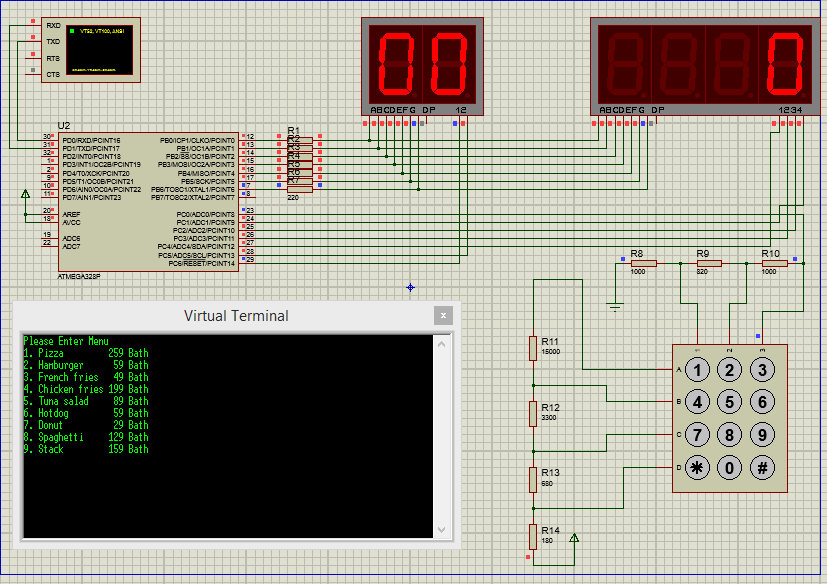
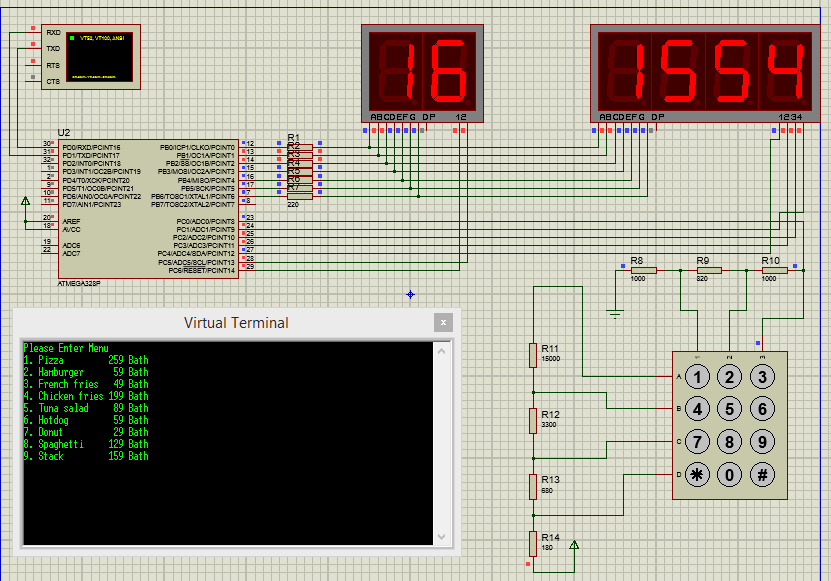
รูปวงจร

