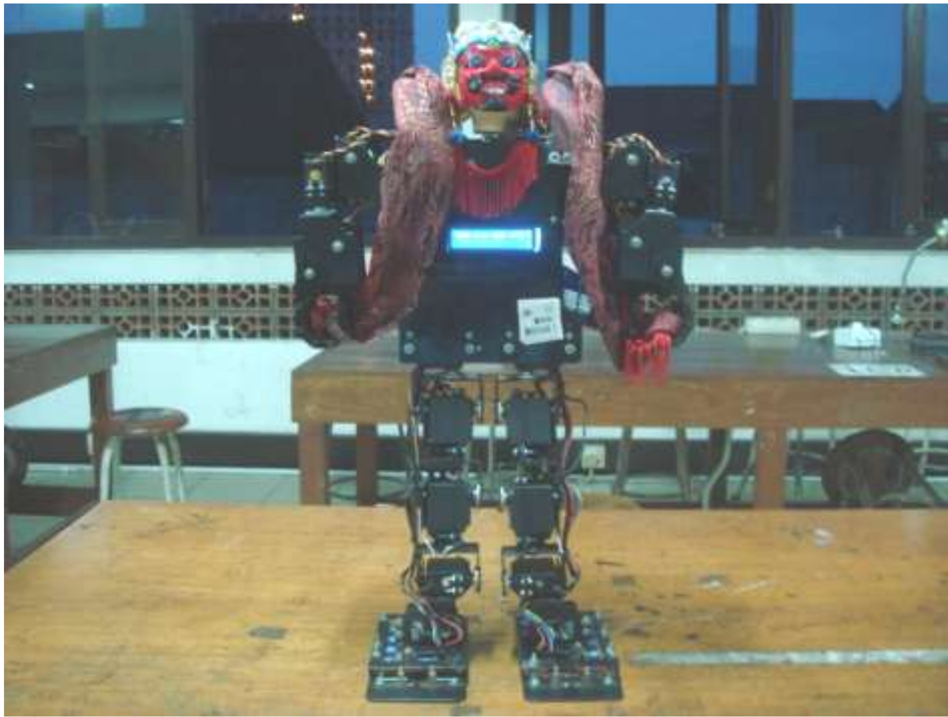
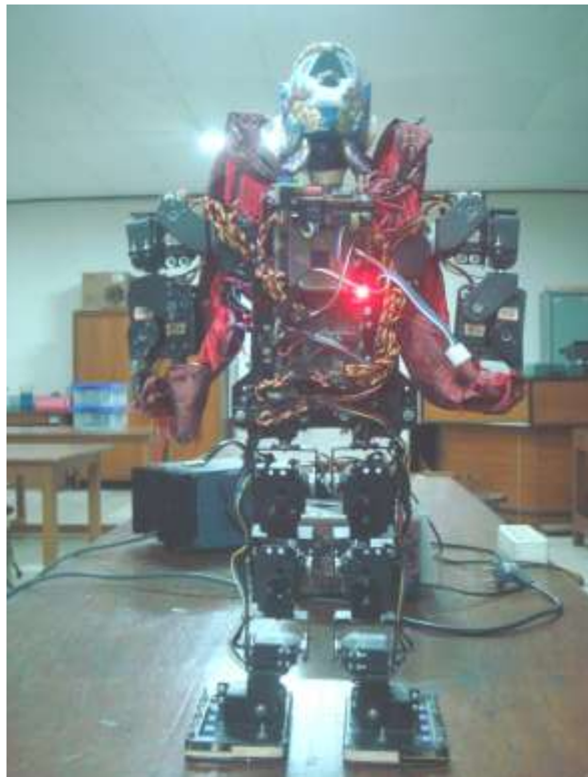
LAMPIRAN A
FOTO ROBOT PENARI KLONO TOPENG CHRENJI

Tampak Depan



Tampak Belakang



Tampak Samping Kiri

Tampak Samping Kanan

Tampak Atas

LAMPIRAN B
PROGRAM PADA PENGONTROL MIKRO ATMEGA 128 SMD

PROGRAM UTAMA

```
/*****************************************************
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    : 4/22/2011
Author  : Panji,Rezaly, Christian
Company : Chrenji
Comments:


Chip type           : ATmega128
Program type        : Application
Clock frequency     : 16.000000 MHz
Memory model        : Small
External SRAM size  : 0
Data Stack size     : 1024
*****************************************************/

#include <mega128.h>
#include <delay.h>
#include <stdio.h>
#include <stdlib.h>
#include <i2c.h>
#include <lcd.h>

#define PIN_SOUND 0
#define ADC_VREF_TYPE 0x00
#define BANYAK_SAMPLE 128
#define FREKUENSI_SAMPLING 8000
#define AMBIL_FREKUENSI 500
#define SOUND_VALUE 250

// Alphanumeric LCD Module functions
#asm
   .equ __lcd_port=0x15 ;PORTC
#endasm

// I2C Bus functions
#asm
   .equ __i2c_port=0x12 ;PORTD
   .equ __sda_bit=1
   .equ __scl_bit=0
#endasm

// Read the AD conversion result
unsigned int read_adc(unsigned char adc_input)
{
ADMUX=adc_input | (ADC_VREF_TYPE & 0xff);
// Start the AD conversion
ADCSRA|=0x40;
// Wait for the AD conversion to complete
while ((ADCSRA & 0x10)==0);
ADCSRA|=0x10;
return ADCW;
}
```

```
char LCD[40],Text[32];
eeprom unsigned int Compass_Value = 0;
unsigned char LineSensorValue = 0;

#include "LineSensor.c"
#include "SSC32.c"

// Declare your global variables here
#define PushButton ((~PINA)>>4)

void main(void)
{
unsigned int Compass_PV=0, counter = 0;
unsigned int ADC[10];
unsigned char i=0,a=0,b=0;
int k;
// 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=0xFF;
DDRA=0x00;

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

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

// Port E 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
PORTE=0x00;
DDRE=0x00;

// Port F 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
PORTF=0x00;
DDRF=0x00;

// Port G initialization
// Func4=In Func3=In Func2=In Func1=In Func0=In
// State4=T State3=T State2=T State1=T State0=T
PORTG=0x00;
DDRG=0x00;

// Timer/Counter 0 initialization
// Clock source: System Clock
// Clock value: Timer 0 Stopped
```

```
// Mode: Normal top=FFh
// OC0 output: Disconnected
ASSR=0x00;
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.
// OC1C 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
// Compare C Match Interrupt: Off
TCCR1A=0x00;
TCCR1B=0x00;
TCNT1H=0x00;
TCNT1L=0x00;
ICR1H=0x00;
ICR1L=0x00;
OCR1AH=0x00;
OCR1AL=0x00;
OCR1BH=0x00;
OCR1BL=0x00;
OCR1CH=0x00;
OCR1CL=0x00;

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

// Timer/Counter 3 initialization
// Clock source: System Clock
// Clock value: Timer 3 Stopped
// Mode: Normal top=FFFFh
// Noise Canceler: Off
// Input Capture on Falling Edge
// OC3A output: Discon.
// OC3B output: Discon.
// OC3C output: Discon.
// Timer 3 Overflow Interrupt: Off
// Input Capture Interrupt: Off
// Compare A Match Interrupt: Off
// Compare B Match Interrupt: Off
// Compare C Match Interrupt: Off
TCCR3A=0x00;
TCCR3B=0x00;
TCNT3H=0x00;
TCNT3L=0x00;
ICR3H=0x00;
ICR3L=0x00;
OCR3AH=0x00;
```

```
OCR3AL=0x00;
OCR3BH=0x00;
OCR3BL=0x00;
OCR3CH=0x00;
OCR3CL=0x00;

// External Interrupt(s) initialization
// INT0: Off
// INT1: Off
// INT2: Off
// INT3: Off
// INT4: Off
// INT5: Off
// INT6: Off
// INT7: Off
EICRA=0x00;
EICRB=0x00;
EIMSK=0x00;

// Timer(s)/Counter(s) Interrupt(s) initialization
TIMSK=0x00;
ETIMSK=0x00;

// USART0 initialization
// Communication Parameters: 8 Data, 1 Stop, No Parity
// USART0 Receiver: On
// USART0 Transmitter: On
// USART0 Mode: Asynchronous
// USART0 Baud rate: 115200
UCSR0A=0x00;
UCSR0B=0x18;
UCSR0C=0x06;
UBRR0H=0x00;
UBRR0L=0x08;

// USART1 initialization
// Communication Parameters: 8 Data, 1 Stop, No Parity
// USART1 Receiver: On
// USART1 Transmitter: On
// USART1 Mode: Asynchronous
// USART1 Baud rate: 38400
UCSR1A=0x00;
UCSR1B=0x18;
UCSR1C=0x06;
UBRR1H=0x00;
UBRR1L=0x11;

// Analog Comparator initialization
// Analog Comparator: Off
// Analog Comparator Input Capture by Timer/Counter 1: Off
ACSR=0x80;
SFIOR=0x00;

// ADC initialization
// ADC Clock frequency: 1000.000 kHz
// ADC Voltage Reference: AREF pin
ADMUX=ADC_VREF_TYPE & 0xff;
ADCSRA=0x84;

// LCD module initialization
lcd_init(16);
// I2C Bus initialization
i2c_init();
```

```
while (1)
{
if(PushButton == 0)            // CEK ADC
{
       GoHome();
       while(1)
       {
              lcd_clear();
              sprintf(LCD,"%d ",read_adc(1));
              lcd_puts(LCD);
              delay_ms(500);
       }
}
else if(PushButton == 2)               //program utama
{
while(1)
       {      GoHomeBaru();
              lcd_clear();
              sprintf(LCD,"%d",read_adc(1),);
              lcd_puts(LCD);
              delay_ms(100);

              if(read_adc(1)== 120 && k==0)
              {
              GoSalamPembuka();
              delay_ms(100);   k=1;
              }
              if(read_adc(1)== 120 && k==1)
              {
              GoTanjakKanan();
              delay_ms(100);   k=2;
              }
              if(read_adc(1)== 120 && k==2)
              {
              GoPrepareTrecekKiri();
              GoTrecekKiri();
              GoTrecekKiri();
              GoHomeBaru();
              delay_ms(100);   k=3;
              }
              if(read_adc(1)== 120 && k==3)
              {
              GoJalanJongkok();
              GoJalanJongkok();GoJalanJongkok();
              delay_ms(100);   k=4;
              }
              if(read_adc(1)== 120  && k==4)
              {
              GoZonaTengah();
              GoHomeBaru();
              delay_ms(100);   k=5;
              }
              if(read_adc(1)== 120  && k==5)
              {
              GoJalanJongkok();
              GoJalanJongkok();GoJalanJongkok();
              delay_ms(100);   k=6;
              }
              if(read_adc(1)== 120  && k==6)
              {
              GoBelokKanan();
              GoBelokKanan();
              GoBelokKanan();
              delay_ms(100);   k=7;
```

```
            }
            if(read_adc(1)== 120  && k==7)
            {
            GoJalanJongkok();
            GoJalanJongkok();GoJalanJongkok();
            delay_ms(100);   k=8;
            }
            if(read_adc(1)== 120  && k==8)
            {
            GoBelokKiri();
            GoBelokKiri();
            GoBelokKiri();
            delay_ms(100);   k=9;
            }
            if(read_adc(1)== 120  && k==9)
            {
            GoJalanJongkok();
            GoJalanJongkok();GoJalanJongkok();
            delay_ms(100);   k=10;
            }
            if(read_adc(1)== 120 && k==9)
            {
            GoSalamPenutup();
            delay_ms(100);   k=0;
            }

            else if(read_adc(1)==! 120)
            {
                        GoHomeBaru();
                         delay_ms(100);
            }
            //else;
      }
}


else if(PushButton == 5)              // Coba Tanjak Kanan
{
      while(1)
      {
            GoTanjakKanan();
            delay_ms(100);
      }
}

else if(PushButton == 6)              //Coba belok Kanan
{
      while(1)
      {
            GoHome();
            delay_ms(1000);
            while(1)
            {
                  GoBelokKanan();
                  delay_ms(100);
            }
      }
}

else if(PushButton == 7)              //Coba belok kiri
{
      while(1)
      {
            GoHome();
```

```
				delay_ms(1000);
				while(1)
				{
					for(i=0;i<4;i++)
					{
						GoBelokKiri();
						delay_ms(100);
					}

				}
			}
		}

		else if(PushButton == 8)		//Coba Trecek Kiri
		{
			while(1)
			{
				GoHome();
				delay_ms(1000);
				while(1)
				{

					GoPrepareTrecekKiri();
					GoTrecekKiri();
					GoTrecekKiri();
					GoTrecekKiri();
					GoTrecekKiri();
					delay_ms(500);
					}
			}
		}

		else if(PushButton == 9)		//Coba Trecek Kanan
		{
			while(1)
			{
				GoHome();
				delay_ms(1000);
				GoPrepareTrecekKanan();
					GoTrecekKanan();
					GoTrecekKanan();
					GoTrecekKanan();
					GoTrecekKanan();
					delay_ms(500);
					GoHomeBaru();
			}
		}

		else if(PushButton == 10)		//Coba Jalan Jongkok
		{
			while(1)
			{
				GoHome();
				delay_ms(1000);
				GoJalanJongkok();

			}
		}

		else if(PushButton == 11)		//Coba Trecek Kiri
		{
		}

		else if(PushButton == 12)		// coba seblak
```

```
{
	while(1)
	{
		GoSeblak();
	}
}

else if(PushButton == 14)		// read Line Sensor Result
{
	GoHome();
	while(1)
	{
		if(LineSensor() == 1)
		{
			lcd_clear();
			lcd_putsf("KIRI");
			delay_ms(100);
		}
		else if(LineSensor() == 2)
		{
			lcd_clear();
			lcd_putsf("KANAN");
			delay_ms(100);
		}
		else if(LineSensor() == 3)
		{
			lcd_clear();
			lcd_putsf("BOTH");
			delay_ms(100);
		}
	}
}

else if(PushButton == 15)		// read Line Sensor value
{
	GoHome();
	while(1)
	{
		lcd_clear();
		sprintf(LCD,"%d %d %d %d\n%d %d %d %d", PINB.0, PINB.1, PINB.2, PINB.3, PINB.4,
PINB.5, PINB.6, PINB.7);
		lcd_puts(LCD);
		delay_ms(100);
	}
}
};
}
```

