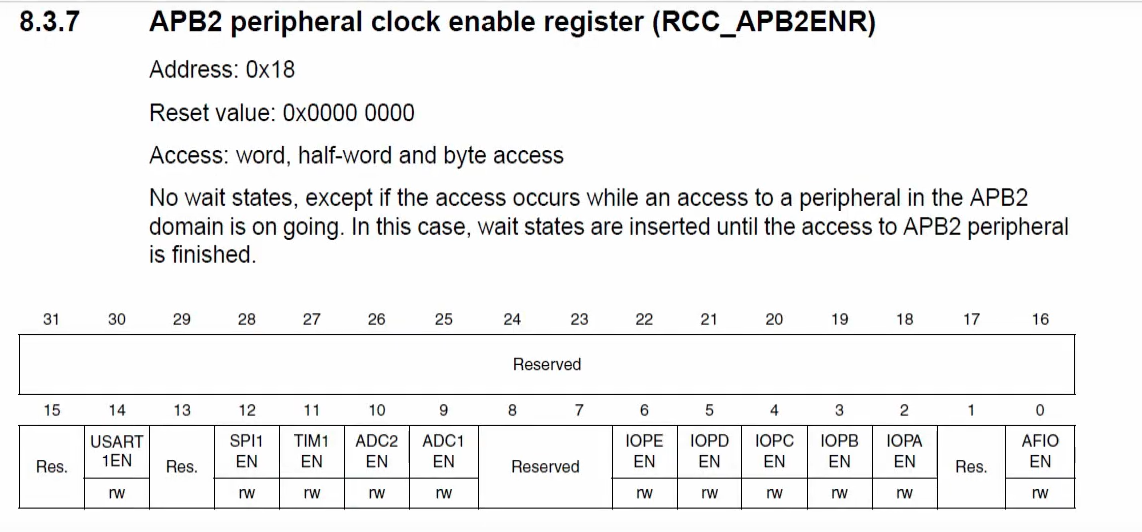
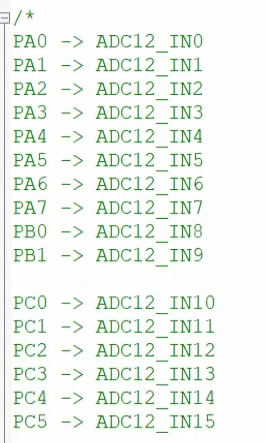
# ADC Basic setup + example

# Initialization:

## 1.Enable clock from APB2ENR, ADC1-> pin 9, ADC2->pin 10. AFIO at pin 0.



## 2. Total 16 channels: Find pin and channel number .



If pin is PA (Low 0-7), then pin number is equal channel number.

If pin is PB (0 and 1), then (pin number +8)= channel number.

If pin is PC ( 0 To 5), then (pin number + 10 ) = channel number.

## 3. ADC regular sequence register setup

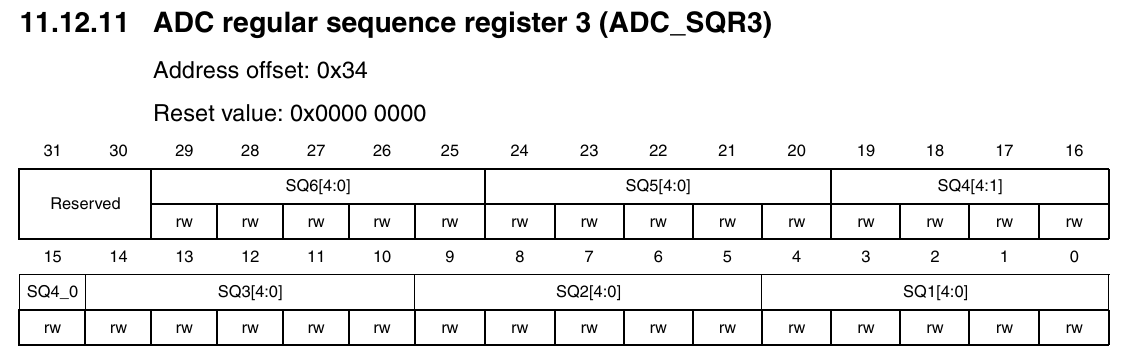
This this the register for scan sequence of the channels.

