Configuring STM32F7508-DISCO

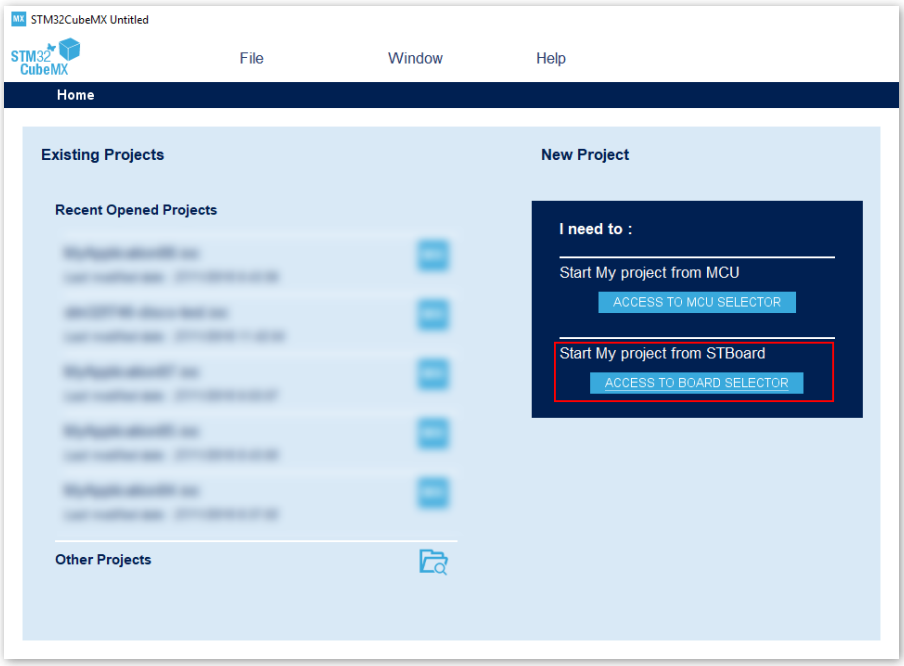
Introduction

CubeMX will configure a project without support for BSP Layer and therefore also BSP files for the touchscreen is missing. The linker script also needs to be updated to comply with all code are executed from external flash. Some changes are therefore needed to run TouchGFX on the STM32F7508-DISCO board. Configuring the CubeMX project to run with TouchGFX is done via the following steps.

1. Creating the initial project based on the STM32F7508-DISCO board.
2. Disable QUADSPI and Adding TouchGFX support in CubeMX.
3. Add missing BSP files and update Linker script.
4. Configure the IAR project.
5. Using the Bootloader from STM32F7Cube

Creating the initial project

To create a project based on a STBoard, create a new project by selecting Access to Board Selector under Start My project from STBoard.



In the Board Selector menu, create a project based on the STM32F7508-DISCO board by using the filtering option

1. Select type of board
2. Select MCU family
3. Select STM32F7508-DISCO board

Related articles

Configuring STM32F769I-DISCO

Configuring STM32F746G-DISCO

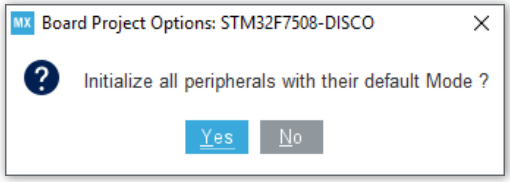
Getting Started with CubeMX and TouchGFX

Configuring STM32F429I-DISCO

Connecting the UI to your system

3. Select STM32F7508-DISCO board
4. Select "Start Project"

After selecting "Start Project" a pop-up asking to Initialize all peripherals to default mode appears. Select yes to the Pop-up.



