PROGRAM PENGENDALI WIROBOT X80

```
Option Explicit

Dim X(3) As Double

Dim n As Double

Dim m As Double

Dim l As Integer

Dim i As Integer

Dim j As Integer

Dim o As Double

Dim h1(3, 3), h2(3, 3) As Single


Private mintCurFrame As Integer ' Current Frame visible


'untuk mendapatkan posisi kursor

Private Declare Function GetCursorPos Lib _
"user32" (lpPoint As POINTAPI) As Long


'variable untuk menampung nilai point

Private pos As POINTAPI


'untuk mendeteksi kursor

Private Declare Function GetAsyncKeyState Lib "user32" (ByVal vkey As Long) As
Integer


'windows virtual key constant untuk kursor

Const VK_LMOUSEBUTTON = 1
```

```
Const VK_RMOUSEBUTTON = 2


Const SINGLE_SENSOR As Long = 0

Const DUAL_SENSOR As Long = 1

Const DECODER_SENSOR As Long = 2


Const NO_CTRL As Long = -32768


Const DIRECT_PWM As Long = 0

Const DIRECT_POSITION As Long = 1

Const DIRECT_VELOCITY As Long = 2


Const cHALF_RANGE As Integer = 599

Const cWHOLE_RANGE As Integer = 1200

Const cFULL_COUNT As Integer = 32767

Dim ControlButtonName As String


Dim leftWheelData As Integer

Dim rightWheelData As Integer


'Set the Max, Min and Init value utk setiap servo

Dim Servo1Max, Servo1Min, Servo1Init As Integer

Dim Servo2Max, Servo2Min, Servo2Init As Integer


Dim continue_tag As Integer
```

```
Dim done_tag As Integer

Dim firstTime As Integer


Private Type BITMAP

    bmType As Long

    bmWidth As Long

    bmHeight As Long

    bmWidthBytes As Long

    bmPlanes As Integer

    bmBitsPixel As Integer

    bmBits As Long

End Type

Private Type POINTAPI

X As Long

Y As Long

End Type

Private Sub btnDisplayStream_Click()

    'Set "Display Streaming" button

    If btnDisplayStream.Caption = "Disable Streaming Display" Then

        btnDisplayStream.Caption = "Enable Streaming Display"

        Timer1.Interval = 0

    Else

        btnDisplayStream.Caption = "Disable Streaming Display"

        Timer1.Interval = 250

    End If
```

```
End Sub

Private Sub btnHeadDown_Click()

Call Kamerabawah

End Sub

Sub Kamerabawah()

   'set the change value when click the "Head Down" Button

   If hscHeadVert.Value > hscHeadVert.Min + 200 Then hscHeadVert.Value =
hscHeadVert.Value - 200

End Sub
```

```
= 200

End Sub

Private Sub btnHeadDown_MouseUp(Button As Integer, Shift As Integer, X As
Single, Y As Single)

   tmrControlServo.Enabled = False

End Sub

Private Sub btnHeadLeft_Click()

Call Kamerakiri

End Sub

Sub Kamerakiri()

   If hscHeadHor.Value < hscHeadHor.Max - 200 Then hscHeadHor.Value =

   ControlButtonName = "btnHeadLeft_Click"

   tmrControlServo.Enabled = True

   tmrControlServo.Interval = 500

End Sub
```

Private Sub btnHeadLeft_MouseUp(Button As Integer, Shift As Integer, X As Single, Y As Single)

   tmrControlServo.Enabled = False

End Sub

Private Sub btnHeadReset_Click()

   tmrControlServo.Enabled = False

   hscHeadHor.Value = Servo2Init

   ControlButtonName = "btnHeadRight_Click"

   tmrControlServo.Enabled = True

   tmrControlServo.Interval = 500

End Sub

Private Sub btnHeadRight_MouseUp(Button As Integer, Shift As Integer, X As Single, Y As Single)

   tmrControlServo.Enabled = False

End Sub

Private Sub btnHeadUp_Click()

Call Kameraatas

End Sub

Sub Kameraatas()

   If hscHeadVert.Value < hscHeadVert.Max - 200 Then hscHeadVert.Value = hscHeadVert.Value + 200

End Sub

Private Sub btnHeadUp_MouseUp(Button As Integer, Shift As Integer, X As Single, Y As Single)

   tmrControlServo.Enabled = False

End Sub

```vb
Private Sub btnSentImageLCD_Click()

    Dim hBitmap As Long

    Dim res As Long

    Dim bmp As BITMAP

    Dim byteAry() As Byte

Long, i As Long

    Dim ImageErr As Boolean


    If txtLCDImageName.Text <> "" Then

        Dim fs

        Set fs = CreateObject("Scripting.FileSystemObject")

        If                  fs.FileExists(txtLCDImageName.Text)                  And
InStrRev(txtLCDImageName.Text, ".bmp") > 0 Then

            'set Err flag

            ImageErr = False


            'set slected image to picture box

            picDetectImage.Picture = LoadPicture(txtLCDImageName.Text)


            hBitmap = picDetectImage.Picture.Handle

            res = GetObject(hBitmap, Len(bmp), bmp) 'mendapatkan BITMAP structure


            totbyte = bmp.bmWidthBytes * bmp.bmHeight 'total size of BYTE for saving
image


            'put picDetectImage's info into ByteAry
```

```
        res = GetBitmapBits(hBitmap, totbyte, byteAry(0))


        If bmp.bmBitsPixel <> 1 Then

            MsgBox "Please select a monochrome bitmap file, and try again",
vbCritical + vbOKOnly, "Error"


        If bmp.bmWidth <> 128 Or bmp.bmHeight <> 64 Then

            MsgBox "Please select 128 pixel x 64 pixel bitmap file, and try again",
vbCritical + vbOKOnly, "Error"

            ImageErr = True

            Exit Sub

        End If


        If ImageErr = False Then

            'kirim image ke layar ketika formatnya benar

            WiRobotSDK1.LcdDisplayPMS txtLCDImageName.Text

            picLCDDisplay.Picture = LoadPicture(txtLCDImageName.Text)

        End If

    Else

        MsgBox "Please select exist BMP format image for LCD Display", vbCritical
+ vbOKOnly, "Error"


    btnServoControl1.Caption = "Disable Servo #1"

Private Sub btnServoControl2_Click()

  If btnServoRun2.Enabled = False Then

    btnServoControl2.Caption = "Disable Servo #2"

    btnServoRun2.Enabled = True
```

```
        WiRobotSDK1.EnableServo 1

    Else

        btnServoControl2.Caption = "Enable Servo #2"

        WiRobotSDK1.DisableServo 1

        btnServoRun2.Enabled = False

    End If

End Sub

Private Sub btnServoRun1_Click()

    'detesi nilai input

    If Servo1Max >= CInt(txtServoValue1.Text) And CInt(txtServoValue1.Text) >=
Servo1Min Then

        WiRobotSDK1.ServoTimeCtr          0,          CInt(txtServoValue1.Text),
CInt(txtServoTime1.Text)

    Else

        MsgBox "Please Note: Command Value should be between " & Servo1Min & "
~ " & Servo1Max, vbCritical + vbOKOnly, "Error"

    End If

End Sub

Private Sub btnServoRun2_Click()

    'deteksi nilai input

    If Servo2Max >= CInt(txtServoValue2.Text) And CInt(txtServoValue2.Text) >=
Servo2Min Then

        WiRobotSDK1.ServoTimeCtr          1,          CInt(txtServoValue2.Text),
CInt(txtServoTime2.Text)

    Else

        MsgBox "Please Note: Command Value should be between " & Servo2Min & "
~ " & Servo2Max, vbCritical + vbOKOnly, "Error"

    End If
```

End Sub

Private Sub btnWheelBack_Click()

Call mundur

End Sub

Sub mundur()

   WiRobotSDK1.SetDcMotorControlMode 0, DIRECT_VELOCITY

   WiRobotSDK1.SetDcMotorControlMode 1, DIRECT_VELOCITY

   WiRobotSDK1.DcMotorVelocityNonTimeCtrAll   CInt(hscSpeed.Value),   -CInt(hscSpeed.Value), NO_CTRL, NO_CTRL, NO_CTRL, NO_CTRL

   Sleep 900

End Sub

Private Sub btnWheelFront_Click()

Call maju

End Sub

Sub maju()

   WiRobotSDK1.SetDcMotorControlMode 0, DIRECT_VELOCITY

   WiRobotSDK1.SetDcMotorControlMode 1, DIRECT_VELOCITY

   WiRobotSDK1.EnableDcMotor 0

   WiRobotSDK1.EnableDcMotor 1

   WiRobotSDK1.DcMotorVelocityNonTimeCtrAll   -CInt(hscSpeed.Value), CInt(hscSpeed.Value), NO_CTRL, NO_CTRL, NO_CTRL, NO_CTRL

End Sub

Private Sub btnWheelLeft_Click()

```
    Call kiri

End Sub

Sub kiri()

    Dim cmd1 As Long

    Dim cmd2 As Long

    'setting control mode

    WiRobotSDK1.SetDcMotorControlMode 0, DIRECT_POSITION

    WiRobotSDK1.SetDcMotorControlMode 1, DIRECT_POSITION

    WiRobotSDK1.EnableDcMotor 0

    WiRobotSDK1.EnableDcMotor 1

    cmd1 = leftWheelData + cWHOLE_RANGE / 5

    cmd2 = rightWheelData + cWHOLE_RANGE / 5

    ' ngubah cmd1, cmd2 ke valid data range

    If (cmd1 < 0) Then cmd1 = cmd1 + cFULL_COUNT

    If (cmd2 < 0) Then cmd2 = cmd2 + cFULL_COUNT

    If (cmd1 > cFULL_COUNT) Then cmd1 = cmd1 - cFULL_COUNT

    If (cmd2 > cFULL_COUNT) Then cmd2 = cmd2 - cFULL_COUNT

    WiRobotSDK1.DcMotorPositionTimeCtrAll CInt(cmd1), CInt(cmd2), NO_CTRL,
NO_CTRL, NO_CTRL, NO_CTRL, 1000

    Sleep 1000

End Sub

Sub Kiri1()

 Dim cmd1 As Long

    Dim cmd2 As Long

    'setting control mode

    WiRobotSDK1.SetDcMotorControlMode 0, DIRECT_POSITION
```

```
    WiRobotSDK1.SetDcMotorControlMode 1, DIRECT_POSITION

    WiRobotSDK1.EnableDcMotor 0

    WiRobotSDK1.EnableDcMotor 1

    cmd1 = leftWheelData + cWHOLE_RANGE / 15

    cmd2 = rightWheelData + cWHOLE_RANGE / 15

    ' ngubah cmd1, cmd2 ke valid data range

    If (cmd1 < 0) Then cmd1 = cmd1 + cFULL_COUNT

    If (cmd2 < 0) Then cmd2 = cmd2 + cFULL_COUNT

    If (cmd1 > cFULL_COUNT) Then cmd1 = cmd1 - cFULL_COUNT

    If (cmd2 > cFULL_COUNT) Then cmd2 = cmd2 - cFULL_COUNT

    WiRobotSDK1.DcMotorPositionTimeCtrAll CInt(cmd1), CInt(cmd2), NO_CTRL,
NO_CTRL, NO_CTRL, NO_CTRL, 1000

    Sleep 1000

End Sub

Private Sub btnWheelLeft_MouseDown(Button As Integer, Shift As Integer, X As
Single, Y As Single)

    tmrWheelLeft.Enabled = True

    tmrWheelLeft.Interval = 500

End Sub

Private Sub btnWheelLeft_MouseUp(Button As Integer, Shift As Integer, X As
Single, Y As Single)

    tmrWheelLeft.Enabled = False

    Call btnWheelStop_Click

End Sub

Private Sub btnWheelRight_Click()

Call kanan

End Sub
```

```vb
Sub kanan()

    Dim cmd1 As Long

    Dim cmd2 As Long


    'setting control mode

    WiRobotSDK1.SetDcMotorControlMode 0, DIRECT_POSITION

    WiRobotSDK1.SetDcMotorControlMode 1, DIRECT_POSITION

    WiRobotSDK1.EnableDcMotor 0

    WiRobotSDK1.EnableDcMotor 1

    cmd1 = leftWheelData - cWHOLE_RANGE / 5

    cmd2 = rightWheelData - cWHOLE_RANGE / 5

    ' ngubah cmd1, cmd2 ke valid data range

    If (cmd1 < 0) Then cmd1 = cmd1 + cFULL_COUNT

    If (cmd2 < 0) Then cmd2 = cmd2 + cFULL_COUNT

    If (cmd1 > cFULL_COUNT) Then cmd1 = cmd1 - cFULL_COUNT

    If (cmd2 > cFULL_COUNT) Then cmd2 = cmd2 - cFULL_COUNT

    WiRobotSDK1.DcMotorPositionTimeCtrAll CInt(cmd1), CInt(cmd2), NO_CTRL,
NO_CTRL, NO_CTRL, NO_CTRL, 1000

    Sleep 1000
End Sub
Sub Kanan1()
Dim cmd1 As Long

    Dim cmd2 As Long

    'setting control mode

    WiRobotSDK1.SetDcMotorControlMode 0, DIRECT_POSITION

    WiRobotSDK1.SetDcMotorControlMode 1, DIRECT_POSITION
```

```vb
    WiRobotSDK1.EnableDcMotor 0

    WiRobotSDK1.EnableDcMotor 1




    cmd1 = leftWheelData - cWHOLE_RANGE / 15

    cmd2 = rightWheelData - cWHOLE_RANGE / 15

    ' ngubah cmd1, cmd2 ke valid data range

    If (cmd1 < 0) Then cmd1 = cmd1 + cFULL_COUNT

    If (cmd2 < 0) Then cmd2 = cmd2 + cFULL_COUNT

    If (cmd1 > cFULL_COUNT) Then cmd1 = cmd1 - cFULL_COUNT

    If (cmd2 > cFULL_COUNT) Then cmd2 = cmd2 - cFULL_COUNT

    WiRobotSDK1.DcMotorPositionTimeCtrAll CInt(cmd1), CInt(cmd2), NO_CTRL,
NO_CTRL, NO_CTRL, NO_CTRL, 1000

    Sleep 1000
End Sub

Private Sub btnWheelRight_MouseDown(Button As Integer, Shift As Integer, X As
Single, Y As Single)

    tmrWheelRight.Enabled = True

    tmrWheelRight.Interval = 500
End Sub

Private Sub btnWheelRight_MouseUp(Button As Integer, Shift As Integer, X As
Single, Y As Single)

    tmrWheelRight.Enabled = False

    Call btnWheelStop_Click
End Sub
```

```
Private Sub btnWheelStop_Click()

Call henti

End Sub

Sub henti()

   WiRobotSDK1.SuspendDcMotor 0  ' stop  unit 0 control output

   WiRobotSDK1.SuspendDcMotor 1  ' stop  unit 1 control output

End Sub




Private Sub cmdExit_Click()

   End

End Sub

Private Sub Dir1_Change()

File1.Path = Dir1.Path

End Sub

Private Sub Drive1_Change()

Dir1.Path = Drive1.Drive

End Sub

Private Sub Form_MouseDown(Button As Integer, Shift As Integer, X As Single, Y
As Single)

If Button = vbLeftButton Then

Text2.Text = "kiri"

End If

End Sub

Private Sub manual_Click()

Timer2.Enabled = False
```

End Sub

Private Sub Timer2_Timer()

Call maju

If txtIRRanger(1).Text > 1700 Or txtIRRanger(7).Text > 1700 Then

Call kanan

ElseIf txtIRRanger(2).Text > 1700 Or txtIRRanger(3).Text > 1700 Then

WiRobotSDK1.PlayAudioFile "snap0031.WAV"

Call mundur

Call kanan

ElseIf txtIRRanger(5).Text > 1700 Or txtIRRanger(4).Text > 1700 Then

Call kiri

ElseIf txtIRRanger(6).Text > 1700 Then

Call maju

End If

End Sub

Private Sub Timer3_Timer()

'mendapatkan posisi (koordinat x,y)

Call GetCursorPos(pos)

'mendeteksi kursor

Call GetAsyncKeyState(VK_LMOUSEBUTTON)

'harus kecil dari 0 karena td uda dideklarasi kalau const VK_LMOUSEBUTTON = 1 (dibiner artinya terletak di least significant number, jadi kalau klik kiri yang least signifikan numbernya di set jd berubah lihat hal: 764 VB Bible

If GetAsyncKeyState(VK_LMOUSEBUTTON) < 0 Then

n = n + 1

```
End If

If n < 8 Then

Text1.Text = pos.X

ElseIf n > 8 And n < 16 Then

Text2.Text = pos.X

ElseIf n > 16 Then

Call Lebar

End If

End Sub


Private Sub Timer4_Timer()

'mendapatkan posisi (koordinat x,y)

Call GetCursorPos(pos)

'mendeteksi kursor

Call GetAsyncKeyState(VK_LMOUSEBUTTON)
```

'harus kecil dari 0 karena td uda dideklarasi kalau const VK_LMOUSEBUTTON = 1
(dibiner artinya terletak di least significant number, jadi kalau klik kiri yang least
signifikan numbernya di set jd berubah lihat hal: 764 VB Bible

```
If GetAsyncKeyState(VK_LMOUSEBUTTON) < 0 Then

n = n + 1

End If

If n < 8 Then

Text1.Text = pos.Y

ElseIf n > 8 And n < 16 Then

Text2.Text = pos.Y

ElseIf n > 16 Then
```

```
Call Panjang

End If

End Sub


Private Sub Timer5_Timer()

If (txtIRRanger(3).Text + 15) < (txtIRRanger(2).Text) Then

Call Kiri1

ElseIf (txtIRRanger(3).Text) > (txtIRRanger(2).Text + 15) Then

Call Kanan1

Else

Call Delay

End If

End Sub

Private Sub Timer6_Timer()

Call Photo

End Sub


Private Sub WiRobotSDK1_CustomSensorEvent()

  'nyari #2 ~ #7 IR Ranger

  txtIRRanger(2).Text = WiRobotSDK1.GetCustomAD3

  txtIRRanger(3).Text = WiRobotSDK1.GetCustomAD4

  txtIRRanger(4).Text = WiRobotSDK1.GetCustomAD5

  txtIRRanger(5).Text = WiRobotSDK1.GetCustomAD6

  txtIRRanger(6).Text = WiRobotSDK1.GetCustomAD7

  txtIRRanger(7).Text = WiRobotSDK1.GetCustomAD8
```

