int main(void){

HAL_BOARD_INIT() /* Initialize hardware */

1. Turn on 16 MHz and 32 MHz clocks, wait for clocks to stabilize.
2. Select 32 MHz and 32 KHz clocks. Disable 16 MHz clock.
3. Enable prefetch mode, configure tri-states.
4. Setup RF frontend.

HalDriverInit() /* Initialize the HAL driver */

1. Initialize HAL timer
2. Initialize HAL ADC (analog/digital converter)
3. Initialize HAL DMA (direct memory access)
4. Initialize HAL AES (advanced encryption standard)
5. Initialize HAL LCD, LEDs
6. Initialize HAL UART
7. Initialize HAL keys (sets certain pins to input, enables GPIOs)
8. Initialize HAL HID capability (USB human input device)

MAC_Init() /* Initialize MAC */

1. Extern function to be called once when the software system is started, before any other MAC API.

osal_init_system() /* Initialize the operating system */

1. osal_mem_init(); /* Initialize memory allocation */
2. osal_qHead = NULL; /* Initialize empty message queue */
3. osalTimerInit(); /*Initialize timers */
4. osal_pwrmgr_init(); /*Initialize power management */
5. osalInitTasks(); /* Initialize system tasks */
    a. Clear, initialize task event memory.
    b. macTaskInit(taskID++);
    c. MSA_Init(taskID++);
    d. Hal_Init(taskID);
6. osal_mem_kick(); /* Search for first free block of heap */
7. return (SUCCESS);

HAL_ENABLE_INTERRUPTS() /* Enable interrupts */

HalKeyConfig(MSA_KEY_INT_ENABLED, MSA_Main_KeyCallback) /* Setup Keyboard callback */

1. Register pointer to callback function.
2. Configure interrupts, Enable interrupts at port, enable CPU interrupts, clear any pending interrupts.
3. Configure rising or falling edge.
4. Configure sleep functionality.
5. Enable / disable general interrupts.

6. Start polling.

HalLedBlink (HAL_LED_4, 0, 40, 200) /* Blink LED on startup */

osal_start_system() /* Start OSAL (No Return from here) */

1. osalTimeUpdate();
2. Hal_ProcessPoll(); /* Handles polling for UART, SPI, Timer, USB if interrupts disabled.*/
3. Process all tasks from tasksEvents[idx], beginning with highest priority.
4. If no tasks are available, go to osal_pwrmgr_powerconserve().

return(0);

}

```
// Method 1; SSN kept low during the transfer of all bytes
int* ReadWriteSPI(int length, int* txBufferMaster)
{
        int i;
        SSN = LOW;
        for (i = 0; i <= length; i++)
        {
                U1DBUF = txBufferMaster[i];
                while (!(U1CSR && U1TX_BYTE));
                U1DBUF = rxBufferMaster[i];
        }
        SSN = HIGH;

        return rxBufferMaster;
}
```