Show all your work in addition to your answers. Closed books/notes and no electronic aids except for a non-programmable calculator. Use backside of this page if necessary.

1. (10%) PIC32 C programs typically contain the “#include <p32xxxx.h>” statement. Can it be replaced with a processor-specific name, e.g. “#include <p32mx795f512l.h>”? If so, why don’t we do this?

Yes, it can. The non-device-specific #include statement is to make the C programs more independent and easier to port to new projects with different but compatible devices. (See P. 6 of Di Jasio.)

2. (10%) At power-up or after a reset, the PIC32 will enter the main( ) function immediately. Is this statement true? Explain your answer if you don’t think this statement is true.

No, the statement is not true. After a reset or at power-up, before entering the main( ), the PIC32 will execute a short initialization code segment (automatically inserted by the MPLAB linker.) This is known as the Startup code, or the crt0 or simply c0 code. (See P. 7 of Di Jasio.)

3. (20%) What is the purpose of the TRISx registers? How are the TRISx registers used in PIC32 C code?

The TRISx registers allow us to specify the input/output directions for PORTx, where 0 means output and 1 means input, for a PORTx pin. An example C statement can be: TRISA = 0; // all PORTA pins for output (See P. 15 of Di Jasio.)

4. (20%) PORTB is by default configured for analog input. Which Special Function Register (SFR) must be used to configure PORTB for digital I/O? Show a code segment that accomplishes this.

The SFR, used in the Day 1 project, is AD1PCFG. A code statement “AD1PCFG = 0xffff” configures all 16 PORTB pins for digital I/O. TRISx, on the other hand, sets the I/O directions. (See P. 18 of Di Jasio.)

5. (20%) Which PORTA pins on PIC32 are multiplexed with JTAG that must be disabled to free up those pins for digital I/O? Write down the C statement that disables the JTAG port.

JTAG multiplexed PORTA pins include RA0, RA1, RA4, and RA5. The code “DDPCONbits.JTAGEN = 0”, or its equivalent, disables the JTAG interface to free up these pins. (See P. 17 of Di Jasio.)

6. (20%) Given a peripheral bus frequency of 36MHz and the delay equation discussed in class, will it be possible to produce a real-time delay of 1.024 seconds on PIC32 I, II or III using the constant DELAY and the C code from the “Day 2” project? Justify your answer. (Hint: Consider Timer1 module of PIC32.)

Using the real-time delay equation: Tdelay = (1/36 MHz)*256*DELAY, from the Day 2 project, where Tdelay = 1.024 s = 1024 x 10^{-3} s, as required by this problem, we will obtain DELAY = 14400.

Since Timer1 of PIC32 is 16-bit wide, the new DELAY value of 14400 is too large to fit into Timer1 at the given peripheral bus frequency of 36 MHz, and a maximum prescaler value of 256. Therefore it will not be possible to produce the required 1.024 seconds of real-time delay. (See P. 32 of Di Jasio.)