# UNIVERSITY COLLEGE LONDON

# EXAMINATION FOR INTERNAL STUDENTS

| | | |
|---|---|---|
| MODULE CODE | : | **ELEC0010** |
| ASSESSMENT PATTERN | : | **ELEC0010A5UD** |
| MODULE NAME | : | **ELEC0010 - Digital Design** |
| LEVEL: | : | **Undergraduate** |
| DATE | : | **27-Apr-2022** |
| TIME | : | **10:00** |

**Controlled Condition Exam: 3 Hours exam**

**You cannot submit your work after the date and time shown on AssessmentUCL – you must ensure to allow sufficient time to upload and hand in your work**

This paper is suitable for candidates who attended classes for this module in the following academic year(s):

**Year**
**2021/22**

| | |
|---|---|
| **Additional material** | |
| **Special instructions** | |
| **Exam paper word count** | |

List of formulae and physical constants:

Velocity of propagation of an electromagnetic wave in vacuum $c = 3 \times 10^{8}$ m s$^{-1}$
Intrinsic impedance of vacuum or air: $Z_0 = 337\ \Omega$
Permittivity of vacuum or air: $\boldsymbol{\varepsilon_o} = 8.85 \times 10^{-12}$ F m$^{-1}$
Permeability of vacuum or air: $\boldsymbol{\mu_o} = 4\pi\times 10^{-7}$ H m$^{-1}$

**Answer ALL questions**

## SECTION A [Answer ALL questions from this section]

1.

(a) Write out the four-level logic (0,1, X, Z) truth table for the combinational logic circuit in Figure 1.1.

[3 marks]

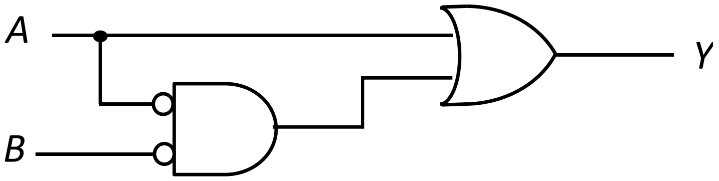


**Figure 1.1 Combinational logic circuit**

(b) (i) Sketch the circuit which would be synthesised from the SystemVerilog module below.

```
module comb_circuit (input logic [3:0] a, b,
                     input logic sel_a, sel_b,
                     output wire [3:0] y);

assign y = (sel_a) ? a : 4'bz;
assign y = (sel_b) ? b : 4'bz;

endmodule
```

[2 marks]

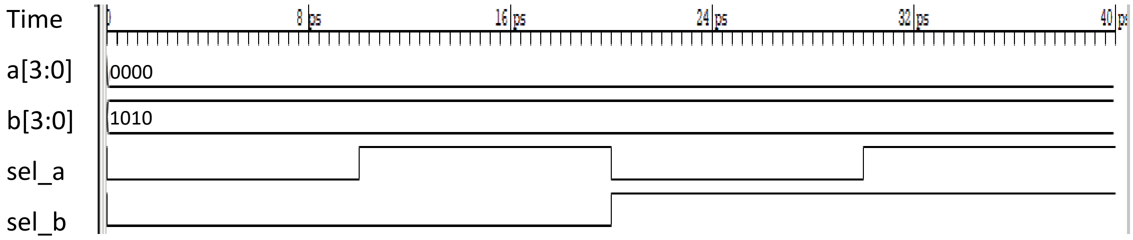


**Figure 1.2 Input signal waveforms for comb_circuit module simulation.**

(ii) Write a testbench to test the behaviour of the **comb_circuit** module, applying the test vectors shown in Figure 1.2.

[5 marks]

(iii) Sketch the waveforms shown in Figure 1.2, and add the waveform for the output, *y*, that you would expect from the simulation using your testbench.

[5 marks]

(c) Write, as a SystemVerilog module with the name **seq_logic**, a description of the behaviour of the circuit in Figure 1.3. In your code, use cyclic behaviours (**always_ff**) to describe the operation of the registers, and continuous assignment statements (**assign**) for the combinational logic between the registers. The registers are updated on the rising edges of the clock.

[5 marks]

(d) For the circuit in Figure 1.3, the setup time and hold time of the registers are 25 ps and 30 ps respectively. The clock-to-Q delay of the registers is 50 ps, the propagation delay through each inverter is 25 ps and through each XOR gate is 35 ps. The clock frequency is 5 GHz.

(i) Assuming the delays from interconnections is negligible, calculate the propagation delay through the combinational logic.

