



life.augmented

STM32 AZURE RTOS Workshop

FileX

Yilmaz KASAPOGLU

1 FileX/LevelX Overview

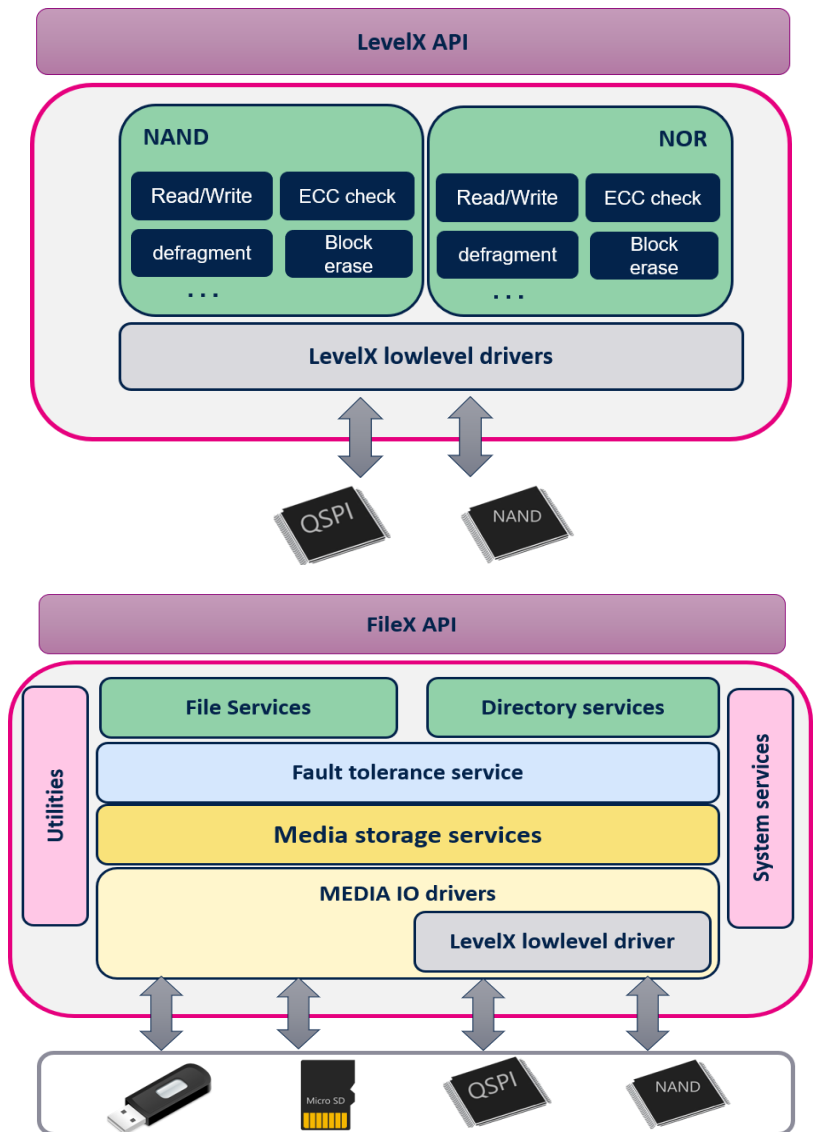
2 STM32 Integration with STM32CubeMX

3 Lab

FileX/LevelX Overview: features

- FileX is a high-performance, file allocation table (FAT)-compatible file system.
 - Storage media management
 - Multi partition support
 - File/Directory Access
 - Fault tolerance
 - Support all common media storage devices
- LevelX is a library that provides API to deal with NAND and NOR flash memories
 - Read/Write Access
 - Wear-leveling to increase disk life time
 - Defragmentation
 - Bad block management

Currently FileX and LevelX requires ThreadX, but there is a plan from MSFT to provide a baremetal version.



FileX/LevelX Overview: APIs

FileX Feature	API	Comments
FileSystem	fx_system_xxx()	Internal init, system time get/set.
Media device	fx_media_xxx()	Managing the media device: format, open, close...
File management	fx_file_xxx() fx_unicode_file_xxx()	Handling file: create, read, write, seek, delete...
Directory management	fx_directory_xxx()	Handling directories:create, delete, find...

LevelX Feature	API	Comments
NAND Flash support	lx_nand_flash_xxx()	NAND flash management: open, close, read block,...
NOR Flash support	lx_nor_flash_xxx()	NOR flash management: open, close, erase block,...

STM32CubeMX: component sector UI

- FileX/LevelX current version requires ThreadX.
- Low level interfaces for FileX and LevelX are exposed in CubeMX. Some of them **are using the BSP/Drivers** to be enhanced in the coming updates.
- It is possible to instantiate different interfaces (SD, OSPI,...)
- Some Interfaces require that the IP HW is enabled in STM32CubeMX.
- Instantiating the same interface multiple times is supported by FileX but **not possible from STM32CubeMX UI**
- FileX USB MSC is managed by the class USBX Host Storage class.

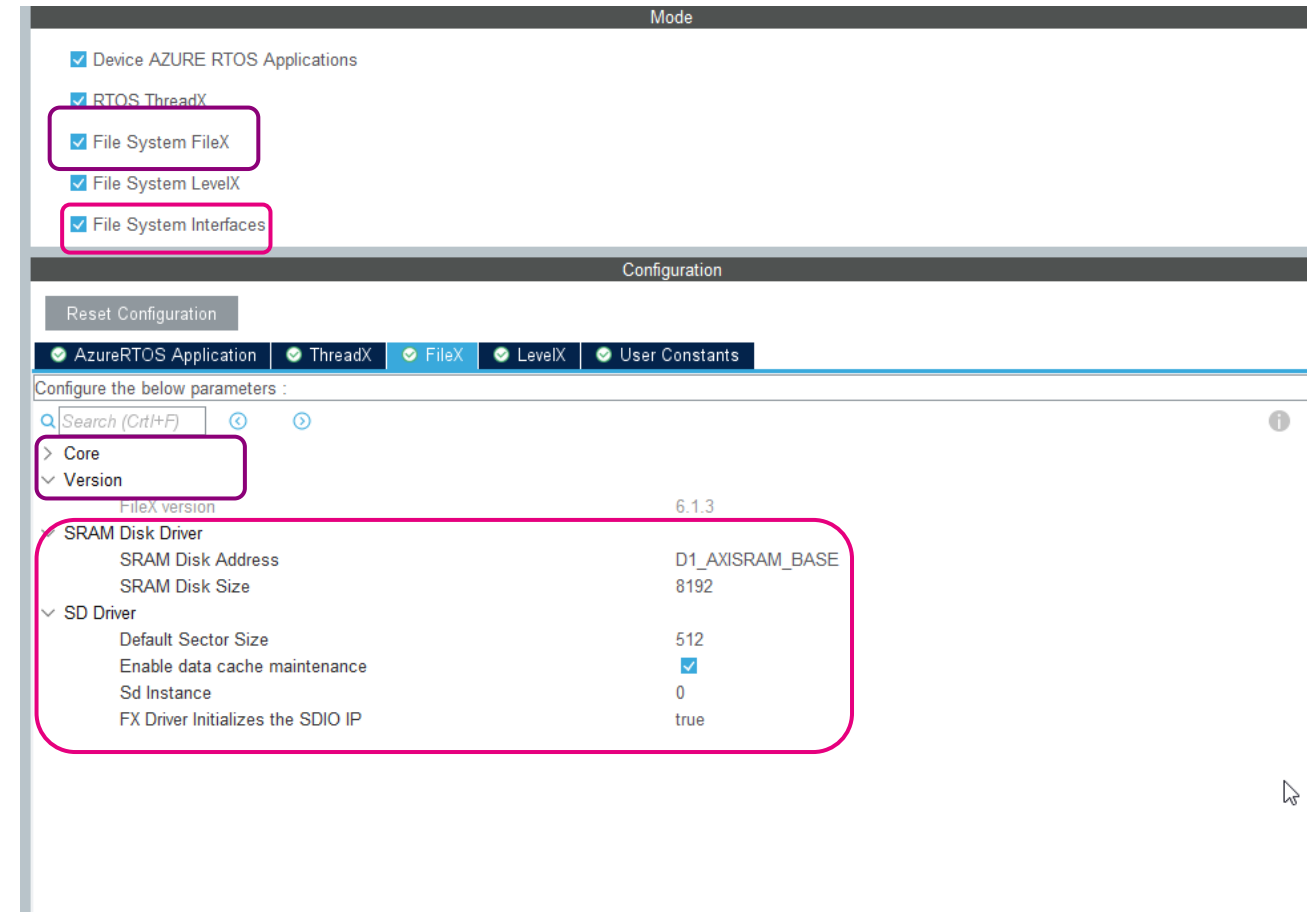
SDMMC1				
SDMMC2				
SPI1				
SPI2				

File System Interfaces				
FileX Internal RAM interface	✓			✓
FileX SD interface	⚠			✓

RTOS ThreadX	✓	6.1.3		
ThreadX / Core	✓			✓
ThreadX / PerformanceInfo				□
ThreadX / TraceX support				□
File System FileX	✓	6.1.3		
FileX / Core	✓			RTOS_M...
FileX / TraceX Support	✓			✓
File System LevelX	✓	6.1.3		
LevelX / NOR Flash Support	✓			✓
LevelX / NAND Flash Support	✓			✓
File System Interfaces	⚠	1.0.0		
FileX Internal RAM interface	✓			✓
FileX SD interface	⚠			✓
FileX LevelX NOR interface				□
FileX LevelX NAND interface				□
FileX Custom interface				□
LevelX QuadSPI memory interface				□
LevelX OctoSPI memory interface				□
LevelX NOR Simulator interface				□
LevelX NOR custom interface				□
LevelX NAND Simulator interface				□
LevelX NAND custom interface				□
Board Part STM32Cube BSP Components		1.0.0		
QuadSPI / MT25TL01G				□
OctoSPI / MX25LM51245G				□
Octal Memory / S70KL1281				□
SDRAM Memory / IS42S16800J				□
SDRAM Memory / IS42S32800J				□
SDRAM Memory / IS42S32800G				□
SDRAM Memory / MT48LC4M32B2				□
Component dependencies				
Component FileX SD interface (from bundle File System Interfaces)				
Requires: component class Device, group SDIO, sub HAL				
Solutions in HAL Drivers:				
Component SDIO/HAL				

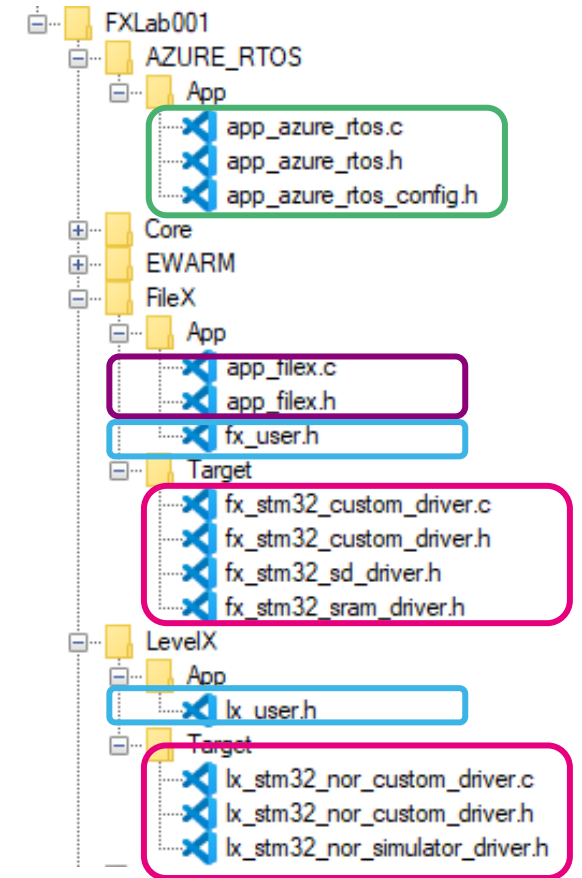
STM32CubeMX: Config UI

- STM32CubeMX Config UI allows the users to:
 - configure the FileX/LevelX core flags.
 - The config flags are written in the file “fx_user.h” generated by STM32CubeMX.
 - Configure specific options for the low-level Interfaces



STM32CubeMX: Project generation content

- `app_azure_rtos.c/.h` `app_azure_rtos_config.h`: files for application start (`MX_AZURE_RTOS_Init()`)
- `app_filex.c/.h` :user application code files to implement FileX support
- `fx_user.h/lx_user.h`: FileX/LevelX config files.
- `fx_stm32_xxx_driver.h/c` : driver files for the low-level interfaces for FileX
- When a custom driver is needed, the driver files should be implemented under the application.



CubeAzure FileX: LowLevel Drivers

STM32 FileX HW low-level drivers:

- FileX drivers allow the interaction with physical media storage
- When combined with LevelX it is possible to have a FileSysten on NOR and NAND memories.
- Each driver is split in 2 files:
 - Driver implementation provided as “pattern”
 - Driver header provided as template at application level.

