#include <mega16.h>

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

// Declare your global variables here
void main(void)
{
    // Declare your local variables here
    unsigned char d, p;

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

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

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

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

// Timer/Counter 1 initialization
// Clock source: System Clock
// Clock value: Timer 1 Stopped
// Mode: Normal top=FFFFh
// OC1A output: Disconnected
// OC1B output: Disconnected
// Noise Canceler: Off
// Input Capture on Falling Edge
// Timer 1 Overflow Interrupt: Off
// Input Capture Interrupt: Off
// Compare A Match Interrupt: Off
// Compare B Match Interrupt: Off
TCCR1A=0x00;
TCCR1B=0x00;
TCNT1H=0x00;
TCNT1L=0x00;
ICR1H=0x00;
ICR1L=0x00;
OCR1AH=0x00;
OCR1AL=0x00;
OCR1BH=0x00;
OCR1BL=0x00;

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

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

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

// USART initialization
// Communication Parameters: 8 Data, 1 Stop, No Parity
// USART Receiver: On
// USART Transmitter: On
// USART Mode: Asynchronous
// USART Baud rate: 9600
UCSRA=0x00;
UCSRB=0x18;
UCSRC=0x86;
UBRRH=0x00;
UBRRL=0x47;

// Analog Comparator initialization
// Analog Comparator: Off
// Analog Comparator Input Capture by Timer/Counter 1: Off
ACSR=0x80;
SFIOR=0x00;
printf("\n Password");

while (1)
{
    // Place your code here
    p=getchar();
    if( p=='2')
    {
        printf("\n Password Benar");
        d=getchar();
        if (d=='A')
        {
            PORTC.0=1;
            if(PORTD.4 ==0)
            {
                delay_ms(200);
                printf("Lampu 1 Off");
            }
            else
            {
                delay_ms(200);
                printf("Lampu 1 On");
            }
        }
        if (d=='D')
        {
            PORTC.0=0;
            if(PORTD.4 ==0)
            {
                delay_ms(200);
                printf("Lampu 1 Off");
            }
            else
            {
                delay_ms(200);
                printf("Lampu 1 On");
            }
        }
        if (d=='G')
        {
            PORTC.1=1;
            if(PORTD.5 ==0)
{ 
delay_ms(200);
printf("Lampu 2 Off");
} 
else 
{ 
delay_ms(200);
printf("Lampu 2 On");
}
}

if (d=='J')
{
PORTC.1=0;
if(PORTD.5 ==0)
{ 
delay_ms(200);
printf("Lampu 2 Off");
} 
else 
{ 
delay_ms(200);
printf("Lampu 2 On");
}
}

if (d=='M')
{
PORTC.2=1;
if(PORTD.6 ==0)
{ 
delay_ms(200);
printf("Lampu 3 Off");
} 
else 
{ 
delay_ms(200);
printf("Lampu 3 On");
}
}

if (d=='P')
{
PORTC.2=0;
if(PORTD.6 ==0)
{
}
delay_ms(200);
printf("Lampu 3 Off");
}
else
{
delay_ms(200);
printf("Lampu 3 On");
}

} if (p!='2')
{
printf("\n Password Salah");
}
};
}
LAMPIRAN B
PROGRAM PADA HANDPHONE (J2ME)

BluetoothDeviceDiscovery

import java.io.IOException;
import java.util.Vector;
import javax.bluetooth.DeviceClass;
import javax.bluetooth.DiscoveryAgent;
import javax.bluetooth.DiscoveryListener;
import javax.bluetooth.LocalDevice;
import javax.bluetooth.RemoteDevice;
import javax.bluetooth.ServiceRecord;

public class BluetoothDeviceDiscovery implements DiscoveryListener {

        public static Object lock=new Object();

        public static Vector vecDevices=new Vector();

        public static void main() throws IOException {
                BluetoothDeviceDiscovery bluetoothDeviceDiscovery=new BluetoothDeviceDiscovery();
                LocalDevice localDevice = LocalDevice.getLocalDevice();
                DiscoveryAgent agent = localDevice.getDiscoveryAgent();
                agent.startInquiry(DiscoveryAgent.GIAC,
                bluetoothDeviceDiscovery);

                try {
                        synchronized(lock) {
                                lock.wait();
                        }
                } catch (InterruptedException e) {
                        e.printStackTrace();
                }

        }

        public void deviceDiscovered(RemoteDevice btDevice, DeviceClass cod) {
                if(!vecDevices.contains(btDevice))
        }
vecDevices.addElement(btDevice);

public void servicesDiscovered(int transID, ServiceRecord[] servRecord)
{
}

public void serviceSearchCompleted(int transID, int respCode)
{
}

public void inquiryCompleted(int discType)
{
    synchronized(lock)
    {
        lock.notify();
    }
}

TerminalIOStream
import javax.microedition.midlet.MIDlet;

public class TerminalIOStream
{
    StringBuffer line;
    private boolean full;

    public TerminalIOStream()
    {
        reset();
    }

    public void append(char c)
    {
        line.append(c);
    }

    public void append(char [] l)
    {
        line.append(l);
    }
public synchronized String read() {
    try {
        while (full == false) {
            wait();
        }
    } catch (Exception e) {
        e.printStackTrace();
    }
    String s = line.toString();
    reset();
    return s;
}

public void write(String s) {
    line.append(s);
    flush();
}

public void reset() {
    line = new StringBuffer();
    full = false;
}

public synchronized boolean isFull() {
    return full;
}

Lampu

import javax.microedition.midlet.*;
import javax.microedition.lcdui.*;
import javax.microedition.io.*;
import java.io.*;
import javax.bluetooth.*;
import javax.microedition.io.Connector;
import java.io.IOException;
import java.util.Stack;
public class lampu extends MIDlet implements Runnable, javax.microedition.lcdui.CommandListener {

