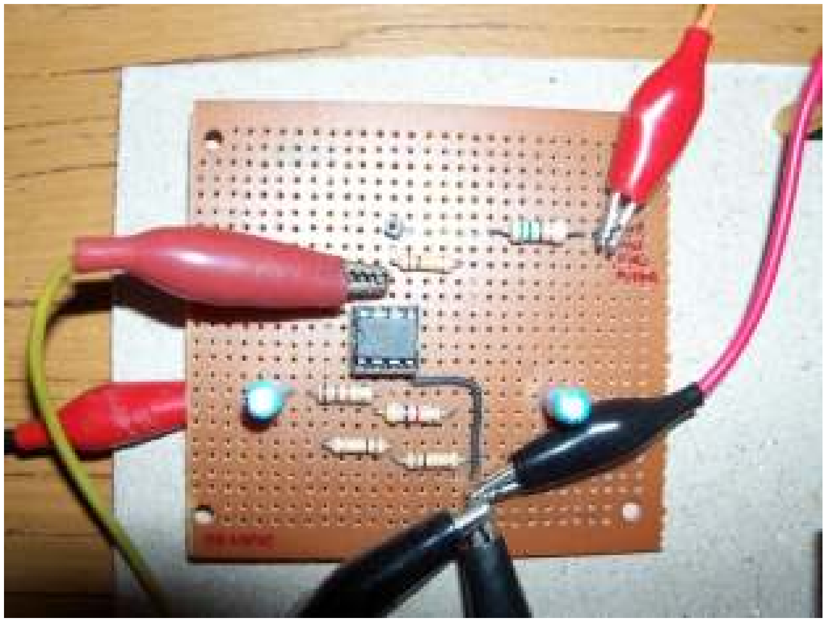
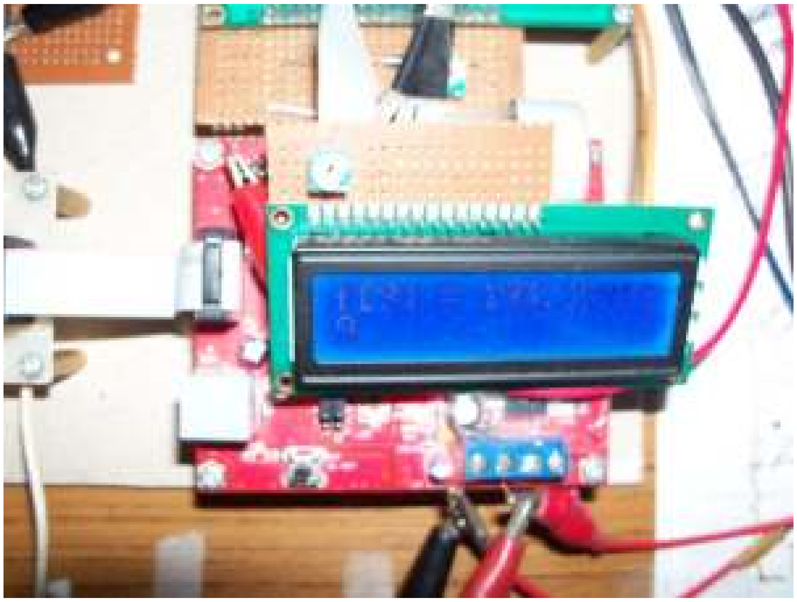
LAMPIRAN A