SUB PROGRAM UNTUK SSC-32

```
#include <string.h>
#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)

flash int HomePosition[1][64] = {

{2550,1500,1500,1500,1500,1500,1500,1500,1500,1500,1500,981,2189,1539,761,1461,1500,2069,950,1500,
2239,1391,1550,1910,730,1491,1519,1461,1111,2300,1450,1509,600,600,600,600,600,600,600,600,600,600,
600,600,600,600,600,600,600,600,600,600,600,600,600,600,600,600,600,600,600,600,600,600}
                    };

flash int TanjakKanan[7][64] = {

{2500,1500,1500,1500,1500,1500,1500,1500,1500,1500,1500,981,2189,1539,761,1461,1500,2069,950,1500,
2239,1391,1550,1910,730,1491,1519,1461,1111,2300,1499,1509,600,600,600,600,600,600,600,600,600,600,
600,600,600,600,600,600,600,600,600,600,600,600,600,600,600,600,600,600,600,600,600,600},

{2420,1500,1500,1500,1500,1500,1500,1500,1500,1500,1500,1816,2164,1539,1380,1461,1937,1290,1710,1
541,2302,1391,877,1439,985,1788,1548,1430,1716,1402,1818,1524,1000,1000,1000,1000,1000,1000,1000,1
000,1000,1000,1000,1000,1000,1000,1000,1000,1000,1000,1000,1000,1000,1000,1000,1000,1000,1000,100
0,1000,1000,1000,1000,1000},

{2420,1500,1500,1500,1500,1500,1500,1500,1500,1500,1500,1710,1598,1539,557,1461,1008,1373,1063,15
41,1520,1391,877,1439,985,1788,1548,1430,1716,1402,1795,1524,600,600,600,600,600,600,600,600,600,60
0,600,600,600,600,600,600,600,600,600,600,600,600,600,600,600,600,600,600,600,600,600,600},

{2420,1500,1500,1500,1500,1500,1500,1500,1500,1500,1500,1816,2164,1539,1380,1461,1937,1290,1710,1
541,2302,1391,877,1439,985,1788,1548,1430,1716,1402,1795,1524,600,600,600,600,600,600,600,600,600,6
00,600,600,600,600,600,600,600,600,600,600,600,600,600,600,600,600,600,600,600,600,600,600},

{2420,1500,1500,1500,1500,1500,1500,1500,1500,1500,1500,1710,1598,1539,557,1461,1008,1373,1063,15
41,1520,1391,877,1439,985,1788,1548,1430,1716,1402,1795,1524,600,600,600,600,600,600,600,600,600,60
0,600,600,600,600,600,600,600,600,600,600,600,600,600,600,600,600,600,600,600,600,600,600},

{2420,1500,1500,1500,1500,1500,1500,1500,1500,1500,1500,952,1459,1682,586,1461,1627,1290,866,1570,
1487,1391,877,1541,985,1717,1592,1430,1692,1461,1795,1590,800,800,800,800,800,800,800,800,800,800,8
00,800,800,800,800,800,800,800,800,800,800,800,800,800,800,800,800,800,800,800,800,800},

{2500,1500,1500,1500,1500,1500,1500,1500,1500,1500,1500,981,2189,1539,761,1461,1500,2069,950,1500,
2239,1391,1550,1910,730,1491,1519,1461,1111,2300,1496,1509,500,500,500,500,500,500,500,500,500,500,
500,500,500,500,500,500,500,500,500,500,500,500,500,500,500,500,500,500,500,500,500,500}
                    };

flash int SalamPembuka[7][64] = {

{2500,1500,1500,1500,1500,1500,1500,1500,1500,1500,1500,981,2189,1539,761,1461,1500,2069,950,1500,
2239,1391,1550,1910,730,1491,1519,1461,1111,2300,1450,1509,600,600,600,600,600,600,600,600,600,600,
600,600,600,600,600,600,600,600,600,600,600,600,600,600,600,600,600,600,600,600,600,600},

{2500,1500,1500,1500,1500,1500,1500,1500,1500,1500,1500,1290,2189,1539,761,1461,1500,1261,1570,15
00,2239,1391,1550,1910,730,1491,1519,1461,1111,2300,1450,1509,600,600,600,600,600,600,600,600,600,6
00,600,600,600,600,600,600,600,600,600,600,600,600,600,600,600,600,600,600,600,600,600,600},

{2500,1500,1500,1500,1500,1500,1500,1500,1500,1500,1500,1880,1402,1539,761,1461,1500,1739,1021,15
00,2239,1391,1550,1910,730,1491,1519,1461,1111,2300,1450,1509,600,600,600,600,600,600,600,600,600,6
00,600,600,600,600,600,600,600,600,600,600,600,600,600,600,600,600,600,600,600,600,600,600},

{2500,1500,1500,1500,1500,1500,1500,1500,1500,1500,1500,1747,2189,1539,1516,1461,1500,1373,1062,1
500,1470,1391,1550,1950,730,1491,1519,1461,1070,2339,1450,1509,600,600,600,600,600,600,600,600,600,
600,600,600,600,600,600,600,600,600,600,600,600,600,600,600,600,600,600,600,600,600,600,600},

{2500,1500,1500,1500,1500,1500,1500,1500,1500,1500,1500,1958,2189,1539,1516,1461,1500,1164,1063,1
500,1470,1391,1550,1719,1181,1241,1519,1461,1300,1889,1700,1509,600,600,600,600,600,600,600,600,60
0,600,600,600,600,600,600,600,600,600,600,600,600,600,600,600,600,600,600,600,600,600,600,600},
```

```
{2500,1500,1500,1500,1500,1500,1500,1500,1500,1500,1500,1747,2189,1539,1516,1461,1500,1373,1062,1
500,1470,1391,1550,1950,730,1491,1519,1461,1070,2339,1450,1509,600,600,600,600,600,600,600,600,600,
600,600,600,600,600,600,600,600,600,600,600,600,600,600,600,600,600,600,600,600,600,600,600,600},

{2500,1500,1500,1500,1500,1500,1500,1500,1500,1500,1500,981,2189,1539,761,1461,1500,2069,950,1500,
2239,1391,1550,1910,730,1491,1519,1461,1111,2300,1450,1509,600,600,600,600,600,600,600,600,600,600,
600,600,600,600,600,600,600,600,600,600,600,600,600,600,600,600,600,600,600,600,600,600,600}
                         };

flash int SalamPenutup[6][64] = {

{2500,1500,1500,1500,1500,1500,1500,1500,1500,1500,1500,981,2189,1539,761,1461,1500,2069,950,1500,
2239,1391,1550,1910,730,1491,1519,1461,1111,2300,1450,1509,600,600,600,600,600,600,600,600,600,600,
600,600,600,600,600,600,600,600,600,600,600,600,600,600,600,600,600,600,600,600,600,600,600},

{2500,1500,1500,1500,1500,1500,1500,1500,1500,1500,1500,1290,2189,1539,761,1461,1500,1261,1570,15
00,2239,1391,1550,1910,730,1491,1519,1461,1111,2300,1450,1509,600,600,600,600,600,600,600,600,600,6
00,600,600,600,600,600,600,600,600,600,600,600,600,600,600,600,600,600,600,600,600,600,600,600},

{2500,1500,1500,1500,1500,1500,1500,1500,1500,1500,1500,1880,1402,1539,761,1461,1500,1739,1021,15
00,2239,1391,1550,1910,730,1491,1519,1461,1111,2300,1450,1509,600,600,600,600,600,600,600,600,600,6
00,600,600,600,600,600,600,600,600,600,600,600,600,600,600,600,600,600,600,600,600,600,600,600},

{2500,1500,1500,1500,1500,1500,1500,1500,1500,1500,1500,1747,2189,1539,1516,1461,1500,1373,1062,1
500,1470,1391,1550,1950,730,1491,1519,1461,1070,2339,1450,1509,600,600,600,600,600,600,600,600,600,
600,600,600,600,600,600,600,600,600,600,600,600,600,600,600,600,600,600,600,600,600,600,600,600},

{2500,1500,1500,1500,1500,1500,1500,1500,1500,1500,1500,1958,2189,1539,1516,1461,1500,1164,1063,1
500,1470,1391,1550,1719,1181,1241,1519,1461,1300,1889,1700,1509,600,600,600,600,600,600,600,600,60
0,600,600,600,600,600,600,600,600,600,600,600,600,600,600,600,600,600,600,600,600,600,600,600},

{2500,1500,1500,1500,1500,1500,1500,1500,1500,1500,1500,1747,2189,1539,1516,1461,1500,1373,1062,1
500,1470,1391,1550,1950,730,1491,1519,1461,1070,2339,1450,1509,600,600,600,600,600,600,600,600,600,
600,600,600,600,600,600,600,600,600,600,600,600,600,600,600,600,600,600,600,600,600,600,600,600}
                         };


flash int BelokKiri[5][64] = {

{2500,1500,1500,1500,1500,1500,1500,1500,1500,1500,1500,981,2189,1539,761,1461,1500,2069,1062,150
0,2239,1391,1491,1960,700,1491,1519,1430,1070,2370,1450,1461,600,600,600,600,600,600,600,600,600,60
0,600,600,600,600,600,600,600,600,600,600,600,600,600,600,600,600,600,600,600,600,600,600,600},

{2500,1500,1500,1500,1500,1500,1500,1500,1500,1500,1500,838,2143,1539,761,1461,1500,1948,1043,150
0,2239,1391,1491,1980,700,1509,1639,1931,1029,2310,1481,1509,600,600,600,600,600,600,600,600,600,60
0,600,600,600,600,600,600,600,600,600,600,600,600,600,600,600,600,600,600,600,600,600,600,600},

{2500,1500,1500,1500,1500,1500,1500,1500,1500,1500,1500,844,2080,1539,761,1461,1500,1878,1043,150
0,2239,1391,1491,1980,700,1509,1633,1683,1194,2310,1481,1573,700,700,700,700,700,700,700,700,700,70
0,700,700,700,700,700,700,700,700,700,700,700,700,700,700,700,700,700,700,700,700,700,700,700},

{2500,1500,1500,1500,1500,1500,1500,1500,1500,1500,1500,1063,2007,1539,761,1461,1500,2218,1043,15
00,2239,1391,1491,1980,700,1509,1633,1461,1131,2310,1481,1573,600,600,600,600,600,600,600,600,600,6
00,600,600,600,600,600,600,600,600,600,600,600,600,600,600,600,600,600,600,600,600,600,600,600},

{2500,1500,1500,1500,1500,1500,1500,1500,1500,1500,1500,981,2189,1539,761,1461,1500,2069,1062,150
0,2239,1391,1491,1960,700,1491,1519,1430,1070,2370,1450,1461,900,900,900,900,900,900,900,900,900,90
0,900,900,900,900,900,900,900,900,900,900,900,900,900,900,900,900,900,900,900,900,900,900,900}
                         };

flash int BelokKanan[7][64] = {
```

```
{2500,1500,1500,1500,1500,1500,1500,1500,1500,1500,1500,981,2189,1539,761,1461,1500,2069,1062,150
0,2239,1391,1491,1960,700,1491,1519,1430,1070,2370,1450,1461,600,600,600,600,600,600,600,600,600,60
0,600,600,600,600,600,600,600,600,600,600,600,600,600,600,600,600,600,600,600,600,600,600},

{2420,1500,1500,1500,1500,1500,1500,1500,1500,1500,1500,981,2189,1539,761,1461,1500,2069,1062,150
0,2239,1391,1488,1951,700,1548,1646,1427,1141,2310,1467,1552,300,300,300,300,300,300,300,300,300,30
0,300,300,300,300,300,300,300,300,300,300,300,300,300,300,300,300,300,300,300,300,300,300},

{2420,1500,1500,1500,1500,1500,1500,1500,1500,1500,1500,981,2189,1539,761,1461,1500,2069,1062,150
0,2239,1391,1205,1895,700,1567,1681,1427,1107,2310,1470,1552,600,600,600,600,600,600,600,600,600,60
0,600,600,600,600,600,600,600,600,600,600,600,600,600,600,600,600,600,600,600,600,600,600},

{2420,1500,1500,1500,1500,1500,1500,1500,1500,1500,1500,981,2189,1539,761,1461,1500,2069,1062,150
0,2239,1391,1205,1895,700,1589,1499,1427,1107,2310,1473,1552,600,600,600,600,600,600,600,600,600,60
0,600,600,600,600,600,600,600,600,600,600,600,600,600,600,600,600,600,600,600,600,600,600},

{2420,1500,1500,1500,1500,1500,1500,1500,1500,1500,1500,981,2189,1539,761,1461,1500,2069,1062,150
0,2239,1391,1205,1848,700,1509,1461,1427,1107,2310,1462,1333,600,600,600,600,600,600,600,600,600,60
0,600,600,600,600,600,600,600,600,600,600,600,600,600,600,600,600,600,600,600,600,600,600},

{2420,1500,1500,1500,1500,1500,1500,1500,1500,1500,1500,981,2189,1539,761,1461,1500,2069,1062,150
0,2239,1391,1504,1941,700,1509,1425,1427,1107,2310,1524,1391,600,600,600,600,600,600,600,600,600,60
0,600,600,600,600,600,600,600,600,600,600,600,600,600,600,600,600,600,600,600,600,600,600},

{2500,1500,1500,1500,1500,1500,1500,1500,1500,1500,1500,981,2189,1539,761,1461,1500,2069,1062,150
0,2239,1391,1491,1960,700,1491,1519,1430,1070,2370,1433,1461,600,600,600,600,600,600,600,600,600,60
0,600,600,600,600,600,600,600,600,600,600,600,600,600,600,600,600,600,600,600,600,600,600}
                          };

flash int HomeBaru[1][64] = {

{2420,1500,1500,1500,1500,1500,1500,1500,1500,1500,1500,981,2189,1539,761,1461,1500,2069,1062,150
0,2239,1391,1488,1951,700,1487,1519,1427,1065,2370,1452,1461,1000,1000,1000,1000,1000,1000,1000,10
00,1000,1000,1000,1000,1000,1000,1000,1000,1000,1000,1000,1000,1000,1000,1000,1000,1000,1000,1000,
1000,1000,1000,1000,1000}
                          };

flash int TurnLeft[7][64] = {

{2500,1500,1500,1500,1500,1500,1500,1500,1500,1500,1500,981,2189,1539,761,1461,1500,2069,950,1500,
2239,1391,1550,1910,730,1491,1519,1461,1111,2300,1450,1509,1000,1000,1000,1000,1000,1000,1000,100
0,1000,1000,1000,1000,1000,1000,1000,1000,1000,1000,1000,1000,1000,1000,1000,1000,1000,1000,1000,1
000,1000,1000,1000,1000},

{2500,1500,1500,1500,1500,1500,1500,1500,1500,1500,1500,981,2189,1539,761,1461,1500,2069,950,1500,
2239,1391,1550,1910,730,1491,1421,1461,1111,2300,1450,1364,600,600,600,600,600,600,600,600,600,600,
600,600,600,600,600,600,600,600,600,600,600,600,600,600,600,600,600,600,600,600,600,600},

{2500,1500,1500,1500,1500,1500,1500,1500,1500,1500,1500,981,2189,1539,761,1461,1500,2069,950,1500,
2239,1391,1550,1910,730,1491,1406,1696,1111,2300,1450,1450,600,600,600,600,600,600,600,600,600,600,
600,600,600,600,600,600,600,600,600,600,600,600,600,600,600,600,600,600,600,600,600,600},

{2500,1500,1500,1500,1500,1500,1500,1500,1500,1500,1500,981,2189,1539,761,1461,1500,2069,950,1500,
2239,1391,1550,1910,730,1491,1459,1704,1045,2300,1450,1509,1000,1000,1000,1000,1000,1000,1000,100
0,1000,1000,1000,1000,1000,1000,1000,1000,1000,1000,1000,1000,1000,1000,1000,1000,1000,1000,1000,1
000,1000,1000,1000,1000},

{2500,1500,1500,1500,1500,1500,1500,1500,1500,1500,1500,981,2189,1539,761,1461,1500,2069,950,1500,
2239,1391,1550,1910,730,1491,1693,1754,1112,2345,1485,1585,600,600,600,600,600,600,600,600,600,600,
600,600,600,600,600,600,600,600,600,600,600,600,600,600,600,600,600,600,600,600,600,600},

{2500,1500,1500,1500,1500,1500,1500,1500,1500,1500,1500,981,2189,1539,761,1461,1500,2069,950,1500,
```

```
2239,1391,1550,1910,730,1491,1601,1461,1111,2349,1450,1603,600,600,600,600,600,600,600,600,600,600,
600,600,600,600,600,600,600,600,600,600,600,600,600,600,600,600,600,600,600,600,600,600},

{2500,1500,1500,1500,1500,1500,1500,1500,1500,1500,1500,981,2189,1539,761,1461,1500,2069,950,1500,
2239,1391,1550,1910,730,1491,1519,1461,1111,2300,1450,1509,1000,1000,1000,1000,1000,1000,1000,100
0,1000,1000,1000,1000,1000,1000,1000,1000,1000,1000,1000,1000,1000,1000,1000,1000,1000,1000,1000,1
000,1000,1000,1000,1000}
                        };

flash int Seblak[5][64] = {

{2500,1500,1500,1500,1500,1500,1500,1500,1500,1500,1500,981,2189,1539,761,1461,1500,2069,1062,150
0,2239,1391,1491,1339,1461,1281,1519,1430,1665,1509,1742,1485,1000,1000,1000,1000,1000,1000,1000,1
000,1000,1000,1000,1000,1000,1000,1000,1000,1000,1000,1000,1000,1000,1000,1000,1000,1000,1000,100
0,1000,1000,1000,1000,1000},

{2500,1500,1500,1500,1500,1500,1500,1500,1500,1500,1500,1880,2500,1513,1459,1461,1739,1177,1298,1
500,2239,1391,1491,1307,1461,1281,1418,1430,1663,1509,1680,1300,1000,1000,1000,1000,1000,1000,100
0,1000,1000,1000,1000,1000,1000,1000,1000,1000,1000,1000,1000,1000,1000,1000,1000,1000,1000,1000,1
000,1000,1000,1000,1000,1000},

{2500,1500,1500,1500,1500,1500,1500,1500,1500,1500,1500,981,2189,1539,761,1461,1500,2069,1062,150
0,2239,1391,1491,1307,1461,1281,1406,1430,1663,2108,1232,1379,1000,1000,1000,1000,1000,1000,1000,1
000,1000,1000,1000,1000,1000,1000,1000,1000,1000,1000,1000,1000,1000,1000,1000,1000,1000,1000,100
0,1000,1000,1000,1000,1000},

{2500,1500,1500,1500,1500,1500,1500,1500,1500,1500,1500,1852,1654,1539,761,1461,866,1177,1062,151
3,1487,1391,1491,1307,1461,1281,1515,1430,1663,2108,1232,1472,1000,1000,1000,1000,1000,1000,1000,1
000,1000,1000,1000,1000,1000,1000,1000,1000,1000,1000,1000,1000,1000,1000,1000,1000,1000,1000,100
0,1000,1000,1000,1000,1000},

{2500,1500,1500,1500,1500,1500,1500,1500,1500,1500,1500,981,2189,1539,761,1461,1500,2069,1062,150
0,2239,1391,1491,1322,1518,1180,1854,1430,1693,1688,1609,1615,1000,1000,1000,1000,1000,1000,1000,1
000,1000,1000,1000,1000,1000,1000,1000,1000,1000,1000,1000,1000,1000,1000,1000,1000,1000,1000,100
0,1000,1000,1000,1000,1000}
                        };



flash int Prepare[3][64] = {

{2500,1500,1500,1500,1500,1500,1500,1500,1500,1500,1500,981,2189,1539,761,1461,1500,2069,1062,150
0,2239,1391,1550,1910,730,1491,1519,1461,1111,2300,1450,1509,600,600,600,600,600,600,600,600,600,60
0,600,600,600,600,600,600,600,600,600,600,600,600,600,600,600,600,600,600,600,600,600,600},

{2420,1500,1500,1500,1500,1500,1500,1500,1500,1500,1500,981,2189,1539,761,1461,1500,2069,1062,150
0,2239,1391,1491,1951,700,1582,1672,1430,1065,2285,1503,1598,600,600,600,600,600,600,600,600,600,60
0,600,600,600,600,600,600,600,600,600,600,600,600,600,600,600,600,600,600,600,600,600,600},

{2420,1500,1500,1500,1500,1500,1500,1500,1500,1500,1500,1710,1992,1539,698,1461,1627,1373,1062,15
41,1768,1391,877,1318,1063,1788,1603,1430,1764,1337,1795,1541,2000,2000,2000,2000,2000,2000,2000,2
000,2000,2000,2000,2000,2000,2000,2000,2000,2000,2000,2000,2000,2000,2000,2000,2000,2000,2000,200
0,2000,2000,2000,2000,2000}
                        };

flash int KlatBahu[2][64] = {

{2420,1500,1500,1500,1500,1500,1500,1500,1500,1500,1500,1964,1992,1539,698,1461,1373,1598,1062,15
41,1768,1391,877,1318,1063,1788,1603,1430,1764,1337,1795,1640,400,400,400,400,400,400,400,400,400,4
