Embedded Systems

CS 397

TRIMESTER 3, AY 2021/22

Hands-On 1-3
STM32CubeIDE: DAC, ADC with Interrupt and DMA

Dr. LIAW Hwee Choo

Department of Electrical and Computer Engineering

DigiPen Institute of Technology Singapore

HweeChoo.Liaw@DigiPen.edu

Objectives

The aims of this session are to

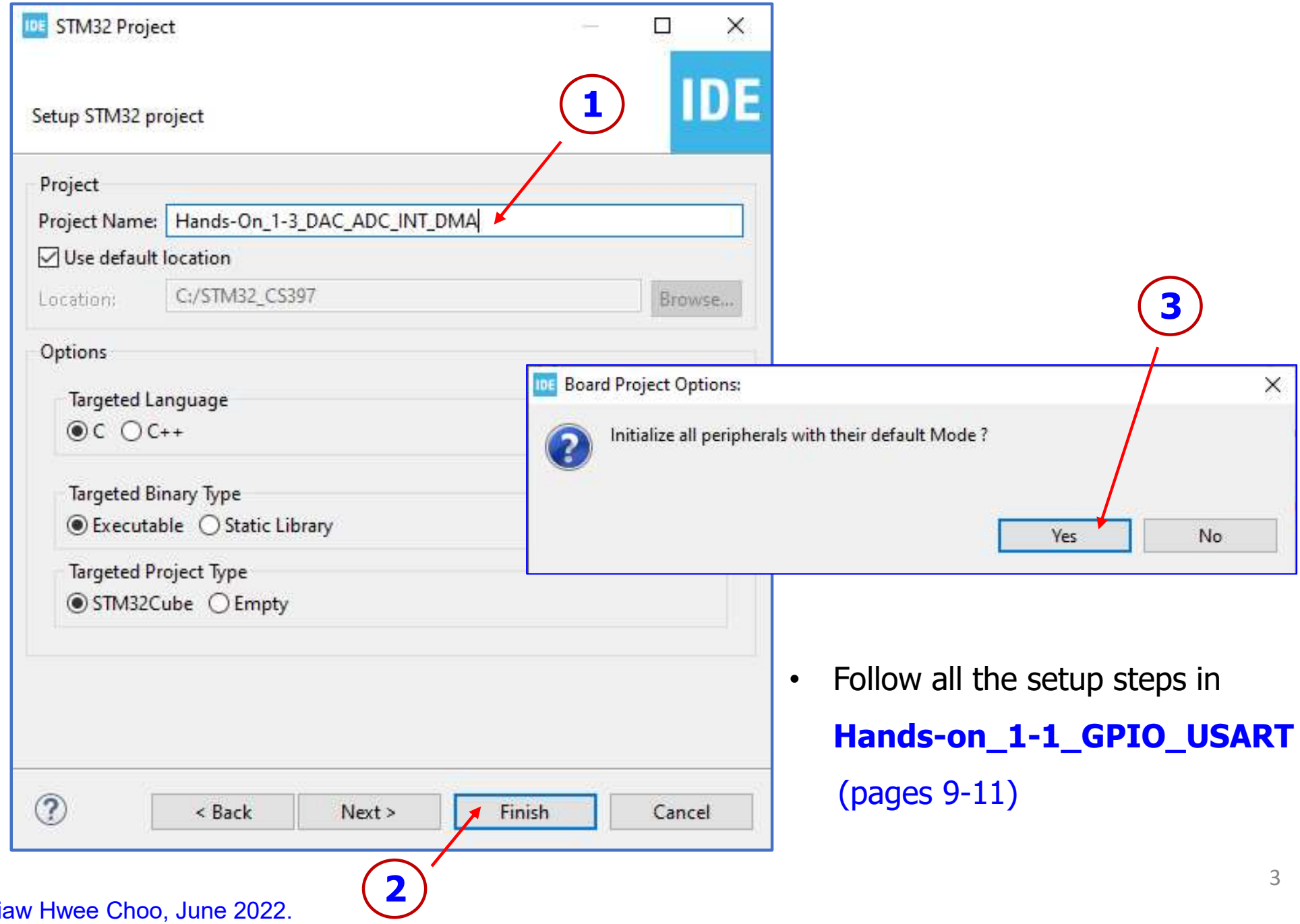
- implement a STM32 (STM32CubeIDE) project
- develop an application using STM32F767ZI microcontroller
- program the DAC and ADC with interrupt and DMA
- test programs using "TM Terminal" or "RealTerm", a serial terminal (COM port) software
- build-up the development knowledge of embedded application

    - Run STM32CubeIDE
    - Select workspace: C:\STM32_CS397
    - File -> Close All Editors
    - Start a New STM32 Project
    - Select the Nucleo-F767ZI Board

Liaw Hwee Choo, June 2022.

STM32CubeIDE – DAC ADC

Enter Project Name: Hands-On_1-3_DAC_ADC_INT_DMA

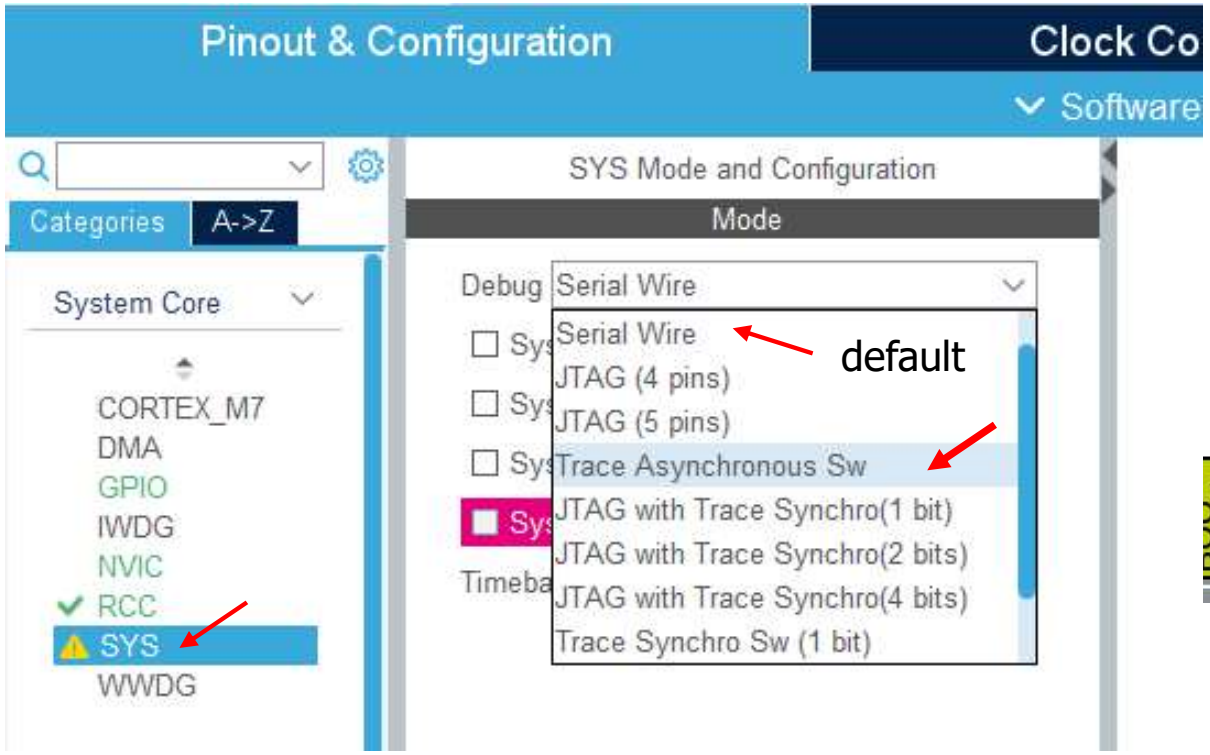


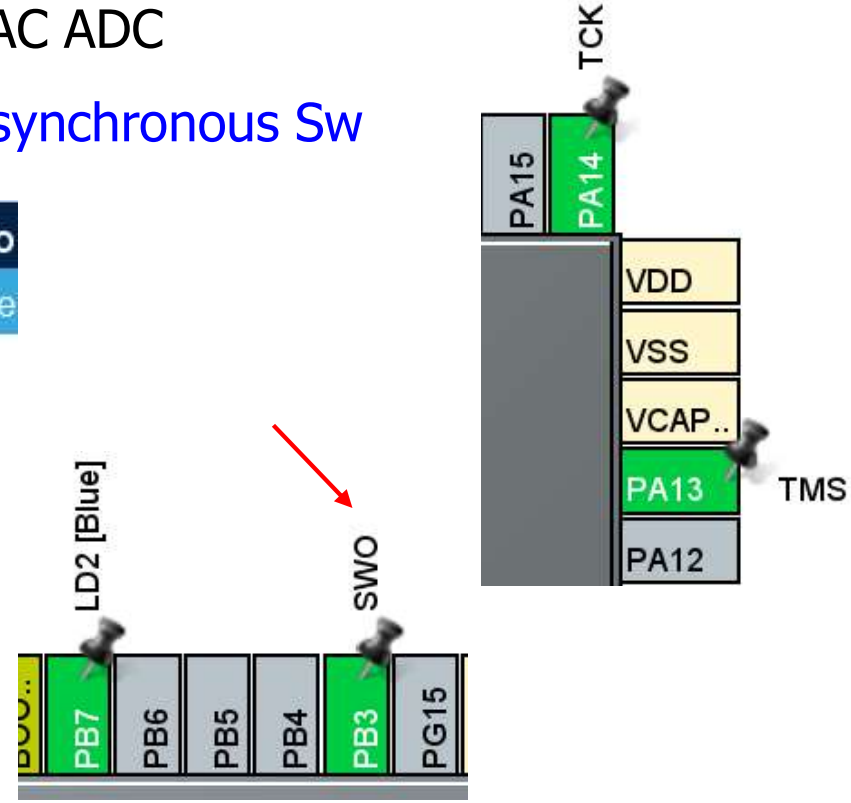
- Follow all the setup steps in **Hands-on_1-1_GPIO_USART** (pages 9-11)

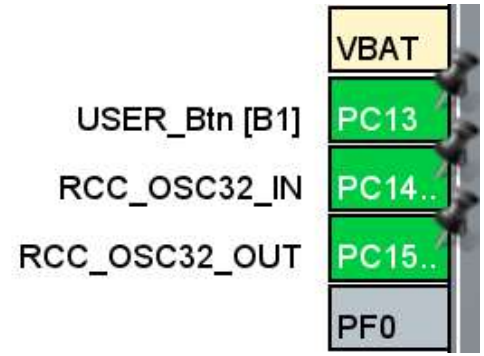
Liaw Hwee Choo, June 2022.

