

Ren Li Assignment Report

by Ren Li

FILE	12098_REN_LI_REN_LI_ASSIGNMENT_REPORT_168540_429723564.DO CX (9.41M)		
TIME SUBMITTED	15-MAR-2019 10:47 PM (UTC+0000)	WORD COUNT	2774
SUBMISSION ID	102498949	CHARACTER COUNT	34013

Critical and Real-Time Application Programming Assignment

Ren Li

1. Knowledge and Understanding Skills

The definition of real-time system is the system judged not only by the correctness of output result but also the time spent [1], where often a deadline time is specified to ensure its performance. For hard real-time system, the deadline is a strict requirement and the exceeding means failure of the total system and will cause serious consequence. Whereas for soft-time real-time, the completion within time limit is the optimal situation and the violation means decline in system quality rather than thorough failure. The loss of missing deadline is acceptable [2].

For car satnav system, to determine which type of real-time system it is we should analyse the consequence of not satisfying the deadline. When driving, the late reminder of waypoint may lead driver to the wrong direction (e.g. not turn left, miss the destination). However, if the updating does not occur too late and provides the correction route the driver can go back to the right way quickly, which will not cause serious consequences. Therefore, car satnavs are soft real-time systems. 

2. Practical Skills

To achieve the functionality of merging images, there are two functions being implemented, which are `mergeImgArray` and `mergeImgArrayWithMask`. The function prototype of former one is `"extern unsigned char* mergeImgArray (unsigned char foregroundImage[], unsigned char backgroundImage[], int length);"`, where first two arguments are foreground and background images to be merged. The third argument is the pixel amount, which is specified to 256 in the programme. This function is implemented to merge images without mask and will return the beginning address of result array. For `mergeImgArrayWithMask`, its function prototype is `"extern unsigned char* mergeImgArrayWithMask (unsigned char foregroundImage[], unsigned char backgroundImage[], int length, unsigned char bitmask[]);"`, the only difference is the forth argument, which denotes bitmask array. The return value is also the beginning address of result array. The code of programme will be appended to the appendix.

3. Cognitive and Intellectual Skills

As for a washing machines, there are many factors we should consider to make it work normally, like the motor should vibrate in an appropriate frequency, the volume of water should not be too small (clothes cannot be cleaned thoroughly) or too large (water will spill out of the roller), and the water temperature is also supposed to be suitable for both cleaning and clothes. Therefore, the washing machine system should be modelled carefully and be set corresponding timing constraints to guarantee these conditions. As the system can be regarded as the combination of many different modules like motor, heater, pump, we will research these components respectively and discuss the whole system then.

3.1 Motor Part

As for motor part, there are three relevant classes representing the “Motor”, “MotorController” and “Panel”. The Motor class is responsible for the mechanical movements of the washing machine like rotating to wash the clothes. The MotorController class takes charge of the operation of motor like starting, stopping, slowing down. The panel class is used to warn the users from safety issues like not be too closed when the machine is in high frequency rotating.

For the motor, there will be two scenarios: washing and drying. In washing scenario, it sends three events to the controller which are MotorStartedEvent, MotorOverSpeedEvent and WashingCompletedEvent. The MotorStartedEvent is to inform the controller that the motor has turned into the pre-set speed and started washing. The MotorOverSpeedEvent happens when the motor's speed is too fast and need to be lowered down by controller. The WashingCompletedEvent reports that washing is finished and motor can be stopped for a while. The time model of this scenario is following:

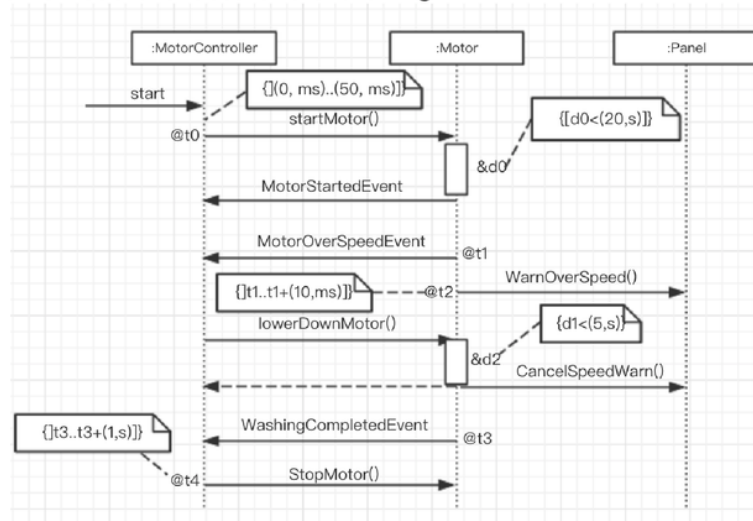


Figure1 Washing scenario time constraints

Firstly, the motor controller receives a start instruction and requires a maximum of 50ms to send a start command to motor. When the motor receives the start command, it should begin to speed up and achieve the pre-set frequency from interface module in a maximum of 20s. The maximum duration of 20s makes it will not cost too much time in this step and also insures that the motor will not speed up too fast to cause potential safety risk. And when the motor achieves the appropriate speed, the motor will send the MotorStartedEvent to motor controller to report the status. In addition, if sometime the speed of motor gets too fast when in service, firstly within 10ms it calls the WarnOverSpeed() method of Panel to display the overspeed information and the panel will warn users to be careful and keep away. Then the motor will send MotorOverSpeedEvent to controller and the controller will calls the lowerDownMotor() method to adjust the motor to the normal speed. The duration of adjusting should be as soon as possible and in a maximum of 5s to avoid any safety issues to users. When problem solved the motor will send a response to motor controller and calls CancelSpeedWarn() method of panel. Besides, for the end of washing, the motor will send a

WashingCompletedEvent to motor controller and the controller will call StopMotor() method of motor within 1s to make motor go off gradually.

In addition, for the scenario of drying, it is familiar with the washing scenario, but there are three differences between them. The first one is because that the overall weight of roller becomes lighter (there will be much less water compared to washing scenario), so the speeding up duration should be quicker and is 15s at most, and for the same reason the slowing down duration also decreases to 3s. Except for these two changes, the motor will send the DryingCompletedEvent to the motor control rather than WashingCompletedEvent. The time model of this scenario is following:

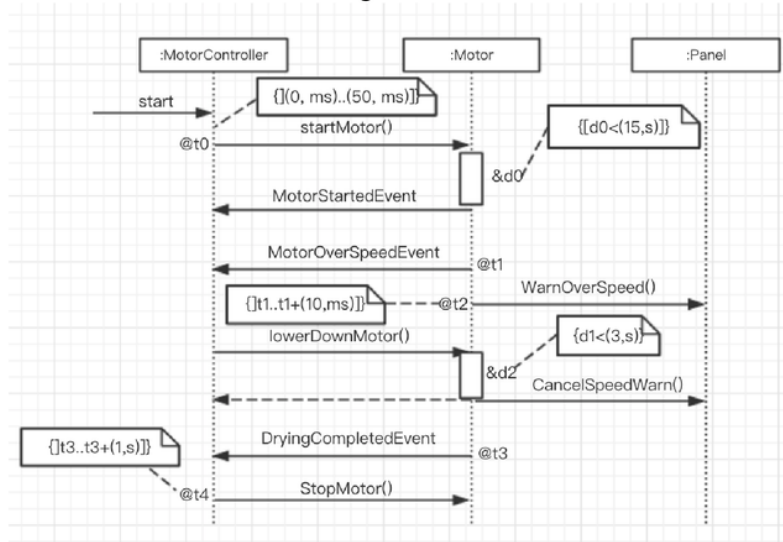


Figure2 Drying scenario time constraints

3.2 Heater Part

For heater part, there are three classes representing the "TemperatureController", "Heater" and the "TemperatureSensor". TemperatureController is responsible for controlling the heater to warm or cool the water. Heater is the heating machine to make the water in an appropriate temperature so clothes can be washed easily and maintained in a good condition. TemperatureSensor is to monitor the temperature and if it too high or too low the sensor will send corresponding event to controller to adjust. In this scenario there are three events which are "TemperatureReachedEvent", "TemperatureTooHighEvent" and "TemperatureTooLowEvent". They are all from the sensor to controller and used to report the temperature status so the controller can know what to do next. The time model of this scenario is below:

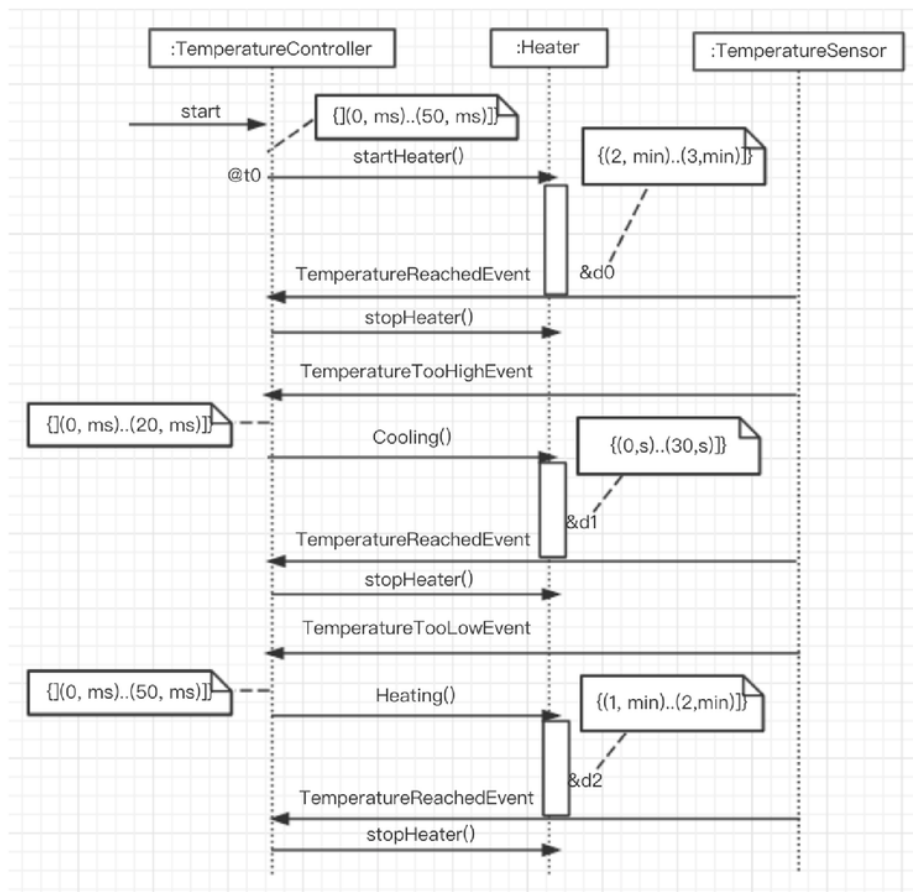


Figure3 Heating scenario time constraints

Firstly, a start instruction arrives in the temperature controller and a maximum of 50ms is required to calls the start method. Then the heater needs to complete the heating process between 2 to 3 minutes. The specification of 2 minute is to make sure that the heating process is smooth. If heating too fast, the roller can be unevenly heated and some areas' temperature will be much higher than others, which can cause damage to laundry. The specification of 3 minute is to avoid taking too much time to heat and influence the continuing steps. Then when temperature reaches the pre-set value, the sensor will send a `TemperatureReachedEvent` to controller and the controller will call `stopHeater()` method of Heater to stop heating. In addition, if sometime the sensor finds the temperature is too high, it will send a `TemperatureTooHighEvent` to controller. Within 20ms the controller will call `Cooling()` method to adjust the temperature. And the process should be completed within a maximum of 30s, to avoid high temperature damaging the clothes and risk of scalding user. Besides, if the temperature is lower than pre-set value, sensor will also send event called `TemperatureTooLowEvent` to controller. Because this problem is not very serious, so the time constraints will be looser. The controller should calls the `startHeater()` method within 50ms and the heating process should be finished between 1 and 2 minutes. After completing, the sensor will also send a `TemperatureReachedEvent` to controller to stop the heater.

