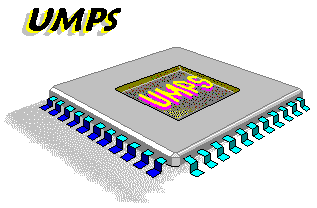
**APPENDIX D**

**IMPLEMENTATION**

**8051 C-Compiler & Assembler KEIL**

Keil develops, manufactures, and distributes embedded software development tools for 8051, 251, ARM, and XC16x/C16x/ST10 microcontroller families.  We provide ANSI C compilers, Macro Assemblers, real-time executives, debuggers and simulators, integrated environments, and evaluation boards. This web site provides the latest information about our development tools, evaluation tools, software updates, application notes, example programs, and links to other sources of information.

**UMPS**

**Universal Microcontroller Programmable Simulator**

UMPS is a universal microcontroller simulator, it runs under Windows 95, Windows 3.11 or Windows NT in a MDI environment. UMPS simulates a microcontroller with its external environment (we call this « resources »). You can simulate a whole system such a clock with:. a LCD panel, a real time I2C clock, 4 push button, a microcontroller. UMPS are able to simulate external components connected to the microcontroller. Then, debug step is dramatically reduced. UMPS is not dedicated to only one microcontroller family, it can simulate all kind of microcontrollers. The main limitation is to have less than 64K-Bytes of RAM and ROM space and the good microcontroller library.

UMPS include an integrated universal assembler/disassembler but is able to use external assembler and compiler and to show source code and variables. UMPS resources can be simply extended if you do not find the external resource you need. There is a complete documentation and example to write your own resources in « C » or « PASCAL » language.There is already two libraries which allows UMPS to use external compiler/assembler instead of its own integrated assembler: [MICROCHIP](http://www.microchip2.com/) Assembler and C Compiler,  [COSMIC Software](http://www.std.com/cosmic/index.html) Assembler and C Compiler

**EPR Serial Debugger**

**Serial port communication**

The serial debug functions are responsible for initializing and communicating with a debug message output device. Typically, this is a serial UART device connected over a NULL modem cable to a terminal emulator on the host computer. The OS or OAL implements the same code, so a single implementation can be shared between the OAL and the boot loader. The first step in debugging RS232 connections is to make sure that you are connected to the correct COM port on your PC and that you have the right cable for connecting the device to your PC.

This serial debugger is used to perform the following operations:

**Erase Write Verify Read**

Simply it debugs errors in serial port communications. And to erase or load hex file in controller.

**Visual Studio 2005**

**Professional editon from microsoft**

Microsoft Visual C# 2005, pronounced C sharp, is a programming language designed for building a wide range of applications that run on the .NET Framework. C# is simple, powerful, type-safe, and object-oriented. With its many innovations, C# enables rapid application development while retaining the expressiveness and elegance of C-style languages.

Visual Studio supports Visual C# with a full-featured Code Editor, project templates, designers, code wizards, a powerful and easy-to-use debugger, and other tools. The .NET Framework class library provides access to a wide range of operating system services and other useful, well-designed classes that speed up the development cycle significantly.

**HyperTerminal **

HyperTerminal is an application you can use in order to connect your computer to other remote systems. These systems include other computers, bulletin board systems, servers, Telnet sites, and online services. However, you would need a modem, an [Ethernet](http://www.tech-faq.com/ethernet.html) connection, or a null modem cable before you can use HyperTerminal.

**LCD AND SENSORS INTERFACING**

/ \* **Filename: lcd.h**

\* Hardware: Controller -> AT89S52

\* I/O RS -> P3.7

\* Enable -> P3.6

\* Data -> P2.0, P2.1, P2.2, P2.3, P2.4, P2.5, P2.6, P2.7

\* Sensors -> P1.0, P1.1, P1.2, P1.3

\* Compiler: keil

\*/

#include <AT89X55.H>

#define LCD\_DELAY 1535 /\* Delay for 1 ms \*/

#define LCD\_clear() LCD\_command(0x1) /\* Clear display LCD \*/

#define LCD\_origin() LCD\_command(0x2) /\* Set to origin LCD \*/

#define LCD\_row1 () LCD\_command(0x80) /\* Begin at Line 1 \*/

#define LCD\_row2 () LCD\_command(0xC0) /\* Begin at Line 2 \*/

sbit LCD\_en=P3^6;

sbit LCD\_rs=P3^7;

sbit SEN\_PIR=P1^0;

sbit SEN\_TEMP=P1^1;

sbit SEN\_IR=P1^2;

sbit SEN\_ULTRA=P1^3;

