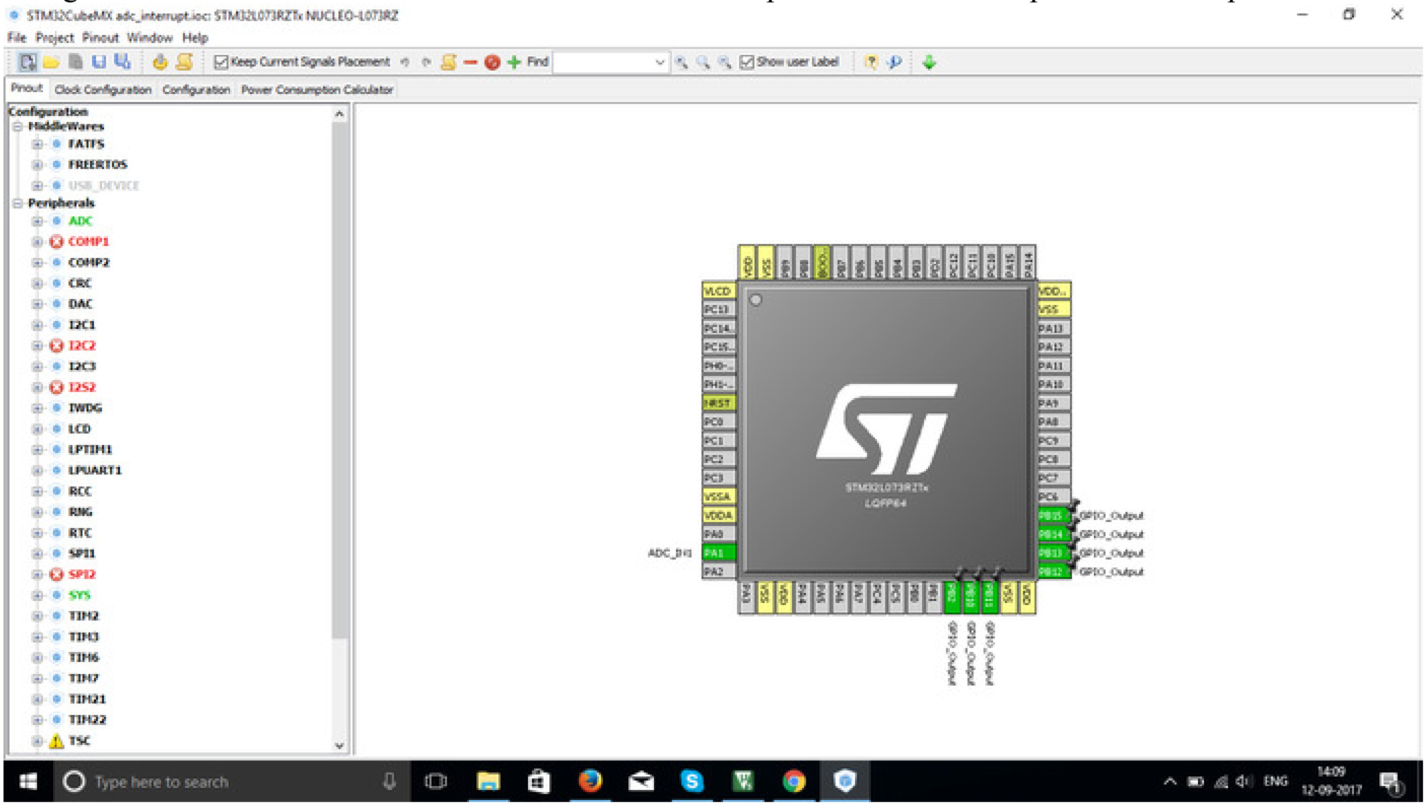
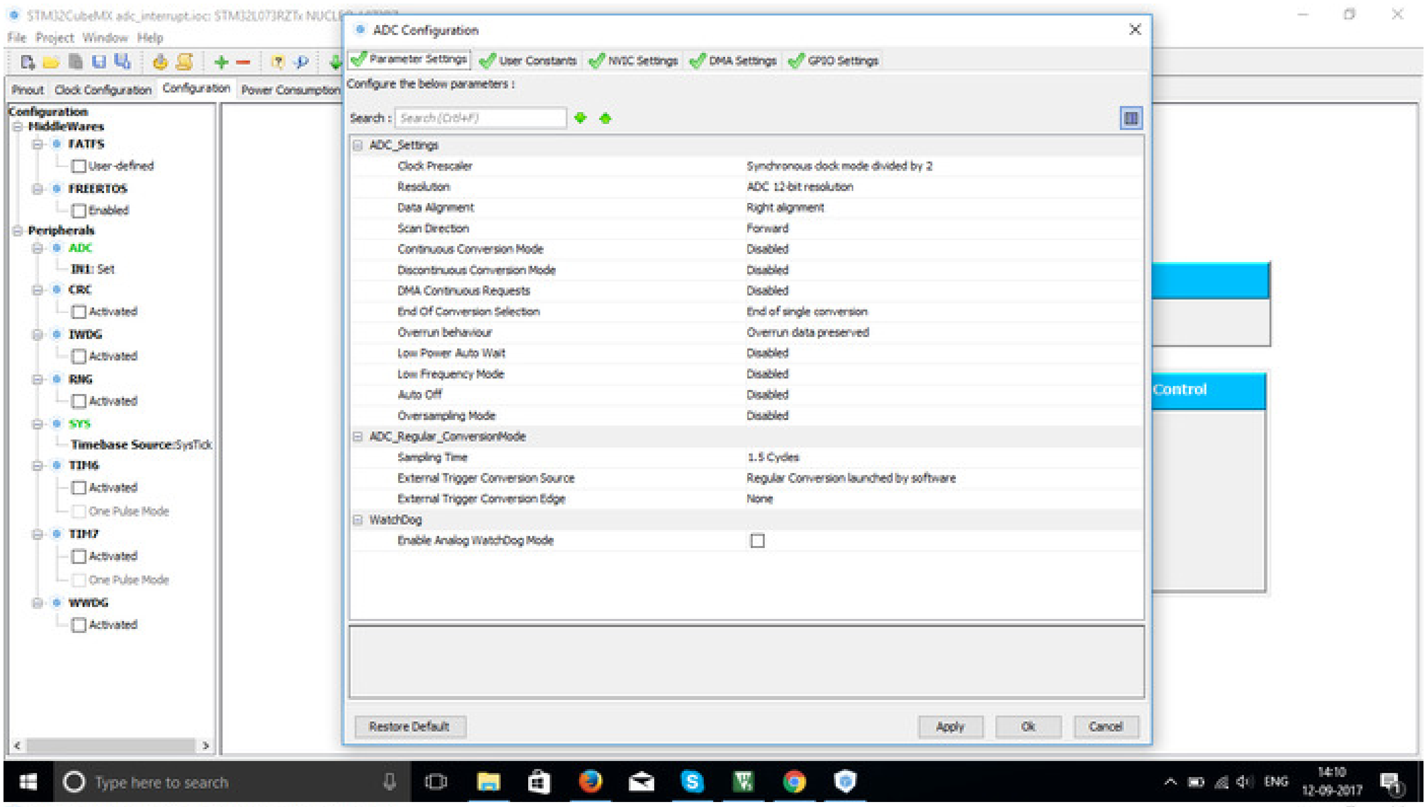
I have interfaced a lcd with stm32l073 nucleo-64 board and getting variable data on analog pin of controller.I have trying to get analog data on lcd and i want to show the recent data whenever voltage on adc pin changes.but I was not successful.Please find documents and pics attached and help to correct the problem.

