LAMPIRAN A

PROGRAM PADA PENGONTROL MIKRO ATMega16
PENGIRIM
/*******************************************/

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: 9/29/2010
Author: F4CG
Company: F4CG
Comments:

Chip type: ATmega16
Program type: Application
Clock frequency: 8.000000 MHz
Memory model: Small
External SRAM size: 0
Data Stack size: 256

/*******************************************/
#include <mega16.h>
#include<delay.h>
#include<stdio.h>

// Alphanumeric LCD Module functions
#asm
   .equ __lcd_port=0x15 ;PORTC
#endasm
#include <lcd.h>
unsigned int n,indexON,indexOFF,start,e,awal,nostart;
char startbitLCD[13];
char dataLCD[14];
unsigned int error,fs,flagON,flagOFF;

// External Interrupt 2 service routine
interrupt [EXT_INT2] void ext_int2_isr(void)
{
    // Place your code here

    if (error==1)
    {
        error=0;
        if (nostart==0)start=24;
        if (nostart==1)start=36;
        if (nostart==2)start=48;e=1;

        if(n==1)
        {
            flagON=indexON;
            sprintf(startbitLCD,"Start=%d",start);
            sprintf(dataLCD,"DataON=%d",dataON[flagON]);
            lcd_clear();
            lcd_gotoxy(0,0);
            lcd_puts(startbitLCD);
            lcd_gotoxy(0,1);
            lcd_puts(dataLCD);
            if(indexON==10)indexON=0;
        }
        if(n==2)

{
  flagOFF=indexOFF;
  sprintf(startbitLCD,"Start=%d",start);
      sprintf(dataLCD,"DataOFF=%d",dataOFF[flagOFF]);
    lcd_clear();
    lcd_gotoxy(0,0);
    lcd_puts(startbitLCD);
    lcd_gotoxy(0,1);
    lcd_puts(dataLCD);
  if(indexOFF==10)indexOFF=0;
}
}
else
{
  e=1;
  fs=1;//flag no_start
  if (nostart==0)start=36;
  if (nostart==1)start=48;
  if (nostart==2)start=24;

  if(n==1)
    {
      flagON=indexON;
      flagON=flagON+1;
      indexON=indexON+1;
      sprintf(startbitLCD,"Start=%d",start);
      sprintf(dataLCD,"DataON=%d",dataON[flagON]);
      lcd_clear();
      lcd_gotoxy(0,0);
      lcd_puts(startbitLCD);
      lcd_gotoxy(0,1);
}
lcd_puts(dataLCD);
if(indexON==10)indexON=0;
}
if(n==2)
{
flagOFF=indexOFF;
flagOFF=flagOFF+1;
indexOFF=indexOFF+1;
sprintf(startbitLCD,"Start=%d",start);
sprintf(dataLCD,"DataOFF=%d",dataOFF[flagOFF]);
lcd_clear();
lcd_gotoxy(0,0);
lcd_puts(startbitLCD);
lcd_gotoxy(0,1);
lcd_puts(dataLCD);
if(indexOFF==10)indexOFF=0;
}
}
}
delay_ms(700);

// Declare your global variables here
unsigned int
data1,data2,data3,data4,data5,data6,data7,data8,data9,data10,data11,data0;

unsigned int
datar1,datar2,datar3,datar4,datar5,datar6,datar7,datar8,datar9,datar10,datar11,datar12;
unsigned int dataEns,flag;
unsigned long int a,ss;
unsigned int index1,count,nostartR;
unsigned int deteksi1,deteksi2,deteksi3,deteksi4,deteksi5,deteksi6,deteksi7,deteksi8,deteksi9,deteksi10,deteksi11,deteksi12;
unsigned int bit0,bit1,bit2,bit3,bit4,bit5,bit6,bit7,bit8,bit9,bit10,bit11;

void header()
{
    PORTA.1=0;
delay_us(2400);
    PORTA.1=1;
delay_us(600);
}

void header1()
{
    PORTA.1=0;
delay_us(3600);
    PORTA.1=1;
delay_us(600);
}

void header2()
{
    PORTA.1=0;
delay_us(4800);
    PORTA.1=1;
delay_us(600);
}

void logic0()
{

PORTA.1=0;
delay_us(600);
PORTA.1=1;
delay_us(600);
}

void logic1()
{
PORTA.1=0;
delay_us(1200);
PORTA.1=1;
delay_us(600);
}

void konversiON()
{
dataEns=dataON[flagON]^255;
data0=dataEns/2;
bit0=dataEns%2;
data1=data0/2;
bit1=data0%2;
data2=data1/2;
bit2=data1%2;
data3=data2/2;
bit3=data2%2;
data4=data3/2;
bit4=data3%2;
data5=data4/2;
bit5=data4%2;
data6=data5/2;
bit6=data5%2;
data7=data6/2;
bit7=data6%2;
data8 = data7/2;
bit8 = data7 % 2;
data9 = data8/2;
bit9 = data8 % 2;
data10 = data9/2;
bit10 = data9 % 2;
data11 = data10/2;
bit11 = data10 % 2;
}

void konversiOFF()
{
dataEns = dataOFF[flagOFF]^255;
   data0 = dataEns/2;
   bit0 = dataEns % 2;
data1 = data0/2;
   bit1 = data0 % 2;
data2 = data1/2;
   bit2 = data1 % 2;
data3 = data2/2;
   bit3 = data2 % 2;
data4 = data3/2;
   bit4 = data3 % 2;
data5 = data4/2;
   bit5 = data4 % 2;
data6 = data5/2;
   bit6 = data5 % 2;
data7 = data6/2;
   bit7 = data6 % 2;
data8 = data7/2;
   bit8 = data7 % 2;
data9 = data8/2;
   bit9 = data8 % 2;
data10=data9/2;
bit10=data9%2;
data11=data10/2;
bit11=data10%2;
}
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=0x33;
  DDRA=0x02;

  // 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=0x45;
  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=0x00;

  // 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=0x00;

// 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
GICR|=0x20;
MCUCR=0x00;
MCUCSR=0x00;
GIFR=0x20;

// 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;

// LCD module initialization
lcd_init(16);

#asm("sei")
indexON=0;
indexOFF=0;
nostart=0;
flag=0;
a=0;
n=0;
e=0;
awal=0;
error=0;
fs=0;
ss=0;

while (1)
{
    // Place your code here
    ulang:
    while(error==1){}
    if(e==1){e=0;goto ulangdata;}
    while(PINB.0==1 && PINB.6==1);
    if(e==1){e=0;goto ulang;}

    if(PINB.0==0)
    {
        while(PINB.0==0);
        flag=1;  //tanda ON
        n=1;
    }
konversiON();
}
if(e==1){e=0;goto ulang;}
if(PINB.6==0)
{
  while(PINB.6==0);
  flag=2;      //tanda OFF
  n=2;
  konversiOFF();
}
if(e==1){e=0;goto ulang;}
  if(e==1){e=0;goto ulang;}
ulangdata:
if(fs==1)
{
  fs=0;
  ss=1;
  if(nostart==0)header1();
  if(nostart==1)header2();
  if(nostart==2)header();
  nostart=nostart+1;
  if(nostart==3)nostart=0;
}
else
{
  if(nostart==0)header();
  if(nostart==1)header1();
  if(nostart==2)header2();
}
if(bit0==0)
{
  logic0();
}
} 
else logic1(); 
if(bit1==0) 
{
    logic0();
} 
else logic1(); 
if(bit2==0) 
{
    logic0();
} 
else logic1(); 
if(bit3==0) 
{
    logic0();
} 
else logic1(); 
if(bit4==0) 
{
    logic0();
} 
else logic1(); 
if(bit5==0) 
{
    logic0();
} 
else logic1(); 
if(bit6==0) 
{
    logic0();
}
A - 14

else logic1();
if(bit7==0)
{
logic0();
}
else logic1();
if(bit8==0)
{
logic0();
}
else logic1();

if(bit9==0)
{
logic0();
}
else logic1();
if(bit10==0)
{
logic0();
}
else logic1();

if(bit11==0)
{
logic0();
}
else logic1();
if(e==1){e=0;goto ulang;}
if(e==1){e=0;goto ulang;}
while(PINA.0==1){if(e==1){e=0;goto ulang;}}
if(e==1){e=0;goto ulang;}
while (PINA.0==0)
{
    delay_us(100);
    count++;
    if(e==1){e=0;goto ulang;}
};
    start=count;
    count=0;
    if(e==1){e=0;goto ulang;}
if ((start>=24 && start<=27)||(start>=36 && start<=39)||(start>=48 && start<=51))
    
    {while (PINA.0==1) {};
    while (PINA.0==0)
    {
        delay_us(100);
        count++;
    };
    deteksi1=count;
    count=0;
    while (PINA.0==1) {};
    while (PINA.0==0)
    {
        delay_us(100);
        count++;
    };
    deteksi2=count;
    count=0;
    while (PINA.0==1) {};
    while (PINA.0==0)
{ 
    delay_us(100);
    count++;
};
    deteksi3=count;
    count=0;
    while (PINA.0==1) {}
    while (PINA.0==0)
    {
        delay_us(100);
        count++;
    }
    deteksi4=count;
    count=0;
    while (PINA.0==1) {}
    while (PINA.0==0)
    {
        delay_us(100);
        count++;
    }
    deteksi5=count;
    count=0;
    while (PINA.0==1) {}
    while (PINA.0==0)
    {
        delay_us(100);
        count++;
    }
    deteksi6=count;
    count=0;
    if(e==1){e=0;goto ulang;}
    while (PINA.0==1) {};
}
while (PIN.A0==0)
{
delay_us(100);
count++;
};
deteksi7=count;
count=0;
while (PIN.A0==1) {};
while (PIN.A0==0)
{
delay_us(100);
count++;
};
deteksi8=count;
count=0;
while (PIN.A0==1) {};
while (PIN.A0==0)
{
delay_us(100);
count++;
};
deteksi9=count;
count=0;
while (PIN.A0==1) {};
while (PIN.A0==0)
{
delay_us(100);
count++;
};
deteksi10=count;
count=0;
while (PIN.A0==1) {};
while (PINA.0==0)
{
delay_us(100);
count++;
};
deteksi11=count;
count=0;
while (PINA.0==1) {};
while (PINA.0==0)
{
delay_us(100);
count++;
};
deteksi12=count;
count=0;
if(e==1){e=0;goto ulang;}

//=========CEK 12 DATA==============
datar1=((deteksi1/5)-1)*1;//data hasil deteksi pulsa dibagi 5 dikurangi 1
datar2=((deteksi2/5)-1)*2;//jadi jika datanya 6 maka outputnya akan = 0
datar3=((deteksi3/5)-1)*4;//sedangkan jika datanya 12 maka outputnya akan = 1
datar4=((deteksi4/5)-1)*8;//lalu hasil tersebut dikalikan dengan nilai2 bit
datar5=((deteksi5/5)-1)*16;
datar6=((deteksi6/5)-1)*32;
datar7=((deteksi7/5)-1)*64;
datar8=((deteksi8/5)-1)*128;
datar9=((deteksi9/5)-1)*256;
datar10=((deteksi10/5)-1)*512;
datar11=((deteksi11/5)-1)*1024;
datar12=((deteksi12/5)-1)*2048;
if(e==1){e=0;goto ulang;}


//======PENJUMLAHAN 12 DATA=====

index1=datar12+datar11+datar10+datar9+datar8+datar7+datar6+datar5+datar4+datar3+datar2+datar1;
index1=index1^255;
if(start>=24 && start<=27){start=24;nostartR=0;}
if(start>=36 && start<=39){start=36;nostartR=1;}
if(start>=48 && start<=51){start=48;nostartR=2;}
if(flag==1)
{
    flag=0;
    if(index1==dataON[indexON] && nostart==nostartR)
    {
        if(e==1){e=0;goto ulang;}
        sprintf(startbitLCD,"Start=%d",start);
        sprintf(dataLCD,"DataON=%d",dataON[indexON]);
        lcd_clear();
        lcd_gotoxy(0,0);
        lcd_puts(startbitLCD);
        lcd_gotoxy(0,1);
        lcd_puts(dataLCD);
        if(e==1){e=0;goto ulang;}else
        {nostart=nostart+1;indexON=indexON+1;flagON=indexON;}
        if (indexON==10)indexON=0;
    }
    else {delay_ms(65);error=1;}
}
if(e==1){e=0;goto ulang;}
if(flag==2)
{
    flag=0;
    if (index1==dataOFF[indexOFF] && nostart==nostartR)
{
    sprintf(startbitLCD,"Start=%d",start);
    sprintf(dataLCD,"DataOFF=%d",dataOFF[indexOFF]);
    lcd_clear();
    lcd_gotoxy(0,0);
    lcd_puts(startbitLCD);
    lcd_gotoxy(0,1);
    lcd_puts(dataLCD);
    if(e==1){e=0;goto ulang;}else
        {nostart=nostart+1;indexOFF=indexOFF+1;flagOFF=indexOFF;}
    if (indexOFF==10)indexOFF=0;
    }
else {delay_ms(65);error=1;}
}
if (nostart==3)nostart=0;
}
if(e==1){e=0;goto ulang;}
delay_ms(100);
};
PENERIMA
#include <mega16.h>
#include<delay.h>
#include<stdio.h>

// Alphanumeric LCD Module functions
#asm
   .equ __lcd_port=0x15 ;PORTC
// Declare your global variables here
unsigned int data1, data2, data3, data4, data5, data6, data7, data8, data9, data10, data11, data0;
unsigned int startBit[5] = {24, 36, 48};
unsigned int count, deteksi1, deteksi2, deteksi3, deteksi4, deteksi5, deteksi6, deteksi7, deteksi8, deteksi9, deteksi10, deteksi11, deteksi12;
unsigned int datar1, datar2, datar3, datar4, datar5, datar6, datar7, datar8, datar9, datar10, datar11, datar12;
unsigned int bit0, bit1, bit2, bit3, bit4, bit5, bit6, bit7, bit8, bit9, bit10, bit11;
unsigned int start, nomorON, nomorOFF, nomorStart, errorOFF, errorON;
unsigned int dataEns;
unsigned int index;
char startbitLCD[16];
char dataLCD[16];

void header()
{
    PORTA.1 = 0;
delay_us(2400);
    PORTA.1 = 1;
delay_us(600);
}

void header1()
PORTA.1=0;
delay_us(3600);
PORTA.1=1;
delay_us(600);
}

void header2()
{
    PORTA.1=0;
delay_us(4800);
    PORTA.1=1;
delay_us(600);
}

void logic0()
{
    PORTA.1=0;
delay_us(600);
    PORTA.1=1;
delay_us(600);
}

void logic1()
{
    PORTA.1=0;
delay_us(1200);
    PORTA.1=1;
delay_us(600);
}

void konversi()
{ 
    dataEns=index^255;
    data0=dataEns/2;
    bit0=dataEns%2;
    data1=data0/2;
    bit1=data0%2;
    data2=data1/2;
    bit2=data1%2;
    data3=data2/2;
    bit3=data2%2;
    data4=data3/2;
    bit4=data3%2;
    data5=data4/2;
    bit5=data4%2;
    data6=data5/2;
    bit6=data5%2;
    data7=data6/2;
    bit7=data6%2;
    data8=data7/2;
    bit8=data7%2;
    data9=data8/2;
    bit9=data8%2;
    data10=data9/2;
    bit10=data9%2;
    data11=data10/2;
    bit11=data10%2;
}

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=0x33;
DDRA=0x02;

// 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=0x00;

// 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=0x88;

// 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;

