```
______________________________________LCD DRIVER______________________________________

//-----------ST7735_Message--------------
//Draws a string and long value one one of the two split screens
//Each logically seperate screen contains four lines
//Used for interaction with the CLI, debugging, and displaying data

void ST7735_Message(int screen,int line,char *string,long value){

    if(screen == 0){
        if(line == 0){
            ST7735_FillRect(0,0,128,8,ST7735_BLACK);
            ST7735_SetCursor(0, 0);
            printf("%s",string);
            ST7735_OutUDec(value);
        }
        else if(line == 1){
            ST7735_FillRect(0,20,128,8,ST7735_BLACK);
            ST7735_SetCursor(0, 2);
            printf("%s",string);
            ST7735_OutUDec(value);
        }
        else if(line == 2){
            ST7735_FillRect(0,40,128,8,ST7735_BLACK);
            ST7735_SetCursor(0, 4);
            printf("%s",string);
            ST7735_OutUDec(value);
        }
        else if(line == 3){
            ST7735_FillRect(0,60,128,8,ST7735_BLACK);
            ST7735_SetCursor(0, 6);
            printf("%s",string);
            ST7735_OutUDec(value);
        }
    }
    else if(screen == 1){
        if(line == 0){
            ST7735_FillRect(0,90,128,8,ST7735_BLACK);
            ST7735_SetCursor(0, 9);
            printf("%s",string);
            ST7735_OutUDec(value);
        }
        else if(line == 1){
            ST7735_FillRect(0,110,128,8,ST7735_BLACK);
            ST7735_SetCursor(0, 11);
            printf("%s",string);
            ST7735_OutUDec(value);
        }
        else if(line == 2){
            ST7735_FillRect(0,130,128,8,ST7735_BLACK);
            ST7735_SetCursor(0, 13);
            printf("%s",string);
            ST7735_OutUDec(value);
        }
        else if(line == 3){
            ST7735_FillRect(0,150,128,8,ST7735_BLACK);
            ST7735_SetCursor(0, 15);
            printf("%s",string);
            ST7735_OutUDec(value);
        }
    }
}

______________________________________ADC DRIVER______________________________________

AddIndexFifo(ADCBuffer, 1000, uint32_t, FIFOSUCCESS, FIFOFAIL)

volatile uint32_t ADCvalue;

void ADC0Seq3_Handler(void){
  ADC0_ISC_R = 0x08;          // acknowledge ADC sequence 3 completion
    ADCvalue = ADC0_SSFIFO3_R;
    ADCBufferFifo_Put(ADCvalue);
}

uint32_t ADC_In(void){
    return ADCvalue;
}

uint16_t* ADC_Collect(uint32_t channelNum, uint32_t fs, uint16_t buffer[], uint32_t numberOfSamples) { int i = 0;
    ADCBufferFifo_Init();
    uint32_t value;
    uint32_t period = 0;
    period = (80000000 / fs);                   // Divide clock cycel by the specified frequency
    ADC_Open(channelNum, period);           //Use ADC_Open to properly open up the specified channel at the designmated frequency
    while(ADCBufferFifo_Size() != numberOfSamples){}
    uint32_t counter;
    for(counter = 0; counter < numberOfSamples; counter++){
        buffer[counter] = ADCBufferFifo_Get(&value);
    }
    return buffer;
}

______________________________________TIMER DRIVER______________________________________


int OS_AddPeriodicThread(void(*task)(void),unsigned long period,unsigned long priority){
    SYSCTL_RCGCTIMER_R |= 0x20;
    PeriodicTask = task;
    TIMER5_CTL_R = 0x00;                    //disable during setup
    TIMER5_CFG_R = 0x00;                    //32 bit mode
    TIMER5_TAMR_R = 0x02;                   //periodic mode
    TIMER5_TAILR_R = period - 1;    //requested reload value
    TIMER5_TAPR_R = 0x00;                   //bus clock resolution, no prescale
    TIMER5_ICR_R = 0x01;                    //clear timeout flag
    TIMER5_IMR_R = 0x01;                    //arm timeout interrupt
    NVIC_PRI23_R = (NVIC_PRI23_R&0xFFFFFF00) | 0x80; //priority 4   (need to change priority,
    maybe left shift 5 times?)
    NVIC_EN2_R = 0x10000000;            //enable IRQ 92
    TIMER5_CTL_R = 0x01;                    //enable timer 5A
    EnableInterrupts();
    return 0;
}

void Timer5A_Handler(void){
    TIMER5_ICR_R = 0x01;                    //acknowledge timeout
    PF1 = PF1^0x02;            // toggle red LED, PF1
    (*PeriodicTask)();
    PF1 = PF1^0x02;            // toggle red LED, PF1
}

void OS_ClearPeriodicTime(void){
    TIMER5_TAILR_R = 0;                     //resets counter to 0, TAILR register
}

unsigned long OS_ReadPeriodicTimer(void){
    return TIMER5_TAILR_R;
}

______________________________________INTERPRETER DRIVER______________________________________

void ProcessCommand(char *command){
    char commandType[COMMAND_MAX];
// Initialize commandType buffer
    for(int j = 0; j < COMMAND_MAX; j++) {
        commandType[j] = 0;
    }
    uint32_t i = 0;
    char commandNum;
    while(1) {
        if(command[i] == ' ')
        {
            break;
        }
        else if(command[i] == NULL) {
            break;
        }
        else {
            commandType[i] = command[i];
        }
        i++;
    }
    if (strcmp(commandType,"ADC") == 0){
        commandType[i] = ' ';
        i++;
        commandNum = 1;
    }
    if (strcmp(commandType,"Timer") == 0){
        commandType[i] = ' ';
        i++;
        commandNum = 2;
    }
    if (strcmp(commandType,"LCD") == 0){
        commandType[i] = ' ';
        i++;
        commandNum = 3;
    }
        switch(commandNum){
            case 1:
                uint32_t ADCValue = ADC_In();
                ST7735_Message(1,3,commandType, ADCValue);
                UART_OutUDec(ADCValue);
                break;
            case 2:
                while(command[i] != 0){
                    commandType[i] = command[i];
                    i++;
                }
                UART_OutString(commandType);
                break;
            case 3:
                while(command[i] != 0){
                    commandType[i] = command[i];
                    i++;
                }
                UART_OutString(commandType);
                ST7735_Message(0,3,commandType,0);
                break;
            default:
                ST7735_Message(2,1,"Default",1);
                UART_OutString(commandType);
                break;
        }
        i = 0;
}
```