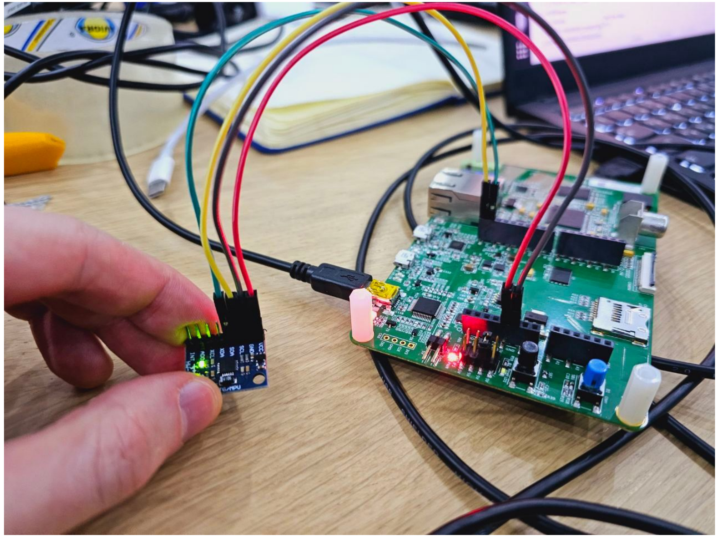
STM32F746g

HAL



Что вынести из этой лекции:

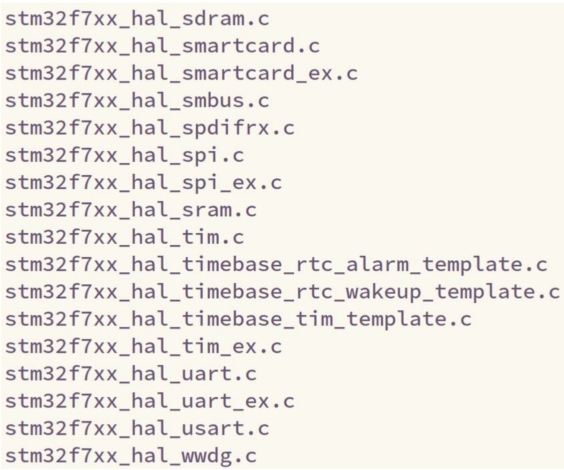
- SysTick Configuration
- Interrupts for GPIO
- UART на HAL
- Timer на HAL
- SPI, I2C на HAL

Drivers/STM32F7xx_HAL_Driver:

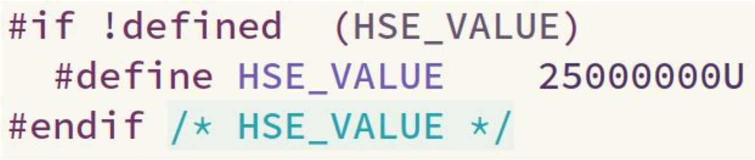
- Drivers/STM32F7xx_HAL_Driver/Inc/stm32f7xx_hal.h
  MACRO: STM32F746xx
  stm32f7xx_hal.h -> stm32f7xx_hal_conf.h

- Drivers/STM32F7xx_HAL_Driver/Inc/stm32f7xx_hal_conf_template.h
  MACRO: HAL_ADC_MODULE_ENABLED, HAL_GPIO_MODULE_ENABLED, etc…

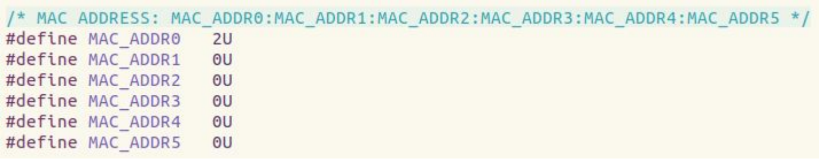
```
#ifdef HAL_ADC_MODULE_ENABLED
  #include "stm32f7xx_hal_adc.h"
#endif /* HAL_ADC_MODULE_ENABLED */
```



```
stm32f7xx_hal_sdram.c
stm32f7xx_hal_smartcard.c
stm32f7xx_hal_smartcard_ex.c
stm32f7xx_hal_smbus.c
stm32f7xx_hal_spdifrx.c
stm32f7xx_hal_spi.c
stm32f7xx_hal_spi_ex.c
stm32f7xx_hal_sram.c
stm32f7xx_hal_tim.c
stm32f7xx_hal_timebase_rtc_alarm_template.c
stm32f7xx_hal_timebase_rtc_wakeup_template.c
stm32f7xx_hal_timebase_tim_template.c
stm32f7xx_hal_tim_ex.c
stm32f7xx_hal_uart.c
stm32f7xx_hal_uart_ex.c
stm32f7xx_hal_usart.c
stm32f7xx_hal_wwdg.c
```



```
#if !defined  (HSE_VALUE)
  #define HSE_VALUE    25000000U
#endif /* HSE_VALUE */
```



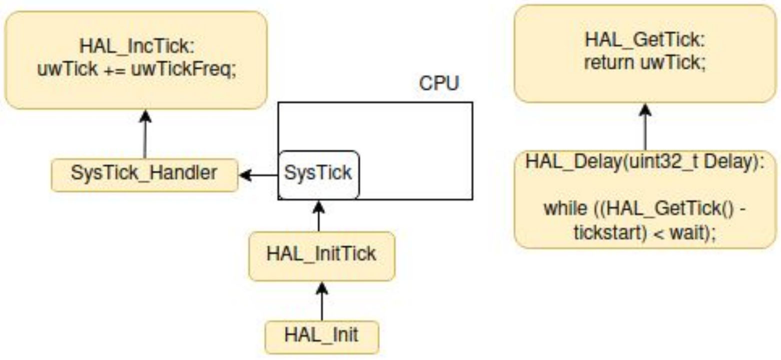
```
/* MAC ADDRESS: MAC_ADDR0:MAC_ADDR1:MAC_ADDR2:MAC_ADDR3:MAC_ADDR4:MAC_ADDR5 */
#define MAC_ADDR0   2U
#define MAC_ADDR1   0U
#define MAC_ADDR2   0U
#define MAC_ADDR3   0U
#define MAC_ADDR4   0U
#define MAC_ADDR5   0U
```

1. include “stm32f7xx_hal.h”
2. В stm32f7xx_hal_conf.h включаем макросы которыми задаем какие части библиотек мы используем
3. Добавляем в проект нужные *.c файлы из HAL

https://github.com/STMicroelectronics/stm32f7xx_hal_driver

https://www.st.com/content/ccc/resource/technical/document/user_manual/45/27/9c/32/76/57/48/b9/DM00189702.pdf/files/DM00189702.pdf/jcr:content/translations/en.DM00189702.pdf

HAL_Init	Configures the SysTick to generate an interrupt each 1 millisecond, Set NVIC Group Priority to 4.
HAL_InitTick	Configure the SysTick to have interrupt each 1 ms
HAL_Delay	This function provides delay (in milliseconds)
HAL_IncTick	This function is called to increment a global variable "uwTick" used as application time base.
HAL_GetTick	Provides a tick value in millisecond.



HAL_IncTick:
uwTick += uwTickFreq;

CPU

SysTick_Handler

SysTick

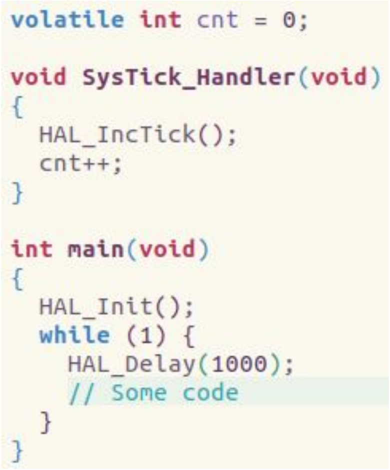
HAL_InitTick

HAL_Init

HAL_GetTick:
return uwTick;

HAL_Delay(uint32_t Delay):

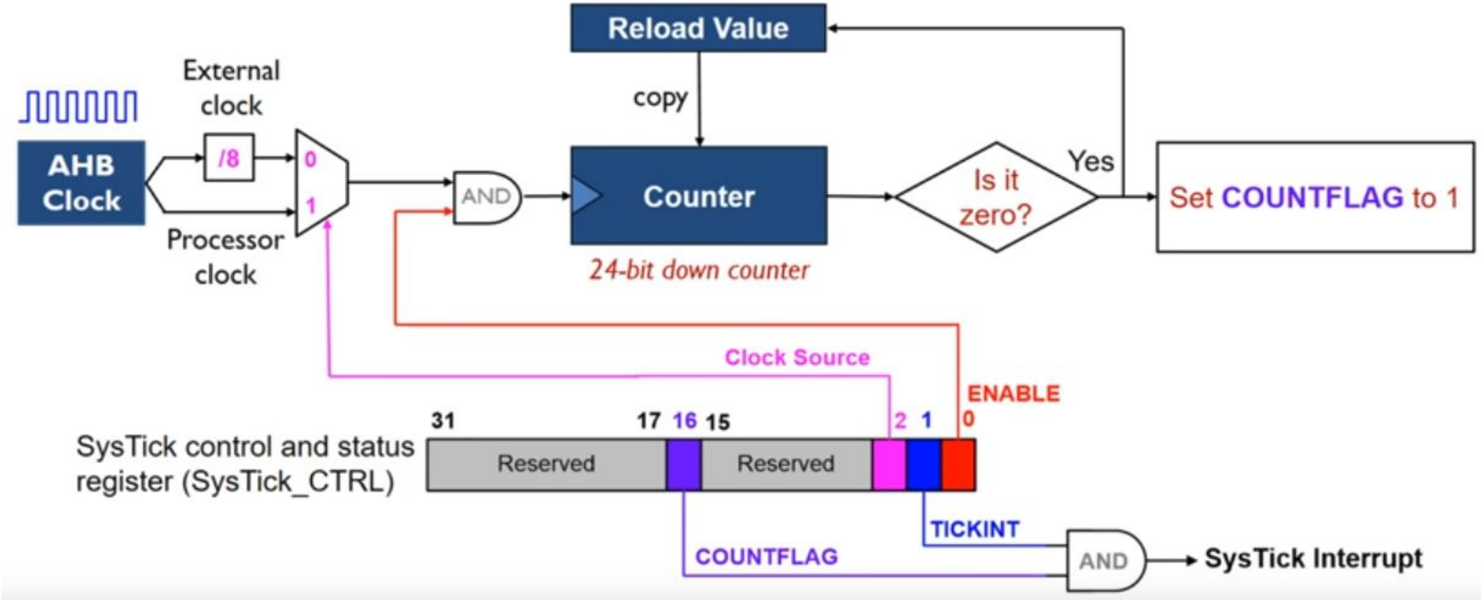
while ((HAL_GetTick() -
tickstart) < wait);

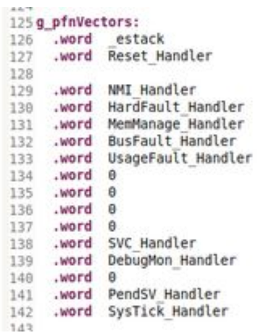


```
volatile int cnt = 0;

void SysTick_Handler(void)
{
  HAL_IncTick();
  cnt++;
}

int main(void)
{
  HAL_Init();
  while (1) {
    HAL_Delay(1000);
    // Some code
  }
}
```

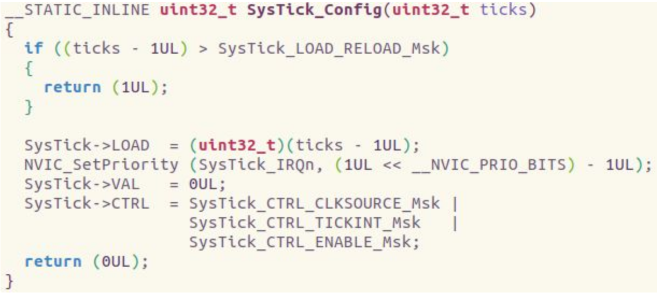
https://www.st.com/content/ccc/resource/technical/document/user_manual/45/27/9c/32/76/57/48/b9/DM00189702.pdf/files/DM00189702.pdf/jcr:content/translations/en.DM00189702.pdf

stm32f7xx_hal.c

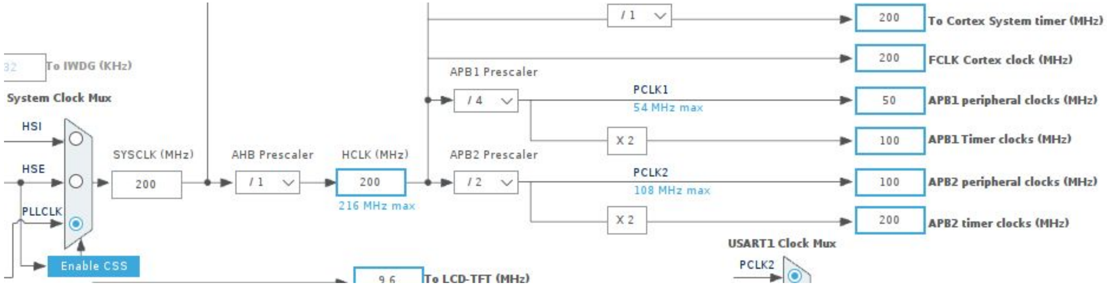




```
g_pfnVectors:
  .word  _estack
  .word  Reset_Handler

  .word  NMI_Handler
  .word  HardFault_Handler
  .word  MemManage_Handler
  .word  BusFault_Handler
  .word  UsageFault_Handler
  .word  0
  .word  0
  .word  0
  .word  0
  .word  SVC_Handler
  .word  DebugMon_Handler
  .word  0
  .word  PendSV_Handler
  .word  SysTick_Handler
```

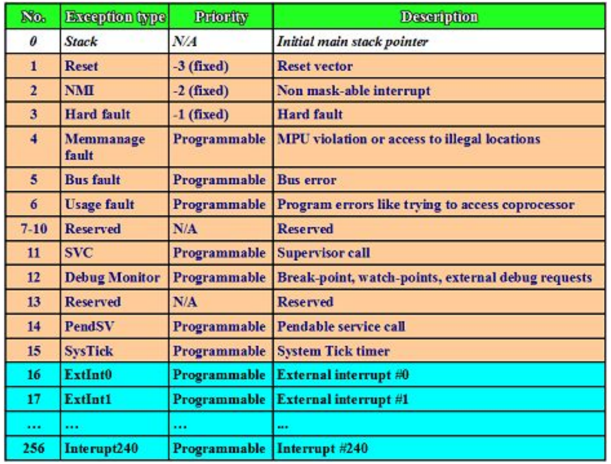


```
__STATIC_INLINE uint32_t SysTick_Config(uint32_t ticks)
{
  if ((ticks - 1UL) > SysTick_LOAD_RELOAD_Msk)
  {
    return (1UL);
  }

  SysTick->LOAD  = (uint32_t)(ticks - 1UL);
  NVIC_SetPriority (SysTick_IRQn, (1UL << __NVIC_PRIO_BITS) - 1UL);
  SysTick->VAL   = 0UL;
  SysTick->CTRL  = SysTick_CTRL_CLKSOURCE_Msk |
                   SysTick_CTRL_TICKINT_Msk   |
                   SysTick_CTRL_ENABLE_Msk;
  return (0UL);
}
```