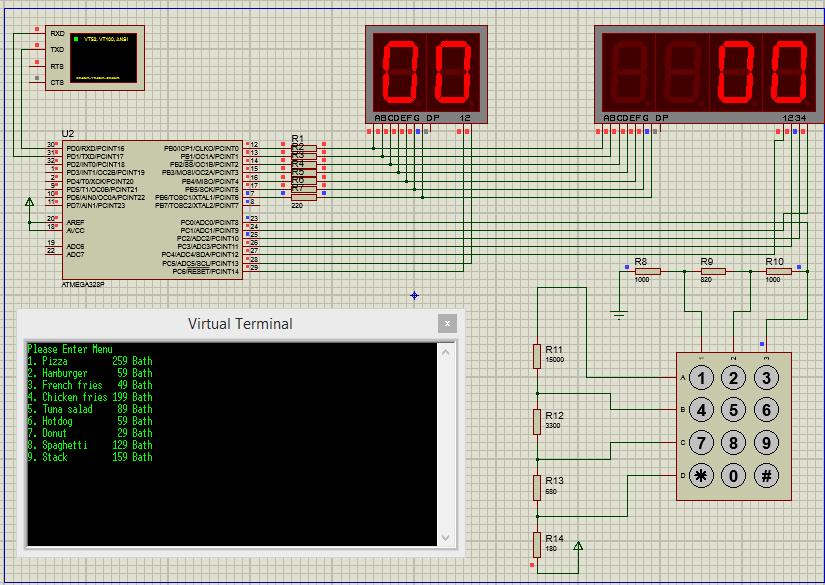
* หน้าหลักเมื่อเริ่มต้นใช้โปรแกรม



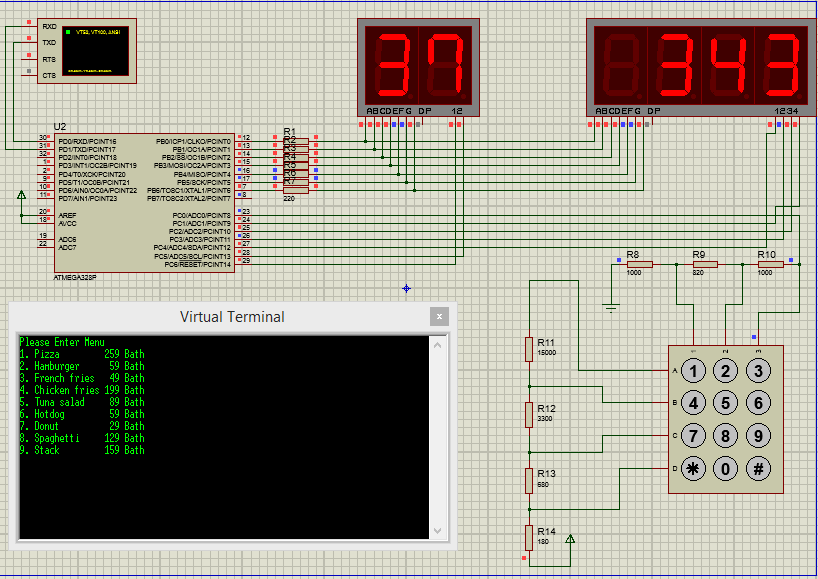
* เมื่อกดเลือกเมนูที่ 1 กด # แล้วเลือกจำนวน 6 ชุด กด # จะได้ว่า 259 x 6 = 1554 บาท



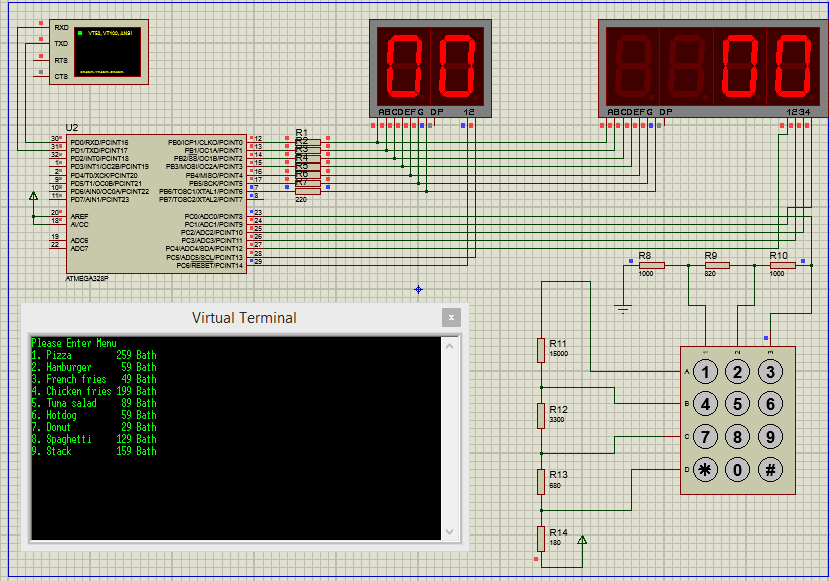
* เมื่อกดปุ่ม # โปรแกรมจะทำการรับค่าต่อไป