    public String output;
    data data;

    public lampu() {
    }

    private void initialize() {
    }

    public void commandAction(javax.microedition.lcdui.Command command, javax.microedition.lcdui.Displayable displayable) {
        if (command == exitCommand1) {
            exitMIDlet();
        }
    }

    private javax.microedition.lcdui.Command get_exitCommand() {
        if (exitCommand == null) {
            exitCommand = new javax.microedition.lcdui.Command("Exit", javax.microedition.lcdui.Command.EXIT, 1);
        }
        return exitCommand;
    }

    protected void exitMIDlet() {
        data.Closeconnection();
        destroyApp(true);
        notifyDestroyed();
    }

    private javax.microedition.lcdui.Command get_okCommand() {
        if (okCommand == null) {
            okCommand = new javax.microedition.lcdui.Command("Ok", javax.microedition.lcdui.Command.OK, 1);
        }
    }
}
private javax.microedition.lcdui.Form get_form1()
{
    if (form1 == null) {
        form1 = new javax.microedition.lcdui.Form(null, new
        javax.microedition.lcdui.Item[] {get_stringItem1()});
        form1.addCommand(get_exitCommand1());
        form1.setCommandListener(this);
    }
    return form1;
}

private javax.microedition.lcdui.StringItem get_stringItem1()
{
    if (stringItem1 == null)
    {
        stringItem1 = new javax.microedition.lcdui.StringItem("Thank You",""));
    }
    return stringItem1;
}

public void startApp()
{
    initialize();
data = new data(Display.getDisplay(this), get_form1());
    Display.getDisplay(this).setCurrent(data);
    new Thread(this).start();
}

public void pauseApp()
{
}

public void destroyApp(boolean unconditional)
{
}

public void run()
{
    try {
        data.write(" L A M P U " + "\n");
        while (true)
        {
            ...
output = "";
String tmp = "";
int input;
if (data.reader != null)
{
    input = data.reader.read();
    output += (char) input;
    if (input == 10)
    {
        data.write("\r\n");
    }
}
data.write(output);
}
}
catch (Exception e)
{
    e.printStackTrace();
}

Data

import java.io.IOException;
import java.util.Vector;
import javax.bluetooth.DeviceClass;
import javax.bluetooth.DiscoveryAgent;
import javax.bluetooth.DiscoveryListener;
import javax.bluetooth.LocalDevice;
import javax.bluetooth.RemoteDevice;
import javax.bluetooth.ServiceRecord;
import java.io.IOException;
import javax.microedition.lcdui.Canvas;
import javax.microedition.lcdui.Command;
import javax.microedition.lcdui.CommandListener;
import javax.microedition.lcdui.Display;
import javax.microedition.lcdui.Displayable;
import javax.microedition.lcdui.Font;
import javax.microedition.lcdui.Graphics;
import javax.microedition.lcdui.Image;
import javax.microedition.lcdui.TextBox;
import javax.microedition.io.StreamConnection;
import java.io.InputStreamReader;
import javax.microedition.io.Connector;
import javax.microedition.midlet.*;
import javax.microedition.lcdui.*;
import javax.microedition.io.*;
import javax.bluetooth.*;
import java.io.*;

public class data extends Canvas implements CommandListener, Runnable {
    InputStreamReader reader;
    public StreamConnection connection;
    public List lamp = null;
    public String URL = "";
    StringBuffer strbuf = new StringBuffer();
    List list = null;
    Vector two = new Vector();
    private Command show;
    private Command conne;
    private Command discon;
    private Command back;
    public String output;
    private Command cmdExit;
    StringBuffer sb = new StringBuffer(0);
    BluetoothDeviceDiscovery disc;
    private Command cmdDiscover;
    private Command cmdStopDiscover;
    public String col[] = null;
    private LocalDevice devLocal;
    private DiscoveryAgent discoverAgent;
    public String BUG = "";
    lampu midi;
    public OutputStreamWriter ow = null;
    public boolean what;
    public OutputStream dos = null;
    private char[][] content;
    private short numberOfRows, numberOfColumns;
    private int cursorRow, cursorCol;
    private int width, height;
    private short charHeight, charWidth;
    private Font font;
    private int color = 0x00FFFFFF;
    private Image offscreenT;
    private Image offscreenG;
    private boolean isWaiting;
    private char[] typeAheadBuffer = new char[20];
boolean timerActive = false, upperCase = false, userInput = false, starInput = false;
private String keys[] = { "0=" , "1:.,", "abc2", "def3", "ghi4", "jkl5", "mno6", "pqr7", "tuv8", "wxyz9", ",?!",@/-:;+&%*=<>$\{ }\~^#|", "#" };
connection = (StreamConnection) Connector.open ("btspp://"+URL+":1", Connector.READ_WRITE);
    reader = new InputStreamReader(connection.openInputStream());
    dos = connection.openOutputStream();
    ow = new OutputStreamWriter(dos, "iso-8859-1");
    }
    catch (IOException ex)
    {
        ex.printStackTrace();
    }
    }

    public void Closeconnection()
    {
        try {
            if (reader != null)
            {
                reader.close();
                reader = null;
            }

            if (dos != null)
            {
                dos.close();
                dos = null;
            }

            if (connection !=null)
            {
                connection.close();
                connection = null;
            }
        } catch (Exception e)
        {
            e.printStackTrace();
        }
    }

    public List getlamp()
    {
        if (lamp==null)
        {
            lamp = new List( "Device list", Choice.IMPLICIT );
            show = new Command("Search", Command.OK, 1);
            back= new Command("Back", Command.OK, 1);
conne = new Command("Connect", Command.OK, 1);
discon = new Command("Disconnect", Command.OK, 1);
lamp.addCommand( show );
lamp.setCommandListener( this );
}
return lamp;
}