00,400,400,400,400,400,400,400,400,400,400,400,400,400,400,400,400,400,400,400,400,400,400},

{2420,1500,1500,1500,1500,1500,1500,1500,1500,1500,1500,1513,1992,1539,1430,1461,1627,1120,1062,1
541,2245,1391,877,1318,1063,1788,1603,1430,1764,1365,1795,1290,600,600,600,600,600,600,600,600,600,
600,600,600,600,600,600,600,600,600,600,600,600,600,600,600,600,600,600,600,600,600,600,600}
```

```
                        };

flash int NguraiRikmo[2][64] = {

{2420,1500,1500,1500,1500,1500,1500,1500,1500,1500,1500,1373,1992,1539,1459,1461,1627,500,1062,15
41,1992,1391,877,1318,1063,1788,1603,1430,1764,1337,1795,1541,1200,1200,1200,1200,1200,1200,1200,1
200,1200,1200,1200,1200,1200,1200,1200,1200,1200,1200,1200,1200,1200,1200,1200,1200,1200,1200,120
0,1200,1200,1200,1200,1200},

{2420,1500,1500,1500,1500,1500,1500,1500,1500,1500,1500,2500,2218,1459,1346,1461,1373,1823,1062,1
541,1992,1391,877,1318,1063,1788,1603,1430,1764,1337,1795,1541,1200,1200,1200,1200,1200,1200,1200,
1200,1200,1200,1200,1200,1200,1200,1200,1200,1200,1200,1200,1200,1200,1200,1200,1200,1200,1200,12
00,1200,1200,1200,1200,1200}
                        };

flash int Berhias[6][64] = {

{2420,1500,1500,1500,1500,1500,1500,1500,1500,1500,1500,1373,1992,1539,1459,1461,1823,838,1107,15
41,1768,1391,877,1318,1063,1788,1603,1430,1764,1337,1795,1541,1000,1000,1000,1000,1000,1000,1000,1
000,1000,1000,1000,1000,1000,1000,1000,1000,1000,1000,1000,1000,1000,1000,1000,1000,1000,1000,100
0,1000,1000,1000,1000,1000},

{2420,1500,1500,1500,1500,1500,1500,1500,1500,1500,1500,1373,1992,1539,1459,1461,1570,922,937,154
1,1768,1391,877,1318,1063,1788,1603,1430,1764,1337,1795,1541,1000,1000,1000,1000,1000,1000,1000,10
00,1000,1000,1000,1000,1000,1000,1000,1000,1000,1000,1000,1000,1000,1000,1000,1000,1000,1000,1000,
1000,1000,1000,1000,1000},

{2420,1500,1500,1500,1500,1500,1500,1500,1500,1500,1500,1373,1992,1539,1459,1461,1852,500,1021,15
41,1768,1391,877,1318,1063,1788,1603,1430,1764,1337,1795,1541,1000,1000,1000,1000,1000,1000,1000,1
000,1000,1000,1000,1000,1000,1000,1000,1000,1000,1000,1000,1000,1000,1000,1000,1000,1000,1000,100
0,1000,1000,1000,1000,1000},

{2420,1500,1500,1500,1500,1500,1500,1500,1500,1500,1500,1373,1992,1539,1459,1461,1852,838,1277,15
41,1964,1391,877,1318,1063,1788,1603,1430,1764,1337,1795,1541,1000,1000,1000,1000,1000,1000,1000,1
000,1000,1000,1000,1000,1000,1000,1000,1000,1000,1000,1000,1000,1000,1000,1000,1000,1000,1000,100
0,1000,1000,1000,1000,1000},

{2420,1500,1500,1500,1500,1500,1500,1500,1500,1500,1500,1373,1992,1539,1459,1461,1739,1148,1000,1
541,1739,1391,877,1318,1063,1788,1603,1430,1764,1337,1795,1541,1000,1000,1000,1000,1000,1000,1000,
1000,1000,1000,1000,1000,1000,1000,1000,1000,1000,1000,1000,1000,1000,1000,1000,1000,1000,1000,10
00,1000,1000,1000,1000,1000},

{2420,1500,1500,1500,1500,1500,1500,1500,1500,1500,1500,1373,1992,1539,1459,1461,1739,980,1043,15
41,1654,1391,877,1318,1063,1788,1603,1430,1764,1337,1795,1541,1000,1000,1000,1000,1000,1000,1000,1
000,1000,1000,1000,1000,1000,1000,1000,1000,1000,1000,1000,1000,1000,1000,1000,1000,1000,1000,100
0,1000,1000,1000,1000,1000}
                        };

flash int Gaya[64] =
{2420,1500,1500,1500,1500,1500,1500,1500,1500,1500,1500,1816,2164,1539,1380,1461,1937,1290,1710,1
541,2302,1391,877,1439,985,1788,1548,1430,1692,1402,1795,1524,1000,1000,1000,1000,1000,1000,1000,1
000,1000,1000,1000,1000,1000,1000,1000,1000,1000,1000,1000,1000,1000,1000,1000,1000,1000,1000,100
0,1000,1000,1000,1000,1000};

flash int GeserKiri[5][64] = {

{2500,1500,1500,1500,1500,1500,1500,1500,1500,1500,1500,981,2189,1539,761,1461,1500,2069,952,1500,
2239,1391,1518,1877,730,1491,1519,1461,1064,2462,1411,1509,1000,1000,1000,1000,1000,1000,1000,100
0,1000,1000,1000,1000,1000,1000,1000,1000,1000,1000,1000,1000,1000,1000,1000,1000,1000,1000,1000,1
000,1000,1000,1000,1000},

{2500,1500,1500,1500,1500,1500,1500,1500,1500,1500,1500,981,2189,1539,761,1461,1500,2069,952,1500,
2239,1391,1518,1908,730,1491,1410,1461,1064,2462,1411,1352,500,500,500,500,500,500,500,500,500,500,
500,500,500,500,500,500,500,500,500,500,500,500,500,500,500,500,500,500,500,500,500,500},
```

```
{2500,1500,1500,1500,1500,1500,1500,1500,1500,1500,1500,981,2189,1539,761,1461,1500,2069,952,1500,
2239,1391,1327,1851,730,1491,1410,1461,1064,2462,1411,1266,1000,1000,1000,1000,1000,1000,1000,100
0,1000,1000,1000,1000,1000,1000,1000,1000,1000,1000,1000,1000,1000,1000,1000,1000,1000,1000,1000,1
000,1000,1000,1000,1000},

{2500,1500,1500,1500,1500,1500,1500,1500,1500,1500,1500,981,2189,1539,761,1461,1500,2069,952,1500,
2239,1391,1327,1851,730,1533,1700,1461,1064,2462,1411,1595,1000,1000,1000,1000,1000,1000,1000,100
0,1000,1000,1000,1000,1000,1000,1000,1000,1000,1000,1000,1000,1000,1000,1000,1000,1000,1000,1000,1
000,1000,1000,1000,1000},

{2500,1500,1500,1500,1500,1500,1500,1500,1500,1500,1500,981,2189,1539,761,1461,1500,2069,952,1500,
2239,1391,1518,1877,730,1491,1519,1461,1064,2462,1411,1509,700,700,700,700,700,700,700,700,700,700,
700,700,700,700,700,700,700,700,700,700,700,700,700,700,700,700,700,700,700,700,700,700}
                          };

flash int GeserKanan[6][64] = {

{2500,1500,1500,1500,1500,1500,1500,1500,1500,1500,1500,981,2189,1539,761,1461,1500,2069,952,1500,
2239,1391,1518,1877,730,1491,1519,1461,1064,2397,1411,1509,1000,1000,1000,1000,1000,1000,1000,100
0,1000,1000,1000,1000,1000,1000,1000,1000,1000,1000,1000,1000,1000,1000,1000,1000,1000,1000,1000,1
000,1000,1000,1000,1000},

{2500,1500,1500,1500,1500,1500,1500,1500,1500,1500,1500,981,2189,1539,761,1461,1500,2069,952,1500,
2239,1391,1518,1877,730,1491,1689,1461,1087,2372,1411,1609,1000,1000,1000,1000,1000,1000,1000,100
0,1000,1000,1000,1000,1000,1000,1000,1000,1000,1000,1000,1000,1000,1000,1000,1000,1000,1000,1000,1
000,1000,1000,1000,1000},

{2500,1500,1500,1500,1500,1500,1500,1500,1500,1500,1500,981,2189,1539,761,1461,1500,2069,952,1500,
2239,1391,1518,1911,730,1560,1689,1636,1165,2368,1411,1609,1000,1000,1000,1000,1000,1000,1000,100
0,1000,1000,1000,1000,1000,1000,1000,1000,1000,1000,1000,1000,1000,1000,1000,1000,1000,1000,1000,1
000,1000,1000,1000,1000},

{2500,1500,1500,1500,1500,1500,1500,1500,1500,1500,1500,981,2189,1539,761,1461,1500,2069,952,1500,
2239,1391,1518,1959,730,1442,1554,1636,1108,2397,1411,1503,1000,1000,1000,1000,1000,1000,1000,100
0,1000,1000,1000,1000,1000,1000,1000,1000,1000,1000,1000,1000,1000,1000,1000,1000,1000,1000,1000,1
000,1000,1000,1000,1000},

{2500,1500,1500,1500,1500,1500,1500,1500,1500,1500,1500,981,2189,1539,761,1461,1500,2069,952,1500,
2239,1391,1518,2011,730,1442,1409,1636,1060,2414,1400,1310,1000,1000,1000,1000,1000,1000,1000,100
0,1000,1000,1000,1000,1000,1000,1000,1000,1000,1000,1000,1000,1000,1000,1000,1000,1000,1000,1000,1
000,1000,1000,1000,1000},

{2500,1500,1500,1500,1500,1500,1500,1500,1500,1500,1500,981,2189,1539,761,1461,1500,2069,952,1500,
2239,1391,1518,1877,730,1491,1519,1461,1064,2397,1411,1509,1000,1000,1000,1000,1000,1000,1000,100
0,1000,1000,1000,1000,1000,1000,1000,1000,1000,1000,1000,1000,1000,1000,1000,1000,1000,1000,1000,1
000,1000,1000,1000,1000}
                          };

flash int TrecekEnd[4][64] = {

{2420,1500,1500,1500,1500,1500,1500,1500,1500,1500,1500,1710,1598,1539,557,1461,1008,1373,1063,15
41,1520,1391,877,1439,985,1788,1548,1430,1692,1402,1795,1524,600,600,600,600,600,600,600,600,600,60
0,600,600,600,600,600,600,600,600,600,600,600,600,600,600,600,600,600,600,600,600,600,600},

{2420,1500,1500,1500,1500,1500,1500,1500,1500,1500,1500,1816,2164,1539,1380,1461,1627,1290,1710,1
541,2302,1391,877,1541,985,1760,1548,1430,1692,1461,1795,1673,1000,1000,1000,1000,1000,1000,1000,1
000,1000,1000,1000,1000,1000,1000,1000,1000,1000,1000,1000,1000,1000,1000,1000,1000,1000,1000,100
0,1000,1000,1000,1000,1000},

{2500,1500,1500,1500,1500,1500,1500,1500,1500,1500,1500,1360,2189,1539,761,1461,1290,1663,1063,15
00,2239,1391,1550,1910,730,1491,1594,1461,1111,2314,1450,1597,1000,1000,1000,1000,1000,1000,1000,1
000,1000,1000,1000,1000,1000,1000,1000,1000,1000,1000,1000,1000,1000,1000,1000,1000,1000,1000,100
0,1000,1000,1000,1000,1000},
```

```
{2550,1500,1500,1500,1500,1500,1500,1500,1500,1500,1500,981,2189,1539,761,1461,1500,2069,950,1500,
2239,1391,1550,1910,730,1491,1519,1461,1111,2300,1450,1509,500,500,500,500,500,500,500,500,500,500,
500,500,500,500,500,500,500,500,500,500,500,500,500,500,500,500,500,500,500,500,500,500}
                        };


flash int TrecekKiri[3][64] = {

{2500,1500,1500,1500,1500,1500,1500,1500,1500,1500,1500,1816,2164,1539,1380,1461,1500,1290,1710,1
541,2302,1391,1550,1756,1261,1104,1408,2091,1294,2077,1421,1509,1000,1000,1000,1000,1000,1000,100
0,1000,1000,1000,1000,1000,1000,1000,1000,1000,1000,1000,1000,1000,1000,1000,1000,1000,1000,1
000,1000,1000,1000,1000,1000},

{2500,1500,1500,1500,1500,1500,1500,1500,1500,1500,1500,1816,2164,1539,1380,1461,1500,1290,1710,1
541,2302,1391,1550,1795,1261,1104,1603,2091,1557,2128,1248,1509,2000,2000,2000,2000,2000,2000,200
0,2000,2000,2000,2000,2000,2000,2000,2000,2000,2000,2000,2000,2000,2000,2000,2000,2000,2000,2
000,2000,2000,2000,2000,2000},

{2500,1500,1500,1500,1500,1500,1500,1500,1500,1500,1500,1816,2164,1539,1380,1461,1500,1290,1710,1
541,2302,1391,1550,1795,1261,1104,1730,2091,1557,1600,1788,1509,1000,1000,1000,1000,1000,1000,100
0,1000,1000,1000,1000,1000,1000,1000,1000,1000,1000,1000,1000,1000,1000,1000,1000,1000,1000,1
000,1000,1000,1000,1000,1000}
                        };
flash int TrecekKanan[3][64] = {

{2500,1500,1500,1500,1500,1500,1500,1500,1500,1500,1500,1816,2164,1539,1380,1461,1500,1290,1710,1
541,2302,1391,1550,1795,1261,1104,1603,2091,1557,2128,1248,1509,2000,2000,2000,2000,2000,2000,200
0,2000,2000,2000,2000,2000,2000,2000,2000,2000,2000,2000,2000,2000,2000,2000,2000,2000,2000,2
000,2000,2000,2000,2000,2000},

{2500,1500,1500,1500,1500,1500,1500,1500,1500,1500,1500,1816,2164,1539,1380,1461,1500,1290,1710,1
541,2302,1391,1550,1795,1261,1104,1730,2091,1557,1600,1788,1509,2000,2000,2000,2000,2000,2000,200
0,2000,2000,2000,2000,2000,2000,2000,2000,2000,2000,2000,2000,2000,2000,2000,2000,2000,2000,2
000,2000,2000,2000,2000,2000},

{2500,1500,1500,1500,1500,1500,1500,1500,1500,1500,1500,1816,2164,1539,1380,1461,1500,1290,1710,1
541,2302,1391,1550,1795,1261,1104,1408,2091,1294,2077,1421,1509,1000,1000,1000,1000,1000,1000,100
0,1000,1000,1000,1000,1000,1000,1000,1000,1000,1000,1000,1000,1000,1000,1000,1000,1000,1000,1
000,1000,1000,1000,1000,1000}
                        };

flash int PrepareTrecekKiri[4][64] = {

{2500,1500,1500,1500,1500,1500,1500,1500,1500,1500,1500,981,2189,1539,761,1461,1500,2069,950,1500,
2239,1391,1550,1910,730,1491,1519,1461,1111,2300,1450,1509,1000,1000,1000,1000,1000,1000,1000,100
0,1000,1000,1000,1000,1000,1000,1000,1000,1000,1000,1000,1000,1000,1000,1000,1000,1000,1000,1
000,1000,1000,1000,1000},

{2500,1500,1500,1500,1500,1500,1500,1500,1500,1500,1500,1816,2164,1539,1380,1461,1500,1290,1710,1
541,2302,1391,1550,1866,730,1516,1408,1461,1111,2339,1450,1294,1000,1000,1000,1000,1000,1000,1000,
1000,1000,1000,1000,1000,1000,1000,1000,1000,1000,1000,1000,1000,1000,1000,1000,1000,1000,1000,10
00,1000,1000,1000,1000,1000},

{2500,1500,1500,1500,1500,1500,1500,1500,1500,1500,1500,1816,2164,1539,1380,1461,1500,1290,1710,1
541,2302,1391,1550,1853,730,1516,1397,2091,1183,2233,1450,1509,1000,1000,1000,1000,1000,1000,1000,
1000,1000,1000,1000,1000,1000,1000,1000,1000,1000,1000,1000,1000,1000,1000,1000,1000,1000,1000,10
00,1000,1000,1000,1000,1000},

{2500,1500,1500,1500,1500,1500,1500,1500,1500,1500,1500,1816,2164,1539,1380,1461,1500,1290,1710,1
541,2302,1391,1550,1756,1261,1104,1408,2091,1294,2077,1421,1509,1000,1000,1000,1000,1000,1000,100
0,1000,1000,1000,1000,1000,1000,1000,1000,1000,1000,1000,1000,1000,1000,1000,1000,1000,1000,1
000,1000,1000,1000,1000,1000}
```

```
};

flash int PrepareKanan[4][64] = {

{2500,1500,1500,1500,1500,1500,1500,1500,1500,1500,1500,981,2189,1539,761,1461,1500,2069,950,1500,
2239,1391,1550,1910,730,1491,1519,1461,1111,2300,1450,1509,1000,1000,1000,1000,1000,1000,1000,100
0,1000,1000,1000,1000,1000,1000,1000,1000,1000,1000,1000,1000,1000,1000,1000,1000,1000,1000,1000,1
000,1000,1000,1000,1000},

{2500,1500,1500,1500,1500,1500,1500,1500,1500,1500,1500,1816,2164,1539,1380,1461,1500,1290,1710,1
541,2302,1391,1550,1866,730,1516,1408,1461,1111,2339,1450,1294,1000,1000,1000,1000,1000,1000,1000,
1000,1000,1000,1000,1000,1000,1000,1000,1000,1000,1000,1000,1000,1000,1000,1000,1000,1000,1000,10
00,1000,1000,1000,1000,1000},

{2500,1500,1500,1500,1500,1500,1500,1500,1500,1500,1500,1816,2164,1539,1380,1461,1500,1290,1710,1
541,2302,1391,1550,1853,730,1516,1397,2091,1183,2233,1450,1509,1000,1000,1000,1000,1000,1000,1000,
1000,1000,1000,1000,1000,1000,1000,1000,1000,1000,1000,1000,1000,1000,1000,1000,1000,1000,1000,10
00,1000,1000,1000,1000,1000},

{2500,1500,1500,1500,1500,1500,1500,1500,1500,1500,1500,1816,2164,1539,1380,1461,1500,1290,1710,1
541,2302,1391,1550,1795,1261,1104,1408,2091,1294,2077,1421,1509,1000,1000,1000,1000,1000,1000,100
0,1000,1000,1000,1000,1000,1000,1000,1000,1000,1000,1000,1000,1000,1000,1000,1000,1000,1000,1000,1
000,1000,1000,1000,1000,1000}
                         };

flash int ZonaTengah[25][64] = {

{2500,1500,1500,1500,1500,1500,1500,1500,1500,1500,1500,981,2189,1539,761,1461,1500,2069,950,1500,
2239,1391,1550,1910,730,1491,1519,1461,1111,2300,1450,1509,1000,1000,1000,1000,1000,1000,1000,100
0,1000,1000,1000,1000,1000,1000,1000,1000,1000,1000,1000,1000,1000,1000,1000,1000,1000,1000,1000,1
000,1000,1000,1000,1000},

{2420,1500,1500,1500,1500,1500,1500,1500,1500,1500,1500,981,2189,1539,761,1461,1500,2069,1063,150
0,2239,1391,1491,1951,700,1585,1672,1430,1113,2285,1503,1598,600,600,600,600,600,600,600,600,600,60
0,600,600,600,600,600,600,600,600,600,600,600,600,600,600,600,600,600,600,600,600,600,600},

{2420,1500,1500,1500,1500,1500,1500,1500,1500,1500,1500,1880,2387,1539,755,1461,1500,1205,1063,15
41,2162,1391,877,1318,1063,1788,1603,1430,1764,1337,1795,1541,2000,2000,2000,2000,2000,2000,2000,2
000,2000,2000,2000,2000,2000,2000,2000,2000,2000,2000,2000,2000,2000,2000,2000,2000,2000,2000,200
0,2000,2000,2000,2000,2000},

{2420,1500,1500,1500,1500,1500,1500,1500,1500,1500,1500,1880,2275,1539,1627,1461,1500,1205,811,15
41,2162,1391,877,1318,1063,1788,1603,1430,1764,1337,1795,1541,400,400,400,400,400,400,400,400,400,4
00,400,400,400,400,400,400,400,400,400,400,400,400,400,400,400,400,400,400,400,400,400,400},

{2420,1500,1500,1500,1500,1500,1500,1500,1500,1500,1500,1880,2245,1908,1654,1461,1500,1487,1063,1
541,2162,1391,877,1318,1063,1788,1603,1430,1764,1337,1795,1541,600,600,600,600,600,600,600,600,600,
600,600,600,600,600,600,600,600,600,600,600,600,600,600,600,600,600,600,600,600,600,600,600},

{2420,1500,1500,1500,1500,1500,1500,1500,1500,1500,1500,1880,2387,1539,755,1461,1500,1205,1063,15
41,2162,1391,877,1318,1063,1788,1603,1430,1764,1337,1795,1541,600,600,600,600,600,600,600,600,600,6
00,600,600,600,600,600,600,600,600,600,600,600,600,600,600,600,600,600,600,600,600,600,600},

{2420,1500,1500,1500,1500,1500,1500,1500,1500,1500,1500,1880,2500,1539,755,1461,1500,1205,980,154
1,1318,1391,877,1318,1063,1788,1603,1430,1764,1337,1795,1541,600,600,600,600,600,600,600,600,600,60
0,600,600,600,600,600,600,600,600,600,600,600,600,600,600,600,600,600,600,600,600,600,600},

{2420,1500,1500,1500,1500,1500,1500,1500,1500,1500,1500,1880,2331,1539,1036,1461,1500,1205,980,98
0,1318,1391,877,1318,1063,1788,1603,1430,1764,1337,1795,1541,600,600,600,600,600,600,600,600,600,60
0,600,600,600,600,600,600,600,600,600,600,600,600,600,600,600,600,600,600,600,600,600,600},

{2420,1500,1500,1500,1500,1500,1500,1500,1500,1500,1500,2500,2218,1459,1346,1461,1500,1823,1063,1
```