/\* Prototype(s)\*/

void LCD\_delay(unsigned char ms);

void LCD\_enable();

void LCD\_command(unsigned char command);

void LCD\_putc(unsigned char ascii);

void LCD\_puts(unsigned char \*lcd\_string);

void LCD\_init();

/\* Sources \*/

void LCD\_delay(unsigned char ms)

{

unsigned char n;

unsigned int i;

for (n=0; n<ms; n++)

{

for (i=0; i<LCD\_DELAY; i++); /\* For 1 ms \*/

}

}

void LCD\_enable()

{

LCD\_en = 0; /\* Clear bit P3.6 \*/

LCD\_delay(1);

LCD\_delay(1);

LCD\_delay(1);

LCD\_en = 1; /\* Set bit P3.6 \*/

}

void LCD\_command(unsigned char command)

{

LCD\_rs = 0; /\* Clear bit P3.7 \*/

P2 = (P2 & 0xF0)|((command>>4) & 0x0F);

LCD\_enable();

P2 = (P2 & 0xF0)|(command & 0x0F);

LCD\_enable();

LCD\_delay(1);

}

void LCD\_putc(unsigned char ascii)

{

LCD\_rs = 1; /\* Set bit P3.7 \*/

P2 = (P2 & 0xF0)|((ascii>>4) & 0x0F);

LCD\_enable();

P2 = (P2 & 0xF0)|(ascii & 0x0F);

LCD\_enable();

LCD\_delay(1);

}

void LCD\_puts(unsigned char \*lcd\_string)

{

while (\*lcd\_string)

{

LCD\_putc(\*lcd\_string++);

}

}

void LCD\_init()

{

LCD\_en = 1; /\* Set bit P3.6 \*/

LCD\_rs = 0; /\* Clear bit P3.7 \*/

LCD\_command(0x38);

LCD\_command(0x0E);

LCD\_clear();

LCD\_command(0x06);

LCD\_command(0x82);

LCD\_delay(256);

LCD\_delay(256);

}

**/ \* Filename: lcdmain.c\*/**

#include <AT89X55.H>

#include "lcd.h"

main( )

{

LCD\_init();

{

LCD\_row1();

LCD\_puts("SWG Vertical View Innovators | sri wils gokul | proudly presenting SWG Dreams. A Humanoid Bipedal Robo with ODMTS using AT89S52");

}

while (1);

if(SEN\_PIR==1)

{

LCD\_command(0xC0);

LCD\_puts("PIR");

LCD\_delay(256);

LCD\_delay(256);

LCD\_delay(256);

LCD\_puts(" ");

}

if(SEN\_TEMP==1)

{

LCD\_command(0xC4);

LCD\_puts("TEMP");

LCD\_delay(256);

LCD\_delay(256);

LCD\_delay(256);

LCD\_puts(" ");

}

if(SEN\_IR==1)

{

LCD\_command(0xC9);

LCD\_puts("IR");

LCD\_delay(256);

LCD\_delay(256);

LCD\_delay(256);

LCD\_puts(" ");

}

if(SEN\_ULTRA==1)

{

LCD\_command(0xC12);

LCD\_puts("ULTRA");

LCD\_delay(256);

LCD\_delay(256);

LCD\_delay(256);

LCD\_puts(" ");

}

}

**HEX FILE**

**/ \* Filename: lcdmain.hex\*/**

:0300000002095E94

:0C095E00787FE4F6D8FD75810A02094B91

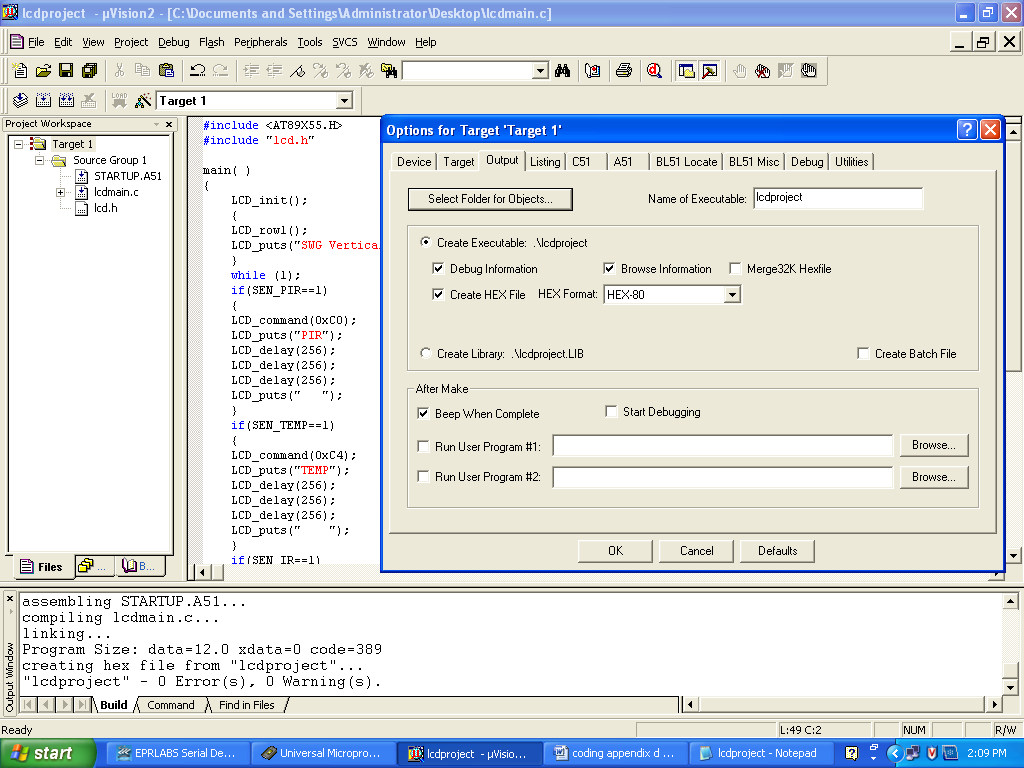
:1008000053574720566572746963616C2056696559

