

You may use a calculator and the provided reference materials. If a binary result is required, give the value in HEX. For any required I2C functionality, use subroutine calls to *i2c_start()*, *i2c_rstart()*, *i2c_stop*, *i2c_put(char byte)*, *i2c_get(char ackbit)*. If you use *i2c_put*, you must pass in as an argument the byte that is to be written to the I2C bus. If you use *i2c_get*, you must pass in as an argument the bit value to be sent back as the acknowledge bit value.

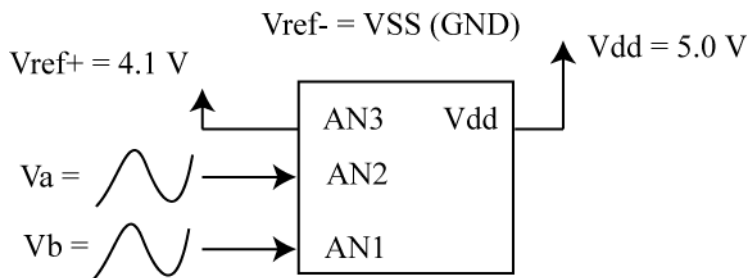
Part I: (74 pts) You must answer all of these questions.

- a. (15 pts) Assume an FOSC of 10 MHz. Write a **main()** function that will use PWM mode to generate a square wave with a 75% duty cycle and a period of 6 KHz on the CCP1 output pin. Divide your solution into two parts. In the first part, show any calculations required. In the second part, show the **main()** code. The last statement in the main code should just be an infinite **while(1){}** loop as the PWM hardware does everything for generating the square wave. The PWM duty-cycle is set by a 10-bit value, I only care about the upper 8-bits (assume the lower 2-bit are zero). Select the prescale that gives the LARGEST value for PR2 that still fits in the PR2 register.

(7 pts) Calculations here (show all work):

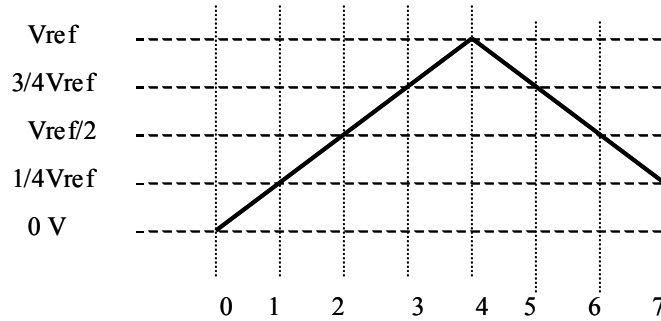
(8 pts) Code here:

- b. (15 pts) (PART 1, 7 pts) For the diagram below, write C code that will configure the A/D module for left justification, AN2, AN1 as analog inputs, AN3 as VREF+. VSS as VREF-. Use the internal FOSC clock, and configure the A/D clock such that it meets the minimum clock period constraint of $1.6 \mu\text{s}$ assuming an FOSC of 12 MHz (use the fastest internal clock choice that meets this constraint). For the configuration code, use individual bit names ADCS2, ADCS1, ADCS0, ADON, ADFM, PCFG3, PCFG2, PCFG1, PCFG0. You do not have to configure the channel select bits, that is done in PART2.



(PART 2, 8 pts) Write a function called `char analog_sum()` that will perform a conversion on each analog input (AN2, AN1) and return the sum of these values as a 'char' value. Do not let the sum exceed 255 (0xFF) (hint: You will need to use an unsigned INT variable to hold the sum, then clip this to 255). When changing A/D channels, use the `DelayUs(x)` function to delay $20 \mu\text{s}$ to give the A/D input a chance to settle. Since you don't know how often this function will be called, use this delay before starting each conversion. You do not know the setting of the channel select bits on entry to the function.

- c. (9 pts) We want to generate the waveform below using an 8-input lookup table. Assume a V_{ref} value of 5V, and an 8 bit DAC. The horizontal axis indicates the indices of the lookup table.



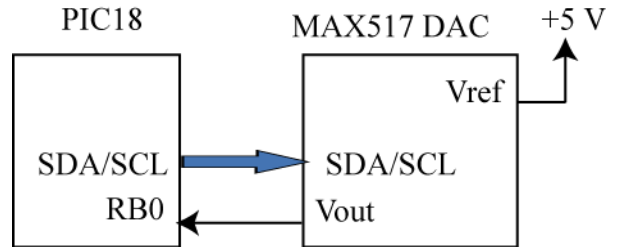
Fill in the contents of the 8 element char array used to

`char my_table[] = { ??? } // 8 entries.`

To make it easier to grade, fill in the table below with your 8 entries (the array indices are shown). Write your values in hex; this is the value to write to the DAC.

0:	4:
1:	5:
2:	6:
3:	7:

- d. (10 pts) In the diagram below, the output of the MAX517 D/A analog output is tied to the RB0 input. We would like to determine exactly what D/A output voltage is interpreted as a logic “1” on the RB0 input. Assume all configuration has been done (serial port, parallel ports, I2C, etc). Write a loop that starts the D/A output at 0 V and increments it each time through the loop by 1 LSB (one least significant bit value). Exit the loop when the RB0 input reads as a “1” value, and print the 8-bit DAC input code to the screen. You will need to use the I2C functions to write to the DAC. Assume both A1 and A0 on the DAC are tied high.



- e. (10 pts) Write a sequence of I2C function calls that will return the byte from location 0x80F0 within the 24LC515 Serial EEPROM. Assume A1 is tied low and A0 is tied high on the EEPROM.
- f. (5 pts) In class, we discussed how the PIC18 *capture* mode could be used to decode both biphasic and space-width encoded IR waveforms. There was a critical difference between how capture mode was used with biphasic decoding and how it was used with space-width encoding, what was this difference?
- g. (5 pts) For the waveform of problem #10, assume TIMER2 is configured with PRE=16, POST=5, PR2 = 0x90, and FOSC = 40 MHz. The periodic interrupt that is generated is used to output each point of the waveform. What is the frequency of the waveform in Hz? Show all work.
- h. (5 pts) When measuring pulse width using capture mode and timer1, give the equation we used that accounted for timer1 overflow. Draw a diagram that illustrates the components of the equation.

Part II: (36 pts) Answer 8 out of the next 10 questions. Cross out the 2 questions that you do not want graded. Each question is worth 3 pts, there is no partial credit.

1. A 3-bit flash A/D has 7 comparators; each whose output is can be either 0 or 1. Assume a $V_{ref} = 4$ V. What is the 7-bit output of these comparators if the input voltage is 2.7 V? Give the 7-bit value in binary, with the LSB the comparator output with the smallest V_{REF} input and the MSB the comparator with the highest V_{REF} input.
2. Draw 0xF0 as a space-width encoded waveform, assume '0' (75% duty cycle) is twice the period of a '1' (50% duty cycle); assume MSB is sent first.
3. How many TIMER1 tics equate to a 500 μ s value assuming an FOSC of 35 MHz and a prescale value of 1?
4. Given a V_{ref} of 4.4 volts, and an input voltage of 2.5 Volts, and a 7 bit A/D, what is the output code (give this in hex) ?

9. Write a C code fragment that performs a STOP condition on the I2C bus (do not use the `i2c_stop` function, I want to know what is inside the `i2c_stop` function!).

10. In some applications using VDD as an A/D reference voltage is fine, but in other cases it is not. When would you not use VDD as an A/D reference voltage?