3.3 Pump Part

For pump part, there are four relevant classes representing the "PumpController", "Sensor", "Pump" and the "Valve". Pump controller is to control the water to be pumped in or pumped out. The sensor is to measure the volume of water in roller. The valve is connected with water pipeline and only it is opened the pump can work normally. The time model of this scenario is below:

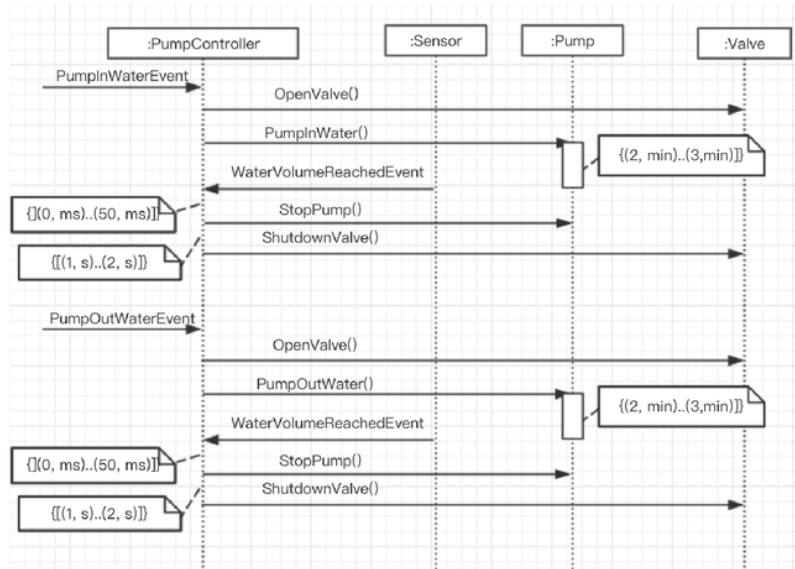


Figure4 Pump scenario time constraints

In this scenario, when pump controller receives the PumpInWaterEvent, it firstly calls OpenValve() method. Then it calls PumpInWater() method to make pump begin to pump in water. The pumping process should be completed between 2 and 3 minutes, to make sure the water will be pumped in with a stable speed. When sensor finds the volume of water is enough, it will send a WaterVolumeReachedEvent to pump controller. Then the controller will calls StopPump() method a maximum 50ms later. Because when the sensor sends the event there is enough water in roller and continuing pumping will add extra water. Therefore, the pump should be stopped as soon as possible. Then the valve should be closed between 1 and 2s. The spesification of 1s is to make sure pump is stopped when valve closes, or pump will be in a unnormal condition. The maximum of 2s is to make pumping process end quickly. In addition, when pump controller receives the PumpOutWaterEvent, the water will be pumped out of roller. The steps are familiar: calls OpenValve() and PumpOutWater() method first, when sensor detects that volume is reached the pre-set value, it will send WaterVolumeReachedEvent. Then controller will stop the pump and close the valve, which are the last two steps of pumping out water.

3.4 The Whole System

When the washing machine begins to work, the components introduced above will closely cooperate and finish the task. The whole wash and dry process will roughly be:

1. The user puts clothes in roller and presses the start button.
2. The machine door is locked, and pump starts to pump in water to a certain volume.

3. The heater starts to heating the water to the pre-set temperature.
4. Motor begins rotating to wash clothes.
5. When the washing finishes the motor will be closed.
6. The pump starts to pump out the water.
7. Motor begins rotating to dry the clothes.
8. The pump pumps out the water generated when drying.
9. Stop the pump and heater, unlock the wash machine door.

During these steps there will be many time constraints (discussed above), which guarantee the whole washing process can proceed normally and avoid the safety risks to users and any damages to clothes.

Reference

- [1] Shin, Kang G., and Parameswaran Ramanathan. "Real-time computing: A new discipline of computer science and engineering." Proceedings of the IEEE 82.1 (1994): 6-24.
- [2] "Real-time computing." Wikipedia: The Free Encyclopedia. Wikipedia, The Free Encyclopedia, 9 March 2019. Web. 15 March 2019, en.wikipedia.org/wiki/Real-time_computing