'nyari DSP Board VCC Nilai Sensor

txtDSPVccValue.Text = WiRobotSDK1.GetCustomAD1

'nyari Motor Vd Nilai Sensor

txtMotorVdValue.Text = WiRobotSDK1.GetCustomAD2

End Sub

Private Sub WiRobotSDK1_MotorSensorEvent()

leftWheelData = WiRobotSDK1.GetEncoderPulse1

rightWheelData = WiRobotSDK1.GetEncoderPulse2

End Sub

Private Sub WiRobotSDK1_standardSensorEvent()

txtUltrasonicValue(1).Text = WiRobotSDK1.GetSensorSonar1

txtUltrasonicValue(2).Text = WiRobotSDK1.GetSensorSonar2

txtUltrasonicValue(3).Text = WiRobotSDK1.GetSensorSonar3

txtIRRanger(1).Text = WiRobotSDK1.GetSensorIRRange

txtIRCodeValue(1).Text = WiRobotSDK1.GetSensorIRCode1

txtIRCodeValue(2).Text = WiRobotSDK1.GetSensorIRCode2

txtIRCodeValue(3).Text = WiRobotSDK1.GetSensorIRCode3

txtIRCodeValue(4).Text = WiRobotSDK1.GetSensorIRCode4

Private Sub Form_Load()

h1(1, 1) = -1: h1(1, 2) = 0: h1(1, 3) = 1

h1(2, 1) = -1: h1(2, 2) = 0: h1(2, 3) = 1

h1(3, 1) = -1: h1(3, 2) = 0: h1(3, 3) = 1

For i = 1 To 3

  For j = 1 To 3

    h2(i, j) = h1(j, i)

Next j

Next i

'timer btn automatic

Timer2.Enabled = False

'timer btn sesuaikan posisi photo

Timer5.Enabled = False

'timer delay untuk photo

 Timer6.Enabled = False

  'Set "Robot ID"

  WiRobotSDK1.connectRobot "drrobot1"

  'Set tab pertama untuk display

  mintCurFrame = 1

  fraControl(mintCurFrame).Visible = True

  'set the servo buttons disabled

  btnServoRun1.Enabled = False

  btnServoRun2.Enabled = False

  WiRobotSDK1.SuspendDcMotor 0 ' stop unit 0 control output

```
WiRobotSDK1.SuspendDcMotor 1  ' stop  unit 1 control output

WiRobotSDK1.SetMotorPolarity1 1

WiRobotSDK1.SetMotorPolarity2 1


WiRobotSDK1.SetDcMotorSensorUsage 0, DECODER_SENSOR ' Configure unit
0 using dual sensor

WiRobotSDK1.SetDcMotorSensorUsage 1, DECODER_SENSOR ' Configure unit
1 using dual sensor


WiRobotSDK1.SetDcMotorControlMode 0, DIRECT_VELOCITY

WiRobotSDK1.SetDcMotorControlMode 1, DIRECT_VELOCITY


WiRobotSDK1.SetDcMotorVelocityControlPID 0, 30, 10, 0

WiRobotSDK1.SetDcMotorVelocityControlPID 1, 30, 10, 0

WiRobotSDK1.SetDcMotorPositionControlPID 0, 600, 20, 5000

WiRobotSDK1.SetDcMotorPositionControlPID 1, 600, 20, 5000


Servo1Max = CInt(ReadIni("servo1", "max"))

Servo1Min = CInt(ReadIni("servo1", "min"))

Servo1Init = CInt(ReadIni("servo1", "init"))


Servo2Max = CInt(ReadIni("servo2", "max"))

Servo2Min = CInt(ReadIni("servo2", "min"))

Servo2Init = CInt(ReadIni("servo2", "init"))


txtServoValue1.Text = Servo1Init
```

```
txtServoValue2.Text = Servo2Init


lblServoRate1.Caption = "(* Range at " & Servo1Min & " ~ " & Servo1Max & ")"

lblServoRate2.Caption = "(* Range at " & Servo2Min & " ~ " & Servo2Max & ")"


'set Head Hor. Servo
' fixed position camera
hscHeadHor.Value = Servo2Init
hscHeadHor.Max = Servo2Max
hscHeadHor.Min = Servo2Min


'set Head Vert. Servo
'fixed position camera
hscHeadVert.Value = Servo1Init
hscHeadVert.Max = Servo1Max
hscHeadVert.Min = Servo1Min


'set Timer and Enable the streaming display
Timer1.Interval = 50 ms


If (firstTime = 0) Then

   firstTime = 1


   'Set LCD Display image
   txtLCDImageName.Text = App.Path & "\wirobot128.bmp"
```

```
WiRobotSDK1.LcdDisplayPMS App.Path & "\wirobot128.bmp"

    'Set init value untuk semua servos

    WiRobotSDK1.ServoTimeCtrAll    Servo1Init,    Servo2Init,    NO_CTRL,
NO_CTRL, NO_CTRL, NO_CTRL, 1000

End Sub

Private Sub Timer1_Timer()

  WiRobotSDK1.TakePhoto

End Sub

Private Sub tmrCOntrolServo_Timer()

  'Tetap panggil event button yg sama

  Select Case ControlButtonName

    Case Is = "btnHeadLeft_Click"

      Call btnHeadLeft_Click

    Case Is = "btnHeadRight_Click"

      Call btnHeadRight_Click

    Case Is = "btnHeadUp_Click"

      Call btnHeadUp_Click

    Case Is = "btnHeadDown_Click"

      Call btnHeadDown_Click

  End Select

End Sub


Private Sub tmrWheelLeft_Timer()

  Call btnWheelLeft_Click

End Sub

Private Sub tmrWheelRight_Timer()
```

```
    Call btnWheelRight_Click
End Sub


Sub IdentifyAxes(diDev As DirectInputDevice8)
  Dim didoEnum As DirectInputEnumDeviceObjects
  Dim dido As DirectInputDeviceObjectInstance
  Dim i As Integer


  For i = 1 To 8
    AxisPresent(i) = False
  Next



  Set didoEnum = diDev.GetDeviceObjectsEnum(DIDFT_AXIS)


  Dim sGuid As String
  For i = 1 To didoEnum.GetCount
     Set dido = didoEnum.GetItem(i)
     sGuid = dido.GetGuidType
     Select Case sGuid
       Case "GUID_XAxis"
         AxisPresent(1) = True
       Case "GUID_YAxis"
         AxisPresent(2) = True
       Case "GUID_ZAxis"
```

```vb
           AxisPresent(3) = True

        Case "GUID_RxAxis"

          AxisPresent(4) = True

        Case "GUID_RyAxis"

          AxisPresent(5) = True

        Case "GUID_RzAxis"

          AxisPresent(6) = True

        Case "GUID_Slider"

          AxisPresent(8) = True

          AxisPresent(7) = True

      End Select

   Next

End Sub


Public Sub DriveRobot()

    Dim leftwheel_PWM As Double

    Dim rightwheel_PWM As Double

    Steeringwheel2PWM leftwheel_PWM, rightwheel_PWM

   ' Left_ControlValue.Text = leftwheel_PWM

   ' Right_ControlValue.Text = rightwheel_PWM

    Call SendWheelDataToRobot(leftwheel_PWM, rightwheel_PWM)

End Sub


Private Sub SendWheelDataToRobot(LeftWheelValue, RightWheelValue)

   LeftWheelValue = LeftWheelValue * 16383 + 16383
```

```
RightWheelValue = RightWheelValue * 16383 + 16384

LeftWheelValue = IIf(LeftWheelValue > 32766, 32766, LeftWheelValue)

LeftWheelValue = IIf(LeftWheelValue < 0, 0, LeftWheelValue)

RightWheelValue = IIf(RightWheelValue > 32766, 32766, RightWheelValue)

RightWheelValue = IIf(RightWheelValue < 0, 0, RightWheelValue)

WiRobotSDK1.DcMotorPwmNonTimeCtr 0, CInt(LeftWheelValue)

WiRobotSDK1.DcMotorPwmNonTimeCtr 1, CInt(RightWheelValue)


End Sub



Public Sub Steeringwheel2PWM(leftwheel_PWM As Double, rightwheel_PWM As
Double)


    Dim X As Double

    Dim Y As Double

    Dim z As Double

    Dim PWMValue As Integer


    Const stallY As Double = 0.01

    Const FricL As Integer = 6000

    Const FricR As Integer = 6000

    Dim PowerR As Double

    Dim maxFric As Integer
```

```
If backwardLevel < 5500 Then 'forward drive

    Y = CDbl(10000 - gasLevel) / 10000

Else

    Y = -CDbl(10000 - gasLevel) / 10000

End If

z = CDbl(steeringLevel - 5000) / 5000 / (1 + gearLevel * 0.2)

'z = IIf(Abs(z) > 0.1, z, 0)

PowerR = gearLevel * 0.3333

maxFric = IIf(FricL > FricR, FricL, FricR)

PWMValue = Y * (16383 - maxFric)

If Abs(Y) > stallY And brakeLevel > 7000 And gearLevel <> 0 Then

    'If Abs(CDbl(steeringLevel - 5000) / 5000) < 0.95 Then

     If Y >= 0 Then 'forward

        If (z <= 0) Then

            rightwheel_PWM = 16383 + FricR + PWMValue * PowerR

            leftwheel_PWM = 16383 + Sgn(0.5 + 0.5 * (1 + 2 * z)) * FricL +
PWMValue * PowerR * (0.5 + 0.5 * (1 + 2 * z))

        Else

            rightwheel_PWM = 16383 + Sgn(0.5 + 0.5 * (1 - 2 * z)) * FricR +
PWMValue * PowerR * (0.5 + 0.5 * (1 - 2 * z))

            leftwheel_PWM = 16383 + FricL + PWMValue * PowerR

        End If

     Else 'backard

        If (z <= 0) Then

            leftwheel_PWM = 16383 - Sgn(0.5 + 0.5 * (1 + 2 * z)) * FricL +
PWMValue * PowerR * (0.5 + 0.5 * (1 + 2 * z))
```

```
PWMValue *

2 * z))

        End If

      ' End If

    End If

  Else

    rightwheel_PWM = 16383

    leftwheel_PWM = 16383

  End If

  leftwheel_PWM = -(leftwheel_PWM - 16383) / 16383

  rightwheel_PWM = (rightwheel_PWM - 16383) / 16383

  If Abs(leftwheel_PWM) > 1 Then

  Y = Y

  End If

End Sub

Private Sub automatic_Click()

Timer2.Enabled = True

End Sub


Private Sub photo_Click()

Timer5.Enabled = True

End Sub

Sub Delay()

Timer5.Enabled = False

Timer6.Enabled = True
```

```
End Sub

Sub Photo()

Timer6.Enabled = False

WiRobotSDK1.TakePhoto

WiRobotSDK1.SavePhotoAsBMP File1.filename

Picture1.Picture = LoadPicture(File1.filename)

Picture2.Picture = LoadPicture(File1.filename)

Call Edgedetection2

End Sub


Private Sub XPButton3_Click()

Call Edgedetection1

Call Edgedetection2

End Sub

Sub Edgedetection2()

Dim X(500, 500) As Integer

Dim n1 As Integer

Dim n2 As Integer

Dim z1 As Integer

Dim z2 As Integer

Dim z As Integer

Dim warna As Single

Dim r As Integer

Dim g As Integer

Dim b As Integer
```

```
Dim u1 As Integer

Dim u2 As Integer

'Convert RGB to Gray

n1 = 0

For i = 1 To Picture2.ScaleWidth Step 15

n1 = n1 + 1

n2 = 0

For j = 1 To Picture2.ScaleHeight Step 15

warna = Picture2.Point(i, j)

r = warna And RGB(255, 0, 0)

g = Int((warna And RGB(0, 255, 0)) / 256)

b = Int(Int((warna And RGB(0, 0, 255)) / 256) / 256)


n2 = n2 + 1

X(n1, n2) = Int((r + g + b) / 3)

Picture2.PSet (i, j), RGB(X(n1, n2), X(n1, n2), X(n1, n2))

Next j

Next i

'proses deteksi tepi

For i = 1 To n1

For j = 1 To n2

z1 = 0

z2 = 0

For u1 = -1 To 1

For u2 = -1 To 1
```

z1 = z1 + h1(u1 + 2, u2 + 2) * X(i + u1, j + u2)

z2 = z2 + h2(u1 + 2, u2 + 2) * X(i + u1, j + u2)

Next u2

Next u1

z = Int(Abs(z1 + z2))

If z > 255 Then z = 255

Picture2.PSet ((i - 1) * 15 + 1, (j - 1) * 15 + 1), RGB(z, z, z)

Next j

Next i

End Sub

Sub Edgedetection1()

Dim X(500, 500) As Integer

Dim n1 As Integer

Dim n2 As Integer

Dim z1 As Integer

Dim z2 As Integer

Dim z As Integer

Dim warna As Single

Dim r As Integer

Dim g As Integer

Dim b As Integer

Dim u1 As Integer

Dim u2 As Integer

'Convert RGB to Gray

n1 = 0

```
For i = 1 To Picture1.ScaleWidth Step 15

n1 = n1 + 1

n2 = 0

For j = 1 To Picture1.ScaleHeight Step 15

warna = Picture1.Point(i, j)

r = warna And RGB(255, 0, 0)

g = Int((warna And RGB(0, 255, 0)) / 256)

b = Int(Int((warna And RGB(0, 0, 255)) / 256) / 256)


n2 = n2 + 1

X(n1, n2) = Int((r + g + b) / 3)

Picture1.PSet (i, j), RGB(X(n1, n2), X(n1, n2), X(n1, n2))

Next j

Next i

'proses deteksi tepi

For i = 1 To n1

For j = 1 To n2

z1 = 0

z2 = 0

For u1 = -1 To 1

For u2 = -1 To 1

z1 = z1 + h1(u1 + 2, u2 + 2) * X(i + u1, j + u2)

z2 = z2 + h2(u1 + 2, u2 + 2) * X(i + u1, j + u2)

Next u2

Next u1
```

z =

(z, z, z)

Next j

Next i

End Sub

Private Sub Width Detection Mode_Click()

'timer cursor

n = 0

Timer4.Enabled = False

Timer3.Interval = 1

Timer3.Enabled = True

End Sub

Sub Lebar()

n = 0

l = Text2.Text

m = Text1.Text

o = l - m

X(2) = CDbl (txtUltrasonicValue(2).Text)

X(3) = ((o*47)/98)

X(1) = Math.Tan(Val(X(2)-60))+ X(3)

Text3.Text = X(1) & "cm"

End Sub

Private Sub Compare_Click()

```
Dim r As Integer

Dim b As Integer

Dim g As Integer

Dim r1 As Integer

Dim b1 As Integer

Dim g1 As Integer

warna4 = 0

Call Edgedetection2

Call Edgedetection1

For i = 1 To Picture2.Width Step 15

For j = 1 To Picture2.Height Step 15

warna2 = Picture2.Point(i, j)

r2 = warna2 And RGB(255, 0, 0)

g2

g = (g2 - g1)

b = (b2 - b1)

warna3 = (r + g + b) / 3

warna4 = warna4 + warna3

warna5 = (1 - warna4) * 100

Sub Panjang()

n = 0

l = Text2.Text

m = Text1.Text

o =

End Sub
```

WiRobot X80 USER MANUAL

WiFi 802.11 Wireless Mobile System

Version: 1.0.5

Jan. 2006

Table of Contents

Chapter I. WiRobot Getting Start Guide

I. Preface

I.1 Audience

This document is written for robot developers in using WiRobot systems. It provides the initial

product information as well as a guide in helping users to understand how to use this system. The

developers should have basic knowledge in Microsoft Visual C++ or VB. Detail programming

information can be found in Chapter III.

I.2 Feedback

If you find any problems in this document, please send us your feedback to support@drrobot.com.

II. WiRobot Overview

WiRobot$_{TM}$ is an integrated electronic and software robotic system extended from Dr Robot's

comprehensive humanoid robot, which has demonstrated its interactive capabilities in the public and

the media. Each WiRobot development system is designed to provide a user-friendly programming
environment for hobbyists, students in robotic areas and researchers to develop their robot programs
and applications at an affordable cost.
The power of WiRobot mobile robot system comes from the Dr Robot's Distributed Computation
Robotic Architecture and System (DIRAS) technology, which offloads most of the computation and
storage intensive tasks to a home PC. Through a digital wireless connection supporting over 100kbps
data communication rate, user programs running on PC are virtually connected directly to the
WiRobot development system. Data such as image, audio, sensor information, and etc. are available to
the user through a set of ActiveX control components (SDK) developed for MS VC++ and VB
programming environment. Multiple PC programs are also allowed to access the data information
obtained from the sensors simultaneously. High level schemes such as tele-operation, navigation,
reasoning, learning, recognition, and image processing routines are programmed and executed on the
PC remotely. Multi-robot coordination is also feasible for applications like soccer game. Microsoft
Visual Studio programming environment is chosen as the development platform due to its popularity
and ease-of-use than the non-user friendly embedded programming interface. Note that
communication protocol for WiRobot system is also available for developers/researchers who prefer
to use different platform or operating systems to communicate and control the WiRobot system. But
this document focuses on how to use the WiRobot system using the SDK under Microsoft platform.
As well, WiRobot system already comes with low-level drivers for all its electronic modules and can
provide a flexible way for users to control the robot. For instance, it allows user to control standard
servo motors and DC motors by using the built-in commands available in the control command library,
which offers several types of DC motor control method including open-loop PWM, closed-loop
position control, closed-loop velocity control, and closed-loop current control. Control parameters are
also configurable.

II.1 Software Components

The WiRobot system comes with a CD containing the following software components and documents:

. "WiRobot Gateway" which is used to connect the PC to the robot and show the connection
status,

. An ActiveX control, called WiRobot SDK ActiveX Module, with a set of APIs is provided for
user to access the robot when developing his/her own applications in MS VC++ or VB,

. Several PC sample applications with source code is provided to demonstrate the capabilities
of the WiRobot system, and

. WiRobot documents. (The latest documents can be found on www.drrobot.com)

In the WiRobot system, low level electronic drivers are pre-programmed and embedded in the
WiRobot controllers (PMS5005 and PMB5010). Data information such as image, audio, sensor
information, and etc. are available to the user via the WiRobot ActiveX control developed for MS VC++
and VB program environment or by using the WiRobot communication protocol. Using this ActiveX
control, user can also send various control commands to the robot. A general connection architecture
of the WiRobot system is shown as follows:

Figure II.1 WiRobot System Communication Architecture

Copyright © Dr Robot Inc. 2006 8

III. Software (WiRobot SDK) Installation

III.1 System Requirements

The PC requirements in using the WiRobot system are:

. PIII 550MHz or faster CPU

. 64 MB RAM or more

. 20 MB hard disk free space

. Microsoft Windows 2000 or XP operating system

As well as, Microsoft Visual VB or VC++ 6.0 (with Service Pack 5) is required for users to develop their
own applications.

III.2 WiRobot System Installation

Insert the WiRobot System CD into the CD ROM and the auto run menu will guide you through the
installation process.