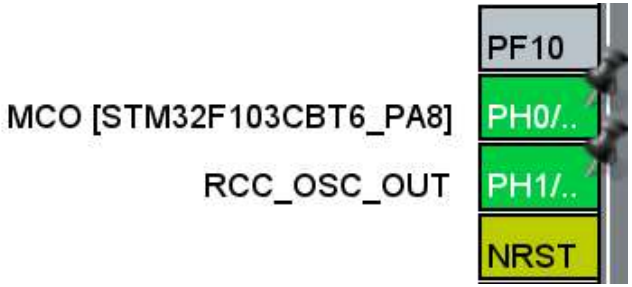
STM32CubeIDE – DAC ADC

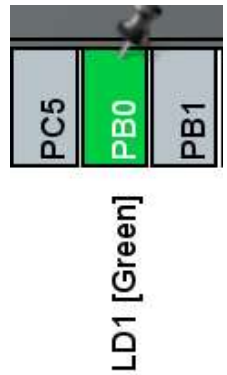
Select: System Core -> SYS -> Debug: Trace Asynchronous Sw

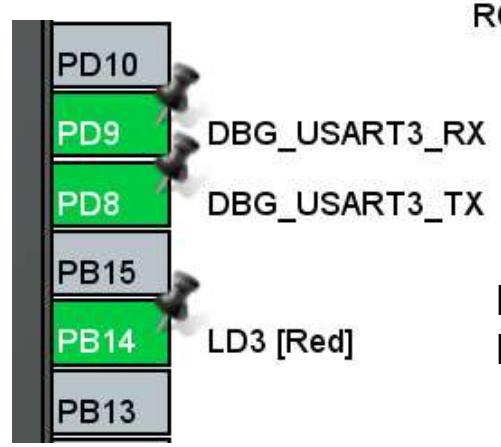












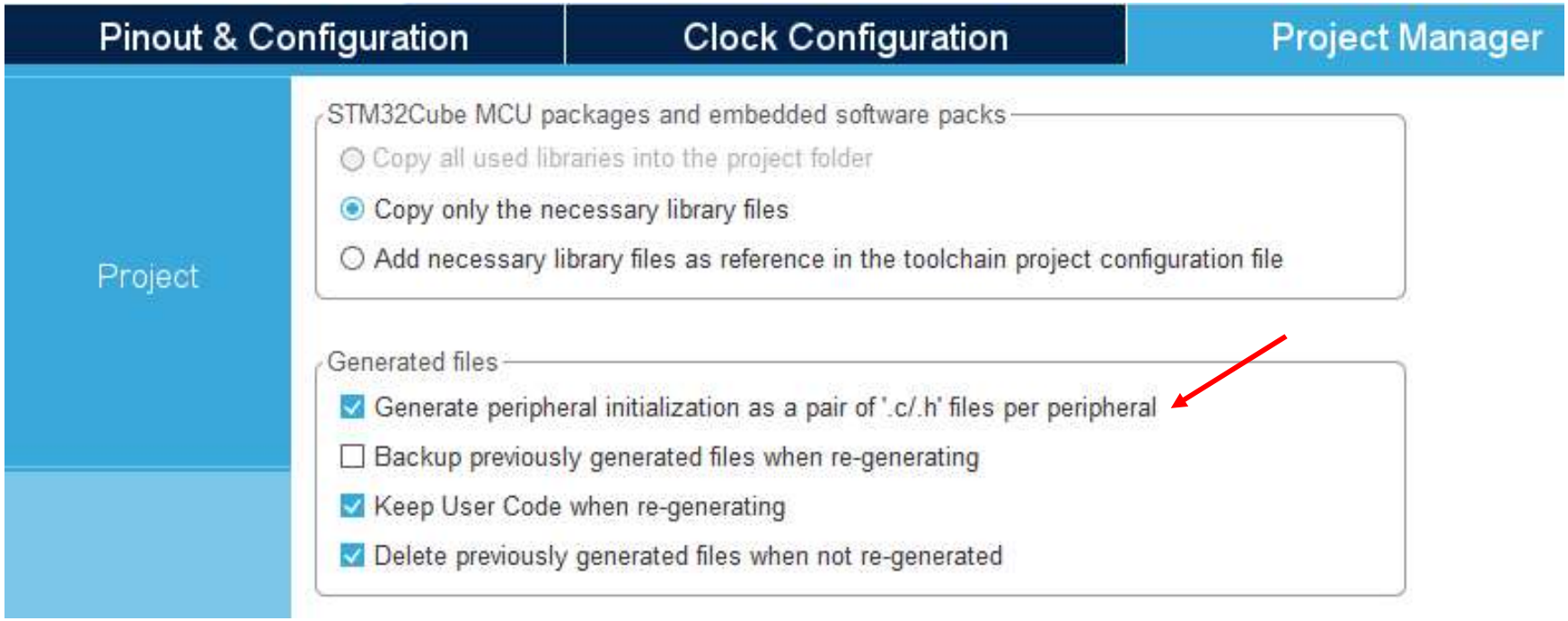
PD9: DBG_USART3_RX
PD8: DBG_USART3_TX

TCK (Test Clock)
TMS (Test Mode Select)
SWO (Single/Serial Wire Output)

Liaw Hwee Choo, June 2022.

STM32CubeIDE – DAC ADC

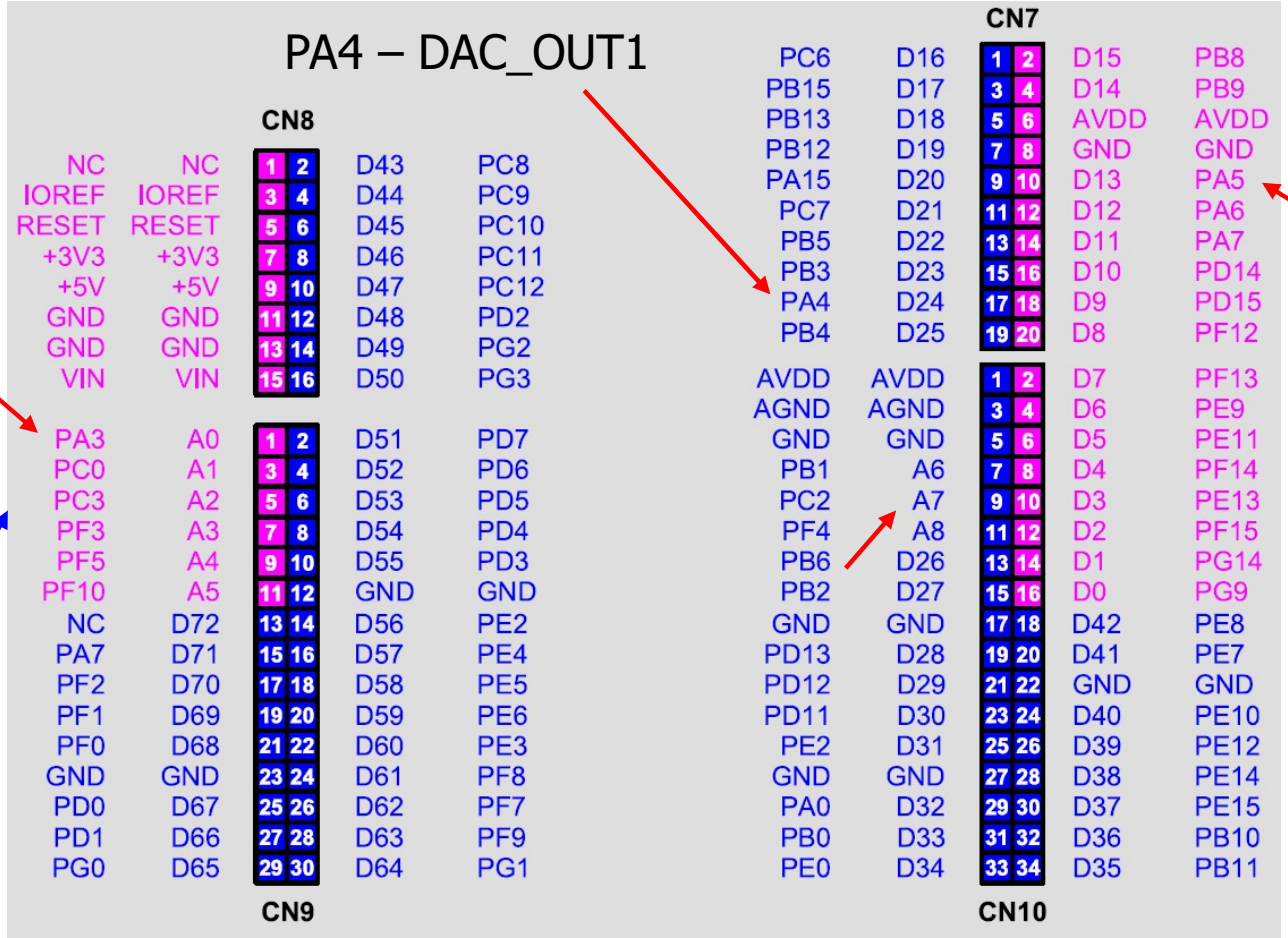
Enable Interrupt for EXTI line[15:10] for USER_Btn [B1], and set Project Manager



Liaw Hwee Choo, June 2022.

