LAMPIRAN A

FOTO ALAT
Rangkaian penguat microphone dan low pass filter

Rangkaian Graphic Liquid Crystal Display
Rangkaian *microcontroller* dan *Liquid Crystal Display*

Rangkaian Penampil Spektrum Frekuensi Portable Berbasis Mikrokontroler ATmega16
LAMPIRAN B

KODE PROGRAM PADA ATMEGA 16
#include <mega32.h>
#include <stdio.h>
#include <stdlib.h>
#include <math.h>
#include <delay.h>

#define rs PORTD.0
#define rw PORTD.1
#define e PORTD.2
#define cs1 PORTD.3
#define cs2 PORTD.4
#define rst PORTD.5
#define data PORTC

#include <lcd.h>

#define ADC_VREF_TYPE 0x60

// Alphanumeric LCD Module functions

asm
.equ __lcd_port=0x18 ;PORTB

endasm

// Read the 8 most significant bits
// of the AD conversion result
unsigned char read_adc(unsigned char adc_input)
{
    ADMUX=adc_input | (ADC_VREF_TYPE & 0xff);
    // Start the AD conversion
    ADCSRA|=0x40;
    // Wait for the AD conversion to complete
    while ((ADCSRA & 0x10)==0);
    ADCSRA|=0x10;
    return ADCH;
}

// Declare your global variables here
flash unsigned char gambar10[8]=
{0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x80};
flash unsigned char gambar11[8]=
{0x00,0x00,0x00,0x00,0x00,0x00,0x00,0Xc0};
flash unsigned char gambar12[8]=
{0x00,0x00,0x00,0x00,0x00,0x00,0x00,0Xe0};
flash unsigned char gambar13[8]=
{0x00,0x00,0x00,0x00,0x00,0x00,0x00,0Xf0};
flash unsigned char gambar14[8]=
{0x00,0x00,0x00,0x00,0x00,0x00,0x00,0Xf8};
flash unsigned char gambar15[8]=
{0x00,0x00,0x00,0x00,0x00,0x00,0x00,0Xfc};
flash unsigned char gambar16[8]=
{0x00,0x00,0x00,0x00,0x00,0x00,0x00,0Xfe};
flash unsigned char gambar17[8]=
{0x00,0x00,0x00,0x00,0x00,0x00,0x00,0Xff};
flash unsigned char gambar20[8]=
{0x00,0x00,0x00,0x00,0x00,0x00,0x00,0X80,0Xff};
flash unsigned char gambar21[8]=
{0x00,0x00,0x00,0x00,0x00,0x00,0x00,0Xc0,0Xff};
flash unsigned char gambar22[8]=
{0x00,0x00,0x00,0x00,0x00,0x00,0x00,0Xe0,0Xff};
flash unsigned char gambar23[8]=
{0x00,0x00,0x00,0x00,0x00,0x00,0x00,0Xf0,0Xff};
flash unsigned char gambar24[8]=
{0x00,0x00,0x00,0x00,0x00,0x00,0x00,0Xf8,0Xff};
flash unsigned char gambar25[8]=
{0x00,0x00,0x00,0x00,0x00,0x00,0x00,0Xfc,0Xff};
flash unsigned char gambar26[8]=
{0x00,0x00,0x00,0x00,0x00,0x00,0x00,0Xfe,0Xff};
flash unsigned char gambar27[8]=
{0x00,0x00,0x00,0x00,0x00,0x00,0x00,0Xff,0Xff};
flash unsigned char gambar30[8]=
{0x00,0x00,0x00,0x00,0x00,0x00,0x00,0X80,0Xff};
flash unsigned char gambar31[8]=
{0x00,0x00,0x00,0x00,0x00,0x00,0x00,0Xc0,0Xff};
flash unsigned char gambar32[8]=
{0x00,0x00,0x00,0x00,0x00,0x00,0x00,0Xfe,0Xff};
flash unsigned char gambar33[8] = 
{0x00,0x00,0x00,0x00,0x00,0Xf0,0Xff,0Xff};

flash unsigned char gambar34[8] =
{0x00,0x00,0x00,0x00,0x00,0Xf8,0Xff,0Xff};

flash unsigned char gambar35[8] =
{0x00,0x00,0x00,0x00,0x00,0Xf8,0Xff,0Xff};

flash unsigned char gambar36[8] =
{0x00,0x00,0x00,0x00,0x00,0Xf8,0Xff,0Xff};

flash unsigned char gambar37[8] =
{0x00,0x00,0x00,0x00,0x00,0Xf8,0Xff,0Xff};

flash unsigned char gambar38[8] =
{0x00,0x00,0x00,0x00,0x00,0Xf8,0Xff,0Xff};

flash unsigned char gambar39[8] =
{0x00,0x00,0x00,0x00,0x00,0Xf8,0Xff,0Xff};

flash unsigned char gambar40[8] =
{0x00,0x00,0x00,0x00,0x80,0Xff,0Xff,0Xff};

flash unsigned char gambar41[8] =
{0x00,0x00,0x00,0x00,0Xc0,0Xff,0Xff,0Xff};

flash unsigned char gambar42[8] =
{0x00,0x00,0x00,0x00,0Xe0,0Xff,0Xff,0Xff};

flash unsigned char gambar43[8] =
{0x00,0x00,0x00,0x00,0Xf0,0Xff,0Xff,0Xff};

flash unsigned char gambar44[8] =
{0x00,0x00,0x00,0x00,0Xe0,0Xff,0Xff,0Xff};

flash unsigned char gambar45[8] =
{0x00,0x00,0x00,0x00,0Xf0,0Xff,0Xff,0Xff};

flash unsigned char gambar46[8] =
{0x00,0x00,0x00,0x00,0Xc0,0Xff,0Xff,0Xff};

flash unsigned char gambar47[8] =
{0x00,0x00,0x00,0x00,0Xf0,0Xff,0Xff,0FF};
flash unsigned char gambar64[8] =
{0x00, 0x00, 0xE8, 0xFF, 0xFF, 0xFF, 0xFF, 0xFF};
flash unsigned char gambar65[8] =
{0x00, 0x00, 0xFC, 0xFF, 0xFF, 0xFF, 0xFF, 0xFF};
flash unsigned char gambar66[8] =
{0x00, 0x00, 0xFE, 0xFF, 0xFF, 0xFF, 0xFF, 0xFF};
flash unsigned char gambar67[8] =
{0x00, 0x00, 0xFF, 0xFF, 0xFF, 0xFF, 0xFF, 0xFF};
flash unsigned char gambar70[8] =
{0x00, 0x80, 0xFF, 0xFF, 0xFF, 0xFF, 0xFF, 0xFF};
flash unsigned char gambar71[8] =
{0x00, 0xC0, 0xFF, 0xFF, 0xFF, 0xFF, 0xFF, 0xFF};
flash unsigned char gambar72[8] =
{0x00, 0xE0, 0xFF, 0xFF, 0xFF, 0xFF, 0xFF, 0xFF};
flash unsigned char gambar73[8] =
{0x00, 0xF0, 0xFF, 0xFF, 0xFF, 0xFF, 0xFF, 0xFF};
flash unsigned char gambar74[8] =
{0x00, 0xF8, 0xFF, 0xFF, 0xFF, 0xFF, 0xFF, 0xFF};
flash unsigned char gambar75[8] =
{0x00, 0xFC, 0xFF, 0xFF, 0xFF, 0xFF, 0xFF, 0xFF};
flash unsigned char gambar76[8] =
{0x00, 0xFE, 0xFF, 0xFF, 0xFF, 0xFF, 0xFF, 0xFF};
flash unsigned char gambar77[8] =
{0x00, 0xFF, 0xFF, 0xFF, 0xFF, 0xFF, 0xFF, 0xFF};
flash unsigned char gambar80[8] =
{0x80, 0xFF, 0xFF, 0xFF, 0xFF, 0xFF, 0xFF, 0xFF};
flash unsigned char gambar81[8] =
{0xC0, 0xFF, 0xFF, 0xFF, 0xFF, 0xFF, 0xFF, 0xFF};
flash unsigned char gambar82[8] =
{0xE0, 0xFF, 0xFF, 0xFF, 0xFF, 0xFF, 0xFF, 0xFF};
flash unsigned char gambar83[8] =
{0xF0, 0xFF, 0xFF, 0xFF, 0xFF, 0xFF, 0xFF, 0xFF};
flash unsigned char gambar84[8] =
{0xF8, 0xFF, 0xFF, 0xFF, 0xFF, 0xFF, 0xFF, 0xFF};
flash unsigned char gambar85[8] =
{0xFC, 0xFF, 0xFF, 0xFF, 0xFF, 0xFF, 0xFF, 0xFF};
flash unsigned char gambar86[8] =
{0xFE, 0xFF, 0xFF, 0xFF, 0xFF, 0xFF, 0xFF, 0xFF};
flash unsigned char gambar87[8] =
{0xFF,0xFF,0xFF,0xFF,0xFF,0xFF,0xFF,0xFF} ;
unsigned char x,y;
unsigned int n;
unsigned char adc[128];
float f[128],s,q;
char buffer[128];