[2 marks]

(ii) Calculate whether the conditions to avoid setup errors are met in this circuit, and hence state whether you would expect the circuit to function correctly at this clock frequency.

[3 marks]

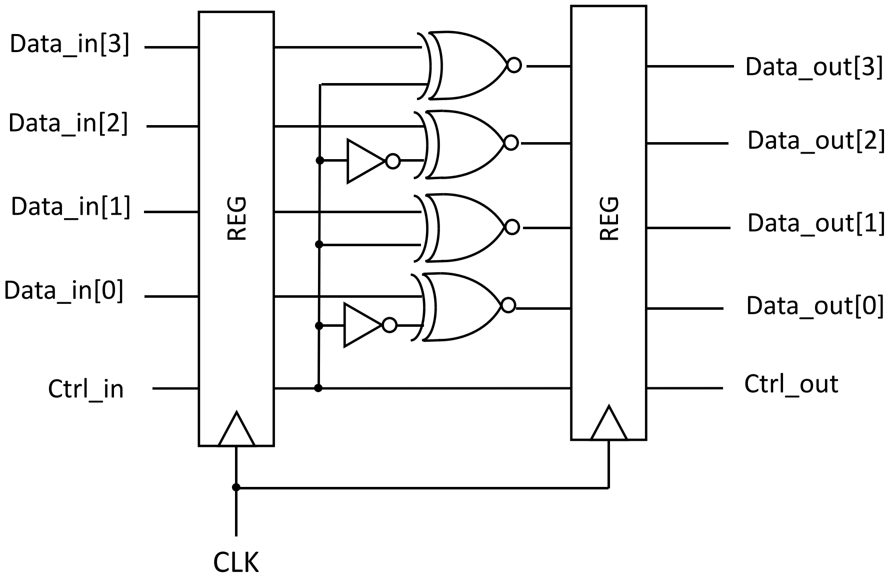


**Figure 1.3: Sequential logic circuit**

2.

The pandemic has induced a high demand for innovative bio-sensing solutions. Swiftly, several R&D entities have answered the call, and several single-use disposable diagnostic tools have been proposed and are currently under investigation or development. The main unit will then analyze the cartridge used for collecting the bio-sample.

You are in charge of the team that needs to develop the microelectronic chip to control the main unit.

(a) The manufactory volume is expected to be low (about 1000 units per year), and a hardware upgrade is scheduled for two years. There are several technologies you have available: ranging from CPU to full-custom. Which one is the technology that suits the application? Provide motivation for your answer

[3 marks]

(b) Assuming $\mu_{n/p,}$ $C_{ox}$, $V_{th}$ $V_{dd}$ respectively, the mobility, oxide capacitance of the technology, threshold voltage and driving voltage is known. Can you estimate the 3dB bandwidth of the technology used when an inverter has a fan out of three inverters? The solution is in the function of W and L, respectively the width and the length of the MOSFETs.

[7 marks]

(c) The agreement with the chip manufacturer will enable you to buy lots of wafers from them. The estimated chip size is 10 [mm$^2$], and the wafer size is 300 [mm] in diameter. The expected defects density is estimated to be 250 [Defects· wafer$^{-1}$]. What is the expected yield?

[5 marks]

(d) Assuming you are using FPGAs with a cost of £50 per unit, assuming, after a successful product launch, the manufacturing volume grows by several orders of magnitude. You are considering opting for a custom process (ASIC). Accounting for the information in (c), considering mask set cost of £100000, for what manufacturing volume (chips units) does the custom (ASIC) process become more cost-effective than the FPGA solution?

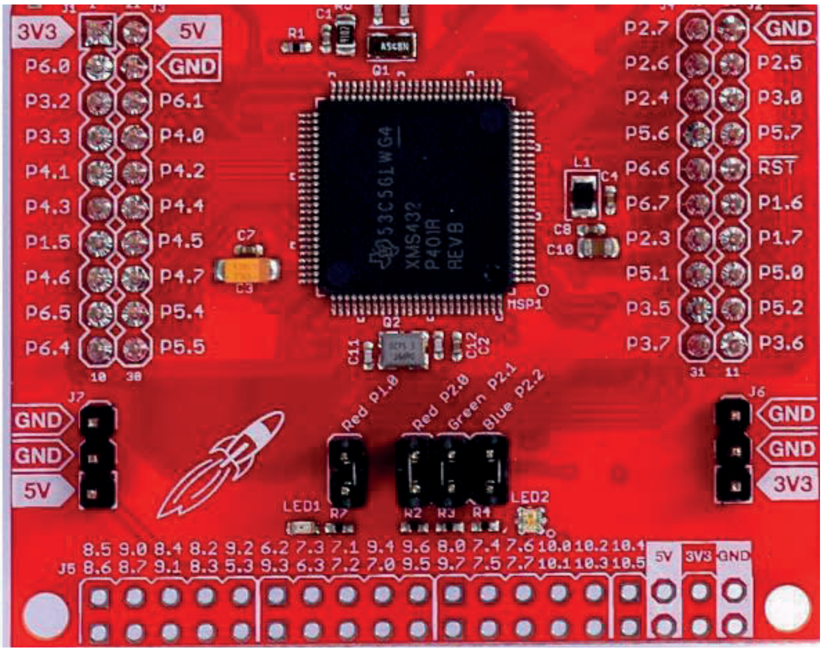
[10 marks]