// LCD module initialization
lcd_init(16);

nomorON=0;
nomorOFF=0;
nomorStart=0;
errorON=0;
errorOFF=0;
while (1)
{
    // Place your code here
    while (PINA.0==1) {}; 
    while (PINA.0==0)
    {

        delay_us(100);
        count++;
    }
    start=count;
    count=0;
    if ((start>=24 && start<=27)||(start>=36 && start<=39)||(start>=48 && start<=51))
    {
        while (PINA.0==1) {}; 
        while (PINA.0==0)
        {
            delay_us(100);
            count++;
        }
        deteksi1=count;
        count=0;
        while (PINA.0==1) {}; 
        while (PINA.0==0)
        {
            delay_us(100);
            count++;
        }
        deteksi2=count;
        count=0;
        while (PINA.0==1) {};
while (PINA.0==0)
{
    delay_us(100);
    count++;
};
    deteksi3=count;
    count=0;
    while (PINA.0==1) {};
    while (PINA.0==0)
{
    delay_us(100);
    count++;
};
    deteksi4=count;
    count=0;
    while (PINA.0==1) {};
    while (PINA.0==0)
{
    delay_us(100);
    count++;
};
    deteksi5=count;
    count=0;
    while (PINA.0==1) {};
    while (PINA.0==0)
{
    delay_us(100);
    count++;
};
    deteksi6=count;
    count=0;
    while (PINA.0==1) {};}
while (PIN.0 == 0) {
    delay_us(100);
    count++;
};
deteksi7 = count;
count = 0;
while (PIN.0 == 1) {};
while (PIN.0 == 0) {
    delay_us(100);
    count++;
};
deteksi8 = count;
count = 0;
while (PIN.0 == 1) {};
while (PIN.0 == 0) {
    delay_us(100);
    count++;
};
deteksi9 = count;
count = 0;
while (PIN.0 == 1) {};
while (PIN.0 == 0) {
    delay_us(100);
    count++;
};
deteksi10 = count;
count = 0;
while (PIN.0 == 1) {};};
while (PINA.0==0) 
{
delay_us(100);
count++;
};
deteksi11=count;
count=0;
while (PINA.0==1) {};
while (PINA.0==0) 
{
delay_us(100);
count++;
};
deteksi12=count;
count=0;

//========CEK 12 DATA=============
datar1=((deteksi1/5)-1)*1;//data hasil deteksi pulsa dibagi 5 dikurangi 1
datar2=((deteksi2/5)-1)*2;//jadi jika datanya 6 maka outputnya akan = 0
datar3=((deteksi3/5)-1)*4;//sedangkan jika datanya 12 maka outputnya akan = 1
datar4=((deteksi4/5)-1)*8;//lalu hasil tersebut dikalikan dengan nilai 2 bit
datar5=((deteksi5/5)-1)*16;
datar6=((deteksi6/5)-1)*32;
datar7=((deteksi7/5)-1)*64;
datar8=((deteksi8/5)-1)*128;
datar9=((deteksi9/5)-1)*256;
datar10=((deteksi10/5)-1)*512;
datar11=((deteksi11/5)-1)*1024;
datar12=((deteksi12/5)-1)*2048;

//=========PENJUMLAHAN 12 DATA====
index=datar12+datar11+datar10+datar9+datar8+datar7+datar6+datar5+datar4+datar3+datar2+datar1;
    index=index^255;
    if(start>=24 && start<=27)start=24;
    if(start>=36 && start<=39)start=36;
    if(start>=48 && start<=51)start=48;
    if(index>=128 && index<=137)
    {
        if (index==dataON[nomorON] && start==startBit[nomorStart])
        {
            if (errorOFF==1){PORTD.3=0;goto lanjut;}
            PORTD.3=1;
            nomorON=nomorON+1;
            nomorStart=nomorStart+1;
            if(nomorStart==3)nomorStart=0;
            if(nomorON==10)nomorON=0;
        } else errorON=1;
    }
else if(index>=138 && index<=147)
    {
        if(index==dataOFF[nomorOFF] && start==startBit[nomorStart])
        {
            if (errorON==1){PORTD.3=1;goto lanjut;}
            PORTD.3=0;
            nomorOFF=nomorOFF+1;
            nomorStart=nomorStart+1;
            if(nomorStart==3)nomorStart=0;
            if(nomorOFF==10)nomorOFF=0;
        } else errorOFF=1;
lanjut:
sprintf(startbitLCD,"StartBit=%d",start);
sprintf(dataLCD,"Data=%d",index);
lcd_gotoxy(0,0);
lcd_puts(startbitLCD);
lcd_gotoxy(0,1);
lcd_puts(dataLCD);
delay_ms(100); //delay 100ms
konversi();
if(start==24)header();
if(start==36)header1();
if(start==48)header2();
if(bit0==0)
{
 logic0();
}
else logic1();
if(bit1==0)
{
 logic0();
}
else logic1();
if(bit2==0)
{
 logic0();
}
else logic1();
if(bit3==0)
{
 logic0();
}
else logic1();
if(bit4==0)
{
 logic0();
}
else logic1();
if(bit5==0)
{
 logic0();
}
else logic1();
if(bit6==0)
{
 logic0();
}
else logic1();
if(bit7==0)
{
 logic0();
}
else logic1();
if(bit8==0)
{
 logic0();
}
else logic1();
if(bit9==0)
{
 logic0();
}
else logic1();
if(bit10==0)
{  
    logic0();  
}
else logic1();

if(bit11==0)  
{  
    logic0();  
}
else logic1();
}

};
}
LAMPIRAN B
DATA SHEET ATMega16
Features

- High-performance, Low-power Atmel® 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
  - One-chip 2-cycle Multiplier
- High Endurance Non-volatile Memory segments
  - 16 Kbytes of In-System Self-programmable Flash program memory
  - 512 Bytes EEPROM
  - 1 Kbyte Internal SRAM
  - Write/Erase Cycles: 10,000 Flash/100,000 EEPROM
  - Data retention: 20 years at 85°C/100 years at 25°C
  - Optional Boot Code Section with Independent Lock Bits
  - In-System Programming by On-chip Boot Program
  - True Read-While-Write Operation
  - 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, 16x, 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 QFN/MLF
- Operating Voltages
  - 2.7V - 5.5V for ATmega16L
  - 4.5V - 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
ATmega16(L)

Figure 1. Pinout ATmega16 Configurations

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.

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: 16 Kbytes of In-System Programmable Flash Program memory with Read-While-Write capabilities, 512 bytes EEPROM, 1 Kbyte 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 (TGFP 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 debuggers/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 51.

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 51.

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.
I/O Ports

Introduction

All AVR ports have true Read-Modify-Write functionality when used as general digital I/O ports. This means that the direction of any port pin can be changed without unintentionally changing the direction of any other pin with the SBI and CBI instructions. The same applies when changing the value (if configured as output) or enabling/disabling of pull-up resistors (if configured as input). Each output buffer has symmetrical drive characteristics with both high sink and source capability. The pin driver is strong enough to drive LED displays directly. All port pins have individually selectable pull-up resistors with a supply-voltage inversion resistance. All I/O pins have protection diodes to both VCC and ground as indicated in Figure 22. Refer to “Electrical Characteristics” on page 265 for a complete list of parameters.

Figure 22. I/O Pin Equivalent Schematic

All registers and bit references in this section are written in general form. A lower case ‘x’ represents the numbering letter for the port, and a lower case ‘n’ represents the bit number. However, when using the register or bit names in a program, the precise form must be used. I.e., PORTB3 for bit no. 3 in Port B, here documented generally as PORTx. The physical I/O Registers and pin locations are listed in “Register Description for I/O Ports” on page 62.

Three I/O memory address locations are allocated for each port, one each for the Data Register – PORTx, Data Direction Register – DDRx, and the Port Input Pins – PINx. The Port Input Pins I/O location is read only, while the Data Register and the Data Direction Register are readable and writable. In addition, the Pull-up Disable – PUx bit in SFIOR disables the pull-up function for all pins in all ports when set.

Using the I/O port as General Digital I/O is described in “Ports as General Digital I/O” on page 49. Most port pins are multiplexed with alternate functions for peripheral features on the device. How each alternate function interfaces with the port pin is described in “Alternate Port Functions” on page 62. Refer to the individual module sections for a full description of the alternate functions.

Note that enabling the alternate function of some of the port pins does not affect the use of the other pins in the port as general digital I/O.
Ports as General Digital I/O

The ports are bi-directional I/O ports with optional internal pull-ups. Figure 23 shows a functional description of one I/O-port pin, here generically called Pin.

Figure 23. General Digital I/O

Note:
1. RPs, WDs, RRs, RPs, and RRs are common to all pins within the same port. DREG, SLEEP, and PUD are common to all ports.

Configuring the Pin

Each port pin consists of three register bits: EDm, PORTm, and Pinm. As shown in "Register Description for I/O Ports" on page 52, the EDm bits are accessed at the DREG i/o address, the PORTm bits at the PORTx I/O address, and the Pinm bits at the PINx I/O address.

The DEm bit in the DDRx Register selects the direction of the pin. If DEm is written logic one, Pinm is configured as an output pin. If DEm is written logic zero, Pinm is configured as an input pin.

If PORTm is written logic one when the pin is configured as an input pin, the pull-up resistor is activated. To switch the pull-up resistor off, PORTm has to be written logic zero or the pin has to be configured as an output pin. The port pins are tri-stated when a reset condition becomes active, even if no clocks are running.

If PORTm is written logic one when the pin is configured as an output pin, the port pin is driven high (1). If PORTm is written logic zero when the pin is configured as an output pin, the port pin is driven low (0).

When switching between tri-state (EDm, PORTm = 0b00) and output high (EDm, PORTm = 0b11), an intermediate state with either pull-up enabled (EDm, PORTm = 0b01) or output low (EDm, PORTm = 0b10) must occur. Normally, the pull-up
enabled state is fully acceptable, as a high-impedance environment will not notice the difference between a strong high driver and a pull-up. If this is not the case, the PUD bit in the SFIOR Register can be set to disable all pull-ups in all ports.

Switching between input with pull-up and output low generates the same problem. The user must use either the tri-state (DDxN, PORTN = 0b00) or the output high state (DDxN, PORTN = 0b11) as an intermediate step.

Table 20 summarizes the control signals for the pin value.

<table>
<thead>
<tr>
<th>DDxN</th>
<th>PORTxN</th>
<th>PUD (in SFIOR)</th>
<th>I/O</th>
<th>Pull-up</th>
<th>Comment</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>1</td>
<td>X</td>
<td>Input</td>
<td>No</td>
<td>Tri-state (Hi-Z)</td>
</tr>
<tr>
<td>0</td>
<td>1</td>
<td>0</td>
<td>Input</td>
<td>Yes</td>
<td>Pin will source current if act. pulled low.</td>
</tr>
<tr>
<td>0</td>
<td>1</td>
<td>1</td>
<td>Input</td>
<td>No</td>
<td>Tri-state (Hi-Z)</td>
</tr>
<tr>
<td>1</td>
<td>0</td>
<td>X</td>
<td>Output</td>
<td>No</td>
<td>Output Low (Sink)</td>
</tr>
<tr>
<td>1</td>
<td>1</td>
<td>X</td>
<td>Output</td>
<td>No</td>
<td>Output High (Source)</td>
</tr>
</tbody>
</table>

**Reading the Pin Value**

Independent of the setting of Data Direction bit DDxN, the port pin can be read through the PORTxN Register bit. As shown in Figure 25, the PORTxN Register bit and the preceding latch constitute a synchronizer. This is needed to avoid metastability if the physical pin changes value near the edge of the internal clock, but it also introduces a delay. Figure 24 shows a timing diagram of the synchronization when reading an externally applied pin value. The maximum and minimum propagation delays are denoted $t_{PLH_{max}}$ and $t_{PLH_{min}}$ respectively.

**Figure 24. Synchronization when Reading an Externally Applied Pin Value**

Consider the clock period starting shortly after the first falling edge of the system clock. The latch is closed when the clock is low, and goes transparent when the clock is high, as indicated by the shaded region of the "SYNC LATCH" signal. The signal value is latched when the system clock goes low. It is clocked into the PORTxN Register at the
succeeding positive clock edge. As indicated by the two arrows $t_{pd\text{max}}$ and $t_{pd\text{min}}$, a single signal transition on the pin will be delayed between $1/2$ and $1$ system clock period depending upon the time of assertion.

When reading back a software assigned pin value, a `nop` instruction must be inserted as indicated in Figure 25. The `out` instruction sets the "SYNC LATCH" signal at the positive edge of the clock. In this case, the delay $t_{pd}$ through the synchronizer is one system clock period.

Figure 25. Synchronization when Reading a Software Assigned Pin Value
Timer/Counter0 and Timer/Counter1 Prescalers

Timer/Counter1 and Timer/Counter0 share the same prescaler module, but the Timer/Counters can have different prescaler settings. The description below applies to both Timer/Counter1 and Timer/Counter0.

Internal Clock Source

The Timer/Counter can be clocked directly by the system clock (by setting the CSn2.0 = 1). This provides the fastest operation, with a maximum Timer/Counter clock frequency equal to system clock frequency (f_{CLK,sys}). Alternatively, one of four taps from the prescaler can be used as a clock source. The prescaled clock has a frequency of either f_{CLK,sys}/8, f_{CLK,sys}/64, f_{CLK,sys}/256, or f_{CLK,sys}/1024.

Prescaler Reset

The prescaler is free running, i.e., operates independently of the clock select logic of the Timer/Counter, and it is shared by Timer/Counter1 and Timer/Counter0. Since the prescaler is not affected by the Timer/Counter's clock select, the state of the prescaler will have implications for situations where a prescaled clock is used. One example of prescaling artifacts occurs when the timer is enabled and clocked by the prescaler (6 > CSn2.0 > 1). The number of system clock cycles from when the timer is enabled to the first count occurs can be from 1 to N+1 system clock cycles, where N equals the prescaler divisor (5, 64, 256, or 1024).

It is possible to use the Prescaler Reset for synchronizing the Timer/Counter to program execution. However, care must be taken if the other Timer/Counter that shares the same prescaler also uses prescaling. A prescaler reset will affect the prescaler period for all Timer/Counters it is connected to.

External Clock Source

An external clock source applied to the T1/T0 pin can be used as Timer/Counter clock (c12\_f\_cfr2). The T1/T0 pin is sampled once every system clock cycle by the pin synchronization logic. The synchronized (sampled) signal is then passed through the edge detector. Figure 28 shows a functional equivalent block diagram of the T1/T0 synchronization and edge detector logic. The registers are clocked at the positive edge of the internal system clock (\(\mathit{clk}_{\text{sys}}\)). The latch is transparent in the high period of the internal system clock.

The edge detector generates one \(\mathit{clk}_{\text{t}}\_\mathit{f}_{\text{cfr}}\_\mathit{edge}\) pulse for each positive (\(\mathit{CSn}2.0 = 7\)) or negative (\(\mathit{CSn}2.0 = 6\)) edge it detects.

Figure 28. T1/T0 Pin Sampling

The synchronization and edge detector logic introduces a delay of 2.5 to 3.5 system clock cycles from an edge has been applied to the T1/T0 pin to the counter is updated.

Enabling and disabling of the clock input must be done when T1/T0 has been stable for at least one system clock cycle, otherwise it is a risk that a false Timer/Counter clock pulse is generated.

Each half period of the external clock applied must be longer than one system clock cycle to ensure correct sampling. The external clock must be guaranteed to have less...
than half the system clock frequency \( f_{\text{sys}} < \frac{f_{\text{sys}}}{2} \) given a 50/50% duty cycle. Since the edge detector uses sampling, the maximum frequency of an external clock it can detect is half the sampling frequency (Nyquist sampling theorem). However, due to variation of the system clock frequency and duty cycle caused by Oscillator source (crystal, resonator, and capacitors) tolerances, it is recommended that maximum frequency of an external clock source be less than \( \frac{f_{\text{sys}}}{2.5} \).

An external clock source cannot be prescaled.

**Figure 99. Prescaler for Timer/Counter0 and Timer/Counter1**

Note: 1. The synchronization logic on the input pins (T1/T0) is shown in Figure 28.