541,1992,1391,877,1318,1063,1788,1603,1430,1764,1337,1795,1541,600,600,600,600,600,600,600,600,600,
600,600,600,600,600,600,600,600,600,600,600,600,600,600,600,600,600,600,600,600,600,600,600,600},

{2420,1500,1500,1500,1500,1500,1500,1500,1500,1500,1500,1880,2387,1539,755,1461,1500,1205,1063,15
41,2162,1391,877,1318,1063,1788,1603,1430,1764,1337,1795,1541,600,600,600,600,600,600,600,600,600,6
00,600,600,600,600,600,600,600,600,600,600,600,600,600,600,600,600,600,600,600,600,600,600,600},

{2420,1500,1500,1500,1500,1500,1500,1500,1500,1500,1500,1430,2387,1539,755,1461,1232,1205,942,154
1,1402,1391,877,1318,1063,1788,1603,1430,1764,1337,1795,1541,600,600,600,600,600,600,600,600,600,60
0,600,600,600,600,600,600,600,600,600,600,600,600,600,600,600,600,600,600,600,600,600,600,600},

{2420,1500,1500,1500,1500,1500,1500,1500,1500,1500,1500,1430,2387,1539,755,1461,1232,528,942,1541,
1654,1391,877,1318,1063,1788,1603,1430,1764,1337,1795,1541,600,600,600,600,600,600,600,600,600,600,
600,600,600,600,600,600,600,600,600,600,600,600,600,600,600,600,600,600,600,600,600,600,600},

{2420,1500,1500,1500,1500,1500,1500,1500,1500,1500,1500,1590,2302,1539,1541,1461,1430,1177,942,15
41,1654,1391,877,1318,1063,1788,1603,1430,1764,1337,1795,1541,600,600,600,600,600,600,600,600,600,6
00,600,600,600,600,600,600,600,600,600,600,600,600,600,600,600,600,600,600,600,600,600,600,600},

{2420,1500,1500,1500,1500,1500,1500,1500,1500,1500,1500,1860,2358,1539,1541,1461,1682,1627,942,15
41,2500,1391,877,1318,1063,1788,1603,1430,1764,1337,1795,1541,600,600,600,600,600,600,600,600,600,6
00,600,600,600,600,600,600,600,600,600,600,600,600,600,600,600,600,600,600,600,600,600,600,600},

{2420,1500,1500,1500,1500,1500,1500,1500,1500,1500,1500,2500,2358,1539,1430,1461,1682,1627,942,15
41,2500,1391,877,1318,1063,1788,1603,1430,1764,1337,1795,1541,600,600,600,600,600,600,600,600,600,6
00,600,600,600,600,600,600,600,600,600,600,600,600,600,600,600,600,600,600,600,600,600,600,600},

