MICROCONTROLLER LAB MANUAL

IV Sem EC/TC/EE/IT/BME

ARUNKUMAR G

Lecturer in E&CE Dept.,

S.T.J.I.T., Ranebennur.

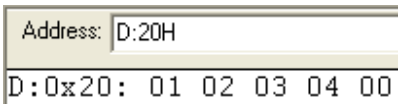
Cell: 9731311770

e-mail: gowda.arun@gmail.com

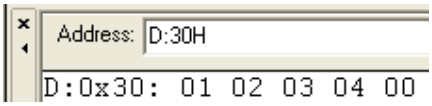
BLOCK MOVE

```
        ORG 00H
        MOV R2,#04H
        MOV R0,#20H
        MOV R1,#30H
UP:
        MOV A,@R0
        MOV @R1,A
        INC R0
        INC R1
        DJNZ R2,UP
        END
```

I/P:



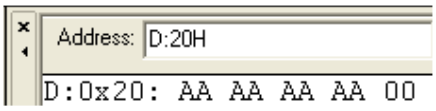
O/P:

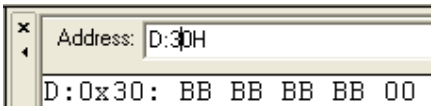


BLOCK EXCHANGE

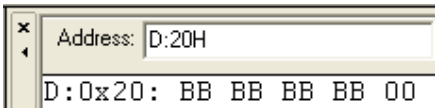
```
        ORG 00H
        MOV R2,#04H
        MOV R0,#20H
        MOV R1,#30H
UP:
        MOV A,@R0
        XCH A,@R1
        MOV @R0,A
        INC R0
        INC R1
        DJNZ R2,UP
        END
```

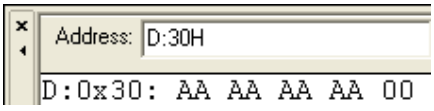
I/P:





O/P:

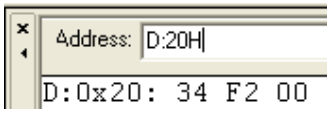


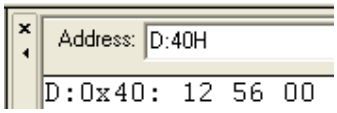


ADDITION OF TWO 16-BIT NUMBER (MULTIBYTE ADDITION)

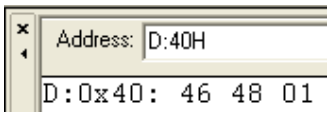
```
        ORG 00H
        MOV R7,#02H
        MOV R0,#20H
        MOV R1,#40H
UP:
        MOV A,@R0
        ADDC A,@R1
        MOV @R1,A
        INC R0
        INC R1
        DJNZ R7,UP
        JNC NOCARRY
        INC R2
NOCARRY: MOV A,R2
        MOV @R1,A
        END
```

I/P:





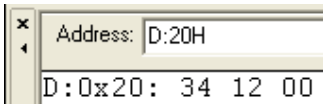
O/P:

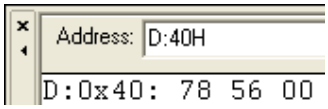


SUBTRACTION OF TWO 16-BIT NUMBER (MULTIBYTESUBTRACTION)

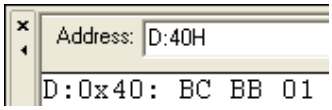
```
        ORG 00H
        MOV R7,#02H
        MOV R0,#20H
        MOV R1,#40H
UP:
        MOV A,@R0
        SUBB A,@R1
        MOV @R1,A
        INC R0
        INC R1
        DJNZ R7,UP
        JNC NOCARRY
        INC R2
NOCARRY: MOV A,R2
        MOV @R1,A
        END
```

I/P:



