PIC32 Study Problems

Many of the questions on this study guide refer to the documentation on the "Introduction to the PIC32" page http://hades.mech.northwestern.edu/index.php/Introduction_to_the_PIC32 on the Mechatronics Wiki. Other sources that may be useful are the code that installed with the MPLAB C Compiler for PIC32; the PIC32MX Reference Manual; the PIC32MX Data Sheet; and the PIC32 Peripheral Libraries Manual. These are all referred to by the Intro page. This study guide approximately follows the Intro page.

PIC32 Hardware

1. Practice with base 2 (binary), base 10 (decimal), and base 16 (hexadecimal):
    a. Write 0xB15F as a binary number and in base 10.
    b. Write 207 (base 10) as an 8-bit binary number and a 2 character hex number.
    c. Write the result of the bitwise OR 0xA2 | 0x46 in hex.
    d. Write the result of the bitwise AND 0x2C & 0xE3 in hex.
    e. Write the result of the shift operation 0x47 >> 3 in binary and hex.
    f. Write the result of the shift operation 122 << 0x02 in binary and hex.
2. Go to the Microchip homepage, check out the Parametric Table of PIC32's, and find a PIC with the following specs: 80 MHz max clock speed, 128 K flash (program memory), 32 K RAM (data memory), 4 channels of DMA, 16 A/D converters with 10-bit resolution, no USB capabilities, 2 comparators, 5 16-bit timers and 1 32-bit timer, 5 PWM channels and 5 input compare channels, 2 UARTs, 2 SPI, and 2 I2C pins, no CAN modules, and 64 pins. What is the part number? What types of packages does it come in (e.g., DIP, or different kinds of surface mount packages, etc.)? How much does it cost in quantity 1? What is the difference in price and features from our PIC, the PIC32MX460F512L?
3. Look at the PIC32 architecture figure showing the SYSCLK and the PBCLK. In one sentence each, without going into detail, explain the basic function of the following items shown in the block diagram: Timing Generation; MCLR; SYSCLK; PBCLK; USBCLK; PORTA...G (and indicate which of these can be used for analog input on our PIC); Timer{1..5}; 10-bit ADC; Comparators; UART 1,2; I2C 1,2; SPI 1,2; IC 1,5; PWM OC 1,5; CN1-22; Data RAM; Program Flash Memory; and Pre-Fetch Module.
4. What does "peripheral" mean for the PIC?
5. Most pins on the PIC have many different labels, according to the functions they serve. Some of the most important to us are the following: ANx (x=0...15), INTx (x=0...4), OCx (x=1...5), Rxy (x=A...G, y=0...15), SCLx, SDAx (x=1,2), SCKx, SDIx, SDOx (x=1,2), SSx (x=1,2), TxCK (x=1...5), UxCTS, UxRTS, UxRX, UxTX (x=1,2), VDD, VSS. Explain, in one sentence each, what these functions are.
6. Describe the four functions that pin 21 of the PIC32 can have.
7. How many pins are on our DIP-like NU32 board? How many of these are directly connected to the PIC? How many pins on the PIC are left disconnected? The remainder of the PIC's pins are used internally on the board. How many is that?
8. If the NU32 board is powered by a USB cable, what voltage does the USB cable provide? What voltage is used to power the PIC on the NU32 board? How do we get the PIC supply voltage from the USB voltage?
9. On the NU32 board, there are 4 LEDs connected to 4 digital outputs. Are the LEDs on when the digital outputs are high or low?

PIC32 Software

These are the primary Microchip folders and files we are interested in. Remember these summaries of what's in each of them:

- C:\Program Files\MPLAB C32\pic32-libs\include
  Contains the file and two folders below (among other things):
  - plib.h: the header file which includes the header files for the peripherals.
  - peripheral: a folder of the .h header files for peripheral library functions for all the peripherals: adc10.h, i2c.h, ports.h, etc.
  - proc: a folder that contains one .h header file for each particular PIC32. These files make processor-dependent definitions. Ours is p32mx460f512l.h.
- C:\Program Files\MPLAB C32\pic32-libs\peripheral
  Contains folders named adc10, i2c, ports, etc., with source code for each peripheral. Many of these are empty, because the .h files provide all we need to use the peripheral.

10. When you first started programming your PIC, you put a "bootloader" on it. What is the purpose of the bootloader?
11. The main.c file for the bootloader sets the behavior of the timing generation circuit that takes output of the 8 MHz crystal used on the NU32 board and creates an 80 MHz SYSCLK. Which of the #pragma commands in main.c defines this conversion from 8 MHz to 80 MHz? (You can refer to Register 6-1 of Section 6 of the Reference Manual.)
12. Give one reason the linker file procdefs.ld for the bootloader project is different from the linker file procdefs.ld for your first simple program, "Hello World."
13. In general, what is the purpose of a Special Function Register (SFR) for a peripheral?
14. What is the purpose of the file HardwareProfile_NU32.h? It refers to LATE, TRISE, and PORTE. What are these, and how are they used in this file?
15. What does the header file plib.h do?
16. In what directory is adc10.h? What does the "10" in this file name signify?
17. The file p32mx460f512l.h contains a number of definitions for our particular PIC32. In which directory is it?
18. (There is a long buildup for these questions, so just answer the questions **in bold**, which are not only at the end.) We will learn about some of the SFRs defined for the ADC. In the file p32mx460f512l.h, search for the first instance of AD1CON1. (Should be about 10% into the file.) It is a variable defined as an "extern volatile unsigned int." This is just a variable, just like any other variable you might define, but it represents a "Special Function Register" (SFR) that helps to determine the function of the ADC. "extern" means that this variable is available to other files (its *scope* is not just limited to use in this file). "volatile" tells the CPU to reload the value from data memory into the CPU registers before doing any computing on it, in case some process the CPU doesn't control has changed AD1CON1. (You will rarely/never need to worry about this for variables you define.) "int" means that it is a 32-bit integer, the default integer size on the PIC32. "unsigned" means that the 32 bits should be interpreted as an integer from 0 to 2^{31} - 1.