Appendix

main.c file

```
#include <asf.h>
```

```
unsigned char backgroundImage[] = {
```

```
    0xFF,0xFF,0xFF,0xFF,0xFF,0xFF,0xFF,0xFF,0xFF,0xFF,0xFF,0xFF,0xFF,0xFF,0
    xFF,0xFF,0xFF,0xFF,0xFF,0xFF,0xFF,0xFF,0xFF,0xFF,0xFF,0xFF,0xFF,0xFF,0
    xFF,0xFF,0xFF,0xFF,0xFF,0xFF,0xFF,0xFF,0xFF,0xFF,0xFF,0xFF,0xFF,0xFF,0
    xFF,0xFF,0xFF,0xFF,0xFF,0xFF,0xFF,0xFF,0xFF,0xFF,0xFF,0xFF,0xFF,0xFF,0
    xFF,0xFF,0xFF,0xFF,0xFF,0xFF,0xFF,0xFF,0xFF,0xFF,0xFF,0xFF,0xFF,0xFF,
```

```
    0xFF,0xFF,0xFF,0xFF,0xFF,0xFF,0xFF,0xFF,0xFF,0xFF,0xFF,0xFF,0xFF,0
    xFF,0xFF,0xFF,0xFF,0xFF,0xFF,0xFF,0xFF,0xFF,0xFF,0xFF,0xFF,0xFF,0
    xFF,0xFF,0xFF,0xFF,0xFF,0xFF,0xFF,0xFF,0xFF,0xFF,0xFF,0xFF,0xFF,0
    xFF,0xFF,0xFF,0xFF,0xFF,0xFF,0xFF,0xFF,0xFF,0xFF,0xFF,0xFF,0xFF,0
    xFF,0xFF,0xFF,0xFF,0xFF,0xFF,0xFF,0xFF,0xFF,0xFF,0xFF,0xFF,0xFF,0
    xFF,0xFF,0xFF,0xFF,0xFF,0xFF,0xFF,0xFF,0xFF,0xFF,0xFF,0xFF,0xFF,
```

```
    0xFF,0xFF,0xFF,0xFF,0xFF,0xFF,0xFF,0xFF,0xFF,0xFF,0xFF,0xFF,0xFF,0
    xFF,0xFF,0xFF,0xFF,0xFF,0xFF,0xFF,0xFF,0xFF,0xFF,0xFF,0xFF,0xFF,0
    xFF,0xFF,0xFF,0xFF,0xFF,0xFF,0xFF,0xFF,0xFF,0xFF,0xFF,0xFF,0xFF,0
    xFF,0xFF,0xFF,0xFF,0xFF,0xFF,0xFF,0xFF,0xFF,0xFF,0xFF,0xFF,0xFF,0
    xFF,0xFF,0xFF,0xFF,0xFF,0xFF,0xFF,0xFF,0xFF,0xFF,0xFF,0xFF,0xFF,0
    xFF,0xFF,0xFF,0xFF,0xFF,0xFF,0xFF,0xFF,0xFF,0xFF,0xFF,0xFF,0xFF,
```

```
    0xFF,0xFF,0xFF,0xFF,0xFF,0xFF,0xFF,0xFF,0xFF,0xFF,0xFF,0xFF,0xFF,0
    xFF,0xFF,0xFF,0xFF,0xFF,0xFF,0xFF,0xFF,0xFF,0xFF,0xFF,0xFF,0xFF,0
    xFF,0xFF,0xFF,0xFF,0xFF,0xFF,0xFF,0xFF,0xFF,0xFF,0xFF,0xFF,0xFF,0
    xFF,0xFF,0xFF,0xFF,0xFF,0xFF,0xFF,0xFF,0xFF,0xFF,0xFF,0xFF,0xFF,0
    xFF,0xFF,0xFF,0xFF,0xFF,0xFF,0xFF,0xFF,0xFF,0xFF,0xFF,0xFF,0xFF,0
    xFF,0xFF,0xFF,0xFF,0xFF,0xFF,0xFF,0xFF,0xFF,0xFF,0xFF,0xFF,0xFF,
```

```
    0xFF,0xFF,0xFF,0xFF,0xFF,0xFF,0xFF,0xFF,0xFF,0xFF,0xFF,0xFF,0xFF,0
    xFF,0xFF,0xFF,0xFF,0xFF,0xFF,0xFF,0xFF,0xFF,0xFF,0xFF,0xFF,0xFF,0
    xFF,0xFF,0xFF,0xFF,0xFF,0xFF,0xFF,0xFF,0xFF,0xFF,0xFF,0xFF,0xFF,0
    xFF,0xFF,0xFF,0xFF,0xFF,0xFF,0xFF,0xFF,0xFF,0xFF,0xFF,0xFF,0xFF,0
    xFF,0xFF,0xFF,0xFF,0xFF,0xFF,0xFF,0xFF,0xFF,0xFF,0xFF,0xFF,0xFF,0
    xFF,0xFF,0xFF,0xFF,0xFF,0xFF,0xFF,0xFF,0xFF,0xFF,0xFF,0xFF,0xFF,
```

```
    0xFF,0xFF,0xFF,0xFF,0xFF,0xFF,0xFF,0xFF,0xFF,0xFF,0xFF,0xFF,0xFF,0
    xFF,0xFF,0xFF,0xFF,0xFF,0xFF,0xFF,0xFF,0xFF,0xFF,0xFF,0xFF,0xFF,0
    xFF,0xFF,0xFF,0xFF,0xFF,0xFF,0xFF,0xFF,0xFF,0xFF,0xFF,0xFF,0xFF,
```