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

```
/*****************************************************
This program was produced by the
CodeWizardAVR V1.25.3 Standard
Automatic Program Generator
© Copyright 1998-2007 Pavel Haiduc, HP InfoTech s.r.l.
http://www.hpinfotech.com

Project :
Version :
Date    : 8/28/2010
Author  : F4CG
Company : F4CG
Comments:


Chip type           : ATmega16
Program type        : Application
Clock frequency     : 4.000000 MHz
Memory model        : Small
External SRAM size  : 0
Data Stack size     : 512
*****************************************************/

#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

// Alphanumeric LCD Module functions
#asm
   .equ __lcd_port=0x18 ;PORTB
#endasm
#include <lcd.h>

#define ADC_VREF_TYPE 0x60

// 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]=

{0x00,0x00,0x00,0x00,0x00,0x00,0x80,0Xff};

flash unsigned char gambar21[8]=

{0x00,0x00,0x00,0x00,0x00,0x00,0Xc0,0Xff};

flash unsigned char gambar22[8]=

{0x00,0x00,0x00,0x00,0x00,0x00,0Xe0,0Xff};

flash unsigned char gambar23[8]=

{0x00,0x00,0x00,0x00,0x00,0x00,0Xf0,0Xff};

flash unsigned char gambar24[8]=

{0x00,0x00,0x00,0x00,0x00,0x00,0Xf8,0Xff};

flash unsigned char gambar25[8]=

{0x00,0x00,0x00,0x00,0x00,0x00,0Xfc,0Xff};

flash unsigned char gambar26[8]=

{0x00,0x00,0x00,0x00,0x00,0x00,0Xfe,0Xff};

flash unsigned char gambar27[8]=

{0x00,0x00,0x00,0x00,0x00,0x00,0Xff,0Xff};

 flash unsigned char gambar30[8]=

{0x00,0x00,0x00,0x00,0x00,0x80,0Xff,0Xff};

flash unsigned char gambar31[8]=

{0x00,0x00,0x00,0x00,0x00,0Xc0,0Xff,0Xff};

flash unsigned char gambar32[8]=

{0x00,0x00,0x00,0x00,0x00,0Xe0,0Xff,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]=
{0x00,0x00,0x00,0x00,0x00,0Xfc,0Xff,0Xff};
flash unsigned char gambar36[8]=
{0x00,0x00,0x00,0x00,0x00,0Xfe,0Xff,0Xff};
flash unsigned char gambar37[8]=
{0x00,0x00,0x00,0x00,0x00,0Xff,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]=
{0x00,0x00,0x00,0x00,0Xf8,0Xff,0Xff,0Xff};
flash unsigned char gambar45[8]=
{0x00,0x00,0x00,0x00,0Xfc,0Xff,0Xff,0Xff};
flash unsigned char gambar46[8]=
{0x00,0x00,0x00,0x00,0Xfe,0Xff,0Xff,0Xff};
flash unsigned char gambar47[8]=
{0x00,0x00,0x00,0x00,0Xff,0Xff,0Xff,0Xff};
flash unsigned char gambar50[8]=
{0x00,0x00,0x00,0x80,0Xff,0Xff,0Xff,0Xff};
flash unsigned char gambar51[8]=
{0x00,0x00,0x00,0Xc0,0Xff,0Xff,0Xff,0Xff};
flash unsigned char gambar52[8]=
{0x00,0x00,0x00,0Xe0,0Xff,0Xff,0Xff,0Xff};
flash unsigned char gambar53[8]=
{0x00,0x00,0x00,0Xf0,0Xff,0Xff,0Xff,0Xff};
flash unsigned char gambar54[8]=
{0x00,0x00,0x00,0Xf8,0Xff,0Xff,0Xff,0Xff};
flash unsigned char gambar55[8]=
{0x00,0x00,0x00,0Xfc,0Xff,0Xff,0Xff,0Xff};
flash unsigned char gambar56[8]=
{0x00,0x00,0x00,0Xfe,0Xff,0Xff,0Xff,0Xff};
flash unsigned char gambar57[8]=
{0x00,0x00,0x00,0Xff,0Xff,0Xff,0Xff,0Xff};
flash unsigned char gambar60[8]=
{0x00,0x00,0x80,0Xff,0Xff,0Xff,0Xff,0Xff};
flash unsigned char gambar61[8]=
{0x00,0x00,0Xc0,0Xff,0Xff,0Xff,0Xff,0Xff};
flash unsigned char gambar62[8]=
{0x00,0x00,0Xe0,0Xff,0Xff,0Xff,0Xff,0Xff};
flash unsigned char gambar63[8]=
{0x00,0x00,0Xf0,0Xff,0Xff,0Xff,0xFF,0xFF};
```

```
flash unsigned char gambar64[8]=
{0x00,0x00,0xF8,0xFF,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)
{
 e=1;
 rs=0;
 rw=0;
 data=0x3E|a;
 delay_us(1);
 e=0;
 delay_us(1);

}
void yadd(unsigned char a)
{
 e=1;
 rs=0;
 rw=0;
 data=0x40|a;
 delay_us(1);
 e=0;
delay_us(1);
}
void xadd(unsigned char a)
{
 e=1;
 rs=0;
 rw=0;
 data=0xB8|a;
 delay_us(1);
 e=0;
 delay_us(1);
}
void zadd(unsigned char a)
{
 e=1;
 rs=0;

rw=0;
 data=0xC0|a;
 delay_us(1);
 e=0;
 delay_us(1);
}
void write(unsigned char a)
```

```
{
 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 B 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
PORTB=0x00;
DDRB=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 1 initialization
// Clock source: System Clock
// Clock value: Timer 1 Stopped
// Mode: Normal top=FFFFh
// 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;
TCCR1B=0x00;
TCNT1H=0x00;
TCNT1L=0x00;

ICR1H=0x00;
ICR1L=0x00;
OCR1AH=0x00;
OCR1AL=0x00;
OCR1BH=0x00;
OCR1BL=0x00;

// Timer/Counter 2 initialization
// Clock source: System Clock
// Clock value: Timer 2 Stopped
// Mode: Normal top=FFh
// OC2 output: Disconnected
ASSR=0x00;
TCCR2=0x00;
TCNT2=0x00;
OCR2=0x00;

// External Interrupt(s) initialization
// INT0: Off
// INT1: Off
// INT2: Off
MCUCR=0x00;
MCUCSR=0x00;
// Timer(s)/Counter(s) Interrupt(s) initialization
TIMSK=0x00;
// Analog Comparator initialization
// Analog Comparator: Off
// Analog Comparator Input Capture by
Timer/Counter 1: Off
ACSR=0x80;
SFIOR=0x00;
```

```
// ADC initialization
// ADC Clock frequency: 1000.000 kHz
// ADC Voltage Reference: AVCC pin
// Only the 8 most significant bits of
// the AD conversion result are used
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);
         zadd(0);
         write(0x00);
         delay_us(1);
      }
   }
   cs1 = 1;
   cs2 = 0;
   for (y = 0; y < 128; y++) {
      adc[y] = read_adc(6);
      delay_us(120);
   }
   for (y = 0; y < 128; y++) {
      rst = 1;
      display(1);
      if(y<64){
      s = 0;
      q = 0;

  for (n = 0; n < 128; n++) {
         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);
        13print(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);
              zadd(0);
              write(gambar10[x]);
              delay_us(1);
           } else if (f[y] > 150 && f[y] <= 175) {
              xadd(x);
              yadd(y);
              zadd(0);
              write(gambar11[x]);
              delay_us(1);
           } else if (f[y] >= 175 && f[y] <= 200) {
              xadd(x);
              yadd(y);
              zadd(0);
              write(gambar12[x]);
              delay_us(1);


            } else if (f[y] >= 200 && f[y] <= 225)
            {
              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);
 } 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(gambar30[x]);
    delay_us(1);
} else if (f[y] >= 550 && f[y] <= 575) {
    xadd(x);
    yadd(y);
    zadd(0);
    write(gambar31[x]);
    delay_us(1);
} else if (f[y] >= 575 && f[y] <= 600) {
    xadd(x);
    yadd(y);
    zadd(0);
    write(gambar32[x]);
    delay_us(1);
} else if (f[y] >= 600 && f[y] <= 625) {
    xadd(x);
    yadd(y);
    zadd(0);
    write(gambar33[x]);
    delay_us(1);
} else if (f[y] >= 625 && f[y] <= 650) {
    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);
} 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)
{
        xadd(x);
        yadd(y);
        zadd(0);
        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);
      } else if (f[y] >= 1250 && f[y] <= 1275)
{
        xadd(x);
        yadd(y);
```

```
            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(gambar80[x]);
delay_us(1);
} else if (f[y] >= 1550 && f[y] <= 1575) {
xadd(x);
yadd(y);
zadd(0);
write(gambar81[x]);
delay_us(1);
} else if (f[y] >= 1575 && f[y] <= 1600)
{
xadd(x);
yadd(y);
zadd(0);
write(gambar82[x]);
delay_us(1);
} else if (f[y] >= 1600 && f[y] <= 1625)
{
xadd(x);
yadd(y);
zadd(0);
write(gambar83[x]);
delay_us(1);

} else if (f[y] >= 1625 && f[y] <= 1650) {
xadd(x);
yadd(y);
zadd(0);
write(gambar84[x]);
```

```
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);
};
}
```