public void initialize()
{
    this.addCommand(Connection = new Command("Connect",
Command.SCREEN, 2));
    this.addCommand(TextboxCommand = new Command("Text input",
Command.SCREEN, 2));
    this.addCommand(BackCommand = new Command("Exit",
Command.SCREEN, 2));

    istream = new TerminalIOStream();
    ostream = new TerminalIOStream();
    setCommandListener(this);
    width = getWidth();
    height = getHeight();
    offscreenT = Image.createImage(width, height);
    offscreenG = Image.createImage(width, height);
    font = Font.getFont( Font.FACE_MONOSPACE, Font.STYLE_PLAIN,
Font.SIZE_SMALL);
    charHeight = (short) font.getHeight();
    charWidth = (short) font.stringWidth("w");
    numberOfColumns = (short) ( width / charWidth );
    numberOfRows = (short) ( height / charHeight );
    content = new char[numberOfRows][];
    for (int i = 0; i < numberOfRows; i++)
    {
        content[i] = new char[numberOfColumns];
    }
    cls();
    new Thread(this).start();
}

public Graphics getGraphics()
{
    return offscreenG.getGraphics();
}

public void cls()
{

clearGraphixScreen();
clearTextScreen();
}

public void clearTextScreen()
{
    for (int i = 0; i < numberOfRows; i++)
    {
        for (int j = 0; j < numberOfColumns; j++)
        {
            content[i][j] = ' ';
        }
    }

cursorRow = 0;
cursorCol = 0;
Graphics g = offscreenT.getGraphics();
g.setColor(0, 0, 0);
g.fillRect(0, 0, width, height);
repaint();
}

public void clearGraphixScreen()
{
    clearTextScreen();
    Graphics g = offscreenG.getGraphics();
g.setColor(0, 0, 0);
g.fillRect(0, 0, width, height);
repaint();
}

private void processLineFeed()
{
    if (cursorRow == numberOfRows - 1)
    {
    }
    else
    {
        cursorRow++;
    }
    if (starInput == true)
    {
        ostream.append(content[cursorRow - 1]);
    }
}

public void update(Graphics g)
public void paint(Graphics g) {
    Graphics g1 = offscreenT.getGraphics();
    g1.setFont(font);
    g1.setColor(0);
    g1.fillRect(0, 0, width, height);
    g1.setColor(255, 255, 255);
    g1.drawImage(offscreenG, 0,0,0 );
    int y = 0;
    content[cursorRow][cursorCol] = '_';
    for (int i = 0; i < numberOfRows; i++) {
        g1.drawChars(content[i], 0, numberOfColumns, 0, y, 0);
        y += charHeight;
    }
    g.drawImage(offscreenT, 0, 0, 0);
}

public void run() {
    String s = null;
    int i = 0;
    while (true) {
        s = istream.read();
        for (i = 0; i < s.length(); i++) {
            userInput = false; writeChar(s.charAt(i));
        }
    }
}

public void writeChar(char c) {
    if (content[cursorRow][cursorCol] == '_') {
        content[cursorRow][cursorCol] = ' ';
    }
    switch (c) {
    case '\r': cursorCol = 0; break;
    case '\n': /* CR+LF */
        cursorCol = 0;
        processLineFeed();
    }
break;
    case '\f': clearTextScreen(); break;

    case '\b':
    if (--cursorCol < 0)
    {
        cursorCol = numberOfColumns - 1;
        cursorRow = (cursorRow - 1 + numberOfRows) % numberOfRows;
        while (content[cursorRow][cursorCol] == ' ' && cursorCol > 0) {
            cursorCol--;
        }
        if (content[cursorRow][cursorCol] != ' ')
        {
            try
            {
                System.out.print(content[cursorRow][cursorCol+1]);
                cursorCol++;
            } catch (ArrayIndexOutOfBoundsException e) { }
        }
    }
    break;
    case '\t':
    do {
        writeChar(' ');
    } while ((cursorCol % 8) != 7);
    break;
    default:
    if (c >= ' ')
    {
        content[cursorRow][cursorCol++] = (upperCase == true) ? Character.toUpperCase(c) : c;
    }
    }
    if (cursorCol >= numberOfColumns)
    {
        cursorCol = 0;
        processLineFeed();
        if (userInput == true)
        {
            ostream.append(content[cursorRow - 1]);
        }
    }
    repaint();

public void axis()
{
    Graphics g = offscreenG.getGraphics();
g.setColor(255, 255, 255);
g.drawLine(width/2, 0, width/2, height);
g.drawLine(0, height/2, width, height/2);
repaint();
}

public void destroyApp(boolean unconditional) {}

public void commandAction(Command command, Displayable displayable)
{
    if (command == BackCommand)
    {
        display.setCurrent(parent);
    }
    if (command == back)
    {
        display.setCurrent(this);
    }
    if (command == TextboxCommand)
    {
        display.setCurrent(get_StarTextBox());
    }
    if (command == Connection)
    {
        display.setCurrent(getlamp());
    }
    if (command == conne)
    {
        URL = "";
        URL = lamp.getString(lamp.getSelectedIndex());
        Connect();
        lamp.removeCommand(conne);
        lamp.addCommand(discon);
        display.setCurrent(this);
    }
    if (command == discon)
    {
        Closeconnection();
        if (reader == null && dos == null)
        {
            lamp.addCommand(conne);
            lamp.removeCommand(discon);
        }
    }
}
else if (command == okCommand)
{
    try
    {
catch (IOException ex) {
    ostream.write("No connection"+"\n");
    ostream.flush();
}
starWrite(tb.getString());
tb.setString(""");
display setCurrent(this);
}
else if (command == show) {
    String str = "Device";
    Ticker t = new Ticker(str);
    lamp removeCommand(show);
    lamp addCommand(back);
    lamp setTicker(t);
    try {
        BluetoothDeviceDiscovery.main();
    } catch (IOException ex) {
        ex.printStackTrace();
    }
    int deviceCount = BluetoothDeviceDiscovery vecDevices.size();
    for (int i = 0; i < deviceCount; i++) {
        RemoteDevice remoteDevice = (RemoteDevice) BluetoothDeviceDiscovery vecDevices.element At(i);
        lamp.append(remoteDevice getBluetoothAddress(), null);
    }
    lamp addCommand(conne);
    display setCurrent(lamp);
}