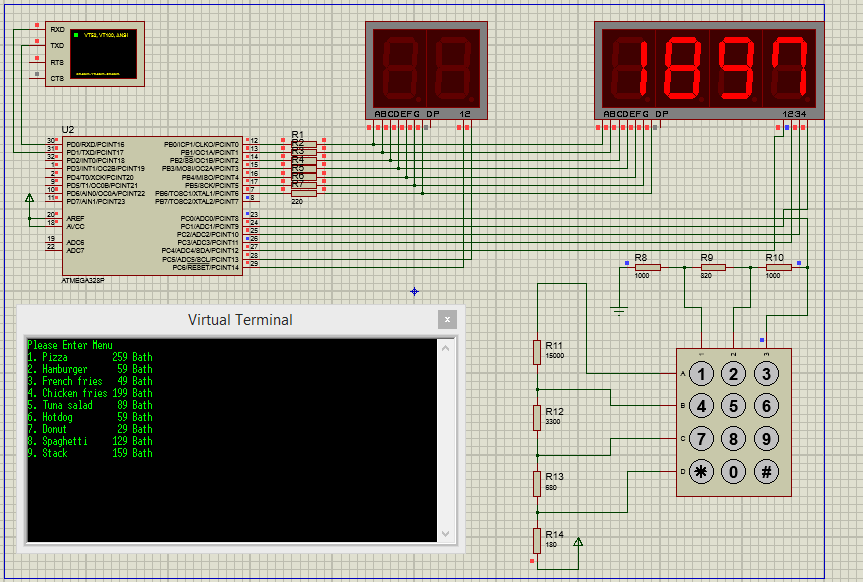
* เมื่อกดเลือกเมนูที่ 3 กด # แล้วเลือกจำนวน 7 ชุด กด # จะได้ว่า 49 x 7 = 343 บาท



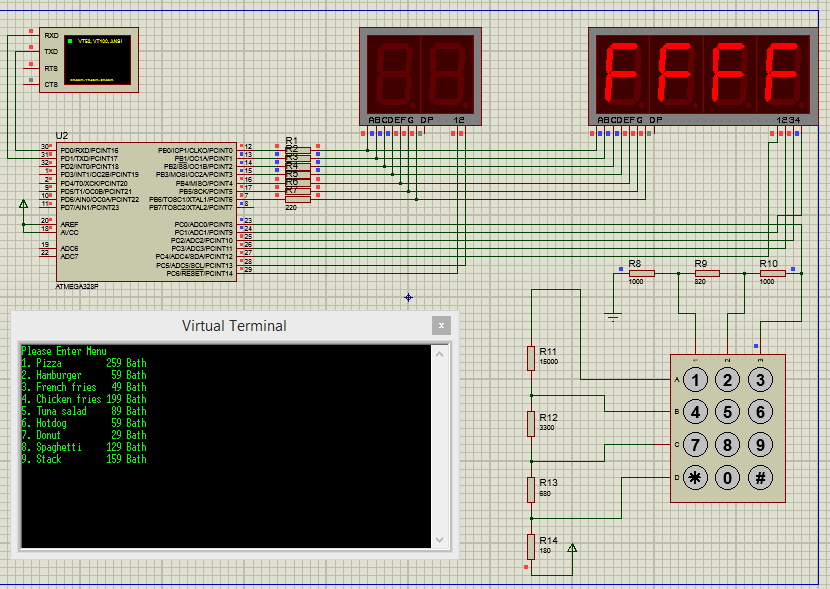
* เมื่อกดปุ่ม # โปรแกรมจะทำการรับค่าต่อไป



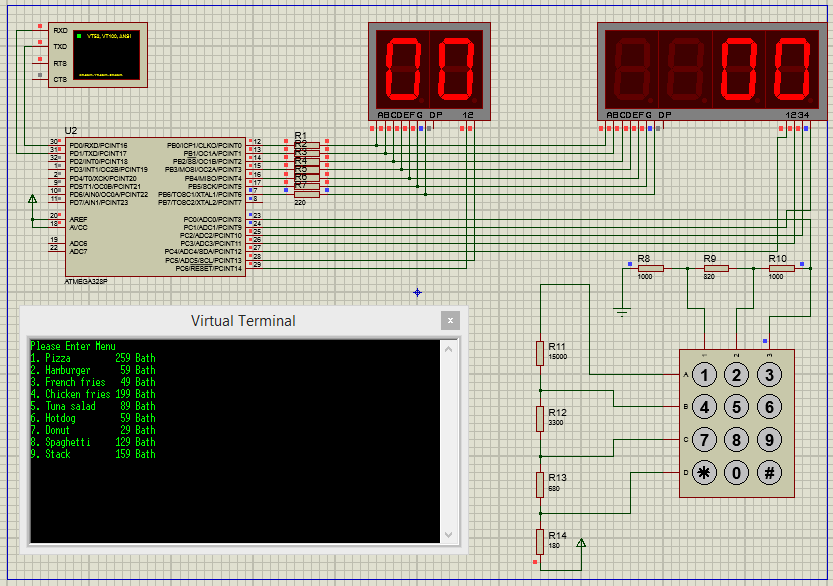
* เมื่อกดปุ่ม \* จะทำการรวมเงินของ 2 เมนู คือ 1554 + 343 = 1897 บาท