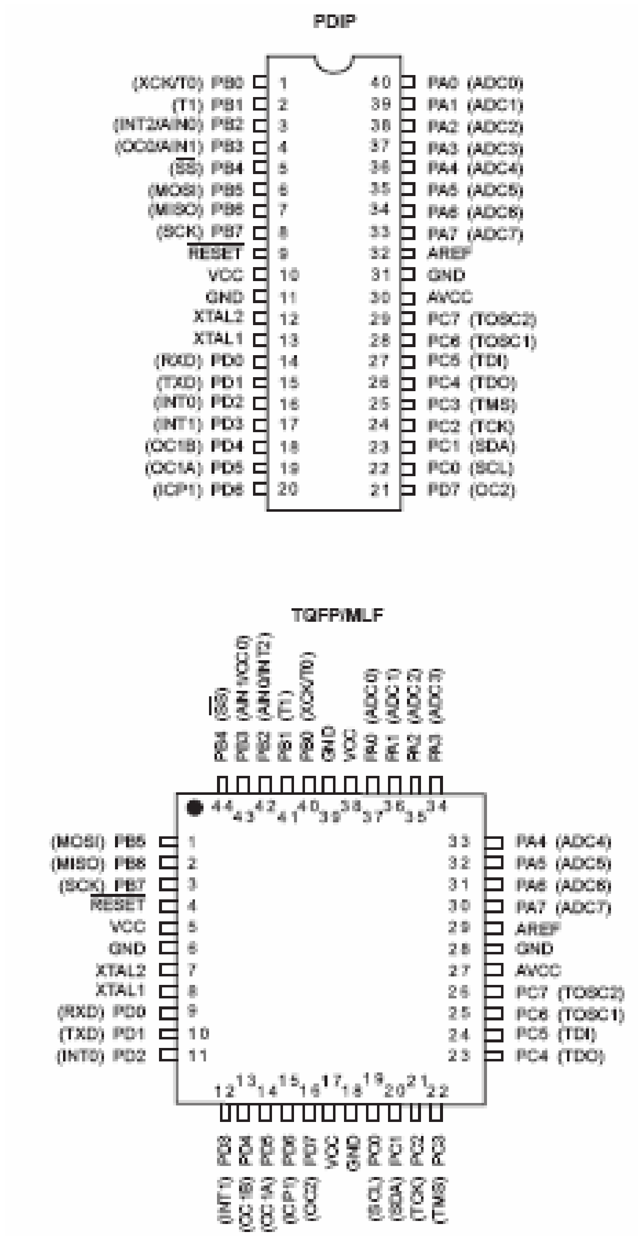
LAMPIRAN C

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
Endurance: 10,000 Write/Erase Cycles
– Optional Boot Code Section with Independent Lock Bits
In-System Programming by On-chip Boot Program
True Read-While-Write Operation
– 512 Bytes EEPROM
Endurance: 100,000 Write/Erase Cycles
– 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