public TerminalIOStream getInputStream()
{
    return ostream;
}

public TerminalIOStream getOutputStream()
{
    return istream;
}
public void write(String s) {
    inputStream.write(s);
}

public void starWrite(String s) {
    starInput = true; userInput = false;
    for (int i = 0; i < s.length(); i++)
    {
        writeChar(s.charAt(i));
    }
    starInput = false;
}
}
LAMPIRAN C
SKEMATIK DAN FOTO
FOTO ALAT
LAMPIRAN D

AIC 1722

ORDERING INFORMATION

ORDER NUMBER | PIN CONFIGURATION
--- | ---
AIC1722-30000 | FRONT VIEW
AIC1722-3000X | 1: VOUT
AIC1722-300XX | 2: GND
AIC1722-300XX | 3: VIN
AIC1722-30CXX | TOP VIEW
AIC1722-30CXX | 1: GND
AIC1722-30CXX | 2: VIN
AIC1722-30CXX | 3: VOUT
AIC1722-30CLX | TO-92 L TYPE
AIC1722-30CLX | 1: VIN
AIC1722-30CLX | 2: GND
AIC1722-30CLX | 3: VIN

ABSOLUTE MAXIMUM RATINGS

Input Supply Voltage 0.3–12V
Operating Junction Temperature Range −40°C to 125°C
Storage Temperature Range −65°C to 150°C
Power Dissipation
SOT-85 Package 0.5W
TO-92 Package 0.5W

TEST CIRCUIT
Refer to the TYPICAL APPLICATION CIRCUIT

ELECTRICAL CHARACTERISTICS (Ta=25°C, C<sub>in</sub>=1µF, C<sub>out</sub>=10µF, unless otherwise specified)

<table>
<thead>
<tr>
<th>PARAMETER</th>
<th>TEST CONDITIONS</th>
<th>MIN.</th>
<th>TYP.</th>
<th>MAX.</th>
<th>UNIT</th>
</tr>
</thead>
<tbody>
<tr>
<td>Output Voltage</td>
<td>V&lt;sub&gt;IN&lt;/sub&gt;=0.0–12V</td>
<td>0.100</td>
<td>0.2</td>
<td>0.300</td>
<td>V</td>
</tr>
<tr>
<td>AIC1722-00</td>
<td>V&lt;sub&gt;IN&lt;/sub&gt;=5.5–12V</td>
<td>4.900</td>
<td>5.0</td>
<td>5.100</td>
<td>V</td>
</tr>
<tr>
<td>AIC1722-12</td>
<td>V&lt;sub&gt;IN&lt;/sub&gt;=4.0–12V</td>
<td>3.725</td>
<td>3.9</td>
<td>3.975</td>
<td>V</td>
</tr>
<tr>
<td>AIC1722-17</td>
<td>V&lt;sub&gt;IN&lt;/sub&gt;=4.0–12V</td>
<td>3.635</td>
<td>3.7</td>
<td>3.775</td>
<td>V</td>
</tr>
<tr>
<td>AIC1722-15</td>
<td>V&lt;sub&gt;IN&lt;/sub&gt;=4.0–12V</td>
<td>3.430</td>
<td>3.5</td>
<td>3.670</td>
<td>V</td>
</tr>
<tr>
<td>AIC1722-33</td>
<td>V&lt;sub&gt;IN&lt;/sub&gt;=3.6–12V</td>
<td>3.235</td>
<td>3.3</td>
<td>3.365</td>
<td>V</td>
</tr>
</tbody>
</table>
1. Hardware & Technical Information