[illegible]

[illegible]

[illegible]

[illegible]

[illegible]

```

0xFF,0xFF,0x00,0x00,0xFF,0xFF,0x00,0x00,0xFF,0xFF,0x00,0x00,0xFF,0xFF,0x00,0x00,
0xFF,0xFF,0x00,0x00,0xFF,0xFF,0x00,0x00,0xFF,0xFF,0x00,0x00,0xFF,0xFF,0x00,0x00,0xFF,0x
FF,0x7F,0x7F,0xFF,0xFF,0x7F,0x7F,0xFF,0xFF,0x7F,0x7F,0xFF,0xFF,0x7F,0x7F,0xFF,0xFF,0x7F,
0x7F,0xFF,0xFF,0x7F,0x7F,0xFF,0xFF,0x7F,0x7F,0xFF,0xFF,0x7F,0x7F,
0xFF,0xFF,0x00,0x00,0xFF,0xFF,0x00,0x00,0xFF,0xFF,0x00,0x00,0xFF,0xFF,0x00,0x00,0xFF,0x
FF,0x7F,0x7F,0xFF,0xFF,0x7F,0x7F,0xFF,0xFF,0x7F,0x7F,0xFF,0xFF,0x7F,0x7F,0xFF,0xFF,0x7F,
0x7F,0xFF,0xFF,0x7F,0x7F,0xFF,0xFF,0x7F,0x7F,0xFF,0xFF,0x7F,0x7F,
0xFF,0xFF,0x00,0x00,0xFF,0xFF,0x00,0x00,0xFF,0xFF,0x00,0x00,0xFF,0xFF,0x00,0x00,0xFF,0x
FF,0x7F,0x7F,0xFF,0xFF,0x7F,0x7F,0xFF,0xFF,0x7F,0x7F,0xFF,0xFF,0x7F,0x7F,0xFF,0xFF,0x7F,
0x7F,0xFF,0xFF,0x7F,0x7F,0xFF,0xFF,0x7F,0x7F,0xFF,0xFF,0x7F,0x7F,
0xFF,0xFF,0x00,0x00,0xFF,0xFF,0x00,0x00,0xFF,0xFF,0x00,0x00,0xFF,0xFF,0x00,0x00,0xFF,0x
FF,0x7F,0x7F,0xFF,0xFF,0x7F,0x7F,0xFF,0xFF,0x7F,0x7F,0xFF,0xFF,0x7F,0x7F,0xFF,0xFF,0x7F,
0x7F,0xFF,0xFF,0x7F,0x7F,0xFF,0xFF,0x7F,0x7F,0xFF,0xFF,0x7F,0x7F,
0xFF,0xFF,0x00,0x00,0xFF,0xFF,0x00,0x00,0xFF,0xFF,0x00,0x00,0xFF,0xFF,0x00,0x00,0xFF,0x
FF,0x7F,0x7F,0xFF,0xFF,0x7F,0x7F,0xFF,0xFF,0x7F,0x7F,0xFF,0xFF,0x7F,0x7F,0xFF,0xFF,0x7F,
0x7F,0xFF,0xFF,0x7F,0x7F,0xFF,0xFF,0x7F,0x7F,0xFF,0xFF,0x7F,0x7F,
0xFF,0xFF,0x00,0x00,0xFF,0xFF,0x00,0x00,0xFF,0xFF,0x00,0x00,0xFF,0xFF,0x00,0x00,0xFF,0x
FF,0x7F,0x7F,0xFF,0xFF,0x7F,0x7F,0xFF,0xFF,0x7F,0x7F,0xFF,0xFF,0x7F,0x7F,0xFF,0xFF,0x7F,
0x7F,0xFF,0xFF,0x7F,0x7F,0xFF,0xFF,0x7F,0x7F,0xFF,0xFF,0x7F,0x7F,
0xFF,0xFF,0x00,0x00,0xFF,0xFF,0x00,0x00,0xFF,0xFF,0x00,0x00,0xFF,0xFF,0x00,0x00,0xFF,0x
FF,0x7F,0x7F,0xFF,0xFF,0x7F,0x7F,0xFF,0xFF,0x7F,0x7F,0xFF,0xFF,0x7F,0x7F,0xFF,0xFF,0x7F,
0x7F,0xFF,0xFF,0x7F,0x7F,0xFF,0xFF,0x7F,0x7F,0xFF,0xFF,0x7F,0x7F,
};