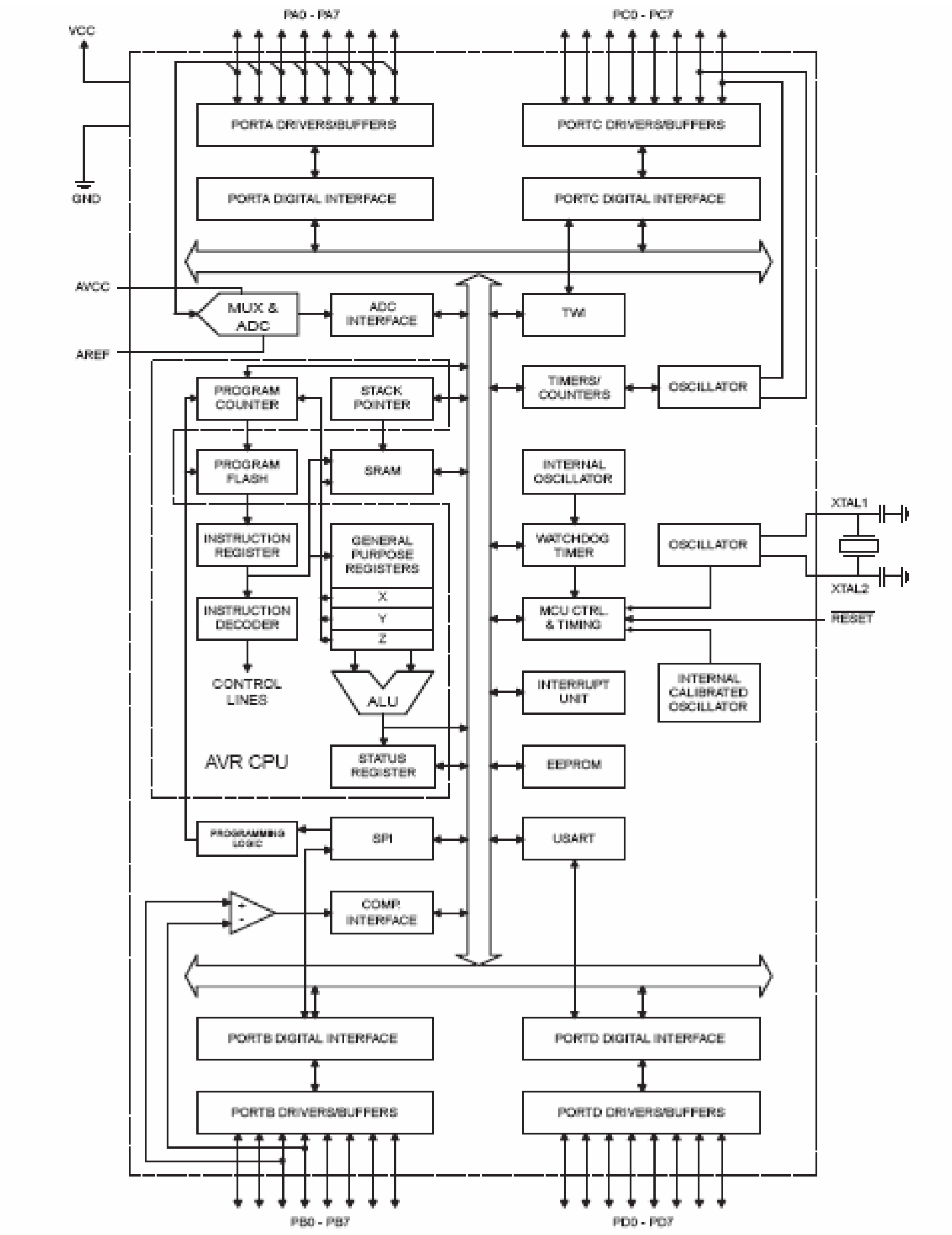
Pin Configurations **Figure 1.** Pinouts ATmega16



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 V_{CC}, even if the ADC is not used. If the ADC is used, it should be connected to V_{CC} through a low-pass filter.

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

Register Summary

Address	Name	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0	Page
$3F ($5F)	SREG	I	T	H	S	V	N	Z	C	7
$3E ($5E)	SPH	–	–	–	–	–	SP10	SP9	SP8	10
$3D ($5D)	SPL	SP7	SP6	SP5	SP4	SP3	SP2	SP1	SP0	10
$3C ($5C)	OCR0	Timer/Counter0 Output Compare Register								83
$3B ($5B)	GICR	INT1	INT0	INT2	–	–	–	IVSEL	IVCE	46, 67
$3A ($5A)	GIFR	INTF1	INTF0	INTF2	–	–	–	–	–	68
$39 ($59)	TIMSK	OCIE2	TOIE2	TICIE1	OCIE1A	OCIE1B	TOIE1	OCIE0	TOIE0	83, 114, 132
$38 ($58)	TIFR	OCF2	TOV2	ICF1	OCF1A	OCF1B	TOV1	OCF0	TOV0	84, 115, 132
$37 ($57)	SPMCR	SPMIE	RWWSB	–	RWWSRE	BLBSET	PGWRT	PGERS	SPMEN	249
$36 ($56)	TWCR	TWINT	TWEA	TWSTA	TWSTO	TWWC	TWEN	–	TWIE	178
$35 ($55)	MCUCR	SM2	SE	SM1	SM0	ISC11	ISC10	ISC01	ISC00	30, 66
$34 ($54)	MCUCSR	JTD	ISC2	–	JTRF	WDRF	BORF	EXTRF	PORF	39, 67, 229
$33 ($53)	TCCR0	FOC0	WGM00	COM01	COM00	WGM01	CS02	CS01	CS00	81
$32 ($52)	TCNT0	Timer/Counter0 (8 Bits)								83
$31[1] ($51)[1]	OSCCAL	Oscillator Calibration Register								28
	OCDR	On-Chip Debug Register								225
$30 ($50)	SFIOR	ADTS2	ADTS1	ADTS0	–	ACME	PUD	PSR2	PSR10	55,86,133,199,219
$2F ($4F)	TCCR1A	COM1A1	COM1A0	COM1B1	COM1B0	FOC1A	FOC1B	WGM11	WGM10	109
$2E ($4E)	TCCR1B	ICNC1	ICES1	–	WGM13	WGM12	CS12	CS11	CS10	112
$2D ($4D)	TCNT1H	Timer/Counter1 – Counter Register High Byte								113
$2C ($4C)	TCNT1L	Timer/Counter1 – Counter Register Low Byte								113
$2B ($4B)	OCR1AH	Timer/Counter1 – Output Compare Register A High Byte								113
$2A ($4A)	OCR1AL	Timer/Counter1 – Output Compare Register A Low Byte								113
$29 ($49)	OCR1BH	Timer/Counter1 – Output Compare Register B High Byte								113
$28 ($48)	OCR1BL	Timer/Counter1 – Output Compare Register B Low Byte								113
$27 ($47)	ICR1H	Timer/Counter1 – Input Capture Register High Byte								114
$26 ($46)	ICR1L	Timer/Counter1 – Input Capture Register Low Byte								114
$25 ($45)	TCCR2	FOC2	WGM20	COM21	COM20	WGM21	CS22	CS21	CS20	127
$24 ($44)	TCNT2	Timer/Counter2 (8 Bits)								129
$23 ($43)	OCR2	Timer/Counter2 Output Compare Register								129
$22 ($42)	ASSR	–	–	–	–	AS2	TCN2UB	OCR2UB	TCR2UB	130
$21 ($41)	WDTCR	–	–	–	WDTOE	WDE	WDP2	WDP1	WDP0	41
$20[2] ($40)[2]	UBRRH	URSEL	–	–	–	UBRR[11:8]				165
	UCSRC	URSEL	UMSEL	UPM1	UPM0	USBS	UCSZ1	UCSZ0	UCPOL	164
$1F ($3F)	EEARH	–	–	–	–	–	–	–	EEAR8	17
$1E ($3E)	EEARL	EEPROM Address Register Low Byte								17
$1D ($3D)	EEDR	EEPROM Data Register								17
$1C ($3C)	EECR	–	–	–	–	EERIE	EEMWE	EEWE	EERE	17
$1B ($3B)	PORTA	PORTA7	PORTA6	PORTA5	PORTA4	PORTA3	PORTA2	PORTA1	PORTA0	64
$1A ($3A)	DDRA	DDA7	DDA6	DDA5	DDA4	DDA3	DDA2	DDA1	DDA0	64
$19 ($39)	PINA	PINA7	PINA6	PINA5	PINA4	PINA3	PINA2	PINA1	PINA0	64
$18 ($38)	PORTB	PORTB7	PORTB6	PORTB5	PORTB4	PORTB3	PORTB2	PORTB1	PORTB0	64
$17 ($37)	DDRB	DDB7	DDB6	DDB5	DDB4	DDB3	DDB2	DDB1	DDB0	64
$16 ($36)	PINB	PINB7	PINB6	PINB5	PINB4	PINB3	PINB2	PINB1	PINB0	64
$15 ($35)	PORTC	PORTC7	PORTC6	PORTC5	PORTC4	PORTC3	PORTC2	PORTC1	PORTC0	65
$14 ($34)	DDRC	DDC7	DDC6	DDC5	DDC4	DDC3	DDC2	DDC1	DDC0	65
$13 ($33)	PINC	PINC7	PINC6	PINC5	PINC4	PINC3	PINC2	PINC1	PINC0	65
$12 ($32)	PORTD	PORTD7	PORTD6	PORTD5	PORTD4	PORTD3	PORTD2	PORTD1	PORTD0	65
$11 ($31)	DDRD	DDD7	DDD6	DDD5	DDD4	DDD3	DDD2	DDD1	DDD0	65
$10 ($30)	PIND	PIND7	PIND6	PIND5	PIND4	PIND3	PIND2	PIND1	PIND0	65