ADC->SQR1: sequence of 13 to 16

ADC->SQR2: sequence of 7 to 12

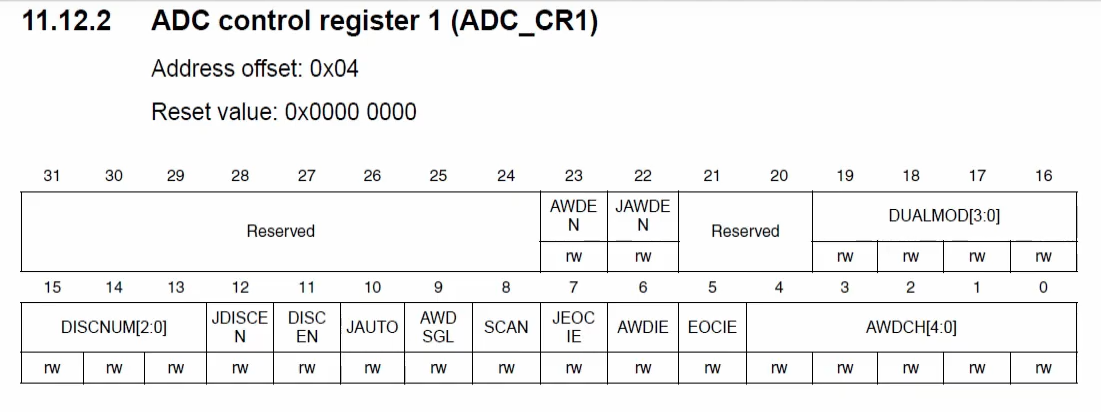
ADC->SQR3: sequence of 0 to 6.

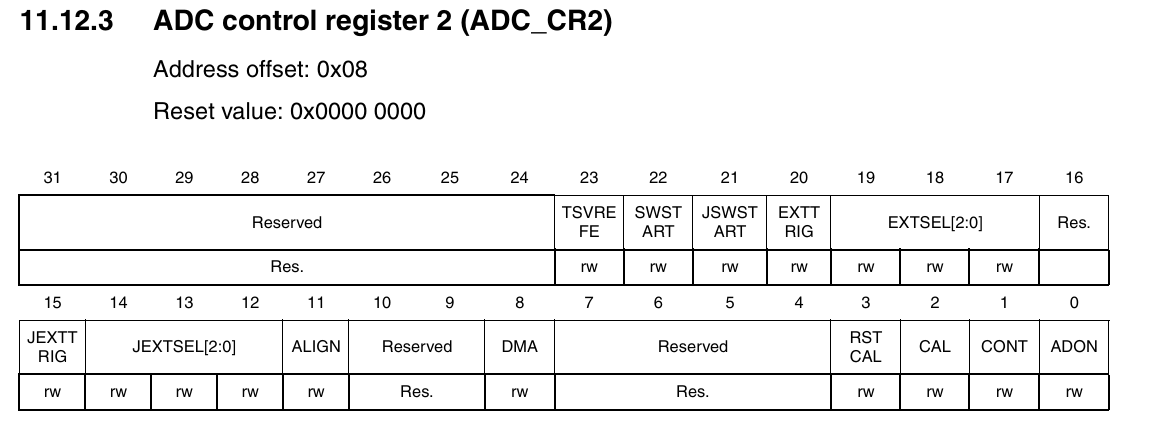
Example we can have a channel sequence like: ch0, ch5, ch1,ch6. These all channel will go to SQR3 from 0 to 3, each slot is 4 bit.



## 

## ADC control register setup





Clear cr1 and cr1 to prevent unwanted configuration

ADC->CR1 =0

ADC->CR2=0;

**Continuous conversion: bit 1 (cont)**

If we want ADC to convert continuously, not stop after single continuous, need to set CONT = 1.