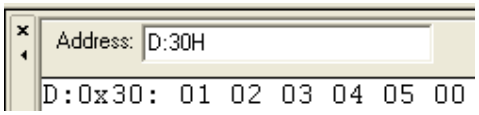
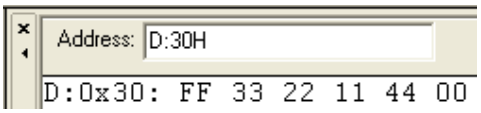
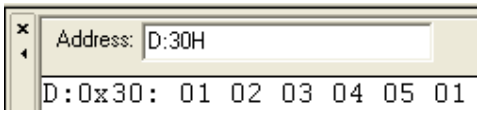
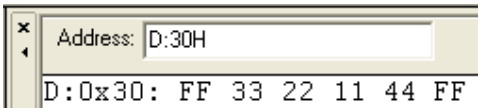


O/P:



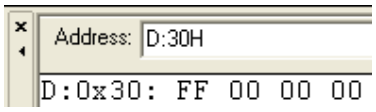
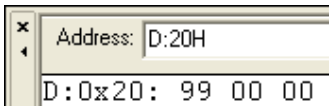
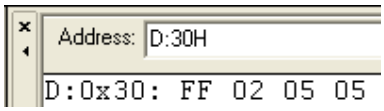
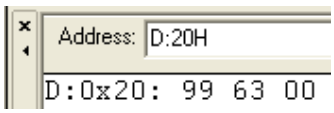
SMALLEST OF N NUMBER	LARGEST OF N NUMBER

ORG 00H
MOV R3,#04H
MOV R0,#30H
MOV A,@R0
INC R0
UP: MOV B,@R0
CJNE A,B,**NEXT**
NEXT: JC **CARRY**
MOV A,@R0
CARRY: INC R0
DJNZ R3,**UP**
MOV @R0,A
END
	ORG 00H
	MOV R3,#04H
	MOV R0,#30H
	MOV A,@R0
	INC R0
	UP: MOV B,@R0
	CJNE A,B,**NEXT**
	NEXT: JNC **NOCARRY**
	MOV A,@R0
	NOCARRY: INC R0
	DJNZ R3,**UP**
	MOV @R0,A
	END

I/P: 	**I/P:** 
O/P: 	**O/P:** 

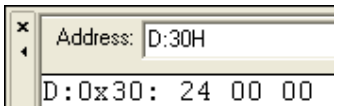
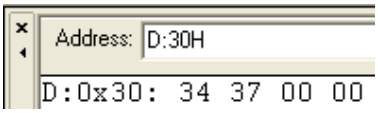
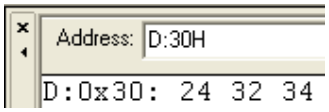
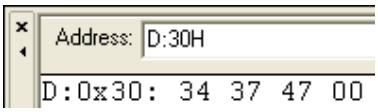
Hexadecimal to BCD	BCD to Hexadecimal

ORG 00H
MOV A,30H
MOV B,#0AH
DIV AB
MOV 33H,B
MOV B,#0AH
DIV AB
MOV 32H,B
MOV 31H,A
END
	ORG 00H
	MOV A,20H
	MOV B,#10H
	DIV AB
	MOV R2,B
	MOV B,#0AH
	MUL AB
	ADD A,R2
	MOV 21H,A
	END

I/P: 	**I/P:** 
O/P: 	**O/P:** 

BCD to ASCII	ASCII to BCD

ORG 0000
MOV A,30H
ANL A,#0F0H
SWAP A
ADD A,#30H
MOV 31H,A
MOV A,30H
ANL A,#0FH
ADD A,#30H
MOV 32H,A
END
	ORG 0000H
	MOV A,30H
	SUBB A,#30H
	SWAP A
	MOV R2,A
	MOV A,31H
	SUBB A,#30H
	ADD A,R2
	MOV 32H,A
	END

I/P: 	**I/P:** 
O/P: 	**O/P:** 

LOGIC GATES
AND, NAND, OR & NOR

```
ORG 00H
MOV C,P1.7
ANL C,P1.6
MOV P1.5,C
CPL C
MOV P1.4,C
MOV C,P1.7
ORL C,P1.6
MOV P1.3,C
CPL C
MOV P1.2,C
END
```

I/O:

A	B	Y=AB	$Y=\overline{AB}$	Y=A+B	$Y=\overline{(A+B)}$
0	0	0	1	0	1
0	1	0	1	1	0
1	0	0	1	1	0
1	1	1	0	1	0