$0F ($2F)	SPDR	SPI Data Register								140
$0E ($2E)	SPSR	SPIF	WCOL	–	–	–	–	–	SPI2X	140
$0D ($2D)	SPCR	SPIE	SPE	DORD	MSTR	CPOL	CPHA	SPR1	SPR0	138
$0C ($2C)	UDR	USART I/O Data Register								161
$0B ($2B)	UCSRA	RXC	TXC	UDRE	FE	DOR	PE	U2X	MPCM	162
$0A ($2A)	UCSRB	RXCIE	TXCIE	UDRIE	RXEN	TXEN	UCSZ2	RXB8	TXB8	163
$09 ($29)	UBRRL	USART Baud Rate Register Low Byte								165
$08 ($28)	ACSR	ACD	ACBG	ACO	ACI	ACIE	ACIC	ACIS1	ACIS0	200
$07 ($27)	ADMUX	REFS1	REFS0	ADLAR	MUX4	MUX3	MUX2	MUX1	MUX0	216
$06 ($26)	ADCSRA	ADEN	ADSC	ADATE	ADIF	ADIE	ADPS2	ADPS1	ADPS0	217
$05 ($25)	ADCH	ADC Data Register High Byte								218
$04 ($24)	ADCL	ADC Data Register Low Byte								218
$03 ($23)	TWDR	Two-wire Serial Interface Data Register								180
$02 ($22)	TWAR	TWA6	TWA5	TWA4	TWA3	TWA2	TWA1	TWA0	TWGCE	180

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.

BRMI	k	Branch if Minus	if (N = 1) then PC ← PC + k + 1	None	1 / 2
BRPL	k	Branch if Plus	if (N = 0) then PC ← PC + k + 1	None	1 / 2
BRGE	k	Branch if Greater or Equal, Signed	if (N ⊕ V= 0) then PC ← PC + k + 1	None	1 / 2
BRLT	k	Branch if Less Than Zero, Signed	if (N ⊕ V= 1) then PC ← PC + k + 1	None	1 / 2
BRHS	k	Branch if Half Carry Flag Set	if (H = 1) then PC ← PC + k + 1	None	1 / 2
BRHC	k	Branch if Half Carry Flag Cleared	if (H = 0) then PC ← PC + k + 1	None	1 / 2
BRTS	k	Branch if T Flag Set	if (T = 1) then PC ← PC + k + 1	None	1 / 2
BRTC	k	Branch if T Flag Cleared	if (T = 0) then PC ← PC + k + 1	None	1 / 2
BRVS	k	Branch if Overflow Flag is Set	if (V = 1) then PC ← PC + k + 1	None	1 / 2
BRVC	k	Branch if Overflow Flag is Cleared	if (V = 0) then PC ← PC + k + 1	None	1 / 2