:100810007720496E6E6F7661746F7273207C2073DF

:1008200072692077696C7320676F6B756C207C2010

:1008300070726F75646C792070726573656E74691F

**CHECK FOR ERROR AND HEX FILE CREATED**



:100840006E672053574720447265616D732E2020D8

:10085000412048756D616E6F6964204269706564FE

:10086000616C20526F626F2077697468204F444D2D

:100870005453207573696E672041543839533532AB

:1008800000504952002020200054454D50002020A7

:10089000202000495200202000554C545241002095

:0508A0002020202000D3

:10091900E4FEEEC39F5011E4FDFC0DBD00010CBCCB

:0809290005F8BDFFF50E80EAA0

:0109310022A3

:1008A500EBC4540FFFE5A054F04FF5A01208BFEBC1

:0A08B500540FFFE5A054F04FF5A02A

:1008BF00C2B67F01120919120919120919D2B622EB

:0C096A00AB07C2B71208A57F01020919F3

:0C097600AB07D2B71208A57F01020919D7

:1008CF008B088A09890AAB08AA09A90A120932609A

:1008DF0013050AE50A7002050914F9120932FF120D

:0508EF00097680E22201

:1008F400D2B6C2B77F3812096A7F0E12096A7F0125

:1009040012096A7F0612096A7F8212096AE4FF12D9

:05091400091902091998

:10094B001208F47F8012096A7BFF7A08790012087B

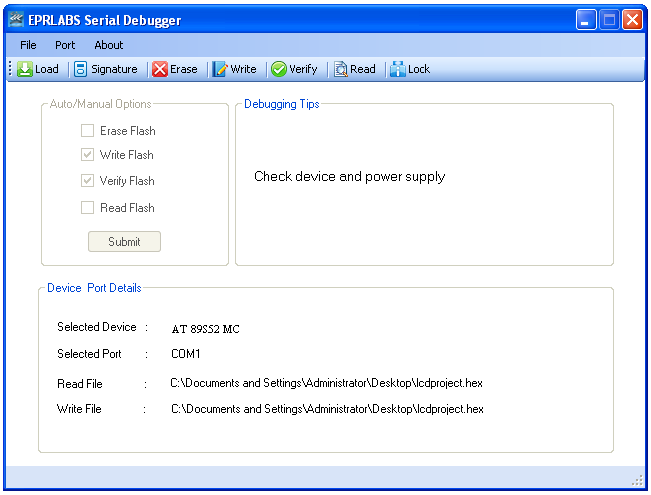
:03095B00CF80FE4C

:10093200BB010689828A83E0225002E722BBFE02C3

:09094200E32289828A83E49322F6

:00000001FF

**LOADING HEX FILE IN MICROCONTROLLER**



**SENSOR USING ADC**

**/ \* Filename: adc.asm\*/**

rd equ P3.7 ;Read signal P3.7

wr equ P3.6 ;Write signal P3.6

cs equ P1.0 ;Chip Select P1.0

intr equ P3.2 ;INTR signal P3.2

adc\_port equ P0 ;ADC data pins P0

adc\_val equ 30H ;ADC read value stored here

; -------main prog

org 0x00

mov r1,#38H ;select 2 lines 5x7 matrix lcd

call command

mov r1,#0EH ;display on cursor blinking

call command

mov r1,#06H ;auto increment the cursor

call command

mov r1,#01H ;clear display

call command

; --------------------start of the program

start: ;Start of Program

acall conv ;Start ADC conversion

acall read ;Read converted value

acall conversion

call data2

call data1

call data

call delay

call delay

call delay

call delay

call delay

call delay

mov r1,#01H ;clear display

call command

call delay

jmp start

; Do it again

; ---------------analog to digital conv

conv: ;Start of Conversion

clr cs ;Make CS low

clr wr ;Make WR Low

nop

setb wr ;Make WR High

setb cs ;Make CS high

wait: jb intr,wait

ret ;Conversion done

; ----------------read from adc

read: ;Read ADC value

clr cs ;Make CS Low

clr rd ;Make RD Low

mov a,adc\_port ;Read the converted value

mov adc\_val,a ;Store it in local variable

setb rd ;Make RD High

setb cs ;Make CS High

ret ;Reading done

; --------------delay prog

delay:

mov r1,#255

l2: mov r0,#255

l1: djnz r0,l1

djnz r1,l2

ret

; -------------command for lcd

command:

clr p1.1

mov p2,r1

setb p1.2

call delay

clr p1.2

ret

; ------------data for lcd

data2:

setb p1.1

mov p2,r4

setb p1.2

clr p1.2

call delay

ret

data1:

setb p1.1

mov p2,r3

setb p1.2

clr p1.2

call delay

ret

data:

setb p1.1

mov p2,r2

setb p1.2

clr p1.2

call delay

ret

;-adc to ascii conv

conversion:

mov b,#10

div ab

mov r2,b

mov b,#10

div ab

orl a,#30h

mov r4,a

mov a,b

orl a,#30h

mov r3,a

mov a,r2

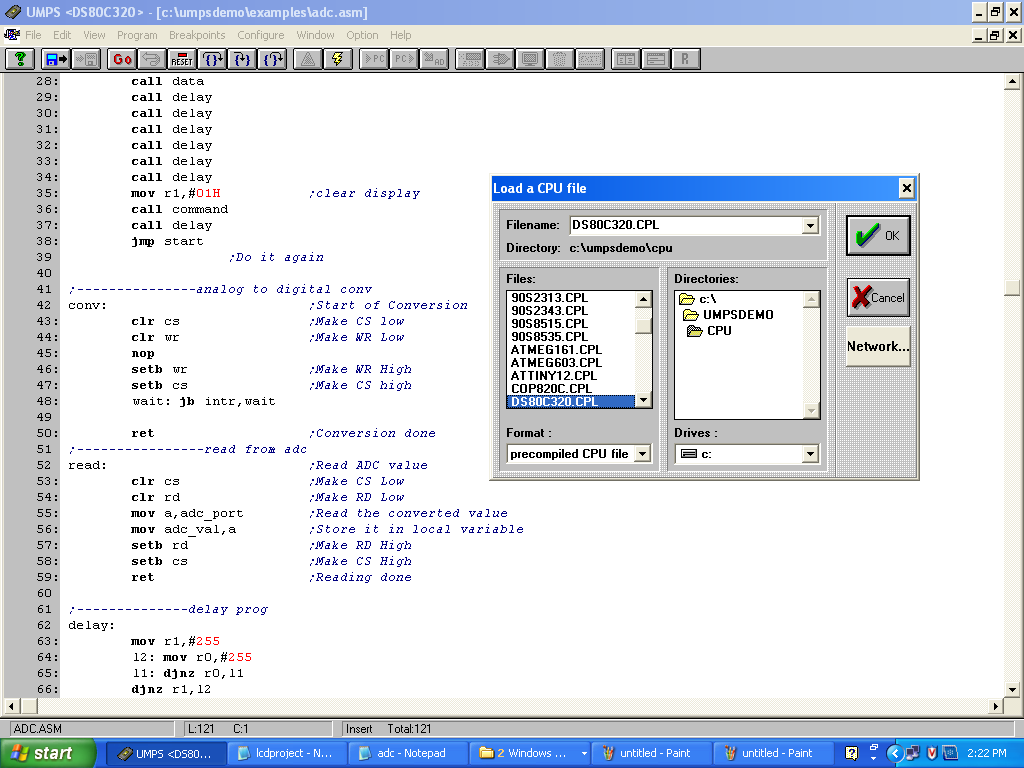
orl a,#30h

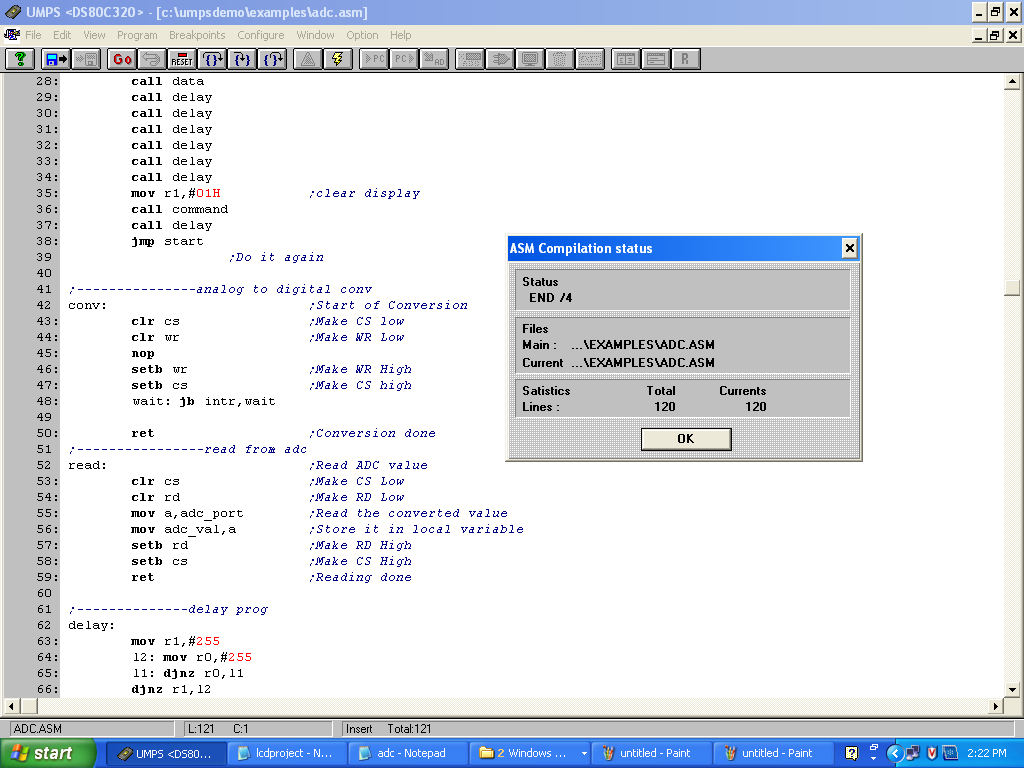
mov r2,a

ret

end

**LOAD CPU REGISTER**





**HEX FILE**

**/ \* Filename: adc.hex\*/**

:100000007938120063790E12006379061200637961

:0100100001EE

:100011001200631140114D119312006F12007B12F7

:02002100008756

:1000230012005A12005A12005A12005A12005A129F

:02003300005A71

:10003500790112006312005A020014C290C2B60080

:10004500D2B6D29020B2FD22C290C2B7E580F5307B

:10005500D2B7D2902279FF78FFD8FED9FA22C29181

:1000650089A0D29212005AC29222D2918CA0D29229

:10007500C29212005A22D2918BA0D292C292120041

:010085005A20

:1000860022D2918AA0D292C29212005A2275F00A06

:1000960084AAF075F00A844430FCE5F04430FBEAAB

:0400A6004430FA22C6 :00000001FF

**RF TRANSMITTER**

**/ \* Filename: rftx.c\*/**

#include<stdio.h>

#include<reg52.h>

void serial\_init()

{

TMOD = 0x20; //Timer MODe register

SCON = 0x50; //Serial port CONtoller register

TH1 = 0xFD;

TR1 = 1;

}

void serial\_write(unsigned char dat)

{

SBUF = dat; //Serial Buffer

while(!TI);

TI = 0; //TX

}

unsigned char serial\_read()

{

while(!RI);

RI = 0; //RX

return SBUF;

}

void main()

{

unsigned char a;

serial\_init();

serial\_write('O');

serial\_write('K');

while(1)

{

a = serial\_read();

switch(a)

{ case '0':

{ serial\_write(48); ///front

break;

}

case '1':

{ serial\_write(49); ///left

break;

}

case '2':

{ serial\_write(50); ///walk

break;

}

case '3':

{ serial\_write(51); ///right

break;

}

case '4':

{ serial\_write(52); ///back

break;

}

case '5':

{

serial\_write(53); ///handshake

break;

}

case '6':

{ serial\_write(54); ///stop

break;

}

case '7':

{ serial\_write(55); ///speak

break;

}

default:

{ serial\_write(54); ///stop

break;