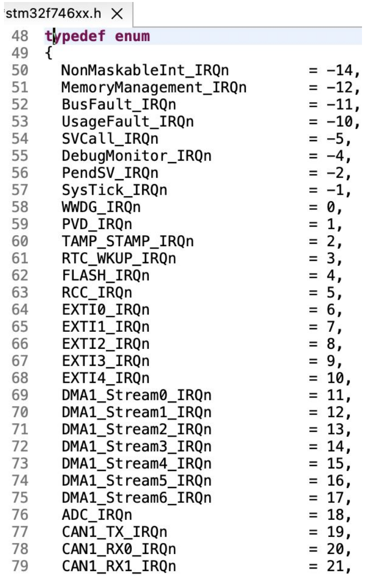


https://www.st.com/content/ccc/resource/technical/document/user_manual/45/27/9c/32/76/57/48/b9/DM00189702.pdf/files/DM00189702.pdf/jcr:content/translations/en.DM00189702.pdf

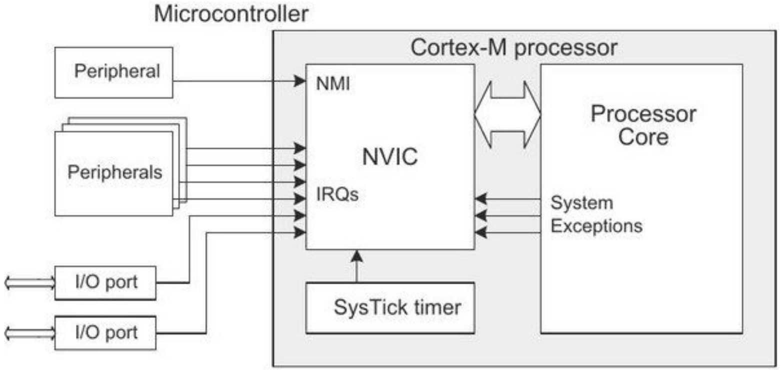
SCB->VTOR



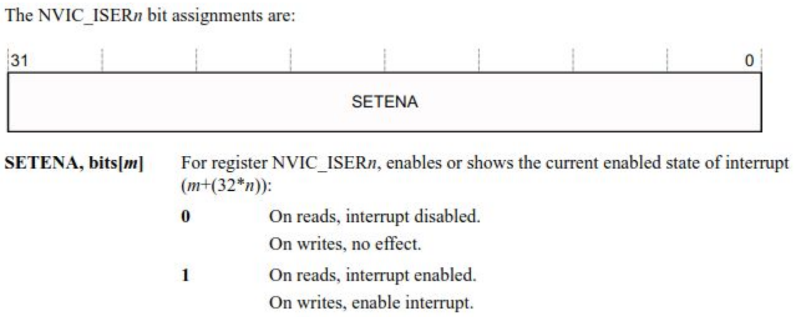
No.	Exception type	Priority	Description
0	*Stack*	*N/A*	*Initial main stack pointer*
1	Reset	-3 (fixed)	Reset vector
2	NMI	-2 (fixed)	Non mask-able interrupt
3	Hard fault	-1 (fixed)	Hard fault
4	Memmanage fault	Programmable	MPU violation or access to illegal locations
5	Bus fault	Programmable	Bus error
6	Usage fault	Programmable	Program errors like trying to access coprocessor
7-10	Reserved	N/A	Reserved
11	SVC	Programmable	Supervisor call
12	Debug Monitor	Programmable	Break-point, watch-points, external debug requests
13	Reserved	N/A	Reserved
14	PendSV	Programmable	Pendable service call
15	SysTick	Programmable	System Tick timer
16	ExtInt0	Programmable	External interrupt #0
17	ExtInt1	Programmable	External interrupt #1
...	...	...	...
256	Interupt240	Programmable	Interrupt #240



```
typedef enum
{
  NonMaskableInt_IRQn          = -14,
  MemoryManagement_IRQn        = -12,
  BusFault_IRQn                = -11,
  UsageFault_IRQn              = -10,
  SVCall_IRQn                  = -5,
  DebugMonitor_IRQn            = -4,
  PendSV_IRQn                  = -2,
  SysTick_IRQn                 = -1,
  WWDG_IRQn                    = 0,
  PVD_IRQn                     = 1,
  TAMP_STAMP_IRQn              = 2,
  RTC_WKUP_IRQn                = 3,
  FLASH_IRQn                   = 4,
  RCC_IRQn                     = 5,
  EXTI0_IRQn                   = 6,
  EXTI1_IRQn                   = 7,
  EXTI2_IRQn                   = 8,
  EXTI3_IRQn                   = 9,
  EXTI4_IRQn                   = 10,
  DMA1_Stream0_IRQn            = 11,
  DMA1_Stream1_IRQn            = 12,
  DMA1_Stream2_IRQn            = 13,
  DMA1_Stream3_IRQn            = 14,
  DMA1_Stream4_IRQn            = 15,
  DMA1_Stream5_IRQn            = 16,
  DMA1_Stream6_IRQn            = 17,
  ADC_IRQn                     = 18,
  CAN1_TX_IRQn                 = 19,
  CAN1_RX0_IRQn                = 20,
  CAN1_RX1_IRQn                = 21,
```



Interrupt Number	Interrupt Handler Address
1	Reset Handler

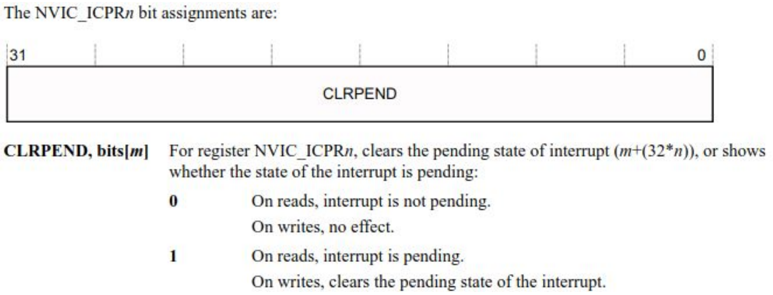


The NVIC_ISER*n* bit assignments are:

SETENA, bits[*m*] For register NVIC_ISER*n*, enables or shows the current enabled state of interrupt (*m*+(32**n*)):

0 On reads, interrupt disabled.
On writes, no effect.

1 On reads, interrupt enabled.
On writes, enable interrupt.



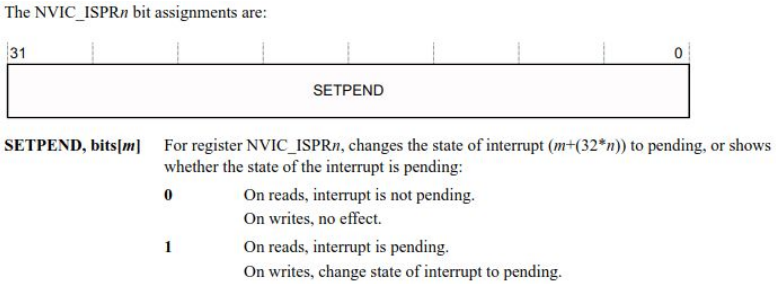
The NVIC_ICPR*n* bit assignments are:

31
0
CLRPEND

CLRPEND, bits[*m*] For register NVIC_ICPR*n*, clears the pending state of interrupt (*m*+(32**n*)), or shows whether the state of the interrupt is pending:

0 On reads, interrupt is not pending.
On writes, no effect.

1 On reads, interrupt is pending.
On writes, clears the pending state of the interrupt.



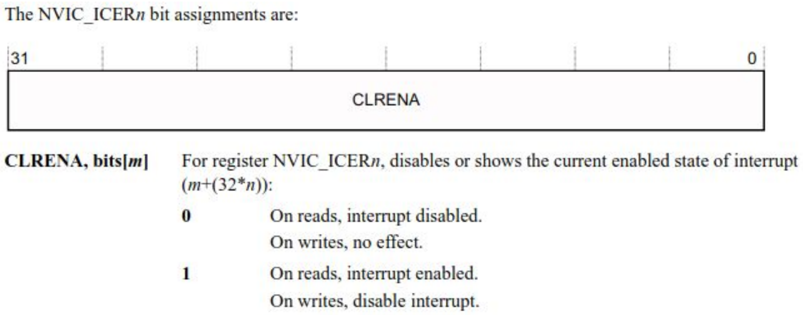
The NVIC_ISPR*n* bit assignments are:

31
0
SETPEND

SETPEND, bits[*m*] For register NVIC_ISPR*n*, changes the state of interrupt (*m*+(32**n*)) to pending, or shows whether the state of the interrupt is pending:

0 On reads, interrupt is not pending.
On writes, no effect.

1 On reads, interrupt is pending.
On writes, change state of interrupt to pending.

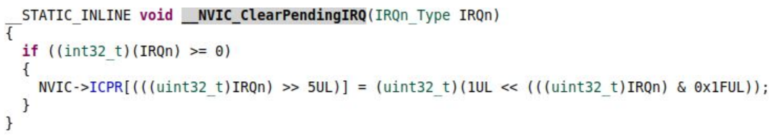


The NVIC_ICER*n* bit assignments are:

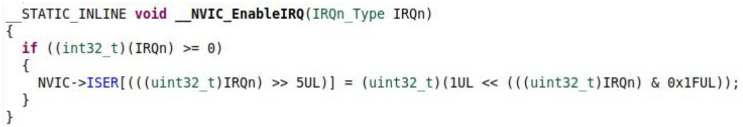
CLRENA, bits[*m*] For register NVIC_ICER*n*, disables or shows the current enabled state of interrupt (*m*+(32**n*)):

0 On reads, interrupt disabled.
On writes, no effect.

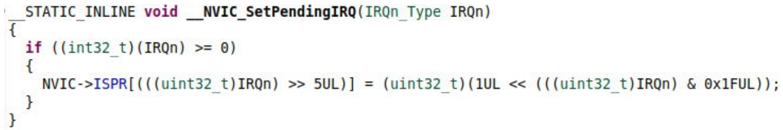
1 On reads, interrupt enabled.
On writes, disable interrupt.



```
__STATIC_INLINE void __NVIC_ClearPendingIRQ(IRQn_Type IRQn)
{
  if ((int32_t)(IRQn) >= 0)
  {
    NVIC->ICPR[(((uint32_t)IRQn) >> 5UL)] = (uint32_t)(1UL << (((uint32_t)IRQn) & 0x1FUL));
  }
}
```



```
__STATIC_INLINE void __NVIC_EnableIRQ(IRQn_Type IRQn)
{
  if ((int32_t)(IRQn) >= 0)
  {
    NVIC->ISER[(((uint32_t)IRQn) >> 5UL)] = (uint32_t)(1UL << (((uint32_t)IRQn) & 0x1FUL));
  }
}
```

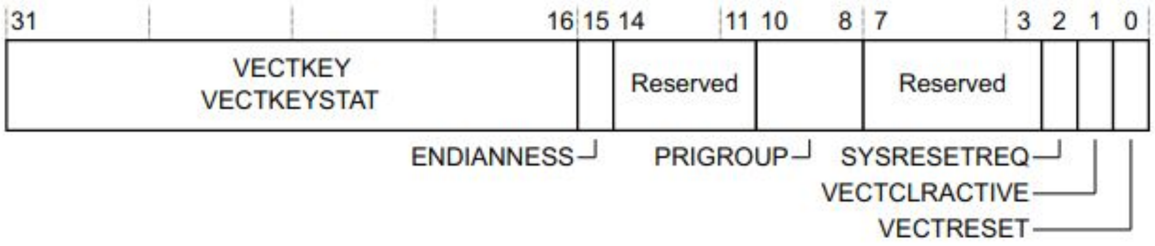


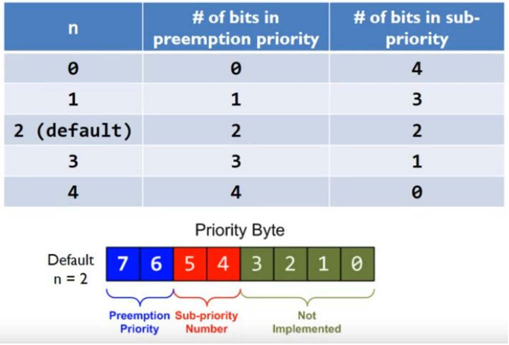
```
__STATIC_INLINE void __NVIC_SetPendingIRQ(IRQn_Type IRQn)
{
  if ((int32_t)(IRQn) >= 0)
  {
    NVIC->ISPR[(((uint32_t)IRQn) >> 5UL)] = (uint32_t)(1UL << (((uint32_t)IRQn) & 0x1FUL));
  }
}
```

https://documentation-service.arm.com/static/606dc36485368c4c2b1bf62f?token=

Cortex M7, Interrupt Registers

The AIRCR bit assignments are:

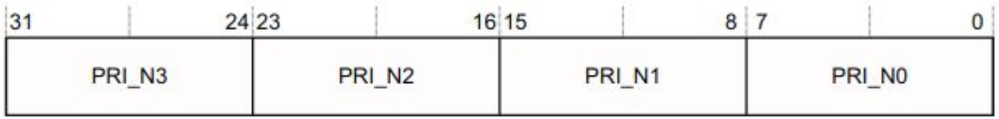




n	# of bits in preemption priority	# of bits in sub-priority
0	0	4
1	1	3
2 (default)	2	2
3	3	1
4	4	0

Priority Byte
Default
n = 2
7
6
5
4
3
2
1
0
Preemption Priority
Sub-priority Number
Not Implemented

The NVIC_IPR*n* bit assignments are:

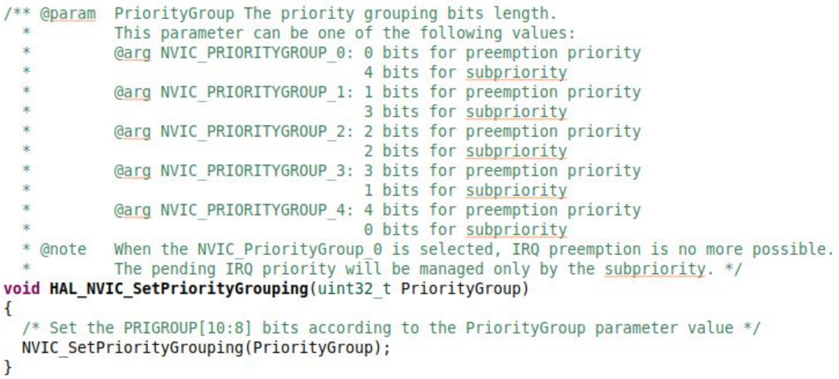


PRI_N3, bits[31:24] For register NVIC_IPR*n*, priority of interrupt number 4*n*+3.

PRI_N2, bits[23:16] For register NVIC_IPR*n*, priority of interrupt number 4*n*+2.

PRI_N1, bits[15:8] For register NVIC_IPR*n*, priority of interrupt number 4*n*+1.