**Special Function IO Register - SFIO**

<table>
<thead>
<tr>
<th>Bit</th>
<th>7</th>
<th>6</th>
<th>5</th>
<th>4</th>
<th>3</th>
<th>2</th>
<th>1</th>
<th>0</th>
</tr>
</thead>
<tbody>
<tr>
<td>Read/Write</td>
<td>W</td>
<td>W</td>
<td>W</td>
<td>W</td>
<td>W</td>
<td>W</td>
<td>W</td>
<td></td>
</tr>
<tr>
<td>Initial Value</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
</tbody>
</table>

- Bit 0 - PSR[10]: Prescaler Reset Timer/Counter1 and Timer/Counter0

When this bit is written to one, the Timer/Counter1 and Timer/Counter0 prescaler will be reset. The bit will be cleared by hardware after the operation is performed. Setting a zero to this bit will have no effect. Note that Timer/Counter1 and Timer/Counter0 share the same prescaler and a reset of this prescaler will affect both timers. This bit will always be read as zero.
16-bit Timer/Counter

The 16-bit Timer/Counter unit allows accurate program execution timing (event management), pulse generation, and signal timing measurement. The main features are:

- True 16-bit Design (i.e., Allows 16-bit PWM)
- Two Independent Output Compare Units
- Double Buffered Output Compare Registers
- One Input Capture Unit
- Input Capture Noise Canceler
- Clear Timer on Compare Match (Auto Reload)
- Global/interrupt Phase Correct Pulse Width Modulator (PWM)
- Variable PWM Period
- Frequency Generator
- External Event Counter
- Four Independent Interrupt Sources (TOV1, OCF1A, OCF1B, and ICP)

Overview

Most register and bit references in this document are written in general form. A lower case “n” replaces the Timer/Counter number, and a lower case “c” replaces the output compare unit channel. However, when using the register or bit definition in a program, the precise form must be used (i.e., TCNT1 for accessing Timer/Counter1 counter value and so on). The physical I/O Register and bit locations for Atmega16 are listed in the "16-bit Timer/Counter Register Descriptions" on page 104.

A simplified block diagram of the 16-bit Timer/Counter is shown in Figure 40. CPU accessible I/O Registers, including I/O bits and I/O pins, are shown in bold.
Figure 43. 16-bit Timer/Counter Block Diagram

| Registers | The Timer/Counter (TCNT1), Output Compare Registers (OCR1A/B), and Input Capture Register (ICRF1) are all 16-bit registers. Special procedures must be followed when accessing the 16-bit registers. These procedures are described in the section “Accessing 16-bit Registers” on page 69. The Timer/Counter Control Registers (TCCR1A/B) are 8-bit registers and have no CPU access restrictions. Interrupt requests (abbreviated to Int Req. in the figure) signals are all visible in the Timer Interrupt Flag Register (TIFR). All interrupts are individually enabled with the Timer Interrupt Mask Register (TIMSK). TIFR and TIMSK are not shown in the figure since these registers are shared by other timer units.

The Timer/Counter can be selected as a 16-bit input by the prescaler, or by an external clock source on the T1 pin. The Clock Select logic block controls which clock source or edge the Timer/Counter uses to increment (or decrement) its value. The Timer/Counter is inactive when no clock source is selected. The output from the clock select logic is referred to as the timer clock (dtclk).

The double buffered Output Compare Registers (OCR1A/B) are compared with the Timer/Counter value at all times. The result of the compare can be used by the waveform generator to generate a PWM or variable frequency output on the Output Compare pin.

ATmega16(L)
ATmega16(L)

[B-14](#) See "Output Compare Units" on page 91. The compare match event will also set the Compare Match Flag (CCF[1:0]) which can be used to generate an output compare interrupt request.

The Input Capture Register can capture the Timer/Counter value at a given external edge triggered event on either the Input Capture Pin (ICP) or on the Analog Comparator pin (See "Analog Comparator" on page 195.) The Input capture still includes a digital filtering unit (Noise Canceler) for reducing the chance of capturing noise spikes.

The TOP value, or maximum Timer/Counter value, can in some modes of operation be defined by either the OCR1A Register, the ICR1 Register, or by a set of fixed values. When using OCR1A as TOP value in a PWM mode, the OCR1A Register cannot be used for generating a PWM output. However, the TOP value will in this case be double buffered allowing the TOP value to be changed in run-time. If a fixed TOP value is selected, the ICR1 Register can be used as an alternative, treating the OCR1A to be used as PWM output.

Definitions

The following definitions are used extensively throughout the document.

<table>
<thead>
<tr>
<th>Definition</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>BOTTOM</td>
<td>The counter reaches the BOTTOM when it becomes 0x0000.</td>
</tr>
<tr>
<td>MAX</td>
<td>The counter reaches its Maximum when it becomes 0xFFFF (decimal 65535).</td>
</tr>
<tr>
<td>TOP</td>
<td>The counter reaches the TOP when it becomes equal to the highest value in the count sequence. The TOP value can be assigned to be one of the fixed values: 0x00FF, 0x0FFF, 0x3FFF, or to the value stored in the OCR1A or ICR1 registers. The assignment is dependent of the module operation.</td>
</tr>
</tbody>
</table>

Compatibility

The 16-bit Timer/Counter has been updated and improved from previous versions of the 16-bit AVR Timer/Counter. This 16-bit Timer/Counter is fully compatible with the earlier versions regarding:

- All 16-bit Timer/Counter related I/O Register address locations, including Timer Interrupt registers.
- Bit locations inside all 16-bit Timer/Counter Registers, including timer interrupt registers.
- Interrupt Vectors.

The following control bits have changed name, but have same functionality and register location:

- PW1 is changed to WGM16.
- PW1 is changed to WGM11.
- CTC1 is changed to WGM12.

The following bits are added to the 16-bit Timer/Counter Control Registers:

- FOC1A and FOC1B are added to TCCR1A.
- WGM8 is added to TCCR1B.

The 16-bit Timer/Counter has improvements that will affect the compatibility in some special cases.
Accessing 16-bit Registers

The TCON1, OCR1A/B, and ICR1 are 16-bit registers that can be accessed by the AVR CPU via the 8-bit data bus. The 16-bit register must be byte accessed using two read or write operations. Each 16-bit timer has a single 8-bit register for temporary storing of the high byte of the 16-bit access. The same temporary register is shared between all 16-bit registers within each 16-bit timer. Accessing the high byte triggers the 16-bit read or write operation. When the high byte of a 16-bit register is written by the CPU, the high byte stored in the temporary register, and the low byte written are both copied into the 16-bit register in the same clock cycle. When the low byte of a 16-bit register is read by the CPU, the high byte of the 16-bit register is copied into the temporary register in the same clock cycle as the low byte is read.

Not all 16-bit accesses use the temporary register for the high byte. Reading the OCR1A/B 16-bit registers does not involve using the temporary register.

To do a 16-bit write, the high byte must be written before the low byte. For a 16-bit read, the low byte must be read before the high byte.

The following code examples show how to access the 16-bit timer registers assuming that no interrupts updates the temporary register. The same principle can be used directly for accessing the OCR1A/B and ICR1 registers. Note that when using "C", the compiler handles the 16-bit access.

Assembly Code Example:

```assembly
...;
ldi r17, 0x00F
ldi r18, 0xFF
out TCON1, r17
out TCON1, r18
...;
```

C Code Example:

```c
unsigned int t;
...
/* Get TCON1 to OCR1F */
TCON1 = 0x00F;
/* Read TCON1 into I */
I = TCON1;
...
```

Note: 1. The example code assumes that the part-specific header file is included.

The assembly code example returns the TCON1 value in the r17:r16 register pair.

It is important to notice that accessing 16-bit registers are atomic operations. If an interrupt occurs between the two instructions accessing the 16-bit register, and the interrupt code updates the temporary register by accessing the same or any other of the 16-bit timer registers, then the result of the access outside the interrupt will be corrupted. Therefore, when both the main code and the interrupt code update the temporary register, the main code must disable the interrupts during the 16-bit access.
The following code examples show how to do an atomic read of the TCNT1 Register contents. Reading any of the OCR1A/B or ICR1 Registers can be done by using the same principle.

Assembly Code Example:

```
    ; Save global interrupt flag
    ldi r25, 0
    ; Enable interrupts
    cli
    ; Save TCNT1 into r23:r24
    in r23, TCNT1
    in r24, TCNT1
    ; Restore global interrupt flag
    out SREG, r25
    ret
```

C Code Example:

```c
    unsigned int TIME_getTCNT1(void )
    {
        unsigned char reg;
        unsigned int i;
        /* Save global interrupt flag */
        reg = SREG;
        /* Enable interrupts */
        cli();
        /* Save TCNT1 into i */
        i = TCNT1;
        /* Restore global interrupt flag */
        SREG = reg;
        return i;
    }
```

Note: The example code assumes that the part-specific header file is included.

The assembly code example returns the TCNT1 value in the r23:r24 register pair.
The following code examples show how to do an atomic write of the TCON1 Register contents. Writing any of the OCR1A/B or ICR1 Registers can be done by using the same principle.

### Assembly Code Example

```assembly
    ; Save global interrupt flag
    mov r14, #0
    ; Enable interrupts
    cli
    ; Set TOCN1 to 01 or 10
    mov r21, #0
    mov r21, #1
    ; Restore global interrupt flag
    setb r14
```

### C Code Example

```c
void TOCN1_write(unsigned int i)
{
    unsigned char src;
    unsigned char dst;
    /* Save global interrupt flag */
    src = SREG;
    /* Enable interrupts */
    CLI;
    /* Set TOCN1 to 1 */
    TOCN1 = i;
    /* Restore global interrupt flag */
    SREG = src;
}
```

Note: 1. The example code assumes that the port specific header file is included.

The assembly code example requires that the r14 register pair contains the value to be written to TOCN1.

### Using the Temporary High Byte Register

If writing to more than one 16-bit register where the high byte is the same for all registers written, then the high byte only needs to be written once. However, note that the same rule of atomic operation described previously also applies in this case.

### Timer/Counter Clock Sources

The Timer/Counter can be clocked by an internal or an external clock source. The clock source is selected by the Clock Select logic which is controlled by the Clock Select (CS2-0) bits located in the Timer/Counter Clock Register (TCCR1B). For details on clock sources and prescaler, see “Timer/Counter and TimerCounter1 Prescaler” on page 81.

### Counter Unit

The main part of the 16-bit Timer/Counter is the programmable 16-bit bi-directional counter unit. Figure 1 shows a block diagram of the counter and its surroundings.

---

**ATmega16(L)**
Figure 41: Counter Unit Block Diagram

Signal description (internal signals):

- **Count**: Increment or decrement TCNT1 by 1.
- **Direction**: Select between increment and decrement.
- **Clear**: Clear TCNT1 (set all bits to zero).
- **CKT1**: Timer/Counter clock.
- **TOP**: Signal that TCNT1 has reached maximum value.
- **BOTTOM**: Signal that TCNT1 has reached minimum value (zero).

The 16-bit counter is mapped into two 8-bit I/O memory locations: Counter High (TCNT1H) containing the upper eight bits of the counter, and Counter Low (TCNT1L) containing the lower eight bits. The TCNT1 Register can only be indirectly accessed by the CPU. When the CPU does an access to the TCNT1 I/O location, the CPU accesses the high byte temporary register (TEMP). The temporary register is updated with the TCNT1 value when the TCNT1L is read, and TCNT1 is updated with the temporary register value when TCNT1L is written. This allows the CPU to read or write the entire 16-bit counter value within one clock cycle via the I/O data bus. It is important to notice that there are special cases of writing to the TCNT1 Register when the counter is counting that will give unpredictable results. The special cases are described in the sections where they are of importance.

Depending on the mode of operation used, the counter is cleared, incremented, or decremented at each timer clock (CKT1). The CKT1 can be generated from an external or internal clock source, selected by the Clock Select bits (CS1:2:0). When no clock source is selected ((CS1:2:0 = 0)) the timer is stopped; however, the TCNT1 value can be accessed by the CPU independent of whether CS1:2:0 is present or not. A CPU write overrides (has priority over) all counter clear or reset operations.

The counting sequence is determined by the setting of the Timer/Counter Operation Mode bits (WGM13:0) located in the Timer/Counter Control Register A (TOCR1A) and TOCR1B. These are close connections between how the counter behaves (counts) and how waveforms are generated on the Output Compare outputs (OC1). For more details about advanced counting sequences and waveform generation, see "Modes of Operation" on page 91.

The Timer/Counter Overflow (TOV1) flag is set according to the mode of operation selected by the WGM13:0 bits. TOV1 can be used for generating a CPU interrupt.
Input Capture Unit

The Timer/Counter incorporates an input capture unit that can capture external events and give them a time-stamp indicating time of occurrence. The external signal indicating an event, or multiple events, can be applied via the ICP1 pin or alternatively via the Analog Comparator unit. The timestamps can then be used to calculate frequency, duty cycle, and other features of the signal applied. Alternatively, the time-stamps can be used for creating a log of the events.

The input capture unit is illustrated by the block diagram shown in Figure 42. The elements of the block diagram that are not directly a part of the input capture unit are grey shaded. The small "*" in register and bit names indicates the Timer/Counter number.

Figure 42. Input Capture Unit Block Diagram

When a change of the logic level (an event) occurs on the Input Capture pin (ICP1), alternatively on the Analog Comparator output (AOC), and this change confirms to the setting of the edge detector, a capture will be triggered. When a capture is triggered, the 16-bit value of the counter (TCNT1) is written to the Input Capture Register (ICR1). The Input Capture Flag (ICF1) is set at the same system clock as the TCNT1 value is copied into ICR1 Register. If enabled (TICE1 = 1), the input capture flag generates an input capture interrupt. The ICF1 flag is automatically cleared when the interrupt is executed. Alternatively the ICF1 flag can be cleared by software by writing a logical one to its I/O bit location.

Reading the 16-bit value in the Input Capture Register (ICR1) is done by first reading the low byte (ICRH) and then the high byte (ICRL). When the low byte is read, the high byte is copied into the high byte temporary register (TEMP). When the CPU reads the ICR1 I/O location it will access the TEMP Register.

The ICR1 register can only be written when using a Waveform Generation mode that utilizes the ICR1 Register for defining the counter's TOP value. In those cases the Waveform Generation mode (WGM1<3:0>) bits must be set before the TOP value can be written to the ICR1 Register. When writing the ICR1 register the high byte must be written to the ICR1H I/O location before the low byte is written to ICR1L.
Input Capture Trigger Source

The main trigger source for the input capture unit is the input Capture pin (ICP). Timer/Counter 1 can alternatively use the Analog Comparator output as trigger source for the input capture unit. The Analog Comparator is selected as trigger source by setting the Analog Comparator Input Capture (ACIC) bit in the Analog Comparator Control and Status Register (ACCSR). Be aware that changing trigger source can trigger a capture. The input capture flag must then be cleared after the change.

Both the Input Capture pin (ICP) and the Analog Comparator output (AOC) inputs are sampled using the same technique as for the T1 pin (Figure 59 on page 31). The edge detector is also identical. However, when the noise canceler is enabled, additional glitches is inserted before the edge detector, which increases the delay by four system clock cycles. Note that the input of the noise canceler and edge detector is always enabled unless the Timer/Counter 1 is set in a waveform generation mode that uses OCR1I to define TOP.

An input capture can be triggered by software by clearing the port of the ICP pin.

Noise Canceler

The noise canceler improves noise immunity by using a simple digital filtering scheme. The noise canceler input is monitored over four samples, and all four must be equal for charging the input that in turn is used by the edge detector.

The noise canceler is enabled by setting the input Capture Noise Canceler (ICNC) bit in Timer/Counter Control Register 1 (TCCR1B). When enabled, the noise canceler introduces additional four system clock cycles of delay for a change applied to the input to the update of the OCR1 Register. The noise canceler uses the system clock and is therefore unaffected by the prescaler.

Using the input Capture Unit

The main challenge when using the input capture unit is to assign enough processor capacity for handling the incoming events. The time between events is critical. If the processor has not read the captured value in the ICR1 Register before the next event occurs, the ICR1 will be overwritten with a new value. In this case the result of the capture will be incorrect.

When using the input capture interrupt, the ICR1 Register should be read as early as the interrupt handler routine as possible. Even though the input capture interrupt has relatively high priority, the maximum interrupt response time is dependent on the maximum number of clock cycles it takes to handle any of the other interrupt requests.

Using the input capture unit in any mode of operation where the TOP value (resolution) is actively changed during operation, is not recommended.

Measurement of an external signal's duty cycle requires that the trigger edge is changed after each capture. Changing the edge sensing must be done as early as possible after the ICR1 Register has been read. After a change of the edge, the input capture flag (ICF1) must be cleared by software (writing a logical one to the ICF1 bit location). For measuring frequency only, the clearing of the ICF1 flag is not required if an interrupt handler is used.