{2420,1500,1500,1500,1500,1500,1500,1500,1500,1500,1500,1880,2387,1539,755,1461,1500,1205,1063,15
41,2162,1391,877,1318,1063,1788,1603,1430,1764,1337,1795,1541,600,600,600,600,600,600,600,600,600,6
00,600,600,600,600,600,600,600,600,600,600,600,600,600,600,600,600,600,600,600,600,600,600,600},

{2420,1500,1500,1500,1500,1500,1500,1500,1500,1500,1500,1880,2387,1539,755,1461,1500,1205,1063,15
41,2162,1391,877,1318,1063,1788,1603,1430,1764,1337,1795,1541,600,600,600,600,600,600,600,600,600,6
00,600,600,600,600,600,600,600,600,600,600,600,600,600,600,600,600,600,600,600,600,600,600,600},

{2420,1500,1500,1500,1500,1500,1500,1500,1500,1500,1500,1880,2189,1539,1468,1461,1500,1205,980,15
41,1346,1391,877,1318,1063,1788,1603,1430,1764,1337,1795,1541,600,600,600,600,600,600,600,600,600,6
00,600,600,600,600,600,600,600,600,600,600,600,600,600,600,600,600,600,600,600,600,600,600,600},

{2420,1500,1500,1500,1500,1500,1500,1500,1500,1500,1500,1880,2189,1539,1468,1461,1290,1205,980,15
41,1346,1391,877,1318,1063,1788,1603,1430,1764,1337,1795,1541,1000,1000,1000,1000,1000,1000,1000,1
000,1000,1000,1000,1000,1000,1000,1000,1000,1000,1000,1000,1000,1000,1000,1000,1000,1000,1000,100
0,1000,1000,1000,1000,1000},

{2420,1500,1500,1500,1500,1500,1500,1500,1500,1500,1500,1880,2189,1539,1468,1461,1682,1205,980,15
41,1346,1391,877,1318,1063,1788,1603,1430,1764,1337,1795,1541,1000,1000,1000,1000,1000,1000,1000,1
000,1000,1000,1000,1000,1000,1000,1000,1000,1000,1000,1000,1000,1000,1000,1000,1000,1000,1000,100
0,1000,1000,1000,1000,1000},

{2420,1500,1500,1500,1500,1500,1500,1500,1500,1500,1500,2048,2189,1539,1318,1461,1290,917,980,154
1,1513,1391,877,1318,1063,1788,1603,1430,1764,1337,1795,1541,1000,1000,1000,1000,1000,1000,1000,10
00,1000,1000,1000,1000,1000,1000,1000,1000,1000,1000,1000,1000,1000,1000,1000,1000,1000,1000,1000,
1000,1000,1000,1000,1000},

{2420,1500,1500,1500,1500,1500,1500,1500,1500,1500,1500,2048,2189,1539,1318,1461,1710,917,980,154
1,1513,1391,877,1318,1063,1788,1603,1430,1764,1337,1795,1541,1000,1000,1000,1000,1000,1000,1000,10
00,1000,1000,1000,1000,1000,1000,1000,1000,1000,1000,1000,1000,1000,1000,1000,1000,1000,1000,1000,
1000,1000,1000,1000,1000},

{2420,1500,1500,1500,1500,1500,1500,1500,1500,1500,1500,1768,2189,1539,1205,1461,1710,1261,923,15
41,1513,1391,877,1318,1063,1788,1603,1430,1764,1337,1795,1541,1000,1000,1000,1000,1000,1000,1000,1
000,1000,1000,1000,1000,1000,1000,1000,1000,1000,1000,1000,1000,1000,1000,1000,1000,1000,1000,100
0,1000,1000,1000,1000,1000},

```
{2420,1500,1500,1500,1500,1500,1500,1500,1500,1500,1500,2020,2189,1539,1459,1461,1710,866,923,154
1,2106,1391,877,1318,1063,1788,1603,1430,1764,1337,1795,1541,1000,1000,1000,1000,1000,1000,1000,10
00,1000,1000,1000,1000,1000,1000,1000,1000,1000,1000,1000,1000,1000,1000,1000,1000,1000,1000,1000,
1000,1000,1000,1000,1000},

{2420,1500,1500,1500,1500,1500,1500,1500,1500,1500,1500,2020,2189,1539,1063,1461,1710,866,923,154
1,1739,1391,877,1318,1063,1788,1603,1430,1764,1337,1795,1541,1000,1000,1000,1000,1000,1000,1000,10
00,1000,1000,1000,1000,1000,1000,1000,1000,1000,1000,1000,1000,1000,1000,1000,1000,1000,1000,1000,
1000,1000,1000,1000,1000}
                            };


flash int JalanJongkok[5][64] = {

{2500,1500,1500,1500,1500,1500,1500,1500,1500,1500,1500,981,2189,1539,761,1461,1500,2069,1062,150
0,2239,1391,1491,1339,1461,1281,1519,1430,1665,1509,1742,1485,1000,1000,1000,1000,1000,1000,1000,1
000,1000,1000,1000,1000,1000,1000,1000,1000,1000,1000,1000,1000,1000,1000,1000,1000,1000,1000,100
0,1000,1000,1000,1000,1000},

{2500,1500,1500,1500,1500,1500,1500,1500,1500,1500,1500,981,2189,1539,761,1461,1500,2069,1062,150
0,2239,1391,1491,1307,1461,1281,1418,1430,1663,1509,1680,1300,1000,1000,1000,1000,1000,1000,1000,1
000,1000,1000,1000,1000,1000,1000,1000,1000,1000,1000,1000,1000,1000,1000,1000,1000,1000,1000,100
0,1000,1000,1000,1000,1000},

{2500,1500,1500,1500,1500,1500,1500,1500,1500,1500,1500,981,2189,1539,761,1461,1500,2069,1062,150
0,2239,1391,1491,1307,1461,1281,1406,1430,1663,2108,1232,1379,1000,1000,1000,1000,1000,1000,1000,1
000,1000,1000,1000,1000,1000,1000,1000,1000,1000,1000,1000,1000,1000,1000,1000,1000,1000,1000,100
0,1000,1000,1000,1000,1000},

{2500,1500,1500,1500,1500,1500,1500,1500,1500,1500,1500,981,2189,1539,761,1461,1500,2069,1062,150
0,2239,1391,1491,1307,1461,1281,1515,1430,1663,2108,1232,1472,1000,1000,1000,1000,1000,1000,1000,1
000,1000,1000,1000,1000,1000,1000,1000,1000,1000,1000,1000,1000,1000,1000,1000,1000,1000,1000,100
0,1000,1000,1000,1000,1000},

{2500,1500,1500,1500,1500,1500,1500,1500,1500,1500,1500,981,2189,1539,761,1461,1500,2069,1062,150
0,2239,1391,1491,1322,1518,1180,1854,1430,1693,1688,1609,1615,1000,1000,1000,1000,1000,1000,1000,1
000,1000,1000,1000,1000,1000,1000,1000,1000,1000,1000,1000,1000,1000,1000,1000,1000,1000,1000,100
0,1000,1000,1000,1000,1000}
                            };
// Get a character from the USART1 Receiver
#pragma used+
char getchar1(void)
{
char status,data;
while (1)
     {
     while (((status=UCSR1A) & RX_COMPLETE)==0);
     data=UDR1;
     if ((status & (FRAMING_ERROR | PARITY_ERROR | DATA_OVERRUN))==0)
        return data;
     };
}
#pragma used-

// Write a character to the USART1 Transmitter
#pragma used+
void putchar1(char c)
{
while ((UCSR1A & DATA_REGISTER_EMPTY)==0);
UDR1=c;
}
#pragma used-
```

```
void SendUSART1(char text[]){
int i=0;
for (i=0; i<strlen(text); i++)
{
	putchar1(text[i]);
}
}

void Movement(flash int position[])
{
unsigned char i,Pin;
unsigned int Time;
int LastValue = 0;
char text[32];
for(i=0; i<32; i++)
{
	Pin = i;
	Time = position[i+32];
	sprintf(text,"#%d P%d T%d \r", Pin, position[i], Time);
	SendUSART1(text);
	if(Time > LastValue){
		LastValue = Time;
		}
}
delay_ms(LastValue);
}

void GoHome()
{
	 lcd_clear();
	 sprintf(LCD,"Home Position\n%d",LineSensor());
	 lcd_puts(LCD);
	 delay_ms(100);

	 Movement(HomePosition[0]);
}

void GoHomeBaru()
{
	 lcd_clear();
	 sprintf(LCD,"I'm CHRENJI\n From MCU%d",LineSensor());
	 lcd_puts(LCD);
	 delay_ms(100);

	 Movement(HomeBaru[0]);
}


void GoJalanJongkok()
{
	unsigned char i;

	 lcd_clear();
	 sprintf(LCD,"Jalan Jongkok\n%d",LineSensor());
	 lcd_puts(LCD);
	 delay_ms(100);

	for(i=0;i<5;i++)
	{
		Movement(JalanJongkok[i]);

	}
}
```

```
void GoBelokKiri()
{
	unsigned char i;

	 lcd_clear();
	 sprintf(LCD,"Belok Kiri\n%d",LineSensor());
	 lcd_puts(LCD);
	 delay_ms(100);

	for(i=0;i<5;i++)
	{
		Movement(BelokKiri[i]);

	}
}

void GoBelokKanan()
{
	unsigned char i;

	 lcd_clear();
	 sprintf(LCD,"Belok Kanan\n%d",LineSensor());
	 lcd_puts(LCD);
	 delay_ms(100);

	for(i=0;i<6;i++)
	{
		Movement(BelokKanan[i]);

	}
}

void GoLeft()
{
	 unsigned char i;

	 lcd_clear();
	 sprintf(LCD,"Turn Left\n%d",LineSensor());
	 lcd_puts(LCD);
	 delay_ms(20);

	 for(i=0;i<7;i++)
	 {
		Movement(TurnLeft[i]);
	 }
	 LineSensorValue = LineSensor();
}

// void GoRight()
// {
//	unsigned char i;
//
//	lcd_clear();
//	sprintf(LCD,"Turn Right\n%d",LineSensor());
//	lcd_puts(LCD);
//	delay_ms(20);
//
//	for(i=0;i<7;i++)
//	{
//		Movement(TurnRight[i]);
//	}
//	LineSensorValue = LineSensor();
// }
```

```
void GoPrepare()
{
	unsigned char i;

	 lcd_clear();
	 sprintf(LCD,"Kuda-Kuda\n%d",LineSensor());
	 lcd_puts(LCD);
	 delay_ms(100);

	for(i=0;i<3;i++)
	{
		Movement(Prepare[i]);
	}
}

void GoTrecekEnd()
{
	unsigned char i;

	 lcd_clear();
	 sprintf(LCD,"Selesai Trecek\n%d",LineSensor());
	 lcd_puts(LCD);
	 delay_ms(100);

	for(i=0;i<4;i++)
	{
		Movement(TrecekEnd[i]);
	}
}

void GoSalamPembuka()
{
	unsigned char i;

	 lcd_clear();
	 sprintf(LCD,"hi I'm CHRENJI\n%d",LineSensor());
	 lcd_puts(LCD);
	 delay_ms(100);

	for(i=0;i<7;i++)
	{
		Movement(SalamPembuka[i]);
	}
}

void GoSalamPenutup()
{
	unsigned char i;

	 lcd_clear();
	 sprintf(LCD,"see you soon\n%d",LineSensor());
	 lcd_puts(LCD);
	 delay_ms(100);

	for(i=0;i<6;i++)
	{
		Movement(SalamPenutup[i]);
	}
}

void GoTanjakKanan()
{
	unsigned char i;
```

```
        lcd_clear();
        sprintf(LCD,"Tanjak Kanan\n%d",LineSensor());
        lcd_puts(LCD);
        delay_ms(100);

       for(i=0;i<7;i++)
       {
              Movement(TanjakKanan[i]);
       }
}

void GoSeblak()
{
        unsigned char i;

        lcd_clear();
        sprintf(LCD,"Seblak\n%d",LineSensor());
        lcd_puts(LCD);
        delay_ms(100);

        for(i=0;i<10;i++)
        {
              Movement(Seblak[i]);
              if(i == 2 || i == 6)
              {
                     delay_ms(800);
                     if(i == 6)
                     {
                            LineSensorValue = LineSensor();
                     }
              }
        }
}


void GoTrecekKanan()
{
       unsigned char i;

        lcd_clear();
        sprintf(LCD,"Trecek Kanan\n%d",LineSensor());
        lcd_puts(LCD);
        delay_ms(100);

       for(i=0;i<3;i++)
       {
              Movement(TrecekKanan[i]);
              if(i == 3)
              {
                     delay_ms(500);
              }
       }
}

void GoPrepareTrecekKanan()
{
       unsigned char i;

        lcd_clear();
        sprintf(LCD,"Trecek Kanan\n%d",LineSensor());
        lcd_puts(LCD);
        delay_ms(100);

       for(i=0;i<4;i++)
```

```
	{
		Movement(PrepareKanan[i]);
	}
}

void GoTrecekKiri()
{
	unsigned char i;

	 lcd_clear();
	 sprintf(LCD,"Trecek Kiri\n%d",LineSensor());
	 lcd_puts(LCD);
	 delay_ms(100);

	for(i=0;i<3;i++)
	{
		Movement(TrecekKiri[i]);
		if(i == 3)
		{
			delay_ms(500);
		}
	}
}

void GoPrepareTrecekKiri()
{
	unsigned char i;

	 lcd_clear();
	 sprintf(LCD," prepare Trecek Kiri\n%d",LineSensor());
	 lcd_puts(LCD);
	 delay_ms(100);

	for(i=0;i<4;i++)
	{
		Movement(PrepareTrecekKiri[i]);
	}
}

void GoKlatBahu()
{
	unsigned char i,j;

	 lcd_clear();
	 sprintf(LCD,"Klat Bahu\n%d",LineSensor());
	 lcd_puts(LCD);
	 delay_ms(100);

	for(j=0;j<5;j++)
	{
		for(i=0;i<2;i++)
		{
			Movement(KlatBahu[i]);
		}
	}
}

void GoNguraiRikmo()
{
	unsigned char i,j;

	 lcd_clear();
	 sprintf(LCD,"Ngurai Rikmo\n%d",LineSensor());
	 lcd_puts(LCD);
```

```
        delay_ms(100);

       for(j=0;j<3;j++)
       {
              for(i=0;i<2;i++)
              {
                     Movement(NguraiRikmo[i]);
              }
       }
}

void GoBerhias()
{
       unsigned char i;

        lcd_clear();
        sprintf(LCD,"Berhias\n%d",LineSensor());
        lcd_puts(LCD);
        delay_ms(100);

       for(i=0;i<6;i++)
       {
              Movement(Berhias[i]);
              delay_ms(1000);
       }
}

void GoZonaTengah()
{
       unsigned char i;

        lcd_clear();
        sprintf(LCD,"Zona Tengah\n%d",LineSensor());
        lcd_puts(LCD);
        delay_ms(100);

       for(i=0;i<25;i++)
       {
              Movement(ZonaTengah[i]);
              delay_ms(1000);
       }
}

void GoGaya()
{
        lcd_clear();
        sprintf(LCD,"CHRENJI V1.1");
        lcd_puts(LCD);
        delay_ms(100);

        Movement(Gaya);
}

void GoGeserKiri()
{
       unsigned char i;

        lcd_clear();
        sprintf(LCD,"GeserKiri\n%d",LineSensor());
        lcd_puts(LCD);
        delay_ms(100);

       for(i=0;i<5;i++)
       {
```

```
		Movement(GeserKiri[i]);

	}
	LineSensorValue = LineSensor();
}

void GoGeserKanan()
{
	unsigned char i;

	 lcd_clear();
	 sprintf(LCD,"GeserKanan\n%d",LineSensor());
	 lcd_puts(LCD);
	 delay_ms(100);

	for(i=0;i<6;i++)
	{
		Movement(GeserKanan[i]);

	}
	LineSensorValue = LineSensor();
}

void End()
{
	 lcd_clear();
	 lcd_putsf("It is the end of the move");
	 delay_ms(100);
}
```