With the project initialized the project should be saved and the initial code should be generated. This is done by mowing to the Project Manager tab and do the following:

1. Set the name for the project
2. Use the Generate Code button to save the project and generate the initial code

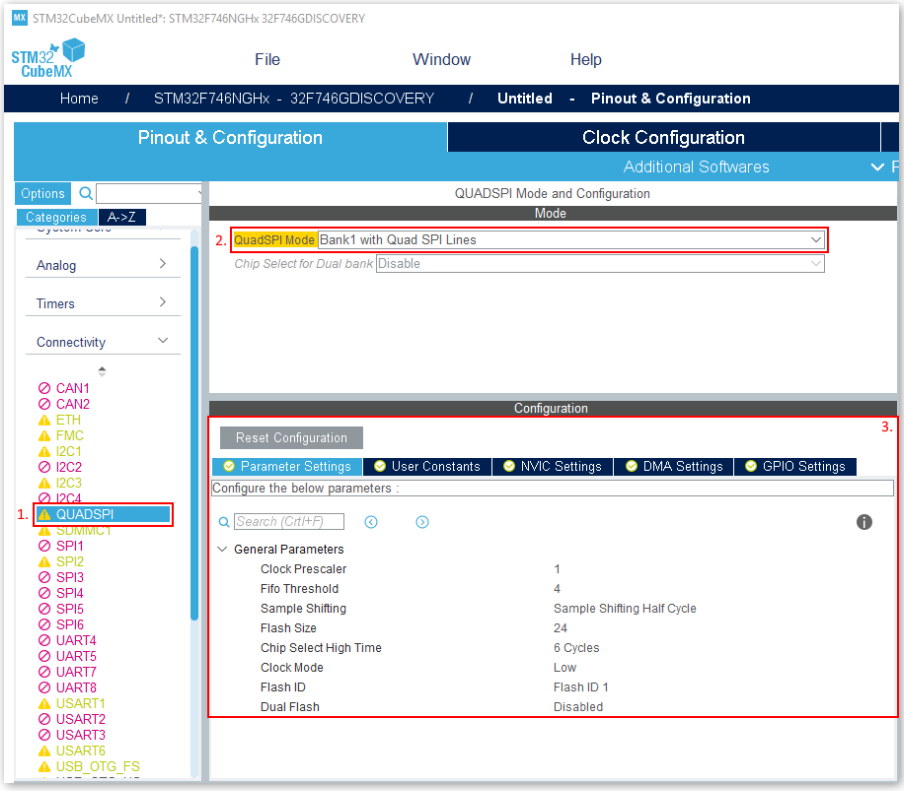
Adding TouchGFX support and disable QUADSPI component

With the project created and the initial code generated, we are ready to perform the configuring TouchGFX to the project and remove the QUADSPI component.

Disable the QUADSPI component is done in the following way:

1. Select QUADSPI under connectivity.
2. Open the drop down menu for "QUADSPI Mode" and select "Disable

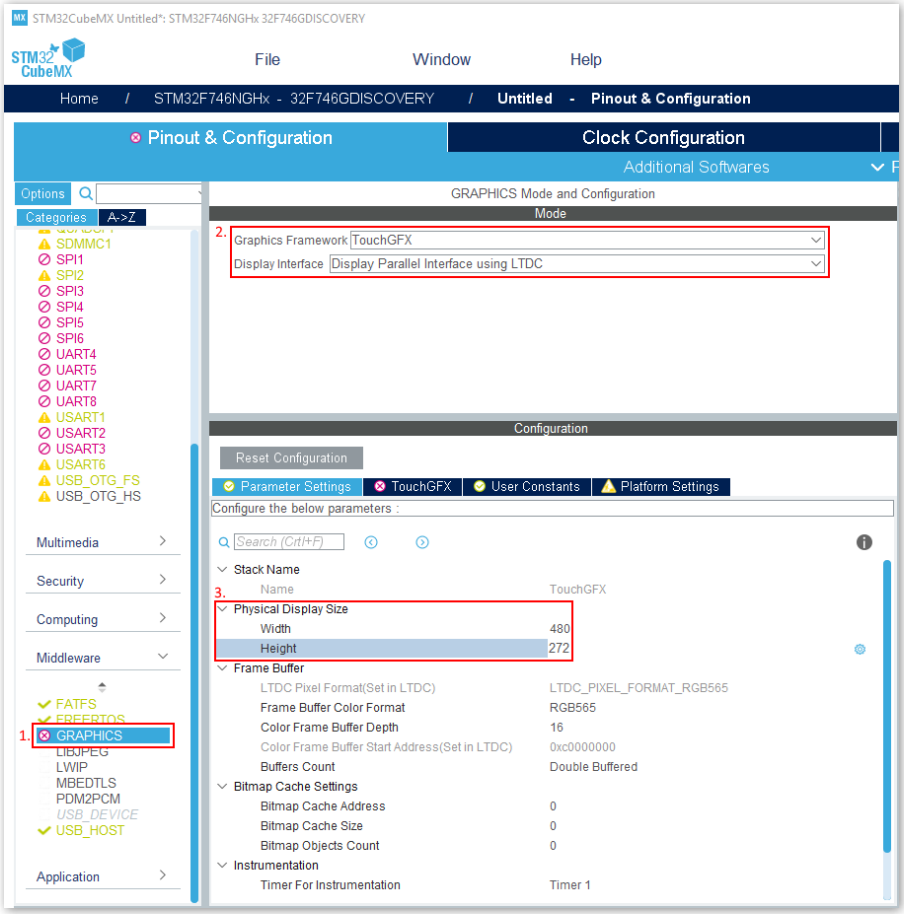
Disabling QUADSPI might seems non-intuitive since the application is executed from external flash accessed using QUADSPI, but the QUADSPI configuration is done in the bootloader and is there not needed to be done by the application.