void display(unsigned char a)
{
ed=1;
rs=0;
rw=0;
data=0x3E|a;
delay_us(1);
ed=0;
delay_us(1);
}

void yadd(unsigned char a)
{
ed=1;
rs=0;
rw=0;
data=0xB8|a;
delay_us(1);
ed=0;
delay_us(1);
}

void xadd(unsigned char a)
{
ed=1;
rs=0;
rw=0;
data=0x40|a;
delay_us(1);
ed=0;
delay_us(1);
}

void zadd(unsigned char a)
{
ed=1;
rs=0;
rw=0;
data=0xC0|a;
delay_us(1);
ed=0;
delay_us(1);
}

void write(unsigned char a)
{
ed=1;
rs=0;
rw=0;
data=0x40|a;
delay_us(1);
ed=0;
delay_us(1);
}
```c
{ 
e=1;
rs=1;
rw=0;
data=0x00|a;
delay_us(1);
e=0;
delay_us(1);
}
void main(void)
{
   // Declare your local variables here

   // Input/Output Ports initialization
   // Port A initialization
   // Func7=In Func6=In Func5=In Func4=In Func3=In Func2=In Func1=In Func0=In
   // State7=T State6=T State5=T State4=T State3=T State2=T State1=T State0=T
   PORTA=0x00;
   DDRA=0x00;

   // Port C initialization
   // Func7=In Func6=In Func5=In Func4=In Func3=In Func2=In Func1=In Func0=In
   // State7=T State6=T State5=T State4=T State3=T State2=T State1=T State0=T
   PORTC=0x00;
   DDRC=0xFF;

   // Port D initialization
   // Func7=In Func6=In Func5=In Func4=In Func3=In Func2=In Func1=In Func0=In
   // State7=T State6=T State5=T State4=T State3=T State2=T State1=T State0=T
   PORTD=0x00;
   DDRD=0xFF;

   // Timer/Counter 0 initialization
   // Clock source: System Clock
   // Clock value: Timer 0 Stopped
   // Mode: Normal top=FFh
   // OC0 output: Disconnected
   TCCR0=0x00;
   TCNT0=0x00;
}```
OCR0=0x00;  // Timer/Counter 2 initialization

// Timer/Counter 1 initialization
// Clock source: System Clock
// Clock value: Timer 1 Stopped
// Mode: Normal top=FF00h
// OC1A output: Discon.
// OC1B output: Discon.
// Noise Canceler: Off
// Input Capture on Falling Edge
// Timer 1 Overflow Interrupt: Off
// Input Capture Interrupt: Off
// Compare A Match Interrupt: Off
// Compare B Match Interrupt: Off
TCCR1A=0x00;  // External Interrupt(s) initialization
TCCR1B=0x00;
TCNT1H=0x00;
TCNT1L=0x00;

ASSR=0x00;
TCC2=0x00;
TCNT2=0x00;
OCR2=0x00;

// External Interrupt(s) initialization
// INT0: Off
// INT1: Off
// INT2: Off
MCUCR=0x00;
MCUCSR=0x00;
TIMSK=0x00;

// Analog Comparator initialization
ICR1H=0x00;
ICR1L=0x00;
OCR1AH=0x00;
OCR1AL=0x00;
OCR1BH=0x00;
OCR1BL=0x00;

// Analog Comparator: Off
// Analog Comparator Input Capture by Timer/Counter 1: Off
ACSR=0x80;
SFIOR=0x00;
// ADC initialization
ADMUX = ADC_VREF_TYPE & 0xff;
ADCSRA = 0x82;

// LCD module initialization
lcd_init(16);

while (1) {
    rst = 1;
    display(1);
    for (y = 0; y < 128; y++) {
        for (x = 0; x < 8; x++) {
            if (y < 64) {
                cs1 = 1;
                cs2 = 0;
            } else {
                cs1 = 0;
                cs2 = 1;
            }
            xadd(x);
            yadd(y);
        }
    }
    adc[y] = read_adc(6);
    delay_us(120);
}

for (y = 0; y < 128; y++) {
    rst = 1;
    display(1);
    for (y = 0; y < 128; y++) {
        if(y<64){
            s = 0;
            q = 0;
        } else {
            s = s + adc[n] * cos((6.2831) * y * n / 128));
            q = q + adc[n] * sin((6.2831) * y * n / 128));
        }
    }
    f[y] = sqrt((s * s + q * q));
}
else{
    if(y==64){
        f[y] = f[63];
    }
    else{
        f[y] = f[128-y];
    }
}
lcd_clear();
lcd_gotoxy(0, 0);

print(buffer, “f[%u] = %f”, y, f[y]);
lcd_puts(buffer);

for (x = 0; x < 8; x++) {
    if (y >=64&&y<=127) {
        cs1 = 1;
        cs2 = 0;
    } else  
        {  
        cs1 = 0;
        cs2 = 1;
    }
    if (f[y] <= 150) {
        xadd(x);
        yadd(y);
    } else if (f[y] > 150 && f[y] <= 175) {
        xadd(x);
        yadd(y);
        zadd(0);
        write(gambar10[x]);
        delay_us(1);
    } else if (f[y] >= 175 && f[y] <= 200) {
        xadd(x);
        yadd(y);
        zadd(0);
        write(gambar11[x]);
        delay_us(1);
    } else if (f[y] >= 200 && f[y] <= 225)
        {  
        xadd(x);
        yadd(y);
        zadd(0);
        write(gambar12[x]);
        delay_us(1);
    } else if (f[y] >= 225 && f[y] <= 250)
        {  
        xadd(x);
        yadd(y);
        zadd(0);
        write(gambar13[x]);
        delay_us(1);
    } else if (f[y] >= 225 && f[y] <= 250) {
        xadd(x);
        yadd(y);
        zadd(0);
        write(gambar14[x]);
        delay_us(1);
    }
}
yadd(y);
zadd(0);
write(gambar14[x]);
delay_us(1);
}
else if (f[y] >= 250 && f[y] <= 275) {
xadd(x);
yadd(y);
zadd(0);
write(gambar15[x]);
delay_us(1);
}
else if (f[y] >= 275 && f[y] <= 300) {
xadd(x);
yadd(y);
zadd(0);
write(gambar16[x]);
delay_us(1);
}
else if (f[y] >= 300 && f[y] <= 325) {
xadd(x);
yadd(y);
zadd(0);
write(gambar17[x]);
delay_us(1);
}
else if (f[y] >= 325 && f[y] <= 350) {
xadd(x);
yadd(y);
zadd(0);
write(gambar20[x]);
delay_us(1);
}
else if (f[y] >= 350 && f[y] <= 375) {
xadd(x);
yadd(y);
zadd(0);
write(gambar21[x]);
delay_us(1);
}
else if (f[y] >= 375 && f[y] <= 400) {
xadd(x);
yadd(y);
zadd(0);
write(gambar22[x]);
delay_us(1);
}
else if (f[y] >= 400 && f[y] <= 425) {
xadd(x);
yadd(y);
zadd(0);
write(gambar23[x]);
delay_us(1);
}
else if (f[y] >= 425 && f[y] <= 450) {
xadd(x);
yadd(y);
zadd(0);
write(gambar24[x]);
delay_us(1);
}
else if (f[y] >= 450 && f[y] <= 475) {
xadd(x);
yadd(y);
zadd(0);
write(gambar25[x]);
delay_us(1);
}
else if (f[y] >= 475 && f[y] <= 500) {
xadd(x);
yadd(y);
zadd(0);
write(gambar26[x]);
delay_us(1);
}
else if (f[y] >= 500 && f[y] <= 525) {
xadd(x);
yadd(y);
zadd(0);
write(gambar27[x]);
delay_us(1);
}
else if (f[y] >= 525 && f[y] <= 550) {
xadd(x);
yadd(y);
zadd(0);
write(gambar28[x]);
delay_us(1);
}
else if (f[y] >= 550 && f[y] <= 575) {
xadd(x);
yadd(y);
zadd(0);
write(gambar29[x]);
delay_us(1);
}
else if (f[y] >= 575 && f[y] <= 600) {
xadd(x);
yadd(y);
zadd(0);
write(gambar30[x]);
delay_us(1);
}
else if (f[y] >= 600 && f[y] <= 625) {
xadd(x);
yadd(y);
zadd(0);
write(gambar31[x]);
delay_us(1);
}
else if (f[y] >= 625 && f[y] <= 650) {
xadd(x);
yadd(y);
zadd(0);
write(gambar32[x]);
delay_us(1);
}
else if (f[y] >= 650 && f[y] <= 675) {
xadd(x);
yadd(y);
zadd(0);
write(gambar33[x]);
delay_us(1);
}
else if (f[y] >= 675 && f[y] <= 700) {
xadd(x);
yadd(y);
zadd(0);
write(gambar34[x]);
delay_us(1);
