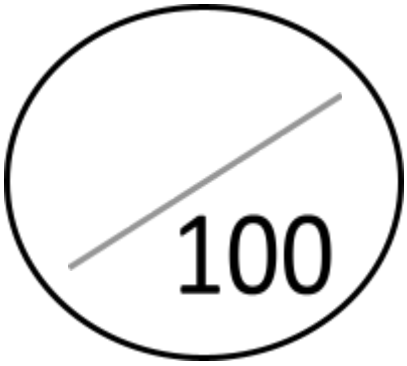
Total Score



Question 1 (2 points)

Development boards are most useful for:

A. Cost-optimized final product designs

B. Permanent industrial deployments

C. Prototyping and learning

D. Replacing discrete MCUs in all cases

Question 2 (2 points)

What is the main purpose of the `<stdint.h>` library?

A. To improve floating-point performance

B. To provide fixed-size integer types

C. To declare volatile variables

D. To manage heap memory

Question 3 (2 points)

In C, what does the `sizeof()` operator return?

A. The current value of a variable

B. The number of bytes used by a type, array, or struct

C. The address of a variable in memory

D. The maximum stack size

Question 4 (2 points)

What will the expression `7 / 2` evaluate to in C (integer division)?

A. 3

B. 3.5

C. 4

D. 2

Question 5 (2 points)

Which memory region stores program instructions?

A. Heap

B. Stack

C. Text

D. BSS

Question 6 (2 points)

In the **stm32f0xx.h** header file, it is stated that the interrupt associated with **GPIO pin 1** is at **offset 7** in the vector table.If the interrupt vector table begins at address **0x0000 0000**, what is the address in the vector table where the **ISR address for EXTI1** will be found?

A. `0x0000 001C`

B. `0x0000 0020`

C. `0x0000 0014`

D. `0x0000 0028`

Question 7 (2 points)

Which of the following correctly creates a mask for **bit 6** of a 32-bit register?

A. `uint32_t mask = (1 << 6);`

B. `uint32_t mask = (6 << 1);`

C. `uint32_t mask = (1 >> 6);`

D. `uint32_t mask = (0x6);`

Use the following code segment to answer the subsequent questions:

```c
#include <stdio.h>

int global_counter = 10;
static int static_flag = 1;

int main(void) {
    int local_value = 5;
    static int local_static = 20;
    char* buffer = malloc(50);
    return 0;
}
```

Question 8 (2 points)

Which memory location does `buffer` belong to?

A. Data

B. BSS

C. Stack

D. Heap

Question 9 (2 points)

Which memory location does `global_counter` belong to?

A. Data

B. BSS

C. Stack

D. Heap

Question 10 (2 points)

Which memory location does `static_flag` belong to?

A. Data

B. BSS

C. Stack

D. Heap

Question 11 (2 points)

Which memory location does the data allocated by malloc() belong to?

E. Data

F. BSS

G. Stack

H. Heap

Question 12 (2 points)

An STM32 microcontroller is using a **12-bit ADC** with a **reference voltage of 3.3 V**. If the input analog voltage is **1.65 V**, what digital value will the ADC output?

A. 1023

B. 2047

C. 2482

D. 4095

Question 13 (4 points)

Which of the following struct implementations will minimize the amount of padding on a 32-bit machine?

A.
```c
typedef struct {
    uint16_t a;
    uint32_t b;
    uint8_t c;
    uint8_t d;
} myStruct_t;
```

B.
```c
typedef struct {
    uint16_t a;
    uint8_t b;
    uint32_t c;
    uint8_t d;
} myStruct_t;
```

C.
```c
typedef struct {
    uint32_t a;
    uint16_t b;
    uint8_t c;
    uint8_t d;
} myStruct_t;
```

D.
```c
typedef struct {
    uint8_t a;
    uint32_t b;
    uint16_t c;
    uint8_t d;
} myStruct_t;
```

E.
```c
typedef struct {
    uint8_t a;
    uint32_t b;
    uint8_t c;
    uint16_t d;
} myStruct_t;
```

For the following questions, please reference the included datasheet figures.