Output Compare Units

The 16-bit comparator continuously compares TCNT1 with the Output Compare Register (OCR1A). If TCNT1 equals OCR1A the comparator signals a match. A match will set the Output Compare Flag (OCF1A) at the next timer clock cycle. If enabled (OCIE1A = 1), the Output Compare Flag generates an output compare interrupt. The OCF1A flag is automatically cleared when the interrupt is executed. Alternatively the OCF1A flag can...
be cleared by software by writing a logical one to its I/O bit location. The Waveform Generator uses the match signal to generate an output according to operating modes set by the Waveform generation mode (WGM) bits and Compare Output mode (CS6/3/1) bits. The TOP and BOTTOM signals are used by the Waveform Generator for handling the special cases of the extreme values in some modes of operation (See "Modes of Operation" on page 94.)

A special feature of output compare unit A allows it to define the Timer/Counter TOP value (i.e., counter resolution). In addition to the counter resolution, the TOP value defines the period time for waveforms generated by the Waveform Generator.

Figure 43 shows a block diagram of the output compare unit. The small "i" in the register and bit names indicates the device number (i.e., i = 1 for Timer/Counter1), and the "x" indicates output compare unit (i.e., x = B). The elements of the block diagram that are not directly a part of the output compare unit are gray shaded.

Figure 43: Output Compare Unit, Block Diagram

The OCR1x Register is double buffered when using any of the twelve Pulse Width Modulation (PWM) modes. For the normal and Clear Timer on Compare (CTC) modes of operation, the double buffering is disabled. The double buffering synchronizes the update of the OCR1x Compare Register to either TOP or BOTTOM of the counting sequence. The synchronization prevents the occurrence of odd-length, non-symmetrical PWM pulses, thereby making the output glitch-free.

The OCR1x Register access may seem complex, but this is not case. When double buffering is enabled, the CPU has access to the OCR1x Buffer Register, and if double buffering is disabled the CPU will access the OCR1x directly. The content of the OCR1x (buffer or compare) register is only changed by a write operation (the Timer/Counter does not update the register automatically as the TCNT1 and OCR1 Register). Therefore OCR1x is touch-proof via the high byte temporary register (TEMP). However, it is a good practice to read the low byte first as when accessing other 16-bit registers. Writing the OCR1x Registers must be done via the TEMP Register since the compare of all 16 bits is done continuously. The high byte (OCR1h) has to be written first. When the high
byte I/O location is written by the CPU, the TEMP Register will be updated by the value written. Then when the low byte (COM1x1.0) is written to the lower eight bits, the high byte will be copied into the upper 8 bits of either the OCRxU buffer or OCRxU Compare Register, in the same system clock cycle.

For more information on how to access the 16-bit registers refer to "Accessing 16-Bit Registers" on page 96.

**Force Output Compare**

In non-PWM waveform generation modes, the match output of the comparator can be forced by writing a value to the Force Output Compare (FOCX) bit. Setting a compare match will not set the OCRxU flag or reinitialize the timer, but the OCX pin will be updated as if a real compare match had occurred (the CCMPn bits settings define whether the OCX pin is set, cleared, or toggled).

**Compare Match Blocking by TCNT1 Write**

All CPU writes to the TCNT1 Register will block any compare match that occurs in the next timer clock cycle, even when the timer is stopped. This feature allows OCRxU to be initialized to the same value as TCNT1 without triggering an interrupt when the Timer/Counter clock is enabled.

**Using the Output Compare Unit**

Since writing TCNT1 in any mode of operation will block all compare matches for one timer clock cycle, there are some issues when changing TCNT1 when using any of the output compare channels, independent of whether the Timer/Counter is running or not.

If the value written to TCNT1 equals the OCRxU value, the compare match will be missed, resulting in incorrect waveform generation. Do not write the TCNT1 equal to TOP in PWM modes with variable TOP Values. The compare match for the TOP will be ignored and the counter will continue to count. Similarly, do not write the TCNT1 value equal to BOTTOM when the counter is decrementing.

The setup of the OCRxU should be performed before setting the data direction register for the port pin to output. The easiest way of setting the OCRxU value is to use the force output compare (FOCX) strobe bits in normal mode. The OCRxU register keeps its value even when changing between waveform generation modes.

Be aware that the COM1x1.0 bits are not double buffered together with the compare value. Changing the COM1x1.0 bits will take effect immediately.

**Compare Match Output Unit**

The Compare Output mode (COM1x1.0) bits have two functions. The Waveform Generator uses the COM1x1.0 bits for defining the Output Compare (OCRxU) state at the next compare match. Secondly, the COM1x1.0 bits control the OCX pin output source. Figure 44 shows a simplified schematic of the logic affected by the COM1x1.0 bit setting. The I/O Registers, I/O bits, and I/O pins in the figure are shown in bold. Only the parts of the general I/O Port Control registers (DCR and PCRT) that are affected by the COM1x1.0 bits are shown. When referring to the OCX state, the reference is to the internal OCRxU register, not the OCX pin. If a System Reset occurs, the OCRxU Register is reset to '0'.

ATmega16(L)
The general I/O port function is overridden by the Output Compare (OCx) from the Waveform Generator if either of the COMxI:0 bits are set. However, the OCx pin direction (input or output) is still controlled by the Data Direction Register (DDR) for the port pin. The Data Direction Register bit for the OCx pin (DDR_OCx) must be set as output before the OCx value is visible on the pin. The port override function is generally independent of the Waveform Generation mode, but there are some exceptions. Refer to Table 44, Table 45 and Table 46 for details.

The design of the output compare pin logic allows initialization of the OCx line before the output is enabled. Note that some COMx:0 bit settings are reserved for certain modes of operation. See "Hitachi Timer/Counter Register Description" on page 104.

The COMx:0 bits have no affect on the input capture unit.

**Compare Output Mode and Waveform Generation**

The Waveform Generator uses the COMx:0 bits differently in normal, CTC, and PWM modes. For all modes, setting the COMx:0 = 1 tells the Waveform Generator that no action on the OCx Register is to be performed on the next compare match. For compare output actions in the non-PWM modes refer to Table 44 on page 104. For fast PWM mode refer to Table 45 on page 105, and for phase correct and phase and frequency correct PWM refer to Table 46 on page 105.

A change of the COMx:0 bits state will have affect at the first compare match after the bits are written. For non-PWM modes, the action can be delayed to have immediate effect by using the FOC0x strobe bits.

**Modes of Operation**

The mode of operation, i.e., the behavior of the Timer/Counter and the output compare pins, is defined by the combination of the Waveform Generation mode (WGMx:0) and Compare Output mode (COMx:0) bits. The Compare Output mode bits do not affect the counting sequence, while the Waveform Generation mode bits do. The COMx:0 bits control whether the PWM output generated should be inverted or not (inverted or non-inverted PWM). For non-PWM modes the COMx:0 bits control whether the output should be cleared, cleared or legged at a compare match (see "Compare Match Output Unit" on page 102).

For detailed timing information refer to "Timer/Counter Timing Diagrams" on page 102.
Normal Mode

The simplest mode of operation is the Normal mode (WGM12:0 = 0). In this mode the counting direction is always up (incrementing), and no counter clear is performed. The counter simply overruns when it passes its maximum 16-bit value (MAX = 0xFFFF) and then restarts from the BOTTOM (0x0000). In normal operation the TimerOverflow ErrorFlag (TOVF) will be set in the same timer clock cycle as the TOV1 becomes zero. The TOV1 flag in this case behaves like a Timer1, except that it is only set, not cleared. However, combined with the timer overflow interrupt that automatically clears the TOV1 flag, the timer resolution can be increased by software. There are no special cases to consider in the normal mode, a new counter value can be written anytime.

The input capture unit is easy to use in normal mode. However, observe that the maximum interval between the external events must not exceed the resolution of the counter. If the interval between events is too long, the timer overflow interrupt of the prescaler must be used to extend the resolution for the capture unit.

The output compare units can be used to generate interrupts at some given time. Using the output compare to generate waveforms in Normal mode is not recommended, since this will occupy too much of the CPU time.

Clear Timer on Compare Match (CTC) Mode

In Clear Timer on Compare or CTC mode (WGM12:0 = 4 or 12), the OCR1A or ICR1 Register are used to manipulate the counter resolution. In CTC mode, the counter is cleared to zero when the counter value (TCNT1) matches either the OCR1A (WGM12:0 = 4) or the ICR1 (WGM12:0 = 12). The OCR1A or ICR1 define the top value for the counter, twice also its resolution. This mode allows greater control of the compare match output frequency. It also simplifies the operation of counting external events.

The timing diagram for the CTC mode is shown in Figure 45. The counter value (TCNT1) increases until a compare match occurs with either OCR1A or ICR1, and then counter (TOV1) is cleared.

Figure 45. CTC Mode, Timing Diagram

An interrupt can be generated at each time the counter value reaches the TOP value by either using the OCR1A or ICR1 flag according to the register used to define the TOP value. If the interrupt is enabled, the interrupt service routine can be used to update the TOP value. However, changing the TOP to a value close to BOTTOM when the counter is running with none or a low prescaler value must be done with care since the CTC mode does not have the double buffering feature. If the new value written to OCR1A or ICR1 is lower than the current value of TCNT1, the counter will miss the compare match. The counter will then have to count to its maximum value (0xFFFF) and wrap around starting at 0x0000 before the compare match can occur. In many cases
this feature is not desirable. An alternative will then be to use the fast PWM mode using
OCR1A for defining TOP (WGM12:0 = 15) since the OCR1A then will be double buffered.

For generating a waveform output in CTC mode, the OCR1A output can be set to toggle
its logical level on each compare match by setting the compare output mode bits to logi-
tical mode (OCR1A:3 = 1). The OCR1A value will not be visible on the port pin unless the
data direction for the pin is set to output (DDR_OC1A = 1). The waveform generated will
have a maximum frequency of \( f_{OC1A} = \frac{f_{PPWM}}{2 \times N \times (1 + OCR1A)} \) when OCR1A is set to zero (0x0000). The
waveform frequency is defined by the following equation:

\[
\frac{f_{OC1A}}{f_{PPWM}} = \frac{1}{2 \times N \times (1 + OCR1A)}
\]

The \( N \) variable represents the prescaler factor (1, 64, 256, or 1024).

As for the normal mode of operation, the TOV1 flag is set in the same timer clock cycle
that the counter counts from MAX 0x00000000.

**Fast PWM Mode**

The Fast Pulse Width Modulation or Fast PWM mode (WGM12:0 = 6, 7, 14, or 15) pro-
vides a high-frequency PWM waveform generation option. The fast PWM differs from
the other PWM options by its single-slope operation. The counter counts from BOTTOM
to TOP then restarts from BOTTOM. In non-inverting Compare Output mode, the output
from Compare (OCR1A) is set on the compare match between TCNT1 and OCR1A, and
cleared at TOP. In inverting Compare Output mode, output is cleared on compare match
and set at TOP. Due to the single-slope operation, the operating frequency of the fast
PWM mode can be twice as high as the phase and frequency control PWM modes that use
dual-slope operation. This high frequency makes the fast PWM mode well suited for
power regulation, rectification, and DAC applications. High frequency allows physically
small sized external components (capacitors, inductors), hence reduces total system cost.

The PWM resolution for fast PWM can be fixed to 8, 9, or 10-bit, or defined by either
ICR1 or OCR1A. The minimum resolution allowed is 2-bit (ICR1 or OCR1A set to 0x0000)
and the maximum resolution is 15-bit (ICR1 or OCR1A set to MAX). The PWM
resolution in bits can be calculated by using the following equation:

\[
R_{PWM} = \frac{2^{TOP} - 1}{2^2}
\]

In fast PWM mode the counter is incremented until the counter value matches either
one of the fixed values 0x00FF, 0x00FF, or 0x00FF (WGM12:0 = 5, 6, or 7), the value in
ICR1 (WGM12:0 = 14), or the value in OCR1A (WGM12:0 = 15). The counter is then
cleared at the following timer clock cycle. The timing diagram for the fast PWM mode is
shown in Figure 45. The figure shows fast PWM mode when OCR1A or ICR1 is used to
define TOP. The TCNT1 value is in the timing diagram shown as a histogram for illus-
trating the single-slope operation. The diagram includes non-inverted and inverted PWM
outputs. The small horizontal line marks on the TCNT1 slopes represent compare
matches between OCR1A and TCNT1. The OC1A latching flag will be set when a com-
parematch occurs.
The Timer/Counter Overflow Flag (TOV1) is set each time the counter reaches TOP. In addition the OCF1A or OCF1B flag is set at the same timer clock cycle as TOV1 is set when either OCR1A or OCR1B is used for defining the TOP value. If one of the interrupts is enabled, the interrupt handler routine can be used for updating the TOP and compare values.

When changing the TOP value the program must ensure that the new TOP value is higher or equal to the value of all of the compare registers. If the TOP value is lower than any of the compare registers, a compare match will never occur between the TCNT1 and the OCR1x. Note that when using fixed TOP values the unused bits are masked to zero when any of the OCRx registers are written.

The procedure for updating ICR1 differs from updating OCR1x when used for defining the TOP value. The ICR1 Register is not double buffered. This means that if ICR1 is changed to a low value when the counter is running with none or a low prescaler value, there is a risk that the new ICR1 value written is lower than the current value of TCNT1. The result will then be that the counter will miss the compare match at the TOP value. The counter will then have to count to the MAX value (0xFFFF) and wrap around starting at 0x0000 before the compare match can occur. The OCR1A Register however, is double buffered. This feature allows the OCR1A I/O location to be written anytime. When the OCR1A I/O location is written, the value written will be put into the OCR1A Buffer Register. The OCR1A Compare Register will then be updated with the value in the buffer register at the next timer clock cycle the TCNT1 matches TOP. The update is done at the same timer clock cycle as the TOV1 is cleared and the TOV1 flag is set.

Using the ICR1 Register for defining TOP works well when using fixed TOP values. By using ICR1, the OCR1x Register is free to be used for generating a PWM output on OCA1. However, if the base PWM frequency is actively changed by changing the TOP value, using the OCR1x as TOP is clearly a better choice due to its double buffer feature.

In free PWM mode, the compare units allow generation of PWM waveforms on the OCR1x pins. Setting the COM1a1:0 bits to 2 will produce an non-inverted PWM and an inverted PWM output can be generated by setting the COM1a1:0 bits to 3 (See Table 44 on page 104). The actual OCR1x value will only be visible on the port pin if the direction of the port pins set as output (DDRA:DDRx). The PWM waveform is generated by...
setting or clearing the OCR1x Register at the compare match between OCR1x and TCNT1, and clearing or setting the OCR1x Register at the timer clock cycle the counter is cleared (changes from TOP to BOTTOM).

The PWM frequency for the output can be calculated by the following equation:

\[ f_{OC} = \frac{f_{OCx}}{2^{N} \cdot (1 + TOP)} \]

The N variable represents the prescaler divides (1, 0, 54, 255, or 1024).

The extreme values for the OCR1x Register represents special cases when generating a PWM waveform output in the fast PWM mode. If the OCR1x is set equal to BOTTOM (0x0000) the output will be a narrow spike for each TOP+1 timer clock cycle. Setting the OCR1x to TOP will result in a constant high or low output depending on the polarity of the output output by the COMxM0:1 bits.

A frequency with 50% duty cycle waveform output in fast PWM mode can be achieved by setting OCR1x to toggle its logic level on each compare match (COMxM0:1 = 1). This option only if OCR1x is used to define the TOP value (WGM1:0 = 15). The waveform generated will have a maximum frequency of \( f_{OC} = \frac{f_{OCx}}{2^{N} \cdot 1} \) when OCR1x is set to zero (0x0000). This feature is similar to the OCRx toggle in CTC mode, except the double buffer feature of the output compare unit is enabled in the fast PWM mode.

**Phase Correct PWM Mode**

The phase correct PWM mode (WGM1:0 = 11) provides a high resolution phase correct PWM waveform generation option. The phase correct PWM mode is, like the phase and frequency correct PWM mode, based on a dual-slope operation. The counter counts repeatedly from BOTTOM (0x0000) to TOP and then from TOP to BOTTOM. In non-inverting Compare Output mode, the Output Compare (OCx) is cleared on the compare match between TCNT1 and OCRx while counting up, and set on the compare match while counting down. In inverting output compare mode, the opposite is inverted. The dual-slope operation has lower maximum operation frequency than single-slope operation. However, due to the symmetrical feature of the dual-slope PWM mode, these modes are preferred for motor control applications.