else if (f[y] >= 650 && f[y] <= 675) {
  xadd(x);
  yadd(y);
  zadd(0);
  write(gambar35[x]);
  delay_us(1);
}

else if (f[y] > 675 && f[y] <= 700) {
  xadd(x);
  yadd(y);
  zadd(0);
  write(gambar36[x]);
  delay_us(1);
}

else if (f[y] >= 700 && f[y] <= 725) {
  xadd(x);
  yadd(y);
  zadd(0);
  write(gambar37[x]);
  delay_us(1);
}

else if (f[y] >= 725 && f[y] <= 750) {
  xadd(x);
  yadd(y);
  zadd(0);
  write(gambar40[x]);
  delay_us(1);
}

else if (f[y] >= 750 && f[y] <= 775) {
  xadd(x);
  yadd(y);
  zadd(0);
  write(gambar41[x]);
  delay_us(1);
}

else if (f[y] >= 775 && f[y] <= 800) {
  xadd(x);
  yadd(y);
  zadd(0);
  write(gambar42[x]);
  delay_us(1);
}

else if (f[y] >= 800 && f[y] <= 825) {
  xadd(x);
  yadd(y);
  zadd(0);
  write(gambar43[x]);
  delay_us(1);
}

else if (f[y] >= 825 && f[y] <= 850) {
  xadd(x);
  yadd(y);
  zadd(0);
  write(gambar44[x]);
  delay_us(1);
}

else if (f[y] >= 850 && f[y] <= 875) {
  xadd(x);
  yadd(y);
  zadd(0);
  write(gambar45[x]);
  delay_us(1);
}

else if (f[y] >= 875 && f[y] <= 900) {
  xadd(x);
  yadd(y);
  zadd(0);
  write(gambar46[x]);
  delay_us(1);
}
xadd(x);
yadd(y);
zadd(0);
write(gambar45[x]);
delay_us(1);
} else if (f[y] >= 875 && f[y] <= 900) {
xadd(x);
yadd(y);
zadd(0);
write(gambar46[x]);
delay_us(1);
} else if (f[y] >= 900 && f[y] <= 925) {
xadd(x);
yadd(y);
zadd(0);
write(gambar47[x]);
delay_us(1);
} else if (f[y] >= 925 && f[y] <= 950) {
xadd(x);
yadd(y);
zadd(0);
write(gambar50[x]);
delay_us(1);
} else if (f[y] >= 950 && f[y] <= 975) {
xadd(x);
yadd(y);
zadd(0);
write(gambar51[x]);
delay_us(1);
} else if (f[y] >= 975 && f[y] <= 1000) {
xadd(x);
yadd(y);
zadd(0);
write(gambar52[x]);
delay_us(1);
} else if (f[y] >= 1000 && f[y] <= 1025) {
xadd(x);
yadd(y);
zadd(0);
write(gambar53[x]);
delay_us(1);
} else if (f[y] >= 1025 && f[y] <= 1050) {
xadd(x);
yadd(y);
zadd(0);
write(gambar54[x]);
delay_us(1);
} else if (f[y] >= 1050 && f[y] <= 1075) {
xadd(x);
yadd(y);
zadd(0);
write(gambar55[x]);
delay_us(1);
}
else if (f[y] >= 1075 && f[y] <= 1100)
{
xadd(x);
yadd(y);
zadd(0);
write(gambar56[x]);
delay_us(1);
}
else if (f[y] >= 1100 && f[y] <= 1125)
{
xadd(x);
yadd(y);
zadd(0);
write(gambar57[x]);
delay_us(1);
}
else if (f[y] >= 1125 && f[y] <= 1150)
{
xadd(x);
yadd(y);
zadd(0);
write(gambar60[x]);
delay_us(1);
}
else if (f[y] >= 1150 && f[y] <= 1175)
{
write(gambar61[x]);
delay_us(1);
}
else if (f[y] >= 1175 && f[y] <= 1200)
{
xadd(x);
yadd(y);
zadd(0);
write(gambar62[x]);
delay_us(1);
}
else if (f[y] >= 1200 && f[y] <= 1225)
{
xadd(x);
yadd(y);
zadd(0);
write(gambar63[x]);
delay_us(1);
}
else if (f[y] >= 1225 && f[y] <= 1250)
{
xadd(x);
yadd(y);
zadd(0);
write(gambar64[x]);
delay_us(1);
}
zadd(0);
write(gambar65[x]);
delay_us(1);
} else if (f[y] >= 1275 && f[y] <= 1300)
{
  xadd(x);
  yadd(y);
  zadd(0);
  write(gambar66[x]);
  delay_us(1);
} else if (f[y] >= 1300 && f[y] <= 1325)
{
  xadd(x);
  yadd(y);
  zadd(0);
  write(gambar67[x]);
  delay_us(1);
} else if (f[y] >= 1325 && f[y] <= 1350) {
  xadd(x);
  yadd(y);
  zadd(0);
  write(gambar70[x]);
  delay_us(1);
} else if (f[y] >= 1350 && f[y] <= 1375) {
  xadd(x);
  yadd(y);
  zadd(0);
  write(gambar71[x]);
  delay_us(1);
} else if (f[y] >= 1375 && f[y] <= 1400) {
  xadd(x);
  yadd(y);
  zadd(0);
  write(gambar72[x]);
  delay_us(1);
} else if (f[y] >= 1400 && f[y] <= 1425) {
  xadd(x);
  yadd(y);
  zadd(0);
  write(gambar73[x]);
  delay_us(1);
} else if (f[y] >= 1425 && f[y] <= 1450) {
  xadd(x);
  yadd(y);
  zadd(0);
  write(gambar74[x]);
  delay_us(1);
} else if (f[y] >= 1450 && f[y] <= 1475) {
  xadd(x);
  yadd(y);
  zadd(0);
  write(gambar75[x]);
  delay_us(1);
} else if (f[y] >= 1475 && f[y] <= 1500) {
  xadd(x);
  yadd(y);
  zadd(0);
  write(gambar76[x]);
  delay_us(1);
} else if (f[y] >= 1500 && f[y] <= 1525) {
  xadd(x);
  yadd(y);
  zadd(0);
  write(gambar77[x]);
  delay_us(1);
} else if (f[y] >= 1525 && f[y] <= 1550) {
  xadd(x);
  yadd(y);
  zadd(0);
  write(gambar78[x]);
  delay_us(1);
} else if (f[y] >= 1550 && f[y] <= 1575) {
  xadd(x);
  yadd(y);
  zadd(0);
  write(gambar79[x]);
  delay_us(1);
} else if (f[y] >= 1575 && f[y] <= 1600) {
  xadd(x);
  yadd(y);
  zadd(0);
  write(gambar80[x]);
  delay_us(1);
} else if (f[y] >= 1600 && f[y] <= 1625) {
  xadd(x);
  yadd(y);
  zadd(0);
  write(gambar81[x]);
  delay_us(1);
} else if (f[y] >= 1625 && f[y] <= 1650) {
  xadd(x);
  yadd(y);
  zadd(0);
  write(gambar82[x]);
  delay_us(1);
} else if (f[y] >= 1650 && f[y] <= 1675) {
  xadd(x);
  yadd(y);
  zadd(0);
  write(gambar83[x]);
  delay_us(1);
} else if (f[y] >= 1675 && f[y] <= 1700) {
  xadd(x);
  yadd(y);
  zadd(0);
  write(gambar84[x]);
  delay_us(1);
} else if (f[y] >= 1700 && f[y] <= 1725) {
  xadd(x);
  yadd(y);
  zadd(0);
  write(gambar85[x]);
  delay_us(1);
} else if (f[y] >= 1725 && f[y] <= 1750) {
  xadd(x);
  yadd(y);
  zadd(0);
  write(gambar86[x]);
  delay_us(1);
} else if (f[y] >= 1750 && f[y] <= 1775) {
  xadd(x);
  yadd(y);
  zadd(0);
  write(gambar87[x]);
  delay_us(1);
} else if (f[y] >= 1775 && f[y] <= 1800) {
  xadd(x);
  yadd(y);
  zadd(0);
  write(gambar88[x]);
  delay_us(1);
} else if (f[y] >= 1800 && f[y] <= 1825) {
  xadd(x);
  yadd(y);
  zadd(0);
  write(gambar89[x]);
  delay_us(1);
} else if (f[y] >= 1825 && f[y] <= 1850) {
  xadd(x);
  yadd(y);
  zadd(0);
  write(gambar90[x]);
  delay_us(1);
} else if (f[y] >= 1850 && f[y] <= 1875) {
  xadd(x);
  yadd(y);
  zadd(0);
  write(gambar91[x]);
  delay_us(1);
} else if (f[y] >= 1875 && f[y] <= 1900) {
  xadd(x);
  yadd(y);
  zadd(0);
  write(gambar92[x]);
  delay_us(1);
} else if (f[y] >= 1900 && f[y] <= 1925) {
  xadd(x);
  yadd(y);
  zadd(0);
  write(gambar93[x]);
  delay_us(1);
} else if (f[y] >= 1925 && f[y] <= 1950) {
  xadd(x);
  yadd(y);
  zadd(0);
  write(gambar94[x]);
  delay_us(1);
} else if (f[y] >= 1950 && f[y] <= 1975) {
  xadd(x);
  yadd(y);
  zadd(0);
  write(gambar95[x]);
  delay_us(1);
} else if (f[y] >= 1975 && f[y] <= 2000) {
  xadd(x);
  yadd(y);
  zadd(0);
  write(gambar96[x]);
  delay_us(1);
}
xadd(x);
yadd(y);
zadd(0);
write(gambar75[x]);
delay_us(1);
}
else if (f[y] >= 1590 && f[y] <= 1625) {
xadd(x);
yadd(y);
zadd(0);
write(gambar76[x]);
delay_us(1);
}
else if (f[y] >= 1625 && f[y] <= 1650) {