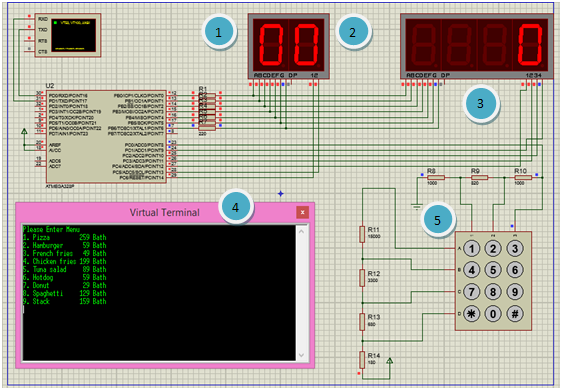
* เมื่อซื้อของเกิน 9999 บาท จะแจ้ง Error แสดง ffff โดยเงื่อนไขการซื้อของไม่เกิน จำนวน 9999 บาท



* เมื่อกดปุ่ม 0 จะทำการเคลียร์ค่าเป็นค่าเริ่มต้นใหม่



รายละเอียดอุปกรณ์



Input

* Keypad 3x4

Output

* Virtual Terminal ( Usart )
* 7 segment 2 digit และ 4 digit

รายละเอียดของหมายเลข

1. 7segment เพื่อแสดงค่าเลขของเมนูอาหาร
2. 7segment เพื่อแสดงค่าจำนวนชุดของเมนูอาหาร
3. 7segment เพื่อแสดงค่ารายระเอียดเกี่ยวกับราคาของเมนูอาหาร
4. Virtual Terminal เพื่อแสดงเมนูอาหาร
5. Keypad สำหรับเลือกเมนูอาหาร จำนวนชุดของเมนูที่เลือก และเลือดโหมดต่างๆ

โหมดในการทำงาน

1. ปุ่ม \* คือ โหมดสำหรับรวมเงินทั้งหมด
2. ปุ่ม 0 คือโหมดเคลียร์ให้เท่ากับค่าเริ่มต้น
3. ปุ่ม # คือโหมดสำหรับคำนวณค่าโดยขึ้นอยู่กับการกดโดยกดตัวเลข แล้วกด # คือเลือกเมนูอาหารเสร็จแล้วใส่เลขจำนวนชุดแล้วกด # จะทำการคำนวณ หากกด # จะทำการเลือกเมนูใหม่

เทคนิคที่ใช้ในวงจร

1. ใช้ความรู้เรื่อง Timer/Counter1 (วงจรจับเวลา/นับ ขนาด 16 บิต ) และโปรแกรมบริการอินเตอร์รัพต์ของ Timer/Counter1 ใช้ Timer1 ในการให้จังหวะสัญญาณนาฬิการ่วมกับการแปลง Analog to Digital ของ Keypad
2. ใช้ความรู้เรื่องการสื่อสารข้อมูลแบบอนุกรม ( serial Communication ) เพื่อแสดงค่าเมนูอาหารเมนูต่าง ๆ
3. ใช้ความรู้เรื่อง Delay ในการแสดงผล 7segment โดยจะใช้ Delay ที่เหมาะสมเพื่อให้หลอกตามองตัวเลขที่แสดงนั้นแสดงต่อเนื่องกันในแต่ละ digit