1.1 Pin definition

<table>
<thead>
<tr>
<th>Pin No.</th>
<th>Name</th>
<th>Type</th>
<th>Note</th>
<th>Pin No.</th>
<th>Name</th>
<th>Type</th>
<th>Note</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>Ground</td>
<td></td>
<td></td>
<td>34</td>
<td>Ground</td>
<td></td>
<td></td>
</tr>
<tr>
<td>2</td>
<td>NC</td>
<td></td>
<td></td>
<td>33</td>
<td>RF_OUT</td>
<td>Out</td>
<td></td>
</tr>
<tr>
<td>3</td>
<td>Ground</td>
<td></td>
<td></td>
<td>32</td>
<td>Ground</td>
<td></td>
<td></td>
</tr>
<tr>
<td>4</td>
<td>Test_A</td>
<td></td>
<td>Internal testing</td>
<td>31</td>
<td>PIO0</td>
<td>In/Ou</td>
<td></td>
</tr>
<tr>
<td>5</td>
<td>Test_B</td>
<td></td>
<td>Internal testing</td>
<td>30</td>
<td>PIO1</td>
<td>In/Ou</td>
<td></td>
</tr>
<tr>
<td>6</td>
<td>Reset</td>
<td></td>
<td></td>
<td>29</td>
<td>PIO2</td>
<td></td>
<td></td>
</tr>
<tr>
<td>7</td>
<td>SPI_MISO</td>
<td></td>
<td>Internal testing</td>
<td>28</td>
<td>PIO3</td>
<td>In/Ou</td>
<td></td>
</tr>
<tr>
<td>8</td>
<td>SPI_MOSI</td>
<td></td>
<td>Internal testing</td>
<td>27</td>
<td>PIO4</td>
<td>In/Ou</td>
<td></td>
</tr>
<tr>
<td>9</td>
<td>SPI_CLK</td>
<td></td>
<td>Internal testing</td>
<td>26</td>
<td>PIO5</td>
<td>In/Ou</td>
<td></td>
</tr>
<tr>
<td>10</td>
<td>SPI_CS</td>
<td></td>
<td>Internal testing</td>
<td>25</td>
<td>PIO6</td>
<td>In/Ou</td>
<td></td>
</tr>
<tr>
<td>11</td>
<td>UART_CTS</td>
<td>Input</td>
<td></td>
<td>24</td>
<td>PIO7</td>
<td>Out</td>
<td>Driving LED</td>
</tr>
<tr>
<td>12</td>
<td>UART_TX</td>
<td>Out</td>
<td></td>
<td>23</td>
<td>USB-</td>
<td>In/Ou</td>
<td></td>
</tr>
<tr>
<td>13</td>
<td>UART_RTS</td>
<td>Out</td>
<td></td>
<td>22</td>
<td>USB+</td>
<td>In/Ou</td>
<td></td>
</tr>
<tr>
<td>14</td>
<td>UART_RX</td>
<td>Input</td>
<td></td>
<td>21</td>
<td>PCM_CLK</td>
<td>In/Ou</td>
<td></td>
</tr>
<tr>
<td>15</td>
<td>VCC_5V</td>
<td>Out</td>
<td></td>
<td>20</td>
<td>PCM_IN</td>
<td>In</td>
<td></td>
</tr>
<tr>
<td>16</td>
<td>VCC_3.3V</td>
<td>Input</td>
<td></td>
<td>19</td>
<td>PCM_SYNC</td>
<td>In/Ou</td>
<td></td>
</tr>
<tr>
<td>17</td>
<td>Ground</td>
<td></td>
<td></td>
<td>18</td>
<td>PCM_OUT</td>
<td>Out</td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Pin No.</th>
<th>Name</th>
<th>Type</th>
<th>Note</th>
</tr>
</thead>
<tbody>
<tr>
<td>35</td>
<td>PIO6</td>
<td>In/Ou</td>
<td></td>
</tr>
<tr>
<td>36</td>
<td>PIO9</td>
<td>In/Ou</td>
<td></td>
</tr>
<tr>
<td>37</td>
<td>PIO10</td>
<td>In/Ou</td>
<td></td>
</tr>
<tr>
<td>38</td>
<td>PIO11</td>
<td>In/Ou</td>
<td></td>
</tr>
</tbody>
</table>
1.5 Reference Schematics
4. Standard Setup Information

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>Baud Rate</td>
<td>9600</td>
</tr>
<tr>
<td>Pin Code Prompt</td>
<td>Disable</td>
</tr>
<tr>
<td>Local Name</td>
<td>SPP</td>
</tr>
<tr>
<td>LED PIN24(PIO 7)</td>
<td><strong>Power on:</strong> Flash 26 times[ON time frame: 80ms, OFF time frame: 140ms]</td>
</tr>
<tr>
<td></td>
<td><strong>Connect:</strong> Flash with ON one time within 1 second,[ON time frame: 35ms]</td>
</tr>
<tr>
<td></td>
<td><strong>Disconnected:</strong> Flash with ON one time within 3 seconds,[ON time frame : 35ms]</td>
</tr>
</tbody>
</table>

5. Customization Information

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>Baud Rate</td>
<td></td>
</tr>
<tr>
<td>Pin Code Prompt</td>
<td></td>
</tr>
<tr>
<td>Local Name</td>
<td></td>
</tr>
<tr>
<td>LED PIN24(PIO 7)</td>
<td><strong>Connected</strong></td>
</tr>
<tr>
<td></td>
<td><strong>Disconnected</strong></td>
</tr>
</tbody>
</table>
### Contact Specification
- Contact form: 5H, 1Z
- Contact resistance: ≤100mΩ / 1A @ 5VDC
- Contact material: AgCdO / AgCdO
- Contact capacity: 1A @ 5VAC/DC, 10A @ 5VAC/DC

### Technical Specification
- Insulation resistance: ≥100MΩ
- Dielectric strength: Extension coil and contact ≥1000VAC 1min.
- Welding time: Extension coil and contact ≥30s
- Terminal type: PCB mounting

### Coil Specification
- Rated coil power: 50mW (60mA @ 0.8VDC)

### Coil Data

<table>
<thead>
<tr>
<th>Nominal Voltage (VDC)</th>
<th>Operate Voltage (VDC)</th>
<th>Release Voltage (VDC)</th>
<th>Coil Resistance (Ω)</th>
</tr>
</thead>
<tbody>
<tr>
<td>6</td>
<td>2.29</td>
<td>0.3</td>
<td>25</td>
</tr>
<tr>
<td>8</td>
<td>3.76</td>
<td>0.6</td>
<td>75</td>
</tr>
<tr>
<td>10</td>
<td>4.5</td>
<td>0.9</td>
<td>100</td>
</tr>
<tr>
<td>12</td>
<td>6.9</td>
<td>0.8</td>
<td>220</td>
</tr>
<tr>
<td>16</td>
<td>10.5</td>
<td>1.9</td>
<td>900</td>
</tr>
<tr>
<td>24</td>
<td>19.0</td>
<td>2.4</td>
<td>1600</td>
</tr>
<tr>
<td>48</td>
<td>46.0</td>
<td>4.9</td>
<td>4800</td>
</tr>
</tbody>
</table>

### Outline Dimension
- Mounting holes (Bottom view)
- Wiring diagram (Bottom view)
ULN 2803

Motorola

Octal High Voltage, High Current Darlington Transistor Arrays

The eight NPN Darlington connected transistors in this family of arrays are ideally suited for interfacing between low logic level digital circuitry (such as TTL, CMOS or PMOS/NMOS) and the higher current/voltage requirements of lamps, relays, printer hammers or other similar loads for a broad range of computer, industrial, and consumer applications. All devices feature open-collector outputs and free-wheeling clamp diodes for transient suppression.