xadd(x);
yadd(y);
zadd(0);
write(gambar77[x]);
delay_us(1);
}
else if (f[y] >= 1650 && f[y] <= 1675) {
xadd(x);
yadd(y);
zadd(0);
write(gambar78[x]);
delay_us(1);
}
else if (f[y] >= 1675 && f[y] <= 1700) {
xadd(x);
yadd(y);
zadd(0);
write(gambar79[x]);
delay_us(1);
}
delay_us(1);
} else if (f[y] >= 1650 && f[y] <= 1675)
{
    xadd(x);
    yadd(y);
    zadd(0);
    write(gambar85[x]);
    delay_us(1);
}
else if (f[y] >= 1675 && f[y] <= 1700)
{
    xadd(x);
    yadd(y);
    zadd(0);
    write(gambar86[x]);
    delay_us(1);
}
else if (f[y] >= 1700 && f[y] <= 1725) {
    xadd(x);
    yadd(y);
    zadd(0);
    write(gambar87[x]);
    delay_us(1);
}
else if (f[y] > 1725) {
    xadd(x);
    yadd(y);
    zadd(0);
    write(gambar87[x]);
    delay_us(1);
}
cs1 = 1;
cs2 = 1;
delay_ms(6000);
LAMPIRAN C
DATASHEET ATMega 16
Features

• High-performance, Low-power AVR® 8-bit Microcontroller
• Advanced RISC Architecture
  – 131 Powerful Instructions – Most Single-clock Cycle Execution
  – 32 x 8 General Purpose Working Registers
  – Fully Static Operation
  – Up to 16 MIPS Throughput at 16 MHz
  – On-chip 2-cycle Multiplier
• Nonvolatile Program and Data Memories
  – 16K Bytes of In-System Self-Programmable Flash
  – Optional Boot Code Section with Independent Lock Bits
  – In-System Programming by On-chip Boot Program
  – True Read-While-Write Operation
  – 512 Bytes EEPROM
  – 1K Byte Internal SRAM
  – Programming Lock for Software Security
• JTAG (IEEE std. 1149.1 Compliant) Interface
  – Boundary-scan Capabilities According to the JTAG Standard
  – Extensive On-chip Debug Support
  – Programming of Flash, EEPROM, Fuses, and Lock Bits through the JTAG Interface
• Peripheral Features
  – Two 8-bit Timer/Counters with Separate Prescalers and Compare Modes
  – One 16-bit Timer/Counter with Separate Prescaler, Compare Mode, and Capture Mode
  – Real Time Counter with Separate Oscillator
  – Four PWM Channels
  – 8-channel, 10-bit ADC
  – 8 Single-ended Channels
  – 7 Differential Channels in TQFP Package Only
  – 2 Differential Channels with Programmable Gain at 1x, 10x, or 200x
  – Byte-oriented Two-wire Serial Interface
  – Programmable Serial USART
  – Master/Slave SPI Serial Interface
  – Programmable Watchdog Timer with Separate On-chip Oscillator
  – On-chip Analog Comparator
• Special Microcontroller Features
  – Power-on Reset and Programmable Brown-out Detection
  – Internal Calibrated RC Oscillator
  – External and Internal Interrupt Sources
  – Six Sleep Modes: Idle, ADC Noise Reduction, Power-save, Power-down, Standby and Extended Standby
• I/O and Packages
  – 32 Programmable I/O Lines
  – 40-pin PDIP, 44-lead TQFP, and 44-pad MLF
• Operating Voltages
  – 2.7 - 5.5V for ATmega16L
  – 4.5 - 5.5V for ATmega16
• Speed Grades
  – 0 - 8 MHz for ATmega16L
  – 0 - 16 MHz for ATmega16
• Power Consumption @ 1 MHz, 3V, and 25°C for ATmega16L
  – Active: 1.1 mA
  – Idle Mode: 0.35 mA
  – Power-down Mode: < 1 µA
**Pin Configurations**  Figure 1. Pinouts ATmega16

**Disclaimer**  Typical values contained in this datasheet are based on simulations and characterization of other AVR microcontrollers manufactured on the same process technology. Min and Max values will be available after the device is characterized.
**Overview** The ATmega16 is a low-power CMOS 8-bit microcontroller based on the AVR enhanced RISC architecture. By executing powerful instructions in a single clock cycle, the ATmega16 achieves throughputs approaching 1 MIPS per MHz allowing the system designer to optimize power consumption versus processing speed.

**Block Diagram** Figure 2. Block Diagram
The AVR core combines a rich instruction set with 32 general purpose working registers. All the 32 registers are directly connected to the Arithmetic Logic Unit (ALU), allowing two independent registers to be accessed in one single instruction executed in one clock cycle. The resulting architecture is more code efficient while achieving throughputs up to ten times faster than conventional CISC microcontrollers.

The ATmega16 provides the following features: 16K bytes of In-System Programmable Flash Program memory with Read-While-Write capabilities, 512 bytes EEPROM, 1K byte SRAM, 32 general purpose I/O lines, 32 general purpose working registers, a JTAG interface for Boundary-scan, On-chip Debugging support and programming, three flexible Timer/Counters with compare modes, Internal and External Interrupts, a serial programmable USART, a byte oriented Two-wire Serial Interface, an 8-channel, 10-bit ADC with optional differential input stage with programmable gain (TQFP package only), a programmable Watchdog Timer with Internal Oscillator, an SPI serial port, and six software selectable power saving modes. The Idle mode stops the CPU while allowing the USART, Two-wire interface, A/D Converter, SRAM, Timer/Counters, SPI port, and interrupt system to continue functioning. The Power-down mode saves the register contents but freezes the Oscillator, disabling all other chip functions until the next External Interrupt or Hardware Reset. In Power-save mode, the Asynchronous Timer continues to run, allowing the user to maintain a timer base while the rest of the device is sleeping. The ADC Noise Reduction mode stops the CPU and all I/O modules except Asynchronous Timer and ADC, to minimize switching noise during ADC conversions. In Standby mode, the crystal/resonator Oscillator is running while the rest of the device is sleeping. This allows very fast start-up combined with low-power consumption. In Extended Standby mode, both the main Oscillator and the Asynchronous Timer continue to run. The device is manufactured using Atmel's high density nonvolatile memory technology.

The On-chip ISP Flash allows the program memory to be reprogrammed in-system through an SPI serial interface, by a conventional nonvolatile memory programmer, or by an On-chip Boot program running on the AVR core. The boot program can use any interface to download the application program in the Application Flash memory. Software in the Boot Flash section will continue to run while the Application Flash section is updated, providing true Read-While-Write operation. By combining an 8-bit RISC CPU with In-System Self-Programmable Flash on a monolithic chip, the Atmel ATmega16 is a powerful microcontroller that provides a highly-flexible and cost-effective solution to many embedded control applications. The ATmega16 AVR is supported with a full suite of program and system development tools including: C compilers, macro assemblers, program debugger/simulators, in-circuit emulators, and evaluation kits.