Next, add TouchGFX to the project and configure to work on the STM32F7508-DISCO board.

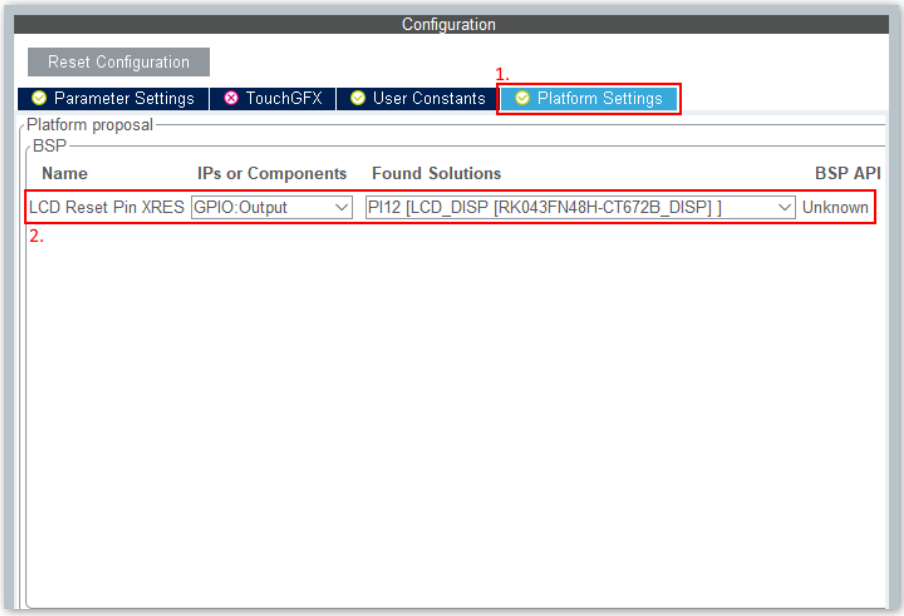
First TouchGFX is added and its parameters are updated by performing the steps below:

1. Select GRAPHICS under middleware.
2. Select TouchGFX as the Graphics Framework and Display Parallel Interface using LTDC.
3. Change Physical Display Size to 480 x 272.



The Reset Pin is set up under Platform settings

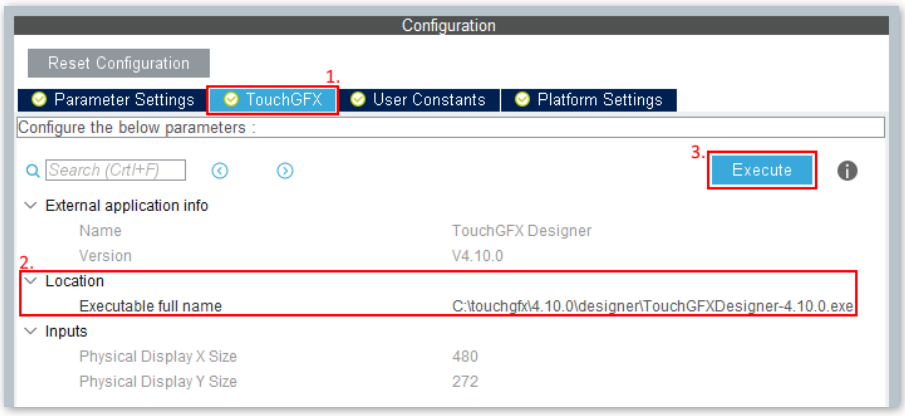
1. Select Platform Settings under Configuration
2. Set the LCD Reset Pin to PI12 (LCD_DIPS) by scrolling down in the drop down menu under Found Solutions



