UNIVERSITY OF CAPE TOWN

DEPARTMENT OF ELECTRICAL ENGINEERING

EEE226S/EEE233S/EEE234S/EEE370S

Module D: Measurement and Microprocessors

FINAL EXAMINATION NOVEMBER 2005

TIME: 2 hours

TOTAL MARKS: 100 (See Note 3 below)

INSTRUCTIONS

1. This is a Closed Book Examination. Candidates will be supplied with the Instruction Set for the 68HC908JK1 (attached).

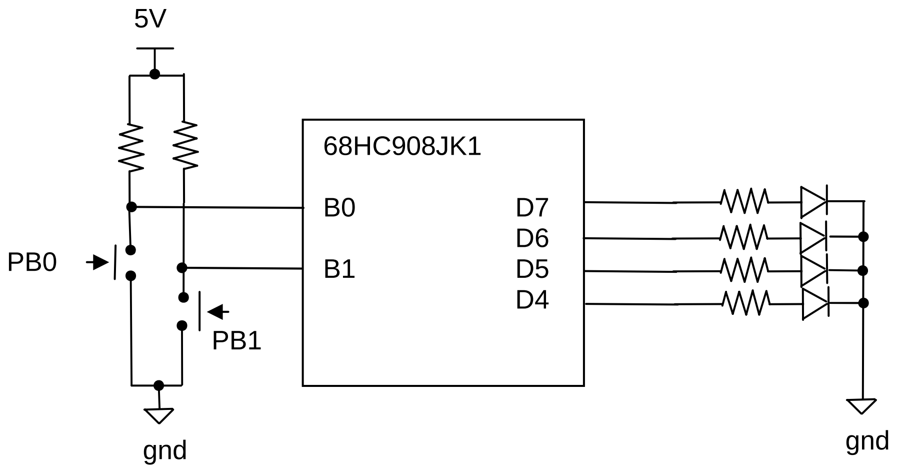
2. All numerical answers must be given to the appropriate number of significant figures, and the base of the number system must be indicated if it is not base 10.

3. Answer all questions. There are 100 marks available. There is no sub-minimum in this course.

INTERNAL EXAMINER: PROF. J. TAPSON
EXTERNAL EXAMINER:

Question 1:

A Motorola / Freescale 68HC908JK1 microprocessor is connected to push buttons on pins B0 and B1, and to LEDs on pins D7, D6, D5, and D4, as shown below (all other connections such as power, ground, the oscillator and so on can be assumed to be standard):



All resistors are 1kΩ.

a) Complete the following program, by writing the middle section, to output the binary number 1000 on the LEDs (with D7 as the most significant bit, down to D4 as the LSB). If PB0 is pushed, the number must increase by one. If PB1 is pushed, the number must decrease by one. If the number gets up to 1111 or down to 0000 it must stay at that count until the microprocessor is reset.

You may use all the defined variables and can define extra ones if you wish. You do not need to reproduce all the lines below in you answer book – just write down the missing code. If you need a delay routine, simply put **jsr delay** and we will assume that some predefined delay routine exists – you do not have to write a delay routine – but you must say in a comment how long the delay should be.

You must provide switch debouncing in your program.

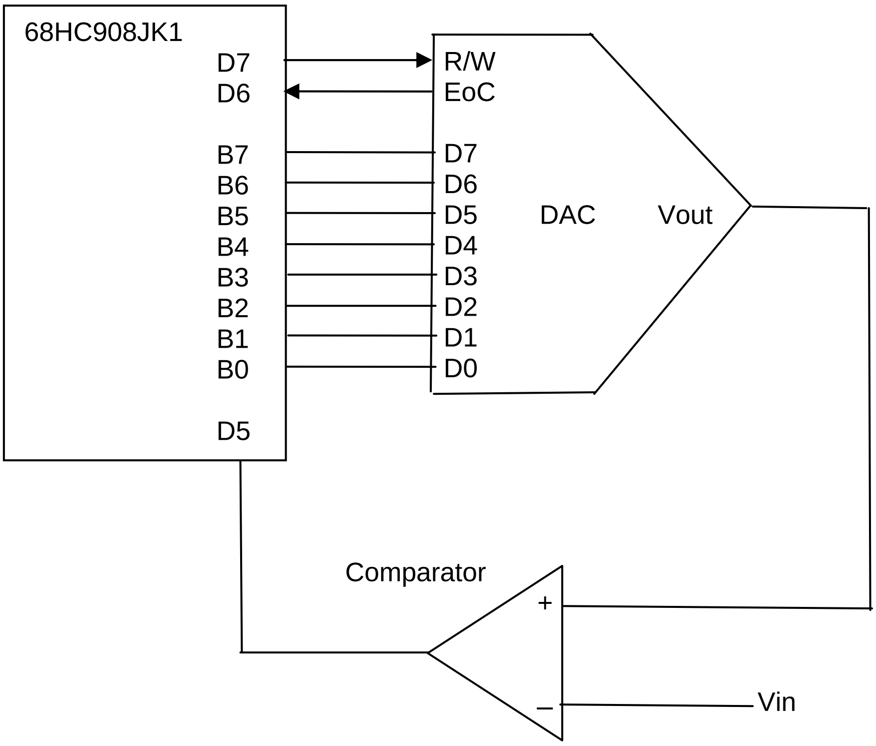
```
RomStart        EQU     $F600           ; Start of program memory
RamStart        EQU     $0080           ; Start of data memory
PTB             EQU     $0001           ; Port B
DDRB            EQU     $0005           ; Port B direction register
PTD             EQU     $0003           ; Port D
DDRD            EQU     $0007           ; Port D direction register

                org     RamStart        ; Start variables here

Timeout1        ds    1                 ; Define a byte variable
Count           ds    1                 ; Define a byte variable
```

```
                org     RomStart        ; Start program here
main_init:

;****************** YOUR CODE STARTS HERE  ***********************


;****************** END OF YOUR CODE  ****************************

                org $FFFE               ; postion of reset vector
                dw  main_init           ; Reset Vector
```

total: (15)