Link to FileX low-level drivers

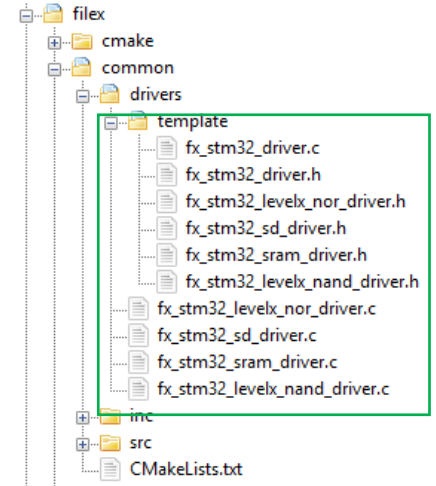
- Each FileX driver has a main entry function that is registered by the main application like via functions “fx_media_format()” or “fx_media_open()” functions

```
VOID fx_stm32_sd_driver(FX_MEDIA *media_ptr)
```

```
/* Open the SD disk driver. */  
status = fx_media_open(&sdio_disk, "STM32_SDIO_DISK", fx_stm32_sd_driver, 0, (VOID *) media_memory, sizeof(media_memory));
```



life.augmented



FileX lowLevel drivers

```
VOID fx_stm32_sd_driver(FX_MEDIA *media_ptr)
{
    UINT status;
    UINT unaligned_buffer = 0;
    ULONG partition_start;
    ULONG partition_size;

    #if (FX_DRIVER_CALLS_SD_INIT == 0)
        is_initialized = 1; /* the SD was initialized by the application */
    #endif

    /* before performing any operation, check the status of the SDMMC */
    if (is_initialized == 1)
    {
        if (check_sd_status(SD_INSTANCE) != BSP_ERROR_NONE)
        {
            media_ptr->fx_media_driver_status = FX_IO_ERROR;
            return;
        }
    }

    /* Process the driver request specified in the media control block. */
    switch(media_ptr->fx_media_driver_request)
    {
        case FX_DRIVER_INIT:
        {
            #if (FX_DRIVER_CALLS_SD_INIT == 1)
                /* Initialize the SD instance */
                if (is_initialized == 0)
                {
                    status = fx_sd_driver_init(SD_INSTANCE);

                    if (status == BSP_ERROR_NONE)
                    {
                        is_initialized = 1;
                    }
                }
            #endif

            /* Create a counting semaphore to check the DMA transfer status */
            if (tx_semaphore_create(&transfer_semaphore, "sdmmc dma transfer semaphore", 1) != TX_SUCCESS)
            {
                media_ptr->fx_media_driver_status = FX_IO_ERROR;
            }
            else
            {
                media_ptr->fx_media_driver_status = FX_SUCCESS;
            }
        }

        #if (FX_DRIVER_CALLS_SD_INIT == 1)
        {
            }
        else
        {
            media_ptr->fx_media_driver_status = FX_IO_ERROR;
        }
    }

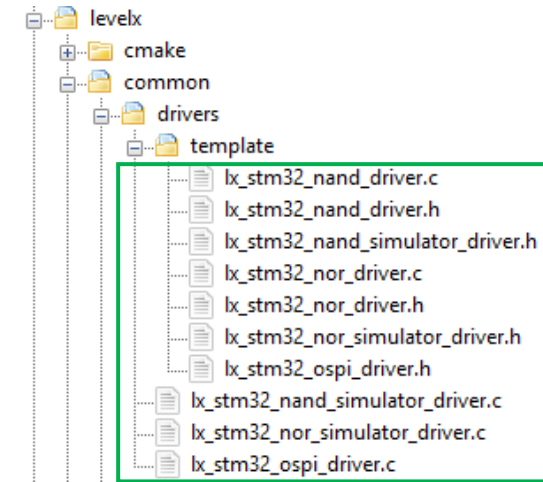
    #endif
    break;
}

case FX_DRIVER_UNINIT:
```


CubeAzure LevelX: LowLevel Drivers

Link to LevelX low-level drivers

- Both NOR and NAND drivers requires a main entry function that is the unique interface to handle the LevelX core stack requests (read, write, erase block...)
- The driver is registered by the application when needed.



LevelX low-level drivers.

```
#include "lx_api.h"

UINT lx_stm32_qspi_initialize(LX_NOR_FLASH *nor_flash);

UINT lx_nor_flash_open(LX_NOR_FLASH *nor_flash, CHAR *name, UINT (*nor_driver_initialize)(LX_NOR_FLASH *));

UINT lx_stm32_qspi_initialize(LX_NOR_FLASH *nor_flash)
{
    BSP_QSPI_Init_t qspi_info;

    if (is_initialized == LX_FALSE)
    {
        if(BSP_QSPI_GetInfo(QSPI_INSTANCE, &qspi_info) != BSP_ERROR_NONE)
        {
            return LX_ERROR;
        }
    }

    #if (LX_DRIVER_CALLS_QSPI_INIT == 1)

    TX_INTERRUPT_SAVE_AREA
    BSP_QSPI_Init_t qspi_config;

    /* QSPI device configuration */
    qspi_config.InterfaceMode = BSP_QSPI_QPI_MODE;
    qspi_config.TransferRate = BSP_QSPI_DTR_TRANSFER;

    TX_DISABLE

    if(BSP_QSPI_Init(QSPI_INSTANCE, &qspi_config) != BSP_ERROR_NONE)
    {
        return LX_ERROR;
    }

    #if (LX_DRIVER_ERASES_QSPI_AFTER_INIT == 1)
    if( BSP_QSPI_EraseChip(QSPI_INSTANCE) != BSP_ERROR_NONE)
    {
        return LX_ERROR;
    }

    if(check_status() != LX_SUCCESS)
    {
        return LX_ERROR;
    }
    #endif

    TX_RESTORE
    #endif
}
```

Lab – FileX Implementation

- Implementing FileX support using STM32CubeMx from scratch
- By this lab, we will;
 - create a file system on the internal SRAM,
 - open the file in write mode and write data,
 - re-open the file in read mode and read the file content.

Launch STM32CubeMx

The screenshot displays the STM32CubeMX application window. The top menu bar includes 'Window' and 'Help'. On the right, there are social media icons for Facebook, YouTube, Twitter, and LinkedIn, along with the ST logo. The main interface is divided into two panels. The left panel, titled 'New Project', contains a sidebar with four 'MX' icons and a central dark blue box with the heading 'I need to :'. This box lists three options: 'Start My project from MCU' with an 'ACCESS TO MCU SELECTOR' button, 'Start My project from ST Board' with an 'ACCESS TO BOARD SELECTOR' button (highlighted with a red box), and 'Start My project from Example' with an 'ACCESS TO EXAMPLE SELECTOR' button. The right panel, titled 'Manage software installations', contains two sections: 'Check for STM32CubeMX and embedded software package...' with a 'CHECK FOR UPDATES' button, and 'Install or remove embedded software packages' with an 'INSTALL / REMOVE' button. A bottom panel shows a preview of the 'New Project' workflow with a blue overlay text: 'Start your project from one of the 5000+ tested examples provided by ST' and 'Half of examples directly modifiable'.

Window Help

10th Anniversary

f y t

ST

New Project

I need to :

Start My project from MCU

ACCESS TO MCU SELECTOR

Start My project from ST Board

ACCESS TO BOARD SELECTOR

Start My project from Example

ACCESS TO EXAMPLE SELECTOR

Manage software installations

Check for STM32CubeMX and embedded software package...

CHECK FOR UPDATES

Install or remove embedded software packages

INSTALL / REMOVE

Start your project from one of the 5000+ tested examples provided by ST

Half of examples directly modifiable

Start new project with board selector



STM32CubeMX Board Selector

MX New Project from a Board

MCU/MPU Selector Board Selector Example Selector Cross Selector

Board Filters

Commercial Part Number: H723
Vendor: >
Type: >
MCU/MPU Series: >
Other: >
Peripheral: >

1 Type H723 in the search box

2 Select NUCLEO-H723ZG in the list box


New multicore STM32MP1 Series for Industrial and IoT applications

STM32MP1

OpenSTLinux Distribution

ST

Boards List: 1 item

	Overview	Commercial Part No	Type	Marketing Status	Unit Price (US\$)	Mounted Device
☆		NUCLEO-H723ZG	Nucleo-144	Active	29.0	STM32H723ZGtx

STM32CubeMX Board Selector

MX New Project from a Board

MCU/MPU Selector Board Selector Example Selector Cross Selector

Board Filters

- Commercial Part Number: NUCLEO-H723ZG
- Vendor: >
- Type: >
- MCU/MPU Series: >
- Other: >
- Peripheral: >

Features Large Picture Docs & Resources Datasheet Buy Start Project

STM32H7 Series

NUCLEO-H723ZG **STMicroelectronics NUCLEO-H723ZG Board Support and Examples**

ACTIVE Active
Product is in mass production

Part Number : NUCLEO-H723ZG
Commercial Part Number : NUCLEO-H723ZG

Unit Price (US\$) : 29.0
Mounted Device : [STM32H723ZGTx](#)

The STM32 Nucleo-144 board provides an affordable and flexible way for users to try out new concepts and build prototypes by choosing from the various combinations of performance and power consumption features, provided by the STM32 microcontroller. For the compatible boards, the internal or external SMPS significantly reduces power consumption in Run mode.

The ST Zio connector, which extends the ARDUINO® Uno V3 connectivity, and the ST morpho headers provide an easy means of expanding the functionality of the Nucleo open development platform with a wide choice of specialized shields.

Require any separate probe as it integrates the ST-LINK


the STM32 comprehensive free software libraries and examples available

MX Board Project Options: NUCLEO-H723ZG

? Initialize all peripherals with their default Mode ?

Yes No

Boards List: 1 item

	Overview	Commercial Part No	Type	Marketing Status	Unit Price (US\$)	Mounted Device
1		NUCLEO-H723ZG	Nucleo-144	Active	29.0	STM32H723ZGTx

Export

Double click on the board picture or name

Initialize all peripherals with default mode for Nucleo-H723ZG

Add X-CUBE-AZRTOS-H7 Support

MX STM32CubeMX Untitled*: STM32H723ZGTx NUCLEO-H723ZG

The screenshot shows the STM32CubeMX interface with the 'Pinout & Configuration' tab selected. The 'Software Packs' menu is open, and the 'Select Components' option is highlighted. A callout box with the text 'Start Software Packs Component Selector' points to this option. The pinout diagram of the STM32H723ZGTx is visible on the right side of the interface.

STM32CubeMX

File Window Help

Home > STM32H723ZGTx - NUCLEO-H723ZG > Untitled - Pinout & Configuration

Pinout & Configuration Clock Configuration Project Manager

Search

Categories A->Z

- System Core >
- Analog >
- Timers >
- Connectivity >
- Multimedia >
- Security >
- Computing >
- Middleware >
- Trace and Debug >
- Power and Thermal >

Start Software Packs Component Selector

Software Packs

- Select Components Alt-O
- Manage Software Packs Alt-U

Pinout

Pinout view System view

Add pack software component to the project

STM32H723ZGTx

Add X-CUBE-AZRTOS-H7 Support

Software Packs Component Selector

Packs

Select components to enable Azure RTOS and FileX support for internal SRAM

- 1 Expand X-CUBE-AZRTOS-H7
- 2 Expand RTOS ThreadX
- 3 Select ThreadX / Core

Pack / Bundle / Component	Status	Version	Selection
> RoweBots.I-CUBE-UNISONRTOS		5.5.0-4	Install
> STMicroelectronics.X-CUBE-AI		6.0.0	Install
> STMicroelectronics.X-CUBE-ALGOBUILD		1.2.0	Install
▼ STMicroelectronics.X-CUBE-AZRTOS-H7	✓	2.0.0	
▼ RTOS ThreadX	✓	6.1.7	
ThreadX / Core	✓		<input checked="" type="checkbox"/>
ThreadX / PerformanceInfo			<input type="checkbox"/>
ThreadX / TraceX support			<input type="checkbox"/>
ThreadX / Low Power support			<input type="checkbox"/>
> File System FileX		6.1.7	
> File System LevelX		6.1.7	
> File System Interfaces		2.0.0	
> USB USBX		6.1.7	
> Network NetXDuo		6.1.7	
> Network Interfaces		2.0.0	
> Board Part STM32Cube_BSP_Components		1.1.0	
> STMicroelectronics.X-CUBE-BLE2		3.2.0	Install
> STMicroelectronics.X-CUBE-GNSS1		5.2.0	Install

Ok Cancel

Add X-CUBE-AZRTOS-H7 Support

Software Packs Component Selector

Packs

Pack / Bundle / Component	Status	Version	Selection
> RoweBots.I-CUBE-UNISONRTOS		5.5.0-4	Install
> STMicroelectronics.X-CUBE-AI		6.0.0	Install
> STMicroelectronics.X-CUBE-ALGOBUILD		1.2.0	Install
✓ STMicroelectronics.X-CUBE-AZRTOS-H7	✓	2.0.0	
✓ RTOS ThreadX	✓	6.1.7	
ThreadX / Core	✓		<input checked="" type="checkbox"/>
ThreadX / PerformanceInfo			<input type="checkbox"/>
ThreadX / TraceX support			<input type="checkbox"/>
ThreadX / Low Power support			<input type="checkbox"/>
✓ File System FileX	✓	6.1.7	
FileX / Core	✓		<input checked="" type="checkbox"/>
FileX / TraceX Support			<input type="checkbox"/>
> File System LevelX		6.1.7	
> File System Interfaces		2.0.0	
> USB USBX		6.1.7	
> Network NetXDuo		6.1.7	
> Network Interfaces		2.0.0	
> Board Part STM32Cube_BSP_Components		1.1.0	

1 Expand File System FileX

2 Select FileX / Core

Ok Cancel

Add X-CUBE-AZRTOS-H7 Support

Software Packs Component Selector

Packs

Pack / Bundle / Component	Status	Version	Selection
ThreadX / TraceX support			<input type="checkbox"/>
ThreadX / Low Power support			<input type="checkbox"/>
File System FileX	✓	6.1.7	
FileX / Core	✓		<input checked="" type="checkbox"/>
FileX / TraceX Support			<input type="checkbox"/>
File System LevelX		6.1.7	
File System Interfaces	✓	2.0.0	
FileX Internal RAM interface	✓		<input checked="" type="checkbox"/>
FileX SD interface			<input type="checkbox"/>
FileX LevelX NOR interface			<input type="checkbox"/>
FileX LevelX NAND interface			<input type="checkbox"/>
FileX Custom interface			<input type="checkbox"/>
LevelX QuadSPI memory interface			<input type="checkbox"/>
LevelX OctoSPI memory interface			<input type="checkbox"/>
LevelX NOR Simulator interface			<input type="checkbox"/>
LevelX NOR custom interface			<input type="checkbox"/>
LevelX NAND Simulator interface			<input type="checkbox"/>
LevelX NAND custom interface			<input type="checkbox"/>

1 Expand File System Interfaces

2 Select FileX Internal RAM Interface

Click OK

3

Ok Cancel

Configure X-CUBE-AZRTOS

Home > STM32H723ZGTx - NUCLEO-H723ZG > Untitled - Pinout & Configuration

Pinout & Configuration | Clock Configuration | Project Manager

Software Packs | Pinout

Search: []

Categories: A->Z

- System Core >
- Analog >
- Timers >
- Connectivity >
- Multimedia >
- Security >
- Computing >
- Middleware
- Trace and Debug
- Power and Thermal
- Software Packs

1

STM32H723ZGTx - NUCLEO-H723ZG

Mode

- ☒ RTOS ThreadX
- ☒ File System FileX
- ☒ File System Interfaces

2

Select all

Configuration

Memory Configuration

Memory Allocation	Use Static MemPool Allocation
ThreadX memory pool size	1024
FileX memory pool size	1024

Click on Software Packs & STM32H723ZGTx - NUCLEO-H723ZG

Pinout view

STM32H723ZGTx - NUCLEO-H723ZG

STM32 LQ

Configure X-CUBE-AZRTOS

Home > STM32H723ZGTx - NUCLEO-H723ZG > Untitled - Pinout & Configuration

Pinout & Configuration | Clock Configuration | Project Manager

Software Packs | Pinout

Search []

Categories: A->Z

- System Core >
- Analog >
- Timers >
- Connectivity >
- Multimedia >
- Security >
- Computing >

STMicroelectronics.X-CUBE-AZRTOS-H7.2.0.0 Mode and Configuration

Mode

- ☒ RTOS ThreadX
- ☒ File System FileX
- ☒ File System Interfaces

Configuration

2

Select FileX tab

1

Set FileX memory pool size to 3*1024

Configure the below parameters :

Search (Ctrl+F)

Memory Configuration

Memory Allocation	Use Static MemPool Allocation
ThreadX memory pool size	1024
FileX memory pool size	3*1024

Pinout view

Pinout diagram showing various pins and their connections.