CODE

|  |
| --- |
| #include <inttypes.h>  #include <avr/io.h>  #include <avr/interrupt.h>  #include <avr/sleep.h>  #define INTERVAL\_time 65528  #include <util/delay.h>  #define F\_CPU 12000000UL  volatile unsigned int bitone,bittwo,bitthree,mode,v,ans,tmp,a,COST,tmp2,allcost,all,count,check;  void USART\_Transmit(unsigned char data)  {  while(! (UCSR0A & (1 << UDRE0)))  ;  UDR0 = data;  }  unsigned char TEXT[30] = {"Please Enter Menu "};  unsigned char m1[30] = {"1. Pizza 259 Bath"};  unsigned char m2[30] = {"2. Hamburger 59 Bath"};  unsigned char m3[30] = {"3. French fries 49 Bath"};  unsigned char m4[30] = {"4. Chicken fries 199 Bath"};  unsigned char m5[30] = {"5. Tuna salad 89 Bath"};  unsigned char m6[30] = {"6. Hotdog 59 Bath"};  unsigned char m7[30] = {"7. Donut 29 Bath"};  unsigned char m8[30] = {"8. Spaghetti 129 Bath"};  unsigned char m9[30] = {"9. Stack 159 Bath"};  unsigned char i;  unsigned int ACCUMULATED, DIVISOR, QUOTIENT, DECODED;  unsigned char TABLE7SEG[] ={  0b00111111, // 0  0b00000110, // 1  0b01011011, // 2  0b01001111, // 3  0b01100110, // 4  0b01101101, // 5  0b01111101, // 6  0b00100111, // 7  0b01111111, // 8  0b01101111, // 9  0b01110111, // A  0b11111100, // b  0b00111001, // C  0b00111111, // D  0b01111001, // E  0b01110001, // F  0b00000000 }; // turn n off all segment    int main()  {  DDRC = 0xFE;  DDRB = 0xFF;  PORTB = 0xFF;  ADMUX = 0b00100000;  ADCSRA = 0b10101101;  ADCSRB = 0b00000110;  TIMSK1 = 0x01;  TCCR1A = 0x00;  TCCR1B = 0x04;  mode = 0;  cli();  TCNT1 = INTERVAL\_time;  sei();  ACCUMULATED = 0;  DIVISOR = 0;  a = 0;  tmp = 0;  allcost = 0;  all = 0;  count = 0;  check = 0;    UCSR0A = 0x02;  UCSR0B = 0x98;  UCSR0C = 0x06;  UBRR0H = 0;  UBRR0L = 207;  TEXT [26] = 13;  TEXT [27] = 10;  for (i = 0; i<30; i++)  {  USART\_Transmit(TEXT[i]);  }  m1[26] = 13;  m1[27] = 10;  for (i = 0; i<30; i++)  {  USART\_Transmit(m1[i]);  }  m2[26] = 13;  m2[27] = 10;  for (i = 0; i<30; i++)  {  USART\_Transmit(m2[i]);  }  m3[26] = 13;  m3[27] = 10;  for (i = 0; i<30; i++)  {  USART\_Transmit(m3[i]);  }  m4[26] = 13;  m4[27] = 10;  for (i = 0; i<30; i++)  {  USART\_Transmit(m4[i]);  }  m5[26] = 13;  m5[27] = 10;  for (i = 0; i<30; i++)  {  USART\_Transmit(m5[i]);  }  m6[26] = 13;  m6[27] = 10;  for (i = 0; i<30; i++)  {  USART\_Transmit(m6[i]);  }  m7[26] = 13;  m7[27] = 10;  for (i = 0; i<30; i++)  {  USART\_Transmit(m7[i]);  }  m8[26] = 13;  m8[27] = 10;  for (i = 0; i<30; i++)  {  USART\_Transmit(m8[i]);  }  m9[26] = 13;  m9[27] = 10;  for (i = 0; i<30; i++)  {  USART\_Transmit(m9[i]);  }  while (1){  if(mode != 2 && mode != 5 && allcost <= 9999 && ans <= 9999){    PORTC = 0xFC;  PORTB = TABLE7SEG[bitone];  \_delay\_ms(10);    if(check>0)  {  PORTC = 0xFA;  PORTB = TABLE7SEG[bittwo];  \_delay\_ms(10);  }    PORTC = 0x3E;  PORTB = TABLE7SEG[a];  \_delay\_ms(10);  PORTC = 0x5E;  PORTB = TABLE7SEG[tmp];  \_delay\_ms(10);    if(bitthree != 0)  {  PORTC = 0xF6;  PORTB = TABLE7SEG[bitthree];  \_delay\_ms(10);  }  }  else if(mode == 2 && allcost <= 9999 && ans <= 9999)  {  PORTC = 0xFC;  PORTB = TABLE7SEG[ans%10];  \_delay\_ms(10);  PORTC = 0xFA;  PORTB = TABLE7SEG[(ans/10)%10];  \_delay\_ms(10);    if((ans/100)%10!