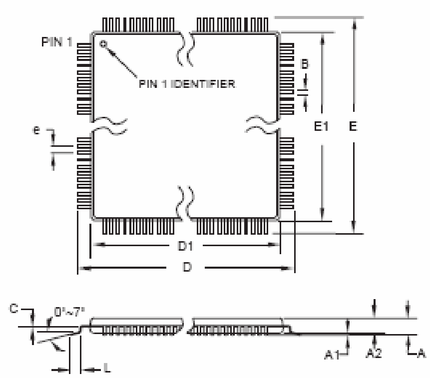
Mnemonics	Operands	Description	Operation	Flags	#Clocks
BRIE	k	Branch if Interrupt Enabled	if (I = 1) then PC ← PC + k + 1	None	1 / 2
BRID	k	Branch if Interrupt Disabled	if (I = 0) then PC ← PC + k + 1	None	1 / 2
DATA TRANSFER INSTRUCTIONS					
MOV	Rd, Rr	Move Between Registers	Rd ← Rr	None	1
MOVW	Rd, Rr	Copy Register Word	Rd+1:Rd ← Rr+1:Rr	None	1
LDI	Rd, K	Load Immediate	Rd ← K	None	1
LD	Rd, X	Load Indirect	Rd ← (X)	None	2
LD	Rd, X+	Load Indirect and Post-Inc.	Rd ← (X), X ← X + 1	None	2
LD	Rd, - X	Load Indirect and Pre-Dec.	X ← X - 1, Rd ← (X)	None	2
LD	Rd, Y	Load Indirect	Rd ← (Y)	None	2
LD	Rd, Y+	Load Indirect and Post-Inc.	Rd ← (Y), Y ← Y + 1	None	2
LD	Rd, - Y	Load Indirect and Pre-Dec.	Y ← Y - 1, Rd ← (Y)	None	2
LDD	Rd,Y+q	Load Indirect with Displacement	Rd ← (Y + q)	None	2
LD	Rd, Z	Load Indirect	Rd ← (Z)	None	2
LD	Rd, Z+	Load Indirect and Post-Inc.	Rd ← (Z), Z ← Z+1	None	2
LD	Rd, -Z	Load Indirect and Pre-Dec.	Z ← Z - 1, Rd ← (Z)	None	2
LDD	Rd, Z+q	Load Indirect with Displacement	Rd ← (Z + q)	None	2
LDS	Rd, k	Load Direct from SRAM	Rd ← (k)	None	2
ST	X, Rr	Store Indirect	(X) ← Rr	None	2
ST	X+, Rr	Store Indirect and Post-Inc.	(X) ← Rr, X ← X + 1	None	2
ST	- X, Rr	Store Indirect and Pre-Dec.	X ← X - 1, (X) ← Rr	None	2
ST	Y, Rr	Store Indirect	(Y) ← Rr	None	2
ST	Y+, Rr	Store Indirect and Post-Inc.	(Y) ← Rr, Y ← Y + 1	None	2
ST	- Y, Rr	Store Indirect and Pre-Dec.	Y ← Y - 1, (Y) ← Rr	None	2
STD	Y+q,Rr	Store Indirect with Displacement	(Y + q) ← Rr	None	2
ST	Z, Rr	Store Indirect	(Z) ← Rr	None	2
ST	Z+, Rr	Store Indirect and Post-Inc.	(Z) ← Rr, Z ← Z + 1	None	2
ST	-Z, Rr	Store Indirect and Pre-Dec.	Z ← Z - 1, (Z) ← Rr	None	2
STD	Z+q,Rr	Store Indirect with Displacement	(Z + q) ← Rr	None	2
STS	k, Rr	Store Direct to SRAM	(k) ← Rr	None	2
LPM		Load Program Memory	R0 ← (Z)	None	3
LPM	Rd, Z	Load Program Memory	Rd ← (Z)	None	3
LPM	Rd, Z+	Load Program Memory and Post-Inc	Rd ← (Z), Z ← Z+1	None	3
SPM		Store Program Memory	(Z) ← R1:R0	None	-
IN	Rd, P	In Port	Rd ← P	None	1
OUT	P, Rr	Out Port	P ← Rr	None	1
PUSH	Rr	Push Register on Stack	STACK ← Rr	None	2
POP	Rd	Pop Register from Stack	Rd ← STACK	None	2
BIT AND BIT-TEST INSTRUCTIONS					
SBI	P,b	Set Bit in I/O Register	I/O(P,b) ← 1	None	2
CBI	P,b	Clear Bit in I/O Register	I/O(P,b) ← 0	None	2
LSL	Rd	Logical Shift Left	Rd(n+1) ← Rd(n), Rd(0) ← 0	Z,C,N,V	1
LSR	Rd	Logical Shift Right	Rd(n) ← Rd(n+1), Rd(7) ← 0	Z,C,N,V	1
ROL	Rd	Rotate Left Through Carry	Rd(0)←C,Rd(n+1)← Rd(n),C←Rd(7)	Z,C,N,V	1
ROR	Rd	Rotate Right Through Carry	Rd(7)←C,Rd(n)← Rd(n+1),C←Rd(0)	Z,C,N,V	1
ASR	Rd	Arithmetic Shift Right	Rd(n) ← Rd(n+1), n=0..6	Z,C,N,V	1
SWAP	Rd	Swap Nibbles	Rd(3..0)←Rd(7..4),Rd(7..4)←Rd(3..0)	None	1
BSET	s	Flag Set	SREG(s) ← 1	SREG(s)	1
BCLR	s	Flag Clear	SREG(s) ← 0	SREG(s)	1
BST	Rr, b	Bit Store from Register to T	T ← Rr(b)	T	1
BLD	Rd, b	Bit load from T to Register	Rd(b) ← T	None	1
SEC		Set Carry	C ← 1	C	1
CLC		Clear Carry	C ← 0	C	1
SEN		Set Negative Flag	N ← 1	N	1
CLN		Clear Negative Flag	N ← 0	N	1
SEZ		Set Zero Flag	Z ← 1	Z	1
CLZ		Clear Zero Flag	Z ← 0	Z	1
SEI		Global Interrupt Enable	I ← 1	I	1
CLI		Global Interrupt Disable	I ← 0	I	1
SES		Set Signed Test Flag	S ← 1	S	1
CLS		Clear Signed Test Flag	S ← 0	S	1
SEV		Set Twos Complement Overflow.	V ← 1	V	1
CLV		Clear Twos Complement Overflow	V ← 0	V	1
SET		Set T in SREG	T ← 1	T	1
CLT		Clear T in SREG	T ← 0	T	1
SEH		Set Half Carry Flag in SREG	H ← 1	H	1