## SECTION B [Answer ALL questions from this section]

### Reference Material for the MSP432

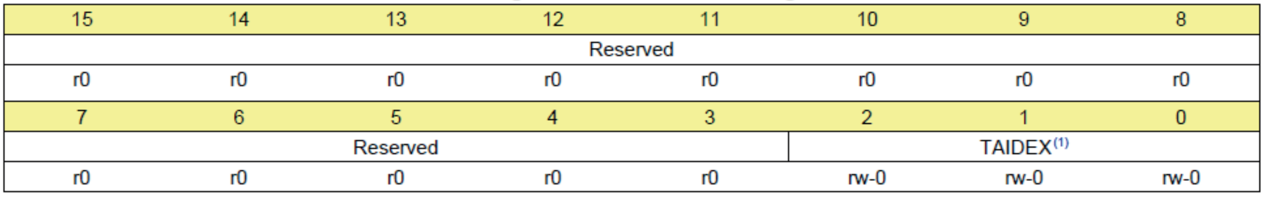
**NOTE**: Where there is a reference to the MSP432, it is assumed that the Texas Instruments MSP432P401R Launchpad development board is used.

*MSP432P401R Launchpad Pinout Configuration*



*Timer Control Registers (extract from technical reference manual)*

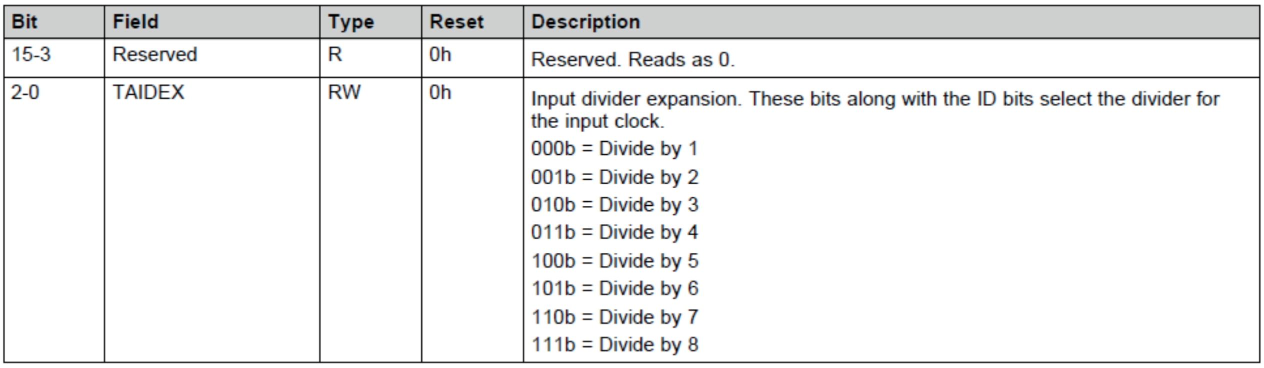
**Figure 19-20. TAxEX0 Register**



| 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 |
|---|---|---|---|---|---|---|---|
| Reserved | | | | | | | |
| r0 | r0 | r0 | r0 | r0 | r0 | r0 | r0 |
| 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
| Reserved | | | | | TAIDEX[1] | | |
| r0 | r0 | r0 | r0 | r0 | rw-0 | rw-0 | rw-0 |

[1] After programming TAIDEX bits and configuration of the timer, set TACLR bit to ensure proper reset of the timer divider logic.

