Module Manager

Table of Contents

1 Main.

1.1 Reset handler.

1.1.1 File '*PHYSICAL/STM32H735/system/stm32h735.ld*' sets '*resetHandler*' as the entry code.

1.1.2 File '*stm32h735.s'* implements the *'resetHandler'* code, which initializes all memory, and then invokes the '*main*' function.

1.1.3 **Important – This code does not support static class constructors!** (We still support static and global variables initiations).

1.2 Main function.

1.2.1 '*APPLICATION/APP_STM32H735/basic_stm32h735.cc*' implements the '*main*' function.

1.2.2 When entering the main function, the first action is to declare the '*ApplicationMain*' singleton.

1.2.3 Following that, we declare all the application specific active module managers, using the flags generated only for the main directory.

1.2.4 Each module manager is automatically registered by the '*Main*' singleton.

1.2.5 After declaring all modules managers, we invoke the '*main*' loop in the 'Main' module.

1.3 Main class.

1.3.1 The 'Main' class has two phases – *initialization*, and *ticking*.

1.3.2 Initialization phase:

1.3.3 During each phase, the '*Main*' implementation invokes '*registerPhaseStarted*' for the '*ApplicationMain*' implementation, followed by invoking the '*doAction*' method for each active module manager.

1.3.4 '*MODULE_ACTION_OPEN*' – Initialize class implementation related variables that rely on other modules, or accessing virtual methods.

1.3.5 '*MODULE_ACTION_CONFIGURE_PLL_CLOCKS*' – Setup the pll clocks for modules usage.

1.3.6 '*MODULE_ACTION_START_PLL_CLOCKS*' – Enable all active pll clocks. Setup clock system.

1.3.7 '*MODULE_ACTION_ACTIVATE_GPIO*' – Setup active gpios (Must come after pll clocks setup).

1.3.8 '*MODULE_ACTION_START_MDMA*' – Enables the sdram (Must come after gpios setup).

1.3.9 '*MODULE_ACTION_START*' – All leftover setups.

1.3.10 Ticking phase – the main loops repeatedly invokes '*registerPhaseStarted*' for the main application, followed by invoking '*doAction (MODULE_ACTION_TICK)*' for each active module manager.

1.3.11 **We currently limit execution time to 12 days (0x3FFFF000 milliseconds)**.

2 Module managers.

2.1 Inheritance.

2.1.1 **IMPORTANT NOTE**: Module managers are the only classes that may support multiple inheritance. They must inherit the '*ModuleManager*' class, and may also implement any number of pure virtual interfaces (but only pure virtual).

2.2 ModuleManagerClockControl

2.2.1 Manages all shared clock related issues (such as pll definitions).

2.3 ModuleManagerInterrupts

2.3.1 Manages all interrupts.

2.3.2 This class, as well as the 'CriticalSection' class, are described in 'Interrupts' documentation.

2.4 ModuleManagerFlash

2.4.1 Access the external flash chip. All flash accesses are blocking. (using the *OCTOSPI* module).

2.5 ModuleManagerSdram

2.5.1 Access the external sdram chip. All sdram accesses are done using *dma*. (using the *OCTOSPI* module).

2.6 ModuleManagerLeds

2.6.1 Platform specific for setting the leds.

2.7 ModuleManagerLcd

2.7.1 Access the lcd screen using the *LTDC* module.

2.8 *ModuleManagerTouchScreen*

2.8.1 Access the touch screen using the I2C module.