```
@done
@
@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@
@
@
@ Audio.s
@
@ Description: Contains code for controlling the audio component of
@ the EE52 VoIP Project
@
@ Table of Contents:
@   - audio_init: Call to initialize this code file before
@                   calling anything else
@   - call_start: Call to initiate a call
@   - call_halt:  Call to halt a call that has been started with call_start
@   - update_rx:  Handler used to update the receive buffer
@   - update_tx:  Handler used to update the transmit buffer
@   - setVolume:  Call to set the volume
@   - audioDemo:  A test function used to demo the audio code on its own,
@                 by looping the audio input back to the audio output.
@
@
@ Revision History:
@ Name          Comment                  Date
@ Will Werst    Initial version          Some lonely night around 6/10/17
@ Will Werst    Comment                  October 2017
@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@

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


.text
.arm



@ audio_init
@
@ Description: Call to initialize everything in this file
@
@ Operational Description: Initializes the registers to the
@                          initialization values for this project
@
@ Arguments: None
@
@ Return values: None
@
@ Local variables: None
@
@ Shared variables: None
@
@ Global Variables: None
@
@ Inputs: SSC0 (audio serial communication) - initialized
@
@ Outputs: SSC0 (audio serial communication) - initialized
@
@ 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 audio_init
audio_init:
    mSTARTFNC
    mSET_HREG   PMC_PCER,   (1 << 14)       @Enable the peripheral clock
    mSET_HREG   SSC0_CR,    SSC0_CR_VAL     @Setup the serial controller
    mSET_HREG   SSC0_CMR,   SSC0_CMR_VAL    @Setup the serial clock rate
    mSET_HREG   SSC0_RCMR,  SSC0_RCMR_VAL   @Setup receive clock mode
    mSET_HREG   SSC0_RFMR,  SSC0_RFMR_VAL   @Setup receive frame mode
    mSET_HREG   SSC0_TCMR,  SSC0_TCMR_VAL   @Setup transmit clock mode
    mSET_HREG   SSC0_TFMR,  SSC0_TFMR_VAL   @Setup transmit frame mode
    mSET_HREG   PIOB_ASR,   0xF             @Setup serial output pins
    mSET_HREG   PIOB_PDR,   0xF             @Setup serial output pins
    mSET_HREG   PIOB_OER,   0xF             @Setup serial output pin
    mSET_HREG   SSC0_CR,    0x00000101      @Enable serial
    mSET_HREG   SSC0_PTCR,  0x00000101      @Enable DMA
    mRETURNFNC


@ call_start
@
@ Description: Initializes a call with the initial receive buffer pointer (r0)
@
@ Operational Description: Passes the initial buffer pointer (r0) to the
@                          function that handles adding new buffers
@                          to the system.
@
@ Arguments: r0 - pointer to first buffer in DRAM
@
@ Return values: None
@
@ 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: update_rx should be called, not update_tx. This ends up being
@             fine for long term operation because only the first buffer
@             played is affected, which is why it worked in the demo.
```

```
@
@ Special notes: None
@
@ Revision History:
@ Name             Comment              Date
@ Will Werst        Initial version     Some lonely night around 6/10/17

.global call_start
call_start:
    mSTARTFNC                               @Call start function macro
    BL  update_tx
    mRETURNFNC                              @Call return from function macro


@ call_halt
@
@ Description: Halts the current call regardless of the state of buffers.
@              If the call is already halted, this has no effect.
@
@ Operational Description: The counter registers for the DMA engine are
@                          cleared. This will stop any memory accesses
@                          from occurring through the DMA and into the
@                          buffers.
@
@ Arguments: None
@
@ Return values: None
@
@ Local variables: None
@
@ Shared variables: None
@
@ Global Variables: None
@
@ Inputs: None
@
@ Outputs: SSC0 - the counter registers for DMA are cleared
@
@ 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 call_halt
call_halt:
    mSTARTFNC                               @Call start function macro
    @Clear the counters in all DMA registers for audio
    mSET_HREG   SSC0_RNCR, 0                @Clear the counter register for the
                                            @next DMA buffer receive
    mSET_HREG   SSC0_RCR, 0                 @Clear the counter register for the
                                            @buffer currently being received
    mSET_HREG   SSC0_TNCR, 0                @Clear the counter register for the
                                            @next DMA buffer transmit
    mSET_HREG   SSC0_TCR, 0                     @Clear the counter register for the
                                                @buffer currently being transmitted
    mRETURNFNC                                  @Call return from function macro


@ update_rx
@
@ Description: Takes a pointer, and either uses it and returns true, or
@              discards it and returns false.
@
@ Operational Description: The next counter register is pulled from
@                          the DMA engine and compared to 0, to see
@                          if the next register in DMA is empty. If
@                          it is, the new register is added to the
@                          DMA engine, and true is returned. Else,
@                          false is returned.
@
@ Arguments: r0 - pointer to new record buffer
@
@ Return values: bool - true if passed buffer is used, else false.