Figure III.1 WiRobot Installation Step 1

Figure III.2 WiRobot Installation Step 2

Copyright © Dr Robot Inc. 2006 9

By default, all these components will be installed under the directory "C:\Program

Files\DrRobot\WiRobot-System" unless user specifies another location during the installation.

Figure III.3 WiRobot Installation Step 3

IV. Connecting to WiRobot System

IV.1 WiFi Wireless Connecting

IV.1.1 Configuration your 802.11b (or 802.11b compatible) wireless access point/router

Configuration your 802.11b (or 802.11b compatible) wireless access point/router with following

default settings:

SSID: dri

SSID BROADCAST: Enable

AUTHENTICATION: SHARED KEY

WEP 128 bit key: 112233445566778899AABBCCDD

Router IP: 192.168.0.200

If you prefer to change the router setting, such as SSID, router IP and/or key, you have to configure

the WiFi module (see Chapter IV, session IV. WFS802b WiFi 802.11 Serial Module with Antenna) on

the robot to match your changed settings.

Copyright © Dr Robot Inc. 2006 10

IV.1.2 Run the WiRobot Gateway

Figure IV.2 WiFi Gateway

Check the WiFi Connection and input the IP of the robot which you want to connect, and input the

Port with 10001, and then click the Connect button. You can find the robot IP under the robot.

IV.2 Serial Cable Connecting

The user can also connect the WiRobot system to a PC through a null modem cable (RS232 Crossover

Serial Cable) as follows:

Figure IV.3 WiRobot System Setup without Wireless Connection

Copyright © Dr Robot Inc. 2006 11

IV.2.1 Connecting the cable and module

. Connect Serial cable, make sure the serial cable is connected to the COM1 socket of your PC

at one end, and the other end should be connected to the RS232 interface module.

Figure IV.4 Connection of RS232 with Serial Cable

. Unplug the serial WIFI (or Bluetooth) wireless module which had already plugged in the lower

socket board PMB5010 of the robot.

Figure IV.5 Upper reach SCI0 of Lower Socket Board PMB5010

. Then plug the RS232 interface module in Upper Reach SCI0 on the PMB5010 which is the

lower socket board of the robot. (Picture available in the PMB5010 Multimedia Controller

User Manual, Page 94)

IV.2.2 Turn on the robot

Check the LED lights on the socket board, and find out if they are flashing on the socket board. There
should be 2 LED lights keep flashing fast on the upper board PMS5005 in the right rear corner of the
robot and 1 LED light keep flashing on the lower board in the right front corner of the robot. If these 3
LED lights are flashing, the robot is started completely.

IV.2.3 Run the WiRobot Gateway

. Select COM1 and Serial Cable

. Set "drrobot1" (default) as "Robot ID"

. Click the "Connect" button when you are sure that the robot is completely started.

. Wait 1 to 3 seconds, the WiRobot Gateway will minimize automatically when connected.

. If it is not connected, close the WiRobot Gateway and turn off the robot try it again 10
seconds later.

1.
RS232
2. 3.

Figure IV.6 GUI of WiRobot Gateway on PC

The "WiRobot Gateway" will provide connection status information between the PC and the robot.

This program is required to keep running as long as the user wants to access and control the robot
through the sample applications or their custom programs. Robot data acquisitions including human
sensor, ultrasonic sensor, Infrared distance sensor, tilting sensor, potentiometer, color CMOS image
sensor, microphone, and etc. and motion control can be done by making function calls offered by the
"WiRobot SDK ActiveX Module". Details of this ActiveX control component can be found in the
Chapter III. WiRobot SDK API Reference.

After the connection is established between the robot and the PC, user can start to use the WiRobot
system by running the sample applications offered in the WiRobot software package.

V. Building PC Applications Using SDK

This section will discuss how to program user's applications. Several sample applications with source
code are provided to help user kick start in using the WiRobot system. All these source code will be
stored in the "SampleApps" folder under the WiRobot SDK installation location.

V.1 Using WiRobot SDK Component ActiveX Control

When user starts to write an application, he/she first adds the WiRobot SDK Component in your VB or

VC++ project. The ActiveX object is installed during the installation process and the following is a step

to step guideline showing how to incorporate the ActiveX Control into a VB 6.0 project:

. Create a new VB project

13

. Click "Project" in the menu and choose the "Components"

Figure V.1 Using ActiveX module under VB Step 1

. Uncheck the "Selected Items Only" box to show all components, choose the "WiRobot SDK

ActiveX Module" and click "OK"

Figure V.2 Using ActiveX module under VB Step 2

. A new icon on the left menu bar will appear and user can simply drag and drop this icon to the

Project's Form and start using the APIs offered by this ActiveX control. By default the variable name of this component is "WiRobotSDK1".

14

Figure V.3 Using ActiveX module under VB Step 3

V.2 Sample Application 1 – WiRobot X80 Controller (VB)

This sample application demonstrates the basic capabilities of the WiRobot X80 using Microsoft VB

with source code provided. This program can read sensor data, obtain image and audio, play wave file,

and control the robot movement with command or joystick. The GUI of this program is shown as

follows:

15

Figure V.4 GUI of the WiRobot X80 Controller (VB)

User can click the button on the interface to control the robot as long as the "WiRobot Gateway" is

connected to the robot.

The following are some functions used in this sample application for controlling the servos mounted on

the head:
WiRobotSDK1.EnableServo 0
WiRobotSDK1.ServoTimeCtr 0, 3800, 1000

The WiRobotSDK1 is the WiRobot SDK Component ActiveX control. The first line will enable the channel

0 servo. The second line will control the servo to take 1000ms to the target position of 3800.

User can also obtain and control the multimedia information of the robot simply by calling the

following functions
WiRobotSDK1.PlayAudioFile sourceFileName

The function will play the wave file to the robot with the sound file stored in the "sourceFileName"

which is in .WAV format.

16

V.3 Sample Application 2 - WiRobot DRK8000 Controller (VB)

This sample application demonstrates the basic capabilities of the WiRobot DRK8000 using
Microsoft VB with source code provided. This program can read sensor data, obtain image and audio,
play wave file, set the LCD display image, and control the robot movement. The GUI of this program is
shown as follows:

Figure V.5 GUI of the WiRobot DRK8000 Controller (VB)

User can click the button on the interface to control the robot as long as the "WiRobot Gateway" is
connected to the robot.

The following are some functions used in this sample application for controlling the servos mounted on
the head:

```
WiRobotSDK1.EnableServo 0
WiRobotSDK1.ServoTimeCtr 0, 3800, 1000
```

The WiRobotSDK1 is the WiRobot SDK Component ActiveX control. The first line will enable the channel
0 servo. The second line will control the servo to take 1000ms to the target position of 3800.

User can also obtain and control the multimedia information of the robot simply by calling the
following functions

```
WiRobotSDK1.LcdDisplayPMS sourceFileName
WiRobotSDK1.TakePhoto
WiRobotSDK1.SavePhotoAsBMP destinationfileName
```

The first function will change the LCD display on the robot to the image stored in the "sourceFileName" which is in bitmap format. The second function will request the robot to take a
picture and an "ImageEvent" will be triggered when this image is ready for pickup. The third function
will save the image to the file with "destinationfileName" in bitmap format.

V.4 Sample Application 3- WiRobot DRK6000/8000 Controller (VC++)

The second sample application demonstrates how to program a VC++ application using the WiRobot
system. The GUI of this program interface is shown as follows:

17

Figure V.6 GUI of the WiRobot DRK6000/8000 Controller (VC++)

To obtain an image from the robot, user can call the following function where m_ctlSDK is a member
variable of the class CWiRobotSDK imported from the "WiRobot SDK ActiveX Module"

```
m_ctlSDK.TakePhoto ();
```

To control the robot to move forward continuously, user can call the following functions

```
m_ctlSDK.SetDcMotorControlMode (0, 0);
m_ctlSDK.SetDcMotorControlMode (1, 0);
m_ctlSDK.DcMotorPwmTimeCtrAll (32066, 32066, -32768, -32768, -32768, -32768, 800);
```

To stop the robot, the following commands can be used:

```
m_ctlSDK.SuspendDcMotor (0);
m_ctlSDK.SuspendDcMotor (1);
```

For details on how to control the robot using Microsoft Visual C++, please refer to the sample C++

source code and Chapter III. WiRobot SDK API (Page 26).

V.5 Other Sample Applications

More sample applications are available in the "SampleApps" folder for reference.

VI. Miscellaneous

VI.1 System Update

Dr Robot will provide software update for both the PC software as well as the DSP software to

enhance the existing features. User can visit www.DrRobot.com to check for any new updates

available for the existing system.

Chapter II. X80 System Specification

-X80 Wireless Mobile System

I. WiRobot X80 Overview

WiRobot is an integrated electronic and software robotic system extended from Dr Robot's

comprehensive humanoid robot. Each WiRobot development system is designed to provide a userfriendly

programming environment for hobbyists, students and researchers to develop their advanced

robot programs and applications at an affordable cost. The X80 development system includes the

respective mechanical structure, electronic modules as well as the software development kit (SDK).

The mechanical structure is already pre-built and the electronic system is setup with a Multimedia

Controller (PMB5010), a Sensing-Motion Controller (PMS5005) and various peripheral electronic

modules. The software component will be installed on a PC and is responsible to establish a wireless

connection and exchange data with the robot. User can develop their own applications in VC++ or VB

using the APIs offered in "WiRobot SDK ActiveX Module" which accesses the sensor information,

sends control command and configures the system setting.

This ready to use mobile robot platform is designed for researchers developing advanced robot

applications such as remote monitoring, telepresence and autonomous navigation/patrol.

Mechanics

The X80 is the result of extensive efforts to develop a robot that would be fast and strong,

while itself remaining lightweight and nimble. The wheel-based platform's two 12V DC motors

each supply 300 oz.-inches of torque to the X80's 18 cm (7 in.) wheels, yielding a top speed

in excess of 1 m/s (3.3 ft/s). Two high-resolution (1200 count per wheel cycle) quadrature

encoders mounted on each wheel provide high-precision measurement and control of wheel

movement. Weighing only 3.5 kg (7.7 lb.), the system is light, but it can carry an additional

payload of 10 kg (22 lb.).

Sensors

X80 offers full WiFi (802.11b) wireless, multimedia, sensing and motion capabilities and

comes with a wide range of sensor, camera, and audio modules, sufficient to serve in any

variety of applications. The X80 offers broad expandability as well for projects that may

require additional sensors, even specialized modules. Powered by separate RC servo motors,

the integrated camera head can pan and tilt independently.

Architecture

The X80's underlying technology evolved from Dr Robot's Distributed Computation Robotic

Architecture, originally developed for Dr Robot's Humanoid (HR) Robot. Using this approach,

high-level control of the robot is maintained by a remote or local PC/server communicating by

a secure wireless link. Low-level functionality is managed by an onboard digital signal

processor (DSP) while computationally intensive operations are performed offboard. The

result is a robot that's lighter, draws less power, runs longer and is dramatically less expensive

than a fully bundled or self-contained system. Moreover, since primary processing resides in a

server, any hardware upgrades to the central unit are shared by all the robots it controls.

With its integrated high bandwidth (11Mbps) WiFi 802.11 wireless module, the system can

upload all sensor data (including encoder sensor readings) to a PC or server at rates in excess

of 10Hz. Similarly, streaming audio (8Hz x 8bits) and video (up to 4 fps) either for direct

monitoring or for processing by high-level AI schemes is a snap. Commands and instructions

sent to the X80 via the same wireless link also pass at rates exceeding 10Hz, providing realtime

control and access.

20

The X80 includes all WiRobot development software components (for MS Windows 2000

and up), enabling easy access to all data and information in a standard Microsoft Windows

programming environment (e.g., MS VB and VC++). Under the approach of using a separate

PC for high-level control, there are no longer onboard restrictions on a mobile system's

processing power, memory and storage.

With the X80 system, researchers can develop a specialized intelligent robotic assistant,

security robot or simply use it as a platform for a variety of projects built around applications

such as human-machine interaction, mobile system navigation, robot behavior, image

processing, object recognition, voice recognition, teleoperation, remote sensing, map building

and localization etc.

The X80 system is fully integrated and each robot is fully assembled and tested prior to

shipping so that it arrives ready for use.

Mechanical and Control Highlights

- Two 12V motors with over 300oz.-inch torque each
- 7 inch driving wheel
- Max speed of 1 m/sec
- Dimensions:
  o 38.0 cm (15.0 inch) diameter
  o 25.5 cm (10.0 inch) height
- Weight: 3.5 kg
- Large top mounting deck for additional devices such as a notebook computer
- Additional carrying payload: 10 kg
- Pre-programmed fine speed and position control achieved by an integrated PMS5005

module employing two 1200 count per wheel-cycle quadrature encoders.

Electronic System Highlights

- Fully integrated WiFi (802.11b) system with dual serial communication channels (max of

912.6 Kbps per channel), supporting both UDP and TCP/IP protocol.

- Full color video and two-way audio capability. (CMOS color image module and audio module

are fully integrated.)

- Battery: 3700mAh with over 3 hours for nominal operation.
- Collision detection sensors include 3 sonar range sensors and 7 IR range sensors
- Two pyroelectric sensors for human motion detection

. Additional sensors such as supplementary sonar sensors, temperature sensors, acceleration /

tilting sensor, or customized sensors can be added.

21

Standard Electronics components and Operation Detail

WiRobot X80 Specifications:

Table I.1 WiRobot X80 Specifications

On-Board CPUs TI 120MIPS & Motorola 40MIPS 16-bit fix-point DSP

On-Board Storage 1M x 16-bit words flash, Up to 256K x 16-bit words SRAM

Degree of Freedom 2 x wheel motion, 2 x camera motion (Pan + Tilt) , up to 6 servos and 3 DC motors)

Built-in Peripheral Interface
and Modules

8-bit CIF (352 x 288) Color CMOS Camera Module

Audio codec and amplifier module with mic. and speakers

WiFI 802.11b wireless module

General-purpose PWM DC motor control

DC motors (up to 3)

Servo motors (up to 6)

Quadrature Encoder Inputs

Ultrasonic sensor modules (up to 6)

Human sensor module (up to 2)

Infrared range sensor input

Digital inputs

Digital outputs

X 1

X 1

X 1

X 6

X 2

X 2

X 2

X 3

X 2

X 7

X 8

X 8

Wireless Operation Range >25 meter indoor

>100 meter line of sight

Power Supply 7.2V Ni-MH 3700mAh

Smart fast charger

X 1

X 1

Operation Time Nominal usage with 3700mAh battery > 3hr

Maximum Moving Speed Approx. 1 meter per second

Additional/Optional
Peripheral Modules

Ambient temperature sensor modules

Tilt/acceleration sensor modules

DC motor driver module (3 DC motors)

Servo

Ultrasonic sensor module
128 x 64 graphic LCD display module
Potentiometer position feedback sensor (up to 6)
Full duplex infrared remote control and communication interface
X 1
X 1
X 1
X 4
X 1
X 1
X 6
X 1

22

The standard WiRobot DRK series system electronic modules:

Table I.2 Standard WiRobot DRK Series System Electronic Modules

Part Number Name X80
PMS5005 Robot Sensing and Motion Controller 1
PMS5010 Multimedia Controller 1
MDM5253 DC Motor Driver Module with Position and Current
Feedback
1
MCI3908 Color Image Module With Camera 1
DUR5200 Ultrasonic Range Sensor Module 3
DHM5150 Pyroelectric Human Motion Sensor Module 2
GP2Y0A21YK Sharp IR Distance Measuring Sensor Module 7
MCR3210 RS232 Interface Module 1
WFS802b WiFi802.11b Wireless Serial Module 1
BAS8100 8Ohm 1W Speaker 1
MAC5310 Audio Codec and Audio Power Amplifier Module 1
SAM5247 Uni-directional Electret Microphone 1
CCR2150 RS232 Cross-over Serial Cable 1
N/A Servo 2
N/A 12V DC Motor 2
BPN7240 7.2V Ni-MH 3700mAh Battery Pack 1
2300333 7.2V/9.6V NI-CD/NI-MH R/C PACK CHARGER 1

23

I.1 Mechanical Specification

The following diagram illustrates the mechanical structure of the WiRobot X80
system:

Figure I.1 WiRobot X80

I.2 Electrical

I.2.1 Power

The X80 is powered by a single 7.2V battery pack. This battery pack is connected to
both PMS5005
and PMB5010 through a switch. User can turn on or turn off the system (both
PMS5005 and
PMB5010) by pressing the switch next to the head base.

I.2.2 Communication

In the X80 system, PMS5005 and PMB5010 are connected together between
PMB5010's Lower

Reach SCI1 and PMS5005's Upper reach SCI0. A wireless module is placed on PMB5010's Upper
Reach SCI0 in order to communicate with a PC.

24

I.2.3 Electrical Modules
In this system, all electrical modules are located and connected as followed:
Table I.3 Electrical Modules Located and Connection

Electrical Module X80 Location / Setting

Ultrasonic #1 ○A Left front

Ultrasonic #2 ○D Middle front

Ultrasonic #3 ○G Right front

Human Sensor #1 ○H Left front, upper lever

Human Sensor #2 ○I Right front, upper lever

Infrared Range Sensor #1 ○B Front

Infrared Range Sensor #2 ○C Front

Infrared Range Sensor #3 ○E Front

Infrared Range Sensor #4 ○F Front

Infrared Range Sensor #5 ○K Right side

Infrared Range Sensor #6 ○L Rear

Infrared Range Sensor #7 ○J Left side

Servo #1 ○P To control the left/right movement of the neck (use channel 1)

Servo #2 ○O To control the up/down movement of the neck (use channel 2)

DC Motor #1 with quadrature encoder ○M Left , use channel 1

DC Motor #2 with quadrature encoder ○N Right, use channel 2

Camera ○Q Middle front

Speaker ○S Middle front, under the camera

Microphone ○R Beside the speaker

Figure I.2 Electrical Modules Located

25

Please refer to Chapter IV.I PMS5005 (Page 67) for details on how to connect different sensors, DC
motors, servos, and LCD display to the system. For camera, speaker and microphone, please refer to
Chapter IV.II PMB5010 (Page 82).

I.3 Other Specification
Table I.4 Other Specification

X80
Weight (including one battery pack) ~3.5kg
Recommended Maximum payload ~10kg

II. Miscellaneous

II.1 Battery Recharging

User can simply take out the battery at the lowest deck of the robot to recharge. It will normally take
about 20 hours to fully recharge the 3700mAh battery if slow charging is chosen. Fast charge would
take about 1-2 hours.

II.2 Sensor Location

User can change the sensor mounted on the robot to different location to suit his/her needs. As well,
user can add new sensors to the systems by making use of the available I/Os on the Sensing and
Motion Controller (PMS5005). Driver for these I/Os have been pre-programmed, data will be sent to
the PC for processing.

II.3 Known Issues

. When the power level is low, the robot's electrical system will become unstable. User has to
monitor the power level and recharge the battery when it is low.