    Directly below the definition of AD1CON1, you find a "typedef" command that creates a new data type called __AD1CON1bits_t. This is a data type in the same way that "float" and "int" are data types. This particular data type consists of the union of three "struct"s (structures) with "fields" called DONE, SAMP, ASAM, etc., up to ON, for the first struct; SSRC, FORM, etc. for the second; and "w" for the third. The numbers after the colons indicate how many bits long that particular field is. In the first struct, the field

DONE is 1 bit, SAMP is 1 bit, ASAM is 1 bit, the next bit has no name, CLRASAM is 1 bit, etc. The second struct is an alternative representation of the bits of the SFR; it has 5 unnamed bits, then 3 collectively named SSRC, 3 named FORM, 2 unnamed bits, 1 named ADSIDL, 1 unnamed, and 1 named ADON. Finally, the last struct has all 32 bits in a field named w.

Just below this type definition is defined the variable AD1CON1bits of type __AD1CON1bits_t. This allows us to refer to specific bits using more understandable variable names, rather than just talking about the 7th bit of AD1CON1. AD1CON1bits does not appear to be used much in the adc10.h library, however (only in BusyADC10()). (AD1CON1bits actually resolves to the same data memory location as AD1CON1. We now have two ways to refer to bits in this memory location: by AD1CON1, of type int, or by AD1CON1bits, of type __AD1CON1bits_t. To refer to the SAMP bit of AD1CON1bits, for example, we just type AD1CON1bits.SAMP. The name after the period accesses the appropriate bits.)

Let's keep searching for the text AD1CON1. We can see it defined for assembly language about 30% of the way in the file (along with CLR, SET, and INV). We will mostly ignore this assembly code, which is defining memory locations (addresses) for the variables, but notice that consecutive addresses differ by 4. **Question: Why is this?** (These are actually "virtual" memory addresses, which then get translated to actual RAM addresses. This is beyond our scope.)

Next, about 55% of the way into the file, we see a bunch of #define statements, the first of which is #define _AD1CON1_DONE_POSITION 0x00000000. There are definitions for POSITION, MASK, and LENGTH for DONE, SAMP, ASAM, CLRASAM, SSRC0, etc., up to ON, corresponding to the first struct in the type __AD1CON1_bits_t; for SSRC, FORM, ADSIDL, and ADON for the second struct; and w for the third struct.

Questions: Referring as necessary to the SFRs for the ADC in the Reference Manual, what function do the bits DONE, SAMP, ASAM, CLRASAM, SSRC0, SSRC1, SSRC2, FORM0, FORM1, FORM2, SIDL, FRZ, and ON serve? What purpose do the POSITION, MASK, and LENGTH definitions serve? Now answer the same questions for the bits FORM. Are any of these POSITION, MASK, or LENGTH definitions used in adc10.h? Give an example of each (if there is an example) and explain what that line of code does.

At the end of the file, we see the constant _ADC10_BASE_ADDRESS defined. This is the address in (virtual) data memory where the where the ADC10 SFRs begin. You can see that this address is identical to the one defined for AD1CON1 in the assembly language definitions.

19. Refer to the definition of the AD1CON1 SFR and Section 17.3 of the Reference Manual. Let's say you want to set up the ADC to do the following: be on, continue operating in the debug exception mode, stop when the PIC is in IDLE mode, format the ADC output as a 32-bit integer, use an internal counter to end sampling and start conversion (auto conversion), continuously overwrite the buffer storing the conversion results, and begin the next sample immediately after the last conversion completes. (By reading Section 17.3 of the Reference Manual, you see that an ADC reading consists of two steps: sampling, where the voltage on the input pin is carried through to the output of a sample and hold circuit; and conversion, when the voltage at the output of the sample and hold is

disconnected from the input pin and held steady, so that this voltage is constant during the analog-to-digital conversion. The sampling process should be long enough that the output of the sample and hold circuit reflects the input voltage, and the conversion process takes 12 clock cycles.) What value will the AD1CON1 SFR have when the ADC sample and hold is currently sampling? Write it in hexadecimal, 0x ...

20. The #defines in adc10.h use definitions from other peripheral library files, like ports.h and int.h in pic32_libs\include\peripheral. For example, #define OpenADC10 in adc10.h uses mPORTBSetPinsAnalogIn(), which is defined in ports.h. Find the definition of this function in ports.h. What does it do?
21. The ADC peripheral library is simple enough that all of the library functions can be defined in the header file adc10.h using #define commands. Other library functions require more lines of code, and therefore are defined in .c files. An example is writing a string using the UART (e.g., RS-232 communication). The library function putsUART1 is located in puts_uartx_lib.c, which is under pic32_libs\peripheral\uart\source. This library function uses the library functions BusyUART1() and putcUART1(). In what file and directory are these functions defined?