**Table 19-9. TAxEX0 Register Description**



| Bit | Field | Type | Reset | Description |
|---|---|---|---|---|
| 15-3 | Reserved | R | 0h | Reserved. Reads as 0. |
| 2-0 | TAIDEX | RW | 0h | Input divider expansion. These bits along with the ID bits select the divider for the input clock.<br>000b = Divide by 1<br>001b = Divide by 2<br>010b = Divide by 3<br>011b = Divide by 4<br>100b = Divide by 5<br>101b = Divide by 6<br>110b = Divide by 7<br>111b = Divide by 8 |

### Figure 19-15. TAxCTL Register

| 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 |
|---|---|---|---|---|---|---|---|
| Reserved | | | | | | TASSEL | |
| rw-0 | rw-0 | rw-0 | rw-0 | rw-0 | rw-0 | rw-0 | rw-0 |
| **7** | **6** | **5** | **4** | **3** | **2** | **1** | **0** |
| ID | | MC | | Reserved | TACLR | TAIE | TAIFG |
| rw-0 | rw-0 | rw-0 | rw-0 | rw-0 | w-0 | rw-0 | rw-0 |

### Table 19-4. TAxCTL Register Description

| Bit | Field | Type | Reset | Description |
|---|---|---|---|---|
| 15-10 | Reserved | RW | 0h | Reserved |
| 9-8 | TASSEL | RW | 0h | Timer_A clock source select<br>00b = TAxCLK<br>01b = ACLK<br>10b = SMCLK<br>11b = INCLK |
| 7-6 | ID | RW | 0h | Input divider. These bits along with the TAIDEX bits select the divider for the input clock.<br>00b = /1<br>01b = /2<br>10b = /4<br>11b = /8 |
| 5-4 | MC | RW | 0h | Mode control. Setting MCx = 00h when Timer_A is not in use conserves power.<br>00b = Stop mode: Timer is halted<br>01b = Up mode: Timer counts up to TAxCCR0<br>10b = Continuous mode: Timer counts up to 0FFFFh<br>11b = Up/down mode: Timer counts up to TAxCCR0 then down to 0000h |
| 3 | Reserved | RW | 0h | Reserved |
| 2 | TACLR | RW | 0h | Timer_A clear. Setting this bit resets TAxR, the timer clock divider logic, and the count direction. The TACLR bit is automatically reset and is always read as zero. |
| 1 | TAIE | RW | 0h | Timer_A interrupt enable. This bit enables the TAIFG interrupt request.<br>0b = Interrupt disabled<br>1b = Interrupt enabled |
| 0 | TAIFG | RW | 0h | Timer_A interrupt flag<br>0b = No interrupt pending<br>1b = Interrupt pending |