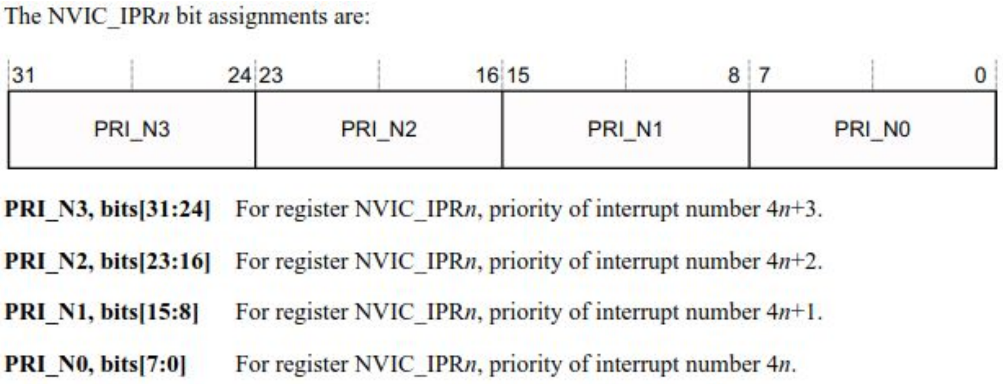
PRI_N0, bits[7:0] For register NVIC_IPR*n*, priority of interrupt number 4*n*.



```
/** @param  PriorityGroup The priority grouping bits length.
  *         This parameter can be one of the following values:
  *         @arg NVIC_PRIORITYGROUP_0: 0 bits for preemption priority
  *                                    4 bits for subpriority
  *         @arg NVIC_PRIORITYGROUP_1: 1 bits for preemption priority
  *                                    3 bits for subpriority
  *         @arg NVIC_PRIORITYGROUP_2: 2 bits for preemption priority
  *                                    2 bits for subpriority
  *         @arg NVIC_PRIORITYGROUP_3: 3 bits for preemption priority
  *                                    1 bits for subpriority
  *         @arg NVIC_PRIORITYGROUP_4: 4 bits for preemption priority
  *                                    0 bits for subpriority
  * @note   When the NVIC_PriorityGroup_0 is selected, IRQ preemption is no more possible.
  *         The pending IRQ priority will be managed only by the subpriority. */
void HAL_NVIC_SetPriorityGrouping(uint32_t PriorityGroup)
{
  /* Set the PRIGROUP[10:8] bits according to the PriorityGroup parameter value */
  NVIC_SetPriorityGrouping(PriorityGroup);
}
```

Cortex M7, Interrupt Registers

https://documentation-service.arm.com/static/606dc36485368c4c2b1bf62f?token=



The NVIC_IPR*n* bit assignments are:

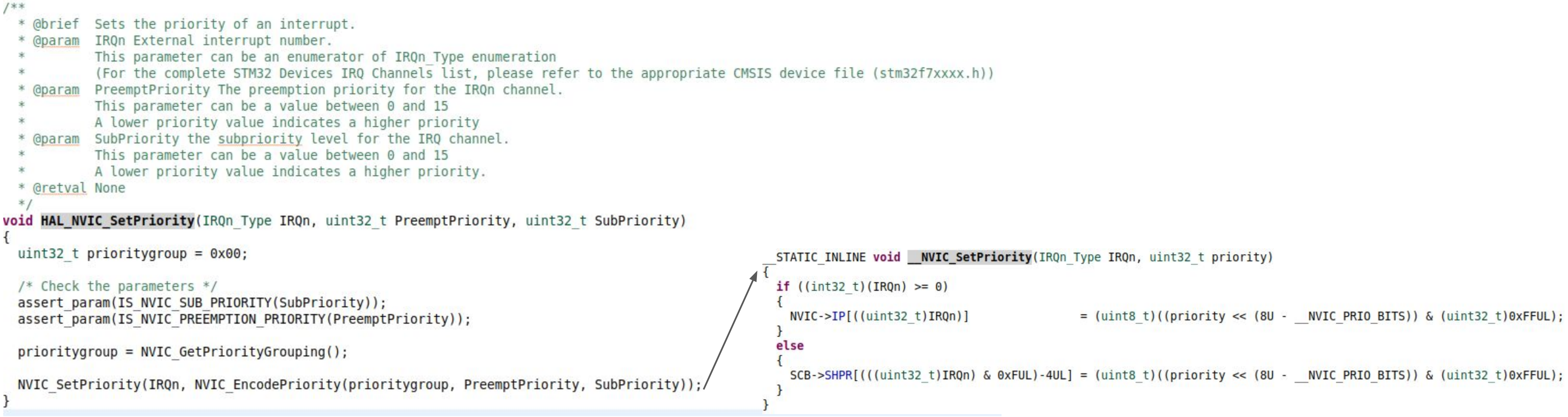
31 24	23 16	15 8	7 0
PRI_N3	PRI_N2	PRI_N1	PRI_N0

PRI_N3, bits[31:24] For register NVIC_IPR*n*, priority of interrupt number 4*n*+3.

PRI_N2, bits[23:16] For register NVIC_IPR*n*, priority of interrupt number 4*n*+2.

PRI_N1, bits[15:8] For register NVIC_IPR*n*, priority of interrupt number 4*n*+1.

PRI_N0, bits[7:0] For register NVIC_IPR*n*, priority of interrupt number 4*n*.



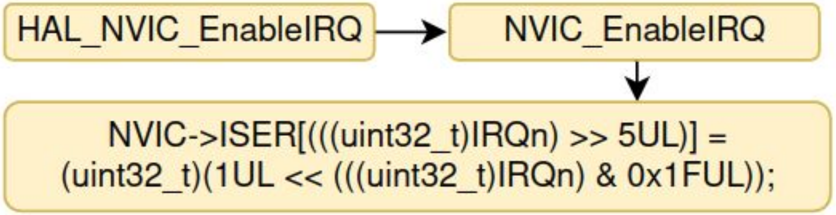
```c
/**
  * @brief  Sets the priority of an interrupt.
  * @param  IRQn External interrupt number.
  *         This parameter can be an enumerator of IRQn_Type enumeration
  *         (For the complete STM32 Devices IRQ Channels list, please refer to the appropriate CMSIS device file (stm32f7xxxx.h))
  * @param  PreemptPriority The preemption priority for the IRQn channel.
  *         This parameter can be a value between 0 and 15
  *         A lower priority value indicates a higher priority
  * @param  SubPriority the subpriority level for the IRQ channel.
  *         This parameter can be a value between 0 and 15
  *         A lower priority value indicates a higher priority.
  * @retval None
  */
void HAL_NVIC_SetPriority(IRQn_Type IRQn, uint32_t PreemptPriority, uint32_t SubPriority)
{
  uint32_t prioritygroup = 0x00;

  /* Check the parameters */
  assert_param(IS_NVIC_SUB_PRIORITY(SubPriority));
  assert_param(IS_NVIC_PREEMPTION_PRIORITY(PreemptPriority));

  prioritygroup = NVIC_GetPriorityGrouping();

  NVIC_SetPriority(IRQn, NVIC_EncodePriority(prioritygroup, PreemptPriority, SubPriority));
}
```

```c
__STATIC_INLINE void __NVIC_SetPriority(IRQn_Type IRQn, uint32_t priority)
{
  if ((int32_t)(IRQn) >= 0)
  {
    NVIC->IP[((uint32_t)IRQn)]               = (uint8_t)((priority << (8U - __NVIC_PRIO_BITS)) & (uint32_t)0xFFUL);
  }
  else
  {
    SCB->SHPR[(((uint32_t)IRQn) & 0xFUL)-4UL] = (uint8_t)((priority << (8U - __NVIC_PRIO_BITS)) & (uint32_t)0xFFUL);
  }
}
```

https://documentation-service.arm.com/static/606dc36485368c4c2b1bf62f?token=

Cortex M7, Interrupt Registers

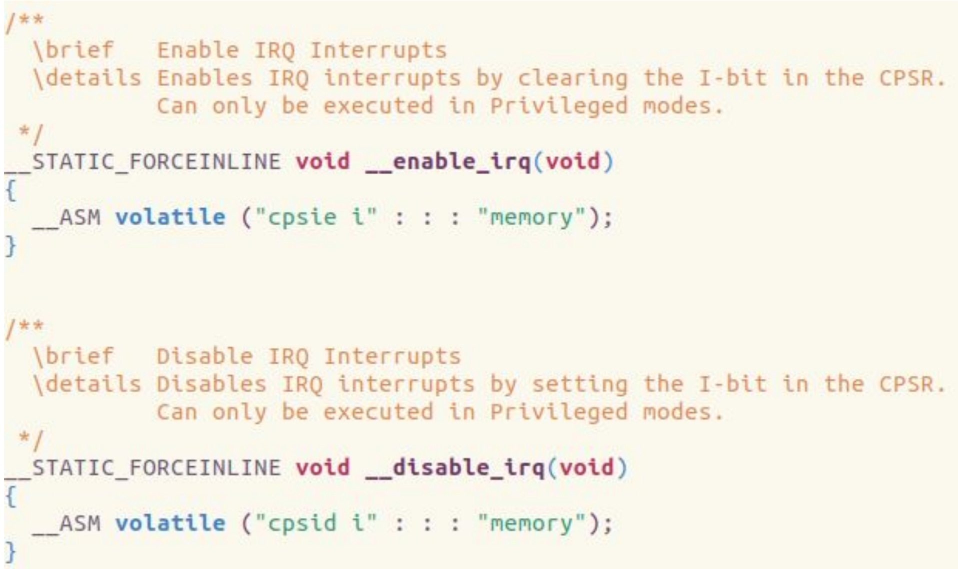
HAL_NVIC_EnableIRQ HAL_NVIC_DisableIRQ	Enable/Disable a device specific interrupt in the NVIC interrupt controller.
HAL_NVIC_SetPriority	Set the preemption priority and subpriority for an interrupt.
HAL_NVIC_SetPriorityGrouping	Set the priority grouping field 0: 0 bits for preemption priority 4 bits for subpriority 1: 1 bits for preemption priority 3 bits for subpriority 2: 2 bits for preemption priority 2 bits for subpriority 3: 3 bits for preemption priority 1 bits for subpriority 4: 4 bits for preemption priority 0 bits for subpriority When the NVIC_PriorityGroup_0 is selected, IRQ preemption is no more possible. The pending IRQ priority will be managed only by the subpriority.
HAL_NVIC_SetPendingIRQ **HAL_NVIC_ClearPendingIRQ**	Set/Clear Pending bit of an external interrupt



HAL_NVIC_EnableIRQ → NVIC_EnableIRQ

```
NVIC->ISER[(((uint32_t)IRQn) >> 5UL)] =
(uint32_t)(1UL << (((uint32_t)IRQn) & 0x1FUL));
```

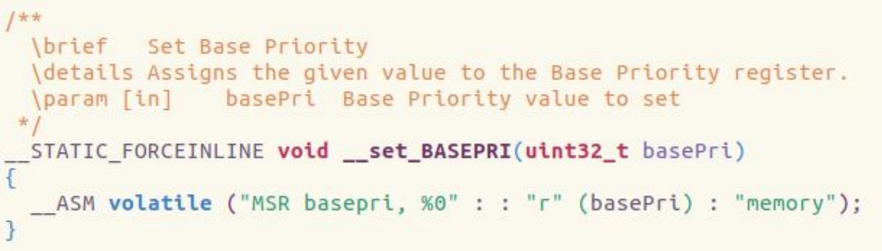
https://www.st.com/content/ccc/resource/technical/document/user_manual/45/27/9c/32/76/57/48/b9/DM00189702.pdf/files/DM00189702.pdf/jcr:content/translations/en.DM00189702.pdf



```
/**
  \brief   Enable IRQ Interrupts
  \details Enables IRQ interrupts by clearing the I-bit in the CPSR.
           Can only be executed in Privileged modes.
 */
__STATIC_FORCEINLINE void __enable_irq(void)
{
  __ASM volatile ("cpsie i" : : : "memory");
}


/**
  \brief   Disable IRQ Interrupts
  \details Disables IRQ interrupts by setting the I-bit in the CPSR.
           Can only be executed in Privileged modes.
 */
__STATIC_FORCEINLINE void __disable_irq(void)
{
  __ASM volatile ("cpsid i" : : : "memory");
}
```

CPS changes the **PRIMASK** special register values.

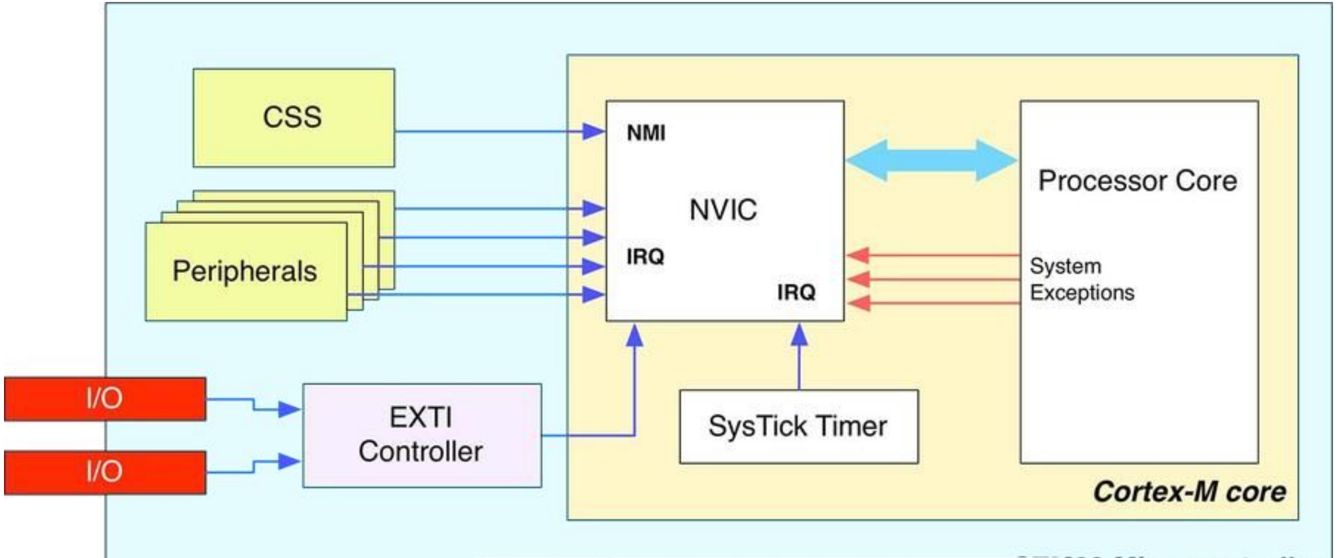
- CPSID causes interrupts to be disabled by setting PRIMASK.
- CPSIE cause interrupts to be enabled by clearing PRIMASK.

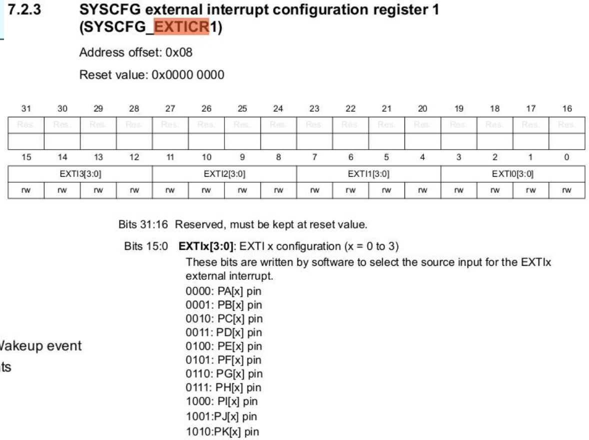


```
/**
  \brief   Set Base Priority
  \details Assigns the given value to the Base Priority register.
  \param [in]    basePri  Base Priority value to set
 */
__STATIC_FORCEINLINE void __set_BASEPRI(uint32_t basePri)
{
  __ASM volatile ("MSR basepri, %0" : : "r" (basePri) : "memory");
}
```

The **BASEPRI** register defines the minimum priority for exception processing. When BASEPRI is set to a nonzero value, it prevents the activation of all exceptions with the same or lower priority level as the BASEPRI value.

https://developer.arm.com/documentation/dui0497/a/BABHBAAB





7.2.3 **SYSCFG external interrupt configuration register 1 (SYSCFG_EXTICR1)**

Address offset: 0x08

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
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.