SUB PROGRAM UNTUK LINE SENSOR

```
#define RIGHT_SENSOR (PINB & 0x07)
#define LEFT_SENSOR (PINB & 0xF0)

unsigned char LineSensor()
{
	char data;
	delay_ms(500);

	if (LEFT_SENSOR != 0x00)
		{
			data = 1;
		}
	if (RIGHT_SENSOR != 0x00)
		{
			data = 2;
		}
	if ((LEFT_SENSOR != 0x00) && (RIGHT_SENSOR != 0x00))
		{
			data = 0;
		}
	return(data);
```

LAMPIRAN C
DATASHEET

ATMEGA 128 SMD………………………………………………………… C-1
FLOWSTONE…………………………………………………………….. C-6
FLOWBOARD……………………………………………………………. C-8
SSC-32……………………………………………………………………. C-12

Features

- High-performance, Low-power AVR® 8-bit Microcontroller
- Advanced RISC Architecture
  - 133 Powerful Instructions – Most Single Clock Cycle Execution
  - 32 x 8 General Purpose Working Registers + Peripheral Control Registers
  - Fully Static Operation
  - Up to 16 MIPS Throughput at 16 MHz
  - On-chip 2-cycle Multiplier
- High Endurance Non-volatile Memory segments
  - 128K Bytes of In-System Self-programmable Flash program memory
  - 4K Bytes EEPROM
  - 4K Bytes Internal SRAM
  - Write/Erase cycles: 10,000 Flash/100,000 EEPROM
  - Data retention: 20 years at 85°C/100 years at 25°C[1]
  - Optional Boot Code Section with Independent Lock Bits
    In-System Programming by On-chip Boot Program
    True Read-While-Write Operation
  - Up to 64K Bytes Optional External Memory Space
  - Programming Lock for Software Security
  - SPI Interface for In-System Programming
- 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
  - Two Expanded 16-bit Timer/Counters with Separate Prescaler, Compare Mode and Capture Mode
  - Real Time Counter with Separate Oscillator
  - Two 8-bit PWM Channels
  - 6 PWM Channels with Programmable Resolution from 2 to 16 Bits
  - Output Compare Modulator
  - 8-channel, 10-bit ADC
    8 Single-ended Channels
    7 Differential Channels
    2 Differential Channels with Programmable Gain at 1x, 10x, or 200x
  - Byte-oriented Two-wire Serial Interface
  - Dual Programmable Serial USARTs
  - Master/Slave SPI Serial Interface
  - Programmable Watchdog Timer with 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
  - Software Selectable Clock Frequency
  - ATmega103 Compatibility Mode Selected by a Fuse
  - Global Pull-up Disable
- I/O and Packages
  - 53 Programmable I/O Lines
  - 64-lead TQFP and 64-pad QFN/MLF
- Operating Voltages
  - 2.7 - 5.5V ATmega128L
  - 4.5 - 5.5V ATmega128
- Speed Grades
  - 0 - 8 MHz ATmega128L
  - 0 - 16 MHz ATmega128

8-bit AVR® Microcontroller with 128K Bytes In-System Programmable Flash

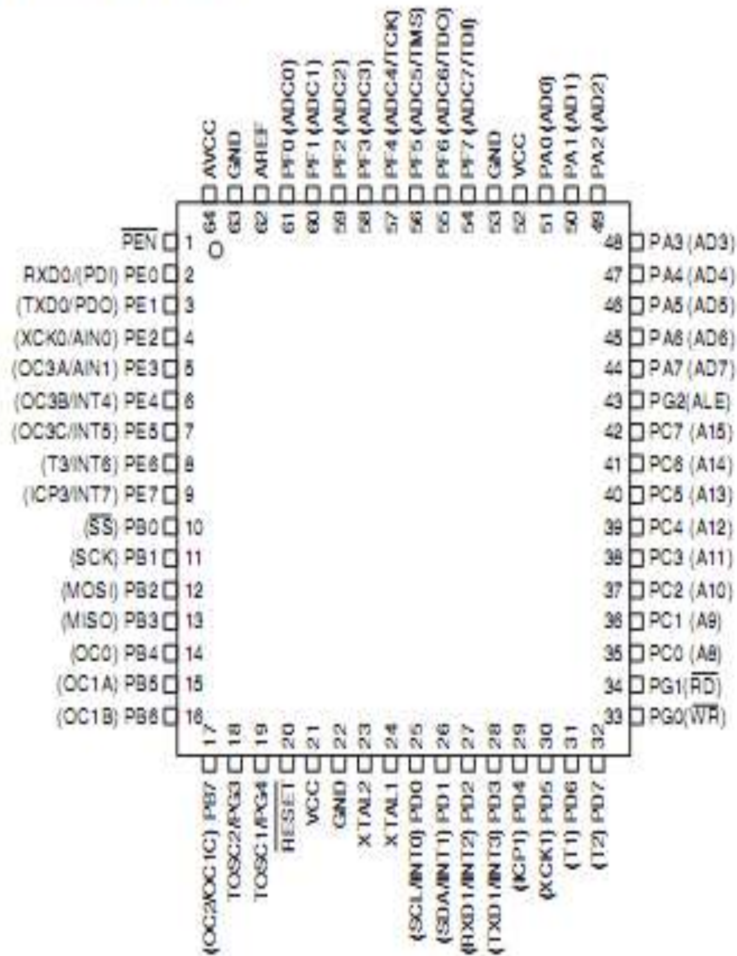
ATmega128
ATmega128L

Note: Not recommended for new designs.

Rev. 2467S-AVR-07/09

Pin Configurations

Figure 1. Pinout ATmega128



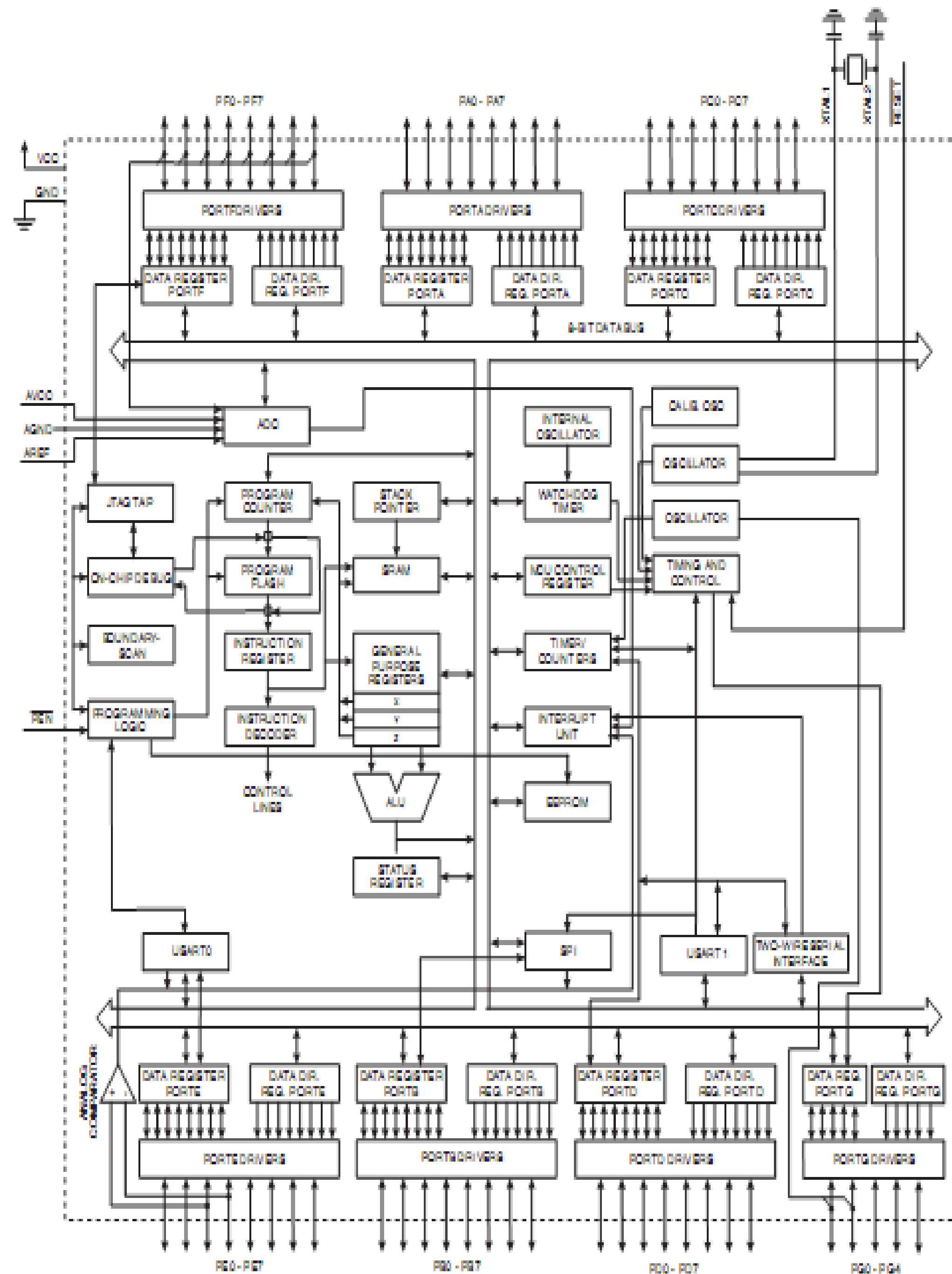
Note: The Pinout figure applies to both TQFP and MLF packages. The bottom pad under the QFN/MLF package should be soldered to ground.

Overview

The ATmega128 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 ATmega128 achieves throughputs approaching 1 MIPS per MHz allowing the system designer to optimize power consumption versus processing speed.

Block Diagram

Figure 2. Block Diagram



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

The ATmega128 provides the following features: 128K bytes of In-System Programmable Flash with Read-While-Write capabilities, 4K bytes EEPROM, 4K bytes SRAM, 53 general purpose I/O lines, 32 general purpose working registers, Real Time Counter (RTC), four flexible Timer/Counters with compare modes and PWM, 2 USARTs, a byte oriented Two-wire Serial Interface, an 8-channel, 10-bit ADC with optional differential input stage with programmable gain, programmable Watchdog Timer with Internal Oscillator, an SPI serial port, IEEE std. 1149.1 compliant JTAG test interface, also used for accessing the On-chip Debug system and programming and six software selectable power saving modes. The Idle mode stops the CPU while allowing the 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 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 ATmega128 is a powerful microcontroller that provides a highly flexible and cost effective solution to many embedded control applications.

The ATmega128 AVR is supported with a full suite of program and system development tools including: C compilers, macro assemblers, program debugger/simulators, in-circuit emulators, and evaluation kits.

ATmega103 and ATmega128 Compatibility

The ATmega128 is a highly complex microcontroller where the number of I/O locations supersedes the 64 I/O locations reserved in the AVR instruction set. To ensure backward compatibility with the ATmega103, all I/O locations present in ATmega103 have the same location in ATmega128. Most additional I/O locations are added in an Extended I/O space starting from $60 to $FF, (i.e., in the ATmega103 internal RAM space). These locations can be reached by using LD/LDS/LDD and ST/STS/STD instructions only, not by using IN and OUT instructions. The relocation of the internal RAM space may still be a problem for ATmega103 users. Also, the increased number of interrupt vectors might be a problem if the code uses absolute addresses. To solve these problems, an ATmega103 compatibility mode can be selected by programming the fuse M103C. In this mode, none of the functions in the Extended I/O space are in use, so the internal RAM is located as in ATmega103. Also, the Extended Interrupt vectors are removed.

The ATmega128 is 100% pin compatible with ATmega103, and can replace the ATmega103 on current Printed Circuit Boards. The application note "Replacing ATmega103 by ATmega128" describes what the user should be aware of replacing the ATmega103 by an ATmega128.

ATmega103 Compatibility Mode

By programming the M103C fuse, the ATmega128 will be compatible with the ATmega103 regards to RAM, I/O pins and interrupt vectors as described above. However, some new features in ATmega128 are not available in this compatibility mode, these features are listed below:

- One USART instead of two, Asynchronous mode only. Only the eight least significant bits of the Baud Rate Register is available.
- One 16 bits Timer/Counter with two compare registers instead of two 16-bit Timer/Counters with three compare registers.
- Two-wire serial interface is not supported.
- Port C is output only.
- Port G serves alternate functions only (not a general I/O port).
- Port F serves as digital input only in addition to analog input to the ADC.
- Boot Loader capabilities is not supported.
- It is not possible to adjust the frequency of the internal calibrated RC Oscillator.
- The External Memory Interface can not release any Address pins for general I/O, neither configure different wait-states to different External Memory Address sections.

In addition, there are some other minor differences to make it more compatible to ATmega103:

- Only EXTRF and PORF exists in MCUCSR.
- Timed sequence not required for Watchdog Time-out change.
- External Interrupt pins 3 - 0 serve as level interrupt only.
- USART has no FIFO buffer, so data overrun comes earlier.

Unused I/O bits in ATmega103 should be written to 0 to ensure same operation in ATmega128.

Pin Descriptions

VCC

Digital supply voltage.

GND

Ground.

Port A (PA7..PA0)

Port A is an 8-bit bi-directional I/O port with internal pull-up resistors (selected for each bit). The Port A output buffers have symmetrical drive characteristics with both high sink and source capability. As inputs, Port A pins that are externally pulled low will source current if the 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 A also serves the functions of various special features of the ATmega128 as listed on page 73.

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 ATmega128 as listed on page 74.

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.

Port C also serves the functions of special features of the ATmega128 as listed on page 77. In ATmega103 compatibility mode, Port C is output only, and the port C pins are **not** tri-stated when a reset condition becomes active.

Note: The ATmega128 is by default shipped in ATmega103 compatibility mode. Thus, if the parts are not programmed before they are put on the PCB, PORTC will be output during first power up, and until the ATmega103 compatibility mode is disabled.

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 ATmega128 as listed on page 78.