=0)  {  PORTC = 0xF6;  PORTB = TABLE7SEG[(ans/100)%10];  \_delay\_ms(10);  }  if((ans/1000)!=0)  {  PORTC = 0xEE;  PORTB = TABLE7SEG[ans/1000];  \_delay\_ms(10);  }  PORTC = 0x3E;  PORTB = TABLE7SEG[a];  \_delay\_ms(10);  PORTC = 0x5E;  PORTB = TABLE7SEG[tmp];  \_delay\_ms(10);    }  else if(mode == 5 && allcost <= 9999 && ans <= 9999)  {  PORTC = 0xFC;  PORTB = TABLE7SEG[allcost%10];  \_delay\_ms(10);    if(allcost>0){  PORTC = 0xFA;  PORTB = TABLE7SEG[(allcost/10)%10];  \_delay\_ms(10);  if((allcost/100)%10!=0)  {  PORTC = 0xF6;  PORTB = TABLE7SEG[(allcost/100)%10];  \_delay\_ms(10);  }  if((allcost/1000)!=0)  {  PORTC = 0xEE;  PORTB = TABLE7SEG[allcost/1000];  \_delay\_ms(10);  }  }  }  else  {  PORTC = 0xFC;  PORTB = TABLE7SEG[15];  \_delay\_ms(10);  PORTC = 0xFA;  PORTB = TABLE7SEG[15];  \_delay\_ms(10);  PORTC = 0xF6;  PORTB = TABLE7SEG[15];  \_delay\_ms(10);  PORTC = 0xEE;  PORTB = TABLE7SEG[15];  \_delay\_ms(10);  }  }  return 0;  }    ISR(TIMER1\_OVF\_vect){  TCNT1 = INTERVAL\_time;  }  ISR(ADC\_vect){  PORTD = ~PORTD;  if(ADCH > 9){ //ผลลัพธ์จากการแปลงแรงดัน  ACCUMULATED += ADCH;  DIVISOR++; //จำนวนรอบที่แปลง A D แล้ว  }  else {  ACCUMULATED = 0;  DIVISOR = 0;  }  if(DIVISOR == 10){  QUOTIENT = ACCUMULATED/DIVISOR; //ค่าผลลัพธ์จากการแปลง a->D/จำนวนรอบที่แปลง A->D แล้ว  if ((QUOTIENT>9) && (QUOTIENT<16)) {DECODED = 1; COST = 259;}  else if ((QUOTIENT>18) && (QUOTIENT<26)) {DECODED = 2; COST = 58;}  else if ((QUOTIENT>29) && (QUOTIENT<37)) {DECODED = 3; COST = 49;}  else if ((QUOTIENT>46) && (QUOTIENT<53)) {DECODED = 4; COST = 199;}  else if ((QUOTIENT>74) && (QUOTIENT<81)) {DECODED = 5; COST = 89;}  else if ((QUOTIENT>99) && (QUOTIENT<107)) {DECODED = 6; COST = 59;}  else if ((QUOTIENT>133) && (QUOTIENT<141)) {DECODED = 7; COST = 29;}  else if ((QUOTIENT>169) && (QUOTIENT<177)) {DECODED = 8; COST = 129;}  else if ((QUOTIENT>192) && (QUOTIENT<199)) {DECODED = 9 ; COST = 159;}  else if ((QUOTIENT>212) && (QUOTIENT<220)) DECODED = 10 ;  else if ((QUOTIENT>228) && (QUOTIENT<236)) DECODED = 0 ;  else if ((QUOTIENT>236) && (QUOTIENT<244)) DECODED = 11 ;  else DECODED = 16 ;  check++;    if(DECODED == 11)  mode++;    }  if(mode == 0 && DECODED <= 9 && DECODED >= 1)  {  bitone = COST%10;  bittwo = (COST/10)%10;  bitthree = COST/100;  tmp2 = COST;  a = DECODED;  }  else if(mode == 1 && DECODED <= 9 && DECODED >= 1)  {  tmp = DECODED;  }  else if(mode == 2 )  {  ans = tmp2\*tmp;  if(count==0){  all = ans;  allcost = all;  }  else  {  allcost = all + ans;  }  }    else if(mode==3)  {  count++;  all = allcost;  a=0;  tmp=0;  bitone=0;  bittwo=0;  bitthree=0;  mode=0;  }  else if(DECODED==0)  {  a=0;  ans=0;  count=0;  allcost=0;  all=0;  tmp=0;  bitone=0;  bittwo=0;  bitthree=0;  mode=0;  }  else if(DECODED==10)  {  mode = 5;  }    } |