Finally, we need to tel CubMX where the TouchGFX Designer is located and generated a TouchGFX Designer project.

1. Select TouchGFX under Configuration

2. Select the intalled location for the TouchGFX Designer
3. Start the TouchGFX Designer via the button Execute



Warning

- It is recommended that the "Execute" action is only performed once during project creation, because this step instantiates a new empty TouchGFX application - resulting in existing work will be overwritten.

Add missing BSP files and update Linker script.

With the BSP files not added for the QSPI and the touchscreen, theses files needs to be added to the project.

The BSP files are in STM32F7Cube this location can be found by going to Project manager, Use Default Firmware Location.

In the firmware folder, open the folder BSP under Drivers, and copy the two folders "Components" and "STM32F7508-Discovery". In the folder for the CubeMX project create a folder named BSP under Drivers and place the two copied folder to the BSP folder.

Since the application is executed from external flash the linker script needs to be updated accordingly. This includes placing all code and assets in external memory and determining the location of the .intvec section in external flash.

The .intvec is placed at the beginng of the external flash and also exported as a symbol to be referenced in code.

```
define symbol __ICFEDIT_intvec_start__ = 0x90000000;
export symbol __ICFEDIT_intvec_start__;
```

Next the ROM region start and end addresses are set.

```
define symbol __ICFEDIT_region_ROM_start__ = 0x90000000;
define symbol __ICFEDIT_region_ROM_end__ = 0x901FFFFF;
```

Hereafter a section for the assets are made and placed after the ROM region.

```
define symbol __ICFEDIT_region_QSPI_assets_start__ = 0x90200000;
define symbol __ICFEDIT_region_QSPI_assets_end__ = 0x93FFFFFF;

define region QSPI_assets_region = mem:[from __ICFEDIT_region_QSPI_assets_sta
```

```
place at address mem:__ICFEDIT_intvec_start__ { readonly section .intvec };
place in ROM_region { readonly };
place in int_RAM_region { readwrite, block CSTACK, block HEAP };
place in QSPI_assets_region { section ExtFlashSection, section TextFlashSecti
```

Resulting in a linker script as shown below

```
/*-Specials-*/
define symbol __ICFEDIT_intvec_start__ = 0x90000000;
export symbol __ICFEDIT_intvec_start__;
/*-Memory Regions-*/
define symbol __ICFEDIT_region_ROM_start__ = 0x90000000;
define symbol __ICFEDIT_region_ROM_end__ = 0x901FFFFF;
define symbol __ICFEDIT_region_RAM_start__ = 0xC0000000;
define symbol __ICFEDIT_region_RAM_end__ = 0xC07FFFFF;
define symbol __ICFEDIT_region_int_RAM_start__ = 0x20000000;
define symbol __ICFEDIT_region_int_RAM_end__ = 0x2004FFFF;

define symbol __ICFEDIT_region_QSPI_assets_start__ = 0x90200000;
define symbol __ICFEDIT_region_QSPI_assets_end__ = 0x93FFFFFF;

/*-Sizes-*/
define symbol __ICFEDIT_size_cstack__ = 0x400;
define symbol __ICFEDIT_size_heap__ = 0x200;
/**** End of ICF editor section. ###ICF###*/

define memory mem with size = 4G;
define region ROM_region = mem:[from __ICFEDIT_region_ROM_start__ to __ICFEDI
define region QSPI_assets_region = mem:[from __ICFEDIT_region_QSPI_assets_sta
define region RAM_region = mem:[from __ICFEDIT_region_RAM_start__ to __ICFEDI
define region int_RAM_region = mem:[from __ICFEDIT_region_int_RAM_start__ to

define block CSTACK with alignment = 8, size = __ICFEDIT_size_cstack__ { };
define block HEAP with alignment = 8, size = __ICFEDIT_size_heap__ { };

initialize by copy { readwrite };
do not initialize { section .noinit };

place at address mem:__ICFEDIT_intvec_start__ { readonly section .intvec };

place in ROM_region { readonly };
place in int_RAM_region { readwrite, block CSTACK, block HEAP };

place in QSPI_assets_region { section ExtFlashSection, section TextFlashSecti
```