The PWM resolution in the phase correct PWM mode can be set to 8-, 10-, or 12-bit, or defined by either CTC1 or OCR1A. The minimum resolution allowed is 8-bit (OCR1 or OCR1A set to 0x1000), and the maximum resolution is 16-bit (OCR1 or OCR1A set to MAX). The PWM resolution in bits can be calculated by using the following equation:

\[ P_{PWM} = \frac{\log(TOP + 1)}{\log(2)} \]

In phase correct PWM mode the counter is incremented until the counter matches either one of the fixed values 0x00FF, 0x01FF, or 0x02FF (WGM1:0 = 1, 2, or 3), the value in OCR1 (WGM1:0 = 10), or the value in OCR1A (WGM1:0 = 11). The counter has then reached the TOP and changes the count direction. The TCNT1 value will be equal to TOP for one timer clock cycle. The timing diagram for the phase correct PWM mode is shown on Figure 47. The figure shows phases correct PWM mode when OCR1A or OCR1A is used to define TOP. The TCNT1 value is in the timing diagram shown as a histogram for illustrating the dual-slope operation. The diagram includes non-inverted and inverted PWM outputs. The smallest horizontal line marks on the TCNT1 slope represent compare matches between OCRx and TCNT1. The OCRx interrupt flag will be set when a compare match occurs.

**ATmega16(L)**

B-27
Figure 47. Phase Correct PWM Mode Timing Diagram

The Timer/Counter Overflow Flag (TOV1) is set each time the counter reaches BOTTOM. When either OCR1A or OCR1B is used for defining the TOP value, the OCR1A or OCR1B flag is set accordingly at the same timer clock cycle as the OCR1x Registers are updated with the double buffer value (at TOP). The interrupt flags can be used to generate an interrupt each time the counter reaches the TOP or BOTTOM value.

When changing the TOP value the program must ensure that the new TOP value is higher or equal to the value of all of the compare registers. If the TOP value is lower than any of the compare registers, a compare match will never occur between the TCNT1 and the OCR1x. Note that when using fixed TOP values, the unused bits are pinned to zero when any of the OCR1x Registers are written. As the third period shown in Figure 47 illustrates, changing the TOP activity while the Timer/Counter is running in the phase correct mode can result in an asymmetrical output. The reason for this can be found in the time of update of the OCR1x Register. Since the OCR1x update occurs at TOP, the PWM period starts and ends at TOP. This implies that the length of the falling slope is determined by the previous TOP value, while the length of the rising slope is determined by the new TOP value. When these two values differ the two slopes of the period will differ in length. The difference in length gives the asymmetrical result on the output.

It is recommended to use the phase and frequency correct mode instead of the phase correct mode when changing the TOP value while the Timer/Counter is running. When using a static TOP value these are practically no differences between the two modes of operation.

In phase correct PWM mode, the compare units allow generation of PWM waveforms on the OC1x pins. Setting the COM1x1:COM1x0 bits to 2 will produce a non-inverted PWM output, while a non-inverted PWM output can be generated by setting the COM1x1:COM1x0 bits to 3 (see Table 41 on page 104). The actual OCRx value will only be visible on the port pin if the data direction for the port pin is set as output (DDR_OC1x). The PWM waveform is generated by setting (or clearing) the OCR1x Register at the compare match between TCNT1 and TCNT1 when the counter increments, and clearing (or setting) the OCR1x Register at compare match between OCR1x and TCNT1 when the counter decrements. The PWM frequency

---

The text above is a detailed explanation of the phase correct PWM mode in the context of a timer/counter. It describes the behavior of the Timer/Counter Overflow Flag (TOV1) and how the OCR1x Registers are updated with values from the double buffer. It also explains the importance of having the new TOP value higher or equal to the values of all compare registers, and the potential for asymmetrical output when changing the TOP value while the counter is running. The section on phase correct PWM mode provides guidance on how to generate PWM waveforms on the OC1x pins, including the effects of different COM1x1:COM1x0 bit settings. It concludes by describing the PWM waveform generation process in phase correct PWM mode.
for the output when using phase correct PWM can be calculated by the following equation:

\[ f_{OCRI\text{correct}} = \frac{f_{ocritop}}{2^{-N-TOP}} \]

The \( N \) variable represents the prescaler divider (1, 8, 64, 256, or 1024).

The extreme values for the OCRIx register represent special cases when generating a PWM waveform output in the phase correct PWM mode. If the OCRx is set equal to BOTTOM the output will be continuous low and if set equal to TOP the output will be continuously high for non-inverted PWM mode. For inverted PWM the output will have the opposite logic values. If OCRx is used to define the TOP value (WGM13:0 = 11) and OCRx1:0 = 11, the OCRI output will toggle with a 50% duty cycle.

**Phase and Frequency Correct PWM Mode**

The phase and frequency correct PWM (WGM13:0 = 8 or 9) provides a high resolution phase and frequency correct PWM waveform generation option. The phase and frequency correct PWM mode is, like the phase correct PWM mode, based on a dual-slope operation. The counter counts repeatedly from BOTTOM (0x0000) to TOP and then from TOP to BOTTOM. In non-inverting Compare Output mode, the Output Compare (OC1x) is cleared on the compare match between TOCN1 and OCR1x while counting, and set on the compare match while decreasning. In inverting Compare Output mode, the operation is inverted. The dual-slope operation gives a lower maximum operation frequency compared to the single-slope operation. However, due to the symmetric nature of the dual-slope PWM mode, these modes are preferred for motor control applications.

The main difference between the phase correct and the phase and frequency correct PWM modes is that the OCRIx register is updated by the OCR1x buffer register, (see Figure 47 and Figure 48).

The PWM resolution for the phase and frequency correct PWM mode can be defined by either IC1x or OCRIx. The minimum resolution allowed is 2x4 (IC1x or OCRIx set to 0x0000), and the maximum resolution is 2x16 (IC1x or OCRIx set to MAX). The PWM resolution in bits can be calculated using the following equation:

\[ R_{PWM} = \frac{\log(TOP+1)}{\log(2)} \]

In phase and frequency correct PWM mode the counter is incremented until the counter value matches either the value in ICRI (WGM13:0 = 8), or the value in OCR1x (WGM13:0 = 9). The counter has then reached the TOP and changes the count direction. The TCNT1 value will be equal to TOP for one timer clock cycle. The timing diagram for the phase correct and frequency correct PWM mode is shown on figure 48. The figure shows phase and frequency correct PWM mode when OCR1x or IC1x is used to define TOP. The TCNT1 value is in the timing diagram shown as a histogram for illustrating the dual-slope operation. The diagram includes non-inverted and inverted PWM outputs. The small horizontal lines on the TCON1 slopes represent compare matches between OCR1x and TCNT1. The OC1x interrupt flag will be set when a compare match occurs.
The Timer/Counter Overflow Flag (TOV1) is set at the same timer clock cycle as the OCR1x registers are updated with the double buffer value (at BOTTOM). When either OCR1A or OCR1B is used for defining the TOP value, the OCF1A or OCF1B flag is set when TCNT1 has reached TOP. The interrupt flags can then be used to generate an interrupt each time the counter reaches the TOP or BOTTOM value.

When changing the TOP value the program must ensure that the new TOP value is higher or equal to the value of all of the compare registers. If the TOP value is lower than any of the compare registers, a compare match will never occur between the TCNT1 and the OCR1x.

As Figure 48 shows the output generated is, in contrast to the phase correct mode, symmetrical in all periods. Since the OCR1x Registers are updated at BOTTOM, the length of the rising and the falling slopes will always be equal. This gives symmetrical output pulses and is therefore frequency correct.

Using the ICR1 Register for defining TOP works well when using fixed TOP values. By using ICR1, the OCR1A Register is free to be used for generating a PWM output on OC1A. However, if the base PWM frequency is actively changed by changing the TOP value, using the OCR1A as TOP is clearly a better choice due to its double buffer feature.

In phase and frequency correct PWM mode, the compare units allow generation of PWM waveforms on the OCR1x pins. Setting the COM1x1 to 1 will produce a non-inverted PWM and an inverted PWM output can be generated by setting the COM1x1 to 0 (see Table 1 on page 106). The actual OCRx value will only be visible on the port pin if the data direction for the port pin is set as output (DDRB, OCRx). The PWM waveform is generated by setting (or clearing) the OCR1x Register at the compare match between OCR1x and TCNT1 when the counter increments, and clearing (or setting) the OCR1x Register at compare match between OCR1x and TCNT1 when the counter decrements. The PWM frequency for the output when using phase and frequency correct PWM can be calculated by the following equation:

$$f_{PWM} = \frac{\text{fosc}}{2 \cdot N \cdot \text{TOP}}$$

The N variable represents the prescaler divider (1, 8, 64, 256, or 1024).
The extreme values for the OCR1x Register represent special cases when generating a PWM waveform output in the phase correct PWM mode. If the OCR1x is set equal to BOTTOM the output will be continuously low and if set equal to TOP the output will be set to high for non-inverted PWM mode. For inverted PWM the output will have the opposite logic values. If OCR1A is used to define the TOP value (WGM13:0 = 9) and COM1A1:0 = 1, the OC1A output will toggle within a 50% duty cycle.

Timer/Counter Timing Diagrams

The Timer/Counter is a synchronous design and the timer clock (ckT) is therefore shown as a clock enable signal in the following figures. The figures include information on when interrupt flags are set, and when the OCR1x Register is updated with the OCR1x buffer value (only for modes utilizing double buffering). Figure 49 shows a timing diagram for the setting of OCR1x.

Figure 49, Timer/Counter Timing Diagram, Setting of OCR1x, No Prescaler

| ckT | ckT (clkx)
|-----|-----------
| TONTh | OCR1x - 1 | OCR1x | OCR1x + 1 | OCR1x + 2 |
| OCR1x | | | | | |
| OCF1x | | | | | |

Figure 50 shows the same timing data, but with the prescaler enabled.

Figure 50, Timer/Counter Timing Diagram, Setting of OCR1x, with Prescaler (fosc/4x8)

| ckT | ckT (clkx)
|-----|-----------
| TONTh | OCR1x - 1 | OCR1x | OCR1x + 1 | OCR1x + 2 |
| OCR1x | | | | | |
| OCF1x | | | | | |

ATmega16(L)
Figure 51 shows the count sequence close to TOP in various modes. When testing phase and frequency correct PWM mode, the OCR1a Register is updated at BOTTOM. The timing diagrams will be the same, but TOP should be replaced by BOTTOM, TOP - 1 by BOTTOM + 1 and so on. The same reasoning applies for modes that set the TOV1 flag at BOTTOM.

Figure 51. Timer/Counter Timing Diagram, no Prescaling

Figure 52 shows the same timing data, but with the prescaler enabled.

Figure 52. Timer/Counter Timing Diagram, with Prescaler (f Tamb/256)
16-bit Timer/Counter
Register Description

Timer/Countert1 Control
Register A – TCCRA1

<table>
<thead>
<tr>
<th>Bit</th>
<th>7</th>
<th>6</th>
<th>5</th>
<th>4</th>
<th>3</th>
<th>2</th>
<th>1</th>
<th>0</th>
</tr>
</thead>
<tbody>
<tr>
<td>Value</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
</tbody>
</table>

- Bit 7:6 – COM1A1:0: Compare Output Mode for Channel A
- Bit 5:4 – COM1B1:0: Compare Output Mode for Channel B

The COM1A1:0 and COM1B1:0 control the Output Compare pins (OC1A and OC1B, respectively) behavior. If one or both of the COM1A1:0 bits are written to one, the OC1A output overrides the normal port functionality of the I/O pin it is connected to. If one or both of the COM1B1:0 bits are written to one, the OC1B output overrides the normal port functionality of the I/O pin it is connected to. However, note that the Data Direction Register (DDR) bit corresponding to the OC1A or OC1B pin must be set in order to enable the output driver.

When the OC1A or OC1B is connected to the pin, the function of the COM1X1:0 bits is dependent of the WGM13:0 bits setting. Table 44 shows the COM1X1:0 bit functionality when the WGM13:0 bits are set to a normal or a CTC mode (non-PWM).

Table 44. Compare Output Mode, non-PWM

<table>
<thead>
<tr>
<th>COM1A1:COM1B1</th>
<th>COM1A0:COM1B0</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>0</td>
<td>Normal port operation, OC1A/OC1B disconnected.</td>
</tr>
<tr>
<td>0</td>
<td>1</td>
<td>Toggle OC1A/OC1B on compare match</td>
</tr>
<tr>
<td>1</td>
<td>0</td>
<td>Clear OC1A/OC1B on compare match (set output to low level)</td>
</tr>
<tr>
<td>1</td>
<td>1</td>
<td>Set OC1A/OC1B on compare match (set output to high level)</td>
</tr>
</tbody>
</table>

Table 45 shows the COM1X1:0 bit functionality when the WGM13:0 bits are set to the fast PWM mode.
8-bit Timer/Counter2 with PWM and Asynchronous Operation

Timer/Counter2 is a general purpose, single channel, 8-bit Timer/Counter module. The main features are:
- Single Channel Counter
- Clear Timer On Compare Match (Auto Reload)
- Glitch-free Phase Correct Pulse Width Modulator (PWM)
- Frequency Generator
- 16-bit Clock Prescaler
- Overflow and Compare Match Interrupt Sources (TOV and OCF2)
- Allows clocking from External 32 kHz Watch Crystal independent of the I/O Clock

Overview

A simplified block diagram of the 8-bit Timer/Counter is shown in Figure 53. For the actual placement of I/O pins, refer to “Pinouts ATmega16(L)” on page 2. CPR accessible I/O Registers, including I/O bits and I/O pins, are shown in bold. The device-specific I/O register and bit locations are listed in the “8-bit Timer/Counter Register Description” on page 121.

Figure 53. 8-bit Timer/Counter Block Diagram

Registers

The Timer/Counter (TCNT2) and Output Compare Register (OCR2) are 8-bit registers. Interrupt request signals, as well as Int.Req. signals are all visible in the Timer Interrupt Flag Register (TIFR). All interrupts are individually masked with the Timer Interrupt Mask Register (TIMSK). TIFR and TIMSK are not shown in the figure since these registers are shared by other timer units.

The Timer/Counter can be clocked internally via the prescaler, or asynchronously clocked from the TCS01/2 pins, as detailed later in this section. The asynchronous operation is controlled by the Asynchronous Status Register (ASSR). The Clock Select logic block controls which clock source the Timer/Counter uses to increment or decrement its value. The Timer/Counter is inactive when no clock source is selected. The output from the Clock Select logic is selected as the timer clock (TCK).
The double buffered Output Compare Register (OCR2) is compared with the Timer/Counter value at all times. The result of the compare can be used by the waveform generator to generate a PWM or variable frequency output on the Output Compare Pin (OC2). See "Output Compare Unit" on page 116 for details. The compare match event will also set the Compare Flag (CCIF) which can be used to generate an output compare interrupt request.

Definitions

Many register and bit references in this document are written in general form. A lower case "n" replaces the Timer/Counter number, in this case 2. However, when using the register or bit defines in a program, the precise form must be used (i.e., TCNT2 for accessing Timer/Counter 2 counter value and so on). The definitions in Table 49 are also used extensively throughout the document.

| BOTTOM | The counter reaches the BOTTOM when it becomes zero (000).
| MAX | The counter reaches its MAXimum when it becomes 0xFF (decimal 255).
| TOP | The counter reaches the TOP when it becomes equal to the highest value in the counter sequence. The TOP value can be assigned to be the fixed value 0xFF (MAX) or the value stored in the OCR2 register. The assignment is dependent on the mode of operation.

Timer/Counter Clock Sources

The Timer/Counter can be clocked by an internal synchronous or an external asynchronous clock source. The clock source, clkosc, is by default equal to the MCU clock, clkint. When the AS2 bit in the ASR Register is written to logic one, the clock source is taken from the Timer/Counter oscillator connected to TOSS1 and TOSS2. For details on asynchronous operation, see "Asynchronous Status Register – ASR" on page 124. For details on clock sources and prescaler, see "Timer/Counter Prescaler" on page 127.

Counter Unit

The main part of the 8-bit Timer/Counter is the programmable bidirectional counter unit. Figure 50 shows a block diagram of the counter and its surrounding environment.