Question 14 (2 points)

Suppose the **base address** of **GPIOC** is given as: `0x4800 0800`. What is the absolute address of the GPIOC ODR register? (Reference the included datasheet)

1. `0x4800 0804`
2. `0x4800 0810`
3. `0x4800 0814`
4. `0x4800 0820`

Question 15 (2 points)

On STM32 microcontrollers, each GPIO pin's mode is controlled by two bits in the **MODER register**. If the mode bits for a pin are set to **11**, what does this configuration mean?

A. The pin outputs a constant mid-level reference voltage.

B. The pin is connected directly to the ADC/DAC circuitry, disabling digital input/output buffers.

C. The pin functions as both input and output simultaneously.

D. The pin is placed in a high-speed digital output mode.

Question 16 (2 points)

How would I set pin 3 to a low voltage using the BSRR register?

A. BSRR &= ~(1 << 3);

B. BSRR |= 1 << 3;

C. BSRR &= ~(1 << 19);

D. BSRR |= 1 << 19;

Question 17 (2 points)

According to the datasheet, what is the offset of the GPIOx IDR (Input Data Register) from the GPIO base address?

A. 0x08

B. 0x10

C. 0x14

D. 0x00

Question 18 (2 points)

What value must be written to the GPIOx_ODR register to set pin 5 high while leaving all other pins unchanged?

A. `GPIOx->ODR |= (1 << 5);`

B. `GPIOx->ODR = (1 << 5);`

C. `GPIOx->ODR &= ~(1 << 5);`

D. `GPIOx->ODR ^= (1 << 5);`

Question 19 (2 points)

If the GPIOx_PUPDR bits for pin 0 are configured as 10, what does this indicate?

A. No pull-up or pull-down

B. Pull-up resistor enabled

C. Pull-down resistor enabled

D. Output open-drain mode

Question 20 (2 points)

If the GPIOx_MODER bits for pin 7 are 01, what does this configuration mean?

A. Analog mode

B. General-purpose output mode

C. Alternate function mode

D. Input mode

Question 21 (2 points)

If you want to configure PC13 as an input with a pull-up resistor, which register(s) must be modified?

A. Only MODER

B. MODER and ODR

C. MODER and PUPDR

D. MODER and AFR

The table to the right is the memory contents of the STM32F091RC microcontroller after running the following code with a breakpoint set just prior to returning from the “foo()” function.

```
int foo(int a){
    int b = 2;
    b = a + b;

    return b;          <-- break
}

int main() {
    int x = 5;

    int y = foo(x);

    while(1);
}
```

The memory locations containing 0xXX indicate data that has not been initialized yet and contains junk data.This memory dump comes from an unoptimized build, where all registers are spilled on the stack.The STM32 is a little endian 32-bit processor and ints are 32 bits wide.

Memory	Address
0bXXXX XXXX	0x20007FE0
0bXXXX XXXX	0x20007FE1
0bXXXX XXXX	0x20007FE2
0bXXXX XXXX	0x20007FE3
0x05	0x20007FE4
0x00	0x20007FE5
0x00	0x20007FE6
0x00	0x20007FE7
0xF0	0x20007FE8
0x7F	0x20007FE9
0x00	0x20007FEA
0x20	0x20007FEB
0x39	0x20007FEC
0x01	0x20007FED
0x00	0x20007FEE
0x08	0x20007FEF
0x07	0x20007FF0
0x00	0x20007FF1
0x00	0x20007FF2
0x00	0x20007FF3
0x05	0x20007FF4
0x00	0x20007FF5
0x00	0x20007FF6
0x00	0x20007FF7
0xFF	0x20007FF8
0xFF	0x20007FF9
0xFF	0x20007FFA
0xFF	0x20007FFB
0x83	0x20007FFC
0x01	0x20007FFD
0x00	0x20007FFE
0x08	0x20007FFF

Question 22 (4 points)

Which address is associated with the variable x?