SRAM Disk Driver

Please note that;

- **SRAM Disk Address** is configured as **D1_AXISRAM_BASE** by CubeMX, however this address space is used to store FileX version information.
- Therefore we will use another address space in the internal RAM to create our disk media.

Configuration

Reset Configuration

✓ AzureRTOS Application | ✓ ThreadX | **✓ FileX** | ✓ User Constants

Configure the below parameters :

Search (Ctrl+F)

FX_MAX_FAT_CACHE	16
FX_MAX_LAST_NAME_LEN	256
FX_MAX_LONG_NAME_LEN	256
FX_MAX_SECTOR_CACHE	256
FX_MEDIA_DISABLE_SEARCH_CACHE	Disabled
FX_MEDIA_STATISTICS_DISABLE	Disabled
FX_NO_LOCAL_PATH	Disabled
FX_NO_TIMER	Disabled
FX_RENAME_PATH_INHERIT	Disabled
FX_SINGLE_	
FX_SINGLE_	
FX_UPDATE_	
FX_UPDATE_	
MAX_FAT_CACHE_NB_BIT	4
MAX_SECTOR_CACHE_NB_BIT	8

✓ STMicroelectronics.X-CUBE-AZRTOS-H7.2....

Click on the text box containing SRAM Disk Address value

1

2

Type and change the value from D1_AXISRAM_BASE to D1_AXISRAM2_BASE

Set Timebase Source for the RTOS

- When RTOS is used, it is strongly recommended to use a HAL timebase source other than the SysTick.

The screenshot shows the STM32CubeIDE interface with the 'Pinout & Configuration' tab selected. The breadcrumb trail at the top reads: Home > STM32H723ZGTx - NUCLEO-H723ZG > Untitled - Pinout & Configuration. The left sidebar has a search bar and a 'Categories' dropdown set to 'A->Z'. A list of system components is shown, including 'System Core', 'Cortex-M', 'DMA', 'GPIO', 'IWDG1', 'MDMA', 'NVIC', 'RAMECC', 'RCC', 'SYS', and 'Analog'. The 'SYS' component is highlighted with a blue bar. The right pane shows the 'Clock Configuration' tab with a 'Software Packs' dropdown and a 'Pinout' dropdown. Under 'SYS Mode and Configuration', the 'Timebase Source' is set to 'SysTick'. A dropdown menu is open, showing a list of available sources: 'SysTick', 'TIM1', 'TIM2', 'TIM3', 'TIM4', 'TIM5', 'TIM6', and 'TIM7'. 'TIM6' is highlighted with a blue bar. A warning message at the bottom states: 'Warning: This peripheral has no parameters to be configured.'

1 Click on System Core

2 Click on SYS

3 Select TIM6 as the new HAL timebase source

Project Settings

Home > STM32H723ZGTx - NUCLEO-H723

Switch to Project Manager 1

Pinout & Configuration Clock Configuration Project Manager Tools

Project

Project Settings

Project Name
FileX_HandsOn

Project Location
C:\STM32-AZURE-RTOS-WS\HandsOn\ Browse

Application Structure
Advanced ☐ Do not generate the main()

Toolchain Folder Location
C:\STM32-AZURE-RTOS-WS\HandsOn\FileX_HandsOn\

Code Generator

Toolchain / IDE
STM32CubeIDE ☒ Generate Under Root

Advanced Settings

Linker Settings

Minimum Heap Size 0x200

Minimum Stack Size 0x400

Thread-safe Settings

Cortex-M7NS

☐ Enable multi-threaded support

Thread-safe Locking Strategy
Default - Mapping suitable strategy depending on RTOS selection.

Mcu and Firmware Package

Mcu Reference
STM32H723ZGTx

Firmware Package Name and Version
STM32Cube FW_H7 V1.9.0

2 Specify Project Name & Location

3 Select STM32CubeIDE as the toolchain

GENERATE CODE

Code Generation

The screenshot displays the STM32CubeIDE Project Manager window. The top navigation bar includes 'Home', 'STM32H723ZGTx - NUCLEO-H723ZG', 'Untitled - Project Manager', and a 'GENERATE CODE' button. The left sidebar has tabs for 'Project', 'Code Generator', and 'Advanced Settings'. The 'Project' tab is active, showing fields for Project Name (FileX_HandsOn), Project Location (C:\STM32-AZURE-RTOS-WS\HandsOn\), Application Structure (Advanced), Toolchain Folder Location (C:\STM32-AZURE-RTOS-WS\HandsOn\FileX_HandsOn\), Toolchain / IDE (STM32CubeIDE), Linker Settings (Minimum Heap Size: 0x200, Minimum Stack Size: 0x400), Thread-safe Settings (Cortex-M7NS, Enable multi-threaded support: unchecked, Thread-safe Locking Strategy: Default), and Mcu and Firmware Package (Mcu Reference: STM32H723ZGTx, Firmware Package Name and Version: STM32Cube FW_H7 V1.9.0). A 'Code Generation' dialog box is open, displaying the message: 'The Code is successfully generated under : C:/STM32-AZURE-RTOS-WS/HandsOn/FileX_HandsOn' and 'Project language : C'. The dialog has three buttons: 'Open Folder', 'Open Project' (highlighted with a pink box), and 'Close'. A pink callout box with the number '1' and a circular arrow icon points to the 'GENERATE CODE' button in the top bar. Another pink callout box with the number '2' and a circular arrow icon points to the 'Open Project' button in the dialog box.

Home > STM32H723ZGTx - NUCLEO-H723ZG > Untitled - Project Manager > **GENERATE CODE**

Pinout & Configuration | Clock Configuration | Project Manager | Tools

Project

Project Settings

Project Name
FileX_HandsOn

Project Location
C:\STM32-AZURE-RTOS-WS\HandsOn\ Browse

Application Structure
Advanced ☐ Do not generate the main()

Toolchain Folder Location
C:\STM32-AZURE-RTOS-WS\HandsOn\FileX_HandsOn\

Toolchain / IDE
STM32CubeIDE

Code Generator

Linker Settings

Minimum Heap Size 0x200

Minimum Stack Size 0x400

Advanced Settings

Thread-safe Settings

Cortex-M7NS

☐ Enable multi-threaded support

Thread-safe Locking Strategy
Default - Mapping suitable strategy depending on RTOS selection.

Mcu and Firmware Package

Mcu Reference
STM32H723ZGTx

Firmware Package Name and Version
STM32Cube FW_H7 V1.9.0

Code Generation

The Code is successfully generated under :
C:/STM32-AZURE-RTOS-WS/HandsOn/FileX_HandsOn

Project language : C

Open Folder **Open Project** Close

Click on **GENERATE CODE**

Click on **Open Project**

Launch STM32CubeIDE

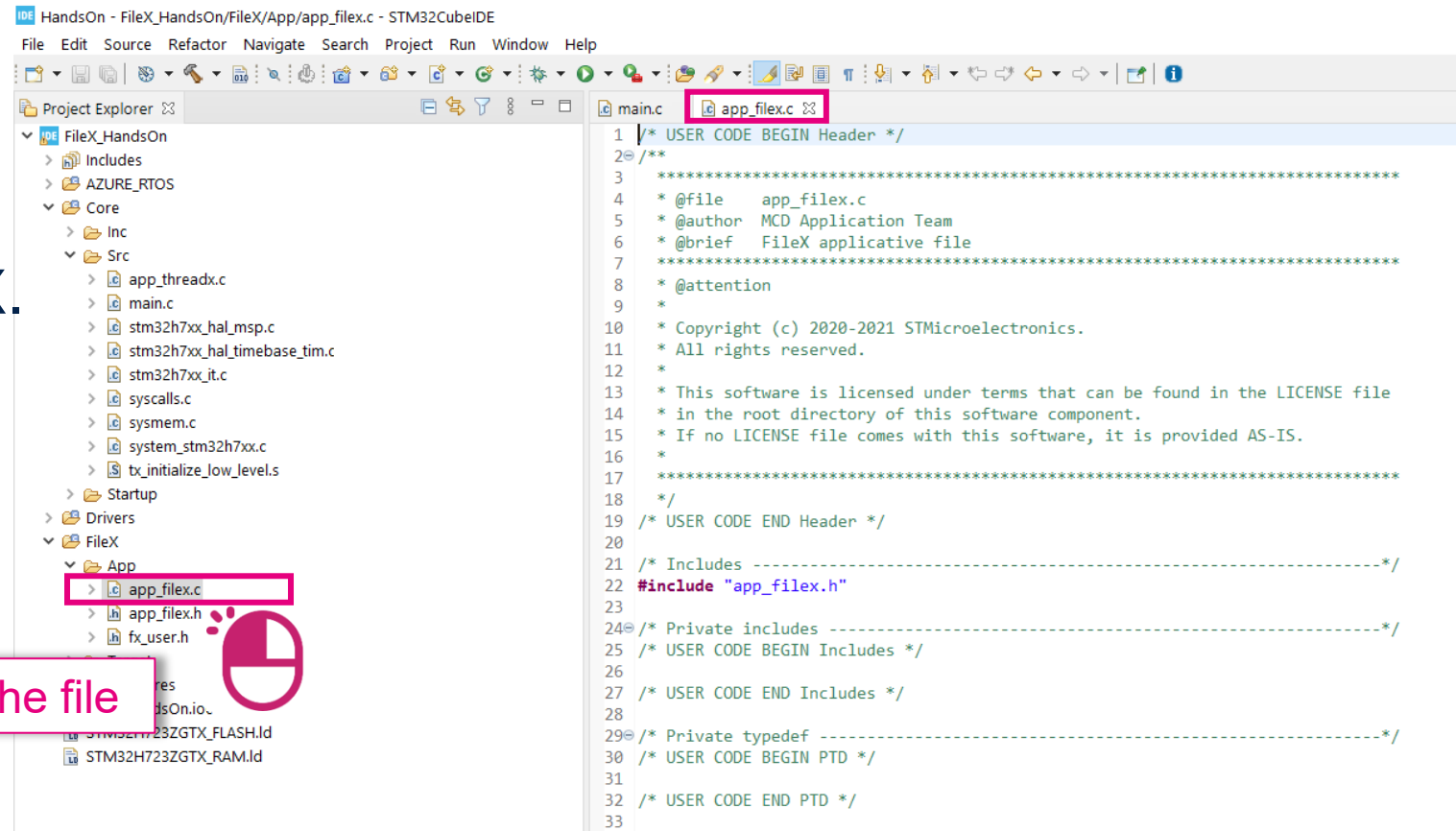
The screenshot shows the STM32CubeIDE Project Manager interface. The top navigation bar includes 'Home', 'STM32H723ZGTx - NUCLEO-H723ZG', 'Untitled - Project Manager', and a 'GENERATE CODE' button. The left sidebar has tabs for 'Pinout & Configuration', 'Clock Configuration', 'Project Manager' (selected), and 'Tools'. The 'Project Manager' tab is active, showing a tree view with 'Project', 'Code Generator', and 'Advanced Settings'. The 'Project' section is expanded, showing fields for 'Project Name' (FileX_HandsOn), 'Project Location' (C:\STM32-AZURE-RTOS-WS\HandsOn), 'Application Structure' (Advanced), 'Toolchain Folder Location' (C:\STM32-AZURE-RTOS-WS\HandsOn), 'Toolchain / IDE' (STM32CubeIDE), 'Linker Settings' (Minimum Heap Size: 0x200, Minimum Stack Size: 0x400), 'Thread-safe Settings' (Cortex-M7NS, Enable multi-threaded support: unchecked, Thread-safe Locking Strategy: Default - Mapping suitable strategy depending on RTOS selection), and 'Mcu and Firmware Package' (Mcu Reference: STM32H723ZGTx, Firmware Package Name and Version: STM32Cube FW_H7 V1.9.0).

A dialog box titled 'STM32CubeIDE Launcher' is open, prompting the user to 'Select a directory as workspace'. The dialog explains that STM32CubeIDE uses the workspace directory to store its preferences and development artifacts. The 'Workspace' field is set to 'C:\STM32-AZURE-RTOS-WS\HandsOn' and is highlighted with a red box and a red circle labeled '1'. A 'Browse...' button is next to the field. Below the workspace field, there is a checkbox for 'Use this as the default and do not ask again' and a section for 'Recent Workspaces'. At the bottom right of the dialog, the 'Launch' button is highlighted with a red box and a red circle labeled '2', and a red circle icon is next to it. A red box with the text 'Specify workspace location & name' is positioned above the workspace field, and another red box with the text 'Click on Launch to start STM32CubeIDE' is positioned below the 'Launch' button.

Generated Code

- Generated code is a buildable project with the project tree structure seen on the right.
- Azure RTOS and FileX support added to the project by CubeMX.
- We will need to **modify** “**app_filex.c**” to implement our file system.