Port E (PE7..PE0)

Port E is an 8-bit bi-directional I/O port with internal pull-up resistors (selected for each bit). The Port E output buffers have symmetrical drive characteristics with both high sink and source capability. As inputs, Port E pins that are externally pulled low will source current if the pull-up resistors are activated. The Port E pins are tri-stated when a reset condition becomes active, even if the clock is not running.

Port E also serves the functions of various special features of the ATmega128 as listed on page 81.

Port F (PF7..PF0)

Port F serves as the analog inputs to the A/D Converter.

Port F 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 F output buffers have symmetrical drive characteristics with both high sink and source capability. As inputs, Port F pins that are externally pulled low will source current if the pull-up resistors are activated. The Port F 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 PF7(TDI), PF5(TMS), and PF4(TCK) will be activated even if a Reset occurs.

The TDO pin is tri-stated unless TAP states that shift out data are entered.

Port F also serves the functions of the JTAG interface.

In ATmega103 compatibility mode, Port F is an input Port only.

Port G (PG4..PG0)

Port G is a 5-bit bi-directional I/O port with internal pull-up resistors (selected for each bit). The Port G output buffers have symmetrical drive characteristics with both high sink and source capability. As inputs, Port G pins that are externally pulled low will source current if the pull-up resistors are activated. The Port G pins are tri-stated when a reset condition becomes active, even if the clock is not running.

Port G also serves the functions of various special features.

The port G pins are tri-stated when a reset condition becomes active, even if the clock is not running.

In ATmega103 compatibility mode, these pins only serves as strobes signals to the external memory as well as input to the 32 kHz Oscillator, and the pins are initialized to PG0 = 1, PG1 = 1, and PG2 = 0 asynchronously when a reset condition becomes active, even if the clock is not running. PG3 and PG4 are oscillator pins.

$\overline{\text{RESET}}$

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 19 on page 51. 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 F and the A/D Converter. It should be externally connected to V_{CC}, even if the ADC is not used. If the ADC is used, it should be connected to V_{CC} through a low-pass filter.

AREF

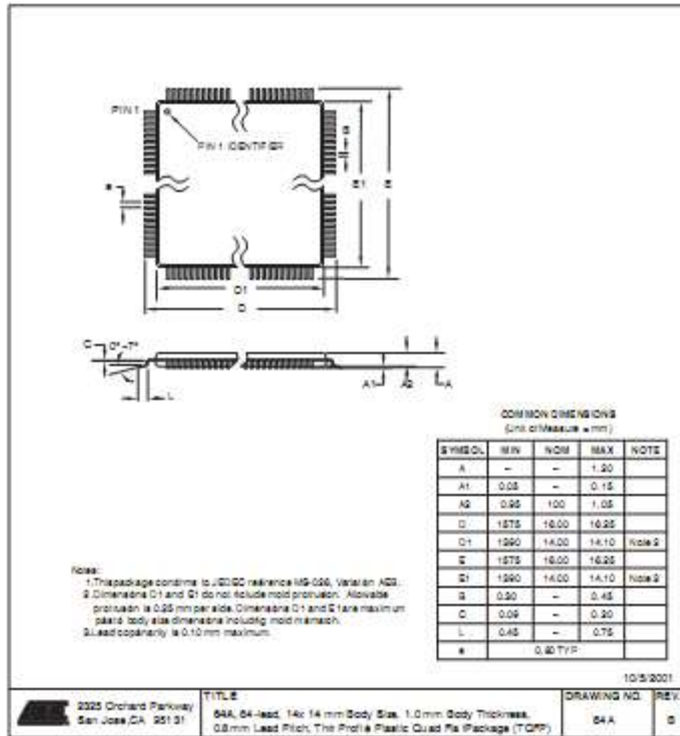
AREF is the analog reference pin for the A/D Converter.

PEN

PEN is a programming enable pin for the SPI Serial Programming mode, and is internally pulled high . By holding this pin low during a Power-on Reset, the device will enter the SPI Serial Programming mode. $\overline{\text{PEN}}$ has no function during normal operation.

Packaging Information

64A



COMMON DIMENSIONS
(Unit of Measure = mm)

SYMBOL	MIN	NOM	MAX	NOTE
A	–	–	1.20	
A1	0.05	–	0.15	
A2	0.95	100	1.05	
D	1575	16.00	16.25	
D1	1390	14.00	14.10	Note 2
E	1575	16.00	16.25	
E1	1390	14.00	14.10	Note 2
B	0.30	–	0.45	
C	0.09	–	0.20	
L	0.45	–	0.75	
e	0.80 TYP			

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

10/5/2001

	TITLE	DRAWING NO.	REV.
2325 Orchard Parkway San Jose, CA 95131	64A, 64-lead, 14 x 14 mm Body Size, 1.0 mm Body Thickness, 0.8 mm Lead Pitch, Thin Profile Plastic Quad Flat Package (TQFP)	64A	B

SSC-32 Manual.

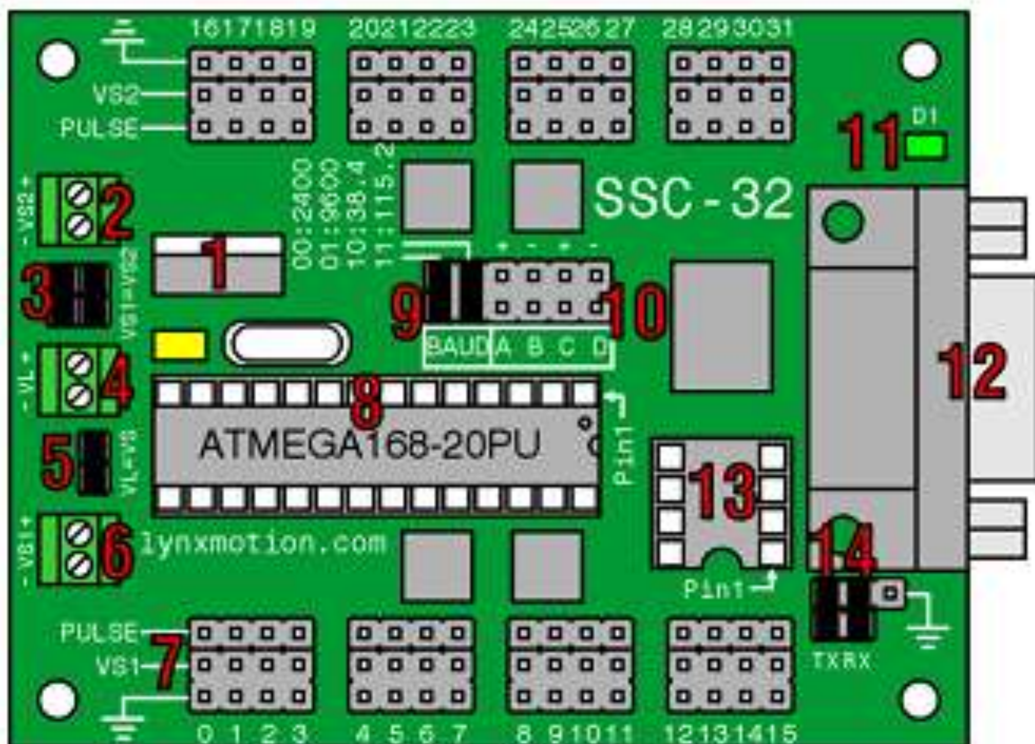
Author: Jim Frye
Version: 2.01XE
Date: March 26, 2009

Table of Contents

- SSC-32
  - SSC-32 Hardware Information
  - Shorting Bar Jumpers and Connections
- Command Formatting for the SSC-32
  - Command Types and Groups
  - Servo Move or Group Move
  - Software Position Offset
  - General Output Info
  - Discrete Output
  - Byte Output
  - Query Movement Status
  - Query Pulse Width
  - Read Digital Inputs
  - Read Analog Inputs
  - 12 Servo Hexapod Sequencer Commands
  - Notes on Hexapod Sequencer
  - Query Hex Sequencer State
  - Get Software Version
  - Firmware Upgrade
  - Transfer to Boot
  - Mini SSC-II Emulation
- SSC-32 Registers
  - Registers General Info
  - Enable Register (R0) Bit Definitions
  - Register Read/Write
  - Miscellaneous Register Commands
  - Startup Strings
  - Additional Examples
- Testing the Controller
  - Troubleshooting Information
- Basic Atom Programming Examples
  - Atom / SSC-32 Test Example
  - Simple Biped Example

Links

- SSC-32 Schematic (pdf)
- LynxTerm Download



SSC-32 Hardware Information.

1. The Low Dropout regulator will provide 5vdc out with as little as 5.5vdc coming in. This is important when operating your robot from a battery. It can accept a maximum of 9vdc in. The regulator is rated for 500mA, but we are de-rating it to 250mA to prevent the regulator from getting too hot.

2. This terminal connects power to servo channels 16 through 31. Apply 4.8vdc to 6.0vdc for most analog or digital servos. This can be directly from a 5-cell NiMH battery pack. 7.2vdc - 7.4vdc can be applied to HSR-5980 or HSR-5990 servos. This can be directly from a 6-cell NiMH battery pack or a 2-cell LiPo battery pack.

Board	Input
VS2 +	RED
VS1 -	BLACK

3. These jumpers are used to connect VS1 to VS2. Use this option when you are powering all servos from the same battery. Use both jumpers. Alternately, if you want to use two separate battery packs, one on each side, then remove both of these jumpers.

4. This is the Logic Voltage, or VL. This input is normally used with a 9vdc battery connector to provide power to the ICs and anything connected to the 5vdc lines on the board. The valid range for this terminal is 6vdc - 9vdc. This input is used to isolate the logic from the Servo Power Input. It is necessary to remove the VS1=VL jumper when powering the servos separately from the logic VL. The SSC-32 should draw 35mA with nothing connected to the 5vdc output.

Board	Input
VL +	RED
VL -	BLACK

5. This jumper allows powering the microcontroller and support circuitry from the servo power supply. This requires at least 6vdc to operate correctly. If the microcontroller resets when too many servos are moving at the same time, it may be necessary to power the microcontroller separately using the VL input. A 9vdc works nicely for this. This jumper must be removed when powering the microcontroller separately!

6. This terminal connects power to servo channels 16 through 31. Apply 4.8vdc to 6.0vdc for most analog or digital

servos. This can be directly from a 5-cell NiMH battery pack. 7.2vdc - 7.4vdc can be applied to HSR-5980 or HSR-5990 servos. This can be directly from a 6-cell NiMH battery pack or a 2-cell LiPo battery pack.

Board	Input
VL1 +	RED
VL1 -	BLACK

7. This is where you connect the servos or other output devices. Use caution and remove power when connecting anything to the I/O bus.

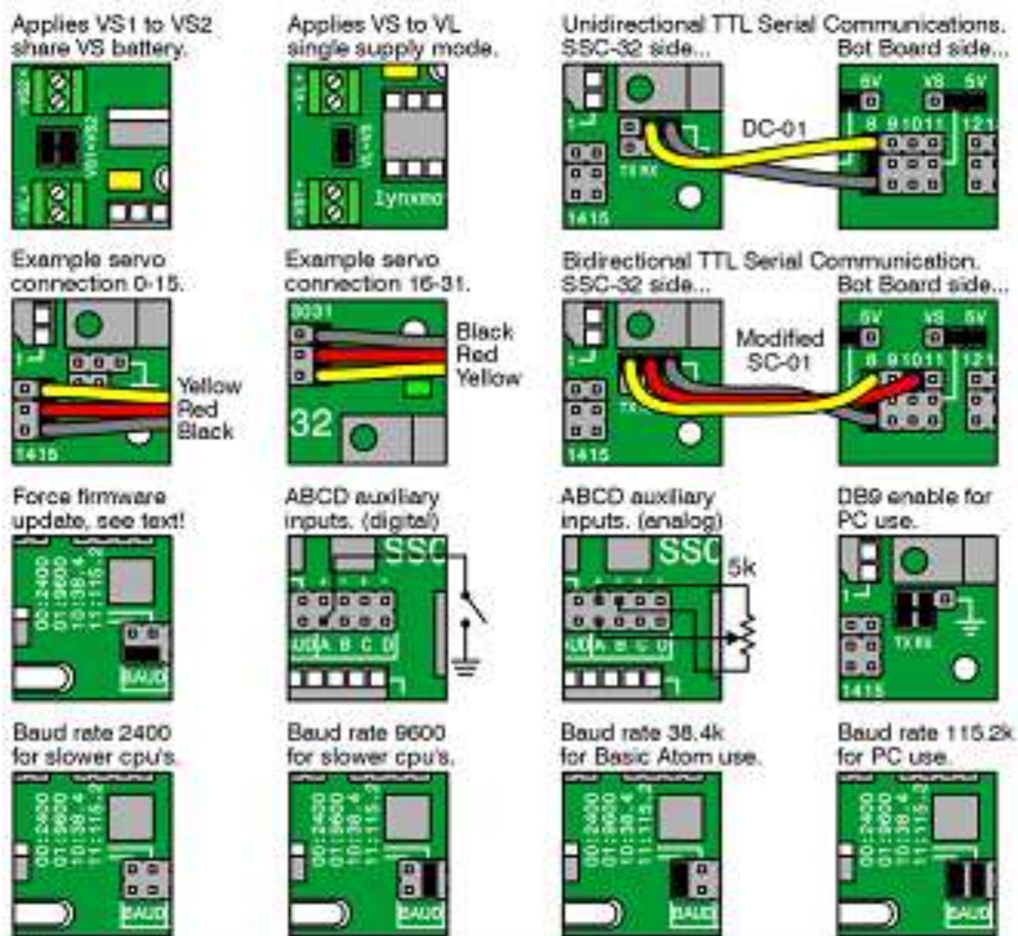
Board	Wire
Pulse	Yellow or White
VS	Red
Ground	Black or Brown

8. This is where the Atmel IC chip goes. Be careful to insert it with Pin 1 in the upper right corner as pictured. Take care to not bend the pins.
9. The two BAUD inputs allow configuring the baud rate. Please see the examples below.

Jumpers	Baud Rate	Usage
0 0	2400	Slower Processors
0 1	9600	Slower Processors
1 0	38.4k	Atom/Stamp Communication
1 1	115.2k	PC Communication, Firmware Update

10. The ABCD inputs have both static and latching support. The inputs have internal weak (50k) pullups that are used when a Read Digital Input command is used. A normally open switch connected from the input to ground will work fine.
11. This is the Processor Good LED. It will light steady when power is applied and will remain lit until the processor has received a valid serial command. It will then go out and will blink whenever it is receiving serial data.
12. Simply plug a straight-through M/F DB9 cable from this plug to a free 9-pin serial port on your PC for receiving servo positioning data. Alternately a USB-to-serial adapter will work well. Note, many USB-to-serial adapters require a separate power supply to work well.
13. This is an 8-pin EEPROM socket. The EEPROM is supported by the 2.01GP firmware.
14. This is the TTL serial port or DB9 serial port enable. Install two jumpers as illustrated below to enable the DB9 port. Install wire connectors to utilize TTL serial communication from a host microcontroller.

Shorting Bar Jumpers and Connectors at a glance.



Command Formatting for the SSC-32.

Command Types and Groups.