STM32CubeIDE – DAC ADC

Nucleo-F767ZI Board Pinout – DAC and ADC Pins on ST Zio



PA5 – DAC_OUT2

PA3 (A0) – ADC1_IN3, ADC2_IN3, ADC3_IN3
PC0 (A1) – ADC1_IN10, ADC2_IN10, ADC3_IN10
PC3 (A2) – ADC1_IN13, ADC2_IN13, ADC3_IN13
PF3 (A3) – ADC3_IN9
PF5 (A4) – ADC3_IN15
PF10 (A5) – ADC3_IN8

PB1 (A6) – ADC1_IN9, ADC2_IN9
PC2 (A7) – ADC1_IN12, ADC2_IN12, ADC3_IN12
PF4 (A8) – ADC3_IN14

Liaw Hwee Choo, June 2022.

STM32CubeIDE – DAC ADC

Set up one DAC channel (OUT1, pin PA4, default name: DAC_OUT1)

Pinout & Configuration | Clock Configuration | Project Manager | Tools

Software Packs | Pinout

Categories | A->Z

System Core

CORTEX_M7
DMA
GPIO
IWDG
NVIC
RCC
SYS
WWDG

Analog

ADC1
ADC2
ADC3
DAC

DAC Mode and Configuration

Mode

OUT1 Configuration
OUT2 Configuration
External Trigger

Configuration

Reset Configuration

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

Configure the below parameters :

Search (Ctrl+F)

DAC Out1 Settings

Output Buffer	Disable
Trigger	None

VDD | PA4 | PA5

DAC_OUT1

Check 5

1

2

3

Check 4

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

Pin Name	Signal on Pin	GPIO outp...	GPIO mode	GPIO Pull-up/Pull-down	Maxi...	Fast ...	User ...	Modif...
PA4	DAC_OUT1	n/a	Analog mode	No pull-up and no pull-down	n/a	n/a		☐

Liaw Hwee Choo, June 2022.

STM32CubeIDE – DAC ADC

Set up one DAC channel – Check GPIO (pin PA4, default name: DAC_OUT1)

System Core

CORTEX_M7
DMA
GPIO
IWDG
NVIC
RCC
SYS
WWDG

Analog

ADC1
ADC2
ADC3
DAC

Timers

Connectivity

CAN1
CAN2
CAN3

Configuration

Group By Peripherals

GPIO | DAC | RCC | SYS | USART | NVIC

Search Signals

Search (Crtl+F)

Show only Modified Pins

Pin Name	Signal on Pin	GPIO output ...	GPIO mode	GPIO Pull-up...	Maximum ou...	Fast Mode	User Label	Modified
PA4	DAC_OUT1	n/a	Analog mode	No pull-up an...	n/a	n/a		☐

PA4 Configuration :

GPIO mode: Analog mode

GPIO Pull-up/Pull-down: No pull-up and no pull-down

User Label:

Liaw Hwee Choo, June 2022.

Set up two ADC channels (IN3, pin PA3, name: ADC1_IN3, and Temperature Sensor)

Pinout & Configuration | Clock Configuration | Project Manag

Software Packs | Pinout

Categories | A->Z

System Core
- CORTEX_M7
- DMA
- GPIO
- IWDG
- NVIC
- RCC
- SYS
- WWDG

Analog
- ADC1
- ADC2
- ADC3
- DAC

Timers

Connectivity
- CAN1
- CAN2
- CAN3
- ETH

ADC1 Mode and Configuration

Mode
- IN3
- IN4
- IN5
- IN6
- IN7
- IN8
- IN9
- IN10
- IN11
- IN12
- IN13
- IN14
- IN15
- Temperature Sensor Channel

Configuration

Reset Configuration

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

Pin Na...	Signal on ...	GPIO out...	GPIO mode	GPIO Pull-...
PA3	ADC1_IN3	n/a	Analog mo...	No pull-up ...

PC3 | VDD | VSSA | VREF+ | VDDA | PA0/.. | PA1 | PA2

PA3 | VSS | VDD | PA4 | PA5

ADC1_IN3

DAC_OUT1

1 2 2 3 4

Liaw Hwee Choo, June 2022.

STM32CubeIDE – DAC ADC

ADC channel – GPIO Settings: pin PA3, name: ADC1_IN3

System Core
CORTEX_M7
DMA
GPIO
IWDG
NVIC
RCC
SYS
WWDG

Analog
ADC1
ADC2
ADC3
DAC

Timers

Connectivity
CAN1
CAN2
CAN3
ETH

Configuration

Group By Peripherals

GPIO | ADC | DAC | RCC | SYS | USART | NVIC

Search Signals

Search (Crtl+F)

Show only Modified Pins

Pin Name	Signal on Pin	GPIO out...	GPIO mode	GPIO Pull-...	Maximu...	Fast Mode	User La...	Modified
PA3	ADC1_IN3	n/a	Analog mode	No pull-up ...	n/a	n/a		☐

PA3 Configuration :

GPIO mode: Analog mode

GPIO Pull-up/Pull-down: No pull-up and no pull-down

User Label:

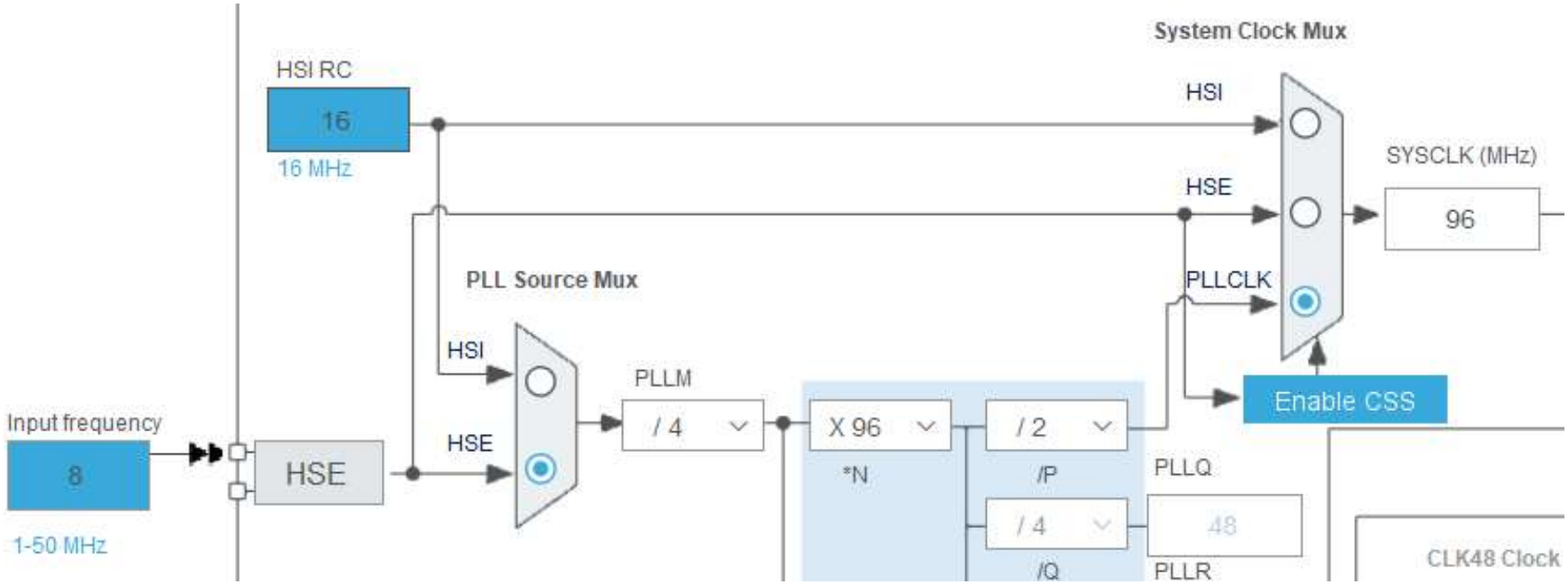
PC3
VDD
VSSA
VREF+
VDDA
PA0/..
PA1
PA2
PA3 (ADC1_IN3)
VSS
VDD
PA4 (DAC_OUT1)
PA5