CONTINUED

*Clock System Control 0 Register (extract from technical reference manual)*

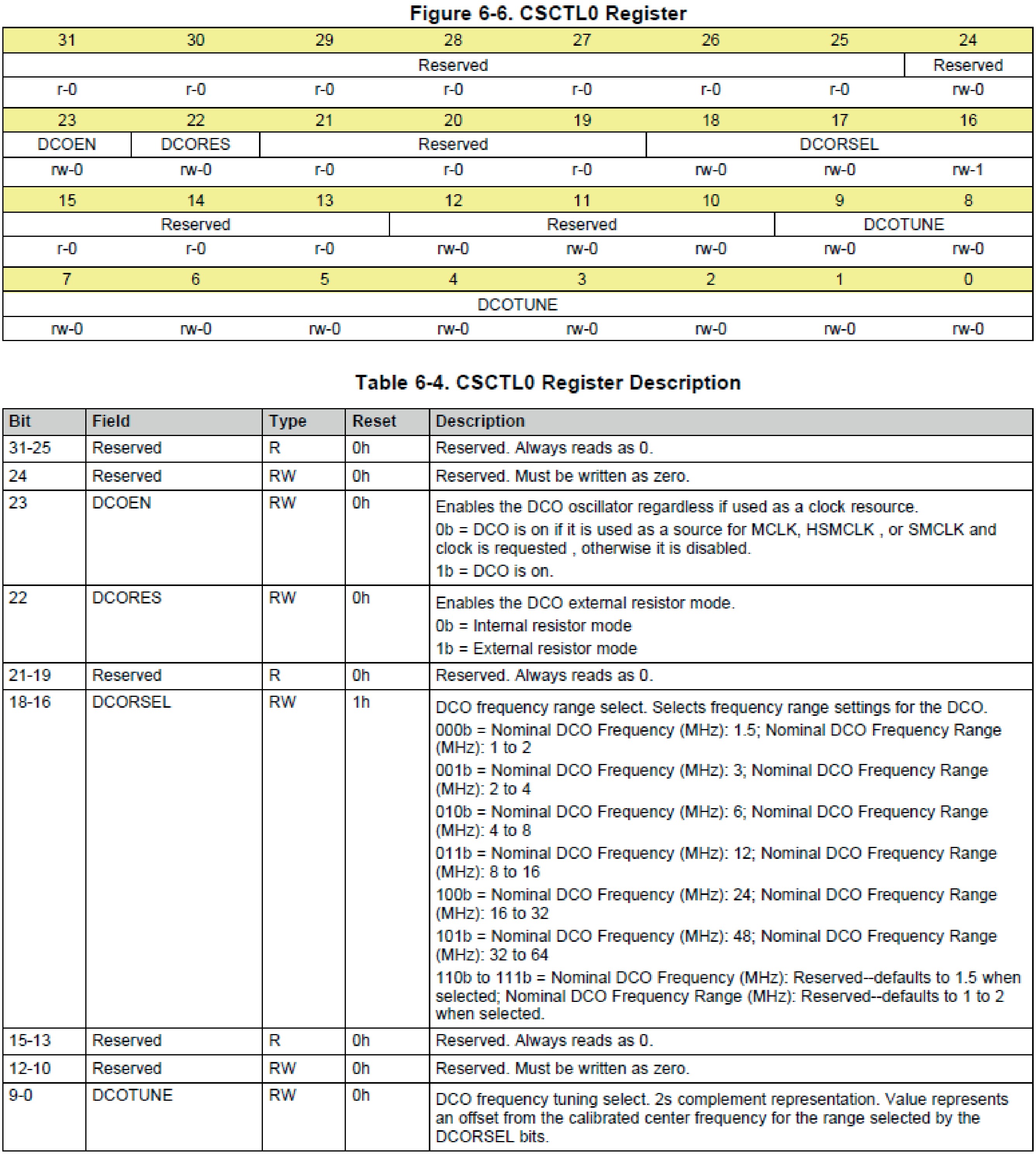


**Figure 6-6. CSCTL0 Register**

| | | | | | | | |
|---|---|---|---|---|---|---|---|
| 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 |
| Reserved | | | | | | | Reserved |
| r-0 | r-0 | r-0 | r-0 | r-0 | r-0 | r-0 | rw-0 |
| 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
| DCOEN | DCORES | Reserved | | | DCORSEL | | |
| rw-0 | rw-0 | r-0 | r-0 | r-0 | rw-0 | rw-0 | rw-1 |
| 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 |
| Reserved | | | Reserved | | | DCOTUNE | |
| r-0 | r-0 | r-0 | rw-0 | rw-0 | rw-0 | rw-0 | rw-0 |
| 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
| DCOTUNE | | | | | | | |
| rw-0 | rw-0 | rw-0 | rw-0 | rw-0 | rw-0 | rw-0 | rw-0 |

**Table 6-4. CSCTL0 Register Description**

| Bit | Field | Type | Reset | Description |
|---|---|---|---|---|
| 31-25 | Reserved | R | 0h | Reserved. Always reads as 0. |
| 24 | Reserved | RW | 0h | Reserved. Must be written as zero. |
| 23 | DCOEN | RW | 0h | Enables the DCO oscillator regardless if used as a clock resource.<br>0b = DCO is on if it is used as a source for MCLK, HSMCLK , or SMCLK and clock is requested , otherwise it is disabled.<br>1b = DCO is on. |
| 22 | DCORES | RW | 0h | Enables the DCO external resistor mode.<br>0b = Internal resistor mode<br>1b = External resistor mode |
| 21-19 | Reserved | R | 0h | Reserved. Always reads as 0. |
| 18-16 | DCORSEL | RW | 1h | DCO frequency range select. Selects frequency range settings for the DCO.<br>000b = Nominal DCO Frequency (MHz): 1.5; Nominal DCO Frequency Range (MHz): 1 to 2<br>001b = Nominal DCO Frequency (MHz): 3; Nominal DCO Frequency Range (MHz): 2 to 4<br>010b = Nominal DCO Frequency (MHz): 6; Nominal DCO Frequency Range (MHz): 4 to 8<br>011b = Nominal DCO Frequency (MHz): 12; Nominal DCO Frequency Range (MHz): 8 to 16<br>100b = Nominal DCO Frequency (MHz): 24; Nominal DCO Frequency Range (MHz): 16 to 32<br>101b = Nominal DCO Frequency (MHz): 48; Nominal DCO Frequency Range (MHz): 32 to 64<br>110b to 111b = Nominal DCO Frequency (MHz): Reserved--defaults to 1.5 when selected; Nominal DCO Frequency Range (MHz): Reserved--defaults to 1 to 2 when selected. |
| 15-13 | Reserved | R | 0h | Reserved. Always reads as 0. |
| 12-10 | Reserved | RW | 0h | Reserved. Must be written as zero. |
| 9-0 | DCOTUNE | RW | 0h | DCO frequency tuning select. 2s complement representation. Value represents an offset from the calibrated center frequency for the range selected by the DCORSEL bits. |