A. 0x20007FF1

B. 0x20007FE0

C. 0x20007FF4

D. 0x20007FF0

Question 23 (4 points)

In the current stack frame, which address is associated with the previous stack pointer?

A. 0x20007FE8

B. 0x20007FF8

C. 0x20007FFC

D. 0x20007FE4

Question 24 (4 points)

What is the return address of the current stack frame?

A. 0x08000139

B. 0xF07F0020

C. 0x08000183

D. 0xFFFFFFFF

Using the included datasheet clippings of register descriptions and the code segment below to answer the multiplier choice questions.

```
1  void GPIO_Init(void) {
2      // Enable clock for GPIOC and SYSCFG
3      RCC->AHBENR  |= RCC_AHBENR_GPIOCEN;
4      RCC->APB2ENR |= RCC_APB2ENR_SYSCFGEN;
5
6      GPIOC->MODER &= ~((3U << (0 * 2)) | (3U << (1 * 2)));
7      GPIOC->PUPDR &= ~((3U << (0 * 2)) | (3U << (1 * 2)));
8      GPIOC->PUPDR |=  ((1U << (0 * 2)) | (1U << (1 * 2)));
9
10     GPIOC->MODER &= ~((3U << (8 * 2)) | (3U << (9 * 2)));
11     GPIOC->MODER |=  ((1U << (8 * 2)) | (1U << (9 * 2)));
12
13     // Note: EXTICR[0] -> Register SYSCFG_EXTIRCR1
14     SYSCFG->EXTICR[0] &= ~(0xFU << (0 * 4));
15     SYSCFG->EXTICR[0] |=  (0x2U << (0 * 4));
16
17     EXTI->IMR  |= (1U << 0);
18     EXTI->RTSR |= (1U << 0);
19
20     NVIC_EnableIRQ(EXTI0_1_IRQn); // Enable EXTI0_1 interrupt in NVIC
21 }
```

Question 25 (2 points)

What is the purpose of the assignment on line10?

A. Set PC8 and PC9 to analog mode

B. Directly configure PC8 and PC9 as outputs

C. Clear the mode bits for PC8 and PC9 (reset to 00) in preparation for setting their mode

D. Enable pull-ups on PC8 and PC9

Question 26 (2 points)

What is being configured on line 15?

A. Route EXTI0 to PA0

B. Route EXTI0 to PB0

C. Route EXTI0 to PC0

D. Route EXTI0 to PD0

Question 27 (2 points)

What is being configured on line 18?

A. EXTI0 triggers on falling edge

B. EXTI0 triggers on rising edge

C. EXTI0 interrupt masked (disabled)

D. Level-sensitive trigger for EXTI0

Question 28 (2 points)

Given the context of the configurations being applied by this code, what is the most likely purpose of the code.

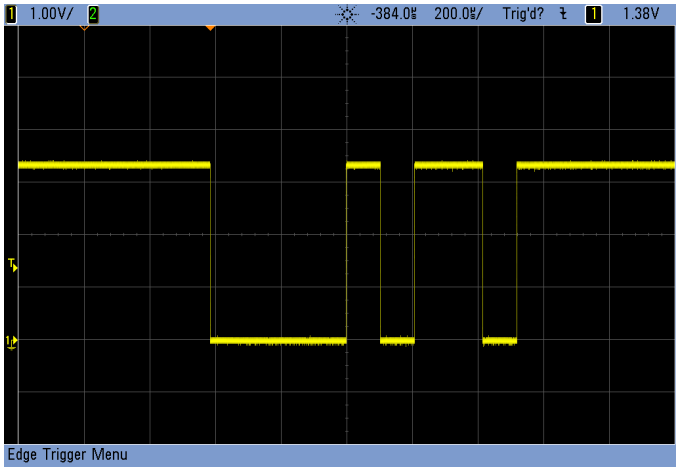
A. Set up PC8/PC9 as UART TX/RX and enable DMA for USART

