# Software Assignment 4

Matt Bailey and Luis Gomez

## Part 1

### Behavior Description

The program reads and an 8 bit unsigned value input from port 0x9A. It then divides it by 3, and then outputs the quotient to port 0x42 and the remainder to port 0x43.

### Flow Chart

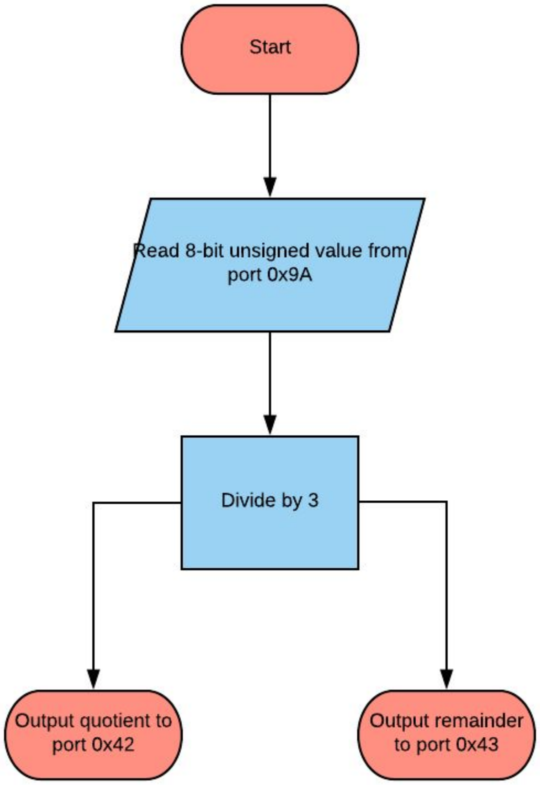
Start
Read 8-bit unsigned value from port 0x9A
Divide by 3
Output quotient to port 0x42
Output remainder to port 0x43

*Figure 1: Software flowchart for RAT MCU*

# Software Assignment 4
# Matt Bailey and Luis Gomez

## Verification

Table 1 shows 7 test cases that can prove the proper operation of the code. The test cases covers an input that is divided by 3 including upper limit 0xFF and lower limit 0x00

| Value | Output | Remainder | Explanation |
|---|---|---|---|
| 0x00 | 0x00 | 0x00 | Lower edge case |
| 0x03 | 0x01 | 0x00 | Divided by itself |
| 0x05 | 0x01 | 0x02 | Divided by a value that will give a remainder |
| 0x09 | 0x03 | 0x00 | Divided by a value that will not give a remainder |
| 0x11 | 0x05 | 0x02 | Divided by a value that will give a remainder |
| 0x12 | 0x06 | 0x00 | Divided by a value that will not give a remainder |
| 0xFF | 0x55 | 0x00 | Higher edge case |

*Table 1: Test Cases*

## Assembly Source Code

```
;---------------------------------------------------------------------
; Software Assignment 4
; Author: Matt Bailey and Luis Gomez
; Date: 02/04/19
; Description : Reads a value in from 0x9A and divides it by 3, and outputs the quotient to
; 0x42 and the remainder to 0x43
;--------------------------------------------------------------------
.EQU COUNTER = 0x00
.CSEG
.ORG 0x01
start:  IN R1, 0x9A
        MOV R3, COUNTER
        MOV R2, R1
loop:   MOV R2, R1
        MOV R4, R3
        ADD R3, 0x01
        SUB R1, 0x03
        BRCC loop
        OUT R2, 0x43
        OUT R4, 0x42
```

# Software Assignment 4

Matt Bailey and Luis Gomez

## Part 2

### Behavior Description

The program reads and an 8 bit unsigned value input from port 0x30. If the input is a multiple of 4, invert all bits, otherwise if input value is is odd, add 17 and divide the result by 2, otherwise subtract 1 from the value. Output the result to port 0x42.