@
@ Local variables:
@
@ Shared variables:
@
@ Global Variables:
@
@ Inputs: SSC0 - sets up DMA receive
@
@ 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 update_rx
update_rx:
    mSTARTFNC                               @Call start function macro
    mLOADTOREG  r1, SSC0_RNCR               @Load the length of the next receive
                                            @buffer currently queued up
    CMP     r1,     #0                      @Check if next receive buffer is empty
    LDRNE   r1,     =FALSE                  @If memory is not empty
                                            @Post-demo comment: I'm pretty sure this
                                            @should have been r0, not sure why this
                                            @worked
    BNE     endUpdate_RX                    @return false since new buffer not needed
    @BEQ    rx_empty                        @Call return function macro
rx_empty:
    mSTOREFROMREG   r0, r1, SSC0_RNPR       @Store r0 (next pointer) into next
                                            @pointer register
    mSET_HREG   SSC0_RNCR,  (AUDIO_BUFLEN /2) - 1 @Store buffer length in half-words
    LDR     r0,     =TRUE                   @Return success
```

```
endUpdate_RX:
    mRETURNFNC


@ update_tx
@
@ Description: Takes a transmit buffer pointer, and either uses it and returns
@              true, or discards it and returns false.
@
@ Operational Description: The next counter register is pulled from
@                          the DMA engine and compared to 0, to see
@                          if the next register in DMA is empty. If
@                          it is, the new register is added to the
@                          DMA engine, and true is returned. Else,
@                          false is returned.
@
@ Arguments: r0 - pointer to new transmit buffer
@
@ Return values: bool - true if passed buffer is used, else false.
@
@ Local variables: None
@
@ Shared variables: None
@
@ Global Variables: None
@
@ Inputs: None
@
@ Outputs: SSC0 - sets up DMA transmit
@
@ 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 update_tx
update_tx:
    mSTARTFNC
    mLOADTOREG  r1, SSC0_TNCR
    CMP     r1,     #0                  @Check if next transmit buffer is empty
    LDRNE   r1,     =FALSE              @If memory is not empty
                                        @Post-demo comment: I'm pretty sure this
                                        @should have been r0, not sure why this
                                        @worked
    BNE     endUpdate_TX                @return false since new buffer not needed
    @BEQ    tx_empty
tx_empty:
    PUSH    {r0-r3}                     @Store registers to free up for temp
    LDR     r1, =(AUDIO_BUFLEN -2)      @Load buffer length
    LDR     r2, =AUDIO_VOLUME           @Load the volume setpoint
    BL      setVolume                   @Set volume
    POP     {r0-r3}                     @Restore register
    mSTOREFROMREG   r0, r1, SSC0_TNPR   @Save next transmit register
    mSET_HREG   SSC0_TNCR,  (AUDIO_BUFLEN / 2) - 1 @Save length of next transmit
                                        @Register in half-words
    LDR     r0,     =TRUE               @Return true
endUpdate_TX:
    mRETURNFNC


@ setVolume
@
@ Description: Goes through the transmit buffer and sets the volume bits
@
@ Operational Description: The transmit buffer is looped over in reverse, and
@                          each byte is OR-masked with the volume, and saved back
@                          in-place in the buffer.
@
@
@ Arguments: r0 - pointer to transmit buffer to set volume for
@            r1 - length of transmit buffer in bytes
@            r2 - volume, encoded as the bits to preserve in each byte
@ Return values: None
@
@ 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

setVolume:
    mSTARTFNC
updateValue:
    LDRH r3, [r0,r1]                @Get byte of audio
    ORR  r3, r3, r2                 @Set the high bits for volume control
    STRH r3, [r0,r1]                @Store byte of audio
    SUB r1, #2                      @Decrement to next byte
    CMP r1, #0                      @Check if at last byte
    BGE updateValue                 @If not, continue
    mRETURNFNC                      @Return


@ audioDemo
@
@ Description: Audio is looped back from the microphone to the speaker
@              continuously
@
```

```
@ Operational Description: Five buffers are looped through, starting with
@                          receive as buffer 1 and transmit as buffer 3.
@                          The loop is unrolled, and the loop has no exit
@                          condition.
@
@ Arguments: None
@
@ Return values: None (never returns)
@
@ Local variables: Buf[1..5] - buffers for storing audio data
@
@ Shared variables: None
@
@ Global Variables: None
@
@ Inputs: None
@
@ Outputs: None
@
@ Error Handling: None
@
@ Algorithms: None
@
@ Data Structures: Circular buffer
@
@ Limitations: Loop is unrolled, and the buffers are defined separately
@              rather than as an array. Changing buffer count is not
@              trivial.
@
@ 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 audioDemo
audioDemo:
    mSTARTFNC
loopAudioDemo:
    Buf1_rx:
    LDR     r0, =Buf1
    BL  update_rx
    CMP r0, #TRUE
    BNE Buf1_rx

    LDR     r0, =Buf3
    BL  update_tx

Buf2_rx:
    LDR     r0, =Buf2
    BL  update_rx
    CMP r0, #TRUE
    BNE Buf2_rx

    LDR     r0, =Buf4
    BL  update_tx

Buf3_rx:
    LDR     r0, =Buf3
    BL  update_rx
    CMP r0, #TRUE
    BNE Buf3_rx
    LDR     r0, =Buf5
    BL  update_tx

Buf4_rx:
    LDR     r0, =Buf4
    BL  update_rx
    CMP r0, #TRUE
    BNE Buf4_rx

    LDR     r0, =Buf1
    BL  update_tx

Buf5_rx:
    LDR     r0, =Buf5
    BL  update_rx
    CMP r0, #TRUE
    BNE Buf5_rx

    LDR     r0, =Buf2
    BL  update_tx

    B   loopAudioDemo
    mRETURNFNC

.data

.balign 4
Buf1:           @Audio buffer 1
    .skip 256
Buf2:           @Audio buffer 2
    .skip 256
Buf3:           @Audio buffer 3
    .skip 256
Buf4:           @Audio buffer 4
    .skip 256
Buf5:           @Audio buffer 5
    .skip 256

.end
```