The ULN2803 is designed to be compatible with standard TTL families while the ULN2804 is optimized for 5 to 15 volt high level CMOS or PMOS.

MAXIMUM RATINGS (TA = 25°C and rating apply to any one device in the package, unless otherwise noted)

<table>
<thead>
<tr>
<th>Rating</th>
<th>Symbol</th>
<th>Value</th>
<th>Unit</th>
</tr>
</thead>
<tbody>
<tr>
<td>Output Voltage</td>
<td>VCE</td>
<td>50</td>
<td>V</td>
</tr>
<tr>
<td>Input Voltage (Except ULN2801)</td>
<td>VIL</td>
<td>3</td>
<td>V</td>
</tr>
<tr>
<td>Collector Current – Continuous</td>
<td>IC</td>
<td>500</td>
<td>mA</td>
</tr>
<tr>
<td>Base Current – Continuous</td>
<td>IB</td>
<td>25</td>
<td>mA</td>
</tr>
<tr>
<td>Operating Ambient Temperature Range</td>
<td>TA</td>
<td>0 to +70</td>
<td>°C</td>
</tr>
<tr>
<td>Storage Temperature Range</td>
<td>Tstg</td>
<td>-55 to +150</td>
<td>°C</td>
</tr>
<tr>
<td>Junction Temperature</td>
<td>TJ</td>
<td>125</td>
<td>°C</td>
</tr>
</tbody>
</table>

RJCases = 150°C/W
Do not exceed maximum current limit per driver.

ORDERING INFORMATION

<table>
<thead>
<tr>
<th>Device</th>
<th>Characteristics</th>
</tr>
</thead>
<tbody>
<tr>
<td>ULN2803A</td>
<td>TTL, 5.0 V CMOS</td>
</tr>
<tr>
<td>ULN2804A</td>
<td>6 to 15 V CMOS, PMOS</td>
</tr>
<tr>
<td></td>
<td>VCC(Max)/IC(Max)</td>
</tr>
<tr>
<td></td>
<td>Operating Temperature Range</td>
</tr>
</tbody>
</table>

1/8 ULN2803

© Motorola, Inc. 1986
Rev 1
5 J2ME and JABWT programming

This chapter will give a thorough description on how to program with JABWT and J2ME. Code samples are provided throughout the chapter, aiming to show the reader how a complete Bluetooth MIDlet is built. This chapter is partly based on [7]. The code samples in this chapter are simplified as much as possible, highlighting the Bluetooth specific Java code instead of describing both J2ME and JABWT issues simultaneously. The event-driven nature of J2ME applications tend to raise the complexity of the source code, making it difficult to understand the structure of the application. The simplified code samples provided in this chapter should make it easier for the reader to understand JABWT programming. It is assumed that the reader is familiar with J2ME programming. Readers who lack general knowledge about J2ME programming should consult [38].

The first code samples provided in this chapter show the structure of a Bluetooth MIDlet. Functionality will be added to these code samples as new functionality is explained. Note that these code samples are not fully functional MIDlets. However, they can be used as a starting point for a complete application. Method declarations have font typeface bold to increase the readability of the code.

5.1 Structure of Bluetooth MIDlet

This section shows the structure in a Java/Bluetooth MIDlet. Several MIDlet examples are available on Sun Microsystems' Mobility website [39].

Usually an event-driven MIDlet with no Bluetooth support looks like this:

```java
import javax.microedition.lcdui.Command;
import javax.microedition.lcdui.CommandListener;
import javax.microedition.lcdui.Displayable;
import javax.microedition.midlet.MIDlet;

class YourMidlet extends MIDlet implements CommandListener {
    public void startApp() {
    }
    public void pauseApp() {
```
The first three methods, `startApp()`, `pauseApp()` and `destroyApp()` are needed for any MIDlet. They come from extending the MIDlet class. The next method, `commandAction()` comes from the `CommandListener` interface. This is needed to catch command events. The MIDlet is extended to support Bluetooth communication in the next code sample.

During device discovery and service discovery, events will be delivered to a `DiscoveryListener` object when devices or services are found or the device discovery or service discovery is completed. An object implementing the `DiscoveryListener` interface is used to catch these events. The MIDlet will then look like this:

```java
import javax.bluetooth.DiscoveryListener;
import javax.bluetooth.DeviceClass;
import javax.bluetooth.ServiceRecord;
import javax.bluetooth.RemoteDevice;
import javax.microedition.lcdui.Command;
import javax.microedition.lcdui.CommandListener;
import javax.microedition.lcdui.Displayable;
import javax.microedition.midlet.MIDlet;

public class YourMidlet extends MIDlet implements CommandListener,
                               DiscoveryListener {
    
    public void startApp() {
    }

    public void pauseApp() {
    }

    public void destroyApp(boolean unconditional) {
    }

    public void commandAction(Command c, Displayable d) {
    }
```
public void deviceDiscovered(RemoteDevice remoteDevice,
        DeviceClass deviceClass) {
    
}

public void inquiryCompleted(int param) {
    
}

public void serviceSearchCompleted(int transID, int respCode) {
    
}

public void servicesDiscovered(int transID,
        ServiceRecord[] serviceRecord) {
    
}

The last four methods, deviceDiscovered(), inquiryCompleted(), serviceSearchCompleted() and servicesDiscovered() are used to catch events during device discovery and service discovery. Device discovery and service discovery will be outlined in the two next sections.

5.2 Device discovery (Inquiry)

Device discovery, introduced in Section 2.4, is the first step required when browsing nearby Bluetooth devices. When we have discovered nearby devices we can find out which services they offer. Note that no UI specific code is included in the following examples, only Bluetooth specific code.