```

```

extern unsigned char* mergelmgArray(unsigned char foregroundImage[], unsigned char
backgroundImage[], int length); // length: pixel amount
extern unsigned char* mergelmgArrayWithMask(unsigned char foregroundImage[], unsigned
char backgroundImage[], int length, unsigned char bitmask[]);

```

```

int main (void)
{
    board_init();

    unsigned char* resultArray = mergelmgArray(foregroundImage, backgroundImage,
256);

    //unsigned char* resultArrayWithMask = mergelmgArrayWithMask(foregroundImage,
backgroundImage, 256, bitmaskBits);

    /* This skeleton code simply sets the LED to the state of the button. */
    while (1) {

```

```

        /* Is button pressed? */
        if (ioport_get_pin_level(BUTTON_0_PIN) == BUTTON_0_ACTIVE) {
            /* Yes, so turn LED on. */
            ioport_set_pin_level(LED_0_PIN, LED_0_ACTIVE);
        } else {
            /* No, so turn LED off. */
            ioport_set_pin_level(LED_0_PIN, !LED_0_ACTIVE);
        }
    }
}

```

mergelmgArray.s file

```

.section .data
ResultArray: .space 500

```

```

.section .text
.global mergelmgArray
.type mergelmgArray, %function

```

```

mergelmgArray:
    // r0: foreground array, r1: background array, r2: pixel amount & store temporaray
result
    push {r4, r5, r6, r7, lr}

```

```

    ldr r3, =0 // pixel loop times & subloop loop times
    ldr r4, =ResultArray // result array head address
    ldr r5, =0 // value to store in result array & store rgb value
    ldr r6, =0 // address offset
    ldr r7, =0 // alpha value

```

```

loop:
    cmp r3, r2
    bge endloop
    push {r2, r3}
    ldr r3, =0
    ldr r2, =0

    ldrb r7, [r0, r6] //store fore alpha
    ldr r2, =0xff
    strb r2, [r4, r6] // store result alpha
    add r6, #1

```

```

subloop:
    cmp r3, #3
    bge endsubloop

    push {r5}
    ldrb r5, [r0, r6] // caculate fore value

```

```

    mov r2, r7
    mul r2, r2, r5
    pop {r5}
    add r5, r2

    push {r5}
    ldrb r5, [r1, r6] // caculate back value
    mvn r2, r7 // caculate back alpha
    add r2, #1
    add r2, #255
    mul r2, r2, r5
    pop {r5}
    add r5, r2
    lsr r5, #8 // divide by 8
    strb r5, [r4, r6] // store one byte to memory

    // prepare for next subloop
    ldr r5, =0
    add r3, #1
    add r6, #1
    b subloop

endsubloop:
    pop {r2, r3}
    add r3, #1
    b loop

endloop:
    mov r0, r4
    pop {r4, r5, r6, r7, pc}

.end
.LTORG

```

mergelmgArrayWithMak.s file

```

.section .data
ResultArray: .space 500

.section .text
.global mergelmgArrayWithMask
.type mergelmgArrayWithMask, %function

mergelmgArrayWithMask:
    // r0: foreground array, r1: background array, r2: pixel amount & store temporaray
    result, r3: bit mask array