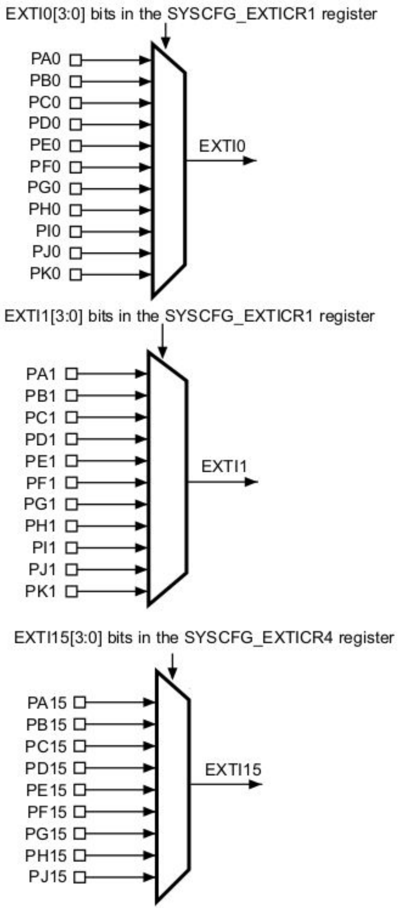
Bits 15:0 **EXTIx[3:0]**: EXTI x configuration (x = 0 to 3)

These bits are written by software to select the source input for the EXTIx external interrupt.

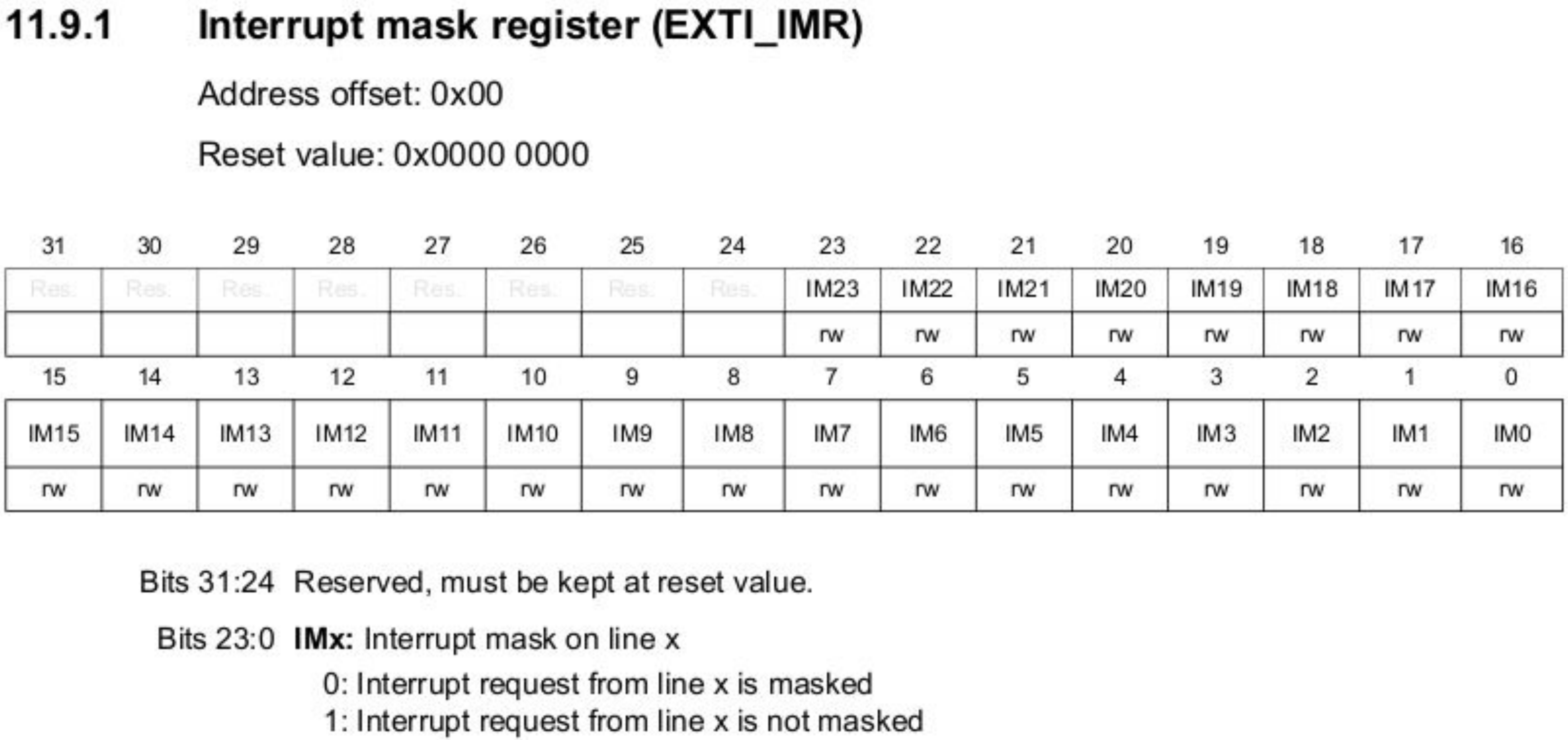
0000: PA[x] pin
0001: PB[x] pin
0010: PC[x] pin
0011: PD[x] pin
0100: PE[x] pin
0101: PF[x] pin
0110: PG[x] pin
0111: PH[x] pin
1000: PI[x] pin
1001:PJ[x] pin
1010:PK[x] pin

The eight other EXTI lines are connected as follows:

- EXTI line 16 is connected to the PVD output
- EXTI line 17 is connected to the RTC Alarm event
- EXTI line 18 is connected to the USB OTG FS Wakeup event
- EXTI line 19 is connected to the Ethernet Wakeup event
- EXTI line 20 is connected to the USB OTG HS (configured in FS) Wakeup event
- EXTI line 21 is connected to the RTC Tamper and TimeStamp events
- EXTI line 22 is connected to the RTC Wakeup event
- EXTI line 23 is connected to the LPTIM1 asynchronous event


EXTI0[3:0] bits in the SYSCFG_EXTICR1 register
PA0
PB0
PC0
PD0
PE0
PF0
PG0
PH0
PI0
PJ0
PK0
EXTI0
EXTI1[3:0] bits in the SYSCFG_EXTICR1 register
PA1
PB1
PC1
PD1
PE1
PF1
PG1
PH1
PI1
PJ1
PK1
EXTI1
EXTI15[3:0] bits in the SYSCFG_EXTICR4 register
PA15
PB15
PC15
PD15
PE15
PF15
PG15
PH15
PJ15
EXTI15

https://documentation-service.arm.com/static/606dc36485368c4c2b1bf62f?token=



11.9.1 Interrupt mask register (EXTI_IMR)

Address offset: 0x00

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.	IM23	IM22	IM21	IM20	IM19	IM18	IM17	IM16
								rw	rw	rw	rw	rw	rw	rw	rw

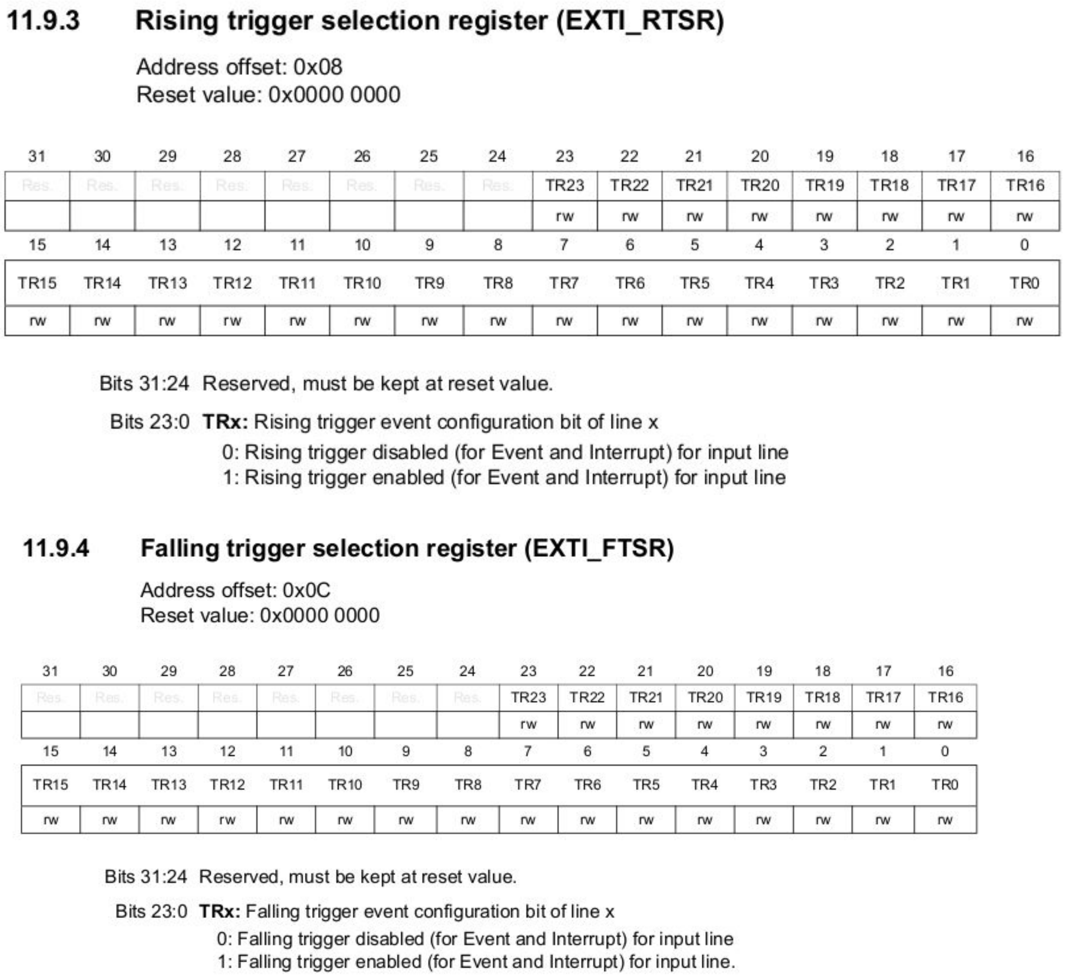
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
IM15	IM14	IM13	IM12	IM11	IM10	IM9	IM8	IM7	IM6	IM5	IM4	IM3	IM2	IM1	IM0
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bits 31:24 Reserved, must be kept at reset value.

Bits 23:0 **IMx:** Interrupt mask on line x

0: Interrupt request from line x is masked
1: Interrupt request from line x is not masked

https://documentation-service.arm.com/static/606dc36485368c4c2b1bf62f?token=



11.9.3 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
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	TR23	TR22	TR21	TR20	TR19	TR18	TR17	TR16
								rw	rw	rw	rw	rw	rw	rw	rw

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
TR15	TR14	TR13	TR12	TR11	TR10	TR9	TR8	TR7	TR6	TR5	TR4	TR3	TR2	TR1	TR0
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bits 31:24 Reserved, must be kept at reset value.

Bits 23:0 **TRx:** Rising trigger event configuration bit of line x

0: Rising trigger disabled (for Event and Interrupt) for input line
1: Rising trigger enabled (for Event and Interrupt) for input line

11.9.4 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
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	TR23	TR22	TR21	TR20	TR19	TR18	TR17	TR16
								rw	rw	rw	rw	rw	rw	rw	rw

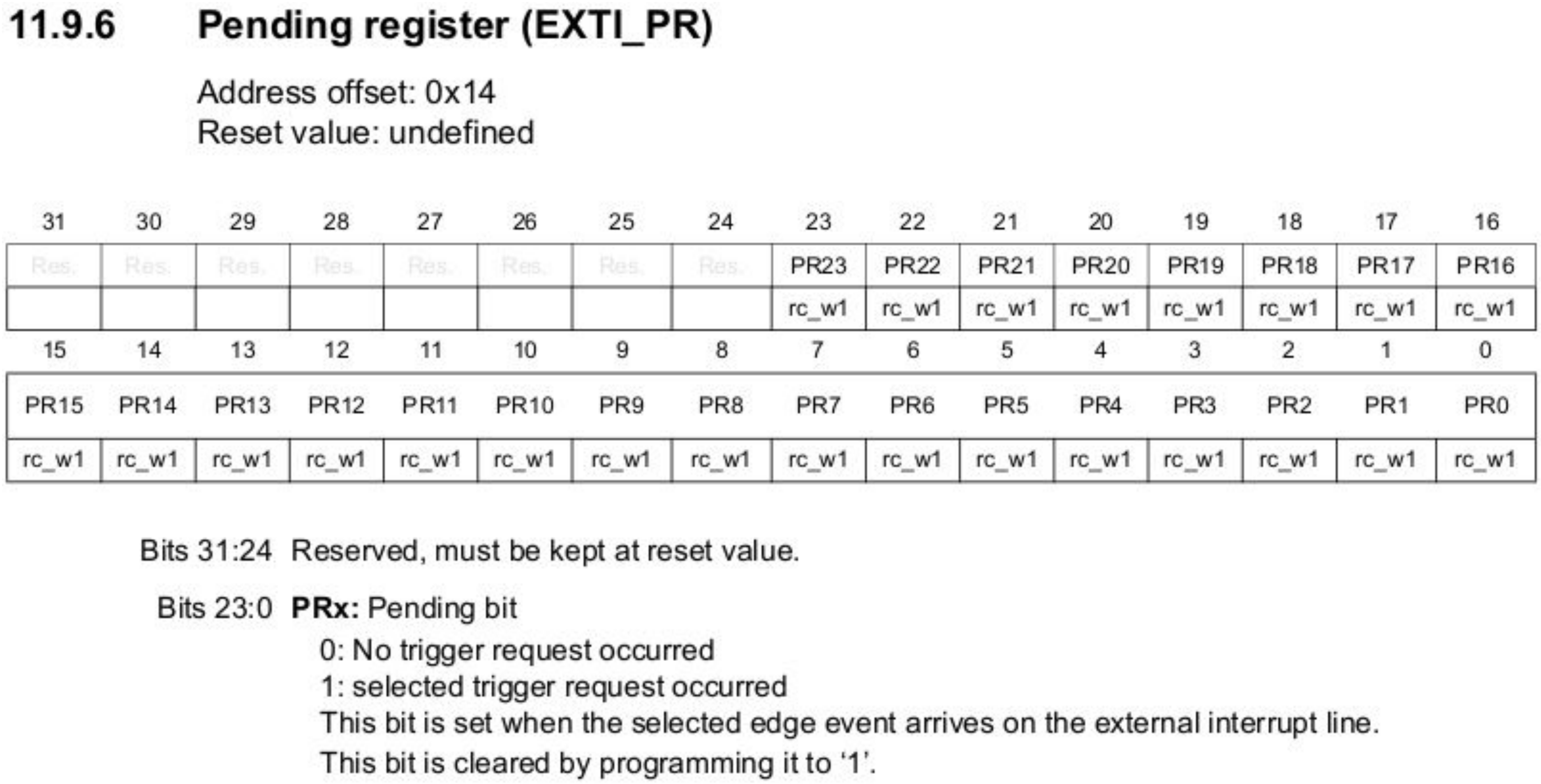
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
TR15	TR14	TR13	TR12	TR11	TR10	TR9	TR8	TR7	TR6	TR5	TR4	TR3	TR2	TR1	TR0
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bits 31:24 Reserved, must be kept at reset value.