HALF ADDER

```
ORG 00H
MOV C,P1.7
ANL C,/P1.6
MOV P1.5,C
MOV C,P1.7
CPL C
ANL C,P1.6
MOV P1.4,C
ORL C,P1.5
MOV P1.3,C
MOV C,P1.7
ANL C,P1.6
MOV P1.2,C
END
```

I/O:

A	B	C=AB	$S=\overline{A}B + A\overline{B}$
0	0	0	0
0	1	0	1
1	0	0	1
1	1	1	0

XOR GATE

```
ORG 00H
MOV C,P1.7
ANL C,/P1.6
MOV P1.5,C
MOV C,P1.7
CPL C
ANL C,P1.6
MOV P1.4,C
ORL C,P1.5
MOV P1.3,C
END
```

I/O:

A	B	$S=\overline{A}B + A\overline{B}$
0	0	0
0	1	1
1	0	1
1	1	0

XOR NOR GATE

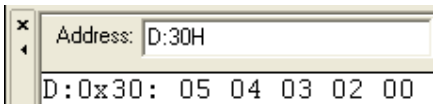
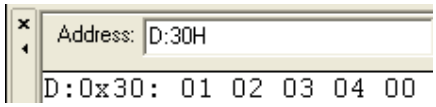
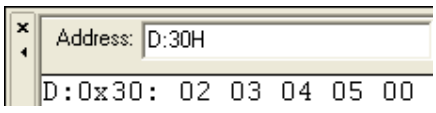
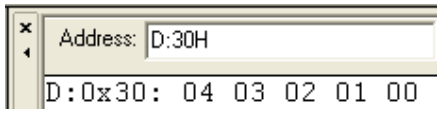
```
ORG 00H
MOV C,P1.7
CPL C
ANL C,/P1.6
MOV P1.5,C
MOV C,P1.7
ANL C,P1.6
MOV P1.4,C
ORL C,P1.5
MOV P1.3,C
END
```

I/O:

A	B	$S=\overline{AB + AB}$
0	0	0
0	1	1
1	0	1
1	1	0

ASCENDING	DESCENDING

ORG 00H
MOV R7,#03H
MAIN: MOV R0,#30H
MOV R6,#03H
UP: MOV A,@R0
INC R0
MOV B,@R0
CJNE A,B,**NEXT**
NEXT:
JC **NOEXCHANGE**
MOV @R0,A
DEC R0
MOV @R0,B
INC R0
NOEXCHANGE:
DJNZ R6,**UP**
DJNZ R7,**MAIN**
END
	ORG 00H
	MOV R7,#03H
	MAIN: MOV R0,#30H
	MOV R6,#03H
	UP: MOV A,@R0
	INC R0
	MOV B,@R0
	CJNE A,B,**NEXT**
	NEXT:
	JNC **NOEXCHANGE**
	MOV @R0,A
	DEC R0
	MOV @R0,B
	INC R0
	NOEXCHANGE:
	DJNZ R6,**UP**
	DJNZ R7,**MAIN**
	END

I/P: 	**I/P:** 
O/P: 	**O/P:** 

SERIAL COMMUNICATION	SERIAL COMMUNICATION

```
        ORG 00H
        MOV TMOD,#20H
        MOV TH1,#-3
        MOV SCON,#50H
        SETB TR1
UP:     MOV A, #'S'
        ACALL SEND
        MOV A, #'T'
        ACALL  SEND
        MOV A, #'J'
        ACALL  SEND
        MOV A, #'I'
        ACALL  SEND
        MOV A, #'T'
        ACALL  SEND
        SJMP UP


SEND :  MOV SBUF, A
HERE:   JNB TI, HERE
        CLR TI
        RET
        END
```