To use any Bluetooth related methods you need to obtain a reference to the LocalDevice object by calling the LocalDevice.getLocalDevice() method. The obtained LocalDevice object gives access to Bluetooth properties for the device, such as the Bluetooth address, friendly name and discovery mode. We will use the LocalDevice object to obtain a DiscoveryAgent object. The DiscoveryAgent object is used for device discovery and service discovery.

The MIDlet now looks like this:

import java.util.Vector;
import javax.bluetooth.BluetoothStateException;
import javax.bluetooth.DiscoveryAgent;
import javax.bluetooth.DiscoveryListener;
import javax.bluetooth.LocalDevice;
import javax.bluetooth.DeviceClass;
import javax.bluetooth.ServiceRecord;
import javax.bluetooth.RemoteDevice;
import javax.microedition.lcdui.Command;
import javax.microedition.lcdui.CommandListener;
import javax.microedition.lcdui.Displayable;
import javax.microedition.midlet.MIDlet;

public class yourMIDlet extends MIDlet implements CommandListener,
               DiscoveryListener {  

    private LocalDevice local = null;
    private DiscoveryAgent agent = null;

    private Vector devicesFound = null;

    public void startApp() {

        /* Add your MIDlet specific code here.
         * You probably want to show the user
         * a welcome screen.
         * The call to doDeviceDiscovery() is
         * here for the example's sake. You
         * should call doDeviceDiscovery() when
         * the user actively asks for it.
         */

        doDeviceDiscovery();
    }

    public void pauseApp() {
    }

    public void destroyApp(boolean unconditional) {
    }

    public void commandAction(Command c, Displayable d) {
    }

    public void deviceDiscovered(RemoteDevice remoteDevice,
                                 DeviceClass deviceClass) {
    }

    public void inquiryCompleted(int param) {
    }

    public void servicesDiscovered(int transID,
                                    ServiceRecord[] serviceRecord) {
    }

    public void serviceSearchCompleted(int transID, int respCode) {
    }
}
private void doDeviceDiscovery() {
    try {
        local = LocalDevice.getLocalDevice();
    } catch (BluetoothStateException bse) {
        // Error handling code here
    }
    agent = local.getDiscoveryAgent();
    devicesFound = new Vector();
    try {
        if (!agent.startInquiry(DiscoveryAgent.GIAC, this)) {
            // Inquiry not started, error handling code here
        }
    } catch (BluetoothStateException bse) {
        // Error handling code here
    }
}

Device discovery is started using the LocalDevice and DiscoveryAgent objects. Observe that the doDeviceDiscovery() method is called in the startApp() method. Searching with the GIAC parameter will find devices which are general discoverable (see Section 2.4). The DiscoveryListener parameter is this, meaning our MIDlet. When devices are discovered or the search is complete, events will be delivered to our MIDlet. Note that the startInquiry() method returns immediately, returning true if the device discovery was initiated or false if the device discovery process was not started. An event will be delivered to the MIDlet when the device discovery is completed. This is important to take into account when designing the flow of execution in the MIDlet.

The deviceDiscovered() and inquiryCompleted() methods are used to catch events related to the device discovery process. When a device is discovered the deviceDiscovered() method of the object this will be called. The parameter remoteDevice will be the discovered device, it is up to us to decide what to do with it. Note that we do not know how many devices will be discovered. A Vector will therefore be an appropriate data structure to save discovered devices.

The inquiryCompleted() method is called when the inquiry ends. The status code supplied in the parameter param should always be checked. The complete code for device discovery follows:

```java
import java.util.Vector;
```
import javax.bluetooth.BluetoothStateException;
import javax.bluetooth.DiscoveryAgent;
import javax.bluetooth.DiscoveryListener;
import javax.bluetooth.LocalDevice;
import javax.bluetooth.DeviceClass;
import javax.bluetooth.ServiceRecord;
import javax.bluetooth.RemoteDevice;
import javax.microedition.lcdui.Command;
import javax.microedition.lcdui.CommandListener;
import javax.microedition.lcdui.Displayable;
import javax.microedition.midlet.MIDlet;

public class yourMIDlet extends MIDlet implements CommandListener, DiscoveryListener {

    private LocalDevice local = null;
    private DiscoveryAgent agent = null;

    Vector devicesFound = null;

    public void startApp() {
        /* Add your MIDlet specific code here.*
        * You probably want to show the user*
        * a welcome screen.*
        * The call to doDeviceDiscovery() is*
        * here for the example's sake. You*
        * should call doDeviceDiscovery when*
        * the user actively asks for it.*
        */

        doDeviceDiscovery();
    }

    public void pauseApp() {
    }

    public void destroyApp(boolean unconditional) {
    }

    public void commandAction(Command c, Displayable d) {
    }

    public void deviceDiscovered(RemoteDevice remoteDevice,
                                 DeviceClass deviceClass) {
        devicesFound.addElement(remoteDevice);
    }

    public void inquiryCompleted(int param) {
        /* We should give the user an alert based on the*
        * inquiry status code*
        */
    }
switch (param) {
    case DiscoveryListener.INQUIRY_COMPLETED:
        /* Inquiry completed normally, add appropriate code here */
        break;
    case DiscoveryListener.INQUIRY_ERROR:
        // Error during inquiry, add appropriate code here.
        break;
    case DiscoveryListener.INQUIRY_TERMINATED:
        /* Inquiry terminated by agent.cancelInquiry() */
        break;
}

public void servicesDiscovered(int transID, ServiceRecord[] serviceRecord) {
}

public void serviceSearchCompleted(int transID, int respCode) {
}

private void doDeviceDiscovery() {
    try {
        local = LocalDevice.getLocalDevice();
    } catch (BluetoothStateException bse) {
        // Error handling code here
    }
    agent = local.getDiscoveryAgent();
    devicesFound = new Vector();
    try {
        if(!agent.startInquiry(DiscovertAgent.GIAC, this)) {
            // Inquiry not started, error handling code here
        }
    } catch (BluetoothStateException bse) {
        // Error handling code here
    }
}
Discovered devices are kept in the DevicesFound vector by adding them as they are discovered. When our search ends, we check if everything went as expected and can alert the user by adding appropriate code in our switch-statement.