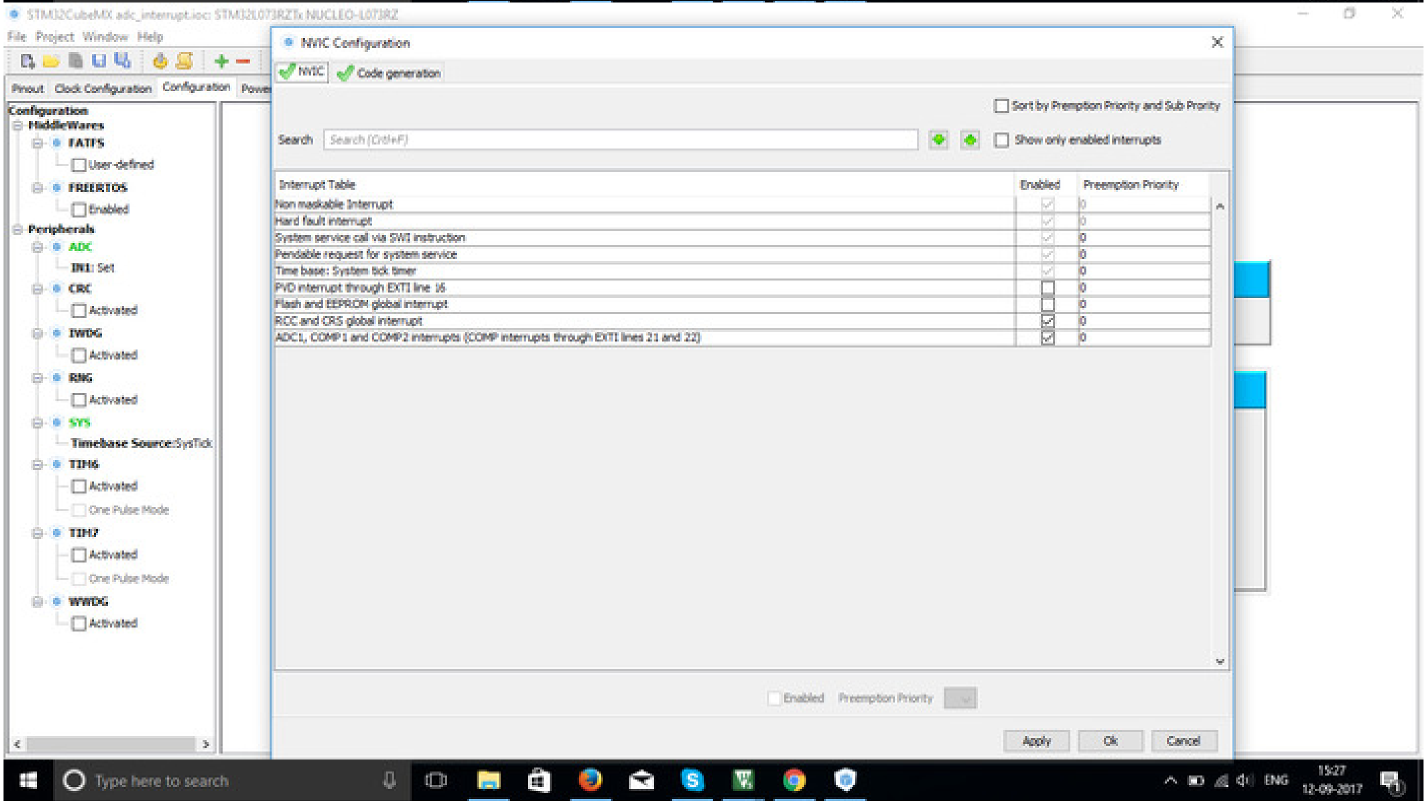



ADC Configuration

Parameter Settings | User Constants | NVIC Settings | DMA Settings | GPIO Settings

Configure the below parameters :