O/P:

```
STJITSTJITSTJI
STJITSTJITSTJI
STJITSTJITSTJI
```

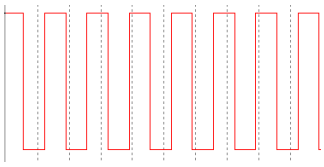
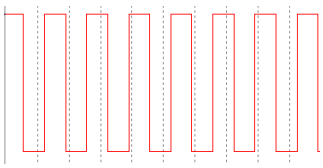
```
        ORG 00h
        MOV TMOD,#20h
        MOV TH1,#-3
        MOV SCON,#50h
        SETB TR1
REPEAT:
        MOV DPTR, #msg
UP:     CLR A
        MOVC A,@A+DPTR
        JZ REPEAT
        ACALL SEND
        INC DPTR
        SJMP  UP

SEND:
        MOV SBUF,A
HERE:   JNB TI, HERE
        CLR TI
        RET

msg:    db "STJIT",0
        END
```

O/P:

```
STJITSTJITSTJI
STJITSTJITSTJI
STJITSTJITSTJI
```

TIMER DELAY PROGRAM	TIMER DELAY PROGRAM

ORG 00H
MOV TMOD, #01H
AGAIN:
MOV TL0,#3EH
MOV TH0,#0B8H
CPL P1.7
ACALL DELAY
SJMP AGAIN
 	
DELAY:
SETB TR0
HERE: JNB TF0, HERE
CLR TR0
CLR TF0
RET
END
	ORG 00H
	MOV TMOD, #01H
	AGAIN:
	MOV TL0,#00H
	MOV TH0,#00H
	CPL P1.7
	ACALL DELAY
	SJMP AGAIN
	
	DELAY:
	SETB TR0
	HERE: JNB TF0, HERE
	CLR TR0
	CLR TF0
	RET
	END

O/P: 	**O/P:** 

COUNTERS

HEX-UP COUNTER	HEX-UP COUNTER

ORG 00H
UP:
MOV P1,A
INC A
ACALL **DELAY**
SJMP **UP**
 	
DELAY:
MOV R0,#60H
MOV R1,#0FFH
MOV R2,#0FFH
BACK:
DJNZ R2,**BACK**
DJNZ R1,**BACK**
DJNZ R0,**BACK**
RET
END
	ORG 00H
	UP:
	MOV P1,A
	INC A
	ACALL **DELAY**
	SJMP **UP**
	
	DELAY:
	MOV TL0,#00H
	MOV TH0,#00H
	SETB TR0
	HERE: JNB TF0,**HERE**
	CLR TR0
	CLR TF0
	RET
	END

HEX-DOWN COUNTER	HEX-DOWN COUNTER

ORG 00H
MOV A,#0FFH
UP:
MOV P1,A
DEC A
ACALL DELAY
SJMP UP
 	
DELAY:
MOV R0,#60H
MOV R1,#0FFH
MOV R2,#0FFH
BACK:
DJNZ R2,**BACK**
DJNZ R1,**BACK**
DJNZ R0,**BACK**
RET
END
	ORG 00H
	MOV A,#0FFH
	UP:
	MOV P1,A
	DEC A
	ACALL DELAY
	SJMP UP
	
	DELAY:
	MOV TL0,#00H
	MOV TH0,#00H
	SETB TR0
	HERE: JNB TF0,**HERE**
	CLR TR0
	CLR TF0
	RET
	END

DECIMAL-UP COUNTER	DECIMAL-UP COUNTER

ORG 00H
UP:
MOV P1,A
ADD A,#01H
DA A
ACALL DELAY
SJMP UP
 	
DELAY:
MOV R0,#60H
MOV R1,#0FFH
MOV R2,#0FFH
BACK:
DJNZ R2,**BACK**
DJNZ R1,**BACK**
DJNZ R0,**BACK**
RET
END
	ORG 00H
	UP:
	MOV P1,A
	ADD A,#01H
	DA A
	ACALL DELAY
	SJMP UP
	
	DELAY:
	MOV TL0,#00H
	MOV TH0,#00H
	SETB TR0
	HERE: JNB TF0,**HERE**
	CLR TR0
	CLR TF0
	RET
	END

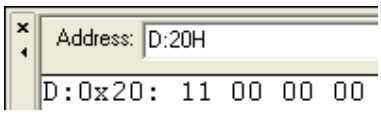
DECIMAL-DOWN COUNTER	DECIMAL-DOWN COUNTER