. The initialization of the robot (when powering on) will take about 3-10 seconds.

. Please make sure that the robot finished its initialization stage before WiRobot Gateway
software (on PC) starts to connect to the robot. This may lead to failure connection between
PC and the robot

Chapter III. WiRobot SDK Application Programming Interface (API) (For MS Windows)

I. Convention

Data Type
int: 16 bit signed interger
UWord16: 16 bit unsigned interger
short: 16 bit signed interger
Syntax
Syntax under each API reference is based on the C/C++ calling convention. Corresponding Visual
Basic calling convention can be found in relevant VB reference book, or from the WiRobot VB code
examples.

II. WiRobot SDK Overview

WiRobot Software Development Kit (SDK) is a part of the WiRobot development system. Being a PCbased
software framework for robotic system development, the SDK contains the facilities for
memory management, system communication and user interface, and the utilities for audio, video
input/output, sensor data acquisition and motion control. Please refer to the Chapter IV.I PMS5005,
Chapter IV.II PMB5010, or Chapter II X80 for the detailed information on the SDK architecture,
organization and system programming.
Under the WiRobot system architecture, all the controllers are connected in a chain. Programs
developed using WiRobot SDK runs on the Host as the central controller of each chain. All the
embedded controllers have at least two SCI ports for the system communications: upper-reach port
and lower-reach port, with the direction respect to the central controller. The WiRobot system
controller-level connection architecture is shown as Figure II.1.
Figure II.1 WiRobot System Architecture
The APIs described in this manual are the interface between the application-level software and the
WiRobot hardware system. Programs developed using WiRobot SDK runs on the PC sending and
receiving data to and from the WiRobot hardware via wire or wireless connection. The firmware on the
embedded controllers take care of all the low level operations of the system functional modules, such
as data acquisition, fast-loop low level motion control, image and audio capture and compression,
audio playback and wireless communication. They are transparent to the high level software system
running on the central PC controller. All the system software development can be carried on solely
under user-friendly PC system. WiRobot SDK for Windows is available for MS Visual C++ and MS
Visual Basic environment.
API exists as a MS ActiveX component, called "WiRobot SDK ActiveX Module". User program uses
this component in VB or VC++ program to communicate with the WiRobot PMS5005 or/and

29

PMB5010 controllers. Data in between WiRobot hardware and the "WiRobot SDK ActiveX Module"
is managed and transferred by the supplied WiRobot Gateway Program (WiRobotGateway.exe) with
the shared memory as shown in Figure II.2.

Figure II.2 WiRobot Software Architecture

III. WiRobot SDK API Reference for PMS5005

WiRobot SDK APIs for PMS5005 are grouped under the categories of Sensor Peripherals, Motion
Control, Multimedia Control and Events.

III.1 Sensor Peripherals

This section contains the APIs for the operations of different sensor peripherals.
III.1.1 Batch Sensor Data Updating API
Standard Sensors: Sonar, human, infrared range, tilt/acceleration, temperature, battery voltage and
infrared remote control receiver
Motor Sensors: Potentiometers, current feedback sensors and encoders.
Custom Sensors: Custom expansion A/D inputs and digital inputs.
1 void SystemMotorSensorRequest(int PacketNumber);
2 void SystemStandardSensorRequest(int PacketNumber);
3 void SystemCustomSensorRequest(int PacketNumber);
4 void SystemAllSensorRequest(int PacketNumber);
Description:
SystemMotorSensorRequest sends a request command to the WiRobot Sensing and
Motion Controller (PMS5005) in order to get the sensor data related to motor control.
SystemStandardSensorRequest sends a request command to the WiRobot Sensing and
Motion Controller (PMS5005) in order to get all the WiRobot standard sensor data.
SystemCustomSensorRequest sends a request command to the WiRobot Sensing and
Motion Controller (PMS5005) in order to get all custom-sensor data,
SystemAllSensorRequest sends a request command to the WiRobot Sensing and Motion
Controller (PMS5005) in order to get all the sensor data.
Syntax: SystemMotorSensorRequest (PacketNumber); // motor related sensors
SystemStandardSensorRequest (PacketNumber); // standard sensors
SystemCustomSensorRequest (PacketNumber); // custom sensors
SystemAllSensorRequest (PacketNumber); // all the sensors
Parameter: short PacketNumber;
The meanings of PacketNumber as follows:
Table III.1 Meanings of PacketNumber
Parameter Action Requested
PacketNumber = 0 Stop sending the sensor data packets
PacketNumber = -1
Send sensor data packet continuously until being asked to
stop
PacketNumber > 0 Send n = PacketNumber packet(s) of sensor data and then

stop sending
Return value: void
Remarks:
(1) The default update rate is 20Hz. User can set up the data refresh rate according

to real system requirements.

(2) System is default to continuously sending all data when bootup.

See Also: SetSysMotorSensorPeriod, SetSysStandardSensorPeriod, SetSysCustomSensorPeriod, SetSysAllSensorPeriod.

5 void EnableMotorSensorSending ();

6 void EnableStandardSensorSending ();

7 void EnableCustomSensorSending ();

8 void EnableAllSensorSending ();

Description:

EnableMotorSensorSending enables batch updating motor-related sensor packets.

EnableStandardSensorSending enables batch updating standard sensor packets.

EnableCustomSensorSending enables batch updating custom sensor packets.

EnableAllSensorSending enables batch updating all the sensor packets.

Syntax: EnableMotorSensorSending(); // motor related sensors

EnableStandardSensorSending (); // standard sensors

EnableCustomSensorSending (); // custom sensors

EnableAllSensorSending (); // all the sensors

Parameter: void

Return value: void

Remarks:

1. The latest request setting of the packet number and the update rate are used.

2. By default, "all sensor data sending" is enabled.

3. Please refer to the remarks under SystemMotorSensorRequest, SystemSatndardSensorRequest, SystemCustomSensorRequest, SystemAllSensorRequest

9 void DisableMotorSensorSending ();

32

10 void DisableStandardSensorSending ();

11 void DisableCustomSensorSending ();

12 void DisableAllSensorSending ();

Description:

DisableMotorSensorSending disables batch updating motor-related sensor packets.

DisableStandardSensorSending disables batch updating standard sensor packets.

DisableCustomSensorSending disables batch updating custom sensor packets.

DisableAllSensorSending disables batch updating all the sensor packets.

Syntax: DisableMotorSensorSending(); // motor related sensors

DisableStandardSensorSending (); // standard sensors

DisableCustomSensorSending (); // custom sensors

DisableAllSensorSending (); // all the sensors

Parameter: void

Return value: void

See Alao: SystemMotorSensorRequest, SystemStandardSensorRequest, SystemCustomSensorRequest, SystemAllSensorRequest.

13 void SetSysMotorSensorPeriod(short PeriodTime) ;

14 void SetSysStandardSensorPeriod(short PeriodTime);

15 void SetSysCustomSensorPeriod(short PeriodTime) ;

16 void SetSysAllSensorPeriod(short PeriodTime) ;

Description:

SetSysMotorSensorPeriod sets refresh rate of batch updating motor-related sensor packets.
SetSysStandardSensorPeriod sets refresh rate of batch updating standard sensor packets.
SetSysCustomSensorPeriod sets refresh rate of batch updating custom sensor packets.
SetSysAllSensorPeriod sets refresh rate of batch updating all the sensor packets.
Syntax: SetSysMotorSensorPeriod (); // motor related sensors
SetSysStandardSensorPeriod (); // standard sensors
SetSysCustomSensorPeriod (); // custom sensors
SetSysAllSensorPeriod (); // all the sensors
Parameter: short PeriodTime; /* Update period (in ms) for batch sensing

packets to PC central controller */
Return value: void
Remarks:
The default PeriodTime = 50 (ms), i.e. update rate is 20Hz. PeriodTime should be bigger
than 50 (ms), i.e. the system data fastest refresh rate is 20Hz.
See Also: SystemMotorSensorRequest, SystemStandardSensorRequest,
SystemCustomSensorRequest, SystemAllSensorRequest.
III.1.2 Range and Distance Sensors
17 short GetSensorSonar1 ();
18 short GetSensorSonar2 ();
19 short GetSensorSonar3 ();
20 short GetSensorSonar4 ();
21 short GetSensorSonar5 ();
22 short GetSensorSonar6 ();
23 short GetSensorSonar (short channel);
Description:
GetSonarSensorX returns the current distance value between the relevant ultrasonic range sensor module (DUR5200) and the object in front of it. The unit is cm.
Syntax: ival = GetSensorSonar1 (); // Sonar #1
ival = GetSensorSonar2 (); // Sonar #2
ival = GetSensorSonar3 (); // Sonar #3
ival = GetSensorSonar4 (); // Sonar #4
ival = GetSensorSonar5 (); // Sonar #5
ival = GetSensorSonar6 (); // Sonar #6
ival = GetSensorSonar (short channel); // Sonar #1, 2, 3, 4, 5, or 6
Parameter: void
short channel; // 0, 1, 2, 3, 4, or 5 for Sonar #1, 2, 3, 4, 5, 6
Return value: short ival;
Return data interpretation:
Table III.2 Meanings of PacketNumber
Return Value Distance to Object
4 0 to 4 cm
4 to 254 4 to 254 cm

255 255 cm or longer
24 short GetSensorIRRange ();
Description:
GetSensorIRRange returns the current distance measurement value between the infrared
range sensor and the object in front of it.
Syntax: ival = GetSensorIRRange ();
Parameter: void
Return value: short ival;
Return data interpretation when using Sharp GP2Y0A21YK:
Table III.3 Return data interpretation
Return Value Distance to Object
<=585 80 cm or longer
585 to 3446 80 to 8 cm
>=3446 0 to 8 cm
Remarks:
The relationship between the return data and the distance is not linear. Please refer to the
sensor's datasheet for distance-voltage curve. The data returned is the raw data of the
analog to digital converter. The output voltage of the sensor can be calculated from the
following equation:
Sensor output voltage = (ival) * 3.0 / 4095 (V)
(e.g. Sharp GP2Y0A21YK
"http://sharp-world.com/products/device/lineup/data/pdf/datasheet/gp2y0a_d_e.pdf")

III.1.3 Human Sensors
25 short GetSensorHumanAlarm1 ();
26 short GetSensorHumanAlarm2 ();
Description:
GetSensorHumanAlarm returns the current human alarm data from DHM5150 Human
Motion Sensor Module. Please refer to the Chapter IV.VIII DHM5150 (Page 106) for detailed information.

35

Syntax: ival = GetSensorHumanAlarm1(); //1$_{st}$ human alarm
ival = GetSensorHumanAlarm2 (); // 2$_{nd}$ human alarm
Parameter: void
Return value: short ival;
Return data interpretation:
The return data is the raw value of the analog to digital converter indicating the amplified (x
5 times) output voltage of the sensor device. The data range is between 0 and 4095. When
there is no human present, the module output voltage is about 1.5 V and return value is
about 2047.
Remarks:

To detect human presence, the application should compare the difference of two samples
(to detect the change from "absence" to "presence"), and also compare the sample data to a
user defined threshold (to determine whether to report an alarm or not). The threshold
determines the sensitivity of sensor. The higher the threshold is the lower the sensitivity
will be.
27 short GetSensorHumanMotion1 ();
28 short GetSensorHumanMotion2 ();
Description:
GetSensorHumanMotion returns the current human motion value from DHM5150 Human
Motion Sensor Module. Please refer to the Chapter IV.VIII DHM5150 (Page 106) for detailed information.
Syntax: ival = GetSensorHumanMotion1 (); // Human direction data #1
ival = GetSensorHumanMotion2 (); // Human direction data #2
Parameter: void
Return value: short ival;
Return data interpretation:
The return data is the un-amplified raw value of the analog to digital converter indicating
the output voltage of the sensor device. The data range is between 0 and 4095.
Remarks:
To detect human motion direction, the application should compare the difference of two
samples of each sensor module's output (to detect the change from "absence" to "presence"), and then compare the sample data of the two sensor modules. For a single
source of human motion, the different patterns of the two sensor modules manifest the
directions of the motion. The relationship can be obtained from the experiments.
III.1.4 Tilt and Acceleration Sensor

29 short GetSensorTiltingX ();
30 short GetSensorTiltingY ();
Description:
GetSensorTiltingX, GetSensorTiltingY, return the current tilt angle values in the relevant
directions from DTA5102 Tilting and Acceleration Sensor Module.
Syntax: ival = GetSensorTiltingX (); // X direction
ival = GetSensorTiltingY (); // Y direction
Parameter: void
Return value: short ival;
Return data interpretation:
Tilting Angle = ArcSin ((ival- ZeroGValue) / abs(90DegreeGValue-ZeroGValue));
Remarks:

Where 90DegreeGValue and ZeroGValue are module-specific values that can be measured
by experiment:
1. Place the sensor level, so that the gravity vector is perpendicular to the measured sensor axis
2. Take the measurement and this value would be the ZeroGValue
3. Rotate the sensor so that the gravity vector is parallel with the measured axis
4. Take the measurement and this value would be the 90DegreeGValue
5. Repeat this step for the other direction
Typical value of ZeroGValue is about 2048 and abs(90DegreeGValue-ZeroGValue) is
about 1250.
III.1.5 Temperature Sensors
31 short GetSensorOverheatAD1 ();
32 short GetSensorOverheatAD2 ();
Description:
GetSensorOverheatADX returns the current air temperature values near the relevant DC
motor drive modules (MDM5253), which could be used for monitoring whether the motor
drivers are overheating or not. This situation usually occurs if the motor currents are kept
high (but still lower than the current limit of the motor driver module) for significant amount
of time, which may result from some unfavorable inner or external system conditions and is
not recommended for regular system operations.
Syntax: ival = GetSensorOverheatAD1(); //1$_{st}$ overheating sensor

37
ival = GetSensorOverheatAD2(); //2$_{nd}$ overheating sensor
Parameter: void
Return value: short ival;
Return data interpretation:
The return data is the raw value of the analog to digital converter indicating the output
voltage of the sensor. The data range of the return value is between 0 and 4095. The output voltage of the sensor can be calculated from the following equation:
Temperature (˚C) = 100- (ival – 980) / 11.6
33 short GetSensorTemperature ();
Description:
GetSensorTemperature returns the current temperature value from DAT5280 Ambient
Temperature Sensor Module.
Syntax: ival = GetSensorTemperature ();
Parameter: void
Return value: short ival;
Return data interpretation:
Temperature (˚C) = (ival – 1256) / 34.8

III.1.6 Infrared Remote Control Handling

34 short GetSensorIRCode1();

35 short GetSensorIRCode2();

36 short GetSensorIRCode3();

37 short GetSensorIRCode4();

Description:

GetSensorIRCodeX returns the four parts of a two-16-bit-code infrared remote control

command captured by the Sensing and Motion Controller (PMS5005) through the Infrared

Remote Controller Module (MIR5500).

Syntax: ival = GetSensorIRCode1 (); // the first code

ival = GetSensorIRCode2 (); // the second code

ival = GetSensorIRCode3 (); // the third code

ival = GetSensorIRCode4 (); // the fourth code

Parameter: void

Return value: short ival

Return data interpretation:

The recovered infrared remote control command (4 bytes code) is as follows:

Key Code: [the third byte] [the second byte] [the first byte]

Repeat Code: [the fourth byte]

where the repeat code would be 255 if button is pressed continuously.

38 void SetInfraredControlOutput (UWord16 LowWord, UWord16 HighWord);

Description:

SetInfraredControlOutput sends two 16-bit words infrared communication output data to

the Sensing and Motion Controller (PMS5005). The PMS5005 will then send the data out

through the infrared Remote Controller Module (MIR5500). In the case of being used for

infrared remote control, the output data serves as the remote control command.

Syntax: SetInfraredControlOutput (LowWord, HighWord);

Parameter: UWord16 LowWord; // 1st word

UWord16 HighWord; // 2nd word

Return value: void

Remarks:

1. In infrared communication application, the data format and the interpretation can be defined by the user at the application level.

2. In infrared remote control application, the control command should be compatible to the device to which the command is sent.

3. This API function is under development and will be available shortly.

III.1.7 Battery Voltage Monitors

39 short GetSensorBatteryAD1 ();

40 short GetSensorBatteryAD2 ();

41 short GetSensorBatteryAD3 ();

Description:

GetSensorBatteryADX returns the current value of the relevant power supply voltage if

the battery voltage monitor is enabled (default), or returns the relevant custom A/D inputs,

if the custom A/D input is enabled which is configured by the jumpers on PMS5005. Please

refer to Chapter IV.I PMS5005 Robot Sensing and Motion Controller (Page 67) for detailed information on hardware setting.

Syntax: ival = GetSensorBatteryAD1(); /* for battery of DSP circuits,

or custom A/D channel #1 */

ival = GetSensorBatteryAD2(); /* for battery of DC motors,

or custom A/D channel #2 */

ival = GetSensorBatteryAD3(); /* battery for servo motors,

or custom A/D channel #3 */

Parameter: void

Return value: short ival;

Return data interpretation:

The return data is the raw value of the analog to digital converter indicating the output

voltage of the monitor. The data range is between 0 and 4095.

When monitoring the voltage of the power supply, following equations can be used to calculate the real voltage values.

(1) Power supply voltage of DSP circuits = (ival / 4095) * 9 (V)

(2) Power supply voltage of DC motors = (ival / 4095) * 24 (V)

(3) Power supply voltage of servo motors = (ival / 4095) * 9 (V)

42 short GetSensorRefVoltage ();

43 short GetSensorPotVoltage ();

Description:

GetSensorRefVoltage returns the current value of the reference voltage of the A/D converter of the controller DSP.

GetSensorPotVoltage returns the current value of the power supply voltage of the potentiometer position sensors.

Syntax: ival = GetSensorRefVoltage ();

ival = GetSensorPotVoltage ();

Parameter: void

Return value: short ival;

Return data interpretation:

40

The return data is the raw value of the analog to digital converter indicating the output

voltage of the monitor. The data range is between 0 and 4095. The following equation can

be used to calculate the real voltage values.

Voltage = (ival / 4095) * 6 (V)

III.1.8 Potentiometer Position Sensors

44 short GetSensorPot1 ();

45 short GetSensorPot2 ();

46 short GetSensorPot3 ();

47 short GetSensorPot4 ();
48 short GetSensorPot5 ();
49 short GetSensorPot6 ();
50 short GetSensorPot (short channel);
Description:
GetSensorPotX returns the current value of the relevant potentiometer position sensors.
GetSensorPot (short channel) returns the current value of the specified potentiometer
position sensor.
Syntax: ival = GetSensorPot1 (); // Potentiometer sensor #1
ival = GetSensorPot2 (); // Potentiometer sensor #2
ival = GetSensorPot3 (); // Potentiometer sensor #3
ival = GetSensorPot4 (); // Potentiometer sensor #4
ival = GetSensorPot5 (); // Potentiometer sensor #5
ival = GetSensorPot6 (); // Potentiometer sensor #6
ival = GetSensorPot (channel); /* Potentiometer sensor
#1, 2, 3, 4, 5, or 6 */
Parameter: void // for GetSensorPotX
short channel; /* 0, 1, 2, 3, 4, or 5 for Potentiometer #
1, 2, 3, 4, 5, 6 */
Return value: short ival;
Return data interpretation and Remark:
1. The return data is the raw value of the analog to digital converter indicating the output
voltage of the sensor. The data range is between 0 and 4095. The angular position can
be calculated as follows, with the 180˚ position defined at sensor's physical middle
position. Single sensor or dual sensor can be used for rotation measurement.

