QUESTION 5 | POINTS | ✖ Delete Question

Late? | 0

PROBLEM | Insert Images | Insert Field

```
NOT  question, tutors to use this to adjust for late submissions

|____|
```

↳Add **Subquestion**

✚ Add **Question 6**

Save Assignment

Q1 DAC vs PWM

5 Points

There are many advantages of using a DAC peripheral over just driving a GPIO bit to make PWM output signals (as discussed in Lecture 21, which can avoid the needing to interface to a DAC). Considering that the effective performance of an embedded system may require accurately generated signals of various voltages, provide a brief motivation for why a DAC could be preferable in an embedded system rather than just using PWM.

(Expected answer length: A paragraph, around 50 chars is sufficient.)

Q2 Benefits of the Flash ADC

3 Points

There are many types of Analog to Digital Converter (ADC) available. The Flash ADC has many advantages over other types of ADC, such as the Successive Approximation ADC. Select which of these is a clear benefit of a Flash ADC over the Successive Approximation ADC:

- ○ The Flash consumes more power
- ○ The Flash provides more digital codes
- ◉ The Flash is faster
- ○ The Flash ADCs are usually easier to interface

Q3 ENOB

5 Points

The Effective Number of Bits (ENOB) is an important factor in determining how effective the operation of an ADC is. What is meant by the ENOB of an ADC? For example, If we have an 8-bit ADC that has an ENOB below 8 what does this imply?

(Expected answer length: A paragraph.)

Q4 Analysing performance

12 Points

The following scenario relates to the two sub-questions that follow:

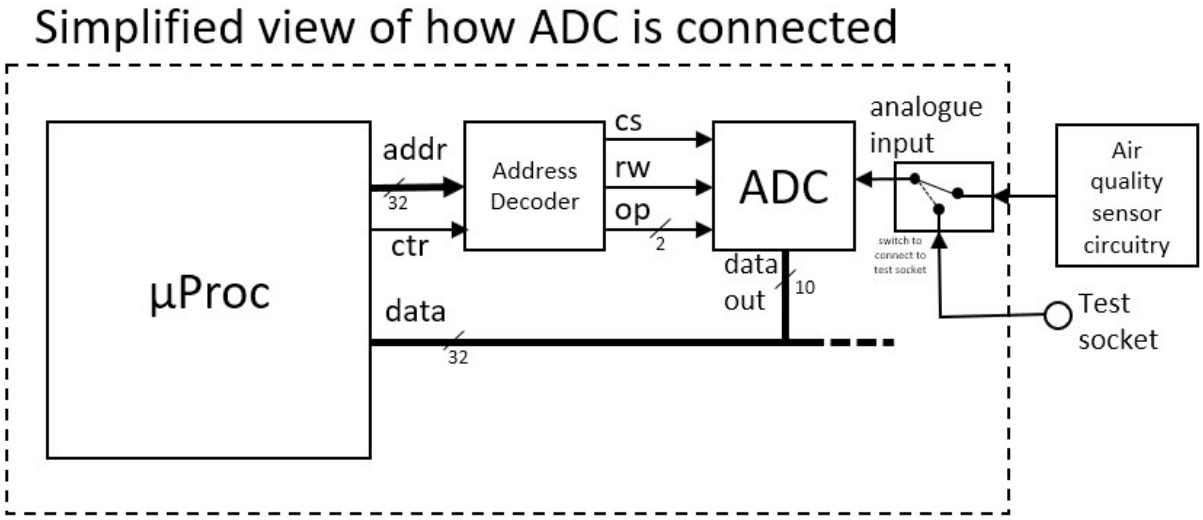
The ADC used

Consider the block diagram showing that a 10-ADC is connected up to a microcontroller via memory-mapped access. The ADC is not a flash, it doesn't always take the same time to return a value.
There are already three commands available (that control the 'op' line as needed) that can be used for reading samples by polling the ADC, these functions are:

```
void ADC_start();     // tell ADC to start a new sample
int  ADC_ready();     // returns true if ADC has a sample ready to read
unsigned ADC_read();  // read a 10-bit sample from the ADC
```

The application

The application using this ADC is for sampling an air quality sensor every 10ms, the sample is then put into an array in memory, a start point of the code is given below. Between samples some other work is done. Just assume it is a Voltage that is coming into the ADC that is between 0V and 3V. The analog input to the ADC can be switched to either reading the air quality sensor (it is usually set to this), or it can be set to a test socket which can be used for feeding in alternate input signals (which should be within the range 0V to max 3.5V otherwise the ADC may be damaged).



Starting code

```
// Air Quality Analysis (AQA) sampling program

// Includes:
#include <stdio.h>
#include <sys/time.h>
#include <time.h>

// Externally defined functions for using ADC:
void ADC_start();     // tell ADC to start a new sample
int  ADC_ready();     // returns true if ADC has a sample ready to read
unsigned ADC_read();  // read a 10-bit sample from the ADC

//extern defined function to read microsecond accurate timer since program started
extern unsigned long gettick_us ();

// Global variables
#define MAXSAMPS 1024       // want to collect 1024 samples for now
```

```
unsigned samps [MAXSAMPS];  // buffer of samples
int      n_samps = 0;       // number of samples read so far

// Assume this function does various processing on samps
void do_processing();

// Entry point to program
void main ()
{
   unsigned x;  // temporary vaiable to hold latest sample
   // contiously sample every 10ms:
   while (1) {
      ADC_start();          // tell ADC to start a new sample
      while (!ADC_ready()); // wait for ADC to be ready
      x = ADC_read();       // get next sample
      samps[n_samps] = x;   // put sample in array
      n_samps = (n_samps+1)%MAXSAMPS; // increase and wrap counter
      do_processing();      // do processing of samples
      delay_ms(10);         // wait 10ms before starting next sample
   }
}
```

Q4.1 Characterizing an ADC offset error

5 Points

What is meant by the offset error of an ADC? Discuss how you might go about characterizing the offset error for this case. You can discuss how you might connect things up, code you might need to write and, if there is an offset error, what you might do to make the provided sampling program work more accurately.

Q4.2 Speed of sampling ADC

4 Points

As mentioned earlier, the ADC in use does not always respond at the same speed. In our code we have a high-resolution time available via the gettick_us function.

Discuss how you might adjust the code to determine how long the ADC took (from calling ADC_start) to the time that the sample is read and put in variable x ready for use.

Q4.3 Improving accuracy

3 Points

In the code provided, the do_processing() operation may take longer than other times, in particular as the samps array fills up, it takes longer to complete. Although samples are meant to be taken every 10ms. Discuss how the code might be improved to improve the likelihood of samples being read regularly at 10ms.

(no need to provide code just a short description / code snippet / pseudocode is sufficient)

Q5 Late?

0 Points

NOT question, tutors to use this to adjust for late submissions