Bits 23:0 **TRx:** Falling trigger event configuration bit of line x

0: Falling trigger disabled (for Event and Interrupt) for input line
1: Falling trigger enabled (for Event and Interrupt) for input line.

https://documentation-service.arm.com/static/606dc36485368c4c2b1bf62f?token=



11.9.6 Pending register (EXTI_PR)

Address offset: 0x14
Reset value: undefined

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	PR23	PR22	PR21	PR20	PR19	PR18	PR17	PR16
								rc_w1	rc_w1	rc_w1	rc_w1	rc_w1	rc_w1	rc_w1	rc_w1
15	**14**	**13**	**12**	**11**	**10**	**9**	**8**	**7**	**6**	**5**	**4**	**3**	**2**	**1**	**0**
PR15	PR14	PR13	PR12	PR11	PR10	PR9	PR8	PR7	PR6	PR5	PR4	PR3	PR2	PR1	PR0
rc_w1	rc_w1	rc_w1	rc_w1	rc_w1	rc_w1	rc_w1	rc_w1	rc_w1	rc_w1	rc_w1	rc_w1	rc_w1	rc_w1	rc_w1	rc_w1

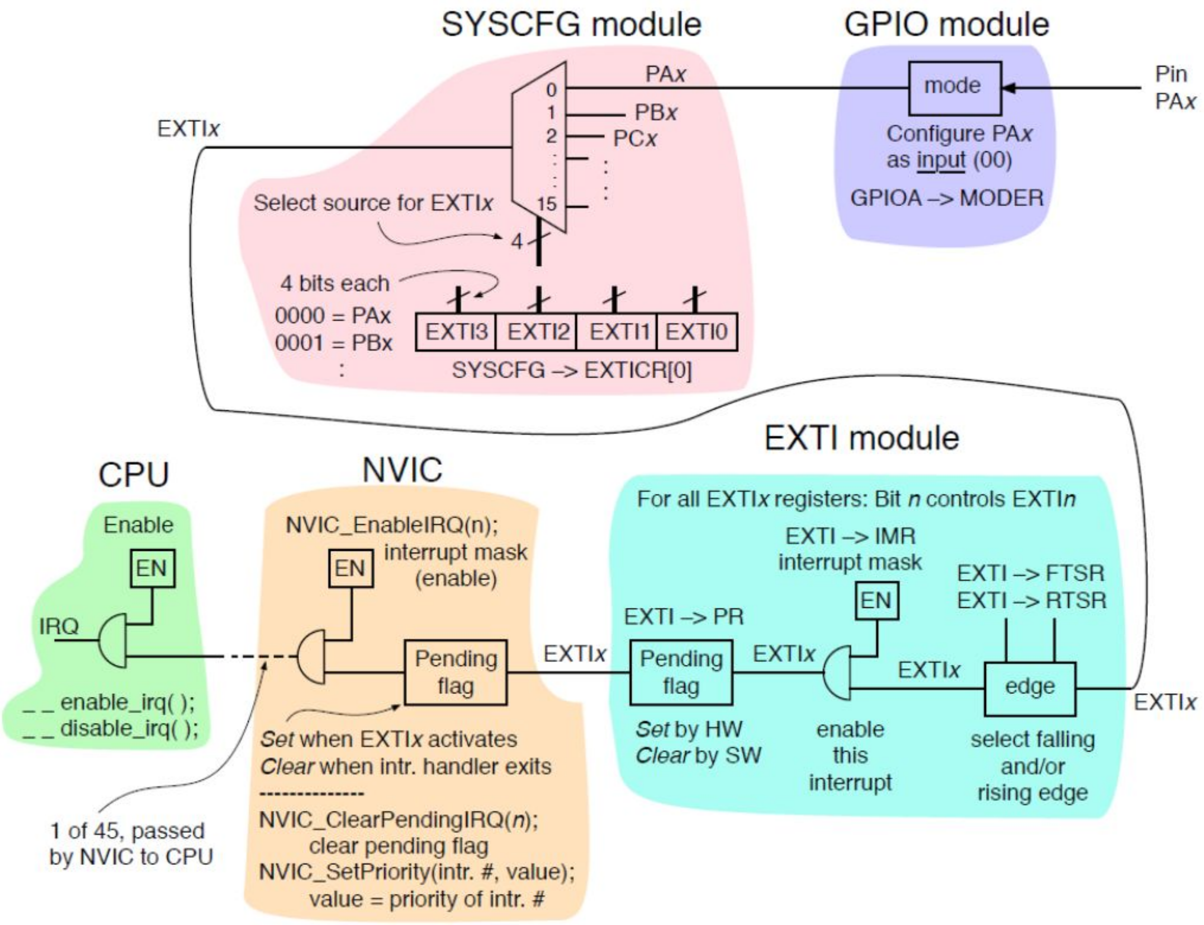
Bits 31:24 Reserved, must be kept at reset value.

Bits 23:0 **PRx:** Pending bit

0: No trigger request occurred
1: selected trigger request occurred
This bit is set when the selected edge event arrives on the external interrupt line.
This bit is cleared by programming it to ‘1’.

NOTE: Необходимо сбрасывать, иначе будут бесконечные прерывания

https://documentation-service.arm.com/static/606dc36485368c4c2b1bf62f?token=



SYSCFG module

GPIO module

PAx

mode

Pin PAx

0

1

PBx

2

PCx

15

EXTIx

Configure PAx as input (00)

GPIOA –> MODER

Select source for EXTIx

4

4 bits each

0000 = PAx

0001 = PBx

:

EXTI3 | EXTI2 | EXTI1 | EXTI0

SYSCFG –> EXTICR[0]

EXTI module

CPU

NVIC

For all EXTIx registers: Bit *n* controls EXTI*n*

Enable

EN

IRQ

__ enable_irq( );

__ disable_irq( );

NVIC_EnableIRQ(n);

EN

interrupt mask (enable)

Pending flag

Set when EXTIx activates

Clear when intr. handler exits

NVIC_ClearPendingIRQ(*n*);
clear pending flag

NVIC_SetPriority(intr. #, value);
value = priority of intr. #

1 of 45, passed by NVIC to CPU

EXTIx

EXTI –> PR

Pending flag

Set by HW

Clear by SW

EXTIx

EXTI –> IMR
interrupt mask

EN

enable this interrupt

EXTIx

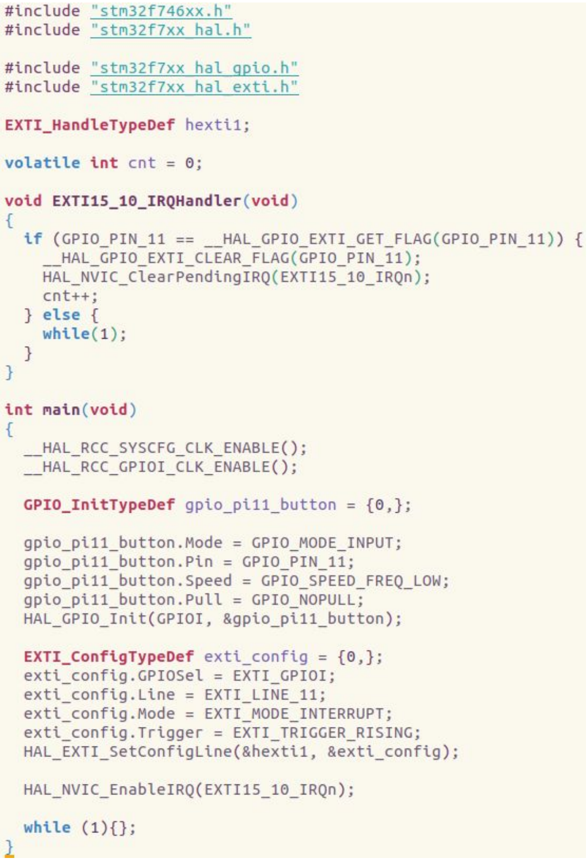
EXTI –> FTSR

EXTI –> RTSR

edge

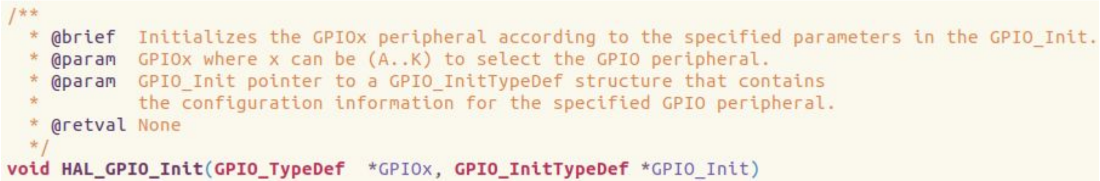
select falling and/or rising edge

EXTIx

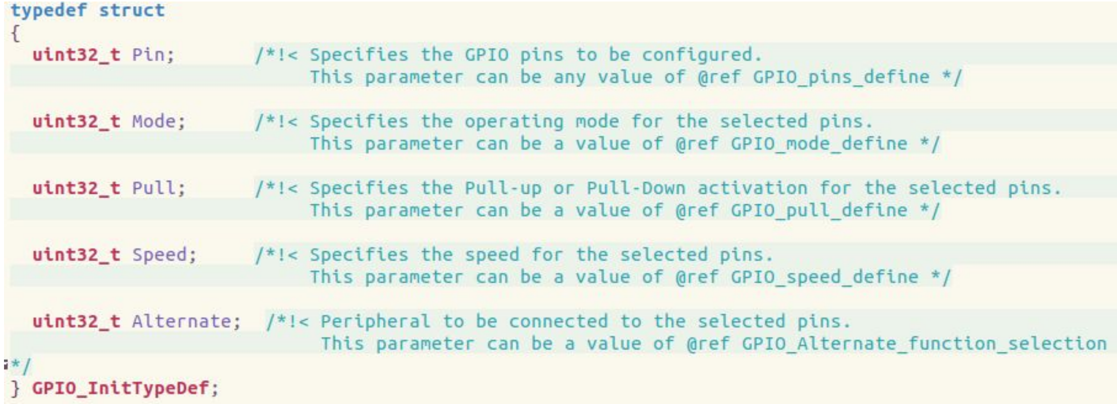


```
#include "stm32f746xx.h"
#include "stm32f7xx_hal.h"

#include "stm32f7xx_hal_gpio.h"
#include "stm32f7xx_hal_exti.h"

EXTI_HandleTypeDef hexti1;

volatile int cnt = 0;

void EXTI15_10_IRQHandler(void)
{
  if (GPIO_PIN_11 == __HAL_GPIO_EXTI_GET_FLAG(GPIO_PIN_11)) {
    __HAL_GPIO_EXTI_CLEAR_FLAG(GPIO_PIN_11);
    HAL_NVIC_ClearPendingIRQ(EXTI15_10_IRQn);
    cnt++;
  } else {
    while(1);
  }
}

int main(void)
{
  __HAL_RCC_SYSCFG_CLK_ENABLE();
  __HAL_RCC_GPIOI_CLK_ENABLE();

  GPIO_InitTypeDef gpio_pi11_button = {0,};

  gpio_pi11_button.Mode = GPIO_MODE_INPUT;
  gpio_pi11_button.Pin = GPIO_PIN_11;
  gpio_pi11_button.Speed = GPIO_SPEED_FREQ_LOW;
  gpio_pi11_button.Pull = GPIO_NOPULL;
  HAL_GPIO_Init(GPIOI, &gpio_pi11_button);

  EXTI_ConfigTypeDef exti_config = {0,};
  exti_config.GPIOSel = EXTI_GPIOI;
  exti_config.Line = EXTI_LINE_11;
  exti_config.Mode = EXTI_MODE_INTERRUPT;
  exti_config.Trigger = EXTI_TRIGGER_RISING;
  HAL_EXTI_SetConfigLine(&hexti1, &exti_config);

  HAL_NVIC_EnableIRQ(EXTI15_10_IRQn);

  while (1){};
}
```

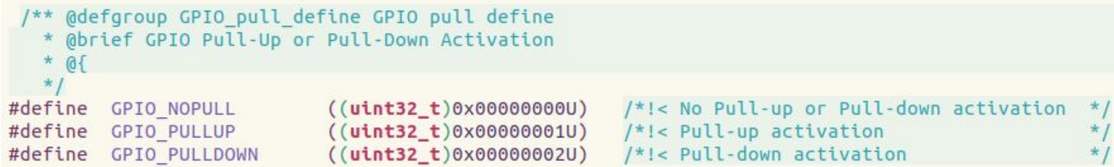
https://www.st.com/content/ccc/resource/technical/document/user_manual/45/27/9c/32/76/57/48/b9/DM00189702.pdf/files/DM00189702.pdf/jcr:content/translations/en.DM00189702.pdf



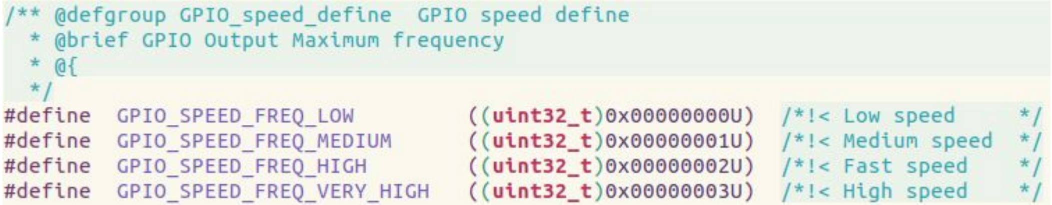
```
/**
  * @brief  Initializes the GPIOx peripheral according to the specified parameters in the GPIO_Init.
  * @param  GPIOx where x can be (A..K) to select the GPIO peripheral.
  * @param  GPIO_Init pointer to a GPIO_InitTypeDef structure that contains
  *         the configuration information for the specified GPIO peripheral.
  * @retval None
  */
void HAL_GPIO_Init(GPIO_TypeDef  *GPIOx, GPIO_InitTypeDef *GPIO_Init)
```



```
typedef struct
{
  uint32_t Pin;       /*!< Specifies the GPIO pins to be configured.
                           This parameter can be any value of @ref GPIO_pins_define */

  uint32_t Mode;      /*!< Specifies the operating mode for the selected pins.
                           This parameter can be a value of @ref GPIO_mode_define */

  uint32_t Pull;      /*!< Specifies the Pull-up or Pull-Down activation for the selected pins.
                           This parameter can be a value of @ref GPIO_pull_define */

  uint32_t Speed;     /*!< Specifies the speed for the selected pins.
                           This parameter can be a value of @ref GPIO_speed_define */

  uint32_t Alternate;  /*!< Peripheral to be connected to the selected pins.
                            This parameter can be a value of @ref GPIO_Alternate_function_selection
*/
} GPIO_InitTypeDef;
```

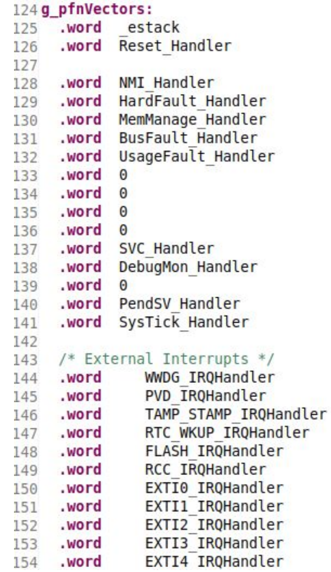


```
 /** @defgroup GPIO_pull_define GPIO pull define
   * @brief GPIO Pull-Up or Pull-Down Activation
   * @{
   */
#define  GPIO_NOPULL        ((uint32_t)0x00000000U)   /*!< No Pull-up or Pull-down activation  */
#define  GPIO_PULLUP        ((uint32_t)0x00000001U)   /*!< Pull-up activation                  */
#define  GPIO_PULLDOWN      ((uint32_t)0x00000002U)   /*!< Pull-down activation                */
```