*ADC14 Control 1 Register (extract from technical reference manual)*

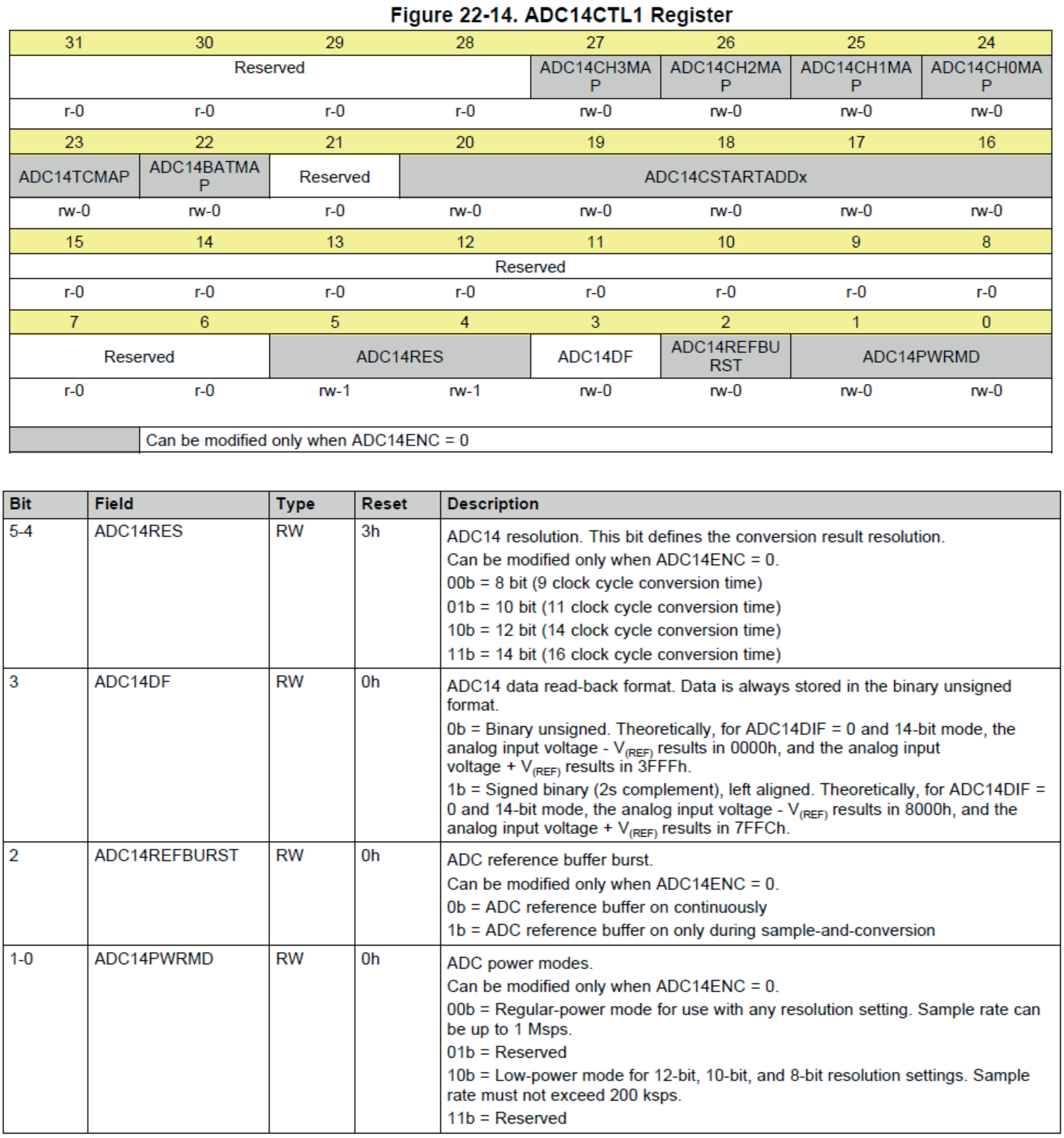


**Figure 22-14. ADC14CTL1 Register**

| 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 |
|---|---|---|---|---|---|---|---|
| Reserved | | | | ADC14CH3MAP | ADC14CH2MAP | ADC14CH1MAP | ADC14CH0MAP |
| r-0 | r-0 | r-0 | r-0 | rw-0 | rw-0 | rw-0 | rw-0 |
| **23** | **22** | **21** | **20** | **19** | **18** | **17** | **16** |
| ADC14TCMAP | ADC14BATMAP | Reserved | ADC14CSTARTADDx | | | | |
| rw-0 | rw-0 | r-0 | rw-0 | rw-0 | rw-0 | rw-0 | rw-0 |
| **15** | **14** | **13** | **12** | **11** | **10** | **9** | **8** |
| Reserved | | | | | | | |
| r-0 | r-0 | r-0 | r-0 | r-0 | r-0 | r-0 | r-0 |
| **7** | **6** | **5** | **4** | **3** | **2** | **1** | **0** |
| Reserved | | ADC14RES | | ADC14DF | ADC14REFBURST | ADC14PWRMD | |
| r-0 | r-0 | rw-1 | rw-1 | rw-0 | rw-0 | rw-0 | rw-0 |

| | |
|---|---|
| ▨ | Can be modified only when ADC14ENC = 0 |

| Bit | Field | Type | Reset | Description |
|---|---|---|---|---|
| 5-4 | ADC14RES | RW | 3h | ADC14 resolution. This bit defines the conversion result resolution.<br>Can be modified only when ADC14ENC = 0.<br>00b = 8 bit (9 clock cycle conversion time)<br>01b = 10 bit (11 clock cycle conversion time)<br>10b = 12 bit (14 clock cycle conversion time)<br>11b = 14 bit (16 clock cycle conversion time) |
| 3 | ADC14DF | RW | 0h | ADC14 data read-back format. Data is always stored in the binary unsigned format.<br>0b = Binary unsigned. Theoretically, for ADC14DIF = 0 and 14-bit mode, the analog input voltage - $V_{(REF)}$ results in 0000h, and the analog input voltage + $V_{(REF)}$ results in 3FFFh.<br>1b = Signed binary (2s complement), left aligned. Theoretically, for ADC14DIF = 0 and 14-bit mode, the analog input voltage - $V_{(REF)}$ results in 8000h, and the analog input voltage + $V_{(REF)}$ results in 7FFCh. |
