LAMPIRAN

A

LAMPIRAN A
LISTING PROGRAM

Di bawah ini adalah Listing Program Aplikasi Alar Pengecek Harga:

```
#include <mega16.h>
#include <delay.h>
#include <stdio.h>
#include <stdlib.h>
#include <spi.h>

unsigned char mystring[]="Ultra Milk Rp 3800 Kacang Garuda Rp 1500 Kue Bollu Rp
1000 Biskuit Oreo Rp 1000";
unsigned char tampung[40];
unsigned char tampung2[40];
unsigned char tampung3[40];
unsigned char sector[512];
void nodata(void);
void lihatdata(void);
char Command(char befF, unsigned int AdrH, unsigned int AdrL, char befH ) ;
int writeramtommc(void);
int sendmmc(void);

int i,a,aa,b,bb,j,c,z;
long pulsa=0;
```

```
// Alphanumeric LCD Module functions
#asm
   .equ __lcd_port=0x15 ;PORTC
#endasm
#include <lcd.h>

// External Interrupt 0 service routine
interrupt [EXT_INT0] void ext_int0_isr(void)
{
// Place your code here

}

#define RXB8 1
#define TXB8 0
#define UPE 2
#define OVR 3
#define FE 4
#define UDRE 5
#define RXC 7

#define FRAMING_ERROR (1<<FE)
#define PARITY_ERROR (1<<UPE)
#define DATA_OVERRUN (1<<OVR)
#define DATA_REGISTER_EMPTY (1<<UDRE)
#define RX_COMPLETE (1<<RXC)

// USART Receiver buffer
#define RX_BUFFER_SIZE 8
char rx_buffer[RX_BUFFER_SIZE];
```

```
#if RX_BUFFER_SIZE<256
unsigned char rx_wr_index,rx_rd_index,rx_counter;
#else
unsigned int rx_wr_index,rx_rd_index,rx_counter;
#endif

// This flag is set on USART Receiver buffer overflow
bit rx_buffer_overflow;

// USART Receiver interrupt service routine
interrupt [USART_RXC] void usart_rx_isr(void)
{
char status,data;
status=UCSRA;
data=UDR;
if ((status & (FRAMING_ERROR | PARITY_ERROR | DATA_OVERRUN))==0)
   {
   rx_buffer[rx_wr_index]=data;
   if (++rx_wr_index == RX_BUFFER_SIZE) rx_wr_index=0;
   if (++rx_counter == RX_BUFFER_SIZE)
      {
      rx_counter=0;
      rx_buffer_overflow=1;
      };
   };
}

#ifndef _DEBUG_TERMINAL_IO_
// Get a character from the USART Receiver buffer
```

```
#define _ALTERNATE_GETCHAR_
#pragma used+
char getchar(void)
{
char data;
while (rx_counter==0);
data=rx_buffer[rx_rd_index];
if (++rx_rd_index == RX_BUFFER_SIZE) rx_rd_index=0;
#asm("cli")
--rx_counter;
#asm("sei")
return data;
}
#pragma used-
#endif

// USART Transmitter buffer
#define TX_BUFFER_SIZE 8
char tx_buffer[TX_BUFFER_SIZE];

#if TX_BUFFER_SIZE<256
unsigned char tx_wr_index,tx_rd_index,tx_counter;
#else
unsigned int tx_wr_index,tx_rd_index,tx_counter;
#endif

// USART Transmitter interrupt service routine
interrupt [USART_TXC] void usart_tx_isr(void)
{
if (tx_counter)
   {
```

```
   --tx_counter;
   UDR=tx_buffer[tx_rd_index];
   if (++tx_rd_index == TX_BUFFER_SIZE) tx_rd_index=0;
   };
}

#ifndef _DEBUG_TERMINAL_IO_
// Write a character to the USART Transmitter buffer
#define _ALTERNATE_PUTCHAR_
#pragma used+
void putchar(char c)
{
while (tx_counter == TX_BUFFER_SIZE);
#asm("cli")
if (tx_counter || ((UCSRA & DATA_REGISTER_EMPTY)==0))
   {
   tx_buffer[tx_wr_index]=c;
   if (++tx_wr_index == TX_BUFFER_SIZE) tx_wr_index=0;
   ++tx_counter;
   }
else
   UDR=c;
#asm("sei")
}
#pragma used-
#endif

// Standard Input/Output functions
#include <stdio.h>

// SPI interrupt service routine
```

```
interrupt [SPI_STC] void spi_isr(void)
{
unsigned char data;
data=SPDR;
// Place your code here

}

// Declare your global variables here

void main(void)
{
// Declare your local variables here

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

// Port B initialization
// Func7=Out Func6=In Func5=Out Func4=Out Func3=In Func2=In Func1=In Func0=In
// State7=0 State6=T State5=0 State4=0 State3=T State2=T State1=T State0=T
PORTB=0x00;
DDRB=0xB0;

// 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;


// USART initialization
