How cristal oscillator works engineering essay

Engineering

ASSIGN BUSTER

D. D. C. WickramarathnaS106462010/s/12296Report on last week's work (12. 02. 2013)

Exercises

How Cristal Oscillator works

A crystal oscillator is an electronic component, which uses to the mechanical resonance of a vibrating crystal of piezoelectric material to create an electrical signal with a very precise frequency. This frequency is commonly used to keep track of time, to provide a stable clock signal for digital integrated circuits and to stabilize frequencies for radio transmitters. The crystal oscillator circuit sustains oscillation by taking a voltage signal from the quartz resonator, amplifying it, and feeding it back to the resonator. The rate of expansion and contraction of the quartz is the resonant frequency, and is determined by the cut and size of the crystal.

How to find Input/output Impedance of Non- Inverting Op-Amp

Input Impedance of Non- Inverting Op-Amp

D: uocph 3032ip impedence. jpgFind Ri : Ii = i+ (ideal i- = i+ = 0)Ri = = ∞Input impedance (Ri) = ∞

Output Impedance of Non- Inverting Op-Amp

D: uocph 3032op impedence. jpgOutput impedance (R0)Find VL : 0 = i1 + i2 + i- (ideal i- = i+ = 0)0 = i1 + i2 + 00 = + (ideal V- = V+ = 0)0 = + (ideal V- = V+ = 0)0 = 0 = VLOutput impedance (R0) = RL = Substitute VL (R0) = RL = = 0Output impedance (R0) = 0

Nyquist Theorem

Nyquist's theorem developed by H. Nyquist, which states that an analog signal waveform may be uniquely reconstructed, without error, from samples taken at equal time intervals. The sampling frequency should be at least twice the highest frequency contained in the signal.

Program Codes of Practical session

Light a LED Bulb Using Switch

```
#include #define F_CPU 1000000ul // 1MHz#define LED PORTD0 // define
PORTD0 as LED#define SW PORTD1 // define PORTD1 as SWint main(void)
{DDRD= 1; // set portd0 as outputwhile(1){PORTD= 0; if((PIND & (1 < }

}
}
```

When we running above program, saw that when we press the switch, the LED is on. In here LED and Switch are controlled using Port D.

Binary Counter

```
#include #define F_CPU 1000000ul // 1MHz#define LED PORTD // define
PORTD as LED#define LED_D DDRD // define DDRD as LED_D#define SW
PORTB // define PORTB as SW#define SW_D DDRB // define DDRB as
SW_D#define SW1 PORTB0 // define PORTB0 as SW1#define SW1_IO PINB //
define PINB as SW1_IOint main(void){unsigned char count= 0; LED_D = 255;
//set the data direction of PortD as outputSW_D = 0; //set the data direction
of PortB as InputSW= 0 < } LED = count;
```

}

}

When we press the switch, 8 LED shows the binary value of the counts of switch pressed (only 0-9 counts). In here LEDs are controlled using Port D and Switch is controlled using Port B.

0 to 9 Counter using a Switch

#include #define F_CPU 1000000UL#define SSD_DATA PORTD // define PORTD as SSD_DATA#define SSD_DATA_D DDRD // define DDRD as SSD_DATA_D#define SW PORTB // define PORTB as SW#define SW_D DDRB // define DDRB as SW_D#define SW1 PORTB0 // define PORTB0 as SW1#define SW1_IO PINB // define PINB as SW1_IOint main(void){DDRC = 255; //set the data direction of PortC as outputPORTC= 1; unsigned char SSD[]={0x3f, 0x06, 0x5b, 0x4f, 0x66, 0x6d, 0x7d, 0x07, 0x7f, 0x6f}; unsigned char count= 0; SSD_DATA_D = 255; //set the data direction of PortD as outputSW_D = 0; //set the data direction of PortB as InputSW = 0 < } SSD_DATA = SSD[count];/*select the array element relevant to value of ‘ count’ and assign that array value to SSD_DATA*/

}

}

When we press the switch, SSD shows the value of the counts of switch pressed (only 0-9 counts). In here SSDs are controlled using Port D and Switch is controlled using Port B.

