Review for Final Exam

Numbers

– Decimal to Hex (signed and unsigned)
– Hex to Decimal (signed and unsigned)
– Binary to Hex
– Hex to Binary
– Addition and subtraction of fixed-length hex numbers
– Overflow, Carry, Zero, Negative bits of CCR

Programming Model

– Internal registers – A, B, (D = AB), X, Y, SP, PC, CCR

Addressing Modes and Effective Addresses

– INH, IMM, DIR, EXT, REL, IDX (Not Indexed Indirect)
– How to determine effective address

Instructions

– What they do - User Guide
– What machine code is generated
– How many cycles to execute
– Effect on CCR
– Branch instructions – which to use with signed and which with unsigned

Machine Code

– Reverse Assembly

Stack and Stack Pointer

– What happens to stack and SP for instructions (e.g., PSHX, JSR)
– What happens to stack and SP for interrupt
– What happens to stack and SP when program leaves an interrupt service routine

Assembly Language

– Be able to read and write simple assembly language program
– Know basic pseudo-ops – e.g., equ, dc.b, ds.w
– Flow charts

C Programming

– Setting and clearing bits in registers

```
PORTA = PORTA | 0x02;
PORTA = PORTA & ~0x0C;
```

– Using pointers to access specific memory location or port.

```
* (unsigned char *) 0x0400 = 0xaa;
#define PORTX (* (unsigned char *) 0x4054)
PORTX = 0xaa;
```

Interrupts

– Interrupt Vectors (and reset vector)
    How to set interrupt vector in C
– How do you enable interrupts (specific mask and general mask)
– What happens to stack when you receive an enabled interrupt
– What happens when you leave ISR with RTI instruction?
– What setup do you need to do before enabling interrupts?
– What do you need to do in interrupt service routine (clear source of interrupt, exit with RTI instruction)?
– How long (approximately) does it take to service an interrupt?

Enhanced Capture Timer module

– Enable Timer
– Timer Prescaler
    How to set
    How it affects frequency of timer clock
– Timer Overflow Interrupt
– Input Capture
– Output Compare
– How to enable interrupts in the timer subsystem
– How to clear flags in the timer subsystem
– Be able to look at timer registers and determine how timer is set up
    Which channels are being used
    Which are being used for Input Capture, which for Output Compare
– How to time differences from Timer counts

Real Time Interrupt

– How to enable
– How to change rate
– How to enable interrupt
– How to clear flag

Pulse Width Modulation

– How to get into 8-bit, left-aligned high-polarity mode
– Calculate how many clock periods it takes to get desired PWM period (frequency)
– How to set PWM period (frequency)
    Using Clock Mode 0
    Using Clock Mode 1
– How to set PWM duty cycle
– How to enable PWM channel
– Be able to look at PWM registers and determine PWM frequency and duty cycle

A/D Converter

– How to power-up A/D converter (ATD1CTL2)
– Write 0x05 to ATD1CTL4 to set at fastest conversion speed and 10-bit conversions
– Write 0x85 to ATD1CTL4 to set at fastest conversion speed and 8-bit conversions
– Select type of conversion sequence and the analog channels sampled, in (ATD1CTL5)
    Right/left justified
    signed/unsigned
    Continuous Scan vs. Single Scan
    Multichannel vs. Single Channel conversions
– How to tell when conversion is complete - ATD1STAT0 register
– How to read results of A/D conversions – ATD1DR0H (8-bit conversions)
– How to read results of A/D conversions – ATD1DR0 (10-bit conversions)
    Be able to convert from digital number to voltage, and from voltage to digital number (need to know VRH and VRL).

SPI

– Pins used – SCLK, MOSI, MISO, SS
– Difference of use in Master and Slave mode
– SPI0CR1 Register
    Enable SPI
    Master or Slave
    Enable interrupts
    Clock polarity
    Clock phase
    Baud rate
    Automatically operate SS for single-byte transfers
    Control SS line manually using Port S (PTS)
    LSB or MSB first
– SPI0CR2 Register— always 0 (normal mode)
– SPI0BR Register— Set speed (master only)
– SPI0SR Register — SPIF and SPTEF flags - clear SPI flag by reading
    SPI0SR, then access (read or write) SPI0DR
– SPI0DR Register —shift register – master starts transfer by writing data to SPI0DR

Interfacing

– Getting into expanded mode — MODA, MODB, BKGD pins of MODE Register
– PEAR Register —enable ECLK, LSTRB, R/W on external pins
– Ports A and B in expanded mode
    Port A – AD 15-8 (Port A is for data for high byte, even addresses)
    Port B – AD 7-0 (Port B is for data for low byte, odd addresses)
– E clock
    Address on AD 15-0 when E low, Data on AD 15-0 when E high
    Need to latch address on rising edge of E clock
    On write (output), external device latches data on signal initiated by falling edge of E
    On read (input), HCS12 latches data on falling edge of E
    E-clock stretch - MISC register
– R/W Line
– LSTRB line
– Single-byte and two-byte accesses
    16-bit access of even address – A0 low, LSTRB low – accesses even and odd bytes
    8-bit access of even address – A0 low, LSTRB high – accesses even byte only
    8-bit access of odd address – A0 high, LSTRB low – accesses odd byte only
– Address decoding

SCI

– Select baud rate
– Select word length and parity
– Enable transmitter
– Testing and clearing TDRE flag

Pulse Accumulators

– How to select PACA
    in Event count mode or Gated time accumulation mode
– Select edge
– Enable interrupt