```

```

    push {r4, r5, r6, r7, lr}

    ldr r4, =ResultArray // result array head address
    ldr r5, =0 // value to store in result array & store rgb value
    ldr r6, =0 // address offset
    ldr r7, =0 // alpha value

loop:
    cmp r5, r2
    bge endloop

    // get current mask byte
    // r4, r6 are used to store temporary values
    push {r2}
    push {r4, r6}
    mov r4, r5
    mov r6, r5

    lsr r4, #3
    ldrb r2, [r3, r4] // get mask byte
    mov r5, r6
    pop {r4, r6}

    b caltmaskbit // get current mask bit and return value in r2
endmask:
    ldrb r7, [r0, r6]
    mul r7, r2 // caculate the mask alpha

    pop {r2}
    push {r2, r5}

    ldr r2, =0xff
    strb r2, [r4, r6] // store result alpha
    add r6, #1

subloop1:
    ldr r5, =0

    // caculate fore value
    push {r5}
    ldrb r5, [r0, r6]
    mov r2, r7
    mul r2, r2, r5
    pop {r5}
    add r5, r2

    // caculate back value

```

```

push {r5}
ldrb r5, [r1, r6]
mvn r2, r7 // caculate back alpha
add r2, #1
add r2, #255
mul r2, r2, r5
pop {r5}
add r5, r2

lsr r5, #8 // divide by 256
strb r5, [r4, r6] // store one byte to memory

add r6, #1 // offset address +1

```

subloop2:

```

ldr r5, =0

// caculate fore value
push {r5}
ldrb r5, [r0, r6]
mov r2, r7
mul r2, r2, r5
pop {r5}
add r5, r2

// caculate back value
push {r5}
ldrb r5, [r1, r6]
mvn r2, r7 // caculate back alpha
add r2, #1
add r2, #255
mul r2, r2, r5
pop {r5}
add r5, r2

lsr r5, #8 // divide by 256
strb r5, [r4, r6] // store one byte to memory

add r6, #1 // offset address +1

```

subloop3:

```

ldr r5, =0

// caculate fore value
push {r5}
ldrb r5, [r0, r6]

```

```

    mov r2, r7
    mul r2, r2, r5
    pop {r5}
    add r5, r2

    // caculate back value
    push {r5}
    ldrb r5, [r1, r6]
    mvn r2, r7 // caculate back alpha
    add r2, #1
    add r2, #255
    mul r2, r2, r5
    pop {r5}
    add r5, r2

    lsr r5, #8 // divide by 256
    strb r5, [r4, r6] // store one byte to memory

    // prepare for next loop
    add r6, #1 // add offset

endsubloop:
    pop {r2, r5}
    add r5, #1
    b loop

endloop:
    mov r0, r4
    pop {r4, r5, r6, r7, pc}

// get the coressponding bit of mask
// r2 is mask byte
caltmaskbit:
    push {r3, r4, r6, r7}
    mov r3, r5
    mov r4, #7
    and r3, r4 // get the lowest three bits
    mov r4, #7
    sub r4, r3 // get the number of bit need to shift
    mov r7, r2
    lsr r7, r4 // move the bit to first
    ldr r6, =1
    and r7, r6 // get the bit
    mov r2, r7
    pop {r3, r4, r6, r7}
    b endmask

```

.end
.LTORG

Ren Li Assignment Report

GRADEMARK REPORT

FINAL GRADE

77 / 100

GENERAL COMMENTS

Instructor

Knowledge (20): 15

Cognitive (40): 35

Practical (40): 27

You have provided reasonable arguments for identifying the sat nav system as being a soft real-time system. You could have provided a little more comparison of hard/soft RT systems.

Very good analysis of the requirements. The sequence diagrams should have been presented in the second assignment but the analysis is quite detailed. Instead, I was looking for a more requirements analysis. However, I like what you have done and will mark it accordingly.

C code uses image arrays although there is no software testing, instead the results were checked by visual inspection.

Two assembler functions called from C, with and without the bitmask being used.

Assembler:

Alpha blending equation provided but the code provided is very difficult to understand and not well commented.

A similar readability problem occurs with the bitmask handling code. For example, getting the individual bits from the mask is a bit over complicated and the code for applying alpha blending is duplicated 3

times once for each colour (should have used a function).

Generally I think you understand assembler but the structure of the code is not as good as it should be which results in readability issues.

PAGE 1



Comment 1

The conclusion and analysis are good.

PAGE 2



Comment 2

means

PAGE 3

PAGE 4

PAGE 5

PAGE 6

PAGE 7

PAGE 8

PAGE 9

PAGE 10

PAGE 11

PAGE 12

PAGE 13

PAGE 14

PAGE 15