B. Configure PC0 as an external interrupt source with pull-up and set PC8/PC9 as output pins (e.g., LEDs) to react in an ISR

C. Initialize an ADC on PC0 and PC1 and stream samples via interrupts

D. Configure a timer to toggle PC8/PC9 periodically without any external input

You have a microcontroller that is communicating with another device over UART. You are having trouble with communication so you decide to connect an oscilloscope to the receive pin on your microcontroller to validate that the UART frame is being properly transmitted. You capture a transmission and see the following image on the oscilloscope display:



The Oscilloscope is set to **200 microseconds per division** and **1 Volt per division**. You know only **one** character is being transmitted.

Question 29 (2 points)

If we assume that the UART peripheral is transmitting at a standard baud rate, what is the baud rate?

A. 9600

B. 10000

C. 4800

D. 19200

Question 30 (2 points)

What is the data being transmitted?

A. 0000 1011

B. 1101 0000

C. 0001 0110

D. 0110 1000

Question 31 (2 points)

If we assume the device has been configured with parity enabled, and assuming the data was transmitted without error, what can be said about the parity?

A. It's configured as even parity.

B. It's configured as odd parity.

C. It's impossible to determine what kind of parity is configured from the data presented on this display.

D. There is not enough room for the parity bit to be represented on this display.

Question 32 (2 points)

How many stop bits have been configured for this transmission?

A. 1

B. 2

C. 0

D. It's impossible to determine from this image.

Question 33 (8 points)

You're writing code for an embedded system where we want to conserve energy by reducing various power-consuming aspects of the system as it remains idle. Your device should automatically enter a deeper power-saving mode as it remains idle longer. Deeper modes must also perform the steps of all lighter modes.

Idle-time policy:

- **Mode 0: No power save** — idle < **30 s**
- **Mode 1: Light** — idle ≥ **30 s,** Action: dim the display to 25% brightness
- **Mode 2: Medium** — idle ≥ **120 s,** Actions: reduce CPU clock to "low" **and** do Mode 1
- **Mode 3: Deep** — idle ≥ **300 s,** Actions: shut down nonessential peripherals **and** do Modes 2 and 1

You'll implement this using a small, hypothetical API:

```
unsigned int get_idle_seconds(void);  // e.g., returns seconds since last user activity
void set_display_brightness(int percent);    // 0–100%
void reduce_clock_speed(const char *level);  // "medium", "low", etc.
void shutdown_nonessential_peripherals(void);
void restore_default(); // resets all default parameters.
```

Implement the behavior described above in the main function provided. Use good coding practice (identions, efficient use of branching, efficient use of memory, no magic numbers, etc.).

```
int main(){









}
```

8.4.1 GPIO port mode register (GPIOx_MODER) (x =A to F)

Address offset:0x00

Reset value: 0x2800 0000 for port A

Reset value: 0x0000 0000 for other ports

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
MODER15[1:0]		MODER14[1:0]		MODER13[1:0]		MODER12[1:0]		MODER11[1:0]		MODER10[1:0]		MODER9[1:0]		MODER8[1:0]	
rw	rw	rw	rw	rw	rw	rw	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
MODER7[1:0]		MODER6[1:0]		MODER5[1:0]		MODER4[1:0]		MODER3[1:0]		MODER2[1:0]		MODER1[1:0]		MODER0[1:0]	
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bits 31:0 **MODER[15:0][1:0]:** Port x configuration I/O pin y (y = 15 to 0)
These bits are written by software to configure the I/O mode.
00: Input mode (reset state)
01: General purpose output mode
10: Alternate function mode
11: Analog mode

8.4.4 GPIO port pull-up/pull-down register (GPIOx_PUPDR) (x = A to F)

Address offset: 0x0C

Reset value: 0x2400 0000 (for port A)

Reset value: 0x0000 0000 (for other ports)

