Configuring STM32F746G-DISCO

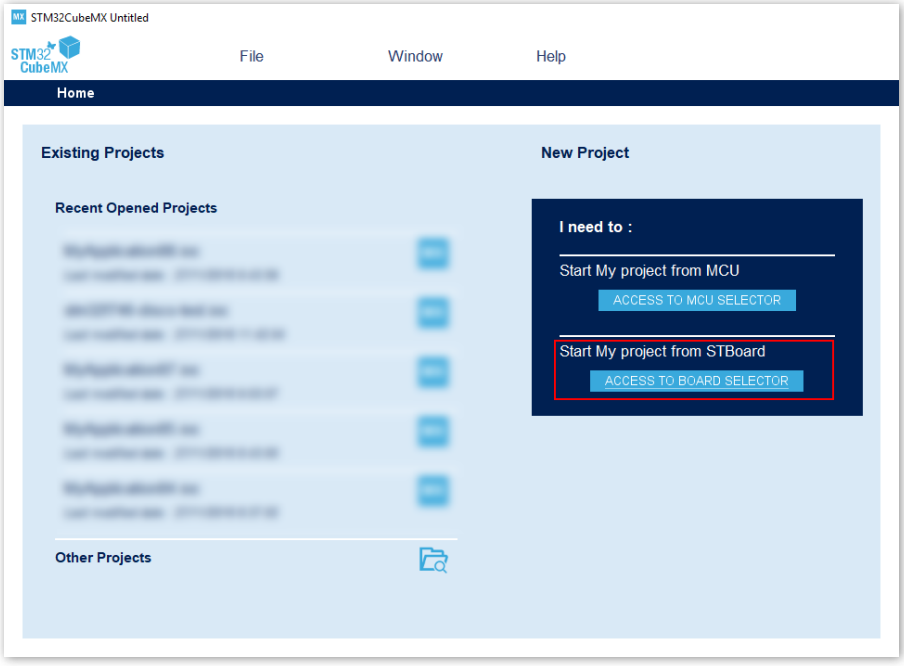
Introduction

CubeMX will configure a project without support for the QSPI flash and therefore are the BSP for the QSPI along with the BSP for the touchscreen missing. The linker script also needs to be updated to access the external flash via QSPI. Some changes are therefore needed to run TouchGFX on the STM32F746G-DISCO board. Configuring the CubeMX project to run with TouchGFX is done via the following steps.

1. Creating the initial project based on the STM32F746G-DISCO board.
2. Setting up QSPI and Adding TouchGFX support in CubeMX.
3. Add missing BSP files and update Linker script.
4. Configure the IAR project.

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 STM32F746G-DISCO board by following the step below.

1. Select type of board
2. Select MCU family

Recently viewed articles

Running TouchGFX without an operating system

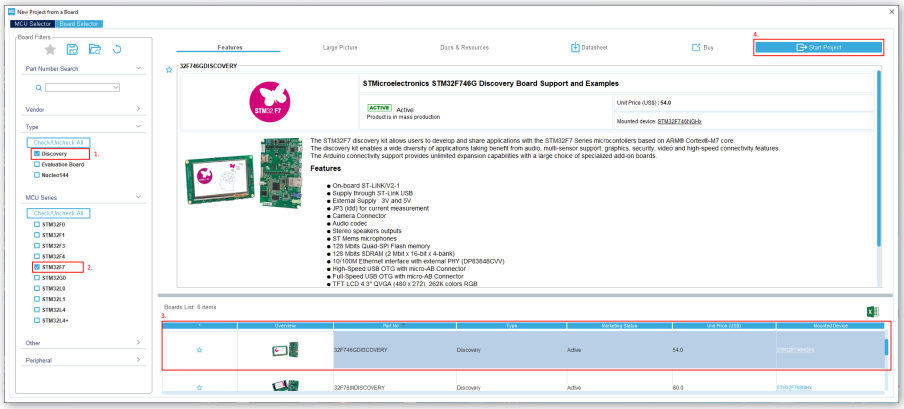
Using other IDEs with TouchGFX

Installing TouchGFX

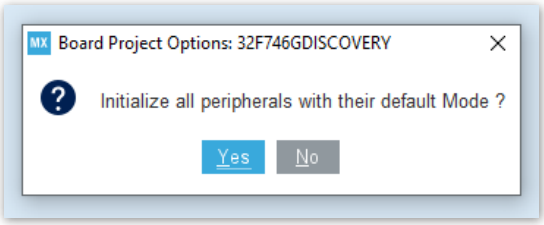
Known Issues

TouchGFX HAL Development

3. Select STM32F746G-DISCO board
4. Select "Start Project"

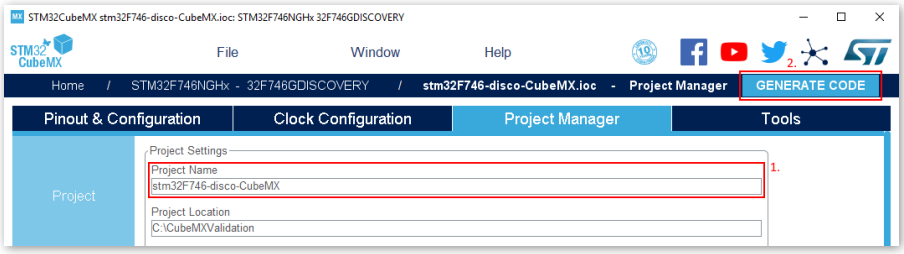


After selecting "Start Project" a pop-up asking to Initialize all peripherals to default mode appears. Select yes to the Pop-up.



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

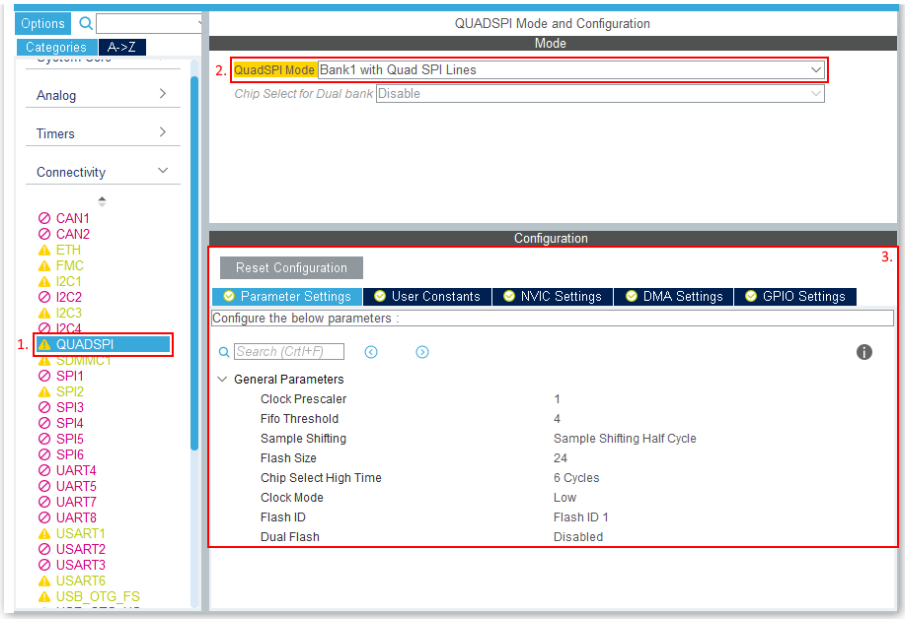


Configuring QSPI and Adding TouchGFX support

With the project created and the initial code generated, we are ready to perform the configuring of the QSPI and add TouchGFX to the project.

Configuring the QSPI is done in the following way:

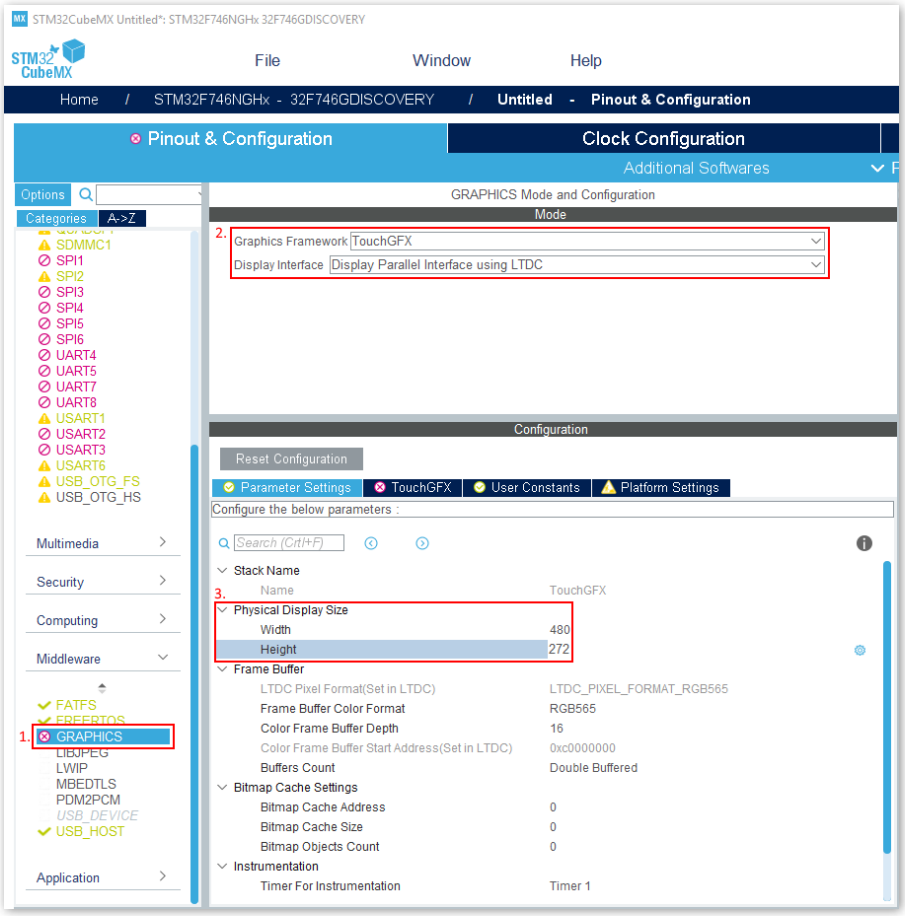
1. Select QUADSPI under connectivity.
2. Ensure that Bank1 with Quad SPI Lines is selected under QuadSPI Mode.
3. Configure the parameters for the QuadSPI as shown in the picture.



Next, add TouchGFX to the project and configure to work on the STM32F746-DISCO board.

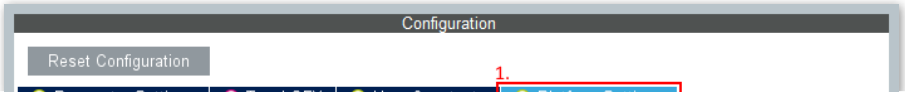
First TouchGFX is added and its parameters are updated by performing the steps below:

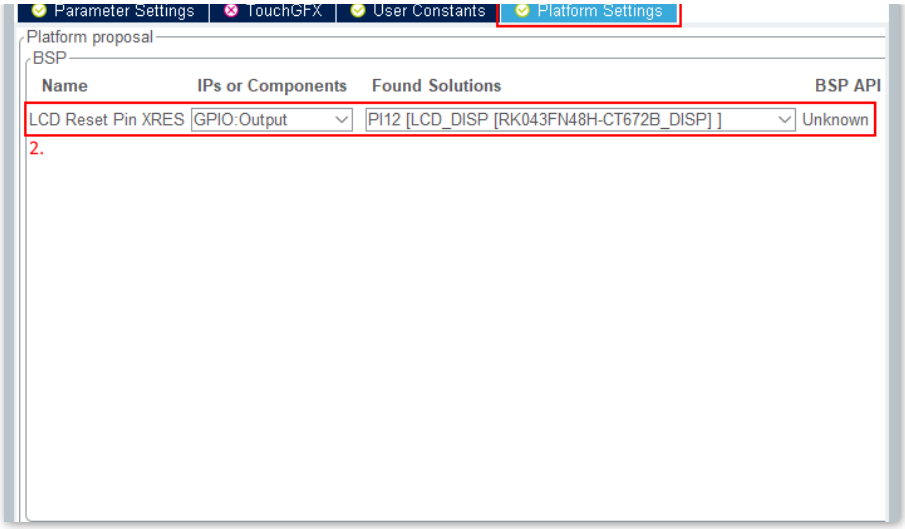
1. Select GRAPHICS under middlewares.
2. Select TouchGFX as the Graphics Framework and Display Parallel Interface using LTDC.
3. Change Physical Display Size to 480 x 272.