Ordering Information

Speed (MHz)	Power Supply	Ordering Code	Package	Operation Range
8	2.7 - 5.5V	ATmega16L-8AC ATmega16L-8PC ATmega16L-8MC	44A 40P6 44M1	Commercial (0°C to 70°C)
		ATmega16L-8AI ATmega16L-8PI ATmega16L-8MI	44A 40P6 44M1	Industrial (-40°C to 85°C)
16	4.5 - 5.5V	ATmega16-16AC ATmega16-16PC ATmega16-16MC	44A 40P6 44M1	Commercial (0°C to 70°C)
		ATmega16-16AI ATmega16-16PI ATmega16-16MI	44A 40P6 44M1	Industrial (-40°C to 85°C)

Packaging Information

44A



COMMON DIMENSIONS
(Unit of Measure = mm)

SYMBOL	MIN	NOM	MAX	NOTE
A	–	–	1.20	
A1	0.05	–	0.15	
A2	0.95	1.00	1.05	
D	11.75	12.00	12.25	
D1	9.90	10.00	10.10	Note 2
E	11.75	12.00	12.25	
E1	9.90	10.00	10.10	Note 2
B	0.30	–	0.45	
C	0.09	–	0.20	
L	0.45	–	0.75	
e	0.80 TYP			

Notes:
1. This package conforms to JEDEC reference MS-026, Variation ACB.
2. Dimensions D1 and E1 do not include mold protrusion. Allowable protrusion is 0.25 mm per side. Dimensions D1 and E1 are maximum plastic body size dimensions including mold mismatch.
3. Lead coplanarity is 0.10 mm maximum.

10/5/2001

	TITLE	DRAWING NO.	REV.
2325 Orchard Parkway San Jose, CA 95131	**44A**, 44-lead, 10 x 10 mm Body Size, 1.0 mm Body Thickness, 0.8 mm Lead Pitch, Thin Profile Plastic Quad Flat Package (TQFP)	44A	B

LAMPIRAN D

DATASHEET GLCD

1. Basic Specifications

1.1 Display Specifications

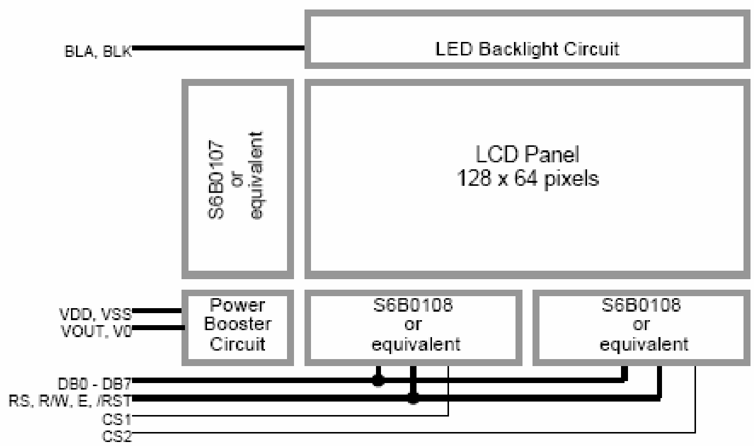
1) LCD Display Mode	: STN, Negative, Transmissive
2) Display Color	: Display Data = "1" : Light Gray (*1)
	: Display Data = "0" : Deep Blue (*2)
3) Viewing Angle	: 6 H
4) Driving Method	: 1/64 duty, 1/ 9bias
5) Back Light	: White LED backlight

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

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

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 form one of the driver/controller in the LCD module at a time, otherwise unexpected data collision may occur.

2. Absolute Maximum Ratings

Items	Symbol	Min.	Max.	Unit	Condition
Supply Voltage	V_{DD}	0	7.0	V	V_{SS} = 0V
Operating Temperature	T_{OP}	-20	70	°C	No Condensation
Storage Temperature	T_{ST}	-30	80	°C	No Condensation

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

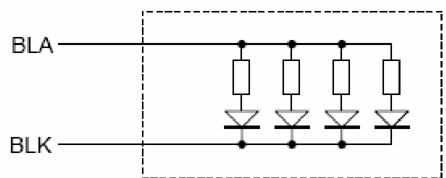
V_{SS}=0V, V_{DD} =5V, T_{OP} =25°C

Items	Symbol	MIN.	TYP.	MAX.	Unit	Applicable Pin
Operating Voltage	V_{DD}	4.8	5.0	5.2	V	VDD
Input High Voltage	V_{IH}	3.5	-	V_{DD}	V	RS, R/W, E, CS1, CS2, DB0-DB7
Input Low Voltage	V_{IL}	0	-	0.4	V	
Operating Current	I_{DD}	-	6.5	15	mA	VDD, VSS