ADC_Settings	
Clock Prescaler	Synchronous clock mode divided by 2
Resolution	ADC 12-bit resolution
Data Alignment	Right alignment
Scan Direction	Forward
Continuous Conversion Mode	Disabled
Discontinuous Conversion Mode	Disabled
DMA Continuous Requests	Disabled
End Of Conversion Selection	End of single conversion
Overrun behaviour	Overrun data preserved
Low Power Auto Wait	Disabled
Low Frequency Mode	Disabled
Auto Off	Disabled
Oversampling Mode	Disabled
ADC_Regular_ConversionMode	
Sampling Time	1.5 Cycles
External Trigger Conversion Source	Regular Conversion launched by software
External Trigger Conversion Edge	None
WatchDog	
Enable Analog WatchDog Mode	☐


NVIC Configuration

NVIC | Code generation

☐ Sort by Premption Priority and Sub Priority

☐ Show only enabled interrupts

Interrupt Table	Enabled	Preemption Priority
Non maskable Interrupt	✓	0
Hard fault interrupt	✓	0
System service call via SWI instruction	✓	0
Pendable request for system service	✓	0
Time base: System tick timer	✓	0
PVD interrupt through EXTI line 16	☐	0
Flash and EEPROM global interrupt	☐	0
RCC and CRS global interrupt	✓	0
ADC1, COMP1 and COMP2 interrupts (COMP interrupts through EXTI lines 21 and 22)	✓	0