Signal description (internal signals):
- count: Increment or decrement TCNT2 by 1.
- direction: Selects between increment and decrement.
- clear: Clears TCNT2 (set all bits to zero).
- clk2: Timer/Counter clock.

ATmega16(L)

B-35
top: Signals that TCNT2 has reached maximum value.
bottom: Signals that TCNT2 has reached minimum value (zero).

Depending on the mode of operation used, the counter is cleared, incremented, or decremented at each timer clock (clk2). clk2 can be generated from an external or internal clock source, selected by the Clock Select bits (CS22:0). When no clock source is selected (CS22:0 = 0) the timer is stopped. However, the TCNT2 value can be accessed by the CPU, regardless of whether clk2 is present or not. A CPU write overwrites (has higher priority over) all counter clear or counter operations.

The counting sequence is determined by the setting of the WGM21 and WGM20 bits located in the Timer/Counter Control Register (TCCR2). There are close connections between how the counter behaves (counts) and how waveforms are generated on the Output Compare output OCR2. For more details about advanced counting sequences and waveform generation, see "Modes of Operation" on page 115.

The Timer/Counter Overflow (TOV2) flag is set according to the mode of operation selected by the WGM21:0 bits. TOV2 can be used for generating a CPU interrupt.

Output Compare Unit

The 8-bit comparator continuously compares TCCR2 with the output compare register (OCR2). Whenever TCCR2 equals OCR2, the comparator signals a match. A match will set the Output Compare Flag (OCF2) at the next timer clock cycle. If enabled (OCIE2 = 1), the output compare flag generates an output compare interrupt. The OCF2 flag is automatically cleared when the interrupt is acknowledged. Alternatively, the OCF2 flag can be cleared by software by writing a logical one to its I/O bit location. The waveform generator uses the match signal to generate an output according to operating mode set by the WGM21:0 bits and Compare Output mode (COM21:0) bits. The match and output signals are used by the waveform generator for handling the special cases of the extreme values in some modes of operation ("Modes of Operation" on page 115). Figure 55 shows a block diagram of the output compare unit.

Figure 55. Output Compare Unit, Block Diagram
The OCR2 Register is double buffered when using any of the Pulse Width Modulation (PWM) modes. For the normal and Clear Timer on Compare (CTC) modes of operation, the double buffering is disabled. The double buffering synchronizes the update of the OCR2 Compare Register to either top or bottom of the counting sequence. The synchronization prevents the occurrence of odd-length, non-symmetrical PWM pulses, thereby nulling the output glitch-free.

The OCR2 Register access may seem complex, but this is not the case. When the double buffering is enabled, the CPU has access to the OCR2 Buffer Register, and if double buffering is disabled the CPU will access the OCR2 directly.

**Force Output Compare**

In non-PWM waveform generation modes, the match output of the comparator can be forced by writing a one to the Force Output Compare (FOC2) bit. Forcing compare match will not set the OCR2 flag or reload/decrement the timer, but the OCF2 pin will be updated as if a real compare match had occurred (the OCF2 bits settings define whether the OCR2 pin is set, cleared or toggled).

**Compare Match Blocking by TCNT2 with**

All CPU write operations to the TCNT2 Register will block any compare match that occurs in the next timer clock cycle, even when the timer is stopped. This feature allows OCR2 to be initialized to the same value as TCNT2 without triggering an interrupt when the Timer/Counter is enabled.

**Using the Output Compare Unit**

Since setting TCNT2 in any mode of operation will block all compare matches for one timer clock cycle, there are risks involved when changing TCNT2 when using the output compare channel, independently of whether the Timer/Counter is running or not. If the value written to TCNT2 equals the OCR2 value, the compare match will be missed, resulting in incorrect waveform generation. Similarly, do not write the TCNT2 value equal to BOTTOM when the counter is counting.

The setup of the OCR2 should be performed before setting the Data Direction Register for the OCF2 pin to output. The easiest way of setting the OCR2 value is to use the Output Compare (FOC2) strobe bit in Normal mode. The OCR2 Register keeps its value even when changing between Waveform Generation modes.

Be aware that the COM2:0 bits are not double buffered together with the compare value. Changing the COM2:0 bits will take effect immediately.

**Compare Match Output Unit**

The Compare Output mode (COM2:0) bits have two functions. The Waveform Generator uses the COM2:1 bits for defining the Output Compare (OCP) state of the next compare match. Also, the COM2:0 bits control the OCF2 pin output source. Figure 56 shows a simplified schematic of the logic affected by the COM2:1 bit setting. The I/O Registers, I0 bits, and I1 pins in the figure are shown in bold. Only the parts of the general I/O port configuration registers (PORTD and PORTI) that are affected by the COM2:0 bits are shown. When referring to the OCP state, the references to the internal OCR2 Register, not the OCF2 pin.

---

**ATmega16(L)**

---

B-37
The general I/O port function is overridden by the Output Compare (OC2) from the waveform generator if either of the COM21:0 bits are set. However, the OC2 pin direction (input or output) is still controlled by the Data Direction Register (DDR) for the port pin. The Data Direction Register bit for the OC2 pin (DDR_OC2) must be set as output before the OC2 value is visible on the pin. The port override function is independent of the Waveform Generation mode.

The design of the output compare pin logic allows initialization of the OC2 state before the output is enabled. Note that some COM21:0 bit settings are reserved for certain modes of operation. See “8-bit Timer/Counter Register Description” on page 121.

**Compare Output Mode and Waveform Generation**

The waveform generator uses the COM21:0 bits differently in Normal, OTfO, and PWM modes. For all modes, setting the COM21:0 = 0 tells the Waveform Generator that no action on the OC2 Register is to be performed on the next compare match. For compare output actions in the non-PWM modes refer to Table 51 on page 122. For fast PWM mode refer to Table 52 on page 122, and for phase correct PWM refer to Table 53 on page 123.

A change of the COM21:0 bits state will have effect at the first compare match after the bits are written. For non-PWM modes, the action can be forced to have immediate effect by using the TOC2 strobe bits.

**Modes of Operation**

The mode of operation, i.e., the behavior of the Timer/Counter and the output compare pins, is defined by the combination of the Waveform Generation mode (WGM2:0) and Compare Output mode (COM21:0) bits. The Compare Output mode bits do not affect the counting sequence, while the Waveform Generation mode bits do. The COM21:0 bits control whether the PWM output generated should be inverted or not (inverted or non-inverted PWM). For non-PWM modes the COM21:0 bits control whether the output should be set, cleared, or toggled at a compare match. See “Compare Match Output Unit” on page 114.

For detailed timing information refer to “Timer/Counter Timing Diagrams” on page 119.

**Normal Mode**

The simplest mode of operation is the Normal mode (WGM2:0 = 0). In this mode the counting direction is always up (not resetting), and no counter clear is performed. The counter simply overflows when it passes its maximum 8-bit value (TOP = 255) and then

---

**Figure 54. Compare Match Output Unit, Schematic**

The schematic shows the internal components and connections for the compare match output unit, including the waveform generator and related circuitry.
Starts from the bottom (0x00). In normal operation the Timer/Counter Overflow Flag (TOV2) will be set in the same timer clock cycle as the TOINT2 becomes zero. The TOV2 flag in this case behaves like a spin bit, except that it is not set, but cleared. However, combined with the timer overflow interrupt that automatically clears the TOV2 flag, the timer resolution can be increased by software. There are no special cases to consider in this normal mode, a new counter value can be written anytime.

The Output Compare unit can be used to generate interrupts at some given time. Using the output compare to generate waveforms in normal mode is not recommended, since this will occupy too much of the CPU time.

In Clear Timer on Compare or CTC mode (WGM21:0 = 2), the OCR2 register is used to manipulate the counter resolution. In CTC mode the counter is cleared to zero when the counter value (TCNT2) matches the OCR2. The OCR2 defines the top value for the counter, hence also its resolution. This mode allows greater control of the compare match output frequency. It also simplifies the operation of counting external events.

The timing diagram for the CTC mode is shown in Figure 57. The counter value (TCNT2) increases until a compare match occurs between TCNT2 and OCR2, and then the counter (TCNT2) is cleared.

Figure 57. CTC Mode, Timing Diagram

An interrupt can be generated each time the counter value reaches the TOP value by using the OCR2 flag. If the interrupt is enabled, the interrupt handler routine can be used for updating the TOP value. However, changing the TOP to a value close to BOTTOM when the counter is running with none or a low prescaler value must be done with care since the CTC mode does not have the double buffering feature. The new value written to OCR2 is lower than the current value of TCNT2, the counter will reset the compare match. The counter will then have to count to its maximum value (0xFF) and wrap around starting at zero before the compare match can occur.

For generating a waveform output in CTC mode, the OCR2 output can be set to toggle its logical level each time compare match occurs by setting the Compare Output mode bits toToggle mode (COM2A1:0 = 1). The OCR2 value will not be visible on the port pin unless the data direction for the pin is set to output. The waveform generated will have a maximum frequency of fOC2 = f_cpu / 2 when OCR2 is set to zero (0x00). The waveform frequency is defined by the following equation:

\[ f_{OC2} = \frac{f_{CPU}}{2 \cdot N \cdot (1 + OCR2)} \]
The N variable represents the prescale factor (1, 8, 16, 32, 64, 128, 256, or 1024).

As for the normal mode of operation, the TOV2 flag is set in the same timer clock cycle that the counter values from MAX to 0.

### Fast PWM Mode

The fast Pulse Width Modulation (PWM) mode (WGM21:0 = 3) provides a high-frequency PWM waveform generation option. The fast PWM differs from the other PWM options by its single-slope operation. The counter counts from BOTTOM to MAX then resets from BOTTOM. In non-inverting Compare Output mode, the Output Compare (OC2) is cleared on the compare match between TCNT2 and OCR2, and set at BOTTOM. In inverting Compare Output mode, the output is set on compare match and cleared at BOTTOM. Due to the single-slope operation, the operating frequency of the fast PWM mode can be twice as high as the phase correct PWM mode that uses dual-slope operation. This high frequency makes the fast PWM mode well suited for power regulation, rectification, and DAC applications. High frequency allows physically small sized external components (capacitors) and therefore reduces total system cost.

In fast PWM mode, the counter is incremented until the counter value matches the MAX value. The counter is then cleared at the following timer clock cycle. The timing diagram for the fast PWM mode is shown in Figure 58. The TOVF2 flag is set in the timing diagram shown as a histogram for illustrating the single-slope operation. The diagram includes non-inverted and inverted PWM outputs. The small horizontal line marks on the TCNT2 slope represent compare matches between OCR2 and TCNT2.

**Figure 58.** Fast PWM Mode, Timing Diagram

The Timer/Counter Overflow Flag (TOV2) is set each time the counter reaches MAX. If the interrupt is enabled, the interrupt handler routines can be used for updating the compare value.

In fast PWM mode, the compare unit allows generation of PWM waveforms on the OC2 pin. Setting the COM21:0 bits to 11 will produce a non-inverted PWM and an inverted PWM output can be generated by setting the COM21:0 bits to 0 (see Table 52 on page 120). The actual OC2 value will only be visible on the port pin if the data direction for the port pin is set as output. The PWM waveform is generated by setting (or clearing) the OC2 Register at the compare match between OCR2 and TCNT2 and clearing (or setting) the OC2 Register at the timer clock cycle the counter is cleared (changes from MAX to BOTTOM).
The PWM frequency for the output can be calculated by the following equation:

\[ f_{\text{PWM}} = \frac{f_{\text{clk} \_\text{io}}}{\text{N}} \]

The \( N \) variable represents the prescaler factor (1, 8, 32, 64, 128, 256, or 1024).

The extreme values for the OC2 compare mode represent special cases when generating a PWM waveform output in the fast PWM mode. If the OC2 is set equal to BOTTOM, the output will be a narrow spike for each MAX1 timer clock cycle. Setting the OC2 equal to MAX will result in a constantly high or low output (depending on the polarity of the output set by the COM2<1:0> bits).

A frequency (with 50% duty cycle) waveform output in fast PWM mode can be achieved by setting OC2 to toggle its logical level on each compare match (COM2<1:0> = 1). The waveform generated will have a maximum frequency of \( f_{\text{PWM}} = \frac{f_{\text{clk} \_\text{io}}}{8} \) when OC2 is set to zero. This feature is similar to the OC2 toggle in CTC mode, except the double-buffer feature of the output compare unit is enabled in the fast PWM mode.

### Phase Correct PWM Mode

The phase correct PWM mode (WGM2<1:0> = 1) provides a high-resolution phase correct PWM waveform generation option. The phase correct PWM mode is based on a dual-stripe operation. The counter counts repeatedly from BOTTOM to MAX and then from MAX to BOTTOM. In non-inverting Compare Output mode, the Output Compare (OC2) is driven in the compare match between TOCII and OC2 while ascending, and set on the compare match while descending. In inverting Output Compare mode, the operation is inverted. The dual-stripe operation increases the maximum operation frequency versus single strobe operation. However, due to the symmetric feature of the dual-stripe PWM modes, these modes are preferred for motor control applications.

The PWM resolution for the phase correct PWM mode is fixed to 8 bits. In phase correct PWM mode, the counter is incremented until the counter value matches MAX. When the counter reaches MAX, it changes the counter direction. The TOCII value will be equal to MAX for one timer clock cycle. The timing diagram for the phase correct PWM mode is shown on Figure 9f. The TOCII value is in the timing diagram shown as a histogram for illustrating the dual-stripe operation. The diagram includes non-inverted and inverted PWM outputs. The small horizontal line marks on the TOCII2 slopes represent compare matches between OC2 and TOCII2.
The Timer/Counter Overflow Flag (TOF) is set each time the counter reaches BOTTOM. The Interrupt flag can be used to generate an interrupt each time the counter reaches the BOTTOM value.

In phase correct PWM mode, the compare match generates a PWM waveform on the OC2 pin. Setting the COM2<1:0> bits to 2 will produce a non-inverted PWM. An inverted PWM output can be generated by setting the COM2<1:0> to 3 (see Table 56 on page 123). The actual OC2 value will only be visible on the port pin if the data direction for the port pin is set as output. The PWM waveform is generated by clearing (or setting) the OC2 Register at the compare match between OCR2 and TCCR2 when the counter increments, and setting (or clearing) the OCR2 Register at compare match between OCR2 and TCCR2 when the counter decrements. The PWM frequency for the output waveform phase correct PWM can be calculated by the following equation:

\[ f_{PWM} = \frac{f_{osc}}{N \cdot 10} \]

The \( N \) variable represents the prescale factor (1, 8, 64, 256, or 1024).

The extreme values for the OCR2 Register represent special cases when generating a PWM waveform output in the phase correct PWM mode. If the OCR2 is set equal to BOTTOM, the output will be continuously low and if set equal to MAX the output will be continuously high for non-inverted PWM mode. For inverted PWM the output will have the opposite logic value.

Timer/Counter Timing Diagrams

The following figures show the Timer/Counter in Synchronous mode and the timer clock (clkT2) is therefore shown as a clock enable signal. In Asynchronous mode, clkT2 should be replaced by the Timer/Counter clock rate. The figure include information on when interrupt flags are set. Figure 66 contains timing data for basic Timer/Counter operations. The figure shows the count sequence close to the MAX value in all modes other than phase correct PWM mode.
Figure 60. Timer/Counter Timing Diagram, no Prescaling

Figure 61. Timer/Counter Timing Diagram, with Prescaler (f(OCF2))

Figure 62 shows the setting of OCF2 in all modes except CTC mode.

Figure 62. Timer/Counter Timing Diagram, Setting of OCF2, with Prescaler (f(OCF2))
Figure 65 shows the setting of OCF2 and the clearing of TCNT2 in CTC mode.

Figure 66. Timer/Counter Timing Diagram, Clear Timer on Compare Match Mode, with Prescaler (fcpu, fosc/8)

8-bit Timer/Counter
Register Description

Timer/Counter Control
Register — TCCR2

<table>
<thead>
<tr>
<th>Bit</th>
<th>7</th>
<th>6</th>
<th>5</th>
<th>4</th>
<th>3</th>
<th>2</th>
<th>1</th>
<th>0</th>
<th>Function</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>FOC2</td>
<td>WGM24</td>
<td>COM2A1</td>
<td>COM2B1</td>
<td>COM2A0</td>
<td>COM2B0</td>
<td>CS2</td>
<td>CS0</td>
<td>OCR2A</td>
</tr>
<tr>
<td>Value</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>4</td>
<td>0</td>
<td>0</td>
</tr>
</tbody>
</table>

- **Bit 7 — FOC2: Force Output Compare**
  
The FOC2 bit is only active when the WGM2 bits specify a non-PWM mode. However, for ensuring compatibility with future devices, this bit must be set to zero when TCCR2 is written when operating in PWM mode. When writing a logical one to the FOC2 bit, an immediate compare match is forced on the waveform generation unit. The OCR2 output is changed according to its COM2A/B bits setting. Note that the FOC2 bit is implemented as a shadow. Therefore, it is the value present in the OCR2A/B bits that determines the effect of the forced compare.