**//before ADON, cont and sqr value must be set**.

**Power on: bit 0 (ADON)**

Then enable ADC by setting bit 0 (ADON). But to start up, we need to powe up bit 0 twice. 1st one do power on, 2nd time start converting.

Example:

ADC->CR2 |= 1UL<<0;

Delay(100ms);

ADC->CR2 |= 1UL<<0;

So the final setup be like: Enabling ADC 1

|  |
| --- |
| // Using ADC1 and pin PA1. which is at channel 1;    RCC->APB2ENR |= RCC\_APB2ENR\_ADC1EN | RCC\_APB2ENR\_AFIOEN;  ADC1->CR2 = 0;  // 1st of all channel and count must be set to proper value, then ADON need to set twitch.  ADC1->SQR3 = 1;  ADC1->CR2 |= ADC\_CR2\_ADON | ADC\_CR2\_CONT;  delay(500);  ADC1->CR2 |= ADC\_CR2\_ADON;//enable adc and start continuous conversion |

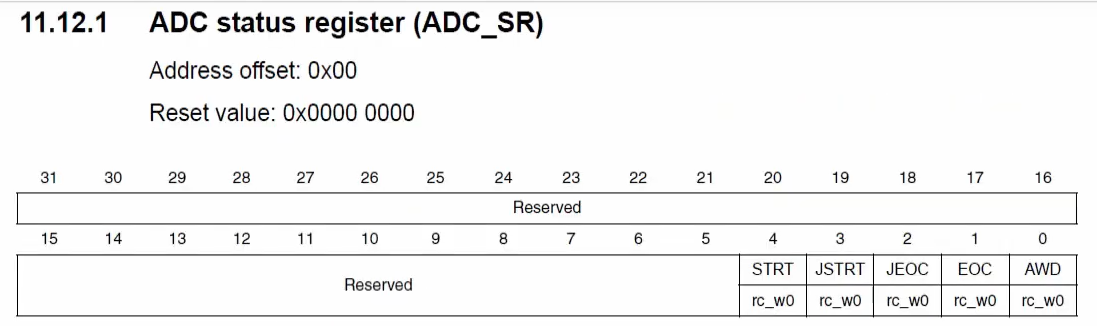
# GPIO pin config

The input pin need to set in analog input mode. Which is CNF =00, MODE = 00;

# ADC\_Check:

Checking ADC has completed conversion or not.

## ADC status register.



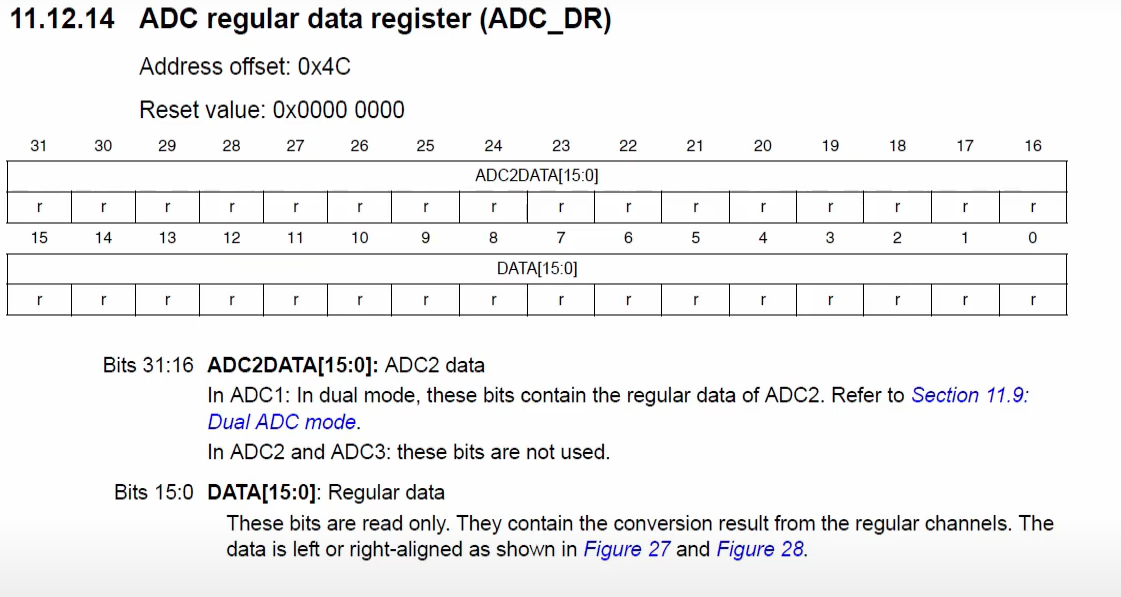
We need the bit 1 (EOC) which is end of conversion. If it is set, then last conversion is done and data is ready to read.

