Assembly Calculator

John McAvoy

10 December 2018

1 Assembly Functions

```
#####################################################
# @file setup_uart.s                                #
# @author John McAvoy                               #
# @desc exports "setup_uart" assembly function      #
#####################################################


.equ eUSCI_A0,  0x40001000  @ eUSCI_A0 base address

# eUSCI_A0 Offsets
.equ UCA0CTLW0, 0x00        @ eUSCI_A0 Control Word 0
.equ UCA0CTLW1, 0x02        @ eUSCI_A0 Control Word 1
.equ UCA0BRW  , 0x06        @ eUSCI_A0 Baud Rate Control
.equ UCA0MCTLW, 0x08        @ eUSCI_A0 Modulation Control
.equ UCA0STATW, 0x0A        @ eUSCI_A0 Status
.equ UCA0RXBUF, 0x0C        @ eUSCI_A0 Receive Buffer
.equ UCA0TXBUF, 0x0E        @ eUSCI_A0 Transmit Buffer
.equ UCA0ABCTL, 0x10        @ eUSCI_A0 Auto Baud Rate Control
.equ UCA0IRCTL, 0x12        @ eUSCI_A0 IrDA Control
.equ UCA0IE   , 0x1A        @ eUSCI_A0 Interrupt Enable
.equ UCA0IFG  , 0x1C        @ eUSCI_A0 Interrupt Flag
.equ UCA0IV   , 0x1E        @ eUSCI_A0 Interrupt Vector

.global setup_uart @ exports setup_uart asm function to be linked to C code

.text
setup_uart:

    ldr     %r1, =eUSCI_A0   @ load eUSCI_A0 base address to R1

    # enabled, smclK, async, UART auto baud rate mode, one-stop bit, 8-bit char, MSB first, no parity
    mov     %r3, #0x26c1             @ table 24-8 of user guide
    strh    %r3, [%r1, #UCA0CTLW0]  @ configure UCA0CTLW0

    # deglitch t0x26c1ime ~5ns
    mov     %r3, #0x00               @ table 24-13 of user guide
    strh    %r3, [%r1, #UCA0CTLW1]  @ configure UCA0CTLW1

    # alternating second stage modulation, 4-th bit irst stage modulation, oversampling
    mov     %r3, #0xAA81             @ table 24-11 of user guide
    strh    %r3, [%r1, #UCA0MCTLW]  @ configure UCA0MCTLW

    # enable rx interrupt
    mov     %r3, #0x01               @ table 24-17 of user guide
    strh    %r3, [%r1, #UCA0IE]     @ configure UCA0IE
```

Listing 1: setup_uart.s

```
#####################################################
# @file read_rx_buffer                              #
# @author John McAvoy                               #
# @desc exports "read_rx_buffer" assembly function  #
#####################################################

.equ eUSCI_A0,  0x40001000  @ eUSCI_A0 base address

# eUSCI_A0 Offsets
.equ UCA0RXBUF, 0x0C        @ eUSCI_A0 Receive Buffer

.global read_rx_buffer      @ exports read_rx_buffer asm function to be linked to C code

.text
read_rx_buffer:

    ldr     %r1, =eUSCI_A0          @ load eUSCI_A0 base address to R1
    ldrb    %r0, [%r1, #UCA0RXBUF]  @ load UCA0RXBUF byte to R0, which is returned
```

Listing 2: read_rx_buffer.s

```
#####################################################
# @file write_tx_buffer                             #
# @author John McAvoy                               #
# @desc exports "write_tx_buffer" assembly function #
#####################################################

.equ eUSCI_A0,  0x40001000  @ eUSCI_A0 base address

# eUSCI_A0 Offsets
.equ UCA0TXBUF, 0x0E        @ eUSCI_A0 Transmit Buffer

.global write_tx_buffer     @ exports read_rx_buffer asm function to be linked to C code

.text
write_tx_buffer:

    ldr     %r1, =eUSCI_A0          @ load eUSCI_A0 base address to R1
    strb    %r0, [%r1, #UCA0TXBUF]  @ store R0 byte to UCA0TXBUF
```

Listing 3: write_tx_buffer.s

2 C Functions

3 Headers

```
/**
 * @file Equation.h
 * @author John McAvoy
 */

#ifndef EQUATION_H
#define EQUATION_H

#include <stdint.h>

// a <op> b
// EX: 3 + 4
typedef struct equation {
    int32_t a;
    char op;
    int32_t b;
} Equation;

#endif // EQUATION_H
```

Listing 4: Equation.h

```c
/**
 * @file calculator.h
 * @author John McAvoy
 */

#ifndef CALCULATOR_H
#define CALCULATOR_H

#include <stdint.h>
#include "Equation.h"

/**
 * @func calculate
 * @param equation_str  a pointer to a char array of an equation chars
 * @param length        length of equation_str
 * @returns the solution to the equation
 */
int32_t calculate(Equation eq);

/**
 * @func calculate
 * @param buffer        a pointer to a char array of an integer
 * @param length        length of buffer
 * @returns the character representation of the integer
 */
int32_t charBuffer2Int(char *buffer, uint8_t length);

/**
 * @func calculates power of number
 * @param a             uint8_t
 * @param pow           uint8_t a raised to power
 * @returns a to the p power
 */
int32_t power(uint8_t a, uint8_t p);

#endif // CALCULATOR_H
```