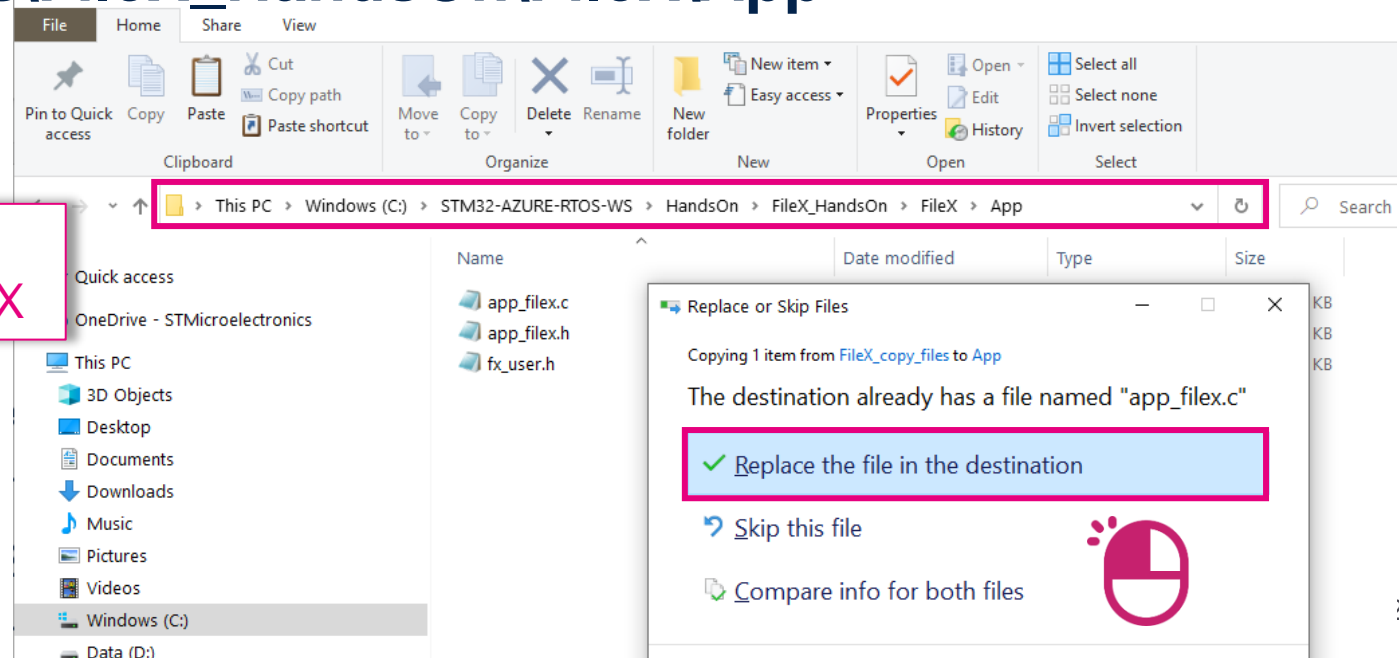
Double click on “app_filex.c” to edit the file



Implementing the File System

- At this step we will use completed code to save time
- Path for the completed code :
C:\STM32-AZURE-RTOS-WS\HandsOn\FileX_copy_files
- Path for the FileX application files in the generated project :
C:\STM32_AzureRTOS_WS\Labs\FileX_HandsOn\FileX\App

Copy the completed “**app_filex.c**” file & replace it with the one generated by CubeMX



Implementing the File System

- In the completed “**app_filex.c**” file, defined by the user:
- **Default sector and stack sizes,**
- **Default thread priority**
- **Default preemption threshold**
- **fx_thread_entry ()**
is the main thread and should be implemented by the user after the code generation

```
main.c  app_filex.c ✕
34 /* Private define -----*/
35 /* USER CODE BEGIN PD */
36 #define DEFAULT_SECTOR_SIZE      512
37 #define DEFAULT_STACK_SIZE      (2 * 1024)
38 /* Thread_0 priority */
39 #define DEFAULT_THREAD_PRIO      10
40 /* Thread_0 preemption priority */
41 #define DEFAULT_PREEMPTION_THRESHOLD  DEFAULT_THREAD_PRIO
42 /* USER CODE END PD */
43
44 /* Private macro -----*/
45 /* USER CODE BEGIN PM */
46
47 /* USER CODE END PM */
48
49 /* Private variables -----*/
50 /* USER CODE BEGIN PV */
51 UCHAR      *media_memory;
52 /* Define FileX global data structures. */
53 FX_MEDIA    ram_disk;
54 FX_FILE     fx_file;
55 /* Define ThreadX global data structures. */
56 TX_THREAD   fx_thread;
57 /* USER CODE END PV */
58
59 /* Private function prototypes -----*/
60 /* USER CODE BEGIN PFP */
61 void fx_thread_entry(ULONG thread_input); ← FileX thread entry function defined by the user
62 void Error_Handler(void);
63 /* USER CODE END PFP */
64
```

Implementing the File System

- App_FileX_Init() is the only function generated by CubeMx in “app_filex.c” file

```
70 ①UINT MX_FileX_Init(VOID *memory_ptr)
71  {
72     UINT ret = FX_SUCCESS;
73     TX_BYTE_POOL *byte_pool = (TX_BYTE_POOL*)memory_ptr;
74
75     /* USER CODE BEGIN MX_FileX_MEM_POOL */
76     (void)byte_pool;
77     /* USER CODE END MX_FileX_MEM_POOL */
78
79     /* USER CODE BEGIN MX_FileX_Init */
80     VOID *pointer;
81
82     /* Allocate memory for the FileX thread's stack */
83     ret = tx_byte_allocate(byte_pool, &pointer, DEFAULT_STACK_SIZE, TX_NO_WAIT);
84
85     if (ret != FX_SUCCESS)
86     {
87         /* Failed at allocating memory */
88         Error_Handler();
89     }
90
```

We need to allocate the memory for the stack of FileX thread with the DEFAULT_STACK_SIZE defined by us on top of the file



Implementing the File System

```
88     Error_Handler();
89 }
90
91 /* Create the main thread. */
92 tx_thread_create(&fx_thread, "thread 0", fx_thread_entry, 0, pointer, DEFAULT_STACK_SIZE, DEFAULT_THREAD_PRIO,
93                 DEFAULT_THREAD_PRIO, TX_NO_TIME_SLICE, TX_AUTO_START);
94
95 /* Allocate memory for the media cache */
96 ret = tx_byte_allocate(byte_pool, (VOID**) &media_memory, DEFAULT_SECTOR_SIZE, TX_NO_WAIT);
97
98 if (ret != FX_SUCCESS)
99 {
100     /* Failed at allocating memory */
101     Error_Handler();
102 }
103
104 /* Initialize FileX. */
105 fx_system_initialize();
106 /* USER CODE END MX_FileX_Init */
107 return ret;
108 }
109
110 /* USER CODE BEGIN 1 */
```

Main thread is created here with the function name `fx_thread_entry()`

Cache memory allocation for the disk media with the `DEFAULT_SECTOR_SIZE`

Finally, Azure RTOS FileX system is initialized

Through the Code

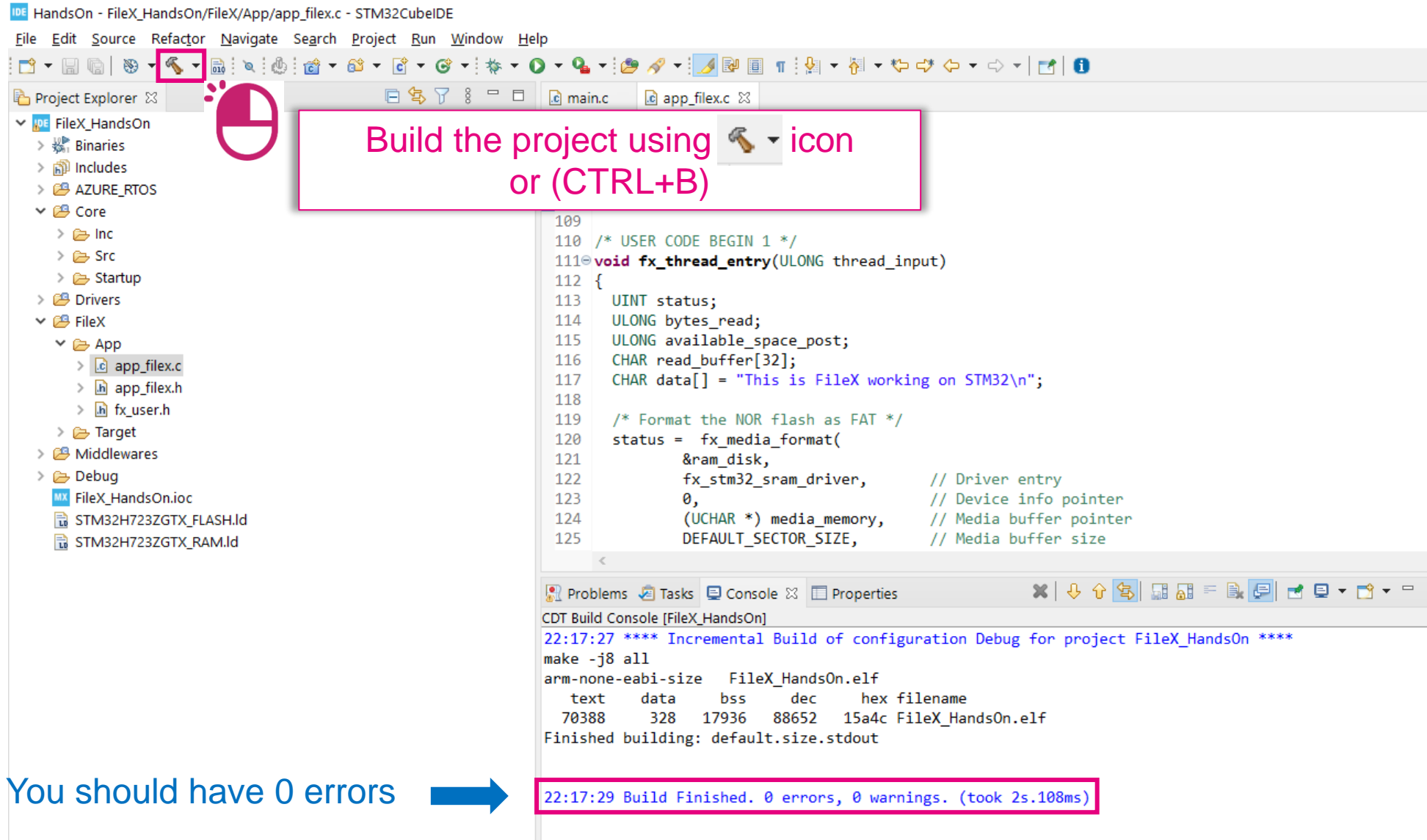
```
110 /* USER CODE BEGIN 1 */
111 void fx_thread_entry(ULONG thread_input)
112 {
113     UINT status;
114     ULONG bytes_read;
115     ULONG available_space_post;
116     CHAR read_buffer[32];
117     CHAR data[] = "This is FileX working on STM32\n";
118
119     /* Format the NOR flash as FAT */
120     status = fx_media_format(
121         &ram_disk,
122         fx_stm32_sram_driver, // Driver entry
123         0, // Device info pointer
124         (UCHAR *) media_memory, // Media buffer pointer
125         DEFAULT_SECTOR_SIZE, // Media buffer size
126         "RAM_DISK", // Volume Name
127         1, // Number of FATs
128         32, // Directory Entries
129         0, // Hidden sectors
130         64, // Total sectors
131         DEFAULT_SECTOR_SIZE, // Sector size
132         8, // Sectors per cluster
133         1, // Heads
134         1 // Sectors per track
135     );
136
```

← Main thread function.

This part of the code will be explained during the debug process.

- Now we are ready to build the code

Build the Project



HandsOn - FileX_HandsOn/FileX/App/app_filex.c - STM32CubeIDE

File Edit Source Refactor Navigate Search Project Run Window Help

Project Explorer

- FileX_HandsOn
 - Binaries
 - Includes
 - AZURE_RTOS
 - Core
 - Inc
 - Src
 - Startup
 - Drivers
 - FileX
 - App
 - app_filex.c
 - app_filex.h
 - fx_user.h
 - Target
 - Middleware
 - Debug
 - FileX_HandsOn.ioc
 - STM32H723ZGTX_FLASH.ld
 - STM32H723ZGTX_RAM.ld

Build the project using icon or (CTRL+B)

```
109
110 /* USER CODE BEGIN 1 */
111 void fx_thread_entry(ULONG thread_input)
112 {
113     UINT status;
114     ULONG bytes_read;
115     ULONG available_space_post;
116     CHAR read_buffer[32];
117     CHAR data[] = "This is FileX working on STM32\n";
118
119     /* Format the NOR flash as FAT */
120     status = fx_media_format(
121         &ram_disk,
122         fx_stm32_sram_driver, // Driver entry
123         0, // Device info pointer
124         (UCHAR *) media_memory, // Media buffer pointer
125         DEFAULT_SECTOR_SIZE, // Media buffer size
126     );
127 }
```

Problems Tasks Console Properties

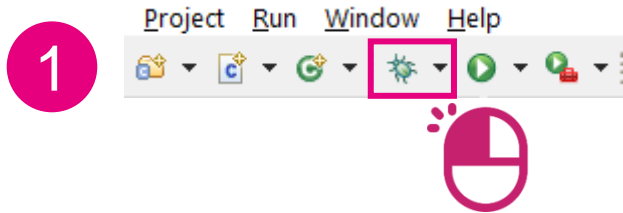
CDT Build Console [FileX_HandsOn]


```
22:17:27 **** Incremental Build of configuration Debug for project FileX_HandsOn ****
make -j8 all
arm-none-eabi-size FileX_HandsOn.elf
  text  data  bss   dec   hex filename
 70388   328 17936 88652 15a4c FileX_HandsOn.elf
Finished building: default.size.stdout
```

22:17:29 Build Finished. 0 errors, 0 warnings. (took 2s.108ms)