Copyright © Dr Robot Inc. 2006 41

2. Single sensor is mainly used for the control of robot joint with limited rotation range.
The effective mechanical rotation range is 14˚ to 346˚, corresponding to the effective
electrical rotation range 0˚ to 332˚.
Angle position (˚) = (ival - 2048)/4095*333 + 180
3. Dual-sensor configuration is mainly used for continuous rotating joint control (such as
wheels). The effective rotation range is 0˚ to 360˚. Dual sensor configuration is only
available for channel 0 and 1. By connecting two potentiometers to potentiometer
channel 0 and channel 5, and specify the sensor type with command
SetDCMotorSensorUsage() to "Dual potentiometer sensor", the channel 0 reading
will combine these two sensor readings into 0° to 360° range. For channel 1, you
should connect the two potentiometers to channel 1 and 4.
Angle position (˚) = (ival - 2214)/2214*180 + 180
See also: SetDcMotorSensorUsage().
III.1.9 Motor Current Sensors
51 short GetMotorCurrent1 ();
52 short GetMotorCurrent2 ();

53 short GetMotorCurrent3 ();
54 short GetMotorCurrent4 ();
55 short GetMotorCurrent5 ();
56 short GetMotorCurrent6 ();
57 short GetMotorCurrent (short channel);
Description:
GetMotorCurrentX returns the sampling value of motor current sensor.
Syntax: ival = GetMotorCurrent1 (); // Current sensor #1
ival = GetMotorCurrent2 (); // Current sensor #2
ival = GetMotorCurrent3 (); // Current sensor #3
ival = GetMotorCurrent4 (); // Current sensor #4
ival = GetMotorCurrent5 (); // Current sensor #5
ival = GetMotorCurrent6 (); // Current sensor #6
ival = GetMotorCurrent (short channel); // Current sensor #1,2,3,4,5,or 6
Parameter: void // for GetMotorCurrentX
short channel; // 0,1,2,3,4,5 for current sensor #1,2,3,4,5,or 6
Copyright © Dr Robot Inc. 2006 42
Return value: short ival;
Return data interpretation:
The return data is the raw value of the analog to digital converter indicating the motor
current. The data range is between 0 and 4095. The real current can be calculated
with the
following formula:
Motor Current (A) = (ival * 3 *375 / 200 /4095) = ival / 728
III.1.10 Encoder
58 short GetEncoderDir1();
59 short GetEncoderDir2();
60 short GetEncoderPulse1();
61 short GetEncoderPulse2();
62 short GetEncoderSpeed1();
63 short GetEncoderSpeed2();
Description:
GetEncoderDirX returns +1, 0 or -1 to indicate the direction of rotation.
GetEncoderPulseX returns the current pulse counter to indicate the position of
rotation.
GetEncoderSpeedX returns the current speed of rotation.
Syntax: ival = GetEncoderDir1(); // direction of channel #1
ival = GetEncoderDir2(); // direction of channel #2
ival = GetEncoderPulse1(); // pulse counter of channel #1
ival = GetEncoderPulse2(); // pulse counter of channel #2
ival = GetEncoderSpeed1(); // speed of channel #1
ival = GetEncoderSpeed2(); // speed of channel #2
Parameter: void
Return value: short ival;
Return data interpretation:
(1) GetEncoderDirX returns -1, 0 or 1. 1 stands for positive direction, -1 stands for
negative
direction, and 0 stands for no movement.

(2) GetEncoderPulseX returns pulse counter. It is an integral value to rotation with range

of 0 ~ 32767 in cycles.

(3) GetEncoderSpeedX returns the rotation speed. The unit is defined as pulse change

within 1 second. And it is the absolute value.

See also: SetDcMotorSensorUsage().

III.1.11 Custom Analog and Digital Inputs and Outputs

64 short GetCustomAD1();

65 short GetCustomAD2();

66 short GetCustomAD3();

67 short GetCustomAD4();

68 short GetCustomAD5();

69 short GetCustomAD6();

70 short GetCustomAD7();

71 short GetCustomAD8();

72 short GetCustomAD (short channel);

Description:

GetCustomADX returns the sampling value of the custom analog to digital input signals. By

default, custom AD1 - AD3 are used as the inputs of power supply voltage monitors for DSP

circuits, DC motors and servo motors. User can change this setting by configuring the

jumpers on PMS5005. Please refer to Chapter IV.I PMS5005 Robot Sensing and Motion

Controller (Page 67) for detailed information on hardware jumper setting.

Syntax: ival = GetCustomAD1(); /* for battery of DSP circuits,

or custom A/D channel #1 */

ival = GetCustomAD2 (); /* for battery of DC motors,

or custom A/D channel #2 */

ival = GetCustomAD3(); /* battery for servo motors,

or custom A/D channel #3 */

ival = GetCustomAD4(); // custom A/D channel #4

ival = GetCustomAD5(); // custom A/D channel #5

ival = GetCustomAD6(); // custom A/D channel #6

ival = GetCustomAD7(); // custom A/D channel #7

ival = GetCustomAD8(); // custom A/D channel #8

ival = GetCustomAD(short channel); /* custom A/D channel #1, 2, 3, 4,

5, 6, 7 or 8 */

Parameter: void

short channel; /* 0, 1, 2, 3, 4, 5, 6 or 7 for

channel #1, 2, 3, 4, 5, 6, 7, 8 */

Return value: short ival;

Return data interpretation:

The return data is the raw value of the analog to digital converter indicating the input

voltage levels. The data range is between 0 and 4095. The voltage levels can be calculated
from the following equation:
Sensor output voltage = (ival) * 3.0 / 4095 (V)
See also: GetSensorBatteryAD1~3
73 short GetCustomDIN();
Description:
GetCustomDIN returns a value with lower 8-bits corresponding to the 8-channel custom
digital inputs.
Syntax: ival = GetCustomDIN ();
Parameter: void
Return value: short ival;
Remarks:
Only lower 8-bit is valid and reflects the 8 input channel states. The MSB of the lower byte
represents channel #8 and LSB of the lower byte represents channel #1.
74 void SetCustomDOUT(short ival);
Description:
SetCustomDOUT sets the 8-channel custom digital outputs.
Syntax: SetCustomDOUT (ival);
Parameter: short ival;
Return value: void
Remarks:
Only the lower 8-bit is valid and can change the corresponding outputs of the 8 channels.
The MSB of the lower byte represents channel #8 and LSB of the lower byte represents
channel #1.

III.2 Motion Control
This section contains the APIs for the operations of DC motors and standard RC servo motors.
The digital controlled DC motor system is depicted as the following diagram.
45
Figure III.1 Digital Controlled DC Motor System
In the case of PID control, the transfer function of the PID controller looks like as:
$$U s E s K K S K S _{P D I}( ) / ( ) = + + /$$
When using potentiometers (optical encoder and etc.) as the rotational position feedback, you have to
set the motor polarity properly using "WiRobotSDK" ActiveX control API "SetMotorPolarityX" so
that the negative feedback is achieved. See "SetMotorPolarityX" for detail.
The control of the standard RC servo motors is carried out by the built-in analog PID controller.
III.2.1 DC Motor Control
75 void SetMotorPolarity1 (short polarity);
76 void SetMotorPolarity2 (short polarity);

77 void SetMotorPolarity3 (short polarity);
78 void SetMotorPolarity4 (short polarity);
79 void SetMotorPolarity5 (short polarity);
80 void SetMotorPolarity6 (short polarity);
81 void SetMotorPolarity (short channel, short polarity);
Description:
SetMotorPolarityX set the motor polarity to 1 or -1 for each motor channel.
1. When the motor is running in positive direction, the potentiometer value is also increasing; motor polarity should be set to 1 which is default.
2. When the motor is running in positive direction, the potentiometer value is decreasing,
motor polarity should be set to -1 or change the sensor mounting so that the potentiometer value increases.
Syntax: ival = SetMotorPolarity1 (short polarity); // Motor #1
ival = SetMotorPolarity2 (short polarity); // Motor #2
ival = SetMotorPolarity3 (short polarity); // Motor #3
ival = SetMotorPolarity4 (short polarity); // Motor #4
-
+ Y
F
G E U
Controller DC Motor
Potentiometer
Copyright © Dr Robot Inc. 2006 46
ival = SetMotorPolarity5 (short polarity); // Motor #5
ival = SetMotorPolarity6 (short polarity); // Motor #6
ival = SetMotorPolarity (short channel, short polarity);
// motor#1, 2, 3, 4, 5, or 6
Parameter: short polarity; //1 or -1
short channel; // 0, 1, 2, 3, 4, or 5 for Sonar #1, 2, 3, 4, 5, 6
Return value: void ival;
82 void EnableDcMotor (short channel);
83 void DisableDcMotor (short channel);
Description:
These functions are obsolete. Please see function ResumeDcMotor(short channel) and
SuspendDcMotor(short channel).
84 void ResumeDcMotor (short channel);
85 void SuspendDcMotor (short channel);
Description:
ResumeDcMotor resumes the specified DC motor control channel.
SuspendDcMotor suspends the specified DC motor control channel. PWM output is all low.
Syntax: ResumeDcMotor (channel);
SuspendDcMotor (channel);
Parameter: short channel; // 0, 1, 2, 3, 4, or 5
Return value: void
Remarks:

1. All motor control channels are initially suspended when the system boot-up.
86 void SetDcMotorPositionControlPID (short channel, short Kp, short Kd, short Ki_x100);
87 void SetDcMotorVelocityControlPID (short channel, short Kp, short Kd, short Ki_x100);
Description:
SetDcMotorPositionControlPID sets up the PID control parameters of the specified DC
motor channel for position control.

SetDcMotorVelocityControlPID sets up the PID control parameters of the specified DC
motor control channel for velocity.
Syntax: SetDcMotorPositionControlPID (channel, K_P, K_D, K_I_x100);
SetDcMotorVelocityControlPID (channel, K_P, K_D, K_I_x100);
Parameter: short channel; // 0, 1, 2, 3, 4, or 5
short K_P; // Proportional gain
short K_D; // Derivative gain
short K_I_x100; // 100 times K_I (the desired Integral gain), when
K_I_x100 = 100, the actual integral control term is K_I
= 1, K_I_x100 with range of 0 ~ 25599
Return value: void
Remarks:
1. When setting K_I = 0, that means NO integral control
2. System default value for position control: K_P = 50; K_D = 5; K_I_x100 = 0.
3. System default value for velocity control: K_P = 50; K_D = 5; K_I_x100 = 0.
See Also: SetDcMotorControlMode
88 void SetDcMotorTrajectoryPlan (short channel, short TrajPlanMthod);
Description:
This function is obsolete.
89 void SetDcMotorSensorFilter (short channel, short FilterMethod);
Description:
This filtering feature is still under development. All data will be treated as raw data.
90 void SetDcMotorSensorUsage (short channel, short SensorType);
Description:
SetDcMotorSensorUsage sets the sensor type for the specified DC motor control channel
on the Sensing and Motion Controller (PMS5005). The available sensor types are single
potentiometer, dual potentiometers, and quadrature encoder. The single potentiometer
sensor is for the control of robot joint with limited rotation range (0˚ to 332˚). The dual
potentiometers and the quadrature sensor are for continuous rotating joint (like wheels)
control.

Syntax: SetDcMotorSensorUsage (channel, SensorType)
Parameter: short channel; // 0, 1, 2, 3, 4, or 5 for single potentiometer sensor

// 0, 1, or 2 for dual potentiometer sensor
// 0 or 1 for quadrature encoder
short SensorType; // 0 -- Single potentiometer sensor
// 1 -- Dual potentiometer sensor
// 2 – Quadrature encoder
Return value: void
Remarks:
1. The electrical angular range of the potentiometer position sensor is 0˚ to 332˚ and
the corresponding mechanical rotation range is 14˚ to 346˚, when the 180
position is defined at sensor's physical middle position.
2. Each DC motor channel for dual potentiometer sensor utilizes two potentiometer
channels. DC motor channel 0 will use potentiometer channel 0 and 5; DC motor
channel 1 will use potentiometer channel 1 and 4; DC motor channel 2 will use
potentiometer channel 2 and 3. Please refer to the relevant application note for the
use of dual potentiometers.
3. Quadrature encoder will only use DC motor channel 0 and 1.
4. System's default setting for sensor usage is single potentiometer.
See also: GetSensorPot
91 void SetDcMotorControlMode (short channel, short controlMode);
Description:
SetDcMotorControlMode sets the control mode of the specified DC motor control
channel
on the Sensing and Motion Controller (PMS5005). The available control modes are
openloop
PWM control, closed-loop position control, closed-loop velocity control.
Syntax: SetDcMotorControlMode (channel, controlMode)
Parameter: short channel; // 0, 1, 2, 3, 4, or 5
short controlMode; // 0 – Open-loop PWM Control
// 1 – Closed-loop Position Control
// 2 – Closed-loop Velocity Control
Return value: void
Remarks:
System's default setting for control mode is Open-loop PWM Control.
See also: SetDcMotorPositionControlPID, SetDcMotorVelocityControlPID

49
92 void DcMotorPositionTimeCtr (short channel, short cmdValue, short timePeriod);
Description:
DcMotorPositionTimeCtr sends the position control command to the specified motion
control channel on the Sensing and Motion Controller (PMS5005). The command
includes
the target position and the target time period to execute the command. The current
trajectory planning method with time control is linear.
Syntax: DcMotorPositionTimeCtr (channel, cmdValue, timePeriod);
Parameter: short channel; // 0, 1, 2, 3, 4, or 5
short cmdValue; // Target position value
short timePeriod; // Executing time in milliseconds
Return value: void
Remarks:

1. Motor will be enabled automatically by the system when this command is received.
2. Target position value is in the A/D sampling data range 0 to 4095 when using single potentiometer, 0-4428 when using dual potentiometers.
3. Please refer to the description of GetSensorPot for data converting between angular values and the A/D sampling data values.
4. When using encoder as sensor input, the target position value is the pulse count in
the range of 0- 32767.
See also: GetSensorPot
93 void DcMotorPositionNonTimeCtr(short channel, short cmdValue);
Description:
DcMotorPositionNonTimeCtr sends the position control command to the specified motion
control channel on the Sensing and Motion Controller (PMS5005). The command includes
the target position but no time period specified to execute the command. The motion controller will drive the motor to the target position at the maximum speed.
Syntax: DcMotorPositionNonTimeCtr (channel, cmdValue);
Parameter: short channel; // 0, 1, 2, 3, 4, or 5
short cmdValue; // Target position value
Return value: void
Remarks:

1. Motor will be enabled automatically by the system when this command is received.
2. Target position value is in the A/D sampling data range 0 to 4095 when using single potentiometer, 0-4428 when using dual potentiometers.
3. Please refer to the description of GetSensorPot for data converting between angular values and the A/D sampling data values.
4. When using encoder as sensor input, the target position value is the pulse count in
the range of 0- 32767.
See also: DcMotorPositionTimeCtr, GetSensorPot
94 void DcMotorVelocityTimeCtr(short channel, short cmdValue, short timePeriods);
Description:
DcMotorVelocityTimeCtr sends the velocity control command to the specified motion
control channel on the Sensing and Motion Controller (PMS5005). The command includes
the target velocity and the time period to execute the command. The current trajectory
planning method for time control is linear.
Syntax: DcMotorVelocityTimeCtr (channel, cmdValue, timePeriod);
Parameter: short channel; // 0, 1, 2, 3, 4, or 5
short cmdValue; // Target velocity value
short timePeriod; // Executing time in milliseconds
Return value: void
Remarks:

1. Motor will be enabled automatically by the system when this command is received
2. No velocity is available for motor channel using single potentiometer sensor
3. The unit of the velocity is (Position change in A/D sampling data) / second when using dual potentiometer sensor for rotational position measurement and pulse/ second when using quadrature encoder.
4. Please refer to the description of GetSensorPot for data conversion between angular values and the A/D sampling data values.
See also: GetSensorPot
95 void DcMotorVelocityNonTimeCtr(short channel, short cmdValue);
Description:
DcMotorVelocityNonTimeCtr sends the velocity control command to the specified motion
control channel on the Sensing and Motion Controller (PMS5005). The command includes
the target velocity but no time period specified to execute the command. The motion controller will drive the motor to achieve the target velocity with maximum effort.

51

Syntax: DcMotorVelocityNonTimeCtr (channel, cmdValue);
Parameter: short channel; // 0, 1, 2, 3, 4, or 5
short cmdValue; // Target velocity value
Return value: void
Remarks:
1. Motor will be enabled automatically by the system when this command is received
2. No velocity is available for motor channel using single potentiometer sensor
3. The unit of the velocity is (Position change in A/D sampling data) / second when using dual potentiometer sensor for rotational position measurement and pulse/ second when using quadrature encoder.
4. Please refer to the description of GetSensorPot for data conversion between angular values and the A/D sampling data values.
See also: DcMotorVelocityTimeCtr, GetSensorPot
96 void DcMotorPwmTimeCtr(short channel, short cmdValue, short timePeriod);
Description:
DcMotorPwmTimeCtr sends the PWM control command to the specified motion control
channel on the Sensing and Motion Controller (PMS5005). The command includes the
target pulse width value and the time period to execute the command. The current trajectory planning method for time control is linear.
Syntax: DcMotorPwmTimeCtr (channel, cmdValue, timePeriod);
Parameter: short channel; // 0, 1, 2, 3, 4, or 5
short cmdValue; // Target pulse width value
short timePeriod; // Executing time in milliseconds
Return value: void
Remarks:
1. The specified channel (motor) will be enabled automatically by the system when this command is received
2. Target pulse width value range is 0 to 32767 (0x7FFF), corresponding to the duty cycle of 0 to 100% linearly.

3. A pulse width value of 16363 means 50% duty cycle, putting motor in "Stop" stage.

Any value in between 16364 - 32767 will put the motor to turn clockwise (facing the front side of the motor) and any value in between 0 – 16362 will put the motor to turn counter-clockwise.

97 void DcMotorPwmNonTimeCtr(short channel, short cmdValue);

Description:

DcMotorPwmNonTimeCtr sends the PWM control command to the specified motion control channel on the Sensing and Motion Controller (PMS5005). The command includes

the target pulse width value without specific execution time period. The motion controller

will set the PWM output of this channel to the target value immediately.