| 2 | ADC14REFBURST | RW | 0h | ADC reference buffer burst.<br>Can be modified only when ADC14ENC = 0.<br>0b = ADC reference buffer on continuously<br>1b = ADC reference buffer on only during sample-and-conversion |
| 1-0 | ADC14PWRMD | RW | 0h | ADC power modes.<br>Can be modified only when ADC14ENC = 0.<br>00b = Regular-power mode for use with any resolution setting. Sample rate can be up to 1 Msps.<br>01b = Reserved<br>10b = Low-power mode for 12-bit, 10-bit, and 8-bit resolution settings. Sample rate must not exceed 200 ksps.<br>11b = Reserved |

CONTINUED

3. Consider a state machine implemented using the C code shown in Figure 3.1.

```
char state = 0;

while(1){

 switch(state){
  case 0:{
   if(event == 0) fnc0();
   if(event == 2) {
    state = 1;
    fnc3();
   }
  }
  break;
  case 1:{
   if(event == 1) {
    state = 0;
    fnc4();
   }
   if(event == 2) {
    state = 2;
    fnc4();
   }
  fnc2();
  }
  break;
  case 2:{
   if(event == 1) {
    state = 1;
    fnc3();
   }
   if(event == 2) {
    state = 3;
    fnc4();
   }
   fnc1();
  }
  break;
  case 3:{
   if(event == 0) {
    state = 0;
    fnc4();
   }
   if(event == 1) {
    state = 2;
    fnc4();
   }
   fnc1();
  }
  break;
 }
}
```

**Figure 3.1**

(a) Draw the state diagram for the system specified by the code in Figure 3.1.

[10 marks]

(b) Comment on the characteristics of the state machine represented by the code in Figure 3.1. Ensure you justify your reasoning.

[2 marks]

(c) Identify and list the number of unique events, unique actions, unique activities and total number of transitions in the state machine represented by the code in Figure 3.1. Comment on any particular observations with respect to the state machine characteristics.

[3 marks]

(d) Given two unsigned numbers presented in hexadecimal format as 0x213 and 0x1B1, respectively, would an 8-bit or a 16-bit processor compute the summation faster? Be sure to show your workings and to state any assumptions made. Would a carry bit have any impact on your justification?

