
1: Indicates that the corresponding exception occurred since 0 was last written to it.

Bit 3 **UFC**: Underflow cumulative exception bit. Cumulative exception bit for floating-point exception.
1: Indicates that the corresponding exception occurred since 0 was last written to it.

Bit 2 **OFC**: Overflow cumulative exception bit. Cumulative exception bit for floating-point exception.
1: Indicates that the corresponding exception occurred since 0 was last written to it.

Bit 1 **DZC**: Division by zero cumulative exception bit. Cumulative exception bit for floating-point exception. 1: Indicates that the corresponding exception occurred since 0 was last written to it.

Bit 0 **IOC**: Invalid operation cumulative exception bit. Cumulative exception bit for floating-point exception. 1: Indicates that the corresponding exception occurred since 0 was last written to it.

Table 56. Effect of a Floating-point comparison on the condition flags

Comparison result	N	Z	C	V
Equal	0	1	1	0
Less than	1	0	0	0
Greater than	0	0	1	0
Unordered	0	0	1	1

4.6.5 Floating-point default status control register (FPDSCR)

Address offset: 0x0C

Reset value: 0x0000000

Required privilege: Privileged

The FPDSCR register holds the default values for the floating-point status control data.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Reserved					AHP	DN	FZ	RMode		Reserved					
					rw	rw	rw	rw	rw						
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved															

Bits 31:27 Reserved, must be kept cleared.

Bit 26 **AHP:** Default value for FPSCR.AHP

Bit 25 **DN:** Default value for FPSCR.DN

Bit 24 **FZ:** Default value for FPSCR.FZ

Bits 23:22 **RMode:** Default value for FPSCR.RMode

Bits 21:0 Reserved, must be kept cleared.

4.6.6 Enabling the FPU

The FPU is disabled from reset. You must enable it before you can use any floating-point instructions.

The example shows an example code sequence for enabling the FPU in both privileged and user modes. The processor must be in privileged mode to read from and write to the CPACR.

Example

```
                ; CPACR is located at address 0xE000ED88
LDR.W   R0, =0xE000ED88
                ; Read CPACR
LDR     R1, [R0]
                ; Set bits 20-23 to enable CP10 and CP11 coprocessors
ORR     R1, R1, #(0xF << 20)
                ; Write back the modified value to the CPACR
STR     R1, [R0]; wait for store to complete
DSB
;reset pipeline now the FPU is enabled
ISB
```

4.6.7 Enabling and clearing FPU exception interrupts

The FPU exception flags are generating an interrupt through the interrupt controller. The FPU interrupt is globally controlled through the interrupt controller.

A mask bit is also provided in the System Configuration Controller (SYSCFG), allowing to enable/disable individually each FPU flag interrupt generation.

Note: In STM32F4xx devices there is no individual mask and the enable/disable of the FPU interrupts is done at interrupt controller level. As it occurs very frequently, the IXC exception flag is not connected to the interrupt controller in these devices , and cannot generate an interrupt. If needed, it must be managed by polling.

Clearing the FPU exception flags depends on the FPU context save/restore configuration:

- No floating-point register saving: when Floating-point context control register (FPCCR) Bit 30 LSPEN=0 and Bit 31 ASPEN=0.
  You must clear interrupt source in Floating-point Status and Control Register (FPSCR).
  Example:

```
register uint32_t fpscr_val = 0;
fpscr_val  = __get_FPSCR();
{ check exception flags }
fpscr_val &= (uint32_t)~0x8F; // Clear all exception flags
__set_FPSCR(fpscr_val);
```

- Lazy save/restore: when Floating-point context control register (FPCCR)
  Bit 30 LSPEN=1 and Bit 31 ASPEN=X.
  In the case of lazy floating-point context save/restore, a dummy read access should be made to Floating-point Status and Control Register (FPSCR) to force state preservation and FPSCR clear.
  Then handle FPSCR in the stack.
  Example:

```
register uint32_t fpscr_val = 0;
register uint32_t reg_val = 0;
reg_val =  __get_FPSCR();        //dummy access
fpscr_val=*(__IO uint32_t*)(FPU->FPCAR +0x40);
{ check exception flags }
fpscr_val &= (uint32_t)~0x8F ;   // Clear all exception flags
*(__IO uint32_t*)(FPU->FPCAR +0x40)=fpscr_val;
__DMB() ;
```

- Automatic floating-point registers save/restore: when Floating-point context control register (FPCCR)
  Bit 30 LSPEN=0 and Bit 31 ASPEN=1.
  In case of automatic floating-point context save/restore, a read access should be made to Floating-point Status and Control Register (FPSCR) to force clear.
  Then handle FPSCR in the stack.
  Example:

```
// FPU Exception handler
void FPU_ExceptionHandler(uint32_t lr, uint32_t sp)
{
register uint32_t fpscr_val;
   if(lr == 0xFFFFFFE9)
   {
```

```
    sp = sp + 0x60;
  }
  else if(lr == 0xFFFFFFED)
  {
    sp = __get_PSP() + 0x60 ;
  }
fpscr_val = *(uint32_t*)sp;
{ check exception flags }
fpscr_val &= (uint32_t)~0x8F ;   // Clear all exception flags
*(uint32_t*)sp = fpscr_val;
__DMB() ;
}
// FPU IRQ Handler
void __asm FPU_IRQHandler(void)
{
IMPORT  FPU_ExceptionHandler
MOV   R0, LR             // move LR to R0
MOV   R1, SP             // Save SP to R1 to avoid any modification to
                         // the stack pointer from FPU_ExceptionHandler
VMRS R2, FPSCR           // dummy read access, to force clear
B     FPU_ExceptionHandler
BX    LR
}
```

۴) روش‌های بسته‌بندی این میکروکنترلر: براساس انطباق‌پذیری محیطی به ۲ دسته packing تقسیم می‌شود،

- VFQFPN 6x6 mm
- UFQFPN48 7x7 mm
- BGA100 10x10 mm
- UFBGA100 7x7 mm
- BGA64 5x5 mm
- LQFP100 14x14 mm
- LQFP64 10x10 mm
- LQFP48 7x7 mm

۵) محدوده‌های دمایی کارکرد را بنویسید؟ دمای محیط: -40 to +85 °C / -40 to +105°C

دمای اتصال: -40 to +125 °C

۶) ولتاژ کاری این میکروکنترلر چند ولت است؟

2.0 V to 3.6 V

PAPCO

Subject:
Date

2. الف) A4826H ب) E7490H ج) FFFF0H د) 10009H

می دانیم آدرس شروع بخش پشته اختیاری است و باید عددی بخش پذیر بر 16 باشد یعنی باید 4 بیت کم ارزش آن 0 باشد.

با این توضیحات فقط گزینه های "ب" و "ج" می توانند صحیح باشند ولی باید توجه داشت آدرس شروع باید از F0000H بیشتر باشد پس "ب" صحیح است.

3- آدرس فیزیکی = C89A0H , CS = C4000H , آدرس منطقی ؟

آدرس منطقی = آدرس فیزیکی − (CS)0 − آدرس پایه سگمنت

= C89A0H − C4000H = 49A0H

4. IP = 2010H و آدرس فیزیکی بالای پشته = E9500H محدوده ی بالا و پایین سگمنت پشته؟

SP = 2010H

(SS)0 + SP = top of stack

(SS)0 = E9500H − 2010H = E74F0H محدوده ی بالا

برای به دست آوردن محدوده ی پایین کافی است آدرس را با FFFFH جمع بزنیم،

E74F0H + FFFFH = F74EFH