### Pin Descriptions

**VCC** Digital supply voltage.

**GND** Ground.

**Port A (PA7..PA0)** Port A serves as the analog inputs to the A/D Converter. Port A also serves as an 8-bit bi-directional I/O port, if the A/D Converter is not used. Port pins can provide internal pull-up resistors (selected for each bit). The Port A output buffers have symmetrical drive characteristics with both high sink and source capability. When pins PA0 to PA7 are used as inputs and are externally pulled low, they will source current if the internal pull-up resistors are activated. The Port A pins are tri-stated when a reset condition becomes active, even if the clock is not running.

**Port B (PB7..PB0)** Port B is an 8-bit bi-directional I/O port with internal pull-up resistors (selected for each bit). The Port B output buffers have symmetrical drive characteristics with both high sink and source capability. As inputs, Port B pins that are externally pulled low will source current if the pull-up resistors are activated. The Port B pins are tri-stated when a reset condition becomes active, even if the clock is not running. Port B also serves the functions of various special features of the ATmega16 as listed on page 56.

**Port C (PC7..PC0)** Port C is an 8-bit bi-directional I/O port with internal pull-up resistors (selected for each bit). The Port C output buffers have symmetrical drive characteristics with both high sink and source
Capability. As inputs, Port C pins that are externally pulled low will source current if the pull-up resistors are activated. The Port C pins are tri-stated when a reset condition becomes active, even if the clock is not running. If the JTAG interface is enabled, the pull-up resistors on pins PC5(TDI), PC3(TMS) and PC2(TCK) will be activated even if a reset occurs. Port C also serves the functions of the JTAG interface and other special features of the ATmega16 as listed on page 59.

**Port D (PD7..PD0)** Port D is an 8-bit bi-directional I/O port with internal pull-up resistors (selected for each bit). The Port D output buffers have symmetrical drive characteristics with both high sink and source capability. As inputs, Port D pins that are externally pulled low will source current if the pull-up resistors are activated. The Port D pins are tri-stated when a reset condition becomes active, even if the clock is not running. Port D also serves the functions of various special features of the ATmega16 as listed on page 61.

**RESET** Reset Input. A low level on this pin for longer than the minimum pulse length will generate a reset, even if the clock is not running. The minimum pulse length is given in Table 15 on page 36. Shorter pulses are not guaranteed to generate a reset.

**XTAL1** Input to the inverting Oscillator amplifier and input to the internal clock operating circuit.

**XTAL2** Output from the inverting Oscillator amplifier.

**AVCC** AVCC is the supply voltage pin for Port A and the A/D Converter. It should be externally connected to Vcc, even if the ADC is not used. If the ADC is used, it should be connected to Vcc through a low-pass filter.

**AREF** AREF is the analog reference pin for the A/D Converter.

---

### Register Summary

<table>
<thead>
<tr>
<th>Address</th>
<th>Name</th>
<th>Bit 7</th>
<th>Bit 6</th>
<th>Bit 5</th>
<th>Bit 4</th>
<th>Bit 3</th>
<th>Bit 2</th>
<th>Bit 1</th>
<th>Bit 0</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>0000 0000</td>
<td>C000</td>
<td>B</td>
<td>B</td>
<td>B</td>
<td>B</td>
<td>B</td>
<td>B</td>
<td>B</td>
<td>B</td>
<td>C-5</td>
</tr>
<tr>
<td>0000 0001</td>
<td>D000</td>
<td>B</td>
<td>B</td>
<td>B</td>
<td>B</td>
<td>B</td>
<td>B</td>
<td>B</td>
<td>B</td>
<td>C-5</td>
</tr>
<tr>
<td>0000 0010</td>
<td>D001</td>
<td>B</td>
<td>B</td>
<td>B</td>
<td>B</td>
<td>B</td>
<td>B</td>
<td>B</td>
<td>B</td>
<td>C-5</td>
</tr>
</tbody>
</table>

---

C-5
1. When the OCDEN Fuse is unprogrammed, the OSCCAL Register is always accessed on this address. Refer to the debugger specific documentation for details on how to use the OCDR Register.

2. Refer to the USART description for details on how to access UBRRH and UCSRC.

3. For compatibility with future devices, reserved bits should be written to zero if accessed. Reserved I/O memory addresses should never be written.

4. Some of the Status Flags are cleared by writing a logical one to them. Note that the CBI and SBI instructions will operate on all bits in the I/O Register, writing a one back into any flag read as set, thus clearing the flag. The CBI and SBI instructions work with registers $00 to $1F only.
<table>
<thead>
<tr>
<th>Mnemonics</th>
<th>Operands</th>
<th>Description</th>
<th>Operation</th>
<th>Flags</th>
<th>#Cycles</th>
</tr>
</thead>
<tbody>
<tr>
<td>MOV</td>
<td>M,A</td>
<td>Move Between Registers</td>
<td>Rm ≤ Ap, Rm ≤ A</td>
<td>Rm ← Rm</td>
<td>1,2</td>
</tr>
<tr>
<td>MOVX</td>
<td>M,A</td>
<td>Move Register to Port</td>
<td>Rm = (AP)</td>
<td>Rm ← Rm</td>
<td>1,2</td>
</tr>
<tr>
<td>INC</td>
<td>M,A</td>
<td>Inc. Register</td>
<td>Rm ≤ A</td>
<td>Rm ← Rm+1</td>
<td>1,2</td>
</tr>
<tr>
<td>DEC</td>
<td>M,A</td>
<td>Dec. Register</td>
<td>Rm ≤ A</td>
<td>Rm ← Rm-1</td>
<td>1,2</td>
</tr>
<tr>
<td>ADD</td>
<td>M,A</td>
<td>Add Registers</td>
<td>Rm ≤ Ap, Rm ≤ A</td>
<td>Rm ← Rm+Rm</td>
<td>1,2</td>
</tr>
<tr>
<td>SUB</td>
<td>M,A</td>
<td>Subtract Registers</td>
<td>Rm ≤ Ap, Rm ≤ A</td>
<td>Rm ← Rm-Rm</td>
<td>1,2</td>
</tr>
<tr>
<td>AND</td>
<td>M,A</td>
<td>AND Registers</td>
<td>Rm ≤ Ap, Rm ≤ A</td>
<td>Rm ← Rm&amp;Mm</td>
<td>1,2</td>
</tr>
<tr>
<td>OR</td>
<td>M,A</td>
<td>OR Registers</td>
<td>Rm ≤ Ap, Rm ≤ A</td>
<td>Rm ← Rm</td>
<td>Mm</td>
</tr>
<tr>
<td>XOR</td>
<td>M,A</td>
<td>XOR Registers</td>
<td>Rm ≤ Ap, Rm ≤ A</td>