if (ADC1->SR & 1UL<<1){

Read ( converted data);}

# Reading converted data (0 to 2^12)

## ADC regular Data register ADC->DR:



Data = ADC1->DR;

Converting the range (0 to 2^12) to (0 to 1000) to get better visual.

Value 2^12 (0xfff) is equal to 1000

Value 1 is equal to 1000/(2^12)

Value x is equal to (x\*1000)/(2^12);

Result = (data\*1000)/0xfff;

Code:

|  |
| --- |
| //pa1 as analog input  #include<stdint.h>  #include<stm32f10x.h>  void En\_clock(void);  void gpio\_setup(void);  void delay\_ms(void);  void delay(uint32\_t count);  void systick\_config(void);  uint8\_t debounce(uint8\_t last);  void ADC\_config(void);  int main(void){  uint8\_t lastb = 0, currentb=0;  En\_clock();  gpio\_setup();  systick\_config();  ADC\_config();    uint16\_t analog\_data = 0;  while(1){  currentb = debounce(lastb);    //If conversion is done, read the data  if(ADC1->SR & ADC\_SR\_EOC){  analog\_data = ADC1->DR;  analog\_data = analog\_data\*1000/0xfff;  delay(2000);  }  //checking loop main function is running  GPIOA->ODR |= GPIO\_ODR\_ODR3;  delay(20);  GPIOA->ODR &= ~GPIO\_ODR\_ODR3;  delay(20);    }    return 0;  }  void En\_clock(void){  RCC->APB2ENR |= RCC\_APB2ENR\_AFIOEN | RCC\_APB2ENR\_IOPAEN;  }  void gpio\_setup(void){  //PA) as push button  GPIOA->CRL &= ~(GPIO\_CRL\_CNF0 | GPIO\_CRL\_MODE0);  GPIOA->CRL |= GPIO\_CRL\_MODE0;    // PA1 as analog in put cnf=00, mode =00;  GPIOA->CRL &= ~(GPIO\_CRL\_CNF1 | GPIO\_CRL\_MODE1);  GPIOA->CRL |= 0UL;    //PA5 as led output cnf =00, mode = 11;  GPIOA->CRL &= ~(GPIO\_CRL\_CNF5 | GPIO\_CRL\_MODE5);  GPIOA->CRL |= GPIO\_CRL\_MODE5;    //PA2 as push-pull output cnf =00, mode = 11;  GPIOA->CRL &= ~(GPIO\_CRL\_CNF2 | GPIO\_CRL\_MODE2);  GPIOA->CRL |= GPIO\_CRL\_MODE2;  //PA3 as push-pull output cnf =00, mode = 11;  GPIOA->CRL &= ~(GPIO\_CRL\_CNF3 | GPIO\_CRL\_MODE3);  GPIOA->CRL |= GPIO\_CRL\_MODE3;    //PA6 as push-pull output cnf =00, mode = 11;  GPIOA->CRL &= ~(GPIO\_CRL\_CNF6 | GPIO\_CRL\_MODE6);  GPIOA->CRL |= GPIO\_CRL\_MODE6;  }  void systick\_config(void){  SysTick->LOAD = 72000-1;  SysTick->VAL = 0;  SysTick->CTRL = SysTick\_CTRL\_CLKSOURCE | SysTick\_CTRL\_ENABLE;  }  void delay\_ms(void){  while(! ( SysTick->CTRL & SysTick\_CTRL\_COUNTFLAG));  }  void delay(uint32\_t count){  while(count--){  delay\_ms();  }}  uint8\_t debounce(uint8\_t last){  uint8\_t current = (GPIOA->IDR & GPIO\_IDR\_IDR0)? 1 : 0;    if ( last!= current){  delay(5);  current = (GPIOA->IDR & GPIO\_IDR\_IDR0)? 1 : 0;  }  return current;  }  void ADC\_config(void){  // Using ADC1 and pin PA1. which is at channel 1;    RCC->APB2ENR |= RCC\_APB2ENR\_ADC1EN | RCC\_APB2ENR\_AFIOEN;  ADC1->CR2 = 0;  ADC1->SQR3 = 1;  ADC1->CR2 |= ADC\_CR2\_ADON | ADC\_CR2\_CONT;//ADC power on  delay(500);  ADC1->CR2 |= ADC\_CR2\_ADON;//enable adc and start continuous conversion    //delay(10);  //ADC1->CR2 |= ADC\_CR2\_SWSTART;  } |