ORG 00H
MOV A,#99H
UP:
MOV P1,A
ADD A,#99H
DA A
ACALL DELAY
SJMP UP
 	
DELAY:
MOV R0,#60H
MOV R1,#0FFH
MOV R2,#0FFH
BACK:
DJNZ R2,**BACK**
DJNZ R1,**BACK**
DJNZ R0,**BACK**
RET
END
	ORG 00H
	MOV A,#99H
	UP:
	MOV P1,A
	ADD A,#99H
	DA A
	ACALL DELAY
	SJMP UP
	
	DELAY:
	MOV TL0,#00H
	MOV TH0,#00H
	SETB TR0
	HERE: JNB TF0,**HERE**
	CLR TR0
	CLR TF0
	RET
	END

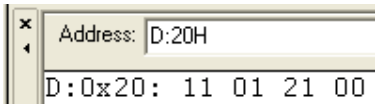
SQUARE OF A GIVEN NUMBER

```
ORG 00H
MOV A,20H
MOV B,A
MUL AB
MOV 21H,B
MOV 22H,A
END
```

I/P:



O/P:

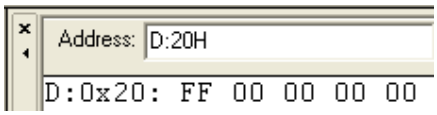


CUBE OF A GIVEN NUMBER

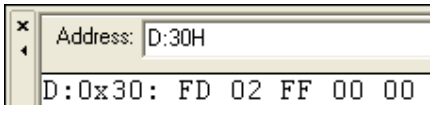
```
ORG 00H
MOV A,20H
MOV B,A
MUL AB
MOV 21H,A
MOV 22H,B
MOV A,20H
MOV B,21H
MUL AB
MOV 23H,A
MOV 24H,B
MOV A,20H
MOV B,22H
MUL AB
MOV 25H,A
MOV 26H,B
MOV 32H,23H
MOV A,24H
ADD A,25H
MOV 31H,A
MOV A,26H
ADDC A,#00H
MOV 30H,A
END
```

Cube I/O:

I/P:



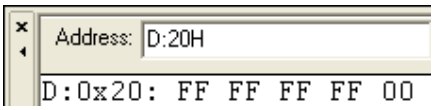
O/P:



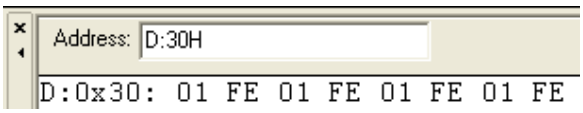
16-BIT MULTIPLICATION

```
        ORG 00H
        MOV A,20H
        MOV B,22H
        MUL AB
        MOV 30H,A
        MOV 31H,B
        MOV A,20H
        MOV B,23H
        MUL AB
        MOV 32H,A
        MOV 33H,B
        MOV A,21H
        MOV B,22H
        MUL AB
        MOV 34H,A
        MOV 35H,B
        MOV A,21H
        MOV B,23H
        MUL AB
        MOV 36H,A
        MOV 37H,B
        MOV 43H,30H
        MOV A,31H
        ADD A,32H
        JNC go
        INC R0
go:     ADD A,34H
        JNC GO1
        INC R0
go1:    MOV 42H,A
        MOV A,33H
        ADD A, R0
        JNC go2
        INC R1
go2:    ADD A,35H
        JNC go3
        INC R1
go3:    ADD A,36H
        JNC go4
        INC R1
go4:    MOV 41H,A
        MOV A,37H
        ADD A,R1
        MOV 40H,A
        END
```

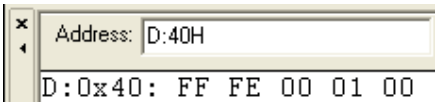
I/P:



Intermediate O/P:



O/P:



BYTE LEVEL LOGICAL OPERATION	

ORG 00H
MOV A,20H
ANL A,21H
MOV 32H,A
MOV A,20H
ORL A,21H
MOV 33H,A
MOV A,20H
XRL A,21H
MOV 34H,A
MOV A,20H
CPL A
MOV 35H,A
MOV A,20H
CLR A
MOV 36H,A
 	