Syntax: DcMotorPwmNonTimeCtr (channel, cmdValue);

Parameter: short channel; // 0, 1, 2, 3, 4, or 5

short cmdValue; // Target pulse width value

Return value: void

Remarks:

1. The specified channel (motor) will be enabled automatically by the system when this command is received

2. Target pulse width value range is 0 to 32767 (0x7FFF), corresponding to the duty cycle of 0 to 100% linearly.

3. A pulse width value of 16363 means 50% duty cycle, putting motor in "Stop" stage.

Any value in between 16364 - 32767 will put the motor to turn clockwise (facing the front side of the motor) and any value in between 0 – 16362 will put the motor to turn counter-clockwise.

See also: DcMotorPwmTimeCtr

98 void DcMotorPositionTimeCtrAll(short cmd1, short cmd2, short cmd3, short cmd4, short cmd5, short cmd6, short timePeriod);

Description:

DcMotorPositionTimeCtrAll sends the position control command to all 6 DC motor control

channels on the sensing and motion controller (PMS5005) at the same time. The command

includes the target positions and the time period to execute the command. The current

trajectory planning method for time control is linear.

Syntax: DcMotorPositionTimeCtrAll (cmd1, cmd2, cmd3, cmd4, cmd5, cmd6, timePeriod);

Parameter: short cmd1; // Target position for channel #1

short cmd2; // Target position for channel #2

short cmd3; // Target position for channel #3

short cmd4; // Target position for channel #4

short cmd5; // Target position for channel #5

short cmd6; // Target position for channel #6

short timePeriod; // Executing time in milliseconds

53
Return value: void
Remarks:
1. All DC Motors will be enabled automatically by the system when this command is received.
2. Target position value is in the A/D sampling data range 0 to 4095 when using single potentiometer, 0-4428 when using dual potentiometers.
3. Please refer to the description of GetSensorPot for data converting between angular values and the A/D sampling data values.
4. When using encoder as sensor input, the target position value is the pulse count in
the range of 0- 32767.
5. When some motors are not under controlled, their command values should be set as
-32768 (0x8000). That means NO_CONTROL.
See also: DcMotorPositionTimeCtr,
99 void DcMotorPositionNonTimeCtrAll(short cmd1, short cmd2, short cmd3,short cmd4,
short cmd5, short cmd6);
Description:
DcMotorPositionNonTimeCtrAll sends the position control command to all 6 DC motor
control channels on the Sensing and Motion Controller (PMS5005) at the same time. The
command includes the target positions without specific execution time period. The motion
controller will drive the motor to reach the target position with maximum effort.
Syntax: DcMotorPositionNonTimeCtrAll(cmd1, cmd2, cmd3, cmd4, cmd5, cmd6);
Parameter: short cmd1; // Target position for channel #1
short cmd2; // Target position for channel #2
short cmd3; // Target position for channel #3
short cmd4; // Target position for channel #4
short cmd5; // Target position for channel #5
short cmd6; // Target position for channel #6
Return value: void
Remarks:
1. All DC motors will be enabled automatically by the system when this command is received.
2. Target position value is in the A/D sampling data range 0 to 4095 when using single potentiometer, 0-4428 when using dual potentiometers.
3. Please refer to the description of GetSensorPot for data converting between angular values and the A/D sampling data values.
4. When using encoder as sensor input, the target position value is the pulse count in
the range of 0- 32767.
54
5. When some motors are not under controlled, their command values should be set as

-32768 (0x8000). That means NO_CONTROL.

See also: DcMotorPositionNonTimeCtr

100 void DcMotorVelocityTimeCtrAll(short cmd1, short cmd2, short cmd3, short cmd4,

short cmd5, short cmd6, short timePeriods);

Description:

DcMotorVelocityTimeCtrAll sends the velocity control command to all 6 DC motor control

channels on the Sensing and Motion Controller (PMS5005) at the same time. The command

includes the target velocities and the time period to execute the command. The trajectory

planning method for time control is linear.

Syntax: DcMotorVelocityTimeCtrAll (cmd1, cmd2, cmd3, cmd4, cmd5, cmd6, timePeriods);

Parameter: short cmd1; // Target velocity for channel #1

short cmd2; // Target velocity for channel #2

short cmd3; // Target velocity for channel #3

short cmd4; // Target velocity for channel #4

short cmd5; // Target velocity for channel #5

short cmd6; // Target velocity for channel #6

short timePeriod; // Executing time in milliseconds

Return value: void

Remarks:

1. Motor will be enabled automatically by the system when this command is received

2. No velocity is available for motor channel using single potentiometer sensor

3. The unit of the velocity is (Position change in A/D sampling data) / second when using dual potentiometer sensor for rotational position measurement and pulse/ second when using quadrature encoder.

4. Please refer to the description of GetSensorPot for data conversion between angular values and the A/D sampling data values.

5. When some motors are not under controlled, their command values should be set as

-32768 (0x8000). That means NO_CONTROL.

See also: DcMotorVelocityTimeCtr

101 void DcMotorVelocityNonTimeCtrAll(short cmd1, short cmd2, short cmd3, short cmd4,

short cmd5, short cmd6);

Description:

55

DcMotorVelocityNonTimeCtrAll sends the velocity control command to all 6 DC motor

control channels on the Sensing and Motion Controller (PMS5005) at the same time. The

command includes the target velocities without specific execution time period. The motion

controller will drive the motor to achieve the target velocity with maximum effort.

Syntax: DcMotorVelocityNonTimeCtrAll (cmd1, cmd2, cmd3, cmd4, cmd5, cmd6);

Parameter: short cmd1; // Target velocity for channel #1

short cmd2; // Target velocity for channel #2

short cmd3; // Target velocity for channel #3

short cmd4; // Target velocity for channel #4

short cmd5; // Target velocity for channel #5

short cmd6; // Target velocity for channel #6

Return value: void

Remarks:

1. Motor will be enabled automatically by the system when this command is received

2. No velocity is available for motor channel using single potentiometer sensor

3. The unit of the velocity is (Position change in A/D sampling data) / second when using dual potentiometer sensor for rotational position measurement and pulse/ second when using quadrature encoder.

4. Please refer to the description of GetSensorPot for data conversion between angular values and the A/D sampling data values.

5. When some motors are not under controlled, their command values should be set as

-32768 (0x8000). That means NO_CONTROL.

See also: DcMotorVelocityNonTimeCtr

102 void DcMotorPwmTimeCtrAll(short cmd1, short cmd2, short cmd3, short cmd4, short

cmd5, short cmd6, short timePeriods);

Description:

DcMotorPwmTimeCtrAll sends the PWM control command to all 6 DC motor control channels on the Sensing and Motion Controller (PMS5005) at the same time. The command

includes the target PWM values and the time period to execute the command. The current

trajectory planning method for time control is linear.

Syntax: DcMotorPwmTimeCtrAll (cmd1, cmd2, cmd3, cmd4, cmd5, cmd6, timePeriods);

Parameter: short cmd1; // Target PWM value for channel #1

short cmd2; // Target PWM value for channel #2

short cmd3; // Target PWM value for channel #3

short cmd4; // Target PWM value for channel #4

short cmd5; // Target PWM value for channel #5

short cmd6; // Target PWM value for channel #6

short timePeriod; // Executing time in milliseconds

Return value: void

Remarks:

1. All channel (motors) will be enabled automatically by the system when this command is received

2. Target pulse width value range is 0 to 32767 (0x7FFF), corresponding to the duty cycle of 0 to 100% linearly.

3. A pulse width value of 16363 means 50% duty cycle, putting motor in "Stop" stage.

Any value in between 16364 - 32767 will put the motor to turn clockwise (facing

the front side of the motor) and any value in between 0 – 16362 will put the motor
to turn counter-clockwise.
4. When some motors are not under controlled, their command values should be set
as
-32768 (0x8000). That means NO_CONTROL.
See also: DcMotorPwmTimeCtr
103 void DcMotorPwmNonTimeCtrAll(short cmd1, short cmd2, short cmd3, short
cmd4,
short cmd5, short cmd6);
Description:
DcMotorPwmNonTimeCtrAll sends the PWM control command to all 6 DC motor
control
channels on the Sensing and Motion Controller (PMS5005) at the same time. The
command
includes the target PWM values without specific execution time period. The motion
controller Send the desired PWM pulse width right away.
Syntax: DcMotorPwmNonTimeCtrAll (cmd1, cmd2, cmd3, cmd4, cmd5, cmd6);
Parameter: short cmd1; // Target PWM value for channel #1
short cmd2; // Target PWM value for channel #2
short cmd3; // Target PWM value for channel #3
short cmd4; // Target PWM value for channel #4
short cmd5; // Target PWM value for channel #5
short cmd6; // Target PWM value for channel #6
Return value: void
Remarks:
1. All channel (motors) will be enabled automatically by the system when this
command is received
2. Target pulse width value range is 0 to 32767 (0x7FFF), corresponding to the duty
cycle of 0 to 100% linearly.
3. A pulse width value of 16363 means 50% duty cycle, putting motor in "Stop"
stage.
Any value in between 16364 - 32767 will put the motor to turn clockwise (facing
the front side of the motor) and any value in between 0 – 16362 will put the motor
to turn counter-clockwise.

4. When some motors are not under controlled, their command values should be set
as
-32768 (0x8000). That means NO_CONTROL.
See also: DcMotorPwmNonTimeCtr
III.2.2 RC Servo Motor Control
104 void EnableServo (short channel);
105 void DisableServo (short channel);
Description:
EnableServo enables the specified servo motor control channel.
DisableServo disables the specified servo motor control channel.
Syntax: EnableServo (channel);
DisableServo (channel);
Parameter: short channel; // 0, 1, 2, 3, 4, or 5

Return value: void

Remarks:

All servo motor channels are disabled initially at system startup. They need to be enabled

explicitly before use.

106 void SetServoTrajectoryPlan(short channel, short TrajPlanMthod);

Description:

This function is obsolete.

107 void ServoTimeCtr(short channel, short cmdValue, short timePeriods);

Description:

ServoTimeCtr sends the position control command to the specified servo motor control

channel on the Sensing and Motion Controller (PMS5005). The command includes the

target position command and the time period to execute the command. The current

trajectory planning method for time control is linear.

Syntax: ServoTimeCtr (channel, cmdValue, timePeriod);

Parameter; short channel; // 0, 1, 2, 3, 4, or 5

short cmdValue; // Target Pulse Width (ms) * 2250

short timePeriod; // Executing time in milliseconds

Return value: void

Remarks:

1. Target position value for cmdValue = (Pulse width in millisecond) * 2250.

2. Usually, a standard remote control servo motor expects to get the specified pulse

width in every 20 milliseconds in order to hold the corresponding angle position.

The pulse width value in millisecond for 0°, 90° and 180° are servo manufacturer

and model dependant, they are around 1ms, 1.5ms and 2.0ms respectively for most

common servos. Experiments are required to obtain the exact value which varies

for different servo motors.

108 void ServoNonTimeCtr(short channel, short cmdValue);

Description:

ServoNonTimeCtr sends the position control command to the specified servo motor

control channel on the Sensing and Motion Controller (PMS5005). The command includes

the target position command without specific execution time period. The motion controller

will send the desired pulse width to the servo motor right away.

Syntax: ServoNonTimeCtr (channel, cmdValue);

Parameter: short channel; // 0, 1, 2, 3, 4, or 5

short cmdValue; // Target Pulse Width (ms) * 2250

Return value: void

Remarks:

Please refer to the remarks under ServoTimeCtr.

See also: ServoTimeCtr

109 void ServoTimeCtrAll(short cmd1, short cmd2, short cmd3, short cmd4, short cmd5,

short cmd6, short timePeriod);

Description:

ServoTimeCtrAll sends the position control command to all 6 servo motor control channels

on the Sensing and Motion Controller (PMS5005) at the same time. The command includes

the target position commands and the time period to execute the command. The current

trajectory planning method for time control is linear.

Syntax: ServoTimeCtrAll (cmd1, cmd2, cmd3, cmd4, cmd5, cmd6, timePeriod);

Parameter: short cmd1; // Target position for channel #1

short cmd2; // Target position for channel #2

short cmd3; // Target position for channel #3

short cmd4; // Target position for channel #4

short cmd5; // Target position for channel #5

short cmd6; // Target position for channel #6

short timePeriod; // Executing time in milliseconds

Return value: void

Remarks:

1. Please refer to the remarks under ServoTimeCtr.

2. When some servo motors are not under controlled, their command values should be

set as -32768 (0x8000). That means NO_CONTROL.

See also: ServoTimeCtr

110 void ServoNonTimeCtrAll (short cmd1, short cmd2, short cmd3, short cmd4, short

cmd5, short cmd6);

Description:

ServoNonTimeCtrAll sends the position control command to all 6 servo motor control

channels on the Sensing and Motion Controller (PMS5005) at the same time. The command

includes the target position commands without specific execution time period. The motion

controller send the desired pulse width to the servo motor right away.

Syntax: ServoNonTimeCtrAll(cmd1, cmd2, cmd3, cmd4, cmd5, cmd6);

Parameter: short cmd1; // Target position for channel #1

short cmd2; // Target position for channel #2

short cmd3; // Target position for channel #3

short cmd4; // Target position for channel #4

short cmd5; // Target position for channel #5

short cmd6; // Target position for channel #6

Return value: void

Remarks:

1. Please refer to the remarks under ServoTimeCtr

2. When some motors are not under controlled, their command values should be set as

-32768 (0x8000). That means NO_CONTROL.

See Also: ServoTimeCtr

III.3 Multimedia Control

III.3.1 LCD Display

111 void LcdDisplayPMS(LPCTSTR bmpFileName);

Description:

LcdDisplayPMS displays the image data in the file bmpFileName (BMP format) on the

graphic LCD connected to the Sensing and Motion Controller (PMS5005).

Syntax: LcdDisplayPMS (bmpFileName);

Parameter: LPCTSTR bmpFileName; // Full path of the BMP file for displaying

Return value: void

Remarks:

The graphic LCD display is mono with dimension of 128 pixels by 64 pixels. The bmp image

must be 128x64 pixels in mono.

III.4 Events

This section documents the four Event mechanisms. When the relevant data arrive from the WiRobot

PMS5005 system, relevant event will be fired, user could write his / her periodic data processing

routine in the relevant event call back function.

112 StandardSensorEvent

Description:

When the standard sensor data arrive, this event will be triggered.

113 CustomSensorEvent

Description:

When the custom expansion sensor (AD and Input) data arrive, this event will be triggered.

114 MotorSensorEvent

Description:

When the motor control related sensor data arrive, this event will be triggered. The motor

control data includes all the motor rotational sensor data such as potentiometer, encoder

and motor current data.

IV. WiRobot SDK API Reference for PMB5010

WiRobot SDK APIs for PMB5010 supports advanced Multimedia Control features.

IV.1 Multimedia Control

This section contains the APIs for the operations of audio input and output, image capturing and LCD

display.

IV.1.1 Audio Input and Output

115 void PlayAudioFile(LPCTSTR fileName);

Description:

PlayAudioFile sends an audio file (.wav format) to the Multimedia Controller (PMB5010).

The file will be played back on the speaker.
Syntax: PlayAudioFile (FileName);
Parameter: LPCTSTR FileName; //the file name with full path
Return value: void
Remarks:
The .wav audio file should contain 16-bit sound wave data sampled at 8 kHz with PCM raw
data format using mono channel. Other supplied wave file format will still be played by the
robot but may have undesired result.
116 void StopAudioPlay ();
Description:
StopAudioPlay stops a playing audio on the Multimedia Controller (PMB5010).
Syntax: StopAudioPlay ();
Return value: void
Remarks:
There will be no effect if no audio is playing.
117 long GetVoiceSegment();
Description:
GetVoiceSegment returns the pointer to current voice data (recorded from robot microphone) in memory.
Copyright © Dr Robot Inc. 2006 62
Syntax: lpVal = GetVoiceSegment();
Parameter: void
Return value: long lpVal; // pointer to current voice data.
Remark:
(1) You should use method GetVoiceSegLength() to get the length of the Voice segment.
(2) Voice data is in PCM raw data format with 16bit, 8KHz sampling rate.
118 long GetVoiceSegLength();
Description:
GetVoiceSegLength returns the length of current voice data in memory.
Syntax: voiceLength = GetVoiceSegLength ();
Parameter: void
Return value: long voiceLength; // Length of current voice data.
See Also: GetVoiceSegment
119 void StartRecord(short voiceSegment);
Description:
StartRecord sends start-recording command to the Multimedia Controller (PMB5010).
The recorded voice data in length specified by voiceSegment will be stored in the shared
memory segment.
Syntax: StartRecord(voiceSegment);
Parameter: short voiceSegment; // segment number for voice data, range 1 -10
Return value: void
Remarks:

The parameter voiceSegment specify the time of voice segment, unit is 256 millisecond
(about 1/4 sec). Value could be 1- 10. For example, if voiceSegment is 4, 1.024 second
voice will be recorded. VoiceSegmentEvent event will fired when the data is ready.

120 void StopRecord();
Description:
StopRecord sends stop-recording command to the Multimedia Controller (PMB5010).
SDK will not send recorded voice data to PC any more.
Syntax: StopRecord();
Parameter: void
Return value: void
Remarks:
There will be no effect if the Multimedia Controller is not recording.

IV.1.2 Image Capturing
121 void TakePhoto();
Description:
TakePhoto sends image capturing command to the Multimedia Controller (PMB5010). The
Multimedia Controller will send back the latest frame of the image data to the WiRobot
shared memory after receiving TakePhoto command. Use SavePhotoAsBMP to obtain the
image.
Syntax: TakePhoto();
Parameter: void
Return value: void
Remarks:
Each TakePhoto command will get one frame of image.
122 BOOL SavePhotoAsBMP(LPCTSTR FileName);
Description:
SavePhotoAsBMP saves current frame of image data into BMP format file FileName.
Syntax: bVal = SavePhotoAsBMP (FileName);
Parameter: LPCTSTR FileName; // the file name with full path, for saving image data in bmp format.
Return value: BOOL bVal; // True: success
// False: failure to save.

Remarks:
1. Before calling SavePhotoAsBMP, the TakePhoto command needs to be called to request image taken.
2. When the image data arrive, the call back event "ImageEvent" will be fired
3. The cause of "failure to save" could be caused because the TakePhoto command was not sent or the file name / path is invalid.

IV.1.3 LCD Display

123 void LcdDisplayPMB(LPCTSTR bmpFileName);

Description:

LcdDisplayPMB displays the image data in the file bmpFileName (BMP format) on the

graphic LCD connected to the Multimedia Controller (PMB5010).