// Communication Parameters: 8 Data, 1 Stop, Even Parity
// USART Receiver: On
// USART Transmitter: On
// USART Mode: Asynchronous
// USART Baud rate: 38400
UCSRA=0x00;
UCSRB=0xD8;
UCSRC=0xA6;
UBRRH=0x00;
UBRRL=0x0B;

// SPI initialization
// SPI Type: Master
// SPI Clock Rate: 1843.200 kHz
// SPI Clock Phase: Cycle Half
// SPI Clock Polarity: Low
// SPI Data Order: MSB First
SPCR=0xD0;
SPSR=0x00;
```

```
// Clear the SPI interrupt flag
#asm
   in   r30,spsr
   in   r30,spdr
#endasm

// LCD module initialization
lcd_init(16);

// Global enable interrupts
#asm("sei")
for(i = 0;i<512;i++)
{
   sector[i] = mystring[z];
   z++;
   if(z==81)
      z = 0;
}
writeramtommc();
sendmmc();
for(i = 0;i<100;i++)
{
  tampung[i] = 0x00;
}
while (1)
     {
     // Place your code here
     if(rx_counter)
     {
```

```
tampung[pulsa] = getchar();
putchar(tampung[pulsa]);
if(tampung[21] == 0xF8 && tampung[19] == 0x80)
{
  a = 10;
  aa = 0;
  b = 7;
  bb = 11;
  lihatdata();
}
else if(tampung[20] == 0xF8 && tampung[23] == 0xF8 && tampung[25] ==  0xF8)
{
  a = 13;
  b = 7;
  aa = 18;
  bb = 33;
  lihatdata();
}
else if(tampung[21] == 0xF8 && tampung[19] == 0xF8 )
{
  a = 10;
  b = 7;
  aa = 41;
  bb = 51;
  lihatdata();
}
else if(tampung[21] == 0x80 && tampung[25] == 0xF8 && tampung[27] == 0xF8)
{
 a = 13;
 b = 7;
 aa = 59;
```

```
    bb = 72;
    lihatdata();
  }
  else
  {
     nodata();
  }
  pulsa++;
}
if(PINA.0 == 0)
{
     lcd_clear();
     pulsa = 0;
     i = 0;
     for(i = 0;i<100;i++)
     {
        tampung[i] = 0x00;
     }
}
if(PINA.1 == 0)
{
    lcd_clear();
    lcd_gotoxy(0,0);
    for(i = 0;i<a;i++)
    {
       tampung2[i] = sector[i+aa];
    }
    lcd_puts(tampung2);
    lcd_gotoxy(0,1);
    for(j = 0;j<b;j++)
    {
```

```
            tampung3[j] = sector[j+bb];
          }
         lcd_puts(tampung3);

      }
      };
}
void nodata(void)
{
   lcd_clear();
   lcd_gotoxy(0,0);
   sprintf(tampung2,"tidak ada data");
   lcd_puts(tampung2);
}
void lihatdata(void)
{
   lcd_clear();
   lcd_gotoxy(0,0);
   sprintf(tampung2,"lihat data");
   lcd_puts(tampung2);
}
char Command(char befF, unsigned int AdrH, unsigned int AdrL, char befH )
{          // sends a command to the MMC
           spi(0xFF);
           spi(befF);
           spi((unsigned int)(AdrH >> 8));
           spi((unsigned int)AdrH);
           spi((unsigned int)(AdrL >> 8));
           spi((unsigned int)AdrL);
           spi(befH);
           spi(0xFF);
```

```
        return spi(0xFF);       // return the last received character
}
int writeramtommc(void)//write ram sector to mmc
{
   int i;
        // 512 byte-write-mode
        if (Command(0x58,0,512,0xFF) !=0) {
                return 1;
        }
        spi(0xFF);
        spi(0xFF);
        spi(0xFE);
        // write ram sectors to MMC
        for (i=0;i<512;i++) {
                spi(sector[i]);
        }
        // at the end, send 2 dummy bytes
        spi(0xFF);
        spi(0xFF);



        // wait until MMC is not busy anymore
        while(spi(0xFF) != (char)0xFF);

   return 0;
}
int sendmmc(void)//send 512 bytes from the mmc and save it to buffer variable
{
   int i;
        // 512 byte-read-mode
        if (Command(0x51,0,512,0xFF) != 0) {
```

```
            return 1;
        }
        // wait for 0xFE - start of any transmission
        // ATT: typecast (char)0xFE is a must!
        while(spi(0xFF) != (char)0xFE);

        for(i=0; i < 512; i++) {
            UDR = spi(0xFF);  // send character
            sector[i] = UDR;
        }
        // at the end, send 2 dummy bytes
        spi(0xFF); // actually this returns the CRC/checksum byte
        spi(0xFF);
        return 0;
}
```