In Core/Src/system_stm32f7xx.c the reference to the intvec tables needs to be added, this is done in the function "void SystemInit(void)" where the last line

```
SCB->VTOR = FLASH_BASE | VECT_TAB_OFFSET;
```

is replaced by

```
SCB->VTOR = (unsigned int)&__ICFEDIT_intvec_start__ | VECT_TAB_OFFSET;
```

Before the function add the external link to:

Before the function add the external link to:

```
extern unsigned int __ICFEDIT_intvec_start__;
```

Modifying the EWARM Project

1. Add components from Cube package to project folder
2. Add include paths to EWARM project
3. Add source files to EWARM project

Open the IAR project in EWARM -> Project.eww

In the project add a sub-group under Drivers called BSP and add the three files *stm32f7508_discovery.c and stm32f7508_discovery_ts.c* which is located in STM32F7Cube installation foler, [STM32F7Cube]Drivers/BSP/STM32f7508-Discovery and the file *ft5336.c* located in [STM32F7Cube]/Drivers/BSP/Components/ft5336

Add the STM32F7508-Discovery folder to included directories by opening the options for the project wither by pressing Alt + F7 or selecting options under Project in the top bar.

In the options menu adding the directory is done in the following way.

1. Select the category C/C++ Compiler
2. Go to the tab Preprocessor
3. Press the button labeled ... to open Edit Include Directories
4. Scroll down and click on <Click to add> to add a new directory
5. Navigate to the STM32F7508-Discovery folder and select the folder

Add Code

1. Open *STM32F7TouchController.cpp*
2. Add include for the toucscreen bsp file inside the "USER CODE" section
   - `#include <stm327f508_discovery_ts.h>`
3. In the STM32F7TouchController::Init remove the comment for "BSP_TS_Init" function inside the "USER CODE" section.
4. In the STM32F7TouchController::sampleTouch remove the comment inside the "USER CODE" section.

```
#include "STM32F7TouchController.hpp"
/* USER CODE BEGIN BSP user includes */
#include <stm32f7508_discovery_ts.h>
/* USER CODE END BSP user includes */

extern "C"
{
  uint32_tLCD_GetXSize();
  uint32_tLCD_GetYSize();
}

using namespace touchgfx;

void STM32F7TouchController::init()
{
  /* USER CODE BEGIN F4TouchController_init */
  /* Add code for touch controller Initialization */
```

```
  BSP_TS_Init(LCD_GetXSize(), LCD_GetYSize());
  /* USER CODE END F4TouchController_init */
}

bool STM32F7TouchController::sampleTouch(int32_t& x, int32_t& y)
{
  /* USER CODE BEGIN F4TouchController_sampleTouch */
  TS_StateTypeDef state = { 0 };
  BSP_TS_GetState(&state);
  if (state.touchDetected)
  {
    x = state.touchX[0];
    y = state.touchY[0];
    return true;
  }
  return false;
  /* USER CODE END F4TouchController_sampleTouch */
}
```

Using the Bootloader from STM32F7Cube

The STM32F7508 needs a bootloader to initialize the peripherals for external memory and then jump to execute code from external flash.

The project for the bootloader is placed in [STM32F7Cube]/Projects/STM32F7508-DISCO/Templates/ExtMem_Boot/.

If using either EWARM or MDK toolchains to build the bootloader project, please ensure that the patch for the STM32F7508 has been run, the patches are located in [STM32F7Cube]/Utilities/PC_Software/patch.

After installing the patch, compile and flash the target with the bootloader to 0x08000000

The bootloader initializes the external flash and jumps to the application's intvec table located at the beginning of the external flash area (0x90000000).