### 5.3 Service search

After device discovery is completed it is time to find out which services are offered by the discovered devices. This is accomplished by doing a service discovery on the device of interest.

The servicesDiscovered() and serviceSearchCompleted() methods must be implemented. They will handle the events occurring when services are found or the service discovery completes. In addition, the doServiceSearch() method has been added to show how a service discovery is initiated. This method will start a service discovery on the RemoteDevice supplied as a parameter, and is called in the inquiryCompleted() method.

The complete example Bluetooth MIDlet looks like this:

```java
import java.util.Vector;
import javax.bluetooth.UUID;
import javax.bluetooth.BluetoothStateException;
import javax.bluetooth.DiscoveryAgent;
import javax.bluetooth.DiscoveryListener;
import javax.bluetooth.LocalDevice;
import javax.bluetooth.DeviceClass;
import javax.bluetooth.ServiceRecord;
import javax.bluetooth.RemoteDevice;
import javax.microedition.lcdui.Command;
import javax.microedition.lcdui.CommandListener;
import javax.microedition.lcdui.Displayable;
import javax.microedition.midlet.MIDlet;

public class YourMIDlet extends MIDlet implements CommandListener,
                        DiscoveryListener {

    private LocalDevice local = null;
    private DiscoveryAgent agent = null;

    private Vector devicesFound = null;
    private ServiceRecord[] servicesFound = null;

    public void startApp() {
```
/* Add your MIDlet specific code here. * You probably want to show the user * a welcome screen. * The call to doDeviceDiscovery() is * here for the example's sake. You * should call doDeviceDiscovery() when * the user actively asks for it. */

doDeviceDiscovery();

public void pauseApp() {
}

public void destroyApp(boolean unconditional) {
}

public void commandAction(Command c, Displayable d) {
}

public void deviceDiscovered(RemoteDevice remoteDevice,
                             DeviceClass deviceClass) {
    devicesFound.addElement(remoteDevice);
}

public void inquiryCompleted(int param) {
    /* We should give the user an alert based on the * inquiry status code */
    switch (param) {
        case DiscoveryListener.INQUIRY_COMPLETED:
            /* Inquiry completed normally, so we * initiate a service discovery on the first * device found. That is, if we actually * found any devices. */
            if (devicesFound.size() > 0) {
                doServiceSearch((RemoteDevice)
                                devicesFound.firstElement());
            }
            break;

        case DiscoveryListener.INQUIRY_ERROR:
            // Error during inquiry, add appropriate code here
            break;
    }
}
case DiscoveryListener.INQUIRY_TERMINATED:
    /*
     * Inquiry terminated by agent.cancelInquiry()
     * Add appropriate code here
     */
    break;
}

public void servicesDiscovered(int transID,
        ServiceRecord[] serviceRecord) {
    /* Services discovered, keep reference to the ServiceRecord
     * array
     */
    servicesFound = serviceRecord;
}

public void serviceSearchCompleted(int transID, int respCode) {
    switch(respCode) {
    case DiscoveryListener.SERVICE_SEARCH_COMPLETED:
        /*
         * Service search completed successfully
         * Add appropriate code here
         */
        break;
    case DiscoveryListener.SERVICE_SEARCH_DEVICE_NOT_REACHABLE:
        // device not reachable, add appropriate code here
        break;
    case DiscoveryListener.SERVICE_SEARCH_ERROR:
        // Service search error, add appropriate code here
        break;
    case DiscoveryListener.SERVICE_SEARCH_NO_RECORDS:
        // No records found, add appropriate code here
        break;
    case DiscoveryListener.SERVICE_SEARCH_TERMINATED:
private void doDeviceDiscovery() {
    try {
        local = LocalDevice.getLocalDevice();
    } catch (BluetoothStateException bse) {
        // Error handling code here
    }
    agent = local.getDiscoveryAgent();
    devicesFound = new Vector();
    try {
        if (!agent.startInquiry(DiscoveryAgent.GIAC, this)) {
            // Inquiry not started, error handling code here
        }
    } catch (BluetoothStateException bse) {
        // Error handling code here
    }
}

private void doServiceSearch(RemoteDevice device) {
    /*
    * Service search will always give the default attributes:
    * ServiceRecordHandle (0x0000), ServiceClassIDList (0x0001),
    * ServiceRecordState (0x0002), ServiceID (0x0003) and
    * ProtocolDescriptorList (0x004).
    *
    * We want additional attributes, ServiceName (0x100),
    * ServiceDescription (0x101) and ProviderName (0x102).
    *
    * These hex-values must be supplied through an int array
    */
    int[] attributes = {0x100, 0x101, 0x102};
    /*
    * Supplying UUIDs in an UUID array enables searching for
    * specific services. PublicBrowseRoot (0x1002) is used in
    */
* this example. This will return any services that are
* public browseable. When searching for a specific service,
* the service's UUID should be supplied here.
*/

UUID[] uuids = new UUID[1];
uuids[0] = new UUID(0x1002);

try {
    agent.searchServices(attributes, uuids, device, this);
} catch (BluetoothStateException e) {
    // Error handling code here
}

The `searchServices()` method will return immediately, returning a transaction
ID for the service discovery. The transaction ID is used to identify the particular
service discovery. When the service discovery completes, an event will be delivered
to the MIDlet.

This example Bluetooth MIDlet will hopefully be of help to J2ME application
developers getting started with Bluetooth programming. Study the Bluetooth Browser
source code in Appendix A to see what a fully functional Bluetooth MIDlet looks
like.