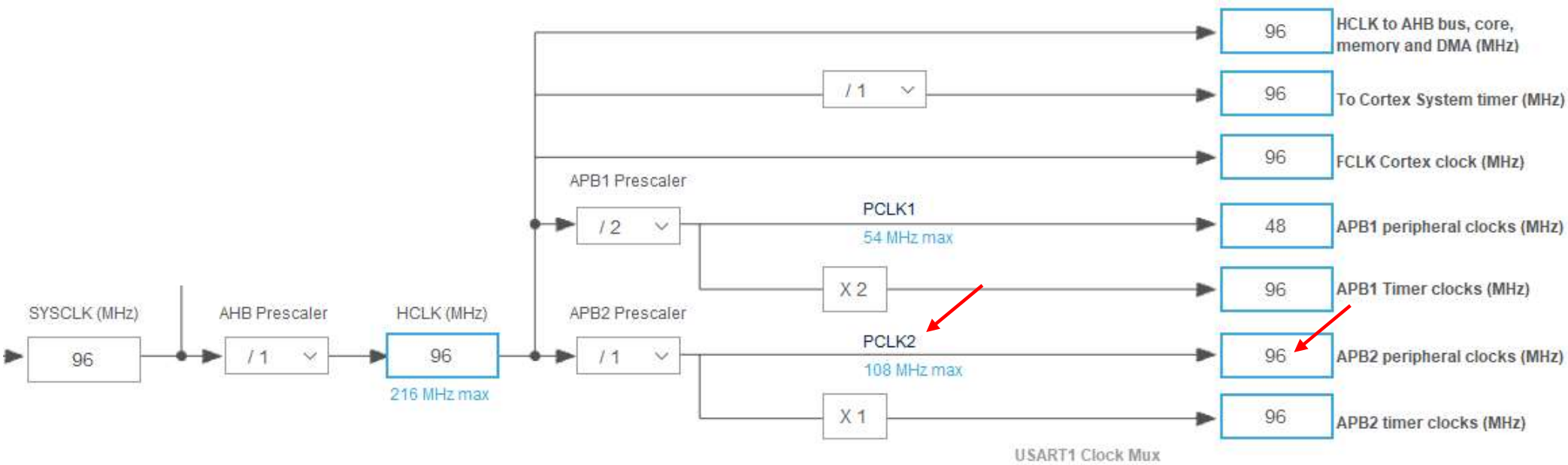
1 2 3

Liaw Hwee Choo, June 2022.

STM32CubeIDE – DAC ADC

Clock Configuration: Use default clock settings





Liaw Hwee Choo, June 2022.

DMA Settings

STM32CubeIDE – DAC ADC

Pinout & Configuration | Clock Configuration | Project Manager

Software Packs | Pinout

Categories | A->Z

System Core
Analog
ADC1
ADC2
ADC3
DAC
Timers
Connectivity
Multimedia
Security
Computing
Middleware

ADC1 Mode and Configuration

Mode

IN0
IN1
IN2
IN3
IN4
IN5

Configuration

Reset Configuration

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

DMA Request	Stream	Direction	Priority
ADC1	DMA2 Stream 0	Peripheral To Memory	Low

Add | Delete

DMA Request Settings

		Peripheral	Memory
Mode	Circular	Increment Address	☐	☑
Use Fifo ☐ Threshold		Data Width	Word	Word
		Burst Size		

Pinout view

NRST, PC0, PC1, PC2, PC3, VDD, VSSA, VREF+, VDDA, PA0/.., PA1, PA2, PA3, VSS, VDD, PA4

ADC1_IN3, DAC_OUT1

ADC Parameter Settings STM32CubeIDE – DAC ADC

Pinout & Configuration | Clock Configuration | Project Mana

Software Packs | Pinout

Categories | A->Z

System Core
Analog
ADC1
ADC2
ADC3
DAC
Timers
Connectivity
Multimedia
Security
Computing
Middleware

ADC1 Mode and Configuration

Mode

IN3

Configuration

Reset Configuration

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

Search (Crtl+F)

ADCs_Common_Settings	
Mode	Independent mode
ADC_Settings	
Clock Prescaler	PCLK2 divided by 4
Resolution	12 bits (15 ADC Clock cycles)
Data Alignment	Right alignment
Scan Conversion Mode	Enabled
Continuous Conversion Mode	Disabled
Discontinuous Conversion Mode	Disabled
DMA Continuous Requests	Enabled
End Of Conversion Selection	EOC flag at the end of single channel conversion
ADC_Regular_ConversionMode	
Number Of Conversion	2
External Trigger Conversion Source	Regular Conversion launched by software
External Trigger Conversion Edge	None
Rank	1
Channel	Channel 3
Sampling Time	480 Cycles
Rank	2
Channel	Channel Temperature Sensor
Sampling Time	3 Cycles
ADC_Injected_ConversionMode	
Number Of Conversions	0

3 cycles may not be enough, you may test it and increase it to reasonable cycles

Enable NVIC Interrupt for ADC

Pinout & Configuration | Clock Configuration | Project Manager

Software Packs | Pinout

Categories | A->Z

- System Core
- Analog
  - ADC1
  - ADC2
  - ADC3
  - DAC
- Timers
- Connectivity
- Multimedia
- Security
- Computing
- Middleware

ADC1 Mode and Configuration

Mode

- [x] IN3
- [ ] IN4
- [ ] IN5
- [ ] IN6
- [ ] IN7
- [ ] IN8
- [ ] IN9
- [ ] IN10
- [ ] IN11
- [ ] IN12
- [ ] IN13
- [ ] IN14
- [ ] IN15
- [x] Temperature Sensor Channel

Configuration

Reset Configuration

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

NVIC Interrupt Table	Enabled	Preemption Priority	Sub Prio...
ADC1, ADC2 and ADC3 global interrupts	[x]	0	0
DMA2 stream0 global interrupt	[x]	0	0

NRST, PC0, PC1, PC2, PC3, VDD, VSSA, VREF+, VDDA, PA0/.., PA1, PA2, PA3, VSS

ADC1_IN3

1

2

Set automatically by DMA settings

Need to set this value higher than 0

STM32CubeIDE – DAC ADC

NVIC Interrupt Table Settings

Pinout & Configuration | Clock Configuration | Project Manager

Software Packs | Pinout

Categories | A->Z

System Core
- CORTEX_M7
- DMA
- GPIO
- IWDG
- NVIC
- RCC
- SYS
- WWDG

Analog
- ADC1
- ADC2
- ADC3
- DAC

Timers

Connectivity

Multimedia

NVIC Mode and Configuration

Configuration

NVIC | Code generation

Priority Group: 4 bits for pre-emption priority 0 bits for subpriority

☐ Sort by Premption Priority and Sub Priority ☐ Sort by interrupts names

Search: Search (Crtl+F) Show: available interrupts ☑ Force DMA channels Interrupts

NVIC Interrupt Table	Enabled	Preemption Priority	Sub Priority
Non maskable interrupt	☑	0	0
Hard fault interrupt	☑	0	0
Memory management fault	☑	0	0
Pre-fetch fault, memory access fault	☑	0	0
Undefined instruction or illegal state	☑	0	0
System service call via SWI instruction	☑	0	0
Debug monitor	☑	0	0
Pendable request for system service	☑	0	0
Time base: System tick timer	☑	0	0
PVD interrupt through EXTI line 16	☐	0	0
Flash global interrupt	☐	0	0
RCC global interrupt	☐	0	0
ADC1, ADC2 and ADC3 global interrupts	☑	1	0
USART3 global interrupt	☐	0	0
EXTI line[15:10] interrupts	☑	0	0
TIM6 global interrupt, DAC1 and DAC2 underrun error interrupts	☐	0	0
DMA2 stream0 global interrupt	☑	0	0
FPU global interrupt	☐	0	0