You should have 0 errors →

Start the Debugger

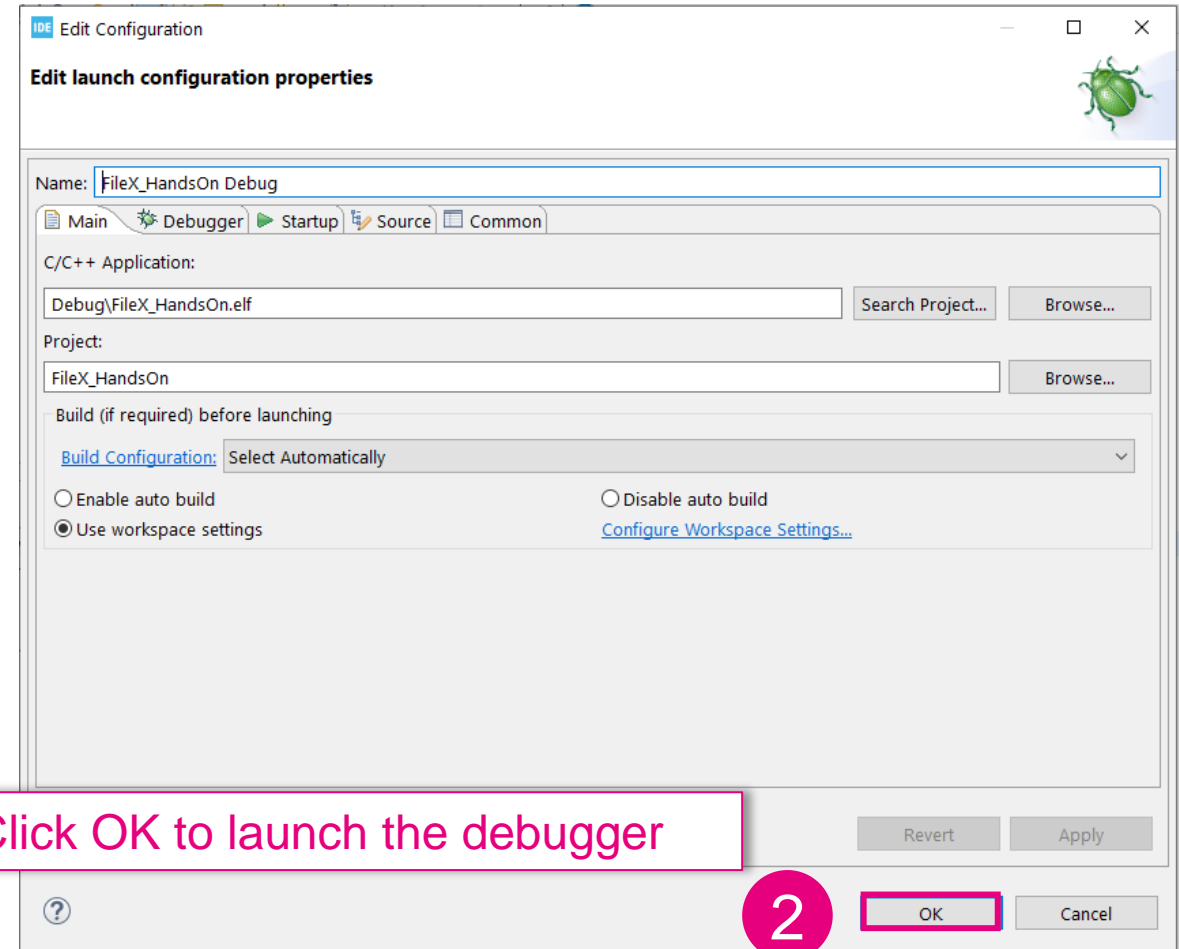


Click on debug  icon
or (F11) to start the debugger

Debug Configuration Window will pop-up



- Default settings are OK for STLink



Click OK to launch the debugger

Debugging the FileX Code

HandsOn - FileX_HandsOn/Core/Src/main.c - STM32CubeIDE

File Edit Source Refactor Navigate Search Project Run Window Help

Debug Project Explorer

<terminated>FileX_HandsOn Debug [STM32 Cortex-M C/C++ Application]
<terminated, exit value: 0>arm-none-eabi-gdb (8.3.1.20191211)
<terminated, exit value: -2147418112>ST-LINK (ST-LINK GDB server)

FileX_HandsOn Debug [STM32 Cortex-M C/C++ Application]
FileX_HandsOn.elf [cores: 0]
Thread #1 [main] 1 [core: 0] (Suspended : Breakpoint)
main() at main.c:102 0x80008da
arm-none-eabi-gdb (8.3.1.20191211)
ST-LINK (ST-LINK GDB server)