MOV A,20H
SWAP A
MOV 37H,A
MOV A,20H
RR A
MOV 38H,A
MOV A,20H
RL A
MOV 39H,A
END	

HARDWARE EXPERIMENTS

NOTE: These programs work for ESA Kits

Speed control of DC Motor

```
#include<REG51XD2.H>
sbit incr=P3^2;
sbit decr=P3^3;
void main()
{
        unsigned int i=0x80;
        P0=i;                   //to rotate motor with Half Speed (full speed=ff)
                while (1)
                        {
                                if (incr==0)
                                        {
                                                if(i>10)
                                                i=i-10;
                                        }
                                if (decr==0)
                                        {
                                                if(i<0xf0)
                                                i=i+10;
                                        }

                        P0=i;
                        }
}
```

Stepper Motor

1. To rotate motor in clkwise direction and anticlockwise direction for infinite number of times.

```
#include <REG51xD2.H>
void delay (unsigned char );
sbit SW=P3^3;
void main()
{
        while(1)
                {
                        if(SW==0)
                                {
                                        P0=0x11;
                                        delay(1);
                                        P0=0x22;
                                        delay(1);
                                        P0=0x44;
                                        delay(1);
                                        P0=0x88;
                                        delay(1);
                                }
                        else
                                {
                                        P0=0x88;
                                        delay(1);
                                        P0=0x44;
                                        delay(1);
                                        P0=0x22;
```

```
                                        delay(1);
                                        P0=0x11;
                                        delay(1);
                                }
                }
}

void delay(unsigned int count)
{
        unsigned int i,j;
        for(i=0;i<count;i++)
                for(j=0;j<1275;j++);
}
```

2. To rotate motor in clkwise direction or anticlockwise direction for an angle of 360 degree.

```
#include <REG51xD2.H>
void delay (unsigned char );
sbit SW=P3^3;
void main()
{
        unsigned char i;
        for(i=0;i<50;i++)
                {
                        if(SW==0)
                                {
                                        P0=0x11;
                                        delay(1);
                                        P0=0x22;
                                        delay(1);
                                        P0=0x44;
                                        delay(1);
```

```
                                        P0=0x88;
                                        delay(1);
                                }
                        else
                                {
                                        P0=0x88;
                                        delay(1);
                                        P0=0x44;
                                        delay(1);
                                        P0=0x22;
                                        delay(1);
                                        P0=0x11;
                                        delay(1);
                                }
                }
}


void delay(unsigned int count)
{
        unsigned int i,j;
        for(i=0;i<count;i++)
                for(j=0;j<1275;j++);
}
```

3. To rotate motor in clkwise direction or anticlockwise direction for an angle of 180 degree.

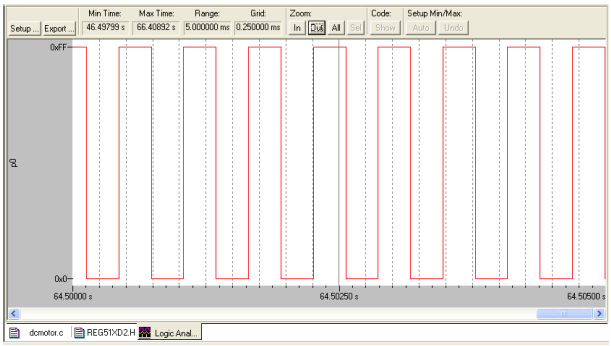
```
#include <REG51xD2.H>
void delay (unsigned char );
sbit SW=P3^3;
void main()
{
        unsigned char i;
        for(i=0;i<25;i++)
                {
                        if(SW==0)
                                {
                                        P0=0x11;
                                        delay(1);
                                        P0=0x22;
                                        delay(1);
                                        P0=0x44;
                                        delay(1);
                                        P0=0x88;
                                        delay(1);
                                }
                        else
                                {
                                        P0=0x88;
                                        delay(1);
                                        P0=0x44;
                                        delay(1);
                                        P0=0x22;
                                        delay(1);
                                        P0=0x11;
```

```
                                        delay(1);
                                    }
                        }
}


void delay(unsigned int count)
{
        unsigned int i,j;
        for(i=0;i<count;i++)
                for(j=0;j<1275;j++);
}
```