Set

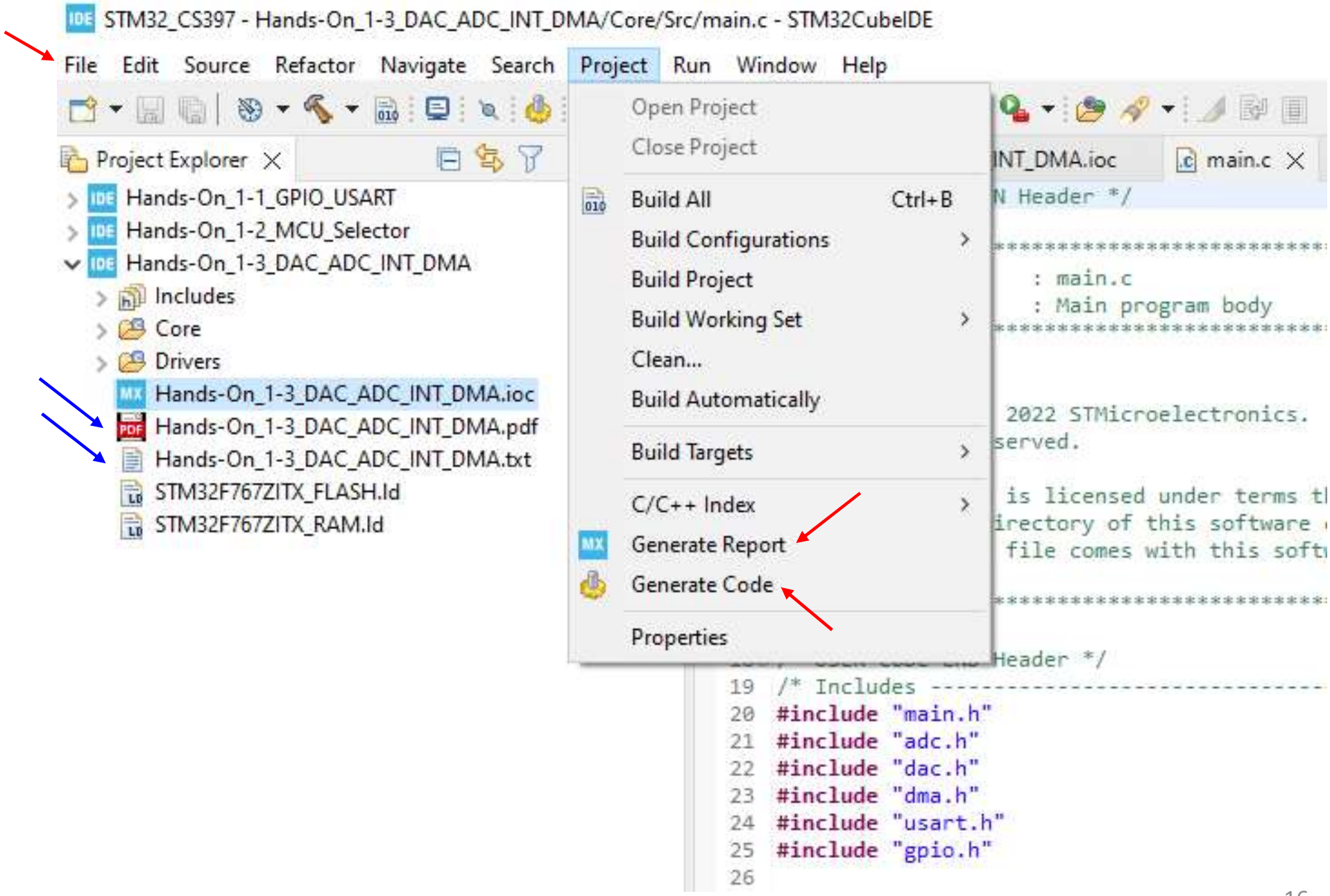
Set automatically by DMA settings

Set ADC interrupt priority lower than the SysTick timer in case **HAL_Delay()** is needed in ADC interrupt routine.

Liaw Hwee Choo, June 2022.

STM32CubeIDE – DAC ADC

Save Project (File -> Save All), Generate Code, and Generate Report



Liaw Hwee Choo, June 2022.

STM32CubeIDE – DAC ADC

Build

STM32_CS397 - Hands-On_1-3_DAC_ADC_INT_DMA/Core/Src/main.c - STM32CubeIDE

File Edit Source Refactor Navigate Search Project Run Window Help

Project Explorer

- Hands-On_1-3_DAC_ADC_INT_DMA
  - Binaries
  - Includes
  - Core
    - Inc
      - adc.h
      - dac.h
      - dma.h
      - gpio.h
      - main.h
      - stm32f7xx_hal_conf.h
      - stm32f7xx_it.h
      - usart.h
    - Src
      - adc.c
      - dac.c
      - dma.c
      - gpio.c
      - main.c
      - stm32f7xx_hal_msp.c
      - stm32f7xx_it.c
      - syscalls.c
      - sysmem.c
      - system_stm32f7xx.c
      - usart.c
    - Startup
  - Drivers
  - Debug
  - Hands-On_1-3_DAC_ADC_INT_DMA.ioc
  - Hands-On_1-3_DAC_ADC_INT_DMA.pdf
  - Hands-On_1-3_DAC_ADC_INT_DMA.txt
  - STM32F767ZITX_FLASH.ld
  - STM32F767ZITX_RAM.ld

Check the created files.

Hands-On_1-3_DAC_ADC_INT_DMA.ioc | main.c

```
1  /* USER CODE BEGIN Header */
2  /**
3    ******************************************************************************
4    * @file           : main.c
5    * @brief          : Main program body
6    ******************************************************************************
7    * @attention
8    *
9    * Copyright (c) 2022 STMicroelectronics.
10   * All rights reserved.
11   *
12   * This software is licensed under terms that can be found in the LICENSE file
13   * in the root directory of this software component.
14   * If no LICENSE file comes with this software, it is provided AS-IS.
15   *
16   ******************************************************************************
17   */
18 /* USER CODE END Header */
19 /* Includes ------------------------------------------------------------------*/
20 #include "main.h"
21 #include "adc.h"
22 #include "dac.h"
23 #include "dma.h"
24 #include "usart.h"
25 #include "gpio.h"
```

Problems | Tasks | Console | Properties | Terminal

CDT Build Console [Hands-On_1-3_DAC_ADC_INT_DMA]

```
arm-none-eabi-objdump -h -S  Hands-On_1-3_DAC_ADC_INT_DMA.elf  > "Hands-On_1-3_DAC_ADC_INT_DMA.list"
   text    data     bss     dec     hex filename
  15760      20    1892   17672    4508 Hands-On_1-3_DAC_ADC_INT_DMA.elf
Finished building: default.size.stdout

Finished building: Hands-On_1-3_DAC_ADC_INT_DMA.list


21:12:39 Build Finished. 0 errors, 0 warnings. (took 5s.424ms)
```

Hands-On_1-3_DAC_ADC_INT_DMA

Liaw Hwee Choo, June 2022.

STM32CubeIDE – DAC ADC

Add code to **main.c**,

```c
/* main.c */
/* Includes */
#include "main.h"
#include "adc.h"
#include "dac.h"
#include "dma.h"
#include "usart.h"
#include "gpio.h"
/* Private includes */
/* USER CODE BEGIN Includes */
#include <stdio.h>
/* USER CODE END Includes */

/* Private variables */
/* USER CODE BEGIN PV */
uint32_t adc[2], buffer[2];
uint32_t dac = 0;

float vsense = 3.3 / 4096.0;
float temperature = 0.0;
/* USER CODE END PV */

/* Private function prototypes */
void SystemClock_Config(void);
```

```c
int main(void)
{
  /* MCU Configuration */
  /* Reset of all peripherals, Initializes ... */
  HAL_Init();
  /* Configure the system clock */
  SystemClock_Config();
  /* Initialize all configured peripherals */
  MX_GPIO_Init();
  MX_USART3_UART_Init();
  MX_DMA_Init();
  MX_ADC1_Init();
  MX_DAC_Init();

  /* USER CODE BEGIN 2 */
  HAL_DAC_Start(&hdac, DAC_CHANNEL_1);
  HAL_DAC_SetValue(&hdac, DAC_CHANNEL_1, DAC_ALIGN_12B_R, dac);

  HAL_ADC_Start_DMA(&hadc1, buffer, 2);
  HAL_ADC_Start_IT(&hadc1);
  /* USER CODE END 2 */

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

STM32CubeIDE – DAC ADC

Add code to **main.c**, USER CODE WHILE

```
/* Infinite loop */
/* USER CODE BEGIN WHILE */
while (1)
{
    HAL_GPIO_TogglePin(GPIOB, LD3_Pin);

    //  project -> properties -> C/C++ Build - Settings -> Tool Settings -> MCU GCC Linker -
    //  - Miscellaneous -> Other Flags: -u _printf_float

    printf("DAC ADC vales = %ld  %ld  %ld  %4.2f deg.C \r\n", dac, adc[0], adc[1], temperature);

    ++dac;
    if( dac > 4095 ) dac = 0;
    HAL_DAC_SetValue(&hdac, DAC_CHANNEL_1, DAC_ALIGN_12B_R, dac);
    HAL_Delay(500);

    HAL_ADC_Start_IT(&hadc1);
    HAL_Delay(500);

    /* USER CODE END WHILE */

    /* USER CODE BEGIN 3 */
}
/* USER CODE END 3 */
```


Liaw Hwee Choo, June 2022.

Add code to **main.c**, USER CODE 4

```
/* USER CODE BEGIN 4 */
void HAL_ADC_ConvCpltCallback(ADC_HandleTypeDef* hadc)
{
    if( hadc->Instance == ADC1 )
    {
        for (int i = 0; i < 2; i++)
        {
            adc[i] = buffer[i];
        }
        temperature = (((adc[1] * vsense ) - 0.76 ) / 0.0025) + 25.0;
    }
}

void HAL_GPIO_EXTI_Callback(uint16_t GPIO_Pin)
{
    if(GPIO_Pin == GPIO_PIN_13)
    {
        HAL_GPIO_TogglePin(GPIOB, LD2_Pin);
    }
}

int __io_putchar(int ch)
{
    uint8_t c[1];
    c[0] = ch & 0x00FF;
    HAL_UART_Transmit(&huart3, &*c, 1, 10);
    return ch;
}
/* USER CODE END 4 */
```

```
/*
int _write(int file, char *ptr, int len)
{
    int DataIdx;
    for(DataIdx= 0; DataIdx < len; DataIdx++)
    {
        __io_putchar(*ptr++);
    }
    return len;
}
*/
```

STM32CubeIDE – DAC ADC

Build Project and Select "Debug As" -> "1 STM32 Cortex-M C/C++ Application"

STM32_CS397 - Hands-On_1-3_DAC_ADC_INT_DMA/Core/Src/main.c - STM32CubeIDE

File Edit Source Refactor Navigate Search Project Run Window Help

Project Explorer

- Hands-On_1-1_GPIO_USART
- Hands-On_1-2_MCU_Selector
- Hands-On_1-3_DAC_ADC_INT_DMA
  - Binaries
  - Includes
  - Core
    - Inc
    - Src
    - Startup
  - Drivers
  - Debug
  - Hands-On_1-3_
  - Hands-On_1-3_
  - Hands-On_1-3_
  - STM32F767ZITX
  - STM32F767ZITX

New
Go Into
Open in New Window
Show In Alt+Shift+W
Show in Local Terminal
Copy Ctrl+C
Paste Ctrl+V
Delete Delete
Source
Move...
Rename... F2
Import...
Export...
Build Project
Clean Project
Refresh F5
Close Project
Close Unrelated Projects

1 Hands-On_1-1_GPIO_USART
Debug As
Debug Configurations...
Organize Favorites...

1 STM32 Cortex-M C/C++ Application