main.c app_filex.c startup_stm32h723zgtx.s

```
90 * @brief The application entry point.  
91 * @retval int  
92 */  
93 int main(void)  
94 {  
95 /* USER CODE BEGIN 1 */  
96  
97 /* USER CODE END 1 */  
98  
99 /* MCU Configuration-----*/  
100  
101 /* Reset of all peripherals, Initializes the Flash interface and the Systick timer.  
102 HAL_Init();  
103  
104 /* USER CODE BEGIN 2 */  
105  
106 /* USER CODE END 2 */  
107  
108 /* Configure the System Clock, External Interrupts and NVIC.  
109 SystemClock_Config();  
110  
111 /* USER CODE BEGIN 3 */  
112  
113 /* USER CODE END 3 */  
114  
115 /* Initialize the hardware, make here all hardware initialization  
116 HAL_Init();  
117  
118 ETH_Init();  
119 MX_USART3_UART_Init();  
120 MX_USB_OTG_HS_USB_Init();  
121 /* USER CODE BEGIN 4 */  
122  
123 /* USER CODE END 4 */  
124  
125 /* USER CODE BEGIN 5 */  
126  
127 /* USER CODE END 5 */  
128  
129 /* USER CODE BEGIN 6 */  
130  
131 /* USER CODE END 6 */  
132  
133 /* USER CODE BEGIN 7 */  
134  
135 /* USER CODE END 7 */  
136  
137 /* USER CODE BEGIN 8 */  
138  
139 /* USER CODE END 8 */  
140  
141 /* USER CODE BEGIN 9 */  
142  
143 /* USER CODE END 9 */  
144  
145 /* USER CODE BEGIN 10 */  
146  
147 /* USER CODE END 10 */  
148  
149 /* USER CODE BEGIN 11 */  
150  
151 /* USER CODE END 11 */  
152  
153 /* USER CODE BEGIN 12 */  
154  
155 /* USER CODE END 12 */  
156  
157 /* USER CODE BEGIN 13 */  
158  
159 /* USER CODE END 13 */  
160  
161 /* USER CODE BEGIN 14 */  
162  
163 /* USER CODE END 14 */  
164  
165 /* USER CODE BEGIN 15 */  
166  
167 /* USER CODE END 15 */  
168  
169 /* USER CODE BEGIN 16 */  
170  
171 /* USER CODE END 16 */  
172  
173 /* USER CODE BEGIN 17 */  
174  
175 /* USER CODE END 17 */  
176  
177 /* USER CODE BEGIN 18 */  
178  
179 /* USER CODE END 18 */  
180  
181 /* USER CODE BEGIN 19 */  
182  
183 /* USER CODE END 19 */  
184  
185 /* USER CODE BEGIN 20 */  
186  
187 /* USER CODE END 20 */  
188  
189 /* USER CODE BEGIN 21 */  
190  
191 /* USER CODE END 21 */  
192  
193 /* USER CODE BEGIN 22 */  
194  
195 /* USER CODE END 22 */  
196  
197 /* USER CODE BEGIN 23 */  
198  
199 /* USER CODE END 23 */  
200  
201 /* USER CODE BEGIN 24 */  
202  
203 /* USER CODE END 24 */  
204  
205 /* USER CODE BEGIN 25 */  
206  
207 /* USER CODE END 25 */  
208  
209 /* USER CODE BEGIN 26 */  
210  
211 /* USER CODE END 26 */  
212  
213 /* USER CODE BEGIN 27 */  
214  
215 /* USER CODE END 27 */  
216  
217 /* USER CODE BEGIN 28 */  
218  
219 /* USER CODE END 28 */  
220  
221 /* USER CODE BEGIN 29 */  
222  
223 /* USER CODE END 29 */  
224  
225 /* USER CODE BEGIN 30 */  
226  
227 /* USER CODE END 30 */  
228  
229 /* USER CODE BEGIN 31 */  
230  
231 /* USER CODE END 31 */  
232  
233 /* USER CODE BEGIN 32 */  
234  
235 /* USER CODE END 32 */  
236  
237 /* USER CODE BEGIN 33 */  
238  
239 /* USER CODE END 33 */  
240  
241 /* USER CODE BEGIN 34 */  
242  
243 /* USER CODE END 34 */  
244  
245 /* USER CODE BEGIN 35 */  
246  
247 /* USER CODE END 35 */  
248  
249 /* USER CODE BEGIN 36 */  
250  
251 /* USER CODE END 36 */  
252  
253 /* USER CODE BEGIN 37 */  
254  
255 /* USER CODE END 37 */  
256  
257 /* USER CODE BEGIN 38 */  
258  
259 /* USER CODE END 38 */  
260  
261 /* USER CODE BEGIN 39 */  
262  
263 /* USER CODE END 39 */  
264  
265 /* USER CODE BEGIN 40 */  
266  
267 /* USER CODE END 40 */  
268  
269 /* USER CODE BEGIN 41 */  
270  
271 /* USER CODE END 41 */  
272  
273 /* USER CODE BEGIN 42 */  
274  
275 /* USER CODE END 42 */  
276  
277 /* USER CODE BEGIN 43 */  
278  
279 /* USER CODE END 43 */  
280  
281 /* USER CODE BEGIN 44 */  
282  
283 /* USER CODE END 44 */  
284  
285 /* USER CODE BEGIN 45 */  
286  
287 /* USER CODE END 45 */  
288  
289 /* USER CODE BEGIN 46 */  
290  
291 /* USER CODE END 46 */  
292  
293 /* USER CODE BEGIN 47 */  
294  
295 /* USER CODE END 47 */  
296  
297 /* USER CODE BEGIN 48 */  
298  
299 /* USER CODE END 48 */  
300  
301 /* USER CODE BEGIN 49 */  
302  
303 /* USER CODE END 49 */  
304  
305 /* USER CODE BEGIN 50 */  
306  
307 /* USER CODE END 50 */  
308  
309 /* USER CODE BEGIN 51 */  
310  
311 /* USER CODE END 51 */  
312  
313 /* USER CODE BEGIN 52 */  
314  
315 /* USER CODE END 52 */  
316  
317 /* USER CODE BEGIN 53 */  
318  
319 /* USER CODE END 53 */  
320  
321 /* USER CODE BEGIN 54 */  
322  
323 /* USER CODE END 54 */  
324  
325 /* USER CODE BEGIN 55 */  
326  
327 /* USER CODE END 55 */  
328  
329 /* USER CODE BEGIN 56 */  
330  
331 /* USER CODE END 56 */  
332  
333 /* USER CODE BEGIN 57 */  
334  
335 /* USER CODE END 57 */  
336  
337 /* USER CODE BEGIN 58 */  
338  
339 /* USER CODE END 58 */  
340  
341 /* USER CODE BEGIN 59 */  
342  
343 /* USER CODE END 59 */  
344  
345 /* USER CODE BEGIN 60 */  
346  
347 /* USER CODE END 60 */  
348  
349 /* USER CODE BEGIN 61 */  
350  
351 /* USER CODE END 61 */  
352  
353 /* USER CODE BEGIN 62 */  
354  
355 /* USER CODE END 62 */  
356  
357 /* USER CODE BEGIN 63 */  
358  
359 /* USER CODE END 63 */  
360  
361 /* USER CODE BEGIN 64 */  
362  
363 /* USER CODE END 64 */  
364  
365 /* USER CODE BEGIN 65 */  
366  
367 /* USER CODE END 65 */  
368  
369 /* USER CODE BEGIN 66 */  
370  
371 /* USER CODE END 66 */  
372  
373 /* USER CODE BEGIN 67 */  
374  
375 /* USER CODE END 67 */  
376  
377 /* USER CODE BEGIN 68 */  
378  
379 /* USER CODE END 68 */  
380  
381 /* USER CODE BEGIN 69 */  
382  
383 /* USER CODE END 69 */  
384  
385 /* USER CODE BEGIN 70 */  
386  
387 /* USER CODE END 70 */  
388  
389 /* USER CODE BEGIN 71 */  
390  
391 /* USER CODE END 71 */  
392  
393 /* USER CODE BEGIN 72 */  
394  
395 /* USER CODE END 72 */  
396  
397 /* USER CODE BEGIN 73 */  
398  
399 /* USER CODE END 73 */  
400  
401 /* USER CODE BEGIN 74 */  
402  
403 /* USER CODE END 74 */  
404  
405 /* USER CODE BEGIN 75 */  
406  
407 /* USER CODE END 75 */  
408  
409 /* USER CODE BEGIN 76 */  
410  
411 /* USER CODE END 76 */  
412  
413 /* USER CODE BEGIN 77 */  
414  
415 /* USER CODE END 77 */  
416  
417 /* USER CODE BEGIN 78 */  
418  
419 /* USER CODE END 78 */  
420  
421 /* USER CODE BEGIN 79 */  
422  
423 /* USER CODE END 79 */  
424  
425 /* USER CODE BEGIN 80 */  
426  
427 /* USER CODE END 80 */  
428  
429 /* USER CODE BEGIN 81 */  
430  
431 /* USER CODE END 81 */  
432  
433 /* USER CODE BEGIN 82 */  
434  
435 /* USER CODE END 82 */  
436  
437 /* USER CODE BEGIN 83 */  
438  
439 /* USER CODE END 83 */  
440  
441 /* USER CODE BEGIN 84 */  
442  
443 /* USER CODE END 84 */  
444  
445 /* USER CODE BEGIN 85 */  
446  
447 /* USER CODE END 85 */  
448  
449 /* USER CODE BEGIN 86 */  
450  
451 /* USER CODE END 86 */  
452  
453 /* USER CODE BEGIN 87 */  
454  
455 /* USER CODE END 87 */  
456  
457 /* USER CODE BEGIN 88 */  
458  
459 /* USER CODE END 88 */  
460  
461 /* USER CODE BEGIN 89 */  
462  
463 /* USER CODE END 89 */  
464  
465 /* USER CODE BEGIN 90 */  
466  
467 /* USER CODE END 90 */  
468  
469 /* USER CODE BEGIN 91 */  
470  
471 /* USER CODE END 91 */  
472  
473 /* USER CODE BEGIN 92 */  
474  
475 /* USER CODE END 92 */  
476  
477 /* USER CODE BEGIN 93 */  
478  
479 /* USER CODE END 93 */  
480  
481 /* USER CODE BEGIN 94 */  
482  
483 /* USER CODE END 94 */  
484  
485 /* USER CODE BEGIN 95 */  
486  
487 /* USER CODE END 95 */  
488  
489 /* USER CODE BEGIN 96 */  
490  
491 /* USER CODE END 96 */  
492  
493 /* USER CODE BEGIN 97 */  
494  
495 /* USER CODE END 97 */  
496  
497 /* USER CODE BEGIN 98 */  
498  
499 /* USER CODE END 98 */  
500  
501 /* USER CODE BEGIN 99 */  
502  
503 /* USER CODE END 99 */  
504  
505 /* USER CODE BEGIN 100 */  
506  
507 /* USER CODE END 100 */  
508  
509 /* USER CODE BEGIN 101 */  
510  
511 /* USER CODE END 101 */  
512  
513 /* USER CODE BEGIN 102 */  
514  
515 /* USER CODE END 102 */  
516  
517 /* USER CODE BEGIN 103 */  
518  
519 /* USER CODE END 103 */  
520  
521 /* USER CODE BEGIN 104 */  
522  
523 /* USER CODE END 104 */  
524  
525 /* USER CODE BEGIN 105 */  
526  
527 /* USER CODE END 105 */  
528  
529 /* USER CODE BEGIN 106 */  
530  
531 /* USER CODE END 106 */  
532  
533 /* USER CODE BEGIN 107 */  
534  
535 /* USER CODE END 107 */  
536  
537 /* USER CODE BEGIN 108 */  
538  
539 /* USER CODE END 108 */  
540  
541 /* USER CODE BEGIN 109 */  
542  
543 /* USER CODE END 109 */  
544  
545 /* USER CODE BEGIN 110 */  
546  
547 /* USER CODE END 110 */  
548  
549 /* USER CODE BEGIN 111 */  
550  
551 /* USER CODE END 111 */  
552  
553 /* USER CODE BEGIN 112 */  
554  
555 /* USER CODE END 112 */  
556  
557 /* USER CODE BEGIN 113 */  
558  
559 /* USER CODE END 113 */  
560  
561 /* USER CODE BEGIN 114 */  
562  
563 /* USER CODE END 114 */  
564  
565 /* USER CODE BEGIN 115 */  
566  
567 /* USER CODE END 115 */  
568  
569 /* USER CODE BEGIN 116 */  
570  
571 /* USER CODE END 116 */  
572  
573 /* USER CODE BEGIN 117 */  
574  
575 /* USER CODE END 117 */  
576  
577 /* USER CODE BEGIN 118 */  
578  
579 /* USER CODE END 118 */  
580  
581 /* USER CODE BEGIN 119 */  
582  
583 /* USER CODE END 119 */  
584  
585 /* USER CODE BEGIN 120 */  
586  
587 /* USER CODE END 120 */  
588  
589 /* USER CODE BEGIN 121 */  
590  
591 /* USER CODE END 121 */  
592  
593 /* USER CODE BEGIN 122 */  
594  
595 /* USER CODE END 122 */  
596  
597 /* USER CODE BEGIN 123 */  
598  
599 /* USER CODE END 123 */  
600  
601 /* USER CODE BEGIN 124 */  
602  
603 /* USER CODE END 124 */  
604  
605 /* USER CODE BEGIN 125 */  
606  
607 /* USER CODE END 125 */  
608  
609 /* USER CODE BEGIN 126 */  
610  
611 /* USER CODE END 126 */  
612  
613 /* USER CODE BEGIN 127 */  
614  
615 /* USER CODE END 127 */  
616  
617 /* USER CODE BEGIN 128 */  
618  
619 /* USER CODE END 128 */  
620  
621 /* USER CODE BEGIN 129 */  
622  
623 /* USER CODE END 129 */  
624  
625 /* USER CODE BEGIN 130 */  
626  
627 /* USER CODE END 130 */  
628  
629 /* USER CODE BEGIN 131 */  
630  
631 /* USER CODE END 131 */  
632  
633 /* USER CODE BEGIN 132 */  
634  
635 /* USER CODE END 132 */  
636  
637 /* USER CODE BEGIN 133 */  
638  
639 /* USER CODE END 133 */  
640  
641 /* USER CODE BEGIN 134 */  
642  
643 /* USER CODE END 134 */  
644  
645 /* USER CODE BEGIN 135 */  
646  
647 /* USER CODE END 135 */  
648  
649 /* USER CODE BEGIN 136 */  
650  
651 /* USER CODE END 136 */  
652  
653 /* USER CODE BEGIN 137 */  
654  
655 /* USER CODE END 137 */  
656  
657 /* USER CODE BEGIN 138 */  
658  
659 /* USER CODE END 138 */  
660  
661 /* USER CODE BEGIN 139 */  
662  
663 /* USER CODE END 139 */  
664  
665 /* USER CODE BEGIN 140 */  
666  
667 /* USER CODE END 140 */  
668  
669 /* USER CODE BEGIN 141 */  
670  
671 /* USER CODE END 141 */  
672  
673 /* USER CODE BEGIN 142 */  
674  
675 /* USER CODE END 142 */  
676  
677 /* USER CODE BEGIN 143 */  
678  
679 /* USER CODE END 143 */  
680  
681 /* USER CODE BEGIN 144 */  
682  
683 /* USER CODE END 144 */  
684  
685 /* USER CODE BEGIN 145 */  
686  
687 /* USER CODE END 145 */  
688  
689 /* USER CODE BEGIN 146 */  
690  
691 /* USER CODE END 146 */  
692  
693 /* USER CODE BEGIN 147 */  
694  
695 /* USER CODE END 147 */  
696  
697 /* USER CODE BEGIN 148 */  
698  
699 /* USER CODE END 148 */  
700  
701 /* USER CODE BEGIN 149 */  
702  
703 /* USER CODE END 149 */  
704  
705 /* USER CODE BEGIN 150 */  
706  
707 /* USER CODE END 150 */  
708  
709 /* USER CODE BEGIN 151 */  
710  
711 /* USER CODE END 151 */  
712  
713 /* USER CODE BEGIN 152 */  
714  
715 /* USER CODE END 152 */  
716  
717 /* USER CODE BEGIN 153 */  
718  
719 /* USER CODE END 153 */  
720  
721 /* USER CODE BEGIN 154 */  
722  
723 /* USER CODE END 154 */  
724  
725 /* USER CODE BEGIN 155 */  
726  
727 /* USER CODE END 155 */  
728  
729 /* USER CODE BEGIN 156 */  
730  
731 /* USER CODE END 156 */  
732  
733 /* USER CODE BEGIN 157 */  
734  
735 /* USER CODE END 157 */  
736  
737 /* USER CODE BEGIN 158 */  
738  
739 /* USER CODE END 158 */  
740  
741 /* USER CODE BEGIN 159 */  
742  
743 /* USER CODE END 159 */  
744  
745 /* USER CODE BEGIN 160 */  
746  
747 /* USER CODE END 160 */  
748  
749 /* USER CODE BEGIN 161 */  
750  
751 /* USER CODE END 161 */  
752  
753 /* USER CODE BEGIN 162 */  
754  
755 /* USER CODE END 162 */  
756  
757 /* USER CODE BEGIN 163 */  
758  
759 /* USER CODE END 163 */  
760  
761 /* USER CODE BEGIN 164 */  
762  
763 /* USER CODE END 164 */  
764  
765 /* USER CODE BEGIN 165 */  
766  
767 /* USER CODE END 165 */  
768  
769 /* USER CODE BEGIN 166 */  
770  
771 /* USER CODE END 166 */  
772  
773 /* USER CODE BEGIN 167 */  
774  
775 /* USER CODE END 167 */  
776  
777 /* USER CODE BEGIN 168 */  
778  
779 /* USER CODE END 168 */  
780  
781 /* USER CODE BEGIN 169 */  
782  
783 /* USER CODE END 169 */  
784  
785 /* USER CODE BEGIN 170 */  
786  
787 /* USER CODE END 170 */  
788  
789 /* USER CODE BEGIN 171 */  
790  
791 /* USER CODE END 171 */  
792  
793 /* USER CODE BEGIN 172 */  
794  
795 /* USER CODE END 172 */  
796  
797 /* USER CODE BEGIN 173 */  
798  
799 /* USER CODE END 173 */  
800  
801 /* USER CODE BEGIN 174 */  
802  
803 /* USER CODE END 174 */  
804  
805 /* USER CODE BEGIN 175 */  
806  
807 /* USER CODE END 175 */  
808  
809 /* USER CODE BEGIN 176 */  
810  
811 /* USER CODE END 176 */  
812  
813 /* USER CODE BEGIN 177 */  
814  
815 /* USER CODE END 177 */  
816  
817 /* USER CODE BEGIN 178 */  
818  
819 /* USER CODE END 178 */  
820  
821 /* USER CODE BEGIN 179 */  
822  
823 /* USER CODE END 179 */  
824  
825 /* USER CODE BEGIN 180 */  
826  
827 /* USER CODE END 180 */  
828  
829 /* USER CODE BEGIN 181 */  
830  
831 /* USER CODE END 181 */  
832  
833 /* USER CODE BEGIN 182 */  
834  
835 /* USER CODE END 182 */  
836  
837 /* USER CODE BEGIN 183 */  
838  
839 /* USER CODE END 183 */  
840  
841 /* USER CODE BEGIN 184 */  
842  
843 /* USER CODE END 184 */  
844  
845 /* USER CODE BEGIN 185 */  
846  
847 /* USER CODE END 185 */  
848  
849 /* USER CODE BEGIN 186 */  
850  
851 /* USER CODE END 186 */  
852  
853 /* USER CODE BEGIN 187 */  
854  
855 /* USER CODE END 187 */  
856  
857 /* USER CODE BEGIN 188 */  
858  
859 /* USER CODE END 188 */  
860  
861 /* USER CODE BEGIN 189 */  
862  
863 /* USER CODE END 189 */  
864  
865 /* USER CODE BEGIN 190 */  
866  
867 /* USER CODE END 190 */  
868  
869 /* USER CODE BEGIN 191 */  
870  
871 /* USER CODE END 191 */  
872  
873 /* USER CODE BEGIN 192 */  
874  
875 /* USER CODE END 192 */  
876  
877 /* USER CODE BEGIN 193 */  
878  
879 /* USER CODE END 193 */  
880  
881 /* USER CODE BEGIN 194 */  
882  
883 /* USER CODE END 194 */  
884  
885 /* USER CODE BEGIN 195 */  
886  
887 /* USER CODE END 195 */  
888  
889 /* USER CODE BEGIN 196 */  
890  
891 /* USER CODE END 196 */  
892  
893 /* USER CODE BEGIN 197 */  
894  
895 /* USER CODE END 197 */  
896  
897 /* USER CODE BEGIN 198 */  
898  
899 /* USER CODE END 198 */  
900  
901 /* USER CODE BEGIN 199 */  
902  
903 /* USER CODE END 199 */  
904  
905 /* USER CODE BEGIN 200 */  
906  
907 /* USER CODE END 200 */  
908  
909 /* USER CODE BEGIN 201 */  
910  
911 /* USER CODE END 201 */  
912  
913 /* USER CODE BEGIN 202 */  
914  
915 /* USER CODE END 202 */  
916  
917 /* USER CODE BEGIN 203 */  
918  
919 /* USER CODE END 203 */  
920  
921 /* USER CODE BEGIN 204 */  
922  
923 /* USER CODE END 204 */  
924  
925 /* USER CODE BEGIN 205 */  
926  
927 /* USER CODE END 205 */  
928  
929 /* USER CODE BEGIN 206 */  
930  
931 /* USER CODE END 206 */  
932  
933 /* USER CODE BEGIN 207 */  
934  
935 /* USER CODE END 207 */  
936  
937 /* USER CODE BEGIN 208 */  
938  
939 /* USER CODE END 208 */  
940  
941 /* USER CODE BEGIN 209 */  
942  
943 /* USER CODE END 209 */  
944  
945 /* USER CODE BEGIN 210 */  
946  
947 /* USER CODE END 210 */  
948  
949 /* USER CODE BEGIN 211 */  
950  
951 /* USER CODE END 211 */  
952  
953 /* USER CODE BEGIN 212 */  
954  
955 /* USER CODE END 212 */  
956  
957 /* USER CODE BEGIN 213 */  
958  
959 /* USER CODE END 213 */  
960  
961 /* USER CODE BEGIN 214 */  
962  
963 /* USER CODE END 214 */  
964  
965 /* USER CODE BEGIN 215 */  
966  
967 /* USER CODE END 215 */  
968  
969 /* USER CODE BEGIN 216 */  
970  
971 /* USER CODE END 216 */  
972  
973 /* USER CODE BEGIN 217 */  
974  
975 /* USER CODE END 217 */  
976  
977 /* USER CODE BEGIN 218 */  
978  
979 /* USER CODE END 218 */  
980  
981 /* USER CODE BEGIN 219 */  
982  
983 /* USER CODE END 219 */  
984  
985 /* USER CODE BEGIN 220 */  
986  
987 /* USER CODE END 220 */  
988  
989 /* USER CODE BEGIN 221 */  
990  
991 /* USER CODE END 221 */  
992  
993 /* USER CODE BEGIN 222 */  
994  
995 /* USER CODE END 222 */  
996  
997 /* USER CODE BEGIN 223 */  
998  
999 /* USER CODE END 223 */  
1000  
1001 /* USER CODE BEGIN 224 */  
1002  
1003 /* USER CODE END 224 */  
1004  
1005 /* USER CODE BEGIN 225 */  
1006  
1007 /* USER CODE END 225 */  
1008  
1009 /* USER CODE BEGIN 226 */  
1010  
1011 /* USER CODE END 226 */  
1012  
1013 /* USER CODE BEGIN 227 */  
1014  
1015 /* USER CODE END 227 */  
1016  
1017 /* USER CODE BEGIN 228 */  
1018  
1019 /* USER CODE END 228 */  
1020  
1021 /* USER CODE BEGIN 229 */  
1022  
1023 /* USER CODE END 229 */  
1024  
1025 /* USER CODE BEGIN 230 */  
1026  
1027 /* USER CODE END 230 */  
1028  
1029 /* USER CODE BEGIN 231 */  
1030  
1031 /* USER CODE END 231 */  
1032  
1033 /* USER CODE BEGIN 232 */  
1034  
1035 /* USER CODE END 232 */  
1036  
1037 /* USER CODE BEGIN 233 */  
1038  
1039 /* USER CODE END 233 */  
1040  
1041 /* USER CODE BEGIN 234 */  
1042  
1043 /* USER CODE END 234 */  
1044  
1045 /* USER CODE BEGIN 235 */  
1046  
1047 /* USER CODE END 235 */  
1048  
1049 /* USER CODE BEGIN 236 */  
1050  
1051 /* USER CODE END 236 */  
1052  
1053 /* USER CODE BEGIN 237 */  
1054  
1055 /* USER CODE END 237 */  
1056  
1057 /* USER CODE BEGIN 238 */  
1058  
1059 /* USER CODE END 238 */  
1060  
1061 /* USER CODE BEGIN 239 */  
1062  
1063 /* USER CODE END 239 */  
1064  
1065 /* USER CODE BEGIN 240 */  
1066  
1067 /* USER CODE END 240 */  
1068  
1069 /* USER CODE BEGIN 241 */  
1070  
1071 /* USER CODE END 241 */  
1072  
1073 /* USER CODE BEGIN 242 */  
1074  
1075 /* USER CODE END 242 */  
1076  
1077 /* USER CODE BEGIN 243 */  
1078  
1079 /* USER CODE END 243 */  
1080  
1081 /* USER CODE BEGIN 244 */  
1082  
1083 /* USER CODE END 244 */  
1084  
1085 /* USER CODE BEGIN 245 */  
1086  
1087 /* USER CODE END 245 */  
1088  
1089 /* USER CODE BEGIN 246 */  
1090  
1091 /* USER CODE END 246 */  
1092  
1093 /* USER CODE BEGIN 247 */  
1094  
1095 /* USER CODE END 247 */  
1096  
1097 /* USER CODE BEGIN 248 */  
1098  
1099 /* USER CODE END 248 */  
1100  
1101 /* USER CODE BEGIN 249 */  
1102  
1103 /* USER CODE END 249 */  
1104  
1105 /* USER CODE BEGIN 250 */  
1106  
1107 /* USER CODE END 250 */  
1108  
1109 /* USER CODE BEGIN 251 */  
1110  
1111 /* USER CODE END 251 */  
1112  
1113 /* USER CODE BEGIN 252 */  
1114  
1115 /* USER CODE END 252 */  
1116  
1117 /* USER CODE BEGIN 253 */  
1118  
1119 /* USER CODE END 253 */  
1120  
1121 /* USER CODE BEGIN 254 */  
1122  
1123 /* USER CODE END 254 */  
1124  
1125 /* USER CODE BEGIN 255 */  
1126  
1127 /* USER CODE END 255 */  
1128  
1129 /* USER CODE BEGIN 256 */  
1130  
1131 /* USER CODE END 256 */  
1132  
1133 /* USER CODE BEGIN 257 */  
1134  
1135 /* USER CODE END 257 */  
1136  
1137 /* USER CODE BEGIN 258 */  
1138  
1139 /* USER CODE END 258 */  
1140  
1141 /* USER CODE BEGIN 259 */  
1142  
1143 /* USER CODE END 259 */  
1144  
1145 /* USER CODE BEGIN 260 */  
1146  
1147 /* USER CODE END 260 */  
1148  
1149 /* USER CODE BEGIN 261 */  
1150  
1151 /* USER CODE END 261 */  
1152  
1153 /* USER CODE BEGIN 262 */  
1154  
1155 /* USER CODE END 262 */  
1156  
1157 /* USER CODE BEGIN 263 */  
1158  
1159 /* USER CODE END 263 */  
1160  
1161 /* USER CODE BEGIN 264 */  
1162  
1163 /* USER CODE END 264 */  
1164  
1165 /* USER CODE BEGIN 265 */  
1166  
1167 /* USER CODE END 265 */  
1168  
1169 /* USER CODE BEGIN 266 */  
1170  
1171 /* USER CODE END 266 */  
1172  
1173 /* USER CODE BEGIN 267 */  
1174  
1175 /* USER CODE END 267 */  
1176  
1177 /* USER CODE BEGIN 268 */  
1178  
1179 /* USER CODE END 268 */  
1180  
1181 /* USER CODE BEGIN 269 */  
1182  
1183 /* USER CODE END 269 */  
1184  
1185 /* USER CODE BEGIN 270 */  
1186  
1187 /* USER CODE END 270 */  
1188  
1189 /* USER CODE BEGIN 271 */  
1190  
1191 /* USER CODE END 271 */  
1192  
1193 /* USER CODE BEGIN 272 */  
1194  
1195 /* USER CODE END 272 */  
1196  
1197 /* USER CODE BEGIN 273 */  
1198  
1199 /* USER CODE END 273 */  
1200  
1201 /* USER CODE BEGIN 274 */  
1202  
1203 /* USER CODE END 274 */  
1204  
1205 /* USER CODE BEGIN 275 */  
1206  
1207 /* USER CODE END 275 */  
1208  
1209 /* USER CODE BEGIN 276 */  
1210  
1211 /* USER CODE END 276 */  
1212  
1213 /* USER CODE BEGIN 277 */  
1214  
1215 /* USER CODE END 277 */  
1216  
1217 /* USER CODE BEGIN 278 */  
1218  
1219 /* USER CODE END 278 */  
1220  
1221 /* USER CODE BEGIN 279 */  
1222  
1223 /* USER CODE END 279 */  
1224  
1225 /* USER CODE BEGIN 280 */  
1226  
1227 /* USER CODE END 280 */  
1228  
1229 /* USER CODE BEGIN 281 */  
1230  
1231 /* USER CODE END 281 */  
1232  
1233 /* USER CODE BEGIN 282 */  
1234  
1235 /* USER CODE END 282 */  
1236  
1237 /* USER CODE BEGIN 283 */  
1238  
1239 /* USER CODE END 283 */  
1240  
1241 /* USER CODE BEGIN 284 */  
1242  
1243 /* USER CODE END 284 */  
1244  
1245 /* USER CODE BEGIN 285 */  
1246  
1247 /* USER CODE END 285 */  
1248  
1249 /* USER CODE BEGIN 286 */  
1250  
1251 /* USER CODE END 286 */  
1252  
1253 /* USER CODE BEGIN 287 */  
1254  
1255 /* USER CODE END 287 */  
1256  
1257 /* USER CODE BEGIN 288 */  
1258  
1259 /* USER CODE END 288 */  
1260  
1261 /* USER CODE BEGIN 289 */  
1262  
1263 /* USER CODE END 289
```

