Porting to STM32F4/F7 boards

The main focus of this article is to take you through creating a *board configuration* for your own board based on one of the ST MCUs in the following lineup, all of which are supported by TouchGFX:

1. `STM32F4x9` (ARM Cortex-M4F)
2. `STM32F7xx` (ARM Cortex-M7F)
3. `STM32L4xx` (ARM Cortex-M4F)

A large amount of work involved in porting TouchGFX to a specific MCU lies in configuring the `LCD` - and `DMA` -controllers. For the MCU lineup mentioned above, this work has already been done by TouchGFX.

The `BoardConfiguration` contains the initialization of additional hardware as well as TouchGFX itself. When porting to a new hardware configuration with an ST MCU, basing yourself on a pre-existing board configuration can speed up the process.

Using an existing port, the following *board configuration* tasks remain:

1. *Initialization of hardware*: System Clock, External RAM, External Flash, LCD, Touch, etc.
2. *Initialization of TouchGFX*: Color depth, display size, framebuffer start address (See Configuring TouchGFX).

> **Location:** Board configurations in TouchGFX projects are located under `MyApplication/target/` in your project. Place your new board here.
>
> **Operating System:** TouchGFX supports FreeRTOS v7.6.0 and v8.2.0 (Cortex-M7 support) and bundles these in its distribution. See any project example we provide (IAR, Keil, gcc) for how to include FreeRTOS in your own projects.
>
> Please see the articles Changing to a different RTOS for information on how to change to a different OS, and Running TouchGFX without an operating system for information on how to run without an OS.

The remainder of this article uses the board configuration for the `STM32F469` based `STM32469I-DISCO` board to exemplify.

In general, board configurations within TouchGFX seperate framework initialization from hardware initialization, the latter containing the largest portion of the work you must do yourself.

Hardware configuration

Recently viewed articles

Understanding TouchGFX

Configuring STM32F429I-DISCO

Running TouchGFX without an operating system

Integrating CubeMX and TouchGFX

TouchGFX memory requirements

Related articles

Integrating CubeMX and TouchGFX

Changing to a different display

Introduction

Introduction

Configuring STM32F429I-DISCO

The main porting effort lies in the configuration of hardware peripherals like SDRAM, Touch driver, External Flash, etc, which are not specific to TouchGFX. Not all components are specifically required for TouchGFx to run. In particular:

Required:

1. RAM for frame buffer (*typically external*).

Optional:

1. *External Flash for images*: If your board has a non-memory mapped flash, please see the article Non-memory mapped external flash (e.g. NAND).
2. *Touchcontroller*: As we'll see shortly, the TouchGFX configuration requires a Touch controller driver to instantiate its `HAL`. Please see the article Changing to a different touch controller for specific details on changing and creating touch controllers.

TouchGFX configuration

The first thing we'll do when configuring TouchGFX is to instantiate and configure the `HAL`. We'll start by defining the size of our display:

```
static const uint16_t dispWidth  = 800;
static const uint16_t dispHeight = 480;
```

We must define an MCU specific specializations of `DMA_Interface`. For ST MCU's, TouchGFX provides the following:

1. `STM32F4DMA`
2. `STM32F7DMA`

For the `STM32469I-DISCO` board, we'll use the following:

```
STM32F4DMA dma;
```

TouchGFX supports two kinds of color depths, each of these represented as a TouchGFX LCD driver in `touchgfx/framework/include/platform/driver/lcd/`:

1. 1 bit-per-pixel
2. 2 bit-per-pixel
3. 4 bit-per-pixel
4. 16 bit-per-pixel
5. 24 bits-per-pixel

Since most boards will have a 16bpp capable displays, we'll define the following display type:

```
LCD16bpp display;
```

The `STM32469I-DISCO` board also uses an `OTM8009` touch controller:

```
OTM8009TouchController tc;
```

Besides LCD-TFT, The `STM32F469` MCU also supports a MIPI-DSI interface. The The `STM32469I-DISCO` board uses a DSI display which requires a different handling of synchronization. The following HAL specializations are available:

1. `STM32F4HAL`
2. `STM32F4HAL_DSI`
3. `STM32F7HAL`

For this board, we'll use the `STM32F4HAL_DSI`. Finally, we'll instantiate the TouchGFX `HAL` with the configuration we've defined so far:

```
HAL& hal = touchgfx_generic_init<STM32F4HAL_DSI>(dma, display, tc, dis
```

> ***Note:*** See the article Optimization for further information on how to optimize DMA transfers for platforms that allow simultanous SDRAM from both DMA and TFT controller, optimizing VSYNC timing and touch

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