Question 2:

A 68HC908JK1 microprocessor is connected to a parallel 8-bit DAC chip as shown in the diagram below:



The DAC works as follows: start with R/W high. The data to be written must be set up on the DAC data lines D7-D0. When R/W is pulled low, the conversion takes place. When the conversion is finished, the EoC line will go high.

Write the following programs using the framework shown below:

```
RomStart        EQU     $F600           ; Start of program memory
RamStart        EQU     $0080           ; Start of data memory
```

```
PTB             EQU     $0001           ; Port B
DDRB            EQU     $0005           ; Port B direction register
PTD             EQU     $0003           ; Port D
DDRD            EQU     $0007           ; Port D direction register

                org     RamStart        ; Start variables here

Timeout1        ds    1                 ; Define a byte variable
Count           ds    1                 ; Define a byte variable
Loop            ds    1                 ; Define a byte variable

                    org     RomStart        ; Start program here
main_init:

;****************** YOUR CODE STARTS HERE  ***********************


;****************** END OF YOUR CODE  ****************************

                org $FFFE               ; postion of reset vector
                dw  main_init           ; Reset Vector
```

a) Write a subroutine called *DAC_out* which performs a DAC conversion. The data byte to be converted must be in the accumulator when *DAC_out* is called.

(10)

b) Write a program using *DAC_out*, which operates the whole system as a ramp ADC converter to convert the unknown voltage Vin., and save the result in Timeout1.

(10)

c) Write a program using *DAC_out*, which operates the whole system shown in the picture as a successive approximation ADC converter to convert the unknown voltage Vin,and save the result in Timeoout1.

(15)

total: (35)

Question 3:

The list file for a 68HC908JK1 program is shown below. Please study it and then answer the questions which follow.

```
   0000                   1  PortB   EQU     $0001
    0000                   2  PortD   EQU     $0003
    0000                   3  DDRB    EQU     $0005
    0000                   4  DDRD    EQU     $0007
    0000                   5  Rom     EQU     $F600
    0000                   6  Ram     EQU     $0080
                           7
    0080                   8          org     Ram
                           9
    0080                  10  count1  ds      1
```

```
0081                    11  count2   ds      1
0082                    12  count3   ds      1
                        13
F600                    14           org     Rom
                        15
F600 [04] 6EFF05        16  main:    mov     #$FF,DDRB
F603 [04] 101F          17           bset    0,$001F
F605 [04] 6E0F01        18  start:   mov     #$0F,PortB
F608 [05] CDF61B        19           jsr     delay
F60B [04] 3901          20           rol     PortB
F60D [05] CDF61B        21           jsr     delay
F610 [04] 3901          22           rol     PortB
F612 [05] CDF61B        23           jsr     delay
F615 [04] 3901          24           rol     PortB
F617 [05] CDF61B        25           jsr     delay
F61A [01] 8F            26           wait
                        27
                        28  ;************************************
                        29  ; Subroutines
                        30  ;***********************************
                        31
F61B [04] 6E0280        32  delay:   mov    #$02,count1
                        33
F61E [04] 6E0381        34  outside: mov     #$03,count2
                        35  inside:
F621 [05] 3B81FD        36           dbnz    count2,inside
F624 [05] 3B80F7        37           dbnz    count1,outside
                        38
F627 [04] 81            39           rts
                        40
FFFE                    41           org     $FFFE
FFFE      F600          42           dw      main
                        43
```

a) What is the real address of the label `outside`? (1)

b) What is the real address of the label `inside`? (2)

c) What is the opcode for the instruction **`rol`**? (1)

d) What would the opcode be for **`rol count1`**? (2)

e) What would the opcode be if line 36 was **`dbnz count3,delay`**? (4)

f) What will the final value of **`count1`** be? (4)

g) What will the final value of address $0001 be? (4)

h) If the instructions take the following number of clock cycles:

Instruction	Cycles
rol	3
dbnz	5
mov	4
jsr	6
rts	4

how many cycles would it take to get from line 32 (`delay: mov #$02,count1`) to line 39 (`rts`), inclusive of lines 32 and 39? (7)

i) If a clock cycle is 250 ns, write a subroutine *two_seconds* which would give a two second delay, exactly, using the program above as a framework. Show mathematically how your subroutine gives this length of delay. (15)

total: (40)

Question 4:

A depth sensor is used for measuring the depth of petrol in a steel tank which is 3.0m deep. The sensor output is 1V/m. The sensor is connected directly into an ADC input of a 68HC908JK1 microprocessor, which digitizes the range 0-5V with 8 bits accuracy.

a) With what accuracy, in metres, can the depth be digitized? (2)
b) Give a circuit that would improve the accuracy, using the same ADC. (3)
c) What would the improved accuracy be? (2)
d) If the sensor was connected directly into a 16-bit ADC digitizing the range from -10V to +10V, what would the accuracy be in metres? (3)

total: (10)