Syntax: LcdDisplayPMB (bmpFileName);

Parameter: LPCTSTR bmpFileName; // Full path of the BMP file for displaying

Return value: void

Remarks:

The graphic LCD display is mono with dimension of 128 pixels by 64 pixels. The bmp image

must be 128x64 pixels in mono.

IV.2 Events

This section documents the two Event mechanisms. When the relevant data arrive from the WiRobot

PMB5010 system, relevant event will be fired, user could write his / her periodic data processing

routine in the relevant event call back function.

124 ImageEvent

Description:

When the image data arrive, this event will be triggered.

125 VoiceSegmentEvent

Description:

When the audio data arrive, this event will be triggered.

V. WiRobot DRK6080/6000/8080/8000 Specific APIs

V.1 Low Level Protection

When bumpers (optional) are installed on WiRobot
RDK6080/6000/8080/8000 with the connection
configuration shown on the right, a build-in low-level bumper
collision protection scheme can be enabled or disabled with
the next two commands. When this bumper protection
feature is enabled:

. The wheels will stop moving forward when either
bumper 0 or 1 is engaged, there will be not affect if
the wheels are moving backward.

. The wheels will stop moving backward when either
bumper 2 or 3 is engaged, there will be not affect if
the wheels are moving forward.

. The bumpers are connected to custom digital I/O
0, 1, 2, and 3.

126 void EnableBumperProtection();

Description: This will enable the low level bumper protection feature.

EnableBumperProtection xxxx.

Syntax: EnableBumperProtection ();

Parameter: void;

Return value: void

Remarks:

By default, the bumper protection feature is disabled when system is booted up.

127 void DisableBumperProtection();

Description: This will disable the low level bumper protection feature.

DisableBumperProtection xxxx.

Syntax: DisableBumperProtection ();

Parameter: void;

Return value: void

D_IN0 D_IN1
D_IN2 D_IN3
Front
Left
Wheel
Bumper_0 Bumper_1
Bumper_2 Bumper_3
Right
Wheel

Figure V.1 WiRobot RDK6080 / 6000
/ 8080 / 8000 Bumper Connection
Configuration

Chapter IV. WiRobot Module

I. PMS5005 Sensing and Motion Controller

I.1 Introduction

The PMS5005 Robot Sensing/Motion Controller can be used as sensing, control, motion execution,

LCD display and wireless communication processing unit for various robotic applications. Its onboard

firmware makes the low level function modules such as motor driver module and wireless

communication module transparent to the users. A host (e.g. PC, DSP, or processor) will be used to

communicate and control the PMS5005 for different applications through the UART (serial) interface.

The system can help robotic and AI researchers and developers focus on the high level logic and

algorithm design, and avoid the hassle of writing low level device drivers, standard control schemes

and troubleshooting the electronic circuits. The ease of use, powerful functionality and onboard

intelligence can eliminate design risk, streamline hardware and software development, and

significantly shorten the time to delivery while effectively reducing the cost. Typical applications

include humanoid robot, legged robot, wheel-based robot, robot head, robot arm and robot hand.

I.1.1 PMS5005 Robot Sensing/Motion Controller Architecture

As shown in Figure I.1, the PMS5005 features functionalities required by most of the robotic

applications, such as sensing, motion control, and data communication.
The PMS5005 contains the following features and capabilities:
- 40MIPS 16-bit fix-point hybrid DSP/MCU
- 36K x 16-bit words flash
- 2.5K x 16-bit words SRAM
- Build-in
. A/D reference voltage monitoring
. Over-heating sensor (x2)
. System voltage monitoring (x1)
. Watchdog timer (x1)
. Full duplex UART (x2)
- Embedded firmware for configurable closed loop position, velocity, various sensor data acquisition,
LCD graphic display, wired and wireless communication
- Interfaces to
. MDM5253 DC motor driver module with position and current feedback (x2), which includes
o General-purpose PWM DC motor interface (x6)
o Motor current feedback interface (x6)
o Potentiometer position feedback sensor interface (x6)
. Quadrature encoder (x2)
. Standard RC servo motor (x6)
. DUR5200 Ultrasonic range sensor module (x6)
. DHM5150 Human sensor module (x2)
. DAT5280 Ambient temperature sensor module (x1)
. GP2Y0A21YK Infrared range sensor (x1)
. DTA5102 2 axis tilt/acceleration sensor module (x1)
. Custom A/D (x 8 including 3 channels of optional battery voltage monitoring). It can connect to
MSA3502 if signal amplifying is needed.
. Custom digital input (x8)
. Custom digital output (x8)
. MGL5128 Graphic LCD display module (128 x 64) (x1)
. MIR5538/5540 Full duplex infrared remote control and communication module (x1)

68
. PMB5010 Multimedia controller (x1)
. MCB3100 Serial Bluetooth wireless module or MCR3210 RS232 interface module (x1) or
WFS802b WiFi802.11b Serial wireless module
Figure I.1 Block Diagram of the PMS5005
I.1.2 PMS5005 Connectors and Jumpers
Figure I.2 shows the function and location of the connectors and jumpers on the PMS5005.
Figure I.2 PMS5005 Connector and Jumper Locations

69
* Note that the size of the PCB board of PMS5005 is about 14.5cm x 10.2cm.

I.2 Operations
The PMS5005 Robot Sensing/Motion Controller is designed to be running as part of the WiRobot
system. The hardware preparation when using the PMS5005 is just simply connecting the relevant

WiRobot modules to the relevant connectors on the PMS5005 board and setting the proper jumper
configurations. Lower device-level operations are handled by the firmware embedded in PMS5005
controller with the following functions:

. Control 6 RC servos

. Driver for LCD display, 6 ultrasonic sensors, 2 human sensors, 1 infrared distance sensor, 1
temperature sensor, 1 tilt/acceleration sensor and 1 infrared remote sensor

. Can interface with different digital devices through the general digital input and output ports

. Can interface with different analog devices through the A/D ports

. Built-in voltage monitoring capability

. Built-in 3 DC motor control schemes, including open-loop PWM control, closed-loop position
control, and closed-loop velocity control. Closed-loop position and velocity control required
the use of encoder or rotary sensor as the feedback device

Users can physically connect the PMS5005 to a host (e.g. PC, processor, or DSP) through null
modem cable or serial wireless modules. By default, the PMS5005's UART setting is 115200, 8, N, 1
with hardware flow control. With this connection, there are two ways to communicate with the
PMS5005:

1. Using WiRobot SDK Software (requires Microsoft platform): High level programs running on
PC can communicate with the PMS5005 firmware using WiRobot SDK Component and
supplied WiRobot Gateway program. Users simply need to make a function call in their
programs to obtain sensor information or to control different devices (e.g. servos, DC motors,
and etc.) without the needs to understand the communication details between PC and
PMS5005. Please refer to the Chapter III. WiRobot SDK API (Page 26) for further
information on programming.

2. Using PMS5005 Communication Protocol: A device (e.g. PC, processor, or DSP) can
communicate with PMS5005 directly using packet-level commands. Such option has no
requirement on the host and provides the freedom for users to choose their development
platform.

I.2.1 PMS5005 Power Supplies and Consumption

Up to three independent groups of power supply can be connected to the PMS5005 supporting board

system circuits (System Power Supply), DC Motor Power Supply and Servo Motor Power Supply

respectively. These power supplies could be connected to the PMS5005 either through the screw

terminals or through the power jacks. Near each screw terminal, there is a connector port for

connecting the power switch or emergency button for each power supply. By default, all three

connector ports are connected together. If power switches are needed, you could disconnect the

connection and add a switch in between for each connector port.

Table I.1 shows the specification of the power supplies. Refer to Section II.2.5 for the connections of

the power jacks and terminals.

Table I.1 Specification of Power Supplies

Power Supply Power
Jack
Screw
Terminals
Switch
Connector
Voltage
Range (V)
Current Capacity
(mA)

70

System J1 PSY S1-0 5.5 – 7.2 500
DC Motor J2 PDM S1-1 6.0 – 25.0 System Specific
Servo Motor J3 PSM S2-1 5.0 – 7.2 System Specific

The system power supply is required at all time for the operation of this board and the power

consumption of PMS5005 without connecting any peripheral modules is about 350mA using a 7.2V

battery pack. We also recommend the use of three different power sources in powering the

PMS5005 (System Power Supply, DC Motor Power Supply and Servo Motor Power Supply) since

high power consumption devices (e.g. high torque servos) may affect the operation of voltage

sensitive devices (e.g. sensors) due to voltage frustration.

Note: Please make sure that the DC motor power supply voltage does not exceed the maximum

allowable voltage for the DC motors.

I.2.2 PMS5005 Jumper Settings

The board address jumpers can be set to any value between 0 and 15. The board address is currently

reserved for future use.

Table I.2 Board Address Jumpers B_ADDR

Bit Pin Value 1 Value 0

0 (LSB) 1, 2 open 1-2 short
1 3, 4 open 3-4 short
2 5, 6 open 5-6 short
3 (MSB) 7, 8 open 7-8 short

JP_AD1, JP_AD2 and JP_AD3 are used for enabling and disabling battery voltage monitoring. If the
jumper is removed, the corresponding power supply monitoring and custom AD_IN will be disabled.

Table I.3 Battery Voltage Monitoring Jumpers
Jumper Position Battery Voltage Monitoring
JP_AD1 1-2 Enable system power supply monitoring
2-3 Disable system power monitoring and connect Custom AD_IN1
JP_AD2 1-2 Enable DC motor power supply monitoring
2-3 Disable DC motor power and connect Custom AD_IN2
JP_AD3 1-2 Enable servo motor power supply monitoring
2-3 Disable servo motor power monitoring and connect Custom
AD_IN3

I.2.3 PMS5005 System Communication Connections
Under the WiRobot system architecture, all controllers are connected in a chain. There is one and only
one host serving as the central controller. All other embedded controllers have at least two SCI ports
for the system communications: upper-reach port and lower-reach port, with the direction respect to
the central controller.

The system communication connection structure of the PMS5005 in the WiRobot system is shown in
Figure I.1. PMS5005 can work solely in the WiRobot system or together with WiRobot Multimedia
Controller PMB5010 when multimedia data (video and audio) is required in the system.

71

Figure I.3 WiRobot System Communication Architecture

The system communication connectors on the PMS5005 are described in Table I.4. Refer to Section
I.2.5 for the definitions of the signals attached to the connector BTOOTH and SCI.

Table I.4 System Communication Connectors
Connector Type Description
BTOOTH Upper Reach SCI port with handshaking and control signals for
both wired and wireless modules
SCI Lower Reach Two-wire serial communication interface (reserved
for future use)

I.2.4 Connecting Peripheral Modules Supported by PMS5005
Table I.5 lists the WiRobot peripheral modules that can be directly connected to the PMS5005 board
and supported by the firmware embedded in PMS5005. Refer to the relevant chapter of these
peripheral modules for the detailed technical information.

Table I.5 Peripheral Modules Supported by PMS5005

Peripheral
Module
Connector Max
No.
Description
WFS802b BTOOTH 1 WiFi802.11 wireless communication
module
MCB3100 BTOOTH 1 Bluetooth wireless communication
module
MCR3210 BTOOTH 1 RS232 interface module
DUR5200 US_1 – 6 6 Ultrasonic range sensor

72

DTA5102 TILT 1 2-Axis tilting and acceleration sensor
DHM5150 HUMAN1 – 2 2 Human motion sensor
DAT5280 TEMPERATURE 1 Ambient temperature sensor
MIR5538/5540 INFRAR 1 Infrared remote controller module
MDM5253 MOTOR1_IN, _OUT
MOTOR2_IN, _OUT
2 3-channel DC motor driver module with
position and current feedback
3rd party SM1 - 6 6 3rd party servo motor
3rd party ENCODER1 - 2 2 3rd party quadrature encoder
GP2Y0A21YK RANGE 1 Infrared range sensor
MGL5128 LCD 1 Mono Graphic LCD display module,
128x64

I.2.5 Connecting DC Motors and Potentiometers to PMS5005

In order to connect DC motors and potentiometers to the PMS5005, MDM5253 (DC Motor Driver
Module with Position and Current Feedback) is required. Each MDM5253 can control up to 3 DC
motors and 3 potentiometers; and each PMS5005 can connect up to 2 MDM5253. The
potentiometer can be used as the position feedback of the DC motor for precise position and velocity
control. Connector MOTOR1-IN and MOTOR1-OUT on PMS5005 are used to connect to a
MDM5253 for DC Motor 1, 2, 3 and Potentiometer 1, 2, 3; and connector MOTOR2-IN and MOTOR2-
OUT are used to connect to a MDM5253 for DC Motor 4, 5, 6 and Potentiometer 4, 5, 6. For details
on how to connect DC motors and potentiometers to the MDM5253, please refer to the Chapter IV.III
MDM5253 (Page 89).

I.2.6 Connecting Custom Sensors/Devices to PMS5005

The PMS5005 has 8 digital inputs, 8 digital outputs and 8 custom A/D extensions. These ports can all
be used to connect to different sensors or output devices. For example, user can connect gyroscope,
more infrared distance sensors or other analog signal devices to PMS5005 by making use of the

available A/D extensions. If a user just wants to have better infrared sensing capabilities in his / her

robot, the PMS5005 can support up to 9 infrared distance sensors (GP2Y0A21YK) through its IR

range sensor port and the 8 custom A/D expansions.

I.2.7 Sample WiRobot Connection Using PMS5005

The following figure illustrates a simple way in using the PMS5005. Note that only a single 7.2V

power source is used to supply power to the system and not all peripheral modules are connected to

the PMS5005 in this figure.

Figure I.4 Sample Connection of WiRobot PMS5005 with Different Peripheral Modules

73

I.2.8 PMS5005 Connections

The definitions of the connector signals of the power supplies and the supported PMS5005

peripheral modules are listed in the following tables.

Table I.6 Connections of the Power Jacks and Terminals

Power Connection Power Jack J1, J2, J3 Screw Terminal PSY, PDM, PSM

Positive Power Source Center Pin 1

Power Supply Ground Circle 2

Table I.7 Upper Reach Communication Port BTOOTH

Pin Name Signal Description

1 VCC +3.3 V

2 RXD Data receiving

3 TXD Data transmitting

4 RTS Request to send

5 CTS Clear to send

6 GND Power supply ground

7 COMRST Reserved

8 BTIN Reserved

Table I.8 Lower Reach Communication Port SCI

Pin Name Signal Description

1 VCC +3.3 V

2 RXD Data receiving

3 TXD Data transmitting

4 GND Power supply ground

Table I.9 Ultrasonic Range Sensor Connectors US_1 - 6

Pin Name Signal Description

1 VCCA +5.0 V

2 URS Ultrasonic echo receiving signal, active rising edge

3 UTE Ultrasonic transmitting enable, active high

4 GND Power supply ground

Table I.10 Tilt and Acceleration Sensor Connector TILT

74

Pin Name Signal Description

1 VCCA +5.0 V

2 AYD Y direction signal, analog 0 – 3.0V

3 AXD X direction signal, analog 0 – 3.0V

4 GND Analog ground

Table I.11 Human Motion Sensor Connectors HUMAN1 - 2

Pin Name Signal Description
1 VCCA +3.0 V
2 HMS Human motion signal, analog 0 – 3.0V
3 HAS Human presence alarm, analog 0 – 3.0V
4 GND Analog ground

Table I.12 Temperature Sensor Connector TEMPERATURE

Pin Name Signal Description
1 VCCA +5.0 V
2 TVS Temperature Data, analog 0 – 3.0V
3 GND Analog ground

Table I.13 Infrared Remote Controller Connector INFRAR

Pin Name Signal Description
1 VCC +3.3 V
2 IRX Receiving from external device, digital
3 ITX Transmitting to external device, digital
4 GND Power supply ground

Table I.14 Servo Motor Connectors SM1 - 6

Pin Name Signal Description
1 SCL Servo motor control
2 VSM Positive servo motor power supply
3 GND Servo motor power supply ground

Table I.15 Quadrature Encoder Connector ENCODER1 - 2

Pin Name Signal Description
1 ENCB Channel B signal
2 VCC +3.3V
3 ENCA Channel A signal
4 ENCI Index signal (reserved for future use)
5 GND Power supply ground

Table I.16 Infrared Range Sensor Connector RANGE

Pin Name Signal Description
1 VCC +5.0 V
2 RVS Range data, analog 0 – 3.0V
3 GND Analog ground

Table I.17 LCD Display Connector LCD

Pin Signal Description
1 VDD +5.0V, power supply for logic
2 VSS Power supply ground
3 Vo LCD operating voltage
4 D0 Data bit 0
5 D1 Data bit 1
6 D2 Data bit 2
7 D3 Data bit 3
8 D4 Data bit 4
9 D5 Data bit 5
10 D6 Data bit 6
11 D7 Data bit 7
12 CS1 Column select 1 ~ 64
13 CS2 Column select 65 ~ 128
14 RESET Reset input

15 R/W Read/write
16 D/I Data/Instruction indication
17 E Enable
18 VEE Negative voltage output
19 A Power supply for LED backlight (+)
20 K Power supply for LED backlight (-)
Table I.18 Custom A/D Expansion Connector EXP-AD

Pin Signal Description
1, 2, 3, 4 + 3.0 V Analog power supply, max. 40mA
5, 6, 15, 16 Ground Analog ground
7 AD_IN1* Analog 0 – 3.0V
8 AD_IN2* Analog 0 – 3.0V
9 AD_IN3* Analog 0 – 3.0V
10 AD_IN4 Analog 0 – 3.0V
11 AD_IN5 Analog 0 – 3.0V
12 AD_IN6 Analog 0 – 3.0V
13 AD_IN7 Analog 0 – 3.0V
14 AD_IN8 Analog 0 – 3.0V
*Note (Table I.18): When the relevant power supply voltage monitoring is enabled, AD_IN1, AD_IN2,
AD_IN3 will be not available to the custom A/D expansions.
Table I.19 Custom Digital I/O Expansion Connector EXP-GPIO
Pin Signal Description
1, 2, 3, 4 + 3.3 V Positive power source, max. 100mA
5 D_OUT0 Digital out
6 D_OUT1 Digital out
7 D_OUT2 Digital out
8 D_OUT3 Digital out
9 D_OUT4 Digital out
10 D_OUT5 Digital out
11 D_OUT6 Digital out
12 D_OUT7 Digital out
13, 14, 15, 16 Ground Power supply ground
17* D_IN0 Digital in
18* D_IN1 Digital in
19* D_IN2 Digital in
20* D_IN3 Digital in
21* D_IN4 Digital in
22* D_IN5 Digital in
23* D_IN6 Digital in
24* D_IN7 Digital in
* NOTE (Table I.19): These pins have been pulled-up to logic high (+ 3.3V) internally.