3.2 LED Backlight Circuit Characteristics

V_{BLK}=0V, If_{BLA}=80mA, T_{OP}=25°C

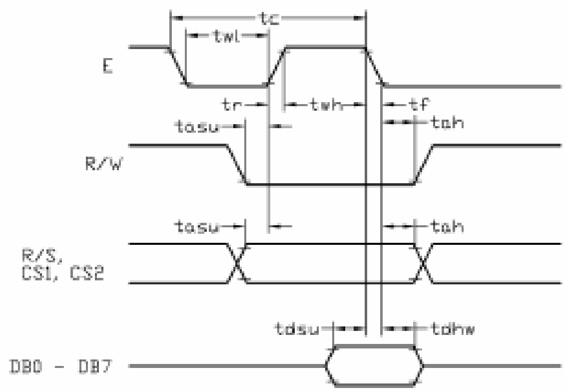
Items	Symbol	MIN.	TYP.	MAX.	Unit	Applicable Pin
Forward Voltage	Vf_{BLA}	-	4.9	-	V	BLA, BLK
Forward Current	If_{BLA}	-	-	120	mA	BLA, BLK



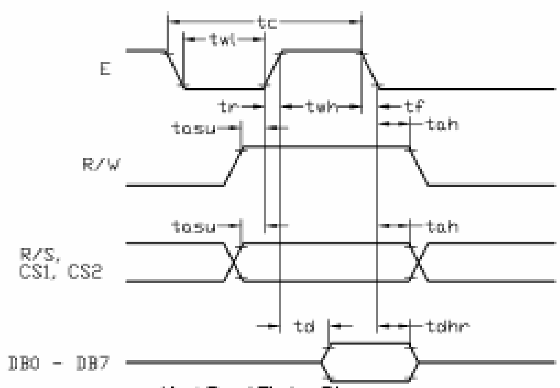
3.3 AC Characteristics

V_{SS}=0V, V_{DD} =5V, T_{OP} =25°C

Item	Symbol	MIN.	TYP.	MAX.	Unit
E cycle time	tc	1500	-	-	ns
E high level width	twh	700	-	-	ns
E low level width	twl	700	-	-	ns
E rise time	tr	-	-	18	ns
E fall time	tf	-	-	18	ns
Address set-up time	tasu	210	-	-	ns
Address hold time	tah	15	-	-	ns
Data set-up time	tdsu	300	-	-	ns
Data delay time	td	-	-	480	ns
Data hold time (write)	tdhw	15	-	-	ns
Data hold time (read)	tdhr	30	-	-	ns



Host Write Timing Diagram



Host Read Timing Diagram

4. Function Specifications

4.1 Basic Setting

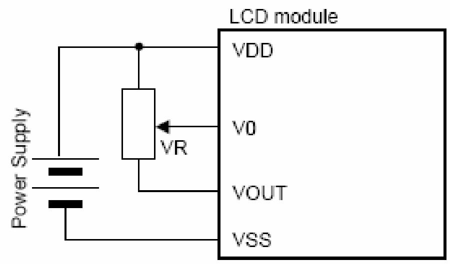
To drive the LCD module correctly and provide normally display, please use the following setting

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 the change of LCD display contrast.
The recommended value of VR is 25k to 50k



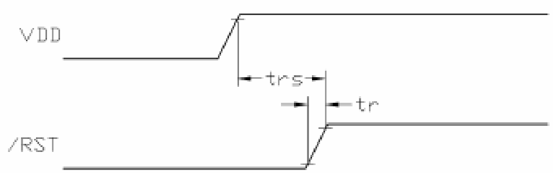
4.3 Resetting the LCD module

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 DB4=0 (clear /RST) and DB7=0 (ready) by status read instruction. The conditions of power supply at initial power up are as follow:

Item	Symbol	MIN.	TYP.	MAX.	Unit
Reset time	trs	2.0	-	-	us
Rise time	tr	-	-	150	ns



4.4 Display Memory Map

Page (X) address	data	LCD Display (front view)	
0	D0 : D7		
1	D0 : D7		
2	D0 : D7		
3	D0 : D7	128x64 pixels	
4	D0 : D7		
5	D0 : D7		
6	D0 : D7		
7	D0 : D7		
Column(Y) Address		00h → 3Fh	00h → 3Fh
Chip Select		CS1=1, CS2=0	CS1=0, CS2=1

Note:

1) Display star 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.

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

<u>Column (Y) Address Counter</u>
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.

<u>Display Start Line (Z) Register</u>
Z address register indicates of display data RAM to LCD top line. It may be used for scrolling the display pattern on the LCD.

4.6 Display Control Instructions

Instructions	Code										Function	Note
	RS	R/W	DB7	DB6	DB5	DB4	DB3	DB2	DB1	DB0		
Display on/off	0	0	0	0	1	1	1	1	1	1/0	Controls the display on or off. Internal status and display data in RAM is not affected. 0: off, 1: on	*2
Set Column (Y) Address	0	0	0	1	Y address (0~63)						Set the Column address into the Y address counter	*2
Set Page (X) Address	0	0	1	0	1	1	1	X address (0~7)			Set the Page address into the X address register	*2
Set Display Start Line (Z address)	0	0	1	1	Z address (0~63)						Indicates the display data RAM displayed at the top line screen	*2
Status Read	0	1	Busy	0	ON/OFF	Reset	0	0	0	0	Read status. Busy 1: Under ready; Busy 0: Ready to accept instruction; ON/OFF 1: Display is on; ON/OFF 0: Display is off; Reset L: Normal Running; Reset H: reset	*1
Write Display Data	1	0	Write data								Write display data into display data RAM. After writing instruction, Y address counter increased by 1 automatically	*2
Read Display Data	1	1	Read data								Read display data from the display data RAM	*1

Note:

*1. Only one section (driver) could be read at the same time.
i.e. only the following setting is valid for the operation:
CS1=1, CS2=0 is valid
CS1=0, CS2=1 is valid

*2. Instruction could be write into both drivers at the same time.

*3. For the details of the Display Control Instructions, please refer to Samsung KS0108 handbook.

5. Design and Handling Precaution

1. The LCD panel is made by glass. Any mechanical shock (eg. dropping form 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 (eg. water) may damage the LCD.
6. When mounting the LCD module, make sure that it is free form 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 (eg. 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