<td>Rm ← Rm</td>
<td>Mm</td>
</tr>
<tr>
<td>NOT</td>
<td>M,A</td>
<td>NOT Register</td>
<td>Rm ≤ A</td>
<td>Rm ← ~Rm</td>
<td>1,2</td>
</tr>
<tr>
<td>MOVF</td>
<td>M,A</td>
<td>Move File Register</td>
<td>Rm ≤ Ap, Rm ≤ A</td>
<td>Rm ← Rm</td>
<td>1,2</td>
</tr>
<tr>
<td>MOVWF</td>
<td>M,A</td>
<td>Move File Register to Port</td>
<td>Rm = (AP)</td>
<td>Rm ← Rm</td>
<td>1,2</td>
</tr>
<tr>
<td>INCX</td>
<td>M,A</td>
<td>Inc. X Register</td>
<td>Rm ≤ X</td>
<td>Rm ← Rm+1</td>
<td>1,2</td>
</tr>
<tr>
<td>DECX</td>
<td>M,A</td>
<td>Dec. X Register</td>
<td>Rm ≤ X</td>
<td>Rm ← Rm-1</td>
<td>1,2</td>
</tr>
<tr>
<td>ADDX</td>
<td>M,A</td>
<td>Add X Register</td>
<td>Rm ≤ Ap, Rm ≤ A</td>
<td>Rm ← Rm+Rm</td>
<td>1,2</td>
</tr>
<tr>
<td>SUBX</td>
<td>M,A</td>
<td>Subtract X Register</td>
<td>Rm ≤ Ap, Rm ≤ A</td>
<td>Rm ← Rm-Rm</td>
<td>1,2</td>
</tr>
<tr>
<td>ANDX</td>
<td>M,A</td>
<td>AND X Register</td>
<td>Rm ≤ Ap, Rm ≤ A</td>
<td>Rm ← Rm&amp;Mm</td>
<td>1,2</td>
</tr>
<tr>
<td>ORX</td>
<td>M,A</td>
<td>OR X Register</td>
<td>Rm ≤ Ap, Rm ≤ A</td>
<td>Rm ← Rm</td>
<td>Mm</td>
</tr>
<tr>
<td>XORX</td>
<td>M,A</td>
<td>XOR X Register</td>
<td>Rm ≤ Ap, Rm ≤ A</td>
<td>Rm ← Rm</td>
<td>Mm</td>
</tr>
<tr>
<td>NOTX</td>
<td>M,A</td>
<td>NOT X Register</td>
<td>Rm ≤ X</td>
<td>Rm ← ~Rm</td>
<td>1,2</td>
</tr>
<tr>
<td>MOVF</td>
<td>M,A</td>
<td>Move File Register</td>
<td>Rm ≤ Ap, Rm ≤ A</td>
<td>Rm ← Rm</td>
<td>1,2</td>
</tr>
<tr>
<td>MOVWF</td>
<td>M,A</td>
<td>Move File Register to Port</td>
<td>Rm = (AP)</td>
<td>Rm ← Rm</td>
<td>1,2</td>
</tr>
<tr>
<td>INCX</td>
<td>M,A</td>
<td>Inc. X Register</td>
<td>Rm ≤ X</td>
<td>Rm ← Rm+1</td>
<td>1,2</td>
</tr>
<tr>
<td>DECX</td>
<td>M,A</td>
<td>Dec. X Register</td>
<td>Rm ≤ X</td>
<td>Rm ← Rm-1</td>
<td>1,2</td>
</tr>
<tr>
<td>ADDX</td>
<td>M,A</td>
<td>Add X Register</td>
<td>Rm ≤ Ap, Rm ≤ A</td>
<td>Rm ← Rm+Rm</td>
<td>1,2</td>
</tr>
<tr>
<td>SUBX</td>
<td>M,A</td>
<td>Subtract X Register</td>
<td>Rm ≤ Ap, Rm ≤ A</td>
<td>Rm ← Rm-Rm</td>
<td>1,2</td>
</tr>
<tr>
<td>ANDX</td>
<td>M,A</td>
<td>AND X Register</td>
<td>Rm ≤ Ap, Rm ≤ A</td>
<td>Rm ← Rm&amp;Mm</td>
<td>1,2</td>
</tr>
<tr>
<td>ORX</td>
<td>M,A</td>
<td>OR X Register</td>
<td>Rm ≤ Ap, Rm ≤ A</td>
<td>Rm ← Rm</td>
<td>Mm</td>
</tr>
<tr>
<td>XORX</td>
<td>M,A</td>
<td>XOR X Register</td>
<td>Rm ≤ Ap, Rm ≤ A</td>
<td>Rm ← Rm</td>
<td>Mm</td>
</tr>
<tr>
<td>NOTX</td>
<td>M,A</td>
<td>NOT X Register</td>
<td>Rm ≤ X</td>
<td>Rm ← ~Rm</td>
<td>1,2</td>
</tr>
</tbody>
</table>

**Notes:****

- The table above outlines various operations and their corresponding descriptions, flags, and cycle counts for different operations in a microcontroller or similar device.
- Each operation is specified with its mnemonics (e.g., MOV, ADD), operands, description, and flags it affects, along with the number of cycles it takes.
- These operations are fundamental to the instruction sets of many microcontrollers, allowing for basic data manipulation and control flow within the device.
### Ordering Information

<table>
<thead>
<tr>
<th>Speed (MHz)</th>
<th>Power Supply</th>
<th>Ordering Code</th>
<th>Package</th>
<th>Operation Range</th>
</tr>
</thead>
<tbody>
<tr>
<td>6</td>
<td>2.7 - 5.5V</td>
<td>ATmega16L-0AC</td>
<td>44A</td>
<td>Commercial (0°C to 70°C)</td>
</tr>
<tr>
<td></td>
<td></td>
<td>ATmega16L-0PC</td>
<td>40P9</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>ATmega16L-0MC</td>
<td>44M1</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>ATmega16L-SAI</td>
<td>44A</td>
<td>Industrial (-40°C to 85°C)</td>
</tr>
<tr>
<td></td>
<td></td>
<td>ATmega16L-SPI</td>
<td>43P9</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>ATmega16L-SMI</td>
<td>44M1</td>
<td></td>
</tr>
<tr>
<td>13</td>
<td>4.5 - 5.5V</td>
<td>ATmega16L-16AC</td>
<td>44A</td>
<td>Commercial (0°C to 70°C)</td>
</tr>
<tr>
<td></td>
<td></td>
<td>ATmega16L-10PC</td>
<td>40P9</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>ATmega16L-10M9C</td>
<td>44M1</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>ATmega16L-16AI</td>
<td>44A</td>
<td>Industrial (-40°C to 85°C)</td>
</tr>
<tr>
<td></td>
<td></td>
<td>ATmega16L-16SPI</td>
<td>43P9</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>ATmega16L-16SMI</td>
<td>44M1</td>
<td></td>
</tr>
</tbody>
</table>

### Packaging Information

44A

---

**DIMENSIONS**

(Unit of Measure = mm)

<table>
<thead>
<tr>
<th>SYMBOL</th>
<th>MIN</th>
<th>NOM</th>
<th>MAX</th>
<th>NOTES</th>
</tr>
</thead>
<tbody>
<tr>
<td>A</td>
<td></td>
<td></td>
<td></td>
<td>NOM 1.00</td>
</tr>
<tr>
<td>A1</td>
<td>0.05</td>
<td>0.15</td>
<td></td>
<td></td>
</tr>
<tr>
<td>A2</td>
<td>0.15</td>
<td>0.30</td>
<td>0.35</td>
<td></td>
</tr>
<tr>
<td>D</td>
<td>11.75</td>
<td>12.00</td>
<td>12.25</td>
<td></td>
</tr>
<tr>
<td>D1</td>
<td>9.95</td>
<td>10.00</td>
<td>10.10</td>
<td>NOTE 2</td>
</tr>
<tr>
<td>E</td>
<td>11.75</td>
<td>12.00</td>
<td>12.25</td>
<td></td>
</tr>
<tr>
<td>E1</td>
<td>9.95</td>
<td>10.00</td>
<td>10.10</td>
<td>NOTE 2</td>
</tr>
<tr>
<td>F</td>
<td>3.00</td>
<td>3.00</td>
<td>3.00</td>
<td></td>
</tr>
<tr>
<td>G</td>
<td>0.05</td>
<td>0.20</td>
<td>0.25</td>
<td></td>
</tr>
<tr>
<td>I</td>
<td>0.16</td>
<td>0.16</td>
<td>0.16</td>
<td></td>
</tr>
</tbody>
</table>

NOTES:
1. This package conforms to JEDEC standard MS-022D and JEDEC flexible.
2. Dimensions D1 and D1 are intended for mask protection. Absolute maximum power dissipation is 3.24 W per side. Dimensions D1 and D1 are minimum plastic body size dimensions including mold flash.
3. Lead coplanarity is ±0.10 mm maximum.
LAMPIRAN D
DATASHEET GLCD
1. Basic Specifications

1.1 Display Specifications
1) LCD Display Mode : STN, Negative, Transmissive
2) Display Color : Display Data = “1” : Light Gray (*1)
                  : Display Data = “0” : Deep Blue (*2)
3) Viewing Angle : 6 H
4) Driving Method : 1/64 duty, 1/ 9 bias
5) Back Light : White LED backlight

Note:
*1. Color tone may slightly change by Temperature and Driving Condition.
*2. The Color is defined as the inactive / background color

1.2 Mechanical Specifications
1) Outline Dimension : 93.0 x 70.0 x 13.8MAX
(see attached Outline Drawing for details)

1.3 Block Diagram