}

}

}

}

**HEX FILE**

**/ \* Filename: rftx.hex\*/**

:030000000208678C

:0C086700787FE4F6D8FD758107020800D8

:0C087300758920759850758DFDD28E227D

:08087F008F993099FDC2992206

:080887003098FDC298AF9922E0

:100800001208737F4F12087F7F4B804F120887EFCB

:1008100024D0B40800504990082575F003A4C5837E

:1008200025F0C5837302083D02084102084502080D

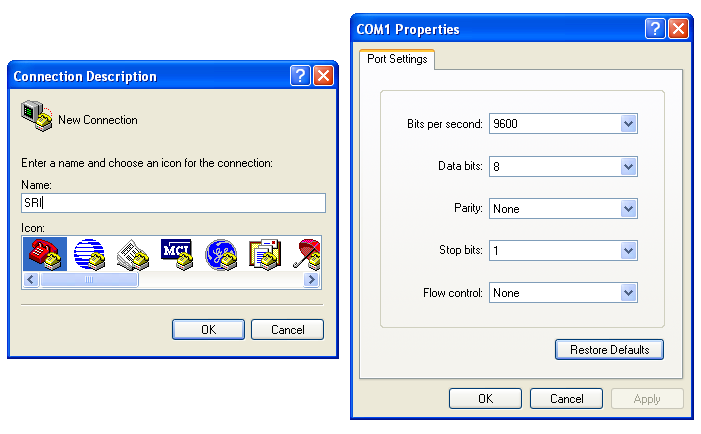
:100830004902084D0208510208550208597F3080CC

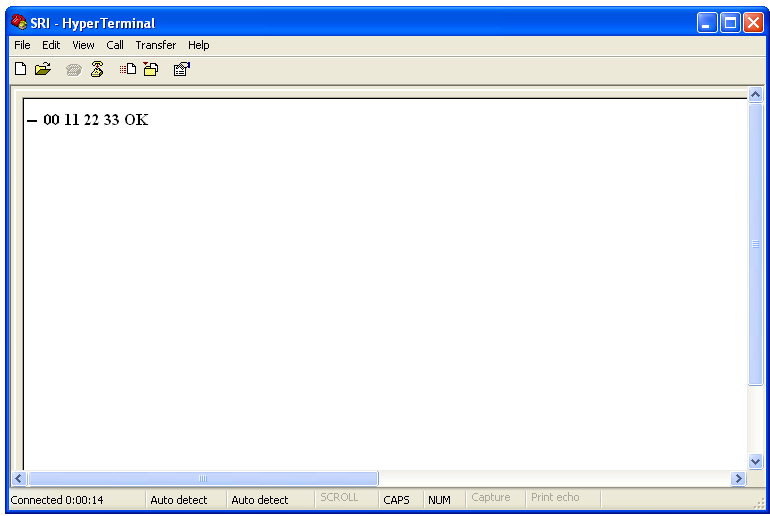
:100840001A7F3180167F3280127F33800E7F348092

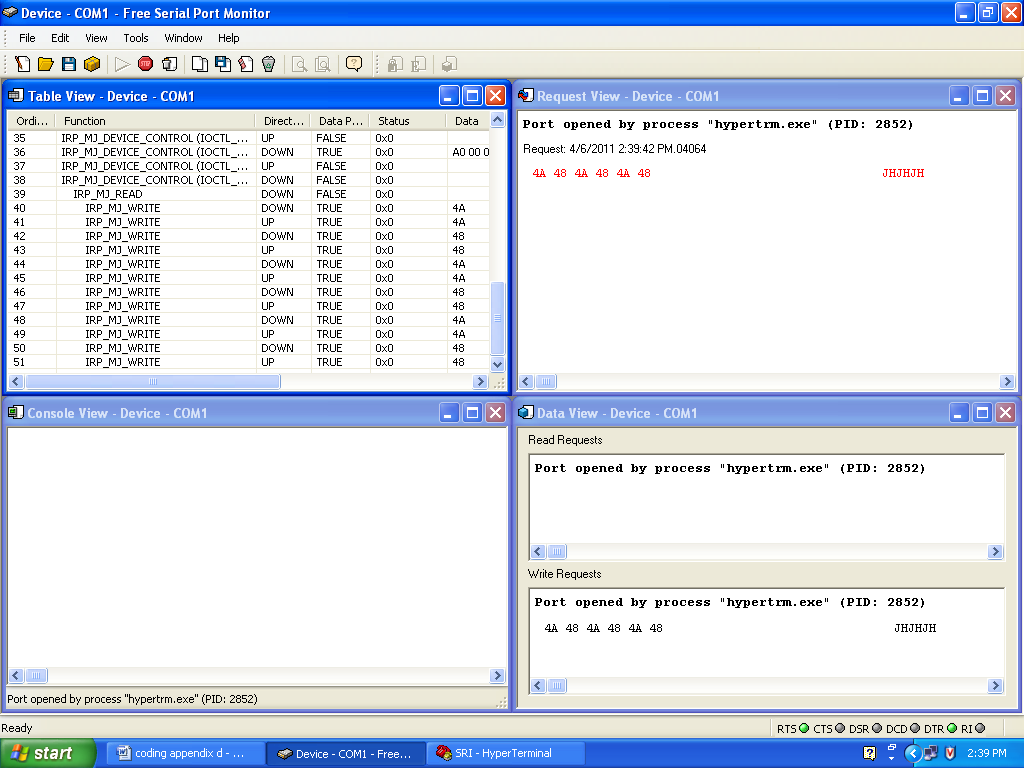
:100850000A7F3580067F3680027F3712087F80ACA2