[5 marks]

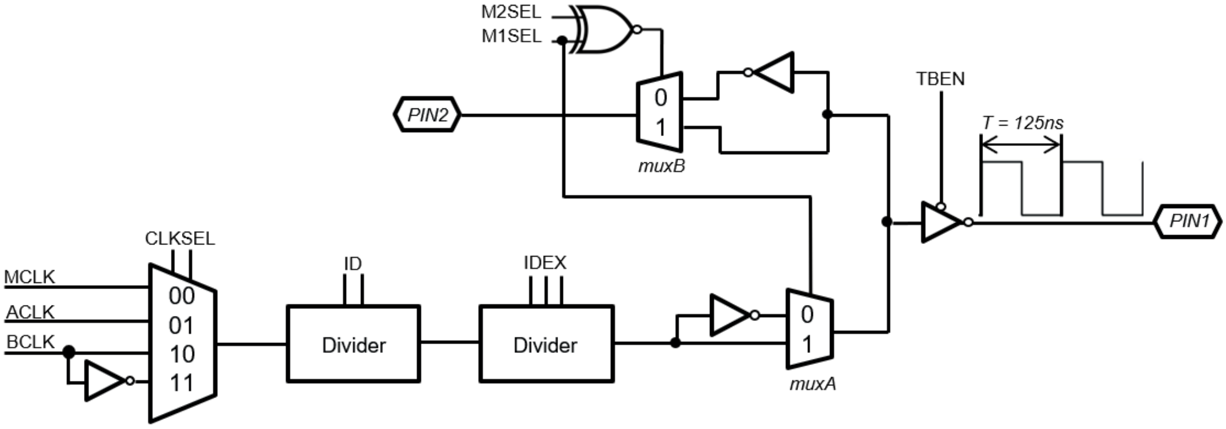
(e) Write an interrupt service routine (in register-level C code) that will toggle the colour of an MSP432's RGB LED when the characters 'R', 'G', 'B' are sent to the microcontroller over channel 0 of the eUSCI_A module in UART mode. Ensure that you comment your code in detail. Explain what one would observe when this routine is executed.

You may assume that the relevant registers have been initialised in the main program.

[5 marks]

4.

(a) Consider the functional block diagram shown in Figure 4.1. This corresponds to an arbitrary microcontroller whose architecture is based partially on that of the MSP432 microcontroller.



Timing control register:

| Bit index | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|
| Bit value | ? | ? | 0 | 0 | ? | ? | ? | ? | 0 | ? |
| Bit name | CLKSEL | | ID | | IDEX | | | TBEN | M1SEL | M2SEL |

**Figure 4.1**

The master clock is driven by a digital controlled oscillator running at the nominal frequency of the range DCORSEL_5 set in the CSCTL0 register, the auxiliary clock is driven by a low-frequency 32.768 kHz watch crystal, and the backup clock is driven by an external 10 kHz clock source. Note that the ID and IDEX pre-scalers are identical to those used for the 16-bit Timer A (TA) peripheral on the MSP432.

Given the above functional block diagram and the partially completed 10-bit timing control register shown in Figure 4.1, determine the remaining bit values of this timing control register and explain what signals we would expect to see at each output pin (we want both pins to be connected to our circuit) with reference to the clock source. Be sure to describe your approach and to justify your reasoning in detail while stating any assumptions made.

[13 marks]

(b) What is the minimum address bus width required to fully address 512 KB of RAM organised in 64-bit addressable words? Be sure to show your workings and to state any assumptions made.

[3 marks]

Question 4 continues on page 12

(c) Consider a program running on the MSP432 that samples one of the ADC channels and controls the RGB LED with reference to AVcc. If you are given the following code extract pertaining to the ADC's configuration parameters:

```
ADC14->CTL0 = ADC14_CTL0_SHT0_2 | ADC14_CTL0_SHP | ADC14_CTL0_ON;
ADC14->CTL1 = ADC14_CTL1_RES_1;
ADC14->MCTL[0] |= ADC14_MCTLN_INCH_1;
```

then write an interrupt service routine (in register-level C code) with the following functionality:

- If the ADC conversion result is greater than 0.75*AVcc, turn on the red LED only.
- If the ADC conversion result is between 0.25*AVcc and 0.75*AVcc, turn on the green LED only.
- If the ADC conversion result is less than 0.25*AVcc, turn on the blue LED only.

Ensure to justify your approach and reasoning, and to comment your code in detail.
You may assume that the relevant registers have been initialised in the main program and that the ADC's conversion result is stored in memory register 0.

[9 marks]

END OF PAPER