1.4 Terminal Functions

<table>
<thead>
<tr>
<th>Pin No.</th>
<th>Pin Name</th>
<th>I/O</th>
<th>Descriptions</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>VSS</td>
<td>Power</td>
<td>Negative Power Supply, Ground (0V)</td>
</tr>
<tr>
<td>2</td>
<td>VDD</td>
<td>Power</td>
<td>Positive Power Supply</td>
</tr>
<tr>
<td>3</td>
<td>V0</td>
<td>Power</td>
<td>LCD Contrast reference</td>
</tr>
<tr>
<td>4</td>
<td>RS</td>
<td>Input</td>
<td>RS = H; DB0 – DB7 = Display RAM data</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>RS = L; DB0 – DB7 = Instruction data</td>
</tr>
<tr>
<td>5</td>
<td>RAW</td>
<td>Input</td>
<td>In read mode</td>
</tr>
<tr>
<td>6</td>
<td>E</td>
<td>Input</td>
<td>RAW = H; Data read from the LCD module, data appears at DB0 – DB7 and can be</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>read by the host while, E = H and the device is being selected</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>In write mode</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>RAW = L; Data write to the LCD module, data appears at DB0 – DB7 will be</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>written into the LCD module at E = H→L and device is being selected</td>
</tr>
<tr>
<td>7</td>
<td>DB0</td>
<td>I/O</td>
<td>Data bus;</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>Three state I/O terminal for display data or instruction data</td>
</tr>
<tr>
<td>14</td>
<td>DB7</td>
<td>I/O</td>
<td></td>
</tr>
<tr>
<td>15</td>
<td>CS1</td>
<td>Input</td>
<td>Chip selection,</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>When CS1 = 1 (*1) enable access to the Left Side (64 column) of the LCD</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>module</td>
</tr>
<tr>
<td>16</td>
<td>CS2</td>
<td>Input</td>
<td>Chip selection,</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>When CS2 = 1 (*1) enable access to the Right Side (64 column) of the LCD</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>module</td>
</tr>
<tr>
<td>17</td>
<td>RST</td>
<td>Input</td>
<td>Reset signal</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>RST = L, Display off</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>display start line register becomes 0</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>no command or instruction data could be accepted</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>RST = H, Normal running</td>
</tr>
<tr>
<td>18</td>
<td>VOUT</td>
<td>Output</td>
<td>Power Booster output for V0</td>
</tr>
<tr>
<td>19</td>
<td>BLA</td>
<td>Power</td>
<td>Positive Power for LED backlight</td>
</tr>
<tr>
<td>20</td>
<td>BLK</td>
<td>Power</td>
<td>Negative Power for LED backlight</td>
</tr>
</tbody>
</table>

Note:

*1. Display or instruction data could write into the LCD module’s driver/controllers individually or at the same time.

Only read display or instruction data from one of the driver/controllers in the LCD module at a time, otherwise unexpected data collision may occur.
2. Absolute Maximum Ratings

<table>
<thead>
<tr>
<th>Items</th>
<th>Symbol</th>
<th>Min.</th>
<th>Max.</th>
<th>Unit</th>
<th>Condition</th>
</tr>
</thead>
<tbody>
<tr>
<td>Supply Voltage</td>
<td>$V_{DD}$</td>
<td>0</td>
<td>7.0</td>
<td>V</td>
<td>$V_{DD} = 0$V</td>
</tr>
<tr>
<td>Operating Temperature</td>
<td>$T_{CP}$</td>
<td>-20</td>
<td>70</td>
<td>°C</td>
<td>No Condensation</td>
</tr>
<tr>
<td>Storage Temperature</td>
<td>$T_{ST}$</td>
<td>-30</td>
<td>80</td>
<td>°C</td>
<td>No Condensation</td>
</tr>
</tbody>
</table>

Cautions:
Any stresses exceeding the Absolute Maximum Ratings may cause substantial damage to the device. Functional operation of this device at other conditions beyond those listed in the specification is not implied and prolonged exposure to extreme conditions may affect device reliability.

3. Electrical Characteristics

3.1 DC Characteristics

$V_{CC} = 0$V, $V_{PP} = 5$V, $T_{OP} = 25$°C

<table>
<thead>
<tr>
<th>Items</th>
<th>Symbol</th>
<th>MIN.</th>
<th>TYP.</th>
<th>MAX.</th>
<th>Unit</th>
<th>Applicable Pin</th>
</tr>
</thead>
<tbody>
<tr>
<td>Operating Voltage</td>
<td>$V_{DD}$</td>
<td>4.8</td>
<td>5.0</td>
<td>5.2</td>
<td>V</td>
<td>VDD</td>
</tr>
<tr>
<td>Input High Voltage</td>
<td>$V_{II}$</td>
<td>3.5</td>
<td>5</td>
<td>$V_{DD}$</td>
<td>V</td>
<td>RS, RW, E, CS1, CS2</td>
</tr>
<tr>
<td>Input Low Voltage</td>
<td>$V_{IL}$</td>
<td>0</td>
<td>-</td>
<td>0.4</td>
<td>V</td>
<td>DB0-DB7</td>
</tr>
<tr>
<td>Operating Current</td>
<td>$I_{DD}$</td>
<td>-</td>
<td>6.5</td>
<td>15</td>
<td>mA</td>
<td>VDD, VSS</td>
</tr>
</tbody>
</table>

3.2 LED Backlight Circuit Characteristics

$V_{CC} = 0$V, $I_{LED} = 50$mA, $T_{OP} = 25$°C

<table>
<thead>
<tr>
<th>Items</th>
<th>Symbol</th>
<th>MIN.</th>
<th>TYP.</th>
<th>MAX.</th>
<th>Unit</th>
<th>Applicable Pin</th>
</tr>
</thead>
<tbody>
<tr>
<td>Forward Voltage</td>
<td>$V_{F/LA}$</td>
<td>-</td>
<td>4.3</td>
<td>-</td>
<td>V</td>
<td>BLA, BLK</td>
</tr>
<tr>
<td>Forward Current</td>
<td>$I_{F/LA}$</td>
<td>-</td>
<td>-</td>
<td>120</td>
<td>mA</td>
<td>BLA, BLK</td>
</tr>
</tbody>
</table>

![LED Backlight Circuit Diagram]
### 3.3 AC Characteristics

![Diagram of Host Write Timing](image)

![Diagram of Host Read Timing](image)

<table>
<thead>
<tr>
<th>Item</th>
<th>Symbol</th>
<th>MIN</th>
<th>TYP</th>
<th>MAX</th>
<th>Unit</th>
</tr>
</thead>
<tbody>
<tr>
<td>E cycle time</td>
<td>tc</td>
<td>1500</td>
<td>-</td>
<td>-</td>
<td>ns</td>
</tr>
<tr>
<td>E high level width</td>
<td>twh</td>
<td>700</td>
<td>-</td>
<td>-</td>
<td>ns</td>
</tr>
<tr>
<td>E low level width</td>
<td>twl</td>
<td>700</td>
<td>-</td>
<td>-</td>
<td>ns</td>
</tr>
<tr>
<td>E rise time</td>
<td>tr</td>
<td>-</td>
<td>-</td>
<td>18</td>
<td>ns</td>
</tr>
<tr>
<td>E fall time</td>
<td>tf</td>
<td>-</td>
<td>-</td>
<td>18</td>
<td>ns</td>
</tr>
<tr>
<td>Address set-up time</td>
<td>tasu</td>
<td>210</td>
<td>-</td>
<td>-</td>
<td>ns</td>
</tr>
<tr>
<td>Address hold time</td>
<td>tah</td>
<td>15</td>
<td>-</td>
<td>-</td>
<td>ns</td>
</tr>
<tr>
<td>Data set-up time</td>
<td>tsdu</td>
<td>300</td>
<td>-</td>
<td>-</td>
<td>ns</td>
</tr>
<tr>
<td>Data delay time</td>
<td>td</td>
<td>-</td>