Listing 5: calculator.h

```c
/**
 * @file uart.h
 * @author John McAvoy
 */

#ifndef UART_H
#define UART_H

#include <stdint.h>
#include "Equation.h"

void setup_uart(void);           // see: setup_uart.s
char read_rx_buffer(void);       // see: read_rx_buffer.s
void write_tx_buffer(char c);    // see: write_tx_buffer.s

/**
 * @func parseInput
 * @param buffer        a char array of an equation
 * @param length        length of buffer
 * @returns a parsed Equation struct
 */
Equation parseInput(char buffer[], uint8_t length);


/**
 * @func send_bytes
 * @param buffer        a char array of message to be sent via UART
 * @param length        length of buffer
 */
void send_bytes(char buffer[], uint8_t length);

#endif // UART_H
```

Listing 6: uart.h

4 Implementations

```c
/**
 * @file calculator.c
 * @author John McAvoy
 */

#include <stdint.h>
#include "msp432p401r.h"
#include "Equation.h"
#include "calculator.h"

/**
 * @func calculate
 * @param equation_str  a pointer to a char array of an equation chars
 * @param length        length of equation_str
 * @returns the solution to the equation
 */
int32_t calculate(Equation eq) {
    switch(eq.op){
        case '+' : return eq.a +  eq.b;
        case '-' : return eq.a -  eq.b;
        case '*' : return eq.a *  eq.b;
        case '/' : return eq.a /  eq.b;
        case '^' : return eq.a ^  eq.b;
        case '|' : return eq.a |  eq.b;
        case '&' : return eq.a &  eq.b;
        case '>' : return eq.a >> eq.b;
        case '<' : return eq.a << eq.b;
    }
}

int32_t charBuffer2Int(char *buffer, uint8_t length){
    uint8_t sign = 1;
    if(*(buffer) == '-'){
        sign = -1;
        *(buffer) = '0';
    }

    int32_t num = 0;
    for (uint8_t i = sign; i < length; i++){
        num += (*(buffer + i) - '0') * power(10, (length - i));
    }

    return num * sign;
}

int32_t power(uint8_t a, uint8_t p){
    if(p == 0) {
        return 1;
    }
    else if(p == 1) {
        return a;
    }
    else {
        return a * power(a, p-1);
    }
}
```

Listing 7: calculator.c

```c
/**
 * @file uart.c
 * @author John McAvoy
 */

#include "msp432p401r.h"
#include "uart.h"
#include "Equation.h"
#include "calculator.h"

extern void setup_uart(void);           // see: setup_uart.s
extern char read_rx_buffer(void);       // see: read_rx_buffer.s
extern void write_tx_buffer(char c);    // see: write_tx_buffer.s

Equation parseInput(char buffer[], uint8_t length){
    Equation eq;
    eq.a = 0;
    eq.b = 0;

    uint8_t op_i = 0; // operator index
    for(uint8_t i = 0; i < length; i++){
        if(buffer[i] == ' ' && op_i == 0){
            // operator index found
            op_i = i + 1;
        }
    }

    eq.op = buffer[op_i];
    eq.a = charBuffer2Int(&buffer[0], op_i - 2);
    eq.b = charBuffer2Int(&buffer[op_i+2], length - op_i -1);

    return eq;
}

void send_bytes(char buffer[], uint8_t length){
    for(uint8_t i = length - 1; i >= 0; i--){
        while(!(EUSCI_A0->IFG & EUSCI_A_IFG_TXIFG)); // wait for TX to send
        write_tx_buffer(buffer[i]); // send char
    }
}
```

Listing 8: uart.c

```c
/**
 * @file main.c
 * @author John McAvoy
 */

#include <stdint.h>
#include <stdio.h>
#include "msp432p401r.h"
#include "calculator.h"
#include "uart.h"

const char CALCULATION_DELIMINATOR = '\n'; // sets character code for end of calculation string
char input_buffer[256]; // sets char buffer for input from UART
uint8_t input_counter = 0; // counts chars added to input buffer

int main() {

    setup_uart();

    return 0;
}

// USCIA0_RX Interrupt Handler
void EUSCIA0_IRQHandler(void) {

    if (EUSCI_A0->IFG & EUSCI_A_IFG_RXIFG) {
        // if UA0RX flag set

        char rx_in = read_rx_buffer(); // read byte

        if(rx_in == '\n'){
            // end of equation
            input_counter = 0;
            Equation eq = parseInput(input_buffer, input_counter);

            char out_buffer[32];
            sprintf(out_buffer, "%d\n", calculate(eq));
            send_bytes(out_buffer, 32);
        }
        else
            input_buffer[input_counter++] = rx_in;
    }
}
```

Listing 9: main.c

5 Build Files

```
PROJECT_NAME=assembly_calculator
BOARD=msp432401R

SRC_DIR ?= src
OBJ_DIR ?= obj
BIN_DIR ?= bin

CC=arm-none-eabi-gcc
CFLAGS ?= -Ilib/TI/Include -Ilib/CMSIS/Include --specs=nosys.specs
AS=arm-none-eabi-as

TARGET=$(BIN_DIR)/$(BOARD)_$(PROJECT_NAME).elf

SRCS := $(shell find $(SRC_DIR) -name *.c -or -name *.s)

$(TARGET): $(OBJS)
	$(CC) $(CFLAGS) $(LDFLAGS) $(SRCS) -o $@

$(OBJ_DIR)/%.o: $(SRCS)
	$(CC) $(CFLAGS) $(LDFLAGS) $(SRCS) -o $@

.PHONY: clean
clean:
	@rm -rf bin/* src/*.o

run:
	@echo "mspdebug tilib "prog $(BINDIR)/$(TARGET)""
	@mspdebug tilib "prog $(BINDIR)/$(TARGET)"
```

Listing 10: Makefile