### Flow Chart

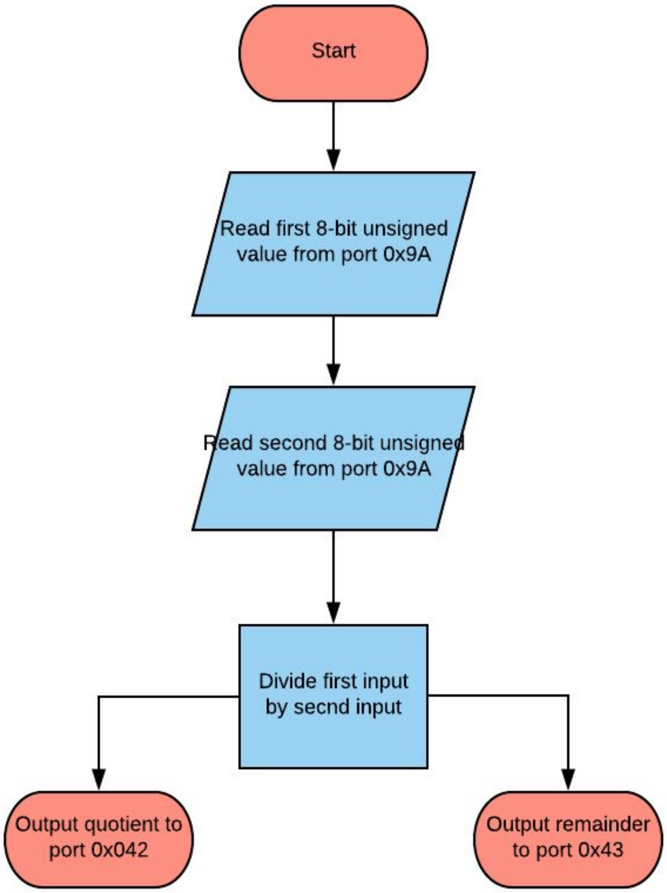
Start
Read first 8-bit unsigned value from port 0x9A
Read second 8-bit unsigned value from port 0x9A
Divide first input by secnd input
Output quotient to port 0x042
Output remainder to port 0x43

*Figure 2: Software flowchart for RAT MCU*

Software Assignment 4

Matt Bailey and Luis Gomez

## Verification

Table 2 shows 6 test cases that can prove the proper operation of the code. The test cases cover inputs that are divided by each other including upper limit 0xFF and lower limit 0x00

| Value #1 Hex | Value #2 Hex | Out Hex | Remainder Hex | Explanation |
|---|---|---|---|---|
| 0x00 | 0x01 | 0x00 | 0x00 | Lower Edge Case |
| 0x14 | 0x05 | 0x04 | 0x00 | Testing no remainder |
| 0x35 | 0x06 | 0x08 | 0x05 | Testing with remainder |
| 0x2A | 0x0F | 0x02 | 0x0C | Testing high remainder |
| 0x78 | 0x05 | 0x18 | 0x00 | Testing with high quotient |
| 0xFF | 0x11 | 0x0F | 0x00 | High Edge Case |

*Table 2: Test Cases*

# Software Assignment 4
# Matt Bailey and Luis Gomez

## Assembly Source Code

```
;-------------------------------------------------------------------
; Software Assignment 4
; Author: Matt Bailey and Luis Gomez
; Date: 02/04/19
; Description : Read two 8-bit unsigned values from port_id 0x9A. Divide the first value by the
2nd ; and output the quotient (result)  to port_id 0x42 and the remainder to port_id 0x43.
;------------------------------------------------------------------
.EQU COUNTER = 0x00
.CSEG
.ORG 0x01
start:
        IN R1,  0x9A ; first input
        IN R2,  0x9A ; second input
        CMP R2, 0x00
        BREQ start
        MOV R3, COUNTER ; initializing counter
loop:
        MOV R4, R2 ; save the incoming first value
        MOV R5, R1 ; save the incoming second value
        SUB R1, R2 ; subtract first value from the second
        BREQ zero_output
        BRCS output
        ADD R3, 0x01 ; add one to the counter
        BRCC loop
zero_output:
        ADD R3, 0x01 ; add one to the counter
        MOV R5, COUNTER ;clears remainder
output:
        OUT R3, 0x42
        OUT R5, 0x43
```