Turing on light if its dark,

A1 is analog input with photo registor…

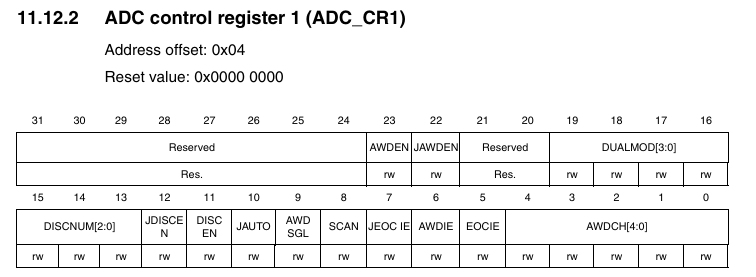
A2, 3, 5 are output depending on registor value.

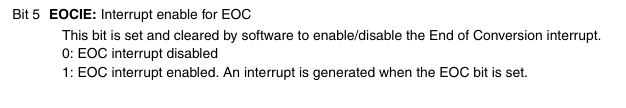
|  |
| --- |
| #include<stdint.h>  #include<stm32f10x.h>  void En\_clock(void);  void gpio\_setup(void);  void delay\_ms(void);  void delay(uint32\_t count);  void systick\_config(void);  uint8\_t debounce(uint8\_t last);  void ADC\_config(void);  int main(void){  uint8\_t lastb = 0, currentb=0;  En\_clock();  gpio\_setup();  systick\_config();  ADC\_config();    uint16\_t analog\_data = 0;    uint16\_t max\_val = 0x0C00;  uint16\_t min\_val = 0x0700;  while(1){      //If conversion is done, read the data  if(ADC1->SR & ADC\_SR\_EOC){  analog\_data = ADC1->DR;  //analog\_data = analog\_data;  }  // Turn on 2 led if its dark, 1 if natural, off if bright    if( analog\_data>max\_val){  //turn on A2 and A3, A5  GPIOA->ODR |= GPIO\_ODR\_ODR2;  GPIOA->ODR |= GPIO\_ODR\_ODR3;  GPIOA->ODR |= GPIO\_ODR\_ODR5;  }  else if ( analog\_data<min\_val){  //Turn on A2. but A3, and A5 turn off  GPIOA->ODR |= GPIO\_ODR\_ODR2;  GPIOA->ODR &= ~GPIO\_ODR\_ODR3;  GPIOA->ODR &= ~GPIO\_ODR\_ODR5;    }  else {    GPIOA->ODR |= GPIO\_ODR\_ODR2;  GPIOA->ODR |= GPIO\_ODR\_ODR3;  GPIOA->ODR &= ~GPIO\_ODR\_ODR5;    }      //checking loop main function is running  GPIOA->ODR |= GPIO\_ODR\_ODR6;  delay(20);  GPIOA->ODR &= ~GPIO\_ODR\_ODR6;  delay(20);    }    return 0;  }  void En\_clock(void){  RCC->APB2ENR |= RCC\_APB2ENR\_AFIOEN | RCC\_APB2ENR\_IOPAEN;  }  void gpio\_setup(void){  //PA) as push button  GPIOA->CRL &= ~(GPIO\_CRL\_CNF0 | GPIO\_CRL\_MODE0);  GPIOA->CRL |= GPIO\_CRL\_MODE0;    // PA1 as analog in put cnf=00, mode =00;  GPIOA->CRL &= ~(GPIO\_CRL\_CNF1 | GPIO\_CRL\_MODE1);  GPIOA->CRL |= 0UL;    //PA5 as led output cnf =00, mode = 11;  GPIOA->CRL &= ~(GPIO\_CRL\_CNF5 | GPIO\_CRL\_MODE5);  GPIOA->CRL |= GPIO\_CRL\_MODE5;    //PA2 as push-pull output cnf =00, mode = 11;  GPIOA->CRL &= ~(GPIO\_CRL\_CNF2 | GPIO\_CRL\_MODE2);  GPIOA->CRL |= GPIO\_CRL\_MODE2;  //PA3 as push-pull output cnf =00, mode = 11;  GPIOA->CRL &= ~(GPIO\_CRL\_CNF3 | GPIO\_CRL\_MODE3);  GPIOA->CRL |= GPIO\_CRL\_MODE3;    //PA6 as push-pull output cnf =00, mode = 11;  GPIOA->CRL &= ~(GPIO\_CRL\_CNF6 | GPIO\_CRL\_MODE6);  GPIOA->CRL |= GPIO\_CRL\_MODE6;  }  void systick\_config(void){  SysTick->LOAD = 72000-1;  SysTick->VAL = 0;  SysTick->CTRL = SysTick\_CTRL\_CLKSOURCE | SysTick\_CTRL\_ENABLE;  }  void delay\_ms(void){  while(! ( SysTick->CTRL & SysTick\_CTRL\_COUNTFLAG));  }  void delay(uint32\_t count){  while(count--){  delay\_ms();  }}  uint8\_t debounce(uint8\_t last){  uint8\_t current = (GPIOA->IDR & GPIO\_IDR\_IDR0)? 1 : 0;    if ( last!= current){  delay(5);  current = (GPIOA->IDR & GPIO\_IDR\_IDR0)? 1 : 0;  }  return current;  }  void ADC\_config(void){  // Using ADC1 and pin PA1. which is at channel 1;    RCC->APB2ENR |= RCC\_APB2ENR\_ADC1EN | RCC\_APB2ENR\_AFIOEN;  ADC1->CR2 = 0;  ADC1->SQR3 = 1;  ADC1->CR2 |= ADC\_CR2\_ADON | ADC\_CR2\_CONT;//ADC power on  delay(500);  ADC1->CR2 |= ADC\_CR2\_ADON;//enable adc and start continuous conversion  } |

# Using EOC interrupt

## Enable interrupt in ADC control register

To enable EOC interrupt, we need to enable EOCIE in ADC1->CR1 at bit position 5.





## Enable NVIC for ADC\_Interrupt

To enable ADC\_interrupt handler, we need to enable NVIC for ADC interrupt

|  |
| --- |
| \_disable\_irq(); //disabling interrupt while working in interrupt vector  NVIC\_EnableIRQ(ADC1\_2\_IRQn); //enabling ADC interrupt.  \_enable\_irq();//enabling interrupts after competing the change |

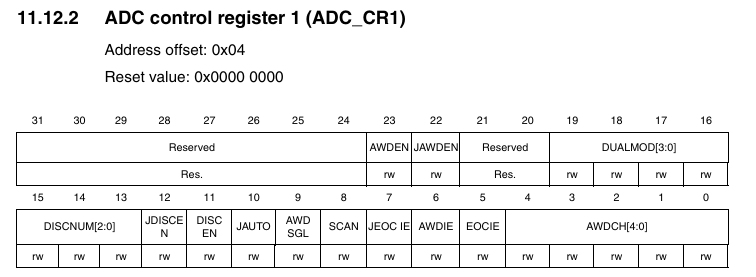
## Edit ADC interrupt handler to get desired output

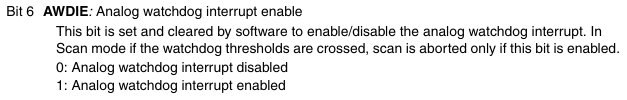
**Void ADC1\_2\_IRQHandler(){} –** This function contains the interrupt response executions. So the desired response code need to write in this function.