  A FOC2 strobe will not generate any interrupt, nor will it clear the timer in CTC mode using OCR2 as TOP.

  The FOC2 bit is always read as zero.

- **Bit 5, 3 — WGM24:3: Waveform Generation Mode**
  
  These bits control the counting sequence of the counter, the source for the maximum (TOP) counter value, and what type of waveform generation to be used. Modes of operation supported by the Timer/Counter unit are: Normal mode, Clear Timer on Compare match (CTC) mode, and two types of Pulse Width Modulation (PWM) modes. See Table 60 and "Modes of Operation" on page 115.
Table 50. Waveform Generation Mode Bit Description

<table>
<thead>
<tr>
<th>Node</th>
<th>WGM1</th>
<th>WGM2</th>
<th>Timer/Counter Mode of Operation</th>
<th>TOP</th>
<th>Update of OCR2</th>
<th>TOV2 Flag</th>
<th>Set &amp; clr</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>0</td>
<td>0</td>
<td>Normal</td>
<td>OFF</td>
<td>Immediate</td>
<td>Max</td>
<td></td>
</tr>
<tr>
<td>1</td>
<td>0</td>
<td>1</td>
<td>PWM, Phase Correct</td>
<td>OFF</td>
<td>TOP</td>
<td>BOTTOM</td>
<td></td>
</tr>
<tr>
<td>2</td>
<td>1</td>
<td>0</td>
<td>CTC</td>
<td>OCR2</td>
<td>Immediate</td>
<td>Max</td>
<td></td>
</tr>
<tr>
<td>3</td>
<td>1</td>
<td>1</td>
<td>Fast PWM</td>
<td>OCR2</td>
<td>Immediate</td>
<td>Max</td>
<td></td>
</tr>
</tbody>
</table>

Note: 1. The CTC and PWM2 bit definitions are now obsolete. Use the WGM2 bits definitions. However, the functionality and location of these bits are compatible with previous versions of the timer.

* Bit 5:4 – COM2:1: Compare Match Output Mode

These bits control the Output Compare pin (OC2) behavior. If one or both of the COM2:1 bits are set, the OC2 output overrides the normal pin functionality of the I/O pin this bit is connected to. However, note that the Data Direction Register (DDR) bit corresponding to OC2 pin must be set in order to enable the output driver.

When OC2 is connected to the pin, the function of the COM2:1 bits depends on the WGM2:0 bit setting. Table 51 shows the COM2:1 bit functionality when the WGM2:0 bits are set to normal or CTC mode (non-PWM).

Table 51. Compare OutputMode non-PWM Mode

<table>
<thead>
<tr>
<th>COM21</th>
<th>COM20</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>0</td>
<td>Normal port operation, OC2 disconnected.</td>
</tr>
<tr>
<td>0</td>
<td>1</td>
<td>Toggle OC2 on compare match</td>
</tr>
<tr>
<td>1</td>
<td>0</td>
<td>Clear OC2 on compare match</td>
</tr>
<tr>
<td>1</td>
<td>1</td>
<td>Set OC2 on compare match</td>
</tr>
</tbody>
</table>

Table 52 shows the COM2:1 bit functionality when the WGM2:0 bits are set to fast PWM mode.

Table 52. Compare OutputMode, Fast PWM Mode

<table>
<thead>
<tr>
<th>COM21</th>
<th>COM20</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>0</td>
<td>Normal port operation, OC2 disconnected.</td>
</tr>
<tr>
<td>0</td>
<td>1</td>
<td>Reserved</td>
</tr>
<tr>
<td>1</td>
<td>0</td>
<td>Clear OC2 on compare match, set OC2 at TOP</td>
</tr>
<tr>
<td>1</td>
<td>1</td>
<td>Set OC2 on compare match, clear OC2 at TOP</td>
</tr>
</tbody>
</table>

Note: 1. A special case occurs when OCR2 equals TOP and COM2:1 is set. In this case, the compare match is ignored, but the set or clear is done at TOP. See "Fast PWM Mode" on page 117 for more details.

Table 53 shows the COM2:1 bit functionality when the WGM2:0 bits are set to phase correct PWM mode.

---

ATmega16(L)

---

B-45
Table 53. Compare Output Mode, Phase Correct PWM Mode(1)

<table>
<thead>
<tr>
<th>COM21</th>
<th>COM20</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>0</td>
<td>Normal port operation, OC1 disconnected.</td>
</tr>
<tr>
<td>0</td>
<td>1</td>
<td>Reserved</td>
</tr>
<tr>
<td>1</td>
<td>0</td>
<td>Clear OC2 or compare match when up-counting, Set OC2 on compare match when down-counting.</td>
</tr>
<tr>
<td>1</td>
<td>1</td>
<td>Set OC2 or compare match when up-counting, Clear OC2 on compare match when down-counting.</td>
</tr>
</tbody>
</table>

Note: 1. A special case occurs when CCGR equals 000 and COM21 is set. In this case, the compare match is ignored, but the set or clear is done at TOP. See "Phase Correct PWM Mode" on page 196 for more details.

* Bit 29 – CS2[2:0]: Clock Select

The three Clock Select bits select the clock source to be used by the Timer/Counter, see Table 54.

Table 54. Clock Select Bit Description

<table>
<thead>
<tr>
<th>CS22</th>
<th>CS21</th>
<th>CS20</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>0</td>
<td>0</td>
<td>No clock source (Timer/Counter stopped).</td>
</tr>
<tr>
<td>0</td>
<td>0</td>
<td>1</td>
<td>CK22[2:0] (no prescaling)</td>
</tr>
<tr>
<td>0</td>
<td>1</td>
<td>0</td>
<td>CK22[2:0] (From prescaler)</td>
</tr>
<tr>
<td>0</td>
<td>1</td>
<td>1</td>
<td>CK22[2:0] (From prescaler)</td>
</tr>
<tr>
<td>1</td>
<td>0</td>
<td>0</td>
<td>CK22[2:0] (From prescaler)</td>
</tr>
<tr>
<td>1</td>
<td>0</td>
<td>1</td>
<td>CK22[2:0] (From prescaler)</td>
</tr>
<tr>
<td>1</td>
<td>1</td>
<td>0</td>
<td>CK22[2:0] (From prescaler)</td>
</tr>
<tr>
<td>1</td>
<td>1</td>
<td>1</td>
<td>CK22[2:0] (From prescaler)</td>
</tr>
</tbody>
</table>

Timer/Counter Register – TCNT2

<table>
<thead>
<tr>
<th>Bit</th>
<th>7</th>
<th>6</th>
<th>5</th>
<th>4</th>
<th>3</th>
<th>2</th>
<th>1</th>
<th>0</th>
</tr>
</thead>
<tbody>
<tr>
<td>Dec/Wr</td>
<td>RW</td>
<td>RW</td>
<td>RW</td>
<td>RW</td>
<td>RW</td>
<td>RW</td>
<td>RW</td>
<td>RW</td>
</tr>
</tbody>
</table>

The Timer/Counter Register gives direct access, both for read and write operations, to the Timer/Counter unit 8-bit counter. Writing to the TCNT2 Register (0x00) clears the compare match on the following timer clock. Modifying the counter (TCNT2) while the counter is running, introduces a risk of missing a compare match between TCNT2 and the OCRn2 Register.
**Output Compare Register – OCR2**

<table>
<thead>
<tr>
<th>Bit</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>7</td>
<td>6</td>
</tr>
<tr>
<td>RW</td>
<td>RW</td>
</tr>
<tr>
<td>Value</td>
<td>0</td>
</tr>
</tbody>
</table>

The Output Compare Register contains an 8-bit value that is continuously compared with the counter value (TCNT2). A match can be used to generate an output compare interrupt, or to generate a waveform output on the OC2 pin.

**Asynchronous Operation of the Timer/Counter**

**Asynchronous Status Register – ASSR**

<table>
<thead>
<tr>
<th>Bit</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>7</td>
<td>6</td>
</tr>
<tr>
<td>RW</td>
<td>RW</td>
</tr>
<tr>
<td>Value</td>
<td>0</td>
</tr>
</tbody>
</table>

- **Bit 3 – AS2:** Asynchronous Timer/Counter2
  When AS2 is written to zero, Timer/Counter 2 is decoupled from the TC clock, clkTC. When AS2 is written to one, Timer/Counter 2 is clocked from a Crystal Oscillator connected to the Timer Oscillator 1 (TOS1) pin. When the value of AS2 is changed, the contents of TCNT2, OCR2, and TOC2 might be corrupted.

- **Bit 2 – TCIO2B:** Timer/Counter2 Update Busy
  When Timer/Counter2 operates asynchronously and TCNT2 is written, this bit becomes set. When TCNT2 has been updated from the temporary storage register, this bit is cleared by hardware. A logical zero in this bit indicates that TCNT2 is ready to be updated with a new value.

- **Bit 1 – TCIO2U:** Output Compare Register2 Update Busy
  When Timer/Counter2 operates asynchronously and OCR2 is written, this bit becomes set. When OCR2 has been updated from the temporary storage register, this bit is cleared by hardware. A logical zero in this bit indicates that OCR2 is ready to be updated with a new value.

- **Bit 0 – TCIO2C:** Timer/Counter Control Register2 Update Busy
  When Timer/Counter2 operates asynchronously and TCOR2 is written, this bit becomes set. When TOC2R has been updated from the temporary storage register, this bit is cleared by hardware. A logical zero in this bit indicates that TOC2R is ready to be updated with a new value.

- If a write is performed to any of the three Timer/Counter2 registers while its update busy flag is set, the updated value might get completed and cause an unintentional interrupt to occur.

The mechanisms for reading TCNT2, OCR2, and TOC2R are different. When reading TCNT2, the actual time value is read. When reading OCR2 or TOC2R, the value in the temporary storage register is read.

**ATmega16(L)**
Asynchronous Operation of Timer/Counter2

When Timer/Counter2 operates asynchronously, some considerations must be taken.

- **Warning:** When switching between asynchronous and synchronous clocking of Timer/Counter2, the timer registers TCNT2, OCR2, and TCGR2 might be corrupted. A safe procedure for switching clock source is:
  1. Disable the Timer/Counter2 interrupts by clearing OCF2E and TOIE2.
  2. Select clock source by setting AS2 as appropriate.
  3. Write new values to TCNT2, OCR2, and TCGR2.
  4. To switch to asynchronous operation, wait for TOCKI, OCR2E, and TCR2E.
  5. Clear the Timer/Counter2 interrupt flags.
  6. Enable interrupts, if needed.

- The Oscillator is optimized for use with a 32.768 kHz watch crystal. Applying an external clock to the TOCS1 pin may result in incorrect Timer/Counter2 operation. The CPU main clock frequency must be more than four times the Oscillator frequency.

- When writing to one of the registers TCNT2, OCR2, or TCGR2, the value is transferred to a temporary register, and latched after two positive edges on TOCS1. The user should not write a new value before the contents of the temporary register have been transferred to its destination. Each of the three internal registers have their own temporary register, which means for example that writing to TCNT2 does not disturb an OCR2Write in progress. To detect that a transfer to the destination register has taken place, the Asynchronous Status Register – ASR2 has been implemented.

- When entering Power-save or Extended Standby mode after having written to TCNT2, OCR2, or TCGR2, the user must wait until the written registers has been updated before Timer/Counter2 is used to wake up the device. Otherwise, the MCU will enter sleep mode before the changes are effective. This is particularly important if the capture compare function is disabled during writing to OCR2E or TCNT2. If the write cycle is not finished, and the MCU enters sleep mode before the OCR2E bit returns to zero, the device will never receive a compare match interrupt and the MCU will not wake up.

- If Timer/Counter2 is used to wake the device up from Power-save or Extended Standby modes, precautions must be taken to ensure that the user wants to re-enter one of these modes. The interrupt logic needs one TOCS1 cycle to be reset. If the time between wake-up and re-entering sleep mode is less than one TOCS1 cycle, the interrupt will not occur, and the device will fall to wake up. If the time is equal to the time before re-entering Power-save or Extended Standby mode is less than one TOCS1 cycle has elapsed, the following algorithm can be used to assure that one TOCS1 cycle has elapsed:
  1. Write a value to TCNT2, OCR2, or TCGR2.
  2. Wait until the corresponding Update Busy flag in ASR2 returns to zero.
  3. Enter Power-save or Extended Standby mode.

- When the asynchronous operation is selected, the 32.768 kHz Oscillator for Timer/Counter2 is always running, except in Power-down and Standby modes. After a Power-up Reset or wake-up from Power-down or Standby mode, the user should be aware of the fact that this Oscillator might take as long as one second to stabilize. The user is advised to wait for at least one second before using Timer/Counter2 after power-up or wake-up from Power-down or Standby mode. The contents of all Timer/Counter2 registers must be considered lost after a wake-up.
Description of wake up from Power-save or Extended Standby mode when the timer is clocked asynchronously: When the interrupt condition is met, the wake up process is started on the following cycle of the timer clock, that is, the timer is always advanced by at least one before the processor can read the counter value. After wake-up, the MCU is halted for four cycles, it executes the interrupt routine, and resumes execution from the instruction following SLEEP.

Reading of the TCNT2 Register shortly after wake-up from Power-save mode may give an incorrect result. Since TCKT2 is clocked on the asynchronous TOSC clock, reading TCKT2 must be done through a register synchronized to the internal I/O clock details. Synchronization takes place for every rising TOSC1 edge. When waking up from Power-save mode, and the I/O clock (PLL/IOL) again becomes active, TCKT2 will read the previous value (before entering sleep) until the next rising TOSC1 edge. The phase of the TOSC clock after waking up from Power-save mode is essentially unpredictable, as it depends on the wake-up time. The recommended procedure for reading TCKT2 is thus as follows:

1. Write any value to either of the registers OCR2 or TCCR2.
2. Wait for the corresponding Update Busy Flag to be cleared.
3. Read TCKT2.

During asynchronous operation, the synchronization of the interrupt flags for the asynchronous timers takes three processor cycles plus one timer cycle. The timer is therefore advanced by at least one before the processor can read the timer value causing the setting of the interrupt flag. The output compare pin is changed on the timer clock and is not synchronized to the processor clock.

**Timer/Counter Interrupt Mask Register – TIMSK**

<table>
<thead>
<tr>
<th>Bit</th>
<th>SW</th>
<th>SW</th>
<th>SW</th>
<th>SW</th>
<th>SW</th>
<th>SW</th>
<th>SW</th>
<th>SW</th>
<th>SW</th>
<th>SW</th>
</tr>
</thead>
<tbody>
<tr>
<td>Data/Write</td>
<td>0</td>
<td>0</td>
<td>0</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>Initial Value</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
</tbody>
</table>

- **Bit 7 – OCE2: Timer/Counter2 output Compare Match interrupt Enable**
  When the OCE2 bit is written to one and the bit in the Status Register is set (one), the Timer/Counter2 Compare Match interrupt is enabled. The corresponding interrupt is executed if a compare match in Timer/Counter2 occurs, i.e., when the OCR2 bit is set in the Timer/Counter Interrupt Flag Register – TIFR.