:070860007F3612087F80A51E

:00000001FF

****

****



**RF RECEIVER**

**/\*file name: rx.c\*/**

//header\_files

#include <REGX51.H>

#include<stdio.h>

//var\_declaration

unsigned int a,i,j;

//prototype

void walk(void);

void front(void);

void back(void);

void left(void);

void right(void);

void handshake(void);

void stop(void);

//port3\_assigned

sbit s0=P0^7;sbit s1=P0^6;

sbit s2=P0^5;sbit s3=P0^4;

sbit s4=P0^3;sbit s5=P0^2;

sbit s6=P0^1;sbit s7=P0^0;

sbit t0=P2^0;sbit t1=P2^1;

sbit t2=P2^2;sbit t3=P2^3;

sbit t4=P2^4;sbit t5=P2^5;

sbit t6=P2^6;sbit t7=P2^7;

sbit r0=P1^2;sbit r1=P1^3;

//delay

void wait(unsigned int a)

{

for(i=0;i<a;i++)

{ for(j=0;j<a;j++)

{

;

}

}

}

void walk()

{

s0=1;s1=0;s2=1;s3=0;

t4=1;t5=0;t0=1;t1=0;

r0=0;r1=1;

wait(60000);

t4=0;t5=0;t0=0;t1=0;

s0=0;s1=0;s2=0;s3=0;

r0=0;r1=0;

wait(50000);

s4=0;s5=1;s6=0;s7=1;

t4=0;t5=1;t0=0;t1=1;

r0=1;r1=0;

wait(60000);

t4=0;t5=0;t0=0;t1=0;

s4=0;s5=0;s6=0;s7=0;

r0=0;r1=0;

wait(50000);

}

void back()

{ s0=0;s1=1;s2=0;s3=1;s4=1;s5=0;s6=1;s7=0;

wait(60000);

s0=0;s1=0;s2=0;s3=0;s4=0;s5=0;s6=0;s7=0;

wait(50000);

}

void front() { }

void left() { }

void right() { }

void stop() { }

void handshake(){ }

//port assigned for serial port communication \_rf

void serial\_init()

{

TMOD = 0x20;

SCON = 0x50;

TH1 = 0xFD;

TR1 = 1;

}

void serial\_write(unsigned char dat)

{

SBUF = dat;

while(!TI);

TI = 0;

}

unsigned char serial\_read()

{

while(!RI);

RI = 0;

return SBUF;

}

//main\_function

void main()

{

unsigned char a;

serial\_init();

serial\_write('O');

serial\_write('K');

while(1)

{

a = serial\_read();

switch(a)

{

case 48:

{

front(); ///front

break;

}

case 49:

{

left(); ///left

break;

}

case 50:

{

walk(); ///walk

break;

}

case 51:

{

right(); ///right

break;

}

case 52:

{

back(); ///back

break;

}

case 53:

{

handshake(); ///handshake

break;

}

case 54:

{

stop(); ///stop

break;

}

case 55:

{

speak(); ///speak

break;

}

default:

{

stop();

break;

}

}

}

}

**HEX FILE**

**/\*file name: rx.hex\*/**

:0300000002092EC4

:0C092E00787FE4F6D8FD75810D0208000A

:0408CF007F607EEADE

:1008D300E4F50AF50BC3E50B9FE50A9E5022E4F508

:1008E3000CF50DC3E50D9FE50C9E500A050DE50DB6

:1008F30070F1050C80ED050BE50B70D9050A80D569

:0109030022D1

:10086B00D287C286D285C284D2A4C2A5D2A0C2A18D

:10087B00C292D2931208CFC2A4C2A5C2A0C2A1C277

:10088B0087C286C285C284C292C2937F507EC31236

:10089B0008D3C283D282C281D280C2A4D2A5C2A005

:1008AB00D2A1D292C2931208CFC2A4C2A5C2A0C237

:1008BB00A1C283C282C281C280C292C2937F507E88

:0408CB00C30208D389

:10090400C287D286C285D284D283C282D281C28077

:100914001208CFC287C286C285C284C283C282C281

:0A09240081C2807F507EC30208D319

:01095600227E

:01095700227D

:01095800227C

:01095900227B

:01095A00227A

:0C093A00758920759850758DFDD28E22B5

:080946008F993099FDC299223E

:08094E003098FDC298AF992218

:1008000012093A7F4F1209467F4B12094612094ED0

:10081000EF24D0B40800504E90082675F003A4C50C

:100820008325F0C5837302083E020843020848028C

:10083000084D02085202085702085C0208611209BA

:100840005680CA12095780C512086B80C012095819

:1008500080BB12090480B612095A80B1120959806E

:0B086000AC12000080A712095980A212 :00000001FF

**USER INTERFACE DESIGN**

**/\*File name:Robotui.cs\*/**

using System;