I.3 Procedure to upgrade the PMS5005 firmware

1. Download and save the latest PMS5005 firmware from www.DrRobot.com
2. Turn off PMS5005 and keep it off until step 9
3. Use a null modem cable to connect the PC to PMS5005 with a RS232 Interface Module
(MCR3210) as shown in Figure I.5. All peripheral modules (e.g. sensors, motors, LCD and etc.)

can still be plugged to the PMS5005 without affecting the upgrade process

Figure I.5 Physical Connection

4. Close all WiRobot software on PC (e.g. WiRobot Gateway and all sample applications)

5. Start the hyper-terminal (which comes with MS Windows OS), give a name to this new

connection and choose the COM port that is connected to the PMS5005 (normally COM1 or

COM2) as shown in the following figure:

Figure I.6 Choosing COM Port Connection

6. Configure the COM port with the setting 115200, 8, N, 1, Xon/Xoff and turn on the "Echo

typed characters locally" under Properties - > Settings -> ASCII Setup. If your PC is slow, you

can turn off this "echo" option for shorter download time but you will not see the download

process during the upgrade.

Figure I.7 COM Port Setting

7. The lower left corner of the hyper-terminal will show the connection status. If the hyperterminal

is still not connected, click the connect icon on the hyper-terminal to establish the connection (don't turn on the PMS5005 yet!).

8. Choose "Transfer -> Send Text File" from the toolbar and set "files of type" to ALL. Locate

the PMS5005 firmware HEX file only by HIGHLIGHTING the file (e.g. PMS5005_v11.dri).

Please make sure that you DON'T double click the file or click the "Open" button

Figure I.8 Locating the HEX File

9. Please read step 10-13 ahead before turning on the PMS5005 in this step

10. After you turn on the PMS5005 (by connecting power to the system power), you should see

the text "(c) 2000-2001 Motorola Inc. S-Record loader. Version 1.1" in the hyper-terminal as

shown in the following figure:

Figure I.9 Status after Turning on the PMS5005

11. Within 5 seconds (start counting when you turn on the PMS5005), you should click the

"Open" button on the Hyper-terminal popup window. Firmware download will then start. If

you fail to start the download within this period of time, the original firmware on PMS5005

will automatically start. You have to turn off the PMS5005, and repeat the download procedure again from Step 2

12. When the download is started, hex numbers will appear on the screen if you have turned on

the "echo" option as described in step 6. Otherwise, you will not see anything but the
download is still running. When the firmware download is completed (takes about 20-
60
seconds, depending on the speed of your PC), you will see the "Application Started"
keyword
as shown in Figure I.10 no matter the "echo" option is turned on or off. The new
downloaded
firmware will automatically start in few seconds and you should see some un-
recognized
characters
Figure I.10 Successful PMS5005 Firmware Upgrade
13. When the download is finished, you could disconnect the COM connection in the
hyperterminal,
and re-start your PMS5005

80

II. PMB5010 Multimedia Controller

II.1 Introduction

The PMB5010 Robot Multimedia Controller can be used as audio, video and
wireless communication
processing unit for various robotic applications. Its onboard firmware makes the low
level function
modules such wireless communication module transparent to the users. A host (e.g.
PC, DSP, or
processor) will be used to communicate and control the PMS5005 for different
applications through
the UART (serial) interface. The system can help robotic and AI researchers and
developers focus on
the high level logic and algorithm designs, and avoid the hassle of writing low level
device drivers,
standard control schemes and troubleshooting the electronic circuits. The ease of
use, powerful
functionality and onboard intelligence can eliminate design risk, streamline hardware
and software
development, and significantly shorten the time to delivery while effectively reducing
the cost. Typical
applications include humanoid robot, legged robot, wheel-based robot, robot head
and intelligent
home device.

II.1.1 PMB5010 Multimedia Controller Architecture

The PMB5010 offers multimedia functionalities that are required by most intelligent
robotic
applications. Figure II.1 shows the system blocks of the PMB5010.
The key features and capabilities are:
- 120MIPS 16-bit fix-point DSP
- 1M x 16-bit words flash
- Up to 256K x 16-bit words SRAM
- Build-in

. Real-time clock

. Full duplex UART (x2)

- Embedded firmware for image capturing, audio recording and playback, and wired and

wireless communication

- Interfaces to

. MAC5310 Audio codec and amplifier module (x1)

. MCI3908 CMOS image sensor module (352 x 288) (x1)

. MCB3100 Serial Bluetooth wireless module or MCR3210 RS232 interface module (x1)

or WFS802b WiFi802.11b serial wireless module

Figure II.1 Block Diagram of the PMB5010

81

II.1.2 PMB5010 Connectors and Jumpers

Figure II.2 shows the function and location of the connectors and jumpers on the PMB5010.

Figure II.2 PMB5010 Connectors and Jumpers

II.2 Operations

The PMB5010 Robot Multimedia Controller is designed to be running as part of the WiRobot system.

The hardware preparation when using PMB5010 is just simply connecting the relevant peripheral

modules to the relevant connectors on the PMB5010 board and setting the proper jumper

configurations. Lower device-level operations are handled by the firmware embedded in PMS5010

controller. High level programs running on PC or other processors are virtually communicating with

the PMB5010 firmware using either WiRobot SDK Component and supplied WiRobot Gateway

program or packet-level commands. Please refer to the Chapter III. WiRobot SDK API (Page 26) for

using WiRobot SDK and WiRobot Communication Protocol for using packet-level commands.

II.2.1 PMB5010 Power Supplies

Up to two power supplies can be connected to the PMB5010 board supporting board system circuits

(System Power Supply) and Parallel Expansion Module (Expansion Power Supply) (reserved)

respectively. These power supplies can be connected to the board either through the screw terminals

or the power jacks. Near each screw terminal, there are two connector ports for connecting power

switches or emergency buttons. By default, these two ports are connectors together. If the power

switches are needed, you could place a switch for each connector port.

Table II.1 Specification of Power Supplies

Power
Supply
Power
Jack
Screw
Terminals
Switch
Connector
Voltage
Range (V)
Current
Capacity (mA)
System JT1 PDMT SW1 5.5 – 7.0 500
Expansion JT2 PSYT SW2 5.0 – 7.2 System Specific

82

II.2.2 PMB5010 Jumper Settings

The board address can be set to any value between 0 and 15. Please refer to the Table II.2 for the
setting values.

Table II.2 Board Address Jumpers B_ADDR

Bit Pin Value 1 Value 0
0 (LSB) 1, 2 open 1-2 short
1 3, 4 open 3-4 short
2 5, 6 open 5-6 short
3 (MSB) 7, 8 open 7-8 short

II.2.3 PMB5010 System Communication Connections

Under the WiRobot system architecture, all the controllers are connected in a chain. There is one and
only one host serving as the central controller. All other embedded controllers have at least two SCI
ports for the system communications: upper-reach port and lower-reach port, with the direction
respect to the central controller.

The system communication connection structure of the PMS5010 in the WiRobot RDK is shown in
Figure II.3. PMB5010 can work solely in the WiRobot system or together with a WiRobot Sensing
and Motion controller PMS5005.

Figure II.3 WiRobot System Communication Architecture

The system communication connectors on the PMB5010 are described in Table II.3. Refer to Section
II.2.5 for the definitions of the signals attached to the connector BLUETOOTHT and SCIT.

Table II.3 System Communication Connectors

83

Connector Type Description
BLUETOOTHT Upper Reach SCI port with handshaking and control signals for
both wired and wireless modules
SCIT Lower Reach Two-wire serial communication interface

II.2.4 Peripheral Modules Supported by PMB5010

Table II.4 lists the WiRobot peripheral modules that can be directly connected to the PMB5010 board
and supported by the firmware embedded in PMB5010. Refer to the relevant chapter of these
peripheral modules for the detailed technical information.

Table II.4 Sub-modules Supported by PMB5010

Sub-module Connector Max No. Description
WFS802b BLUETOOTHT 1 WiFi802.11b wireless communication module
MCB3100 BLUETOOTHT 1 Bluetooth wireless communication module
MCR3210 BLUETOOTHT 1 RS232 interface module
MAC5310 CODEC0 1 Audio codec and amplifier module, which can be used to connect to microphone and speaker
MCI3908 IMAGE 1 CIF CMOS image sensor module

II.2.5 PMB5010 Peripheral Module Connections

The definitions of the connector signals of the power supplies and the PMB5010 peripheral modules
are listed in the following tables.

Table II.5 Connections of the Power Jacks and Terminals

Power Connection Power Jack JT1, JT2 Screw Terminal PDMT, PSYT
Positive Power Source Center 1
Power Supply Ground Circle 2

Table II.6 Upper Reach Communication Port BLUETOOTHT

Pin Name Signal Description
1 VCC +3.3 V
2 RXD Data receiving
3 TXD Data transmitting
4 RTS Request to send
5 CTS Clear to send
6 GND Power supply ground
7 COMRST Reserved
8 BTIN Reserved

84

Table II.7 Lower Reach Communication Port SCIT

Pin Name Signal Description
1 VCC +3.3 V
2 RXD Data receiving
3 TXD Data transmitting
4 GND Power supply ground

Table II.8 Audio Codec and Amplifier Module Connector CODEC0

Pin Signal Description
1 ADIN Data input
2 VCC5 + 5.0V
3 AFS Frame sync
4, 6 GND Power supply ground
5 ADOUT Data output
7 ASCK Shift clock
8 AMCK NC
9 RESET Reset output

10 APDN Power down output
11 AFC Request output for secondary communication
12 AVC3 + 3.3V
Table II.9 CMOS Image Sensor Connector IMAGE
Pin Signal Description
1 VCC5 + 5.0V
2 D0 Image data bit 0
3 ISCL I2C Clock
4 D1 Image data bit 1
5 ISDA I2C data
6 D2 Image data bit 2
7 VS Digital image vertical blank pulse input
8 D3 Image data bit 3
9 HREF Digital image horizontal blank pulse input
10 D4 Image data bit 4
11 RCLK Digital YUV signal synchronized clock input
12 D5 Image data bit 5
13 RESET Reset output
14 D6 Image data bit 6
15 GND Power supply ground

16 D7 Image data bit 7

II.3 Procedure to upgrade the PMB5010 firmware

1. Download and save the latest PMB5010 firmware from www.DrRobot.com
2. Turn off PMB5010 and keep it off until step 9
3. Use a null modem cable to connect the PC to PMB5010. All peripheral modules (e.g. LCD and
etc.) can still be plugged to the PMB5010 without affecting the upgrade process
4. Close all WiRobot software on PC (e.g. WiRobot Gateway and all sample applications)
5. Start the hyper-terminal (which come with MS Windows OS), give a name to this new
connection and choose the COM port that is connected to the PMS5005 (normally COM1 or
COM2) as shown in the following figure:
Figure II.4 Choosing COM Port Connection
6. Configure the COM port with the setting 115200, 8, N, 1, Hardware and turn on "Send line
ends with line feeds" under Properties - > Settings -> ASCII Setup,
Figure II.5 COM Port Setting

7. The lower left corner of the hyper-terminal will show the connection status. If the hyperterminal
is still not connected, click the connect icon on the hyper-terminal to establish the connection (don't turn on the PMB5010 yet!).
8. Choose "Transfer -> Send Text File" from the toolbar and set "files of type" to ALL. Locate
the PMB5010 firmware HEX file only by HIGHLIGHTING the file (e.g. robot.hex). Please

make sure that you DON'T double click the file or click the "Open" button

Figure II.6 Locating the HEX File

9. Please read step 10-13 ahead before turning on the PMB5010 in this step

10. After you turn on the PMB5010, you should see the text "Dr. Robot Inc. PMB5010

Bootloader V1.00 All Right Reserved! 2001, 2003" in the hyper-terminal as shown in the

following figure:

87

Figure II.7 Status after Turning on the PMB5010

11. Within 5 seconds (start counting when you turn on the PMB5010), you should click the

"Open" button on the Hyper-terminal popup window. Firmware download will then start. If

you fail to start the download within this period of time, the original firmware on PMB5010

will automatically start. You have to turn off the PMB5010, and repeat the download procedure again from Step 2

12. When the download is started, you will see the following text. At the end, "Firmware Update

Successfully!" will be shown if the download succeeds. The whole process will take about 1

minute

Figure II.8 Successful PMB5010 Firmware Upgrade

13. When the download is finished, you could disconnect the COM connection in the hyperterminal,

and re-start your PMB5010

88

III. MDM5253 DC Motor Driver Module with Position and Current Feedback

III.1 Introduction

The MDM5253 DC Motor Driver Module with Position and Current Feedback is a three-channel Hbridge

switching power amplifier board. It can be directly controlled by motion controller's logic level

PWM driving signals at a frequency up to 20 KHz. For each of the three independent channels, the

MDM5253 also provides the current feedbacks and connectors for position sensors such as

potentiometers. Each channel is able to drive inductive DC load with current up to 5.0 A and operating

voltage ranging from 5.0 V to 28.0 V.

III.1.1 Features

. 3 Independent channels
. Output 5.0 V to 28.0 V operations
. Up to 5.0 A inductive DC load current capability
. 5.0 V TTL/CMOS compatible Inputs

- PWM Frequencies up to 20 kHz
- Automatic PWM over-current limiting
- Output short circuit protection
- Over-temperature output current reduction and shutdown
- Under-voltage shutdown
- Analog output current feedback
- 3 Connectors for position feedbacks
- Directly plug-on to the WiRobot PMS5005 sensing and motion controller board

III.1.2 Applications

- DC motor and stepper motor control
- Permanent magnet solenoid control
- Robotic systems
- General PWM power amplifier

III.2 Operations

III.2.1 Theory of Operation

When four switches configured as that in Figure III.1, the whole circuit is called an H-bridge. By

controlling the on/off of four switches in certain patterns, the polarity of the supply power on the

control output can be changed. For example, when Control Input 1 and 4 are ON while the Control

Input 2 and 3 are OFF, the controlled load is supplied by power with + on the left and – on the right.

When Control Input 1 and 4 are OFF while the Control Input 2 and 3 are ON, the controlled load is

supplied by power with - on the left and + on the right.

When applying the H-bridge output to a DC motor or other inductive loads with PWM controlled

switching command based on certain algorithms and the feedback signals, full bidirectional magnitude

control, including speed, position and torque control, can be achievable.

89

Figure III.1 H-Bridge Switching Device

In the design of the MDM5253, only one PWM control signal is required to control both the direction

and the magnitude of the output for each channel. The H-bridge's diametrical opposite pairs (control

input 1 and 4, control input 2 and 3) are connected and driven HIGH and LOW together, and the two

pairs are controlled with strictly inverted signals.

Figure III.2 shows the relationship between the PWM duty cycle and system output. The zero average

output occurs when the duty circle is 50%. The direction of the output (in speed control, for example,

the direction of rotation) depends on whether the duty circle is larger than 50% or lower. The

magnitude of the output (rotation speed in speed control) depends on the absolute difference

between the duty circle and 50%.

Figure III.2 Theoretic waveforms of PWM control for the MDM5253

In addition to the PWM control, the MDM5253 can connect up to 3 sensing feedback devices (e.g.

MRS3302). DC motor control schemes, such as position and velocity control, can be implemented by

installing feedback device on DC motor and connecting these devices to MDM5253.

III.2.2 Running as Part of WiRobot System

When using the MDM5253 with the WiRobot system, users simply plug the module onto one of the

DC motor drive expansion connector sets on the PMS5005 Sensing and Motion Controller board

(maximum of 2 MDM5253 modules are supported) and the PMS5005 on-board firmware and device

driver will take care of the motor control and sensing feedback. Since PMS5005 can support 2

MDM5253, it is able to connect and control up to 6 DC motors and have 6 position sensor channels

(POT1-POT6).

Users have an option to use single rotary sensor (e.g. MRS3302 on the Position Sensor Connector),

dual rotary sensor (e.g. 2 MRS3302 on 2 Position Sensor Connectors), or single encoder (Encoder

port on PMS5005) as the feedback device to control each DC motor, if needed. PMS5005 already

has built-in DC motor control schemes and users simply need to select the type of the feedback device

for each DC motor. Note that for single rotary sensor setting, DC motor 1 must use POT1, DC motor 2

must use POT2 and etc.; for dual rotary sensor setting, DC motor 1 must use POT1 and POT6, DC

motor 2 must use POT2 and POT5 and DC motor 3 must use POT3 and POT4; for encoder setting,

DC motor 1 must use ENCODER1 and DC motor 2 must use ENCODER2.

90

By working with the PMS5005, users can simply call a function offered by the WiRobot SDK

software on PC (requires Microsoft platform) or send a data packet (platform independent) to control

the DC motors or to obtain the sensor feedback. Please refer to Chapter III. WiRobot SDK API (Page

26) and Chapter IV.I PMS5005 (Page 67) for the available motor control algorithms and schemes.

III.2.3 Running as a General Purpose DC Motor Driver Module

When using the MDM5253 with third party controllers, the power supply and the input/output signals

should be connected properly (please refer to Section III.3 for connection setting). The controller
sends control commands to the enable pins and the PWM input pins based on your own control
schemes and get current and position feedback data via an analog to digital converter.

III.3 Connections

III.3.1 Board Structure

Figure III.3 shows the structure, locations and functions of the connectors on the MDM5253 module
board.

Figure III.3 MDM5253 Connector Locations

III.3.2 Connector Description

The definitions of the MDM5253 connector signals are listed in the following tables.

Table III.1 Connections of the Load Screw Terminals MOTOR

Terminals Name Description

1 OUT1A Channel #1 output A
2 OUT1B Channel #1 output B
3 OUT2A Channel #2 output A
4 OUT2B Channel #2 output B
5 OUT3A Channel #3 output A
6 OUT3B Channel #3 output B

Table III.2 Position Sensor Connectors POT1 - 3

Pin Name Function

1 VCC3 + 3.0 V
2 PVS Position data, analog 0 – 3.0 V

3 GND3 Signal ground

Table III.3 Control Signal Connector MOTOROUT

Pin Name Function

1 CTL1 Channel #1 PWM control signal
2 CTL2 Channel #2 PWM control signal
3 CTL3 Channel #3 PWM control signal
4 ENA Output enable for all channels:
High: enable; Low: disable
5, 6 GND5 Power supply ground for VCC5
7, 8 VCC5 + 5.0 V
9, 10, 11, 12 GNDM Power supply ground for VCCM
13, 14, 15, 16 VCCM Positive load power source

Table III.4 Feedback Signal Connector MOTORIN

Pin Name Function

1, 2 VCC3 + 3.0 V, positive power source for position sensors
3, 4 GND3 Power supply ground for VCC3
5 CFB1 Channel #1 current feedback data, , analog 0 – 3.0 V
6 CFB2 Channel #2 current feedback data, , analog 0 – 3.0 V
7 CFB3 Channel #3 current feedback data, , analog 0 – 3.0 V
8 PFB1 Channel #1 position feedback data, , analog 0 – 3.0 V
9 PFB2 Channel #2 position feedback data, , analog