Debugging the FileX Code

```
101  /* Reset of all peripherals, Initializes the Flash interface and the Systick. */
102  HAL_Init();
103
104  /* USER CODE BEGIN Init */
105
106  /* USER CODE END Init */
107
108  /* Configure the system clock */
109  SystemClock_Config();
110
111  /* USER CODE BEGIN SysInit */
112
113  /* USER CODE END SysInit */
114
115  /* Initialize all configured peripherals */
116  MX_GPIO_Init();
117  MX_ETH_Init();
118  MX_USART3_UART_Init();
119  MX_USB_OTG_HS_USB_Init();
120  /* USER CODE BEGIN 2 */
121
122  /* USER CODE END 2 */
123
124  MX_ThreadX_Init();
125  /* Infinite loop */
126  /* USER CODE BEGIN WHILE */
```

Click on “New Renderings”

1



2

Double click on “Traditional” to view the data in the memory as string

Debugging the FileX Code

```
101 /* Reset of all peripherals, Initializes the Flash interface and the Systick. */
102 HAL_Init();
103
104 /* USER CODE BEGIN Init */
105
106 /* USER CODE END Init */
107
108 /* Configure the system clock */
109 SystemClock_Config();
110
111 /* USER CODE BEGIN SysInit */
112
113 /* USER CODE END SysInit */
114
115 /* Initialize all configured peripherals */
116 MX_GPIO_Init();
117 MX_ETH_Init();
118 MX_USART3_UART_Init();
119 MX_USB_OTG_HS_USB_Init();
120 /* USER CODE BEGIN 2 */
121
122 /* USER CODE END 2 */
123
124 MX_ThreadX_Init();
125 /* Infinite loop */
126 /* USER CODE BEGIN WHILE */
```

D1_AXISRAM2 Memory content starting with address 0x24020000, after reset

Console Problems Executables Debugger Console Memory ThreadX Thread List

Monitors

Address	Value
0x24020000	477ADE51 41D5EDFA 68120056 CA806143 8D8482CA DCCB7578 65242416 45112BE0 6282EF40
0x24020024	1B40C765 63748432 18802440 E137E3B5 9AD5469E 5D89020B 90198148 D54D5CB7 90C24DE8
0x24020048	E80D83CA B05DC44E FEC863FF F3FDFF71 4880E009 A5C80C88 20FFC965 FC3BCA66 DC72151C
0x2402006C	400C9006 FDFBEEEB D82CFC87 158BA5CF 228328C0 955F1AB5 8B2EAE9F C5618BF0 11DA1C08
0x24020090	574BBDBF 05FE7EBF 36198B88 2CE00076 F63057B7 783E5494 8850CDC3 1191C221 D19DDFBD
0x240200B4	7B5E6E36 1A173837 2A18120C 55DEBBF3 C54FBBFB A5315750 34489B3A FD9F5DA6 2A7F9EFF
0x240200D8	A3D06523 93320C4D DE2DEEEF DBCFD5DD 2C602581 6F07FE3B 59DE558F E0F5D773 8885E6A3
0x240200FC	836E0C04 1D6F3BF4 FDF262A3 832D26A2 4B946900 55EF8F7F 2D3C50A5 41811D7A A729A9A9

0x24020000 : 0x24020000 <Traditional> New Renderings...

0x24020000 QPzGúíÖAV..hCa.ÊÊ...xuËÜ.\$\$eà+.E@i.b
eÇ@.2.tc@\$.µā7á.ÏÖ...JH...VMÖèMÂ.
Ê..èNÄ]°ÿcÈpqqÿó.à.H..Èÿeÿÿ fÊ;ü..rÜ
...@ëiüÿ.ü,ØÏ¥..À(."µ._.°. .ø.aÂ..Ú.
¿%KW¿~p....6v.à..W0ö.T>xÄÍP. !Â. %ß. Ñ
6n^{78....*ó»PUÜ»OÄPW1¥:.H4!]ÿ ÿ..*
#eDEM.2.ÿi-pÿÖÏÜ.%",;p.o.UPYsxöâfæ..
...n.ô;o.Ébðÿç&-...i.K..ÿU¥P<-z..A00)\$

Debugging the FileX Code

HandsOn - FileX_HandsOn/FileX/App/app_filex.c - STM32CubeIDE

File Edit Source Refactor Navigate Search Project Run Window Help

Debug Project Explorer

FileX_HandsOn Debug [STM32 Cortex-M4] Application

FileX_HandsOn.elf [cores: 0]

Thread #1 [main] 1 [core: 0] (Suspended : Breakpoint)

main() at main.c:102 0x80008d10

arm-none-eabi-gdb (8.3.1.20191211)

ST-LINK (ST-LINK GDB server)

1 Open "app_filex.c"

2


3 Run to the breakpoint

Inside "fx_thread_entry()" function, around line 120, put a breakpoint on "fx_media_format()" function

```
104 /* Initialize FileX */
105 fx_system_initial
106 /* USER CODE END Init */
107 return ret;
108 }
109
110 /* USER CODE BEGIN 1 */
111 void fx_thread_entry(ULONG thread_input)
112 {
113     UINT status;
114     ULONG bytes_read;
115     ULONG available_space_post;
116     CHAR read_buffer[32];
117     CHAR data[] = "This is FileX working on STM32\n";
118
119     /* Format the NOR flash as FAT */
120     status = fx_media_format(
121         &ram_disk,
122         fx_stm32_sram_driver, // Driver entry
123         0, // Device info pointer
124         (UCHAR *) media_memory, // Media buffer pointer
125         DEFAULT_SECTOR_SIZE, // Media buffer size
126         "RAM_DISK", // Volume Name
127         1, // Number of FATs
128         32, // Directory Entries
129         0, // Hidden sectors
130         64, // Total sectors
131         DEFAULT_SECTOR_SIZE, // Sector size
132         8, // Sectors per cluster
133         1, // Heads
134         1 // Sectors per track
135     );
136
137     /* Check if the format status */
138     if (status != FX_SUCCESS)
139     {
140         Error_Handler();
141     }
142 }
```

Debugging the FileX Code

`fx_media_format ()` func. will format our memory where we are creating the file system.




```
119  /* Format the NOR flash as FAT */
120  status = fx_media_format(
121      &ram_disk,
122      fx_stm32_sram_driver,    // Driver entry
123      0,                      // Device info pointer
124      (UCHAR *) media_memory, // Media buffer pointer
125      DEFAULT_SECTOR_SIZE,    // Media buffer size
126      "RAM_DISK",             // Volume Name
127      1,                      // Number of FATs
128      32,                     // Directory Entries
129      0,                      // Hidden sectors
130      64,                     // Total sectors
131      DEFAULT_SECTOR_SIZE,    // Sector size
132      8,                      // Sectors per cluster
133      1,                      // Heads
134      1,                      // Sectors per track
135  );
136
```

“RAM_DISK” is the volume name of the ram disk where we will write the file name.