main.c file

```
/**
******************************************************************************
* File Name : main.c
* Description : Main program body
******************************************************************************
*
* COPYRIGHT(c) 2017 STMicroelectronics
*
* Redistribution and use in source and binary forms, with or without modification,
* are permitted provided that the following conditions are met:
* 1. Redistributions of source code must retain the above copyright notice,
* this list of conditions and the following disclaimer.
* 2. Redistributions in binary form must reproduce the above copyright notice,
* this list of conditions and the following disclaimer in the documentation
* and/or other materials provided with the distribution.
* 3. Neither the name of STMicroelectronics nor the names of its contributors
* may be used to endorse or promote products derived from this software
* without specific prior written permission.
*
* THIS SOFTWARE IS PROVIDED BY THE COPYRIGHT HOLDERS AND CONTRIBUTORS "AS IS"
* AND ANY EXPRESS OR IMPLIED WARRANTIES, INCLUDING, BUT NOT LIMITED TO, THE
* IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR PURPOSE
ARE
* DISCLAIMED. IN NO EVENT SHALL THE COPYRIGHT HOLDER OR CONTRIBUTORS BE
LIABLE
* FOR ANY DIRECT, INDIRECT, INCIDENTAL, SPECIAL, EXEMPLARY, OR CONSEQUENTIAL
* DAMAGES (INCLUDING, BUT NOT LIMITED TO, PROCUREMENT OF SUBSTITUTE GOODS OR
* SERVICES; LOSS OF USE, DATA, OR PROFITS; OR BUSINESS INTERRUPTION) HOWEVER
* CAUSED AND ON ANY THEORY OF LIABILITY, WHETHER IN CONTRACT, STRICT LIABILITY,
* OR TORT (INCLUDING NEGLIGENCE OR OTHERWISE) ARISING IN ANY WAY OUT OF THE USE
* OF THIS SOFTWARE, EVEN IF ADVISED OF THE POSSIBILITY OF SUCH DAMAGE.
*
******************************************************************************
*/
/* Includes ------------------------------------------------------------------*/
#include "main.h"
#include "stm32l0xx_hal.h"
#include "lcd_hd44780_stm32l0.h"

/* USER CODE BEGIN Includes */
volatile uint16_t AdcValue;

/* USER CODE END Includes */

/* Private variables ---------------------------------------------------------*/
ADC_HandleTypeDef hadc;

/* USER CODE BEGIN PV */
/* Private variables ---------------------------------------------------------*/

/* USER CODE END PV */

/* Private function prototypes -----------------------------------------------*/
void SystemClock_Config(void);
void Error_Handler(void);
static void MX_GPIO_Init(void);
static void MX_ADC_Init(void);

/* USER CODE BEGIN PFP */
/* Private function prototypes -----------------------------------------------*/

/* USER CODE END PFP */

/* USER CODE BEGIN 0 */

/* USER CODE END 0 */

int main(void)
{

/* USER CODE BEGIN 1 */
/* USER CODE END 1 */

/* MCU Configuration----------------------------------------------------------*/

/* Reset of all peripherals, Initializes the Flash interface and the Systick. */
HAL_Init();

/* Configure the system clock */
SystemClock_Config();

/* Initialize all configured peripherals */
MX_GPIO_Init();
MX_ADC_Init();
LCDInit(0x00);
HAL_ADC_Start_IT(&hadc);
HAL_ADC_ConvCpltCallback(&hadc);

/* USER CODE BEGIN 2 */

/* USER CODE END 2 */

/* Infinite loop */
/* USER CODE BEGIN WHILE */
while (1)
{
/* USER CODE END WHILE */

/* USER CODE BEGIN 3 */

// AdcValue= HAL_ADC_GetValue(&hadc);
LCDGotoXY(0,0);
LCDWriteInt(AdcValue,4);



}
/* USER CODE END 3 */

}

/** System Clock Configuration
*/
void SystemClock_Config(void)
{

RCC_OscInitTypeDef RCC_OscInitStruct;
RCC_ClkInitTypeDef RCC_ClkInitStruct;

/**Configure the main internal regulator output voltage
*/
__HAL_PWR_VOLTAGESCALING_CONFIG(PWR_REGULATOR_VOLTAGE_SCALE1);

/**Initializes the CPU, AHB and APB busses clocks
*/
RCC_OscInitStruct.OscillatorType = RCC_OSCILLATORTYPE_HSI;
RCC_OscInitStruct.HSIState = RCC_HSI_ON;
RCC_OscInitStruct.HSICalibrationValue = 16;
RCC_OscInitStruct.PLL.PLLState = RCC_PLL_ON;
RCC_OscInitStruct.PLL.PLLSource = RCC_PLLSOURCE_HSI;