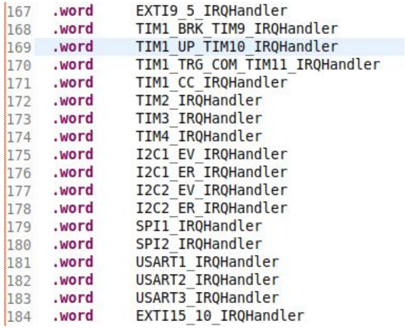


```
/** @defgroup GPIO_speed_define  GPIO speed define
  * @brief GPIO Output Maximum frequency
  * @{
  */
#define  GPIO_SPEED_FREQ_LOW         ((uint32_t)0x00000000U)  /*!< Low speed     */
#define  GPIO_SPEED_FREQ_MEDIUM      ((uint32_t)0x00000001U)  /*!< Medium speed  */
#define  GPIO_SPEED_FREQ_HIGH        ((uint32_t)0x00000002U)  /*!< Fast speed    */
#define  GPIO_SPEED_FREQ_VERY_HIGH   ((uint32_t)0x00000003U)  /*!< High speed    */
```

startup_stm32f746xx.s:

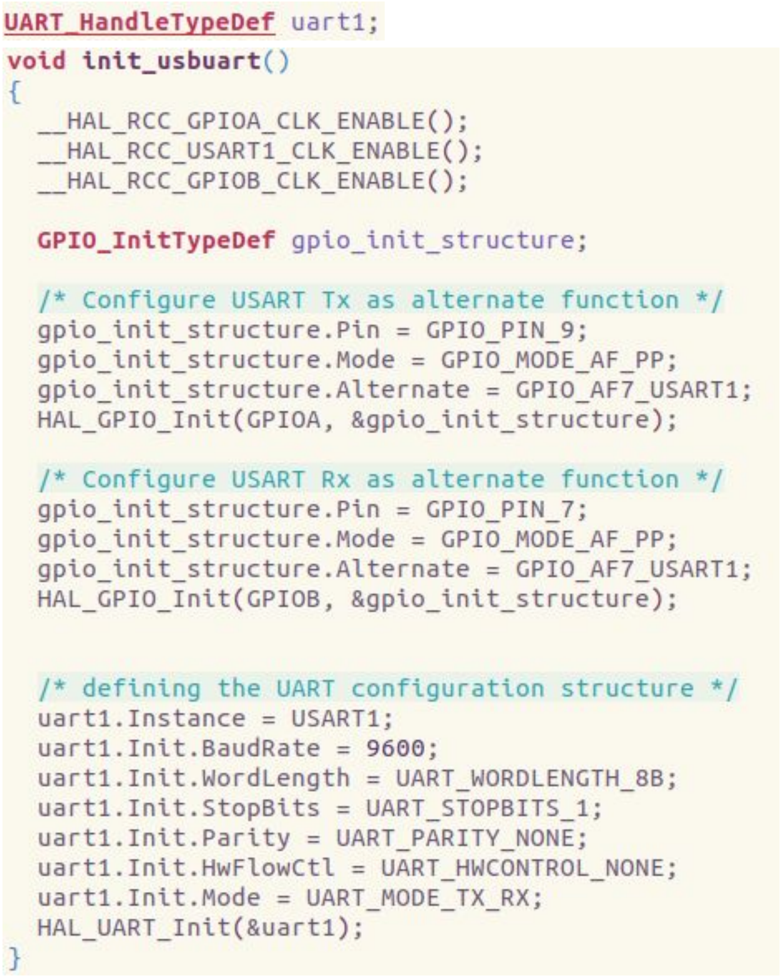


```
g_pfnVectors:
  .word  _estack
  .word  Reset_Handler

  .word  NMI_Handler
  .word  HardFault_Handler
  .word  MemManage_Handler
  .word  BusFault_Handler
  .word  UsageFault_Handler
  .word  0
  .word  0
  .word  0
  .word  0
  .word  SVC_Handler
  .word  DebugMon_Handler
  .word  0
  .word  PendSV_Handler
  .word  SysTick_Handler

  /* External Interrupts */
  .word     WWDG_IRQHandler
  .word     PVD_IRQHandler
  .word     TAMP_STAMP_IRQHandler
  .word     RTC_WKUP_IRQHandler
  .word     FLASH_IRQHandler
  .word     RCC_IRQHandler
  .word     EXTI0_IRQHandler
  .word     EXTI1_IRQHandler
  .word     EXTI2_IRQHandler
  .word     EXTI3_IRQHandler
  .word     EXTI4_IRQHandler
```

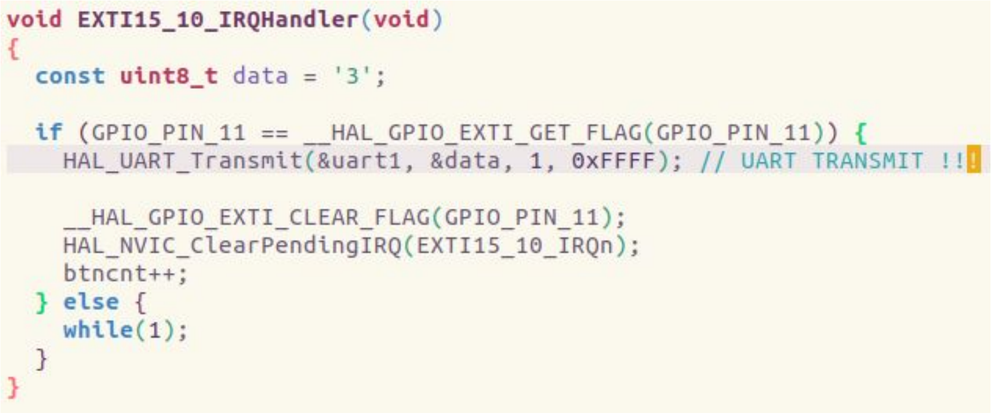


```
  .word     EXTI9_5_IRQHandler
  .word     TIM1_BRK_TIM9_IRQHandler
  .word     TIM1_UP_TIM10_IRQHandler
  .word     TIM1_TRG_COM_TIM11_IRQHandler
  .word     TIM1_CC_IRQHandler
  .word     TIM2_IRQHandler
  .word     TIM3_IRQHandler
  .word     TIM4_IRQHandler
  .word     I2C1_EV_IRQHandler
  .word     I2C1_ER_IRQHandler
  .word     I2C2_EV_IRQHandler
  .word     I2C2_ER_IRQHandler
  .word     SPI1_IRQHandler
  .word     SPI2_IRQHandler
  .word     USART1_IRQHandler
  .word     USART2_IRQHandler
  .word     USART3_IRQHandler
  .word     EXTI15_10_IRQHandler
```

stm32f7xx_hal_gpio.c



```
UART_HandleTypeDef uart1;
void init_usbuart()
{
  __HAL_RCC_GPIOA_CLK_ENABLE();
  __HAL_RCC_USART1_CLK_ENABLE();
  __HAL_RCC_GPIOB_CLK_ENABLE();

  GPIO_InitTypeDef gpio_init_structure;

  /* Configure USART Tx as alternate function */
  gpio_init_structure.Pin = GPIO_PIN_9;
  gpio_init_structure.Mode = GPIO_MODE_AF_PP;
  gpio_init_structure.Alternate = GPIO_AF7_USART1;
  HAL_GPIO_Init(GPIOA, &gpio_init_structure);

  /* Configure USART Rx as alternate function */
  gpio_init_structure.Pin = GPIO_PIN_7;
  gpio_init_structure.Mode = GPIO_MODE_AF_PP;
  gpio_init_structure.Alternate = GPIO_AF7_USART1;
  HAL_GPIO_Init(GPIOB, &gpio_init_structure);


  /* defining the UART configuration structure */
  uart1.Instance = USART1;
  uart1.Init.BaudRate = 9600;
  uart1.Init.WordLength = UART_WORDLENGTH_8B;
  uart1.Init.StopBits = UART_STOPBITS_1;
  uart1.Init.Parity = UART_PARITY_NONE;
  uart1.Init.HwFlowCtl = UART_HWCONTROL_NONE;
  uart1.Init.Mode = UART_MODE_TX_RX;
  HAL_UART_Init(&uart1);
}
```

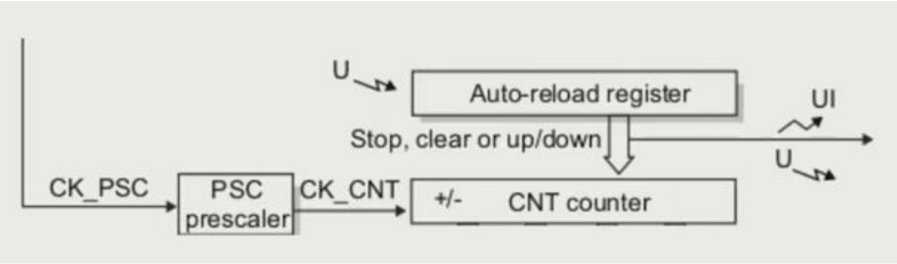


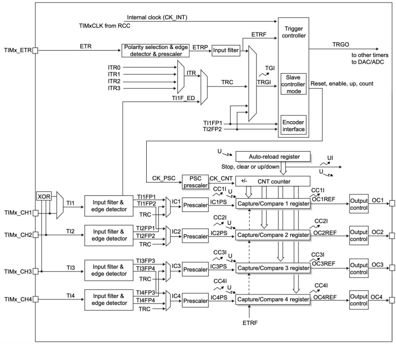
```
void EXTI15_10_IRQHandler(void)
{
  const uint8_t data = '3';

  if (GPIO_PIN_11 == __HAL_GPIO_EXTI_GET_FLAG(GPIO_PIN_11)) {
    HAL_UART_Transmit(&uart1, &data, 1, 0xFFFF); // UART TRANSMIT !!!

    __HAL_GPIO_EXTI_CLEAR_FLAG(GPIO_PIN_11);
    HAL_NVIC_ClearPendingIRQ(EXTI15_10_IRQn);
    btncnt++;
  } else {
    while(1);
  }
}
```

stm32f7xx_hal_uart.c

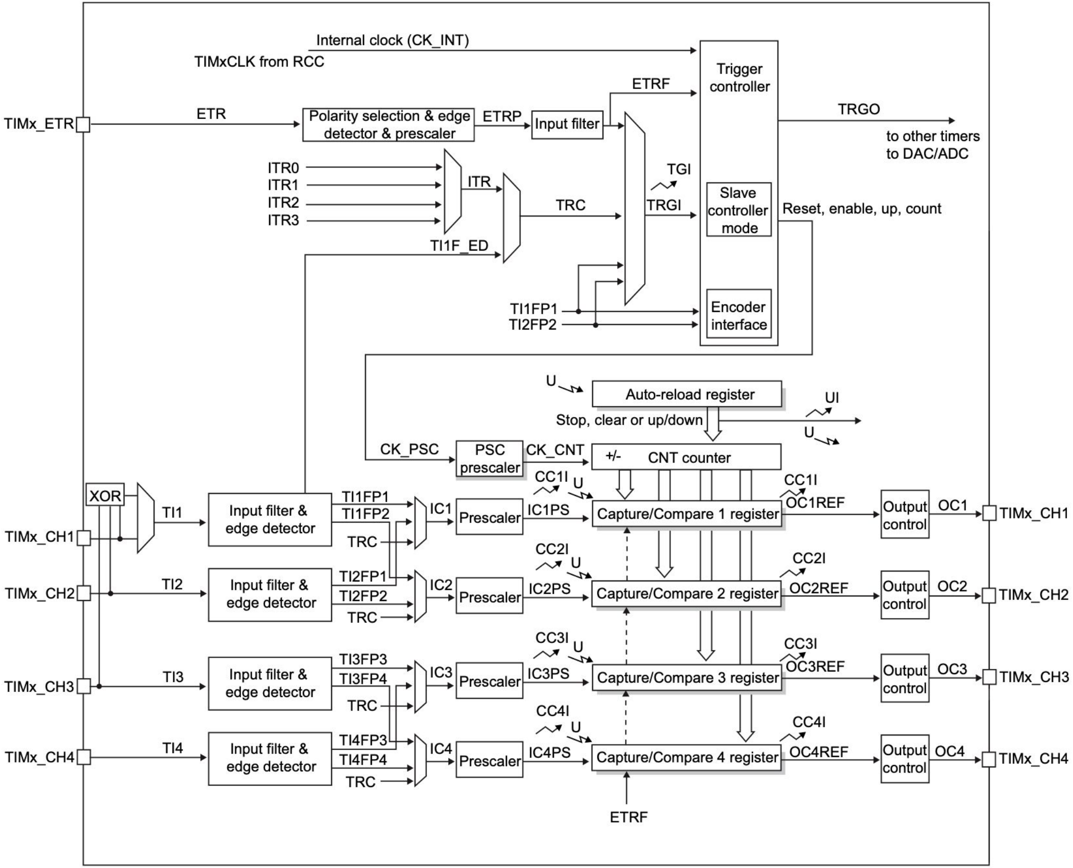
General-purpose TIMx timer features include:

- 16-bit (TIM3, TIM4) or 32-bit (TIM2 and TIM5) up, down, up/down auto-reload counter.
- 16-bit programmable prescaler used to divide (also “on the fly”) the counter clock frequency by any factor between 1 and 65535.
- Up to 4 independent channels for:
  - Input capture
  - Output compare
  - PWM generation (Edge- and Center-aligned modes)
  - One-pulse mode output



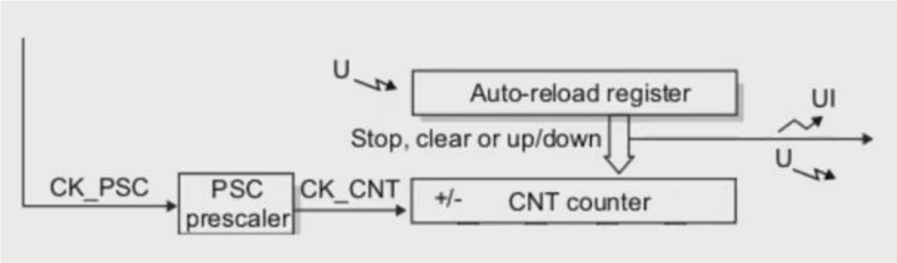


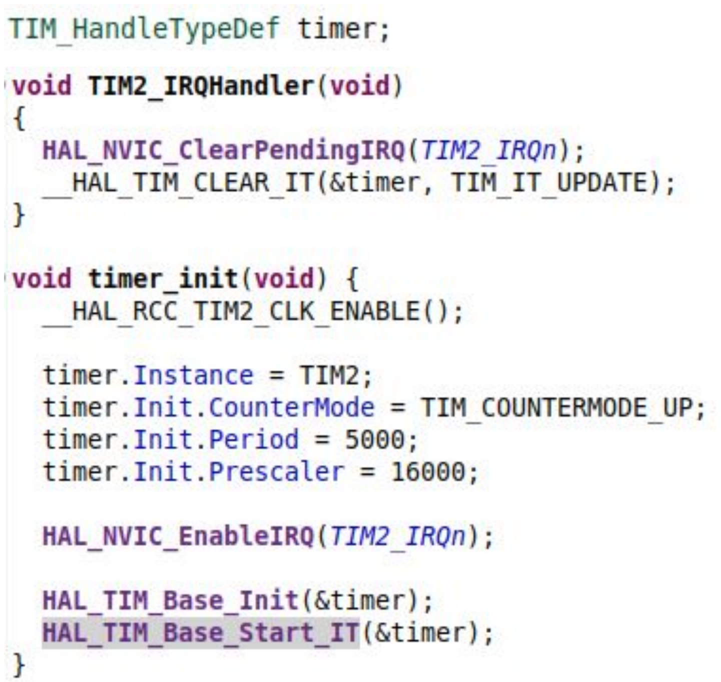
https://www.youtube.com/watch?v=7iBBkNumYlo&list=PLhtMaaf_npBzsEQ94eGn5RnuE-VdGVObR&index=8&ab_channel=%D0%A4%D0%A0%D0%A2%D0%9A%D0%9C%D0%A4%D0%A2%D0%98