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
PUPDR15[1:0]		PUPDR14[1:0]		PUPDR13[1:0]		PUPDR12[1:0]		PUPDR11[1:0]		PUPDR10[1:0]		PUPDR9[1:0]		PUPDR8[1:0]	
rw	rw	rw	rw	rw	rw	rw	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
PUPDR7[1:0]		PUPDR6[1:0]		PUPDR5[1:0]		PUPDR4[1:0]		PUPDR3[1:0]		PUPDR2[1:0]		PUPDR1[1:0]		PUPDR0[1:0]	
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bits 31:0 **PUPDR[15:0][1:0]:** Port x configuration I/O pin y (y = 15 to 0)
These bits are written by software to configure the I/O pull-up or pull-down
00: No pull-up, pull-down
01: Pull-up
10: Pull-down
11: Reserved

8.4.5 GPIO port input data register (GPIOx_IDR) (x = A to F)

Address offset: 0x10

Reset value: 0x0000 XXXX

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
IDR15	IDR14	IDR13	IDR12	IDR11	IDR10	IDR9	IDR8	IDR7	IDR6	IDR5	IDR4	IDR3	IDR2	IDR1	IDR0
r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r

Bits 31:16 Reserved, must be kept at reset value.

Bits 15:0 **IDR[15:0]:** Port x input data I/O pin y (y = 15 to 0)
These bits are read-only. They contain the input value of the corresponding I/O port.

8.4.6 GPIO port output data register (GPIOx_ODR) (x = A to F)

Address offset: 0x14

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
ODR15	ODR14	ODR13	ODR12	ODR11	ODR10	ODR9	ODR8	ODR7	ODR6	ODR5	ODR4	ODR3	ODR2	ODR1	ODR0
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bits 31:16 Reserved, must be kept at reset value.

Bits 15:0 **ODR[15:0]:** Port output data I/O pin y (y = 15 to 0)
These bits can be read and written by software.
Note: For atomic bit set/reset, the ODR bits can be individually set and/or reset by writing to the GPIOx_BSRR register (x = A..F).

8.4.7 GPIO port bit set/reset register (GPIOx_BSRR) (x = A to F)

Address offset: 0x18

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
BR15	BR14	BR13	BR12	BR11	BR10	BR9	BR8	BR7	BR6	BR5	BR4	BR3	BR2	BR1	BR0
w	w	w	w	w	w	w	w	w	w	w	w	w	w	w	w
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
BS15	BS14	BS13	BS12	BS11	BS10	BS9	BS8	BS7	BS6	BS5	BS4	BS3	BS2	BS1	BS0
w	w	w	w	w	w	w	w	w	w	w	w	w	w	w	w

Bits 31:16 **BR[15:0]:** Port x reset I/O pin y (y = 15 to 0)
These bits are write-only. A read to these bits returns the value 0x0000.
0: No action on the corresponding ODRx bit
1: Resets the corresponding ODRx bit
Note: If both BSx and BRx are set, BSx has priority.

Bits 15:0 **BS[15:0]:** Port x set I/O pin y (y = 15 to 0)
These bits are write-only. A read to these bits returns the value 0x0000.
0: No action on the corresponding ODRx bit
1: Sets the corresponding ODRx bit

9.1.2 SYSCFG external interrupt configuration register 1 (SYSCFG_EXTICR1)

Address offset: 0x08

Reset value: 0x0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
EXTI3[3:0]				EXTI2[3:0]				EXTI1[3:0]				EXTI0[3:0]			
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bits 31:16 Reserved, must be kept at reset value.