<td>-</td>
<td>480</td>
<td>ns</td>
</tr>
<tr>
<td>Data hold time (write)</td>
<td>tdhw</td>
<td>15</td>
<td>-</td>
<td>-</td>
<td>ns</td>
</tr>
<tr>
<td>Data hold time (read)</td>
<td>tdhr</td>
<td>30</td>
<td>-</td>
<td>-</td>
<td>ns</td>
</tr>
</tbody>
</table>
4. Function Specifications

4.1 Basic Setting
To drive the LCD module correctly and provide normally display, please use the following setting:

- Display start line (Z address) = 0
- LCD Display = on

Note:
These setting commands should issue to both controllers while start up.
See the Display Control Instructions section for details.

4.2 Adjusting the LCD display contrast
A Variable-Resistor must be connected to the LCD module for providing a reference to V0.
Adjusting the VR will result in the change of LCD display contrast.
The recommended value of VR is 25K to 50K

4.3 Resetting the LCD module
The LCD module should be initialized by setting /RST terminal at low level when turning the power on.

When /RST pull low, the LCD module will:
- Display off
- Display start line register becomes 0. (Z-address=0)

While /RST is low, no instruction can be accepted except status read. Therefore, execute other instructions after making sure that D6=0 (clear /RST) and D67=0 (ready) by status read instruction. The conditions of power supply at initial power up are as follows:

<table>
<thead>
<tr>
<th>Item</th>
<th>Symbol</th>
<th>MIN.</th>
<th>TYP.</th>
<th>MAX.</th>
<th>Unit</th>
</tr>
</thead>
<tbody>
<tr>
<td>Reset time</td>
<td>trs</td>
<td>2.0</td>
<td>-</td>
<td>-</td>
<td>us</td>
</tr>
<tr>
<td>Rise time</td>
<td>tr</td>
<td>-</td>
<td>-</td>
<td>150</td>
<td>ns</td>
</tr>
</tbody>
</table>
4.4 Display Memory Map

<table>
<thead>
<tr>
<th>Page (X) Address</th>
<th>data</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td></td>
</tr>
<tr>
<td>1</td>
<td></td>
</tr>
<tr>
<td>2</td>
<td></td>
</tr>
<tr>
<td>3</td>
<td></td>
</tr>
<tr>
<td>4</td>
<td></td>
</tr>
<tr>
<td>5</td>
<td></td>
</tr>
<tr>
<td>6</td>
<td></td>
</tr>
<tr>
<td>7</td>
<td></td>
</tr>
</tbody>
</table>

Column (Y) Address: 
00h -> 3Fh 00h -> 3Fh

Chip Select: 
CS1=1, CS2=0  CS1=0, CS2=1

Note:
1) Display start line (Z address) = 0
2) The Display Data store separately in two drivers.
3) The Display Data for the left section could be accessed by CS1=1.
   The Display Data for the right section could be accessed by CS2=1.

4.5 Internal Registers

There are three registers in each section of LCD module. Each of them could be controlled independently.

Page (X) Address Register
X Address register designates pages of the internal display data RAM. Count function is not available. The address should set by instruction.

Column (Y) Address Counter
Y address counter designates address of the internal display data RAM. It could be set by instruction and is increased by 1 automatically by read or write display data operations.

Display Start Line (Z) Register
Z address register indicates display data RAM to LCD top line. It may be used for scrolling the display pattern on the LCD.
<table>
<thead>
<tr>
<th>Instruction</th>
<th>D</th>
<th>E</th>
<th>F</th>
<th>G</th>
<th>H</th>
<th>I</th>
<th>J</th>
<th>K</th>
</tr>
</thead>
<tbody>
<tr>
<td>Display data</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Select register</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>Address</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Start display</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Start column</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Start display shift</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Disassemble</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>Unlock display data</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>V1 display data</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Unlock display data</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Note: Disassemble 1 means that the result of the operation.

3. Instruction can be written both columns at the same time.

4. See the details of the display control instructions, please refer to Appendix 5B Display instructions.
5. Design and Handling Precaution

1. The LCD panel is made by glass. Any mechanical shock (e.g., dropping from a high place) will damage the LCD module.
2. Do not add excessive force on the surface of the display, which may cause the Display color change abnormally.
3. The polarizer on the LCD is easily get scratched. If possible, do not remove the LCD protective film until the last step of installation.
4. Never attempt to disassemble or rework the LCD module.
5. Only clean the LCD with Isopropyl Alcohol or Ethyl Alcohol. Other solvents (e.g., water) may damage the LCD.
6. When mounting the LCD module, make sure that it is free from twisting, warping and distortion.
7. Ensure to provide enough space (with cushion) between case and LCD panel to prevent external force adding on it, or it may cause damage to the LCD or degrade the display result.
8. Only hold the LCD module by its side. Never hold LCD module by add force on the heat seal or TAB.
9. Never add force to component of the LCD module. It may cause invisible damage or degrade of the reliability.
10. LCD module could be easily damaged by static electricity. Be careful to maintain an optimum anti-static work environment to protect the LCD module.
11. When peeling off the protective film from LCD, static charge may cause abnormal display pattern. It is normal and will resume to normal in a short while.
12. Take care and prevent get hurt by the LCD panel sharp edge.
13. Never operate the LCD module exceed the absolute maximum ratings.
14. Keep the signal line as short as possible to prevent noisy signal applying to LCD module.
15. Never apply signal to the LCD module without power supply.
16. IC chip (e.g., TAB or COG) is sensitive to the light. Strong lighting environment could possibly cause malfunction. Light sealing structure casing is recommend.
17. LCD module reliability may be reduced by temperature shock.
18. When storing the LCD module, avoid exposure to the direct sunlight, high humidity, high temperature or low temperature. They may damage or degrade the LCD module.