```
/* Infinite loop */
/* USER CODE BEGIN WHILE */
while (1)
{
    HAL_GPIO_TogglePin(GPIOB, LD3_Pin);

    //  project -> properties -> C/C++ Build - Settings -> Tool Settings
    //  - Miscellaneous -> Other Flags: -u _printf_float

    printf("DAC ADC vales = %ld  %ld  %ld  %4.2f deg.C \n\r", dac, adc[0]

    ++dac;
    if( dac > 4095 ) dac = 0;
    HAL_DAC_SetValue(&hdac, DAC_CHANNEL_1, DAC_ALIGN_12B_R, dac);
    HAL_Delay(500);

    HAL_ADC_Start_IT(&hadc1);
    HAL_Delay(500);

  /* USER CODE END WHILE */

  /* USER CODE BEGIN 3 */
}
/* USER CODE END 3 */
```


Liaw Hwee Choo, June 2022.

STM32CubeIDE – DAC ADC

Edit/Debug Configuration, Select "Apply" then "OK" or "Debug" to download program

Edit Configuration

Edit launch configuration properties

Name: Hands-On_1-3_DAC_ADC_INT_DMA

Main | Debugger | Startup | Source | Common

Project:

Hands-On_1-3_DAC_ADC_INT_DMA | Browse...

C/C++ Application:

Debug/Hands-On_1-3_DAC_ADC_INT_DMA.elf | Search Project... | Browse...

Build (if required) before launching

Build Configuration: Select Automatically

Enable auto build | Disable auto build

Use workspace settings | Configure Workspace Settings...

Revert | Apply

OK | Cancel

Edit Configuration

Edit launch configuration properties

Name: Hands-On_1-3_DAC_ADC_INT_DMA

Main | Debugger | Startup | Source | Common

GDB Connection Settings

Autostart local GDB server | Host name or IP address: localhost

Connect to remote GDB server | Port number: 2331

Debug probe: SEGGER J-LINK

GDB Server Command Line Options

Show Command Line

Interface

SWD | JTAG | Initial Speed: 12000 kHz | Use specific J-Link

Device: STM32F767ZI

Revert | Apply

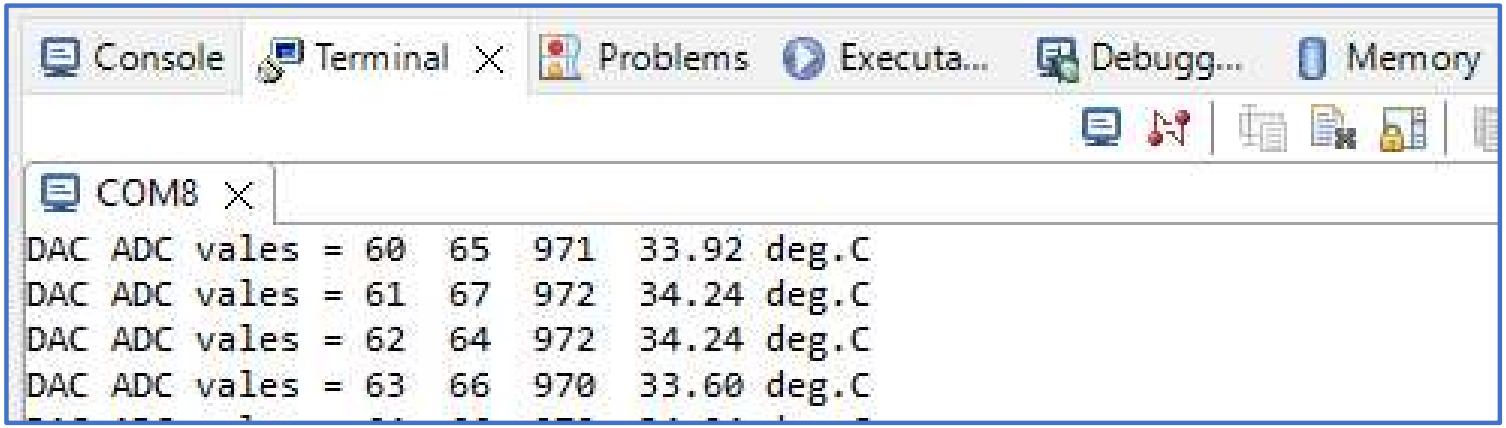
OK | Cancel

Debug

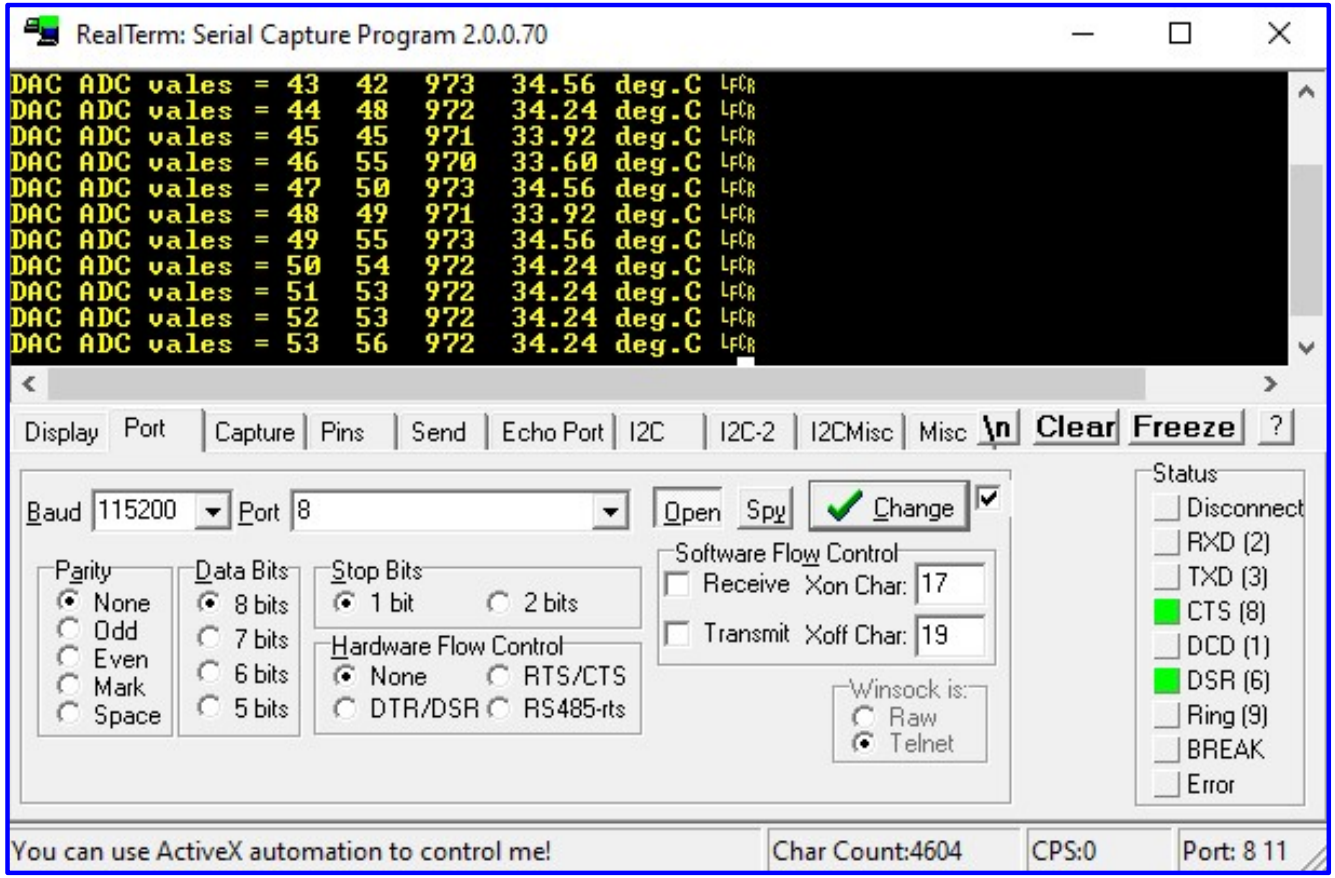
Liaw Hwee Choo, June 2022.

STM32CubeIDE – DAC ADC

Test the developed program with TM Terminal or RealTerm



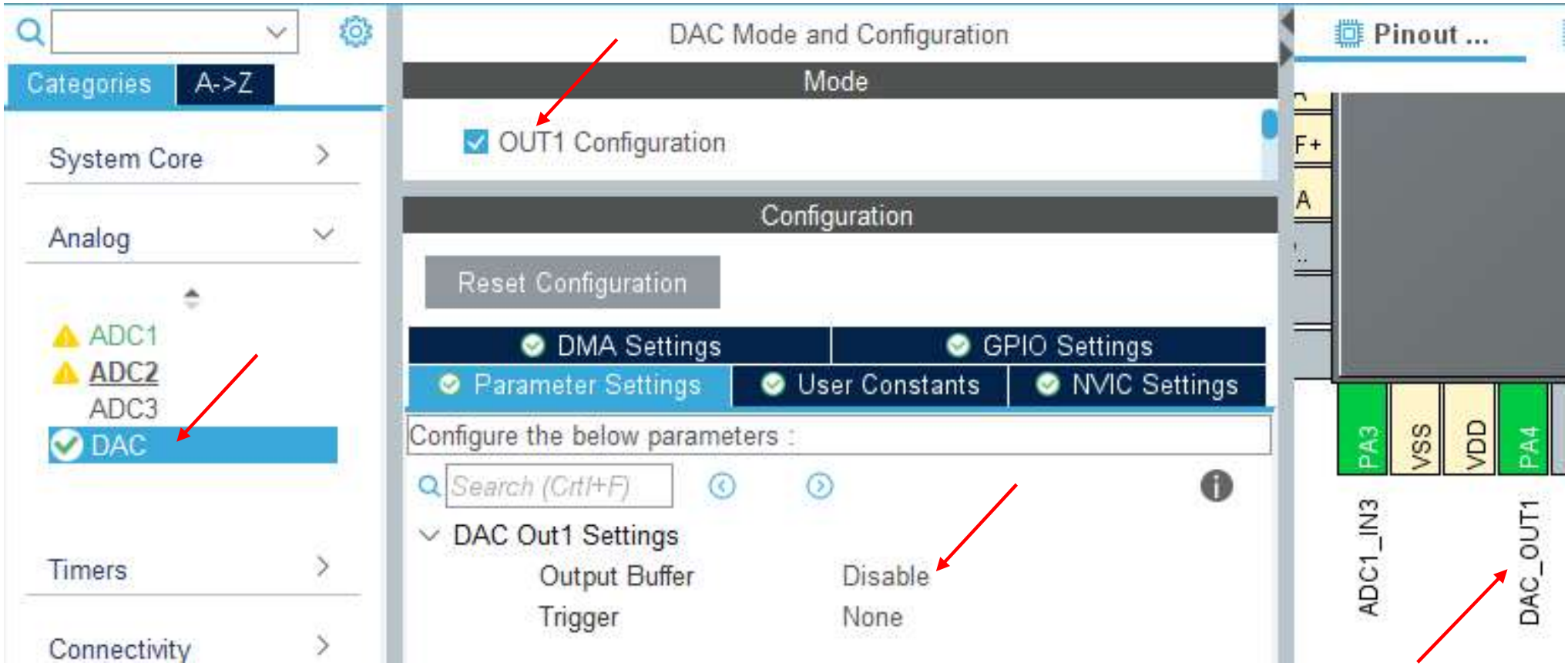
Study the results and understand the program.



Liaw Hwee Choo, June 2022.

DAC Output Buffer

- The DAC has an output buffer that can be either enabled or disabled.
- With a disabled buffer, it has about 1.5 MΩ output impedance and very little offset.
- With an enabled buffer, it has 15 kΩ output impedance and may have up to 200 mV offset.



Liaw Hwee Choo, June 2022.

DAC Output Buffer