With the exception of Mini SSC-II mode, all SSC-32 commands must end with a carriage return character (ASCII 13). Multiple commands of the same type can be issued simultaneously in a Command Group. All of the commands in a command group will be executed after the final carriage return is received. Commands of different types cannot be mixed in the same command group. In addition, numeric arguments to all SSC-32 commands must be ASCII strings of decimal numbers, e.g. "1234". Some commands accept negative numbers, e.g. "-5678". Programming examples will be provided. ASCII format is not case sensitive. Use as many bytes as required. Spaces, tabs, and line feeds are ignored.

Command Types and Groups			
1	Servo Movement	7	Read Analog Inputs
2	Discrete Output	8	12 Servo Hexapod Gait Sequencer
3	Byte Output	9	Query Hex Sequencer
4	Query Movement Status	10	Get Version
5	Query Pulse Width	11	Go to Boot
6	Read Digital Inputs	12	MiniSSC-II Compatibility

Servo Move or Group Move.

# <ch> P <pw> S <spd> ... # <ch> P <pw> S <spd> T <time> <cr>	
<ch>	Channel number in decimal, 0-31

<pw>	Pulse width in microseconds, 500-2500
<spd>	Movement speed in uS per second for one channel (Optional)
<time>	Time in mS for the entire move, affects all channels, 65535 max (Optional)
<cr>	Carriage return character, ASCII 13 (Required to initiate action)
<esc>	Cancel the current action, ASCII 27

Servo Move Example: "#5 P1600 S750 <cr>"

The example will move the servo on channel 5 to position 1600. It will move from its current position at a rate of 750uS per second until it reaches its commanded destination. For a better understanding of the speed argument, consider that 1000uS of travel will result in around 90° of rotation. A speed value of 100uS per second means the servo will take 10 seconds to move 90°. Alternately, a speed value of 2000uS per second equates to 500mS (half a second) to move 90°.

Servo Move Example: "#5 P1600 T1000 <cr>"

The example will move servo 5 to position 1600. It will take 1 second to complete the move regardless of how far the servo has to travel to reach the destination.

Servo Group Move Example: "#5 P1600 #10 P750 T2500 <cr>"

The example will move servo 5 to position 1600 and servo 10 to position 750. It will take 2.5 seconds to complete the move, even if one servo has farther to travel than another. The servos will both start and stop moving at the same time. This is a very powerful command. By commanding all the legs in a walking robot with the Group Move it is easy to synchronize complex gaits. The same synchronized motion can benefit the control of a robotic arm as well.

You can combine the speed and time commands if desired. The speed for each servo will be calculated according to the following rules:

1. All channels will start and end the move simultaneously.
2. If a speed is specified for a servo, it will not move any faster than the speed specified (but it might move slower if the time command requires).
3. If a time is specified for the move, then the move will take at least the amount of time specified (but might take longer if the speed command requires).

Servo Move Example: "#5 P1600 #17 P750 #2 P2250 T2000 <cr>"

The example provides 1600uS on ch5, 750uS on ch17, and 2250uS on ch2. The entire move will take at least 2 seconds, but ch17 will not move faster than 500uS per second. The actual time for the move will depend on the initial pulse width for ch17. Suppose ch17 starts at position 2000. Then it has to move 1250uS. Since it is limited to 500uS per second, it will require at least 2.5 seconds, so the entire move will take 2.5 seconds. On the other hand, if ch17 starts at position 1000, it only needs to move 250uS, which it can do in 0.5 seconds, so the entire move will take 2 seconds.

Important! The first positioning command should be a normal "# <ch> P <pw>" command. Because the controller doesn't know where the servo is positioned on powerup, it will ignore speed and time commands until the first normal command has been received.

Any move that involves more than one servo and uses either the S or T modifier is considered a Group Move, and all servos will start and stop moving at the same time. If you require moving several servos at different speeds, you must issue the commands separately.

Software Position Offset.

# <ch> PO <offset value> ... # <ch> PO <offset value> <cr>	
<ch>	Channel number in decimal, 0-31
<offset value>	100 to -100 in uSeconds
<cr>	Carriage return character, ASCII 13

This command allows the servo centered (1500uS) position to be aligned perfectly. The servo channel will be offset by the amount indicated in offset value. This represents approximately 15° of range. It's important to build the mechanical assembly as close as possible to the desired centered position before applying the servo offset. This makes it easy to setup servos that do not have mechanical alignment. The Position Offset command should be issued only once in your program. When the SSC-32 is turned off it will forget the Position Offsets.

The current SSC-32 now has an internal register method for doing Position Offsets. These are stored in the Atmel chips

internal EEPROM and are retained when power is removed. Use of this feature is covered in the Register Support section of this manual.

General Output Information.

The outputs on the SSC-32 come from four 74HC595 8 bit shift register chips. There are four banks of 8 bit outputs as shown 0-7, 8-15, 16-23 and 24-32. The outputs can sink or source up to 25mA per pin, but a max of 70mA per bank must be observed.

Discrete Output.

# <ch> <lvl> ... # <ch> <lvl> <cr>	
<ch>	Channel number in decimal, 0-31
<lvl>	Logic level for the channel, either 'H' for High or 'L' for Low
<cr>	Carriage return character, ASCII 13

The channel will go to the level indicated within 20mS of receiving the carriage return.

Discrete Output Example: "#3H #4L <cr>"

This example will output a High (+5v) on channel 3 and a Low (0v) on channel 4.

Byte Output.

# <bank> : <value> <cr>	
<bank>	(0 = Pins 0-7, 1 = Pins 8-15, 2 = Pins 16-23, 3 = Pins 24-31)
<value>	Decimal value to output to the selected bank (0-255), Bit 0 = LSB of bank
<cr>	Carriage return character, ASCII 13

This command allows 8 bits of binary data to be written at once. All pins of the bank are updated simultaneously. The banks will be updated within 20mS of receiving the carriage return.

Bank Output Example: "#3:123 <cr>"

This example will output the value 123 (decimal) to bank 3. 123 (dec) = 01111011 (bin), and bank 3 is pins 24-31. So this command will output a "0" to pins 26 and 31, and will output a "1" to all other pins.

Query Movement Status.

Example: "Q <cr>"

This will return a "." if the previous move is complete, or a "+" if it is still in progress.

There will be a delay of 50uS to 5mS before the response is sent.

Query Pulse Width.

Example: "QP <arg> <cr>"

This will return a single byte (in binary format) indicating the pulse width of the selected servo with a resolution of 10uS. For example, if the pulse width is 1500uS, the returned byte would be 150 (binary).

Multiple servos may be queried in the same command. The return value will be one byte per servo. There will be a delay of at least 50uS to 5mS before the response is sent. Typically the response will be started within 100uS.

Read Digital Inputs.

Example: "A B C D AL BL CL DL <cr>"

A, B, C, and D are normal input reads. They read the value on the input as a binary value. It returns ASCII "0" if the input is a low (0v) or an ASCII "1" if the input is a high (+5v).

AL, BL, CL, and DL are latching input reads. They return the value on the input as an ASCII "0" if the input is a low (0v) or if it has been low since the last "L command. It returns a high (+5v) if the input is high and never went low since the last "L command. Simply stated, it will return a low if the input ever goes low. Reading the status simply resets the latch.

The ABCD inputs have a weak pullup (~50k) that is enabled when used as inputs. They are checked approximately every 1mS, and are debounced for approximately 15mS. The logic value for the read commands will not be changed until the input has been at the new logic level continuously for 15mS. The Read Digital Input Commands can be grouped in a single read, up to 8 values per read. They will return a string with one character per input with no spaces.

Read Digital Input Example: "A B C DL <cr>"

Read Analog Inputs.

Example: "VA VB VC VD <cr>"

VA, VB, VC, and VD read the value on the input as analog. It returns a single byte with the 8-bit (binary) value for the voltage on the pin.

When the ABCD inputs are used as analog inputs the internal pullup is disabled. The inputs are digitally filtered to reduce the effect of noise. The filtered values will settle to their final values within 8mS of a change. A return value of 0 represents 0vdc. A return value of 255 represents +4.98vdc. To convert the return value to a voltage, multiply by 5/256. At power up the ABCD inputs are configured for digital input with pullup. The first time a V* command is used, the pin will be converted to analog without pullup. The result of this first read will not return valid data.

Read Analog Input Example: "VA VB <cr>"

This example will return 2 bytes with the analog values of A and B. For example is the voltage on Pin A is 2vdc and Pin B is 3.5vdc, the return value will be the bytes 102 (binary) and 179 (binary).

FLOWBOARD DAN FLOWSTONE

C O N T E N T S

CHAPTER 1 — 1 Introduction — 4

About This Guide 5
What is FlowBoard? 6
Specifications 6

CHAPTER 2 — 2 Getting Started — 8

Package Contents 9
Important Safety Note 9
Connecting Up 10
Step 1 - Connecting Satellite Boards 10
Step 2 - Connecting Power 12
Step 3 - Connect to PC 13

CHAPTER 3 — 3 FlowStone — 14

The FlowBoard Component 15
The FlowBoard GSM Component 16

CHAPTER 4 — 4 Updating Firmware — 18

FlowBoard Updater 19
Using the Updater 19

CHAPTER 5 — 5 Technical Information — 21

Connector Sockets 22
Digital Inputs 22
Digital Outputs 22
Analogue Inputs 23

1 Introduction

About FlowBoard

About This Guide

This manual provides a detailed description of the FlowBoard hardware and extension boards. Its purpose is to show you how the hardware works and what its capabilities are.

If you are looking for tutorials on how to use the FlowBoard with the FlowStone graphical programming software then see the Tutorials section of the DSP Robotics web site:
http://www.dsprobotics.com/tutorials.html

Additional information and articles about the product can be found at:
http://www.dsprobotics.com/support.html

If you have any comments about this guide please email them to info@dsprobotics.com.

IMPORTANT

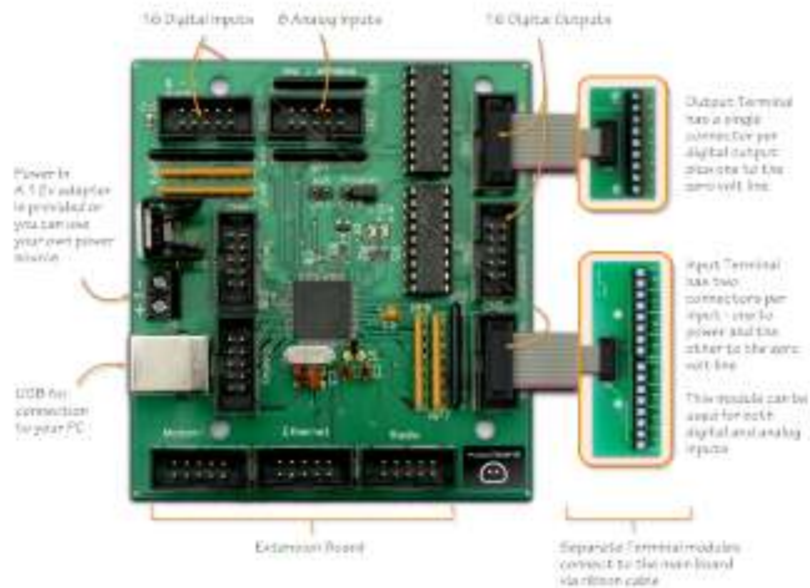
Please read this guide carefully BEFORE using the product(s) you have purchased. The guide contains important safety information that you must be aware of before attempting to power the board or connect anything to it.

What is FlowBoard?

FlowBoard is a low cost Data Acquisition (DAQ) system developed by DSPRobotics specifically for the use with the FlowStone programming language.

FlowBoard uses a modular system of satellite boards which allows you to configure the hardware exactly to your needs. Interchangeable Input and Output terminal boards provide the base interface with added functionality being provided by optional extension boards.

Specifications



At the heart of the system is the FlowBoard DAQ. This connects to your PC via USB and manages communications between the satellite boards and your computer. Power is also supplied through this main board.

2 Getting Started

HOW TO GET UP AND RUNNING

Package Contents

The FlowBoard DAQ Pack contains the following components:

- 1 x FlowBoard DAQ main board
- 2 x Digital Output Terminal boards
- 2 x Digital Input Terminal boards
- 1 x Analog InputTerminal board
- 1 x PSU adapter
- 1 x PSU connector (pre-fitted to power terminals on the main board)
- 5 x connector ribbons
- 1 x USB connector cable

NOTE: the PSU adapter and USB cable may differ slightly in appearance from the picture above. The PSU adapter should also be supplied with the correct pin arrangement for your region.

Important Safety Note

Please read this section carefully before connecting anything to the main board.

Satellite boards should not be connected with the power applied as this may cause damage to either the satellite board or the FlowBoard itself.

Always power off before connecting or disconnecting elements from the FlowBoard.

Connecting your FlowBoard is easy and takes just 3 simple steps.

Step 1 - Connecting Satellite Boards

The first step is to choose your configuration of satellite boards. These must be connected before applying power to the FlowBoard.

The supplied ribbon cables should be used to connect the satellite boards to the main board. The locations of the connector sockets on the main board are outlined below.

Digital Output Terminal Board

There are two sockets labeled Outputs 1-8 and Outputs 9-16. These obviously correspond to the 16 digital outputs.

Digital Input Terminal Board

There are two sockets labeled Inputs 1-8 and Inputs 9-16. These obviously correspond to the 16 digital inputs.

Analog Input Terminal Board

There is a single socket for connecting the analog input terminal board labeled Analogue/Aux.

GSM Board

There is a single socket for connecting the GSM board and its marked Modem.

The GSM board also requires a sim card to be inserted for normal operation (not supplied). For good reception a suitable antenna should also be connected to the input lead.

Step 2 - Connecting Power

The FlowBoard needs to be powered in order to function. The power input terminals are indicated by + and - symbols and their location is shown in the picture opposite.

The FlowBoard comes with a standard 12v power adapter and a connector for attaching this to board. This allows you to get up and running quickly. However, you can connect any power source you like so long as it is within the boards operating limits. The red LED will light to indicate power is on

Power Requirements and Operating Limits

The main board will run over the range 6 to 15 Volts d.c. However, if the Relay extension board is being used the range is reduced to 10 to 15 Volts as the relays themselves have a nominal rating of 12 Volts.

Current consumption is around 40mA excluding the power consumed by the satellite boards. For the total consumption of your configuration add the current used for the main board plus the current used by all the satellite boards. Power supplies should be well regulated.

GSM Board Power Requirements

The GSM Board requires its own separate power supply. This is in addition to the power supplied to the main board.

Once again the the ribbon connection between the GSM board and the main board should be in place BEFORE power is applied to EITHER of them.

Average current consumption of the GSM board is approximately 40mA but the power supplied should be capable of providing currents up to 1A for very short periods. Voltage should be in the range 6 to 15 Volts.

The on-board red LED will flash rapidly while the modem is acquiring the network and then will flash every 2 to 3 seconds once it is on the network.

Step 3 - Connect to PC

The final step is to plug the USB cable into the board using the silver socket. The other end of the cable should then be connected to your PC. The green LED will light to show that the board is connected to the PC and the red LED should turn off.

Your FlowBoard is now connected and ready to use with FlowStone.