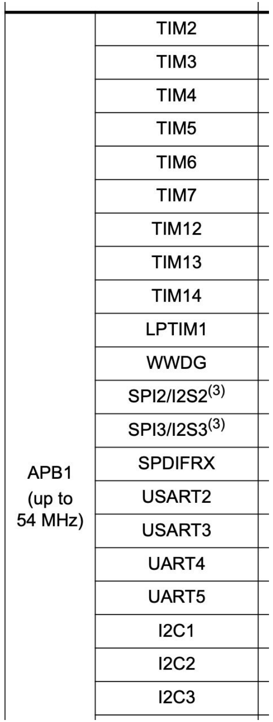
- Обнаружение фронта входного сигнала, запоминание времени, генерация события;
- Генерация события по совпадению кода таймера с заданным значением;
- Формирование ШИМ-сигнала;
- и т.д.

https://www.youtube.com/watch?v=7iBBkNumYlo&list=PLhtMaaf_npBzsEQ94eGn5RnuE-VdGVObR&index=8&ab_channel=%D0%A4%D0%A0%D0%A2%D0%9A%D0%9C%D0%A4%D0%A2%D0%98





```
TIM_HandleTypeDef timer;

void TIM2_IRQHandler(void)
{
  HAL_NVIC_ClearPendingIRQ(TIM2_IRQn);
  __HAL_TIM_CLEAR_IT(&timer, TIM_IT_UPDATE);
}

void timer_init(void) {
  __HAL_RCC_TIM2_CLK_ENABLE();

  timer.Instance = TIM2;
  timer.Init.CounterMode = TIM_COUNTERMODE_UP;
  timer.Init.Period = 5000;
  timer.Init.Prescaler = 16000;

  HAL_NVIC_EnableIRQ(TIM2_IRQn);

  HAL_TIM_Base_Init(&timer);
  HAL_TIM_Base_Start_IT(&timer);
}
```

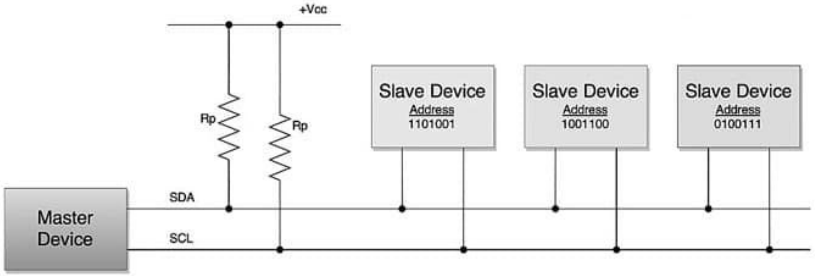


APB1 (up to 54 MHz)	TIM2
	TIM3
	TIM4
	TIM5
	TIM6
	TIM7
	TIM12
	TIM13
	TIM14
	LPTIM1
	WWDG
	SPI2/I2S2(3)
	SPI3/I2S3(3)
	SPDIFRX
	USART2
	USART3
	UART4
	UART5
	I2C1
	I2C2
	I2C3

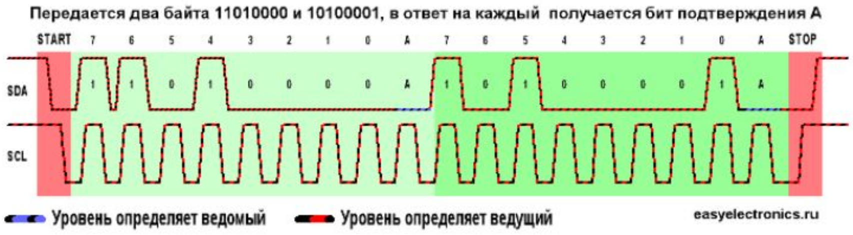
Задание:

- Настроить системный таймер (SysTick) и Timer 2. SysTick считает каждую мс, Timer 2 считает каждую N секунд.
  В обработчике SysTick инкрементируем **переменную** (не забываем volataile) в
  В обработчике Timer 2 проверяем проверяем что **переменная =** (N * 1000) и отправляем значение переменной на ПК через UART.

I2C (*Inter-Integrated Circuit*)



SDA: Шина Данных
SCL: Тактовый Синхросигнал



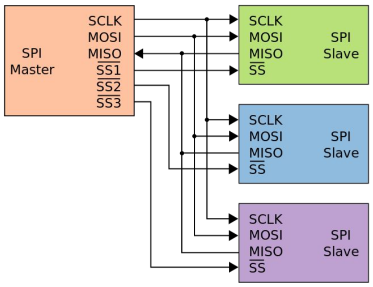
Передается два байта 11010000 и 10100001, в ответ на каждый получается бит подтверждения A
START
7
6
5
4
3
2
1
0
A
7
6
5
4
3
2
1
0
A
STOP
SDA
SCL
1
1
0
1
0
0
0
0
A
1
0
1
0
0
0
0
1
A
Уровень определяет ведомый
Уровень определяет ведущий
easyelectronics.ru

NOTE: *На **SDA** и **SCL** обязательно нужно вешать подтягивающие к питанию резисторы. А **GPIO** настраивать как **Open Drain**.*

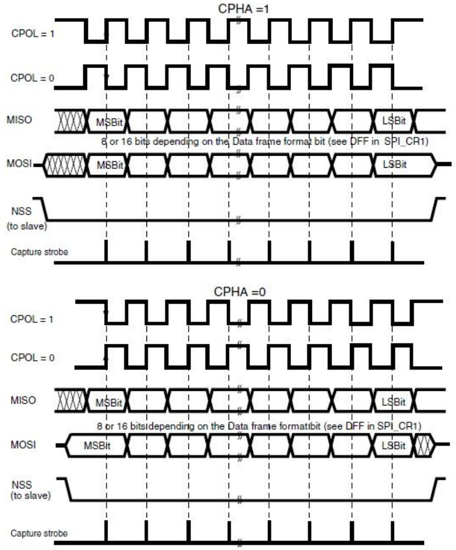
https://www.youtube.com/watch?v=tihKsfD0ASM&list=PLhtMaaf_npBzsEQ94eGn5RnuE-VdGVObR&index=13&ab_channel=%D0%A4%D0%A0%D0%A2%D0%9A%D0%9C%D0%A4%D0%A2%D0%98

http://easyelectronics.ru/interface-bus-iic-i2c.html

SPI (*Serial Peripheral Interface*)

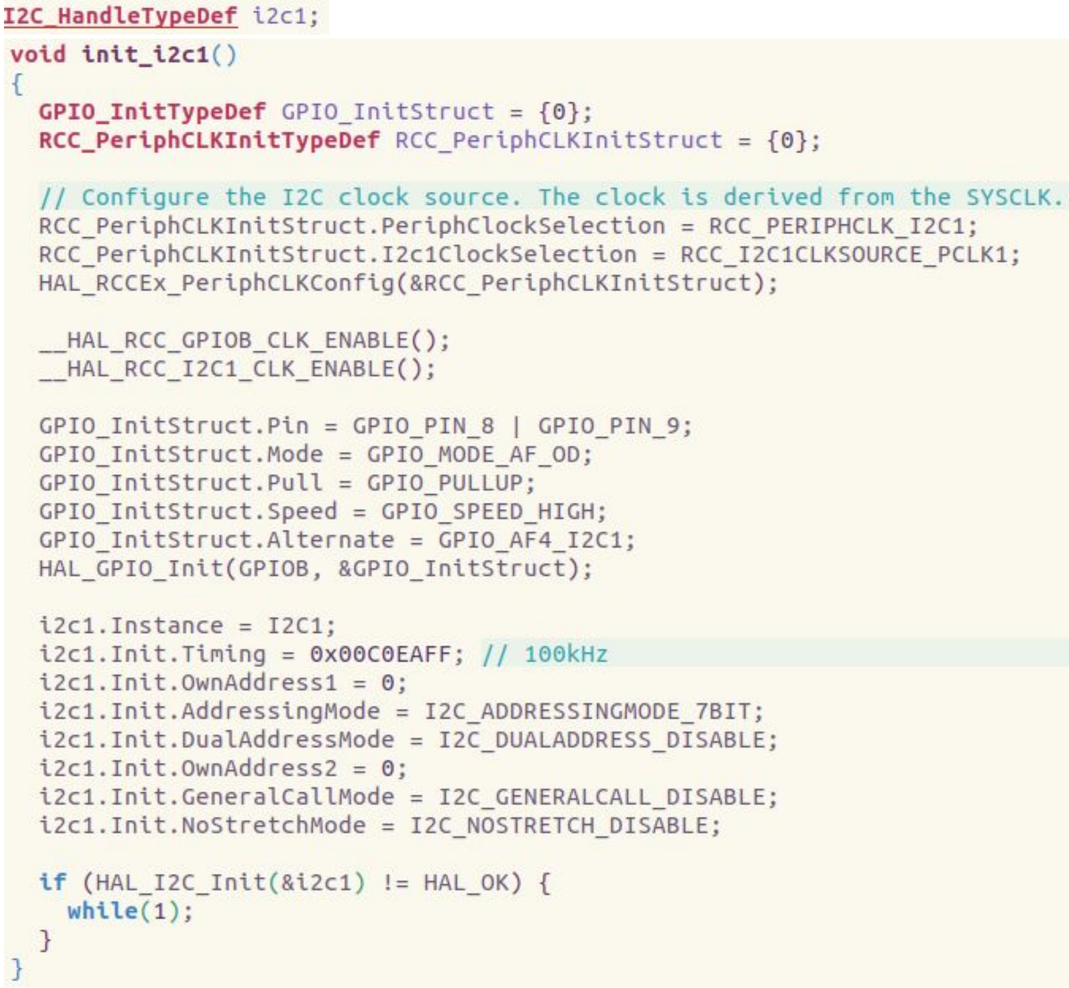


MISO: Master Input
Slave Output
MOSI: Master Output
Slave Input
SCLK: Тактовый Синхросигнал
SS: Slave Select

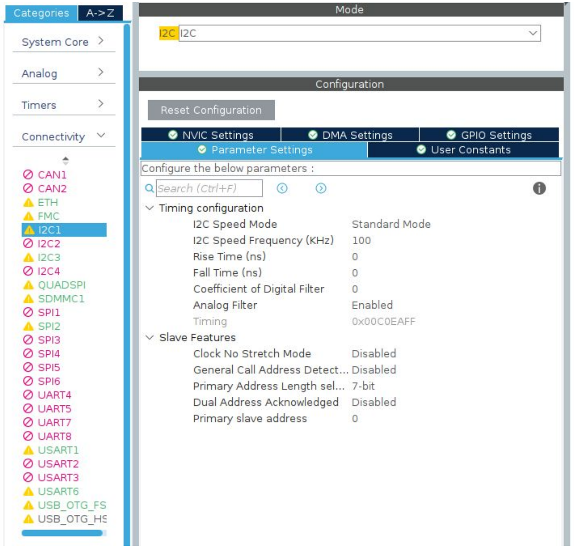


+ Полнодуплексная передача данных по умолчанию.
+ Более высокая пропускная способность
+ Длина пакета не ограничена восемью битами

- Необходимо больше выводов
- Ведомое устройство не может управлять потоком данных
- Нет подтверждения приема данных со стороны ведомого устройства



```c
I2C_HandleTypeDef i2c1;
void init_i2c1()
{
  GPIO_InitTypeDef GPIO_InitStruct = {0};
  RCC_PeriphCLKInitTypeDef RCC_PeriphCLKInitStruct = {0};

  // Configure the I2C clock source. The clock is derived from the SYSCLK.
  RCC_PeriphCLKInitStruct.PeriphClockSelection = RCC_PERIPHCLK_I2C1;
  RCC_PeriphCLKInitStruct.I2c1ClockSelection = RCC_I2C1CLKSOURCE_PCLK1;
  HAL_RCCEx_PeriphCLKConfig(&RCC_PeriphCLKInitStruct);

  __HAL_RCC_GPIOB_CLK_ENABLE();
  __HAL_RCC_I2C1_CLK_ENABLE();

  GPIO_InitStruct.Pin = GPIO_PIN_8 | GPIO_PIN_9;
  GPIO_InitStruct.Mode = GPIO_MODE_AF_OD;
  GPIO_InitStruct.Pull = GPIO_PULLUP;
  GPIO_InitStruct.Speed = GPIO_SPEED_HIGH;
  GPIO_InitStruct.Alternate = GPIO_AF4_I2C1;
  HAL_GPIO_Init(GPIOB, &GPIO_InitStruct);

  i2c1.Instance = I2C1;
  i2c1.Init.Timing = 0x00C0EAFF; // 100kHz
  i2c1.Init.OwnAddress1 = 0;
  i2c1.Init.AddressingMode = I2C_ADDRESSINGMODE_7BIT;
  i2c1.Init.DualAddressMode = I2C_DUALADDRESS_DISABLE;
  i2c1.Init.OwnAddress2 = 0;
  i2c1.Init.GeneralCallMode = I2C_GENERALCALL_DISABLE;
  i2c1.Init.NoStretchMode = I2C_NOSTRETCH_DISABLE;

  if (HAL_I2C_Init(&i2c1) != HAL_OK) {
    while(1);
  }
}
```



I2C I2C

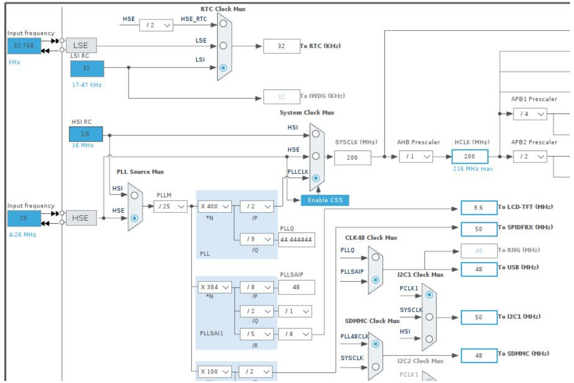
Configure the below parameters :

Timing configuration

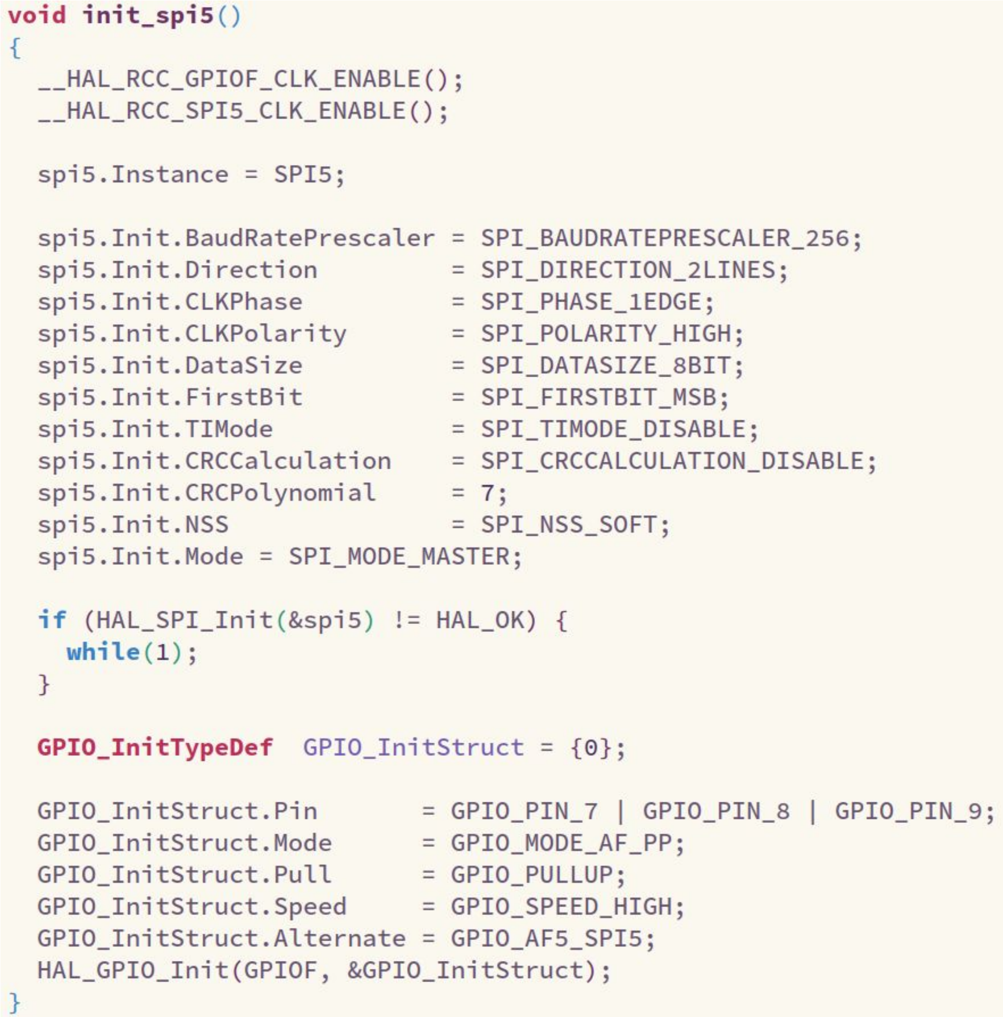
I2C Speed Mode	Standard Mode
I2C Speed Frequency (KHz)	100
Rise Time (ns)	0
Fall Time (ns)	0
Coefficient of Digital Filter	0
Analog Filter	Enabled
Timing	0x00C0EAFF