using System.Collections.Generic;

using System.ComponentModel;

using System.Data;

using System.Drawing;

using System.Text;

using System.Windows.Forms;

using System.IO.Ports;

using swgdreams;

namespace UI

{

public partial class Form1 : Form

{

SerialPort serialport;

SelectPort spobject = new SelectPort();

String port;

public Form1()

{

InitializeComponent();

}

private void button1\_Click(object sender, EventArgs e)

{

if (configSerialport())

{

if (!serialport.IsOpen)

serialport.Open();

byte[] wbuffer = new byte[1];

wbuffer[0] = 0x55;

serialport.Write(wbuffer, 0, wbuffer.Length);

serialport.Close();

}

}

private void button1\_Click\_1(object sender, EventArgs e)

{

//FRONT

if (configSerialport())

{

if (!serialport.IsOpen)

serialport.Open();

byte[] wbuffer = new byte[1];

wbuffer[0] =0X01;

serialport.Write(wbuffer, 0, wbuffer.Length);

serialport.Close();

}

}

private void button2\_Click(object sender, EventArgs e)

{

//LEFT

if (configSerialport())

{

if (!serialport.IsOpen)

serialport.Open();

byte[] wbuffer = new byte[1];

wbuffer[0] = 0x02;

serialport.Write(wbuffer, 0, wbuffer.Length);

serialport.Close();

}

}

private void button3\_Click(object sender, EventArgs e)

{

//STOP

if (configSerialport())

{

if (!serialport.IsOpen)

serialport.Open();

byte[] wbuffer = new byte[1];

wbuffer[0] = 0X03;

serialport.Write(wbuffer, 0, wbuffer.Length);

serialport.Close();

}

}

private void pORTToolStripMenuItem\_Click(object sender, EventArgs e)

{

spobject.ShowDialog();

//creating object for GetPortDevice

GetPortDevice gvobj = spobject.getvaluesobj();

//checking object is null or not if not null getting port value

if (gvobj != null)

{

port = gvobj.getSelectedPort();

//displaying value in label

Selected\_Port\_Label.Text = port;

}

}

private void button4\_Click(object sender, EventArgs e)

{

//RIGHT

if (configSerialport())

{

if (!serialport.IsOpen)

serialport.Open();

byte[] wbuffer = new byte[1];

wbuffer[0] = 0x04;

serialport.Write(wbuffer, 0, wbuffer.Length);

serialport.Close();

}

}

private void button5\_Click(object sender, EventArgs e)

{

//BACK

if (configSerialport())

{

if (!serialport.IsOpen)

serialport.Open();

byte[] wbuffer = new byte[1];

wbuffer[0] = 0x05;

serialport.Write(wbuffer, 0, wbuffer.Length);

serialport.Close();

}

}

public Boolean configSerialport()

{

Boolean b = false;

if (Selected\_Port\_Label.Text.Trim().Length != 0)

{

port = Selected\_Port\_Label.Text;

serialport = new SerialPort(port, 57600, Parity.None);

b = true;

}

else

{

b = false;

MessageBox.Show("please select Port");

}

return b;

}

private void button6\_Click(object sender, EventArgs e)

{

//HANDSHAKE

if (configSerialport())

{

if (!serialport.IsOpen)

serialport.Open();

byte[] wbuffer = new byte[1];

wbuffer[0] = 0x06;

serialport.Write(wbuffer, 0, wbuffer.Length);

serialport.Close();

}

}

private void button7\_Click(object sender, EventArgs e)

{

//WALK

if (configSerialport())

{

if (!serialport.IsOpen)

serialport.Open();

byte[] wbuffer = new byte[1];

wbuffer[0] = 0x07;

serialport.Write(wbuffer, 0, wbuffer.Length);

serialport.Close();

}

}

private void button8\_Click(object sender, EventArgs e)

{

//SPEAK

if (configSerialport())

{

if (!serialport.IsOpen)

serialport.Open();

byte[] wbuffer = new byte[1];

wbuffer[0] = 0x08;

serialport.Write(wbuffer, 0, wbuffer.Length);

serialport.Close();

}

}