- **Bit 6 – TOIE2: Timer/Counter2 Overflow interrupt Enable**
  When the TOIE2 bit is written to one and the bit in the Status Register is set (one), the Timer/Counter2 Overflow interrupt is enabled. The corresponding interrupt is executed if an overflow in Timer/Counter2 occurs, i.e., when the TOV2 bit is set in the Timer/Counter Interrupt Flag Register – TIFR.
Timer/Counter Interrupt Flag Register — TIFR

<table>
<thead>
<tr>
<th>Bit</th>
<th>0</th>
<th>1</th>
<th>2</th>
<th>3</th>
<th>4</th>
<th>5</th>
<th>6</th>
<th>7</th>
</tr>
</thead>
<tbody>
<tr>
<td>Default Value</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
</tbody>
</table>

- **Bit 7 — OCFI2: Output Compare Flag 2**
  
The OCFI2 flag is set (one) when a compare match occurs between the Timer/Counter2 and the data in OCR2 — Output Compare Register 2. OCFI2 is cleared by hardware when executing the corresponding interrupt handling vector. Alternatively, OCFI2 is cleared by writing a logic one to the flag. When the bit in SPIR0, OCIE2 (Timer/Counter2 Compare match interrupt Enable), and OCFI2 are set (one), the Timer/Counter2 Compare match interrupts are executed.

- **Bit 6 — TOV2: Timer/Counter2 Overflow Flag**
  
The TOV2 flag is set (one) when an overflow occurs in Timer/Counter2. TOV2 is cleared by hardware when executing the corresponding interrupt handling vector. Alternatively, TOV2 is cleared by writing a logic one to the flag. When the bit in SREG, TOIE2 (Timer/Counter2 Overflow Interrupt Enable), and TOV2 are set (one), the Timer/Counter2 Overflow interrupt is executed. In PWM mode, this bit is set when Timer/Counter changes counting direction at TOV.

**Timer/Counter Prescaler**

Figure 64. Prescaler for TC0/TC1

The clock source for Timer/Counter2 is named CkOC2. CkOC2 is by default connected to the main system clock Ck0. By setting the AS2 bit in ASSR, Timer/Counter2 is asynchronously docked from the TOSS1 pin. This enables use of Timer/Counter2 as a Real-Time-Clock (RTC). When AS2 is set, pins TOSS1 and TOSS2 are disconnected from Port C. An external crystal can then be connected between the TOSS1 and TOSS2 pins to serve as an independent clock source for Timer/Counter2. The oscillator is optimized for use with a 32.768 kHz crystal. Applying an external clock source to TOSS1 is not recommended.
LAMPIRAN C
DATA SHEET IR – 8510
NOTES:
1. This drawing measure is a standard value. All dimensions are in millimeter.
2. In case of designation is tolerance ± 0.3mm.
3. Lead spacing is measured where the lead emerge from the package.
4. Above specification may be changed without notice. EVERLIGHT will reserve authority on material change for above specification.
5. These specification sheets include materials protected under copyright of EVERLIGHT corporation. Please don't reproduce or cause anyone to reproduce them without EVERLIGHT consent.
6. When using this produce, please observe the absolute maximum ratings and the instructions for use outlined in these specification sheets. EVERLIGHT assumes no responsibility for any damage resulting from use of the product which does not comply with the absolute maximum ratings and the instructions included in these specification sheets.
Description:
1. The module is a small type infrared remote control system receiver which has been developed and designed by utilizing the latest hybrid technology.
2. This single unit type module incorporates a photo diode and a receiving preamplifier IC.
3. The demodulated output signal can directly be decoded by a microprocessor.

Feature:
1. High protection ability to EMI and metal case can be customized.
2. Mold type and metal case type to meet the design of front panel.
3. Elliptic lens to improve the characteristic against
4. Line-up for various center carrier frequencies.
5. Low voltage and low power consumption.
6. High immunity against ambient light.
7. Photodiode with integrated circuit.
8. TTL and CMOS compatibility.

Application:
1. Optical switch
2. Light detecting portion of remote control
   • AV instruments such as Audio, TV, VCR, CD, MD, etc.
   • Home appliances such as Air-conditioner, Fan, etc.
   • The other equipments with wireless remote control.
   • CATV set top boxes
   • Multi-media Equipment
### Absolute maximum ratings:

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Symbol</th>
<th>Ratings</th>
<th>Unit</th>
<th>Notice</th>
</tr>
</thead>
<tbody>
<tr>
<td>Supply Voltage</td>
<td>Vcc</td>
<td>4.3~5.7</td>
<td>V</td>
<td></td>
</tr>
<tr>
<td>Operating Temperature</td>
<td>Topr</td>
<td>-10~+60</td>
<td>°C</td>
<td></td>
</tr>
<tr>
<td>Storage Temperature</td>
<td>Tstg</td>
<td>-20~+70</td>
<td>°C</td>
<td>4mm from mold body less than 5 seconds</td>
</tr>
<tr>
<td>Soldering Temperature</td>
<td>Tsol</td>
<td>260</td>
<td>°C</td>
<td></td>
</tr>
</tbody>
</table>

### Electro Optical Characteristics:

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Symbol</th>
<th>MIN</th>
<th>TYP</th>
<th>MAX</th>
<th>Unit</th>
<th>Condition</th>
</tr>
</thead>
<tbody>
<tr>
<td>Supply Voltage</td>
<td>Vcc</td>
<td>4.7</td>
<td>5</td>
<td>5.3</td>
<td>V</td>
<td>DC voltage</td>
</tr>
<tr>
<td>Supply Current</td>
<td>Icc</td>
<td>-</td>
<td>-</td>
<td>3</td>
<td>mA</td>
<td>No signal input</td>
</tr>
<tr>
<td>B.P.F Center Frequency</td>
<td>fo</td>
<td>-</td>
<td>37.9</td>
<td>-</td>
<td>KHz</td>
<td></td>
</tr>
<tr>
<td>Peak Wavelength</td>
<td>( \lambda_p )</td>
<td>-</td>
<td>940</td>
<td>-</td>
<td>nm</td>
<td></td>
</tr>
<tr>
<td>Transmission Distance</td>
<td>( L_0 )</td>
<td>5</td>
<td>-</td>
<td>-</td>
<td>m</td>
<td>At the ray axis *1</td>
</tr>
<tr>
<td>Transmission Distance</td>
<td>( L_{45} )</td>
<td>2.5</td>
<td>-</td>
<td>-</td>
<td>m</td>
<td></td>
</tr>
<tr>
<td>Half Angle</td>
<td>( \theta )</td>
<td>-</td>
<td>45</td>
<td>-</td>
<td>deg</td>
<td></td>
</tr>
<tr>
<td>High Level Pulse Width</td>
<td>( T_H )</td>
<td>400</td>
<td>-</td>
<td>800</td>
<td>μs</td>
<td>At the ray axis *2</td>
</tr>
<tr>
<td>Low Level Pulse Width</td>
<td>( T_L )</td>
<td>400</td>
<td>-</td>
<td>800</td>
<td>μs</td>
<td></td>
</tr>
<tr>
<td>High Level Output Voltage</td>
<td>( V_H )</td>
<td>4.5</td>
<td>-</td>
<td>-</td>
<td>V</td>
<td></td>
</tr>
<tr>
<td>Low Level Output Voltage</td>
<td>( V_L )</td>
<td>-</td>
<td>0.5</td>
<td>-</td>
<td>V</td>
<td></td>
</tr>
</tbody>
</table>

*1: The ray receiving surface at a vertex and relation to the ray axis in the range of \( \phi = 0° \) and \( \phi = 45° \).
*2: A range from 30cm to the arrival distance. Average value of 50 pulses.
TEST METHOD:
The specified electro-optical characteristics is satisfied under the following Conditions at the controllable distance.

① Measurement place
   A place that is nothing of extreme light reflected in the room.

② External light
   Project the light of ordinary white fluorescent lamps which are not high Frequency lamps and must be less then 10 Lux at the module surface. (Ee \leq 10Lux)

③ Standard transmitter
   A transmitter whose output is so adjusted as to \( V_0 = 400mV \text{p-p} \) and the output Wave form shown in Fig.-1. According to the measurement method shown in Fig.-2 the standard transmitter is specified. However, the infrared photodiode to be used for the transmitter should be \( \lambda_p = 940nm, \Delta \lambda = 50nm \). Also, photodiode is used of PD438B (\( V_R = 5V \)). (Standard light / Light source temperature 2856°K).

④ Measuring system
   According to the measuring system shown in Fig.-3
Module schematic & circuit:

Fig. 1. Transmitter Wave Form

Carrier frequency is adjusted to center frequency of each product.

DUT Output Pulse

\[ \text{Duty} = 0.5 \]

Fig. 2. Measuring Method

Fig. 3. Measuring System
TYPICAL ELECTRICAL/OPTICAL/CHARACTERISTICS CURVES

Fig. 4 Relative Spectral Sensitivity vs. Wavelength

Fig. 5 Relative Transmission Distance vs. Direction

Fig. 6 Output Pulse Length vs. Arrival Distance

Fig. 7 Arrival Distance vs. Supply Voltage

Fig. 8 Relative Transmission Distance vs. Center Carrier Frequency

Fig. 9 Arrival Distance vs. Ambient Temperature
### Reliability test item and condition:
The reliability of products shall be satisfied with items listed below.
Confidence level: 90%
LTPD: 10%

<table>
<thead>
<tr>
<th>Test Items</th>
<th>Test Conditions</th>
<th>Failure Judgement Criteria</th>
<th>Samples(n)</th>
<th>Defective(c)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Operation life</td>
<td>Vcc=5V, Ta: 25°C 1000hrs</td>
<td></td>
<td>n=22, c=0</td>
<td></td>
</tr>
<tr>
<td>Temperature cycle</td>
<td>1 cycle -20°C +25°C +70°C (30min) 5min (30min) 50 cycle test</td>
<td>Lₜ ≤ L × 0.8 Lₜ ≤ L × 0.8</td>
<td>n=22, c=0</td>
<td></td>
</tr>
<tr>
<td>Thermal shock</td>
<td>-10°C to +70°C (5min) (10sec) (5min) 50 cycle test</td>
<td></td>
<td>n=22, c=0</td>
<td></td>
</tr>
<tr>
<td>High temperature storage</td>
<td>Temp: +70°C 1000hrs</td>
<td></td>
<td>n=22, c=0</td>
<td></td>
</tr>
<tr>
<td>Low temperature storage</td>
<td>Temp: -20°C 1000hrs</td>
<td>L: Lower specification limit</td>
<td>n=22, c=0</td>
<td></td>
</tr>
<tr>
<td>High temperature High humidity</td>
<td>Ta: 85°C RH: 85% 1000hrs</td>
<td></td>
<td>n=22, c=0</td>
<td></td>
</tr>
<tr>
<td>Solder heat</td>
<td>Temp: 260± 5°C 5sec 4mm Form the bottom of the package.</td>
<td></td>
<td>n=22, c=0</td>
<td></td>
</tr>
<tr>
<td>Solderability</td>
<td>Temp: 230± 5°C 5sec 4mm Form the bottom of the package.</td>
<td>More than 90% of Lead to be covered by soldering</td>
<td>n=22, c=0</td>
<td></td>
</tr>
</tbody>
</table>
INTERFACING PC STANDARD PARALLEL PORT TO DST

AN-008IR-8510 INFRARED TRANSCEIVER

Infrared Transceiver adalah sebuah sistem yang terdiri dari Infrared Transmitter dan Infrared Receiver di mana sistem ini berfungsi untuk proses komunikasi data

Aplikasi

- Wireless Data Communication
- Alarm System
- Universal Remote Control

Deskripsi

![Infrared Transceiver Diagram](image)

Skema Infrared Transceiver

Untuk memperoleh jarak yang cukup jauh, Diode Infrared memerlukan sinyal dengan frekwensi 30 hingga 50 KHz. Berbeda dengan Diode LED yang hanya memerlukan level tegangan DC saja untuk mengaktifkan LED, Diode Infrared memerlukan sinyal AC dengan frekwensi 30 hingga 50 KHz untuk mengaktifkannya. Cahaya infrared tersebut tidak dapat ditangkap oleh mata manusia, sehingga diperlukan phototransistor untuk mendeteksiinya.

Phototransistor adalah merupakan sebuah transistor yang akan saturasi pada saat menerima sinar infrared dan cut off pada saat tidak ada sinar infrared. IR Module adalah sebuah rangkaian yang terdiri dari sebuah phototransistor dan filter yang terbentuk dalam satu modul di mana collector dari phototransistor adalah merupakan output dari modul ini. Pada saat phototransistor cut off maka tidak terjadi aliran arus dari collector menuju ke emitter sehingga collector yang merupakan output dari IR Module akan berkondisi high. Apabila phototransistor saturasi maka arus mengalir dari collector ke emitter dan output dari IR Module akan berkondisi low.
INTERFACING PC STANDARD PARALLEL PORT TO DST

Transmisidata dilakukan dengan menggunakan prinsip aktif dan non aktifnya LED Infrared sebagai kondisi logic 0 dan logic 1. Seperti telah dijelaskan sebelumnya bahwa untuk mengaktifkan LED Infrared diperlukan frekwensi sebesar 30 hingga 40 KHz, maka dalam hal ini logic 0 berarti sinyal frekwensi 30 KHz mengalir ke LED Infrared dan logic 1 berarti tidak ada sinyal yang mengalir ke LED Infrared, hal ini seperti yang tampak pada hubungan antara TXD dan TX pada Tuning Diagram berikut.

Untuk menghasilkan sinyal seperti yang tampak pada TX dibutuhkan sebuah rangkaian modulator yang terdiri dari sebuah gerbang dan rangkaian R-C sebagai oscillator. Gerbang tersebut menggunakan IC 74HC132 di mana pada saat pin TXD bercondisi high dan TXD bercondisi low maka output dari IC ini sesuai dengan tabel kebenaran yang ada pada data sheet adalah high. Namun bila sebaliknya TXD bercondisi high maka sesaat output dari IC ini berubah ke low sehingga capacitor C1 akan membuang mutannya melalui R1. Bila tegangan C1 terbuka hingga di bawah tegangan ambang 74HC132 maka input pin nomor 4 dari IC ini akan dianggap bercondisi low sehingga outputnya berubah menjadi high.

C1 kembali terisi melalui R1 hingga tegangan pada capacitor ini melebihi tegangan ambang dan input pin nomor 4 dianggap bercondisi high. Bila pada saat itu TXD masih bercondisi high maka output dari gerbang ini yaitu pin nomor 6 akan bercondisi low dan C1 kembali membuang demikian seterusnya C1 akan terisi hingga di atas tegangan ambang 74HC132 (2.5 V) dan terbuka hingga di bawah tegangan ambang 74HC132 pula. Pengisian dan pembuangan pada C1 yang terjadi berkali-kali ini menyebabkan terjadinya osilasi dengan frekwensi yang dapat dihitung dengan menggunakan rumus berikut:

\[ T = \text{Waktu Pengisian C1} + \text{Waktu Pembuangan C1} \]

\[ \text{Waktu Pengisian C1} = \text{Waktu Pembuangan C1 maka} \]

\[ T = 2 \times \text{Waktu Pengisian C1} \]

\[ T = \frac{2RC}{\ln \left[ \frac{V_T^-}{V_T^+} \right] - \ln \left[ \frac{V_T^-}{V_T^+} \right]} \]

di mana VT- adalah batas bawah tegangan ambang 74HC132 yaitu sekitar 2 Volt dan VT+ adalah batas atas dari tegangan ambang 74HC132 yaitu sekitar 3 Volt. Dengan R sebesar 3.9K, C10nF dan Vsupply = 5V maka akan diperoleh harga T = 31.63 uS

\[ f = \frac{1}{T} \]

\[ f = 31.616 \text{ KHz} \]

Jadi pada intinya apabila input TXD bercondisi high maka frekwensi oscillator sebesar 31.616 KHz yang terjadi pada pin nomor 4 akan dilewatkanke outputnya dengan frekwensi yang sama persis, namun bila TXD bercondisi low maka osilasi pada pin nomor 4 akan berhenti dan output dari gerbang adalah high.
INTERFACING PC STANDARD PARALLEL PORT TO DST

Timing Diagram

Ayunan sinyal berfrekuensi 31.6 KHz ini diperkuat lagi oleh gerbang lain dari 74HC132 yang dibentuk menjadi inverter dan diteruskan ke transistor BD400 yang mengalirkan sinyal-sinyal frekwensi hasil dari modulator tersebut ke Diode Infrared.

Pancaran Diode Infrared diterima oleh IR Module dan membuat output modul ini menjadi low hingga pancaran Diode Infrared berhenti dan output dari modul menjadi high. Hasil output dari modul ini yaitu RXD seperti yang tampak pada timing diagram mempunyai bentuk gelombang yang sama persis dengan TXD.