# Use of Watch Dog

## Enable Watch\_dog and Watch\_dog\_interrupt from

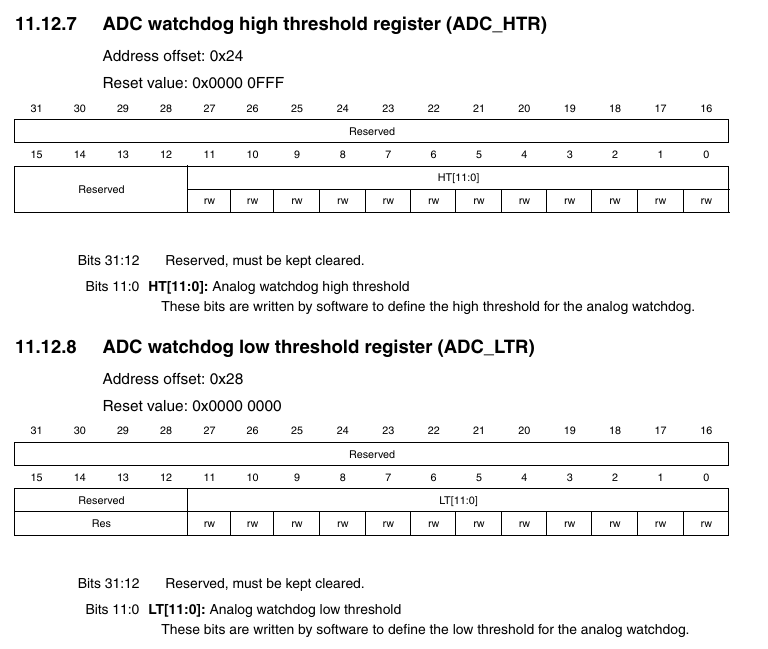
To enable watch dog we need to set AWDEN in ADC1->CR1 at bit position 23, and to enable Watch Dog interrupt, we need to enable AWDIE (analog watch dog interrupt enable) in ADC1->CR1 at bit position 6.



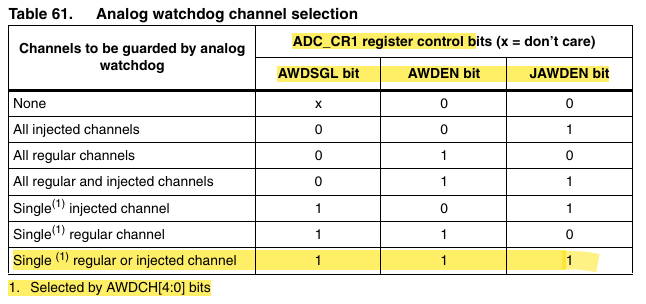


## Setting Threshold values

High Threshold and low threshold value in ADC1->HTR and ADC1->LTR. Value is 12 bit.



## For single channel configuration



## Enable NVIC for ADC\_Interrupt

To enable ADC\_interrupt handler, we need to enable NVIC for ADC interrupt. There is only one int vector for all ADC interrupt.

|  |
| --- |
| \_disable\_irq(); //disabling interrupt while working in interrupt vector  NVIC\_EnableIRQ(ADC1\_2\_IRQn); //enabling ADC interrupt.  \_enable\_irq();//enabling interrupts after competing the change |

## Edit ADC interrupt handler to get desired output

**Void ADC1\_2\_IRQHandler(){} –** This function contains the interrupt response executions. So the desired response code need to write in this function. Here need to check really interrupt is call by WD by checking ADC1->SR\_AWD bit.

|  |
| --- |
| **Void ADC1\_2\_IRQHandler(){**  **If(ADC1->SR & ADC\_SR\_AWD){**  **// user code**  **}**  **}** |

## Checking really a AWD happened or not:

