Integrating CubeMX and TouchGFX

Introduction

This article discusses how to integrate code and or projects generated by CubeMX (C code) with a TouchGFX project (C++ code).

With TouchGFX and the STM32 development kits it is very easy to get started and build prototypes. If you are manufacturing your own (custom) hardware it is recommended to use CubeMX to create a base project for setting up the microcontroller and its peripherals.

You can combine the two projects (TouchGFX and CubeMX) in two ways.

You either want to use:

1. The IAR or KEIL projects that were created by TouchGFX and integrate the CubeMX files
2. The STM32Workbench or Atollic project that was created by CubeMX and insert TouchGFX

Integration Overview

When you generate code and projects from within STM32 CubeMX, you're given a choice of which compiler project to generate.

1. GCC (Stm32workbench, Atollic)
2. IAR
3. Keil

Common for the code generated is that it's all C code. So how do we make TouchGFX (C++) and CubeMX generated code (C) work together?

Generally, we must merge CubeMX generated main.c (Calls to MX_* functions) with TouchGFX main.cpp (sets up touchgfx, RTOS Task and starts the OS scheduler).

Integration example

Here's an example of integrating the TouchGFX main.cpp file with the CubeMX generated main.c file. Code that is TouchGFX related is in bold:

```
void GUI_Task(void const * argument)
{
touchgfx::HAL::getInstance()->taskEntry();
```

Recently viewed articles

Running TouchGFX without an operating system

TouchGFX memory requirements

Installing TouchGFX

TouchGFX HAL Development

Non-memory mapped external flash (e.g. NAND)

Related articles

Interfacing with physical buttons

Debugging Pixel Error Issues

Using texts and fonts

Step 1: Setting up the two Screens

Step 3: Further ideas for improvement

```
}

int main(void)
{

  /* Enable I-Cache---------------------------------------------------
----------*/
  SCB_EnableICache();

  /* Enable D-Cache---------------------------------------------------
----------*/
  SCB_EnableDCache();

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

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

  /* USER CODE BEGIN Init */
  /* USER CODE END Init */

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

  /* USER CODE BEGIN SysInit */
  /* USER CODE END SysInit */

  /* Initialize all configured peripherals */
  MX_GPIO_Init();
  MX_DMA_Init();
  MX_ADC1_Init();
  MX_ADC2_Init();
  MX_DFSDM1_Init();
  MX_DMA2D_Init();
  MX_DSIHOST_DSI_Init();
  MX_FMC_Init();
  MX_I2C3_Init();
  MX_I2C4_Init();
  MX_QUADSPI_Init();
  MX_LTDC_Init();
  MX_RTC_Init();
  MX_SAI2_Init();
  MX_SPI1_Init();
  MX_SPI2_Init();
  MX_SPI6_Init();
  MX_UART4_Init();
  MX_USART1_Init();
  MX_USART2_UART_Init();
  MX_SDMMC1_MMC_Init();
  MX_DAC_Init();
...
```

```
  BSP_DisplayInit(); // Custom method for setting up DSI commands

/* Create the touchgfx GUI Task*/
  osThreadDef(guiTask, GUI_Task, osPriorityHigh, 0, 512);
  guiTaskHandle = osThreadCreate(osThread(guiTask), NULL);

/* Initialize touchgfx */
  touchgfx_init();

  /* Start scheduler */
  osKernelStart();

  while (1) {

  }
}
```

In summary

1. If using an Eclipse-based IDE convert the project from a C project to a C++ project.
2. Create task handle(s) and an OS task with an entry point HAL::taskEntry()
3. Call touchgfx_init() which sets up the framebuffer, DMA, bitdepth, etc. This method is, for existing TouchGFX ports, located inside BoardConfiguration.cpp.

Was this article helpful? 2 out of 2 found this helpful

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