```
/* Code generated automatically by STM32CubeMX */

/* dac.c */

/* Includes */
#include "dac.h"

DAC_HandleTypeDef hdac;

/* DAC init function */
void MX_DAC_Init(void)
{
    DAC_ChannelConfTypeDef sConfig = {0};

    /* DAC Initialization */
    hdac.Instance = DAC;
    if (HAL_DAC_Init(&hdac) != HAL_OK)
    {
        Error_Handler();
    }
    /* DAC channel OUT1 config */
    sConfig.DAC_Trigger = DAC_TRIGGER_NONE;
    sConfig.DAC_OutputBuffer = DAC_OUTPUTBUFFER_DISABLE;
    if (HAL_DAC_ConfigChannel(&hdac, &sConfig, DAC_CHANNEL_1) != HAL_OK)
    {
        Error_Handler();
    }
}
```

HAL_DAC_SetValue()

STM32CubeIDE – DAC ADC

Hands-On_1-3_DAC_AD... | main.c | stm32f7xx_it.c | stm32f7xx_hal.c | stm32f7xx_hal_adc.c

```c
91   /* USER CODE END SysInit */
92
93   /* Initialize all configured peripherals */
94   MX_GPIO_Init();
95   MX_DMA_Init();
96   MX_USART3_UART_Init();
97   MX_ADC1_Init();
98   MX_DAC_Init();
99   /* USER CODE BEGIN 2 */
100  HAL_DAC_Start(&hdac, DAC_CHANNEL_1);
101  HAL_DAC_SetValue(&hdac, DAC_CHANNEL_1, DAC_ALIGN_12B_R, dac);
```

Hover mouse pointer over the function to understand it.

```c
/**
  * @brief  Set the specified data holding register value for DAC channel.
  * @param  hdac pointer to a DAC_HandleTypeDef structure that contains
  *         the configuration information for the specified DAC.
  * @param  Channel The selected DAC channel.
  *          This parameter can be one of the following values:
  *            @arg DAC_CHANNEL_1: DAC Channel1 selected
  *            @arg DAC_CHANNEL_2: DAC Channel2 selected
  * @param  Alignment Specifies the data alignment.
  *          This parameter can be one of the following values:
  *            @arg DAC_ALIGN_8B_R: 8bit right data alignment selected
  *            @arg DAC_ALIGN_12B_L: 12bit left data alignment selected
  *            @arg DAC_ALIGN_12B_R: 12bit right data alignment selected
  * @param  Data Data to be loaded in the selected data holding register.
  * @retval HAL status
  */
HAL_StatusTypeDef HAL_DAC_SetValue(DAC_HandleTypeDef *hdac, uint32_t Channel, uint32_t Alignment, uint32_t Data)
{
  __IO uint32_t tmp = 0UL;

  /* Check the parameters */
  assert_param(IS_DAC_CHANNEL(Channel));
  assert_param(IS_DAC_ALIGN(Alignment));
  assert_param(IS_DAC_DATA(Data));
```

```
/* stm32f7xx_hal_adc.c */
*** Execution of ADC conversions ***
==============================================================================
[..]
(#) ADC driver can be used among three modes: polling, interruption, transfer by DMA.

   *** Polling mode IO operation ***
   =================================
   [..]
     (+) Start the ADC peripheral using HAL_ADC_Start()
     (+) Wait for end of conversion using HAL_ADC_PollForConversion(), at this stage
         user can specify the value of timeout according to his end application
     (+) To read the ADC converted values, use the HAL_ADC_GetValue() function.
     (+) Stop the ADC peripheral using HAL_ADC_Stop()

   *** Interrupt mode IO operation ***
   ===================================
   [..]
     (+) Start the ADC peripheral using HAL_ADC_Start_IT()
     (+) Use HAL_ADC_IRQHandler() called under ADC_IRQHandler() Interrupt subroutine
     (+) At ADC end of conversion HAL_ADC_ConvCpltCallback() function is executed, and
         user can add his own code by customization of function pointer
         HAL_ADC_ConvCpltCallback
     (+) In case of ADC Error, HAL_ADC_ErrorCallback() function is executed, and user
         can add his own code by customization of function pointer
         HAL_ADC_ErrorCallback
     (+) Stop the ADC peripheral using HAL_ADC_Stop_IT()
```

STM32CubeIDE – DAC ADC

```
/* stm32f7xx_hal_adc.c*/
*** Execution of ADC conversions ***
==============================================================================
[..]
(#) ADC driver can be used among three modes: polling, interruption, transfer by DMA.


   *** DMA mode IO operation ***
   ==============================
   [..]
     (+) Start the ADC peripheral using HAL_ADC_Start_DMA(), at this stage the user
         specify the length of data to be transferred at each end of conversion
     (+) At The end of data transfer by HAL_ADC_ConvCpltCallback() function is
         executed, and user can add his own code by customization of function pointer
         HAL_ADC_ConvCpltCallback
     (+) In case of transfer Error, HAL_ADC_ErrorCallback() function is executed, and
         user can add his own code by customization of function pointer
         HAL_ADC_ErrorCallback
     (+) Stop the ADC peripheral using HAL_ADC_Stop_DMA()
```

Polling mode

STM32CubeIDE – DAC ADC

Two ADC channel Settings in Scan and Continuous Conversion Modes

Categories | A->Z

System Core
- CORTEX_M7
- DMA
- GPIO
- IWDG
- NVIC
- RCC
- SYS
- WWDG

Analog
- ADC1
- ADC2
- ADC3
- DAC

Timers

Connectivity
- CAN1
- CAN2
- CAN3
- ETH

ADC1 Mode and Configuration

Mode

IN3

Configuration

Reset Configuration

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

Search (Crtl+F)

ADCs_Common_Settings	
Mode	Independent mode
ADC_Settings	
Clock Prescaler	PCLK2 divided by 4
Resolution	12 bits (15 ADC Clock cycles)
Data Alignment	Right alignment
Scan Conversion Mode	Enabled
* Continuous Conversion Mode	Enabled
Discontinuous Conversion Mode	Disabled
DMA Continuous Requests	Disabled
End Of Conversion Selection	EOC flag at the end of single channel conversion
ADC_Regular_ConversionMode	
Number Of Conversion	2
External Trigger Conversion Source	Regular Conversion launched by software
External Trigger Conversion Edge	None
Rank	1
Channel	Channel 3
Sampling Time	480 Cycles
Rank	2
Channel	Channel Temperature Sensor
Sampling Time	480 Cycles

Pinout view

PC1, PC2, PC3, VDD, VSSA, VREF+, VDDA, PA0/.., PA1, PA2, PA3, VSS, VDD, PA4

ADC1_IN3, DAC_OUT1

Liaw Hwee Choo, June 2022.

Polling mode

STM32CubeIDE – DAC ADC

Add code to **main.c**, USER CODE Includes, USER CODE PV, USER CODE 2

```c
/* main.c */
/* Includes */
#include "main.h"
#include "adc.h"
#include "dac.h"
#include "usart.h"
#include "gpio.h"

/* Private includes */
/* USER CODE BEGIN Includes */
#include <stdio.h>
/* USER CODE END Includes */