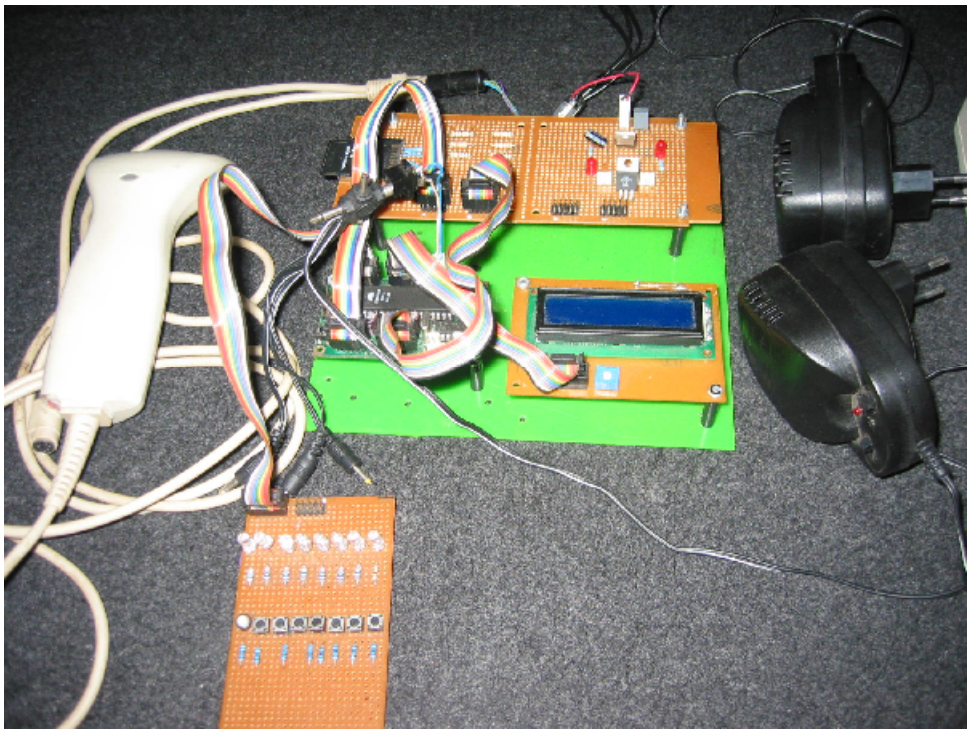
LAMPIRAN B

LAMPIRAN B

TABEL KODE BARCODE DAN FOTO ALAT

Satu (1)	Dua (2)	Tiga (3)	Empat (4)	Lima (5)
Enam (6)	Tujuh (7)	Delapan (8)	Sembilan (9)	Nol (0)

Tampilan Barcode CODE39



Tampilan Foto Alat