Bits 15:0 **EXTIx[3:0]**: EXTI x configuration bits (x = 0 to 3)
These bits are written by software to select the source input for the EXTIx external interrupt.
x000: PA[x] pin
x001: PB[x] pin
x010: PC[x] pin
x011: PD[x] pin
x100: PE[x] pin
x101: PF[x] pin
other configurations: reserved

11.3.1 Interrupt mask register (EXTI_IMR)

Address offset: 0x00
Reset value: 0x0FF4 0000 (STM32F03x devices)
0x7FF4 0000 (STM32F04x devices)
0x0F94 0000 (STM32F05x devices)
0x7F84 0000 (STM32F07x and STM32F09x devices)

Note: The reset value for the internal lines is set to '1' in order to enable the interrupt by default.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
IM31	IM30	IM29	IM28	IM27	IM26	IM25	IM24	IM23	IM22	IM21	IM20	IM19	IM18	IM17	IM16
rw	rw	rw	rw	rw	rw	rw	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
IM15	IM14	IM13	IM12	IM11	IM10	IM9	IM8	IM7	IM6	IM5	IM4	IM3	IM2	IM1	IM0
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bits 31:0 **IMx:** Interrupt Mask on line x (x = 31 to 0)
0: Interrupt request from Line x is masked
1: Interrupt request from Line x is not masked

11.3.3 Rising trigger selection register (EXTI_RTSR)

Address offset: 0x08
Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
RT31	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	RT22	RT21	RT20	RT19	Res.	RT17	RT16
rw									rw	rw	rw	rw		rw	rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
RT15	RT14	RT13	RT12	RT11	RT10	RT9	RT8	RT7	RT6	RT5	RT4	RT3	RT2	RT1	RT0
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bit 31 **RT31:** Rising trigger event configuration bit of line 31
0: Rising trigger disabled (for Event and Interrupt) for input line
1: Rising trigger enabled (for Event and Interrupt) for input line.

Bits 30:23 Reserved, must be kept at reset value.

Bits 22:19 **RTx:** Rising trigger event configuration bit of line x (x = 22 to 19)
0: Rising trigger disabled (for Event and Interrupt) for input line
1: Rising trigger enabled (for Event and Interrupt) for input line.

Bit 18 Reserved, must be kept at reset value.

Bits 17:0 **RTx:** Rising trigger event configuration bit of line x (x = 17 to 0)
0: Rising trigger disabled (for Event and Interrupt) for input line
1: Rising trigger enabled (for Event and Interrupt) for input line.

11.3.4 Falling trigger selection register (EXTI_FTSR)

Address offset: 0x0C
Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
FT31	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	FT22	FT21	FT20	FT19	Res.	FT17	FT16
rw									rw	rw	rw	rw		rw	rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
FT15	FT14	FT13	FT12	FT11	FT10	FT9	FT8	FT7	FT6	FT5	FT4	FT3	FT2	FT1	FT0
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bit 31 **FT31:** Falling trigger event configuration bit of line 31
0: Falling trigger disabled (for Event and Interrupt) for input line
1: Falling trigger enabled (for Event and Interrupt) for input line.

Bits 30:23 Reserved, must be kept at reset value.

Bits 22:19 **FTx:** Falling trigger event configuration bit of line x (x = 22 to 19)
0: Falling trigger disabled (for Event and Interrupt) for input line.
1: Falling trigger enabled (for Event and Interrupt) for input line.

Bit 18 Reserved, must be kept at reset value.

Bits 17:0 **FTx:** Falling trigger event configuration bit of line x (x = 17 to 0)
0: Falling trigger disabled (for Event and Interrupt) for input line.
1: Falling trigger enabled (for Event and Interrupt) for input line.

13.11.6 ADC sampling time register (ADC_SMPR)

Address offset: 0x14

Reset value: 0x0000 0000

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

Bits 31:3 Reserved, must be kept at reset value.

Bits 2:0 **SMP[2:0]:** *Sampling time selection*
These bits are written by software to select the sampling time that applies to all channels.
000: 1.5 ADC clock cycles
001: 7.5 ADC clock cycles
010: 13.5 ADC clock cycles
011: 28.5 ADC clock cycles
100: 41.5 ADC clock cycles
101: 55.5 ADC clock cycles
110: 71.5 ADC clock cycles
111: 239.5 ADC clock cycles
Note: The software is allowed to write this bit only when ADSTART = 0 (which ensures that no conversion is ongoing).