The Reset Pin is set up under Platform settings

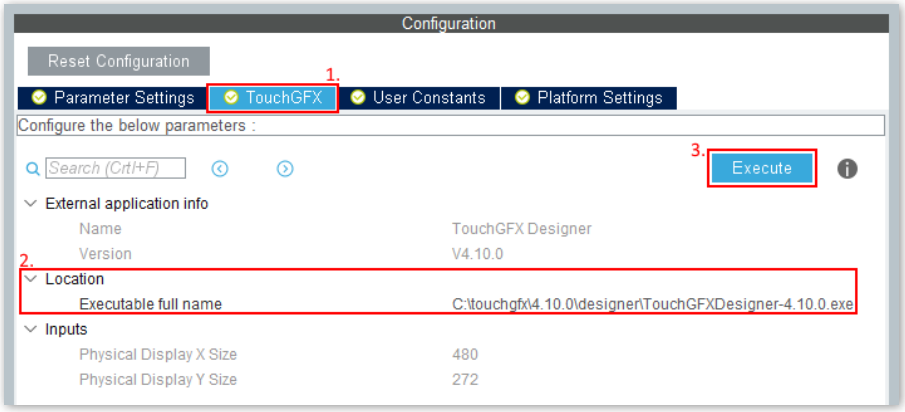
1. Select Platform Settings under Configuration
2. Set the LCD Reset Pin to PI12 (LCD_DIPS) by scrolling down in the drop down menu under Found Solutions





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 installed location for the TouchGFX Designer
3. Start the TouchGFX Designer via the button Execute



When selecting "Execute" CubeMX generates the TouchGFX Designer project and opens the project. For testing purpose, add an image as a background and a button, thereby confirming that the project configurations are correct, by testing that images are stored correctly in the external flash and the touch controller is working when programming the board.

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, these files need to be added to the project.

The BSP files are located with the STM32F7Cube firmware. The location can be found by going to Project manager, Use Default Firmware Location.

In the firmware folder, goto to the folder BSP under Drivers, and copy the two folders

"Components" and "STM32746G-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.

With the QSPI flash added to the project, the linker script needs to be updated accordingly, by defining the region for the QSPI flash and what to place there

Add the following changes to EWARM\stm32f746xx_flash.icf:

```
define symbol __QSPI_start__ = 0x90000000;
define symbol __QSPI_end__ = 0x90100000;

define region QSPI_region = mem:[from __QSPI_start__ to __QSPI_end__];

place in QSPI_region { section ExtFlashSection};
```

Resulting in a linker script as shown below

```
/*###ICF### Section handled by ICF editor, don't touch! ****/
/*-Editor annotation file-*/
/* IcfEditorFile="$TOOLKIT_DIR$\config\ide\IcfEditor\cortex_v1_0.xml" *
/*-Specials-*/
define symbol __ICFEDIT_intvec_start__ = 0x08000000;
/*-Memory Regions-*/
define symbol __ICFEDIT_region_ROM_start__ = 0x08000000;
define symbol __ICFEDIT_region_ROM_end__ = 0x080FFFFF;
define symbol __ICFEDIT_region_RAM_start__ = 0x20000000;
define symbol __ICFEDIT_region_RAM_end__ = 0x2004FFFF;
define symbol __ICFEDIT_region_ITCMRAM_start__ = 0x00000000;
define symbol __ICFEDIT_region_ITCMRAM_end__ = 0x00003FFF;
define symbol __QSPI_start__ = 0x90000000;
define symbol __QSPI_end__ = 0x90100000;
/*-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 __
define region RAM_region = mem:[from __ICFEDIT_region_RAM_start__ to __
define region ITCMRAM_region = mem:[from __ICFEDIT_region_ITCMRAM_start
define region QSPI_region = mem:[from __QSPI_start__ to __QSPI_end__];

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 .intve

place in ROM_region { readonly };
place in RAM_region { readwrite,
 block CSTACK, block HEAP };
place in QSPI_region { section ExtFlashSection};
```