0 to 9999 Counter

```
#include #define F_CPU 1000000#define SW PORTB // define PORTB as SW#define SW_D DDRB // define DDRB as SW_D#define SW1 PORTB0 // define PORTB0 as SW1#define SW1_IO PINB // define PINB as SW1_IO#define SSD_DATA_D DDRD // define DDRD as SSD_DATA_D#define SSD_DATA PORTD // define PORTD as SSD_DATA#define SSD_DIG PORTC // define PORTC as SSD_DIG#define SSD_DIG_D DDRC // define DDRC as SSD_DIG_Dunsigned char SSD[]= {0x3f, 0x06, 0x5b, 0x4f, 0x66, 0x6d, 0x7d, 0x07, 0x7f, 0x6f}; unsigned char dig[]={0, 0, 0, 0}; void decode(unsigned int num){dig[0]= num/1000; dig[1]=(num%1000)/100; dig[2]=(num%100)/10; dig[3]=(num%10);

}
void display(unsigned int rounds){unsigned int i= 0, j= 0; for(j= 0; j!= rounds; j++){for(i= 0; i!= 4; i++){SSD_DATA = SSD[dig[i]]; /*select the ‘ dig’ array element relevant to value of ‘ i’ and also assign that array element value as ‘ SSD’ array element and assign that array value to SSD_DATA*/_delay_us(10); SSD_DIG= 0; //remove ghost effect

}
}
}
int main(void){SSD_DATA_D = 255; //set the data direction of PortD as outputSW_D= 0; //set the data direction of PortB as outputSW= 0 < }
display(100);
```

}

}

When we press the switch once, the MCU begin to count from 0 and seven segment displays shows the counts. That means the counts are reach to 10, it shows as 0010 on the display. We can change delay time between 2 numbers when changing the value of " display (1000)". If we press the switch without release, the counter set as pause condition and temporary pause the counting.

0 to 9999 Bidirectional Counter

When we running above code, saw that the seven segment display shows zero before the values. That means if we want to show 10 on the display, it shows as 0010. Therefore we modified above program to avoid that and to show only 10 on the SSD. Also we modified above code to count forward and reverse when pressing a switch. When we press the switch once, counter begin to count forward. If we press the switch again, counter count reverse form current counter value. If we press the switch again, the counter counts forward beginnings with current counter value. Below code has been written including above modifications.#include #define F_CPU 1000000#define SW PORTB#define SW_D DDRB#define SW1 PORTB0#define SW1_IO PINB#define SSD_DATA_D DDRD#define SSD_DATA PORTD#define SSD_DIG PORTC#define SSD_DIG_D DDRCunsigned char SSD[]={0x3f, 0x06, 0x5b, 0x4f, 0x66, 0x6d, 0x7d, 0x07, 0x7f, 0x6f, 0}; unsigned char dig[]={10, 10, 10, 0}; void decode(unsigned int num){dig[0]= num/1000; if(dig[0]== 0){ //check the value of num/1000dig[0]= 10; /*insert 10 as 0th element of " dig" array. 10th element of " SSD" array is 0. Therefore there is nothing display on the 4th segment of SSD*/dig[1]=(num%1000)/100; if(dig[1]== 0){ // check the value of (num%1000)/100dig[1]= 10; /*insert 10 as 1st element of " dig" array. 10th element of " SSD" array is 0. Therefore there is no displaying anything on the 3rd segment of SSD*/dig[2]=(num%100)/10; if(dig[2]== 0){ // check the value of (num%1000)/10dig[2]= 10; /*insert 10 as 2nd element of " dig" array. 10th element of " SSD" array is 0. Therefore there is no displaying anything on the 2nd segment of SSD*/

}

dig[3]=(num%10);}else{dig[2]=(num%100)/10; dig[3]=(num%10);

}

}else{dig[1]=(num%1000)/100; dig[2]=(num%100)/10; dig[3]=(num%10);

}

}

void display(unsigned int rounds){unsigned int i= 0, j= 0, k= 0; for(j= 0; j!= rounds; j++){for(i= 0; i!= 4; i++){SSD_DATA = SSD[dig[i]]; _delay_us(10); SSD_DIG= 0;

}

}

}

int main(void){SSD_DATA_D = 255; SW_D= 0; SW= 0 < } else {if((SW1_IO & (1 < }

```
}

}
}display(100);

}

}
```