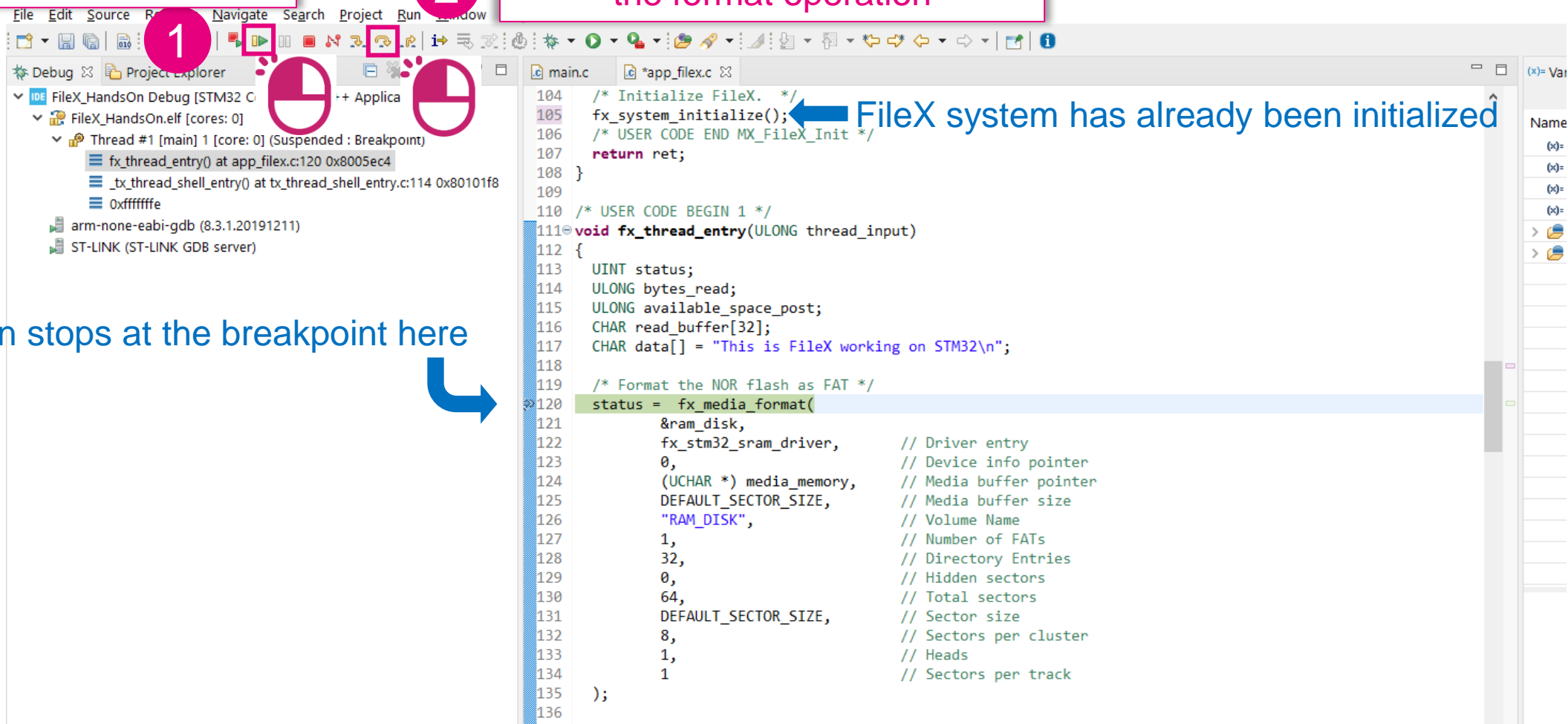
Debugging the FileX Code

Run to the breakpoint

Step Over  to execute the format operation

Execution stops at the breakpoint here

FileX system has already been initialized

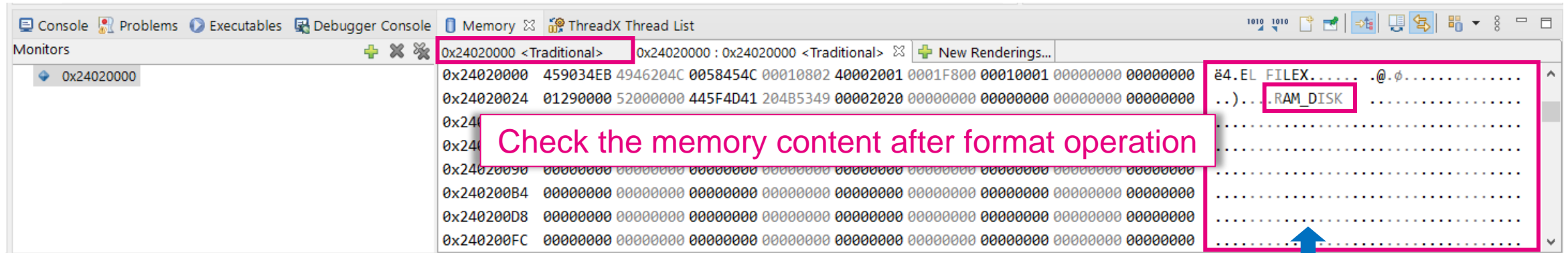


The screenshot shows the STM32CubeIDE debugger interface. The Project Explorer on the left displays the project structure, including the FileX_HandsOn Debug [STM32 C] project. The main code editor shows the FileX code, with the following functions visible:


```
104 /* Initialize FileX. */
105 fx_system_initialize();
106 /* USER CODE END MX_FileX_Init */
107 return ret;
108 }
109
110 /* USER CODE BEGIN 1 */
111 void fx_thread_entry(ULONG thread_input)
112 {
113     UINT status;
114     ULONG bytes_read;
115     ULONG available_space_post;
116     CHAR read_buffer[32];
117     CHAR data[] = "This is FileX working on STM32\n";
118
119     /* Format the NOR flash as FAT */
120     status = fx_media_format(
121         &ram_disk,
122         fx_stm32_sram_driver, // Driver entry
123         0, // Device info pointer
124         (UCHAR *) media_memory, // Media buffer pointer
125         DEFAULT_SECTOR_SIZE, // Media buffer size
126         "RAM_DISK", // Volume Name
127         1, // Number of FATs
128         32, // Directory Entries
129         0, // Hidden sectors
130         64, // Total sectors
131         DEFAULT_SECTOR_SIZE, // Sector size
132         8, // Sectors per cluster
133         1, // Heads
134         1 // Sectors per track
135     );
136 }
```

The execution has stopped at line 120, where the `fx_media_format` function is called. A blue arrow points from the text "Execution stops at the breakpoint here" to the breakpoint. Another blue arrow points from the text "FileX system has already been initialized" to the `fx_system_initialize()` call in the code.

Debugging the FileX Code



Volume "RAM_DISK" has been created after fx_media_format()

- So far, we have successfully formatted the media and created a FAT file system **on** our **ram disk**.
- Now we can keep stepping through the code by step over  and try to understand the main thread code:

Debugging the FileX Code

Step over  until

```
143  /* Open the RAM disk. */  
144  status = fx_media_open(&ram_disk, "RAM DISK", fx_stm32_sram_driver, 0, (VOID *) media_memory, DEFAULT_SECTOR_SIZE);  
145
```

fx_media_open() opens the ram disk media for the file access before creating the file.

Step over  until

```
152  /* Create a file called STM32.TXT in the root directory. */  
153  status = fx_file_create(&ram_disk, "FXTEST.TXT");  
154
```

fx_file_create() creates a file with the name “FXTEST.TXT” on the disk.

Debugging the FileX Code

Step over  until

```
167  /* Open the test file. */  
168  status = fx_file_open(&ram_disk, &fx_file, "FXTEST.TXT", FX_OPEN_FOR_WRITE);  
169
```

fx_file_open() opens the file for **write access**.

Step over  until

```
177  /* Seek to the beginning of the test file. */  
178  status = fx_file_seek(&fx_file, 0);  
179
```

fx_file_seek() seeks for the beginning of the file where we will write into

Debugging the FileX Code

Step over  until

```
187  /* Write a string to the test file. */
188  status = fx_file_write(&fx_file, data, sizeof(data));
189
```

Writes the data string into the file

[illegible]

Debugging the FileX Code

Step over  until

```
197  /* Close the test file. */  
198  status = fx_file_close(&fx_file);  
199
```

Finally, we need to close the file after completing the write operation.

- Now, let's check the created file on the ram disk and data written into this file

Debugging the FileX Code

The screenshot shows a debugger interface with two main panes. The left pane displays C code from `app_filex.c`. The right pane shows the 'Expressions' window with a table of variables. Below the code, the 'Memory' window displays a hex dump of memory starting at address `0x24020000`. A blue arrow points from a text box to the memory dump, and a pink box highlights a specific line in the dump.

```
186
187 /* Write a string to the test file. */
188 status = fx_file_write(&fx_file, data, sizeof(data));
189
190 /* Check the file write status. */
191 if (status != FX_SUCCESS)
192 {
193     /* Error writing to a file, call error handler. */
194     Error_Handler();
195 }
196
197 /* Close the test file. */
198 status = fx_file_close(&fx_file);
199
200 /* Check the file close status. */
201 if (status != FX_SUCCESS)
202 {
203     /* Error closing the file, call error handler. */
204     Error_Handler();
205 }
206
207 /* Open the test file. */
208 status = fx_file_open(&ram_disk, &fx_file, "FXTEST.TXT", FX_OPEN_FOR_READ);
```

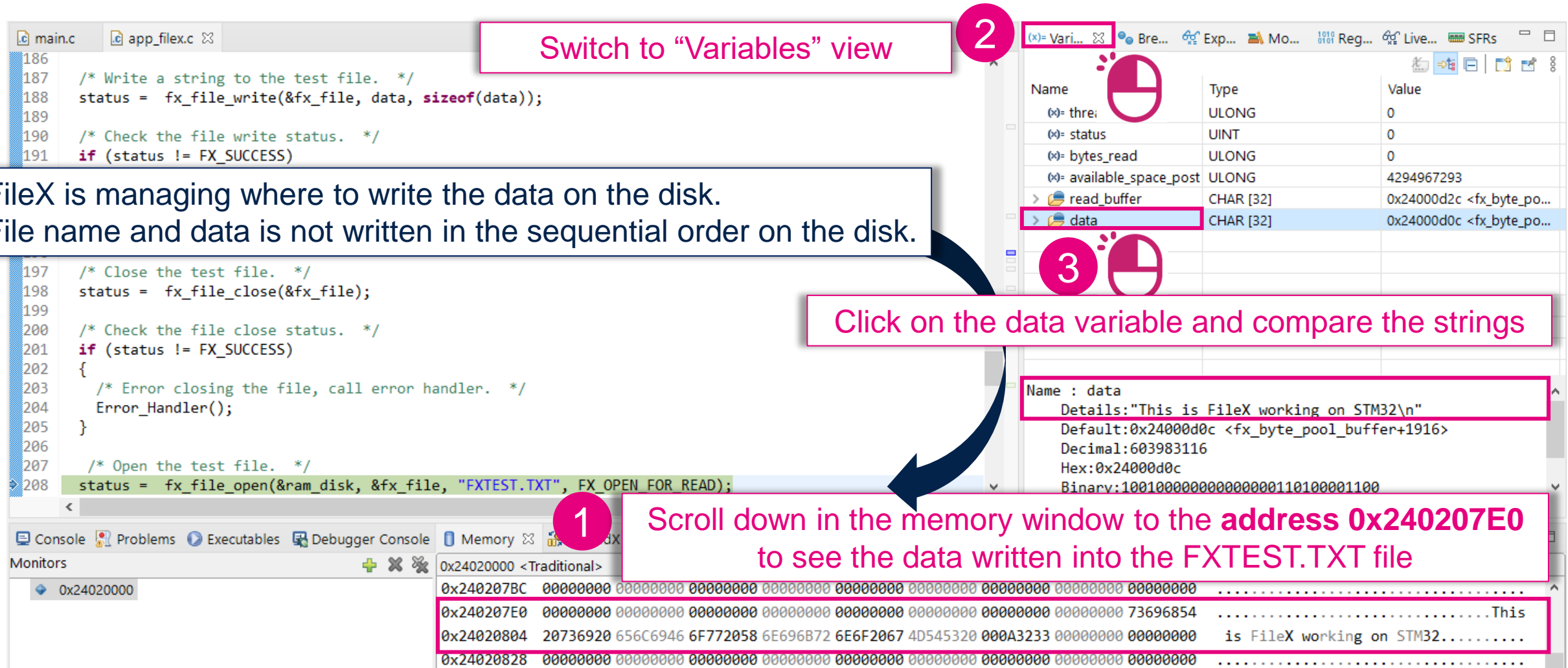
Expression	Type	Value
<code>_fx_version_id</code>	CHAR []	0x24000020 <_fx_version_...
+ Add new expression		

FileX is managing where to write the data on the disk

Scroll down in the memory window to the address **0x240203F0** to see the created file name

Address	Hex Data	ASCII Data
0x24020000	<Traditional>	
0x240203A8	00000000 00000000	
0x240203CC	00000000 00000000 00000000 00000000 00000000 00000000 00000000 00000000
0x240203F0	00000000 00000000 00000000 00000000 45545846 20205453 20545854 00000000 4A214A21FXTEST TXT!J!J
0x24020414	00000000 00004A21 00000000 00000000 00000000 00000000 00000000 00000000!J.....

Debugging the FileX Code



Debugging the FileX Code

Step over  until

```
207  /* Open the test file. */  
208  status = fx_file_open(&ram_disk, &fx_file, "FXTEST.TXT", FX_OPEN_FOR_READ);  
209
```

Opens the file for read access

Step over  until

```
217  /* Seek to the beginning of the test file. */  
218  status = fx_file_seek(&fx_file, 0);  
219
```


Seeks for the beginning of the file where we will read from

Debugging the FileX Code

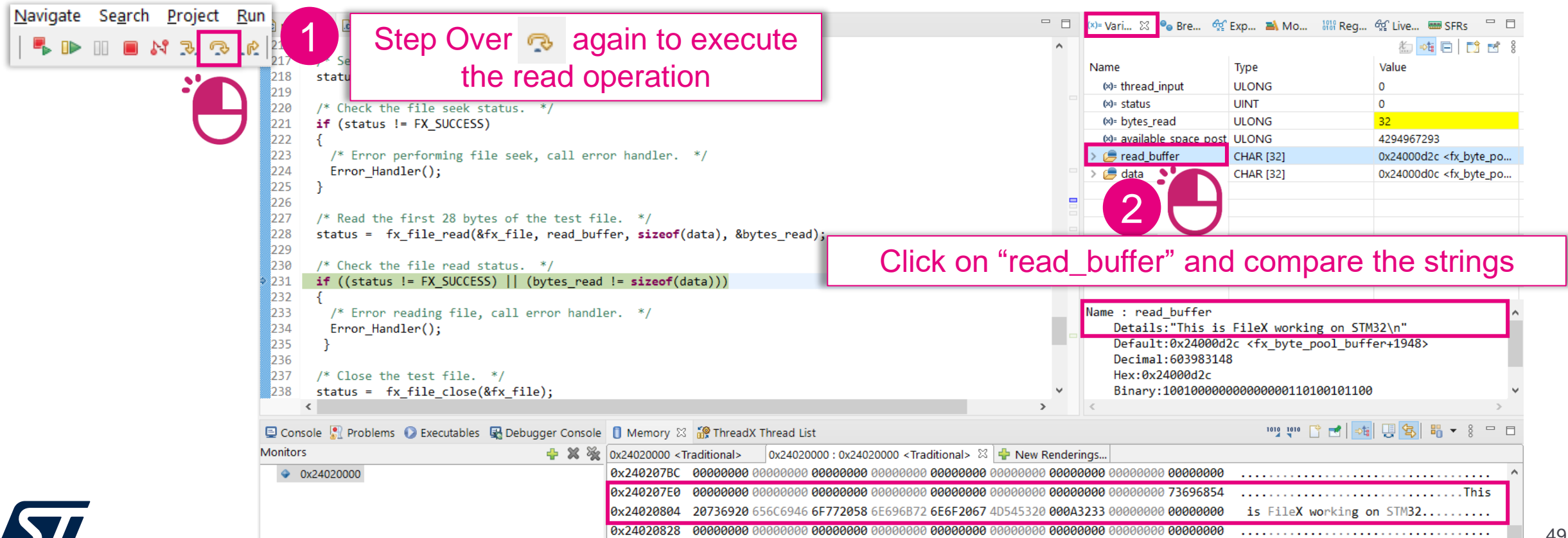
Step over  until

```
227 /* Read the first 28 bytes of the test file. */
228 status = fx_file_read(&fx_file, read_buffer, sizeof(data), &bytes_read);
229
```

Reads the data from the file into “read_buffer”

1 Step Over  again to execute the read operation

2 Click on “read_buffer” and compare the strings



Name	Type	Value
thread_input	ULONG	0
status	UINT	0
bytes_read	ULONG	32
available_space_post	ULONG	4294967293
read_buffer	CHAR [32]	0x24000d2c <fx_byte_po...
data	CHAR [32]	0x24000d0c <fx_byte_po...

Name : read_buffer
Details: "This is FileX working on STM32\n"
Default: 0x24000d2c <fx_byte_pool_buffer+1948>
Decimal: 603983148
Hex: 0x24000d2c
Binary: 10010000000000000000110100101100

Monitors

Address	Value
0x24020804	20736920 656C6946 6F772058 6E696B72 6E6F2067 40545320 000A3233 00000000 00000000 00000000

0x24020804 20736920 656C6946 6F772058 6E696B72 6E6F2067 40545320 000A3233 00000000 00000000 This is FileX working on STM32.....

Debugging the FileX Code

Step over  until

```
237  /* Close the test file. */  
238  status = fx_file_close(&fx_file);  
239
```

Finally, we need to close the file after completing the read operation.

```
247  /* Get the available usable space, after the file has been created */  
248  status = fx_media_space_available(&ram_disk, &available_space_post);  
249
```

If you need, you can also check the available space in the disk after the file create and write operations.

Conclusion

- We have seen how to implement FAT file system using X-CUBE-AZURE on the internal SRAM.
- Using STM32CubeMX, it is also possible to implement the same on different memories like uSD card or NOR flash, etc. by adding related drivers for these memories into the project.

Thank you