13.11.8 ADC channel selection register (ADC_CHSELR)

Address offset: 0x28

Reset value: 0x0000 0000

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

Bits 31:19 Reserved, must be kept at reset value.

Bits 18:0 **CHSELx**: Channel-x selection
These bits are written by software and define which channels are part of the sequence of channels to be converted.
0: Input Channel-x is not selected for conversion
1: Input Channel-x is selected for conversion

13.11.3 ADC control register (ADC_CR)

Address offset: 0x08

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
ADCAL	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
rs															
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	ADSTP	Res.	ADSTART	ADDIS	ADEN
											rs		rs	rs	rs

Bit 4 **ADSTP**: ADC stop conversion command
This bit is set by software to stop and discard an ongoing conversion (ADSTP Command).
It is cleared by hardware when the conversion is effectively discarded and the ADC is ready to accept a new start conversion command.
0: No ADC stop conversion command ongoing
1: Write 1 to stop the ADC. Read 1 means that an ADSTP command is in progress.
Note: Setting ADSTP to '1' is only effective when ADSTART = 1 and ADDIS = 0 (ADC is enabled and may be converting and there is no pending request to disable the ADC)

Bit 1 **ADDIS**: ADC disable command
This bit is set by software to disable the ADC (ADDIS command) and put it into power-down state (OFF state).
It is cleared by hardware once the ADC is effectively disabled (ADEN is also cleared by hardware at this time).
0: No ADDIS command ongoing
1: Write 1 to disable the ADC. Read 1 means that an ADDIS command is in progress.
Note: Setting ADDIS to '1' is only effective when ADEN = 1 and ADSTART = 0 (which ensures that no conversion is ongoing)

Bit 0 **ADEN**: ADC enable command
This bit is set by software to enable the ADC. The ADC is effectively ready to operate once the ADRDY flag has been set.
It is cleared by hardware when the ADC is disabled, after the execution of the ADDIS command.
0: ADC is disabled (OFF state)
1: Write 1 to enable the ADC.
Note: The software is allowed to set ADEN only when all bits of ADC_CR registers are 0 (ADCAL = 0, ADSTP = 0, ADSTART = 0, ADDIS = 0 and ADEN = 0)

13.11.4 ADC configuration register 1 (ADC_CFGR1)

Address offset: 0x0C

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	AWDCH[4:0]					Res.	Res.	AWDEN	AWDSGL	Res.	Res.	Res.	Res.	Res.	DISCEN
	rw	rw	rw	rw	rw			rw	rw						rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
AUTOFF	WAIT	CONT	OVRMOD	EXTEN[1:0]		Res.	EXTSEL[2:0]			ALIGN	RES[1:0]		SCANDIR	DMACFG	DMAEN
rw	rw	rw	rw	rw	rw		rw	rw	rw	rw	rw	rw	rw	rw	rw

Bit 16 **DISCEN**: Discontinuous mode
This bit is set and cleared by software to enable/disable discontinuous mode.
0: Discontinuous mode disabled
1: Discontinuous mode enabled
Note: It is not possible to have both discontinuous mode and continuous mode enabled: it is forbidden to set both bits DISCEN = 1 and CONT = 1.
The software is allowed to write this bit only when ADSTART bit is cleared (this ensures that no conversion is ongoing).

Bit 13 **CONT**: Single / continuous conversion mode
This bit is set and cleared by software. If it is set, conversion takes place continuously until it is cleared.
0: Single conversion mode
1: Continuous conversion mode
Note: It is not possible to have both discontinuous mode and continuous mode enabled: it is forbidden to set both bits DISCEN = 1 and CONT = 1.
The software is allowed to write this bit only when ADSTART bit is cleared (this ensures that no conversion is ongoing).

Bits 4:3 **RES[1:0]**: Data resolution
These bits are written by software to select the resolution of the conversion.
00: 12 bits
01: 10 bits
10: 8 bits
11: 6 bits
Note: The software is allowed to write these bits only when ADEN is cleared.