4. DAC Experiments

Simple Square Wave

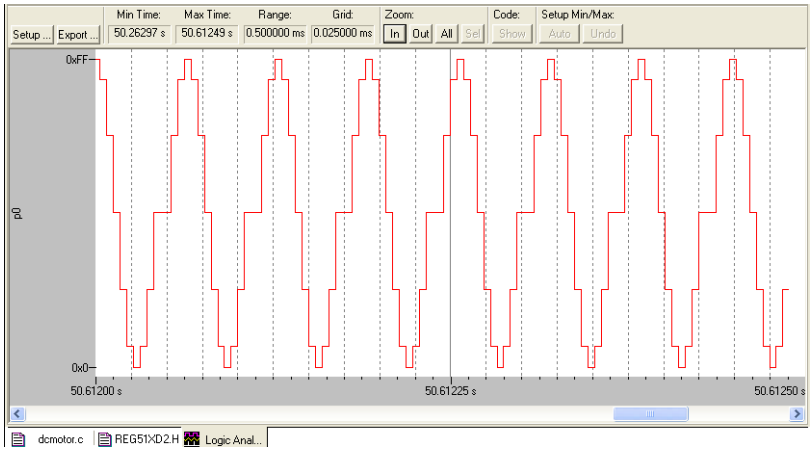
```
#include<REG51xD2.H>
void delay(unsigned int);
void main()
{
        while(1)
                {
                        P0=0x00;
                        P1=0x00;
                        delay(1);

                        P0=0xFF;
                        P1=0xFF;
                        delay(1);
                }
}
```



```
void delay(unsigned int count)
{
        unsigned int i,j;
        for(i=0; i<count; i++)
                for(j=0; j< 100; j++);
}
```

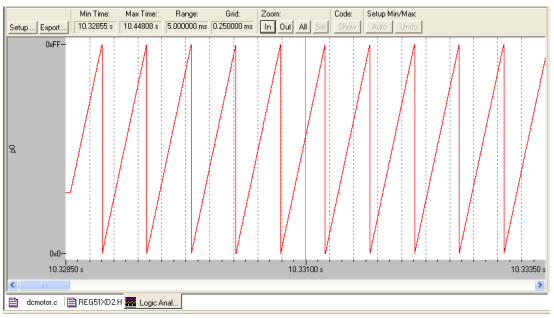
SINEWAVE GENERATION

```
#include<REG51xD2.H>
void main()
        {
        unsigned char table[]={128,192,238,255,238,192,128,64,17,0,17,64,128};
                unsigned char i;
                        while(1)
                                {
                                        for(i=0;i<13;i=i++)
                                                P0=table[i];
                                }
        }
```



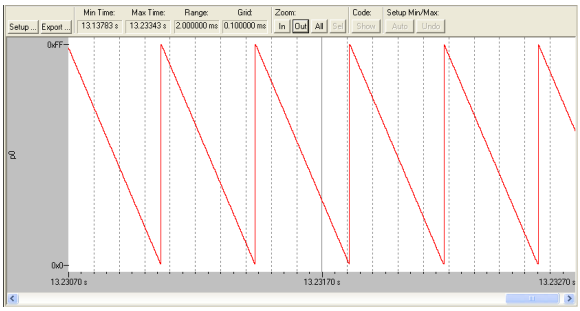
Angles θ in degree	Sinθ	Vout=5v+(5x Sinθ)	Decimal Values sent to DAC Voutx25.6
0	0	5	128
30	0.5	7.5	192
60	0.866	9.33	238
90	1	10	255
120	0.866	9.33	238
150	0.5	7.5	192
180	0	5	128
210	-0.5	2.5	64
240	-0.866	0.699	17
270	-1	0	0
300	-0.866	0.699	17
330	-0.5	2.5	64
360	0	5	128