RCC_OscInitStruct.PLL.PLLMUL = RCC_PLLMUL_4;
RCC_OscInitStruct.PLL.PLLDIV = RCC_PLLDIV_2;
if (HAL_RCC_OscConfig(&RCC_OscInitStruct) != HAL_OK)
{
Error_Handler();
}

/**Initializes the CPU, AHB and APB busses clocks
*/
RCC_ClkInitStruct.ClockType = RCC_CLOCKTYPE_HCLK|RCC_CLOCKTYPE_SYSCLK
|RCC_CLOCKTYPE_PCLK1|RCC_CLOCKTYPE_PCLK2;
RCC_ClkInitStruct.SYSCLKSource = RCC_SYSCLKSOURCE_PLLCLK;
RCC_ClkInitStruct.AHBCLKDivider = RCC_SYSCLK_DIV1;
RCC_ClkInitStruct.APB1CLKDivider = RCC_HCLK_DIV1;
RCC_ClkInitStruct.APB2CLKDivider = RCC_HCLK_DIV1;

if (HAL_RCC_ClockConfig(&RCC_ClkInitStruct, FLASH_LATENCY_1) != HAL_OK)
{
Error_Handler();
}

/**Configure the Systick interrupt time
*/
HAL_SYSTICK_Config(HAL_RCC_GetHCLKFreq()/1000);

/**Configure the Systick
*/
HAL_SYSTICK_CLKSourceConfig(SYSTICK_CLKSOURCE_HCLK);

/* SysTick_IRQn interrupt configuration */
HAL_NVIC_SetPriority(SysTick_IRQn, 0, 0);
}
void HAL_ADC_ConvCpltCallback(ADC_HandleTypeDef* hadc)
{
AdcValue=HAL_ADC_GetValue(hadc);

}

/* ADC init function */
static void MX_ADC_Init(void)
{

ADC_ChannelConfTypeDef sConfig;

/**Configure the global features of the ADC (Clock, Resolution, Data Alignment and number of conversion)
*/
hadc.Instance = ADC1;
hadc.Init.OversamplingMode = DISABLE;
hadc.Init.ClockPrescaler = ADC_CLOCK_SYNC_PCLK_DIV2;
hadc.Init.Resolution = ADC_RESOLUTION_12B;
hadc.Init.SamplingTime = ADC_SAMPLETIME_1CYCLE_5;
hadc.Init.ScanConvMode = ADC_SCAN_DIRECTION_FORWARD;
hadc.Init.DataAlign = ADC_DATAALIGN_RIGHT;
hadc.Init.ContinuousConvMode = DISABLE;
hadc.Init.DiscontinuousConvMode = DISABLE;
hadc.Init.ExternalTrigConvEdge = ADC_EXTERNALTRIGCONVEDGE_NONE;
hadc.Init.ExternalTrigConv = ADC_SOFTWARE_START;
hadc.Init.DMAContinuousRequests = DISABLE;
hadc.Init.EOCSelection = ADC_EOC_SINGLE_CONV;
hadc.Init.Overrun = ADC_OVR_DATA_PRESERVED;
hadc.Init.LowPowerAutoWait = DISABLE;
hadc.Init.LowPowerFrequencyMode = DISABLE;
hadc.Init.LowPowerAutoPowerOff = DISABLE;
if (HAL_ADC_Init(&hadc) != HAL_OK)
{
Error_Handler();
}

/**Configure for the selected ADC regular channel to be converted.
*/
sConfig.Channel = ADC_CHANNEL_1;
sConfig.Rank = ADC_RANK_CHANNEL_NUMBER;
if (HAL_ADC_ConfigChannel(&hadc, &sConfig) != HAL_OK)
{
Error_Handler();
}

}

/** Configure pins as
* Analog
* Input
* Output
* EVENT_OUT
* EXTI
*/
static void MX_GPIO_Init(void)
{

GPIO_InitTypeDef GPIO_InitStruct;

/* GPIO Ports Clock Enable */
__HAL_RCC_GPIOA_CLK_ENABLE();
__HAL_RCC_GPIOB_CLK_ENABLE();

/*Configure GPIO pin Output Level */
HAL_GPIO_WritePin(GPIOB, GPIO_PIN_2|GPIO_PIN_10|GPIO_PIN_11|GPIO_PIN_12
|GPIO_PIN_13|GPIO_PIN_14|GPIO_PIN_15, GPIO_PIN_RESET);

/*Configure GPIO pins : PB2 PB10 PB11 PB12
PB13 PB14 PB15 */
GPIO_InitStruct.Pin = GPIO_PIN_2|GPIO_PIN_10|GPIO_PIN_11|GPIO_PIN_12
|GPIO_PIN_13|GPIO_PIN_14|GPIO_PIN_15;
GPIO_InitStruct.Mode = GPIO_MODE_OUTPUT_PP;
GPIO_InitStruct.Pull = GPIO_NOPULL;
GPIO_InitStruct.Speed = GPIO_SPEED_FREQ_LOW;
HAL_GPIO_Init(GPIOB, &GPIO_InitStruct);

}

/* USER CODE BEGIN 4 */

/* USER CODE END 4 */

/**
* @brief This function is executed in case of error occurrence.
* @param None
* @retval None
*/
void Error_Handler(void)
```

```
{
/* USER CODE BEGIN Error_Handler */
/* User can add his own implementation to report the HAL error return state */
while(1)
{
}
/* USER CODE END Error_Handler */
}

#ifdef USE_FULL_ASSERT

/**
* @brief Reports the name of the source file and the source line number
* where the assert_param error has occurred.
* @param file: pointer to the source file name
* @param line: assert_param error line source number
* @retval None
*/
void assert_failed(uint8_t* file, uint32_t line)
{
/* USER CODE BEGIN 6 */
/* User can add his own implementation to report the file name and line number,
ex: printf("Wrong parameters value: file %s on line %d\r\n", file, line) */
/* USER CODE END 6 */

}

#endif

/**
* @}
*/

/**
* @}
*/

/************************ (C) COPYRIGHT STMicroelectronics *****END OF FILE****/
```