```
@Done
@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@
@                                                                              @
@ crt0.s                                                                       @
@                                                                              @
@ Initialization file for EE52 ARM VoIP phone project.  It sets up the IRQ    @
@ vector table, initializes the stacks for both the IRQ and System modes,     @
@ sets up the Master Clock, all of the chip selects for external memories,    @
@ and will eventually call all of the intialization functions for each        @
@ hardware block.  Finally it invokes the main user interface code.          @
@                                                                              @
@                                                                              @
@ Revision History:                                                            @
@                                                                              @
@   2008/02/02  Joseph Schmitz  Modified code from Arthur Chang to make it    @
@                               available to the students.                    @
@   2011/01/27  Joseph Schmitz  Split into crt0.s and boot.s.                 @
@   2011/01/31  Joseph Schmitz  Removed unused comments.                      @
@   2012/01/24  Glen George     Updated comments, modified included files,    @
@                               and cleaned up the code a little.             @
@   2012/01/29  Glen George     Added comments explaining initialization.     @
@                                                                              @
@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@

.include    "at91rm9200.inc"
.include    "system.inc"
.include    "macro.inc"
.include    "interfac.inc"

.text
.arm

@ In this file you must do at least the following:
@
@   - As you write other functions/code for the various hardware blocks, you
@     will call the initialization functions for them from here as well.



@ low_level_init
@
@ Description: Initializes the ARM processor modes, calls the
@              initialization functions for the different parts
@              of the code, and then calls the main loop.
@
@ Operational Description: Initializes arm processor modes,
@                          then goes through and calls a sequence
@                          of initialization functions for various
@                          files. Finally, the main loop is called
@                          (or for the audio demo, audio_demo is called)
@
@ Arguments: None
@
@ Return values: never returns
@
@ Local variables: None
@
@ Shared variables: None
@
@ Global Variables: None
@
@ Inputs: None
@
@ Outputs: None
@
@ Error Handling: None
```

```
@
@ Algorithms: None
@
@ Data Structures: None
@
@ Limitations: None
@
@ Registers Changed (besides ARM convention r0-r3): None
@
@ Known Bugs: None
@
@ Special notes: None
@
@ Revision History:
@ Name             Comment              Date
@ Will Werst        Initial version     Some lonely night around 6/10/17

.global low_level_init
low_level_init:

    @Uncomment instruction below to prevent ROM code from executing after
    @bootloader runs. This effectively creates a dummy ROM code that does nothing.
    @B low_level_init

@ Stack and IRQ Initialization

    LDR     r0, =TOP_STACK                  @load address for top of the stack

    @ put the CPU in interrupt mode and set the stack pointer for this mode
    MSR     cpsr_c, #ARM_MODE_IRQ | I_BIT | F_BIT
    MOV     sp, r0

    @ adjust the starting stack address for the interrupt stack just setup
    SUB     r0, r0, #IRQ_STACK_SIZE

    @ put the CPU in supervisor mode and set the stack pointer for this mode
    MSR     cpsr_c, #ARM_MODE_SVC | I_BIT | F_BIT
    MOV     sp, r0

    @ adjust the starting stack address for the supervisor mode stack just setup
    SUB     r0, r0, #SVC_STACK_SIZE

    @ finally, put the CPU in user mode and set the stack pointer for this mode
    MSR     cpsr_c, #ARM_MODE_USR | F_BIT
    MOV     sp, r0



@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@
@  user initialization goes here  @
@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@
    BL init_system                          @ Initialize the system
    BL keypad_init                          @ Initialize the keypad
    BL audio_init                           @ Initialize the audio
    BL display_init                         @ Initialize the display

loop:
    @BL      audioDemo              @ Uncomment this line to run the audioDemo
    BL      main                    @ run the main function (no arguments)

    B       low_level_init          @ if main returns (shouldn't)
                                    @ reinitialize everything and start
                                    @ over

.end
```