Slave Features

Clock No Stretch Mode	Disabled
General Call Address Detect...	Disabled
Primary Address Length sel...	7-bit
Dual Address Acknowledged	Disabled
Primary slave address	0

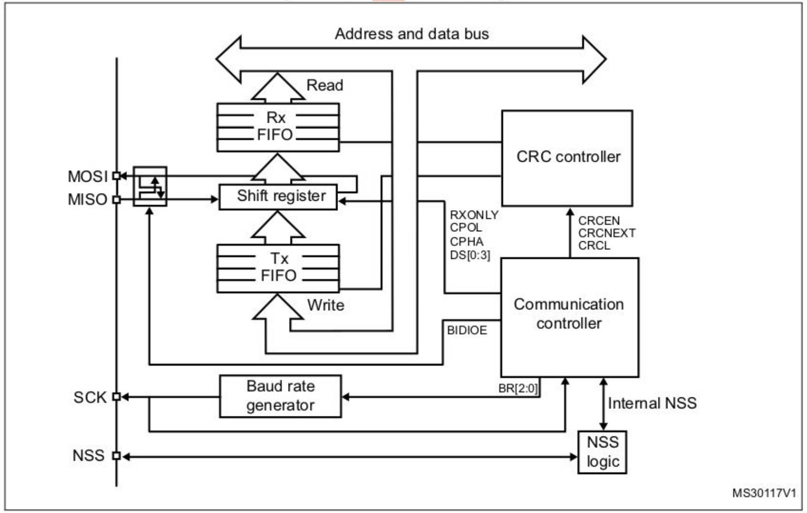


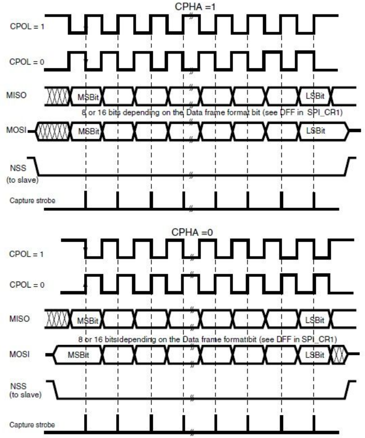
stm32f7xx_hal_i2c.c



```c
void init_spi5()
{
  __HAL_RCC_GPIOF_CLK_ENABLE();
  __HAL_RCC_SPI5_CLK_ENABLE();

  spi5.Instance = SPI5;

  spi5.Init.BaudRatePrescaler = SPI_BAUDRATEPRESCALER_256;
  spi5.Init.Direction         = SPI_DIRECTION_2LINES;
  spi5.Init.CLKPhase          = SPI_PHASE_1EDGE;
  spi5.Init.CLKPolarity       = SPI_POLARITY_HIGH;
  spi5.Init.DataSize          = SPI_DATASIZE_8BIT;
  spi5.Init.FirstBit          = SPI_FIRSTBIT_MSB;
  spi5.Init.TIMode            = SPI_TIMODE_DISABLE;
  spi5.Init.CRCCalculation    = SPI_CRCCALCULATION_DISABLE;
  spi5.Init.CRCPolynomial     = 7;
  spi5.Init.NSS               = SPI_NSS_SOFT;
  spi5.Init.Mode = SPI_MODE_MASTER;

  if (HAL_SPI_Init(&spi5) != HAL_OK) {
    while(1);
  }

  GPIO_InitTypeDef  GPIO_InitStruct = {0};

  GPIO_InitStruct.Pin       = GPIO_PIN_7 | GPIO_PIN_8 | GPIO_PIN_9;
  GPIO_InitStruct.Mode      = GPIO_MODE_AF_PP;
  GPIO_InitStruct.Pull      = GPIO_PULLUP;
  GPIO_InitStruct.Speed     = GPIO_SPEED_HIGH;
  GPIO_InitStruct.Alternate = GPIO_AF5_SPI5;
  HAL_GPIO_Init(GPIOF, &GPIO_InitStruct);
}
```

Figure 357. SPI block diagram





stm32f7xx_hal_spi.c

Что вынести из этой лекции:

- SysTick Configuration
- Interrupts for GPIO
- UART на HAL
- Timer на HAL
- SPI, I2C на HAL

Задание 1:

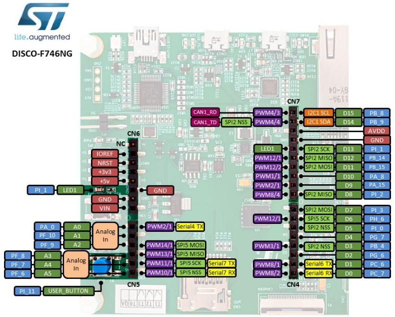
- Завести системный таймер на N миллисекунд, завести таймер общего назначения на M секунд. В прерывание таймера общего назначения передать на ПК через uart(который через USB) значение, которое насчитал системный таймер за период таймера общего назначения.

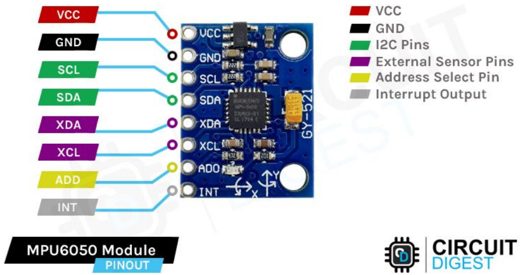
Задание 2:

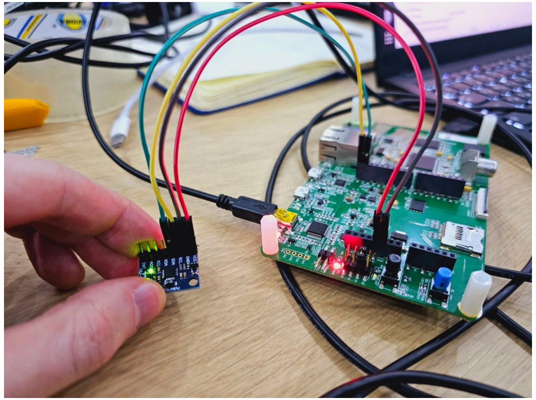
- Настроить I2C, подключить к **MPU6050** микросхеме и попробовать считать из нее данные следующим способом:

```
uint8_t check = 0;
HAL_I2C_Mem_Read(&i2c1, 0xD0, 0x75, 1, &check, 1, 100);
if (check != 104)  { // 0x68 will be returned by the sensor if everything goes well
      while(1);
}
```

- Взять библиотеку для MPU6050 из https://github.com/leech001/MPU6050/blob/master/examples/STM32F401CCU6_MPU6050/Core/Src/mpu6050.c и используя ее считать данные температуры, гироскопа и вывести их на дисплей.

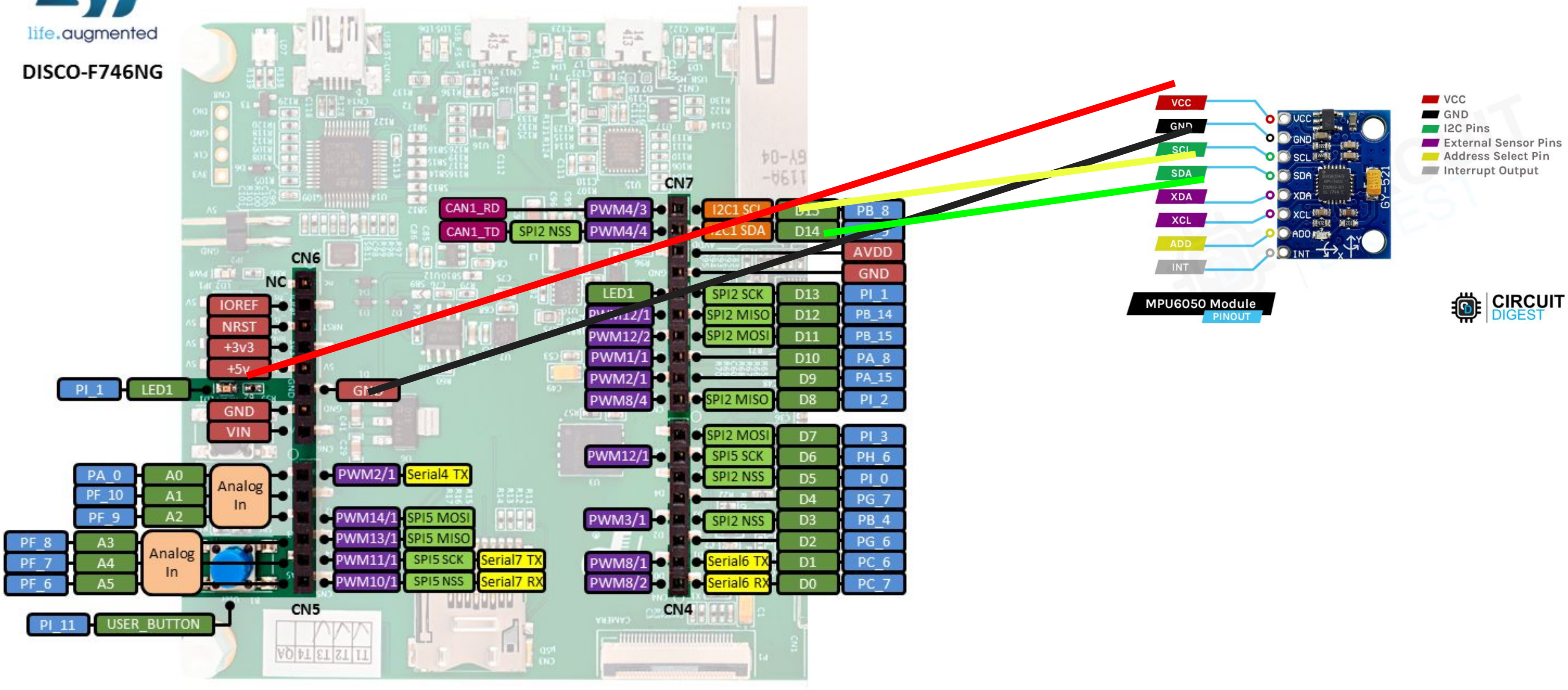


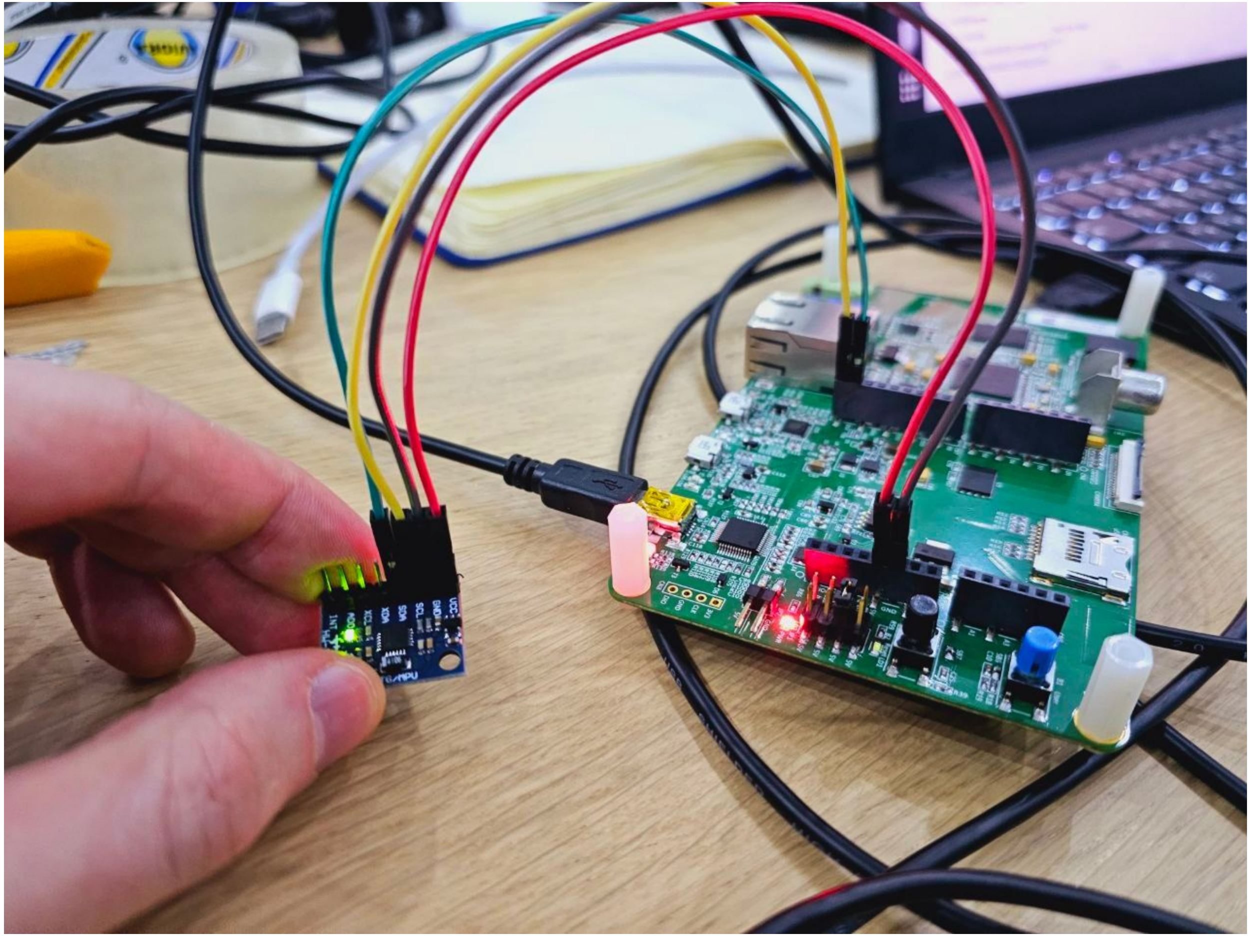




life.augmented

DISCO-F746NG




INT
AD0
XCL
XDA
SDA
SCL
GND
VCC