/* Private variables */
/* USER CODE BEGIN PV */
uint32_t adc[2];
uint32_t dac = 0;
float vsense = 3.3 / 4096.0;
float temperature = 0.0;
/* USER CODE END PV */

/* Private function prototypes */
void SystemClock_Config(void);
```

```c
int main(void)
{
  /* MCU Configuration */
  /* Reset of all peripherals, Initializes ... */
  HAL_Init();

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

  /* Initialize all configured peripherals */
  MX_GPIO_Init();
  MX_USART3_UART_Init();
  MX_ADC1_Init();
  MX_DAC_Init();

  /* USER CODE BEGIN 2 */
  HAL_DAC_Start(&hdac, DAC_CHANNEL_1);
  HAL_DAC_SetValue(&hdac, DAC_CHANNEL_1, DAC_ALIGN_12B_R, dac);
  /* USER CODE END 2 */

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

Polling mode

STM32CubeIDE – DAC ADC

Add code to **main.c**, USER CODE WHILE

```
/* USER CODE BEGIN WHILE */
 while (1)
 {
     HAL_GPIO_TogglePin(GPIOB, LD3_Pin);
     HAL_DAC_SetValue(&hdac, DAC_CHANNEL_1, DAC_ALIGN_12B_R, dac);

     ++dac;
     if(dac > 4095) dac = 0;

     HAL_ADC_Start(&hadc1);

     HAL_ADC_PollForConversion(&hadc1,100);
     adc[0] = HAL_ADC_GetValue(&hadc1);

     HAL_ADC_PollForConversion(&hadc1,100);
     adc[1] = HAL_ADC_GetValue(&hadc1);

     HAL_ADC_Stop(&hadc1);

     temperature = (( adc[1] * vsense - 0.76 ) / 0.0025) + 25.0;

     //  project -> properties -> C/C++ Build - Settings -> Tool Settings -> MCU GCC Linker -
     //  - Miscellaneous -> Other Flags: -u _printf_float
     printf("DAC ADC vales = %ld  %ld  %ld  %4.2f degC \n\r", dac, adc[0], adc[1], temperature);

     HAL_Delay(1000);
   /* USER CODE END WHILE */
```

Interrupt mode

Enable NVIC Interrupt for ADC

*Hands-On_03_DAC_ADC_INT.ioc

Pinout & Configuration | Clock Configuration | Project Manager

Software Packs | Pinout

Categories | A->Z

CORTEX_M7
DMA
GPIO
IWDG
NVIC
RCC
SYS
WWDG

Analog
ADC1
ADC2
ADC3
DAC

Timers

Connectivity
CAN1
CAN2
CAN3
ETH
FMC
I2C1
I2C2

ADC1 Mode and Configuration

Mode

IN3
IN4
IN5
IN6
IN7
IN8
IN9
IN10
IN11
IN12
IN13
IN14
IN15
Temperature Sensor Channel

(one channel only)

Configuration

Reset Configuration

1

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

NVIC Interrupt Table	Enabled	Preemption Priority	Sub ...
ADC1, ADC2 and ADC3 global interrupts	☑	0	0

2

Need to set this value higher than 0

Pinout view

PC2
PC3
VDD
VSSA
VREF+
VDDA
PA0/..
PA1
PA2
PA3 (ADC1_IN3)
VSS
VDD
PA4 (DAC_OUT1)
PA5

Interrupt mode STM32CubeIDE – DAC ADC

Set Pre-emption Priority for ADC Interrupt

*Hands-On_03_DAC_ADC_INT.ioc

Pinout & Configuration | Clock Configuration | Project Manager | Tools

Software Packs | Pinout

Categories | A->Z

System Core
- CORTEX_M7
- DMA
- GPIO
- IWDG
- NVIC
- RCC
- SYS
- WWDG

Analog
- ADC1
- ADC2
- ADC3
- DAC

Timers

Connectivity

NVIC Mode and Configuration

Configuration

NVIC | Code generation

Priority Group: 4 bits for pre-emption priority 0 bits for subpriority

☐ Sort by Premption Priority and Sub Priority ☐ Sort by interrupts names

Search: Search (Crtl+F) Show: available interrupts ☑ Force DMA channels Interrupts

NVIC Interrupt Table	Enabled	Preemption Priority	Sub Priority
Non maskable interrupt	☑	0	0
Hard fault interrupt	☑	0	0
Memory management fault	☑	0	0
Pre-fetch fault, memory access fault	☑	0	0
Undefined instruction or illegal state	☑	0	0
System service call via SWI instruction	☑	0	0
Debug monitor	☑	0	0
Pendable request for system service	☑	0	0
Time base: System tick timer	☑	0	0
PVD interrupt through EXTI line 16	☐	0	0
Flash global interrupt	☐	0	0
RCC global interrupt	☐	0	0
ADC1, ADC2 and ADC3 global interrupts	☑	1	0
USART3 global interrupt	☐	0	0
EXTI line[15:10] interrupts	☑	0	0
TIM6 global interrupt, DAC1 and DAC2 underrun error interrupts	☐	0	0
FPU global interrupt	☐	0	0

Set ADC interrupt priority lower than the System tick timer due to **HAL_Delay()** in ADC interrupt routine.

Interrupt mode STM32CubeIDE – DAC ADC

ADC Settings: Enable Scan Conversion Mode, and take note of other settings

*Hands-On_03_DAC_ADC_INT.ioc

Pinout & Configuration | Clock Configuration | Project Manager

Software Packs | Pinout

Categories | A->Z

System Core
- CORTEX_M7
- DMA
- GPIO
- IWDG
- NVIC
- RCC
- SYS
- WWDG

Analog
- ADC1
- ADC2
- ADC3
- DAC

Timers

Connectivity
- CAN1
- CAN2
- CAN3
- ETH
- FMC

ADC1 Mode and Configuration

Mode

IN3

Configuration

Reset Configuration

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

Search (Crtl+F)

ADCs_Common_Settings	
Mode	Independent mode
ADC_Settings	
Clock Prescaler	PCLK2 divided by 4
Resolution	12 bits (15 ADC Clock cycles)
Data Alignment	Right alignment
Scan Conversion Mode	Enabled
Continuous Conversion Mode	Disabled
Discontinuous Conversion Mode	Disabled
DMA Continuous Requests	Disabled
End Of Conversion Selection	EOC flag at the end of single channel conversion
ADC_Regular_ConversionMode	
Number Of Conversion	1
External Trigger Conversion Source	Regular Conversion launched by software
External Trigger Conversion Edge	None
Rank	1
Channel	Channel 3
Sampling Time	480 Cycles
ADC_Injected_ConversionMode	
Number Of Conversions	0
WatchDog	
Enable Analog WatchDog Mode	☐

Interrupt mode

STM32CubeIDE – DAC ADC

Add code to **main.c**,

```c
/* main.c */
/* Includes */
#include "main.h"
#include "adc.h"
#include "dac.h"
#include "usart.h"
#include "gpio.h"

/* Private includes */
/* USER CODE BEGIN Includes */
#include <stdio.h>
/* USER CODE END Includes */

/* Private variables */
/* USER CODE BEGIN PV */
uint32_t adc = 0;
uint32_t dac = 0;
float vsense = 3.3 / 4096.0;
float temperature = 0.0;
/* USER CODE END PV */

/* Private function prototypes */
void SystemClock_Config(void);
```

```c
int main(void)
{
  /* MCU Configuration */
  /* Reset of all peripherals, Initializes ... */
  HAL_Init();
  /* Configure the system clock */
  SystemClock_Config();
  /* Initialize all configured peripherals */
  MX_GPIO_Init();
  MX_USART3_UART_Init();
  MX_ADC1_Init();
  MX_DAC_Init();

  /* USER CODE BEGIN 2 */
  HAL_DAC_Start(&hdac, DAC_CHANNEL_1);
  HAL_DAC_SetValue(&hdac, DAC_CHANNEL_1, DAC_ALIGN_12B_R, dac);
  HAL_ADC_Start_IT(&hadc1);
  /* USER CODE END 2 */

  /* Infinite loop */
  /* USER CODE BEGIN WHILE */
  while (1)
  {
      HAL_GPIO_TogglePin(GPIOB, LD3_Pin);
      printf("DAC ADC vales = %ld  %ld  \n\r", dac, adc);
      HAL_Delay(1000);
  /* USER CODE END WHILE */
```

Add code to **main.c**, USER CODE 4

```
/* USER CODE BEGIN 4 */
void HAL_ADC_ConvCpltCallback(ADC_HandleTypeDef* hadc)
{
    adc = HAL_ADC_GetValue(&hadc1);
    HAL_DAC_SetValue(&hdac, DAC_CHANNEL_1, DAC_ALIGN_12B_R, dac);

    ++dac;
    if(dac > 4095) dac = 0;
    HAL_Delay(10);
    HAL_ADC_Start_IT(&hadc1);
}

void HAL_GPIO_EXTI_Callback(uint16_t GPIO_Pin)
{
    if(GPIO_Pin == GPIO_PIN_13)
    {
        HAL_GPIO_TogglePin(GPIOB, LD2_Pin);
    }
}

int __io_putchar(int ch)
{
    uint8_t c[1];
    c[0] = ch & 0x00FF;
    HAL_UART_Transmit(&huart3, &*c, 1, 10);
    return ch;
}
/* USER CODE END 4 */
```

```
/*
int _write(int file, char *ptr, int len)
{
    int DataIdx;
    for(DataIdx= 0; DataIdx < len; DataIdx++)
    {
        __io_putchar(*ptr++);
    }
    return len;
}
*/
```


- End -