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 *stm32746g_discovery.c*, *stm32746g_discovery_qspi.c* and *stm32746g_discovery_ts.c* which is located in STM32F7Cube installation folder, [STM32F7Cube]Drivers/BSP/STM32746G-Discovery and the file *ft5336.c* located in [STM32F7Cube]/Drivers/BSP/Components/ft5336

Add the "STM32746G-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 "STM32746G-Discovery" folder and select the folder

Add Code

1. Open *STM32F7TouchController.cpp*
2. Add include for the touchscreen bsp file inside the "USER CODE" section
   - `#include <stm32746g_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 <stm32746g_discovery_ts.h>
/* USER CODE END BSP user includes */

extern "C"
{
uint32_t LCD_GetXSize();
uint32_t LCD_GetYSize();

}

using namespace touchgfx;

void STM32F7TouchController::init()
{
 /* USER CODE BEGIN F4TouchController 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 */
}
```

Go to main.c to add code for initializing the QSPI.

Include the QSPI header inside the "USER CODE BEGIN Includes" section by adding

```
#include <stm32746g_discovery_qspi.h>
```

The initialization code is added under the *static void MX_QUADSPI_Init(void)* function and inserted in "USER CODE BEGIN QUADSPI_Init 2" section. Add the code below:

```
BSP_QSPI_Init();

BSP_QSPI_MemoryMappedMode();
HAL_NVIC_DisableIRQ(QUADSPI_IRQn);

MPU_Region_InitTypeDef MPU_InitStruct;
MPU_InitStruct.Enable = MPU_REGION_ENABLE;
MPU_InitStruct.BaseAddress = 0x90000000;
MPU_InitStruct.Size = MPU_REGION_SIZE_256MB;
MPU_InitStruct.AccessPermission = MPU_REGION_FULL_ACCESS;
MPU_InitStruct.IsBufferable = MPU_ACCESS_NOT_BUFFERABLE;
MPU_InitStruct.IsCacheable = MPU_ACCESS_NOT_CACHEABLE;
MPU_InitStruct.IsShareable = MPU_ACCESS_NOT_SHAREABLE;
MPU_InitStruct.Number = MPU_REGION_NUMBER2;
MPU_InitStruct.TypeExtField = MPU_TEX_LEVEL0;
MPU_InitStruct.SubRegionDisable = 0x00;
MPU_InitStruct.DisableExec = MPU_INSTRUCTION_ACCESS_ENABLE;

HAL_MPU_ConfigRegion(&MPU_InitStruct);
```

```
/* Configure the MPU attributes as WT for QSPI (used 16Mbytes) */
MPU_InitStruct.Enable = MPU_REGION_ENABLE;
MPU_InitStruct.BaseAddress = 0x90000000;
MPU_InitStruct.Size = MPU_REGION_SIZE_16MB; /* NOTE! Change this if you
MPU_InitStruct.AccessPermission = MPU_REGION_FULL_ACCESS;
MPU_InitStruct.IsBufferable = MPU_ACCESS_NOT_BUFFERABLE;
MPU_InitStruct.IsCacheable = MPU_ACCESS_CACHEABLE;
MPU_InitStruct.IsShareable = MPU_ACCESS_NOT_SHAREABLE;
MPU_InitStruct.Number = MPU_REGION_NUMBER3;
MPU_InitStruct.TypeExtField = MPU_TEX_LEVEL0;
MPU_InitStruct.SubRegionDisable = 0x00;
MPU_InitStruct.DisableExec = MPU_INSTRUCTION_ACCESS_ENABLE;

HAL_MPU_ConfigRegion(&MPU_InitStruct);
HAL_MPU_Enable(MPU_PRIVILEGED_DEFAULT);
```

The configuration is done and you should now be able to compile and flash the STM32F746-DISCO board.

Was this article helpful? 1 out of 3 found this helpful

Have more questions? Please create a post on the forum.