SAWTOOTH WAVEFORM



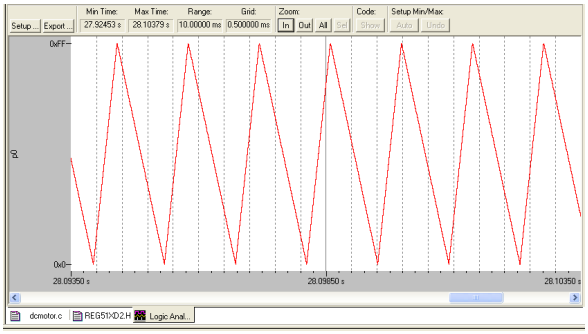
```
#include<REG51xD2.H>
void main()
        {
                unsigned char i;
                        while(1)
                                {
                                        for(i=0x00;i<0xff;i++)
                                                P0=i;
                                }
        }
```

```
#include<REG51xD2.H>
void main()
        {
                unsigned char i;
                        while(1)
                                {
                                        for(i=0xff;i>0x00;i--)
                                                P0=i;
                                }
        }
```



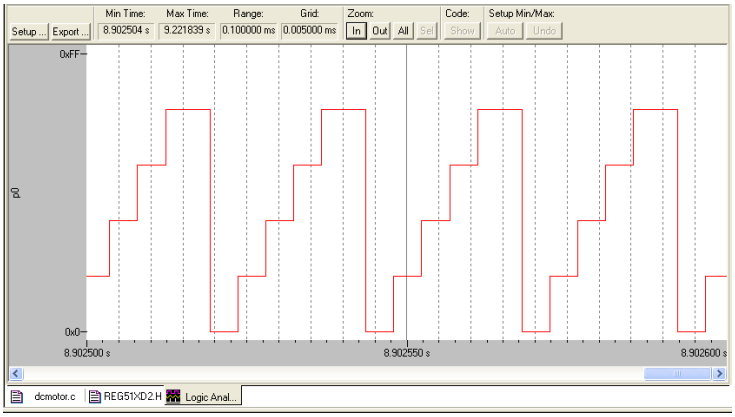
TRIANGULAR WAVE

```
#include<REG51xD2.H>
void main()
        {
                unsigned char i;
                        while(1)
                                {
                                        for(i=0;i<0xff;i++)
                                                P0=i;
                                        for(i=0xff;i>0;i--)
                                                P0=i;
                                }
        }
```

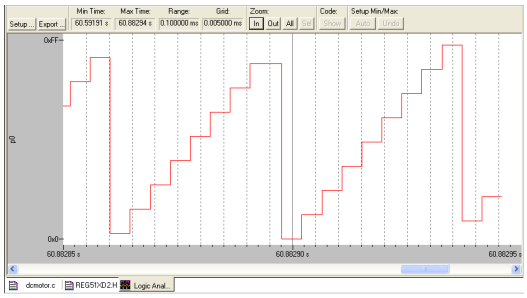


STAIRCASE WAVEFORM

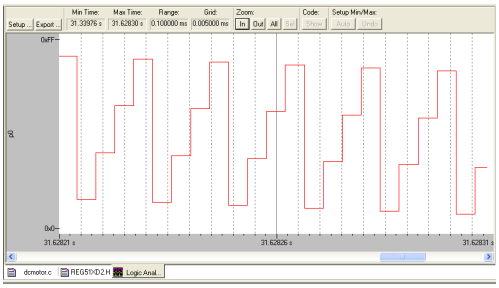
```
#include<REG51xD2.H>
void main()
	{
		unsigned char i;
			while(1)
				{
					for(i=0;i<0xff;i=i+0x33)
						P0=i;
				}
	}
```




```
#include<REG51xD2.H>
void main()
	{
		unsigned char i;
			while(1)
				{
					for(i=0;i<0xff;i=i+0x1F)
						P0=i;
				}
	}
```



```
#include<REG51xD2.H>
void main()
        {
                unsigned char i;
                        while(1)
                                {
                                        for(i=0;i<0xff;i=i+0x3F)
                                                P0=i;
                                }
        }
```



```
#include<REG51xD2.H>
void main()
        {
                unsigned char i;
                        while(1)
                                {
                                        for(i=0XFF;i>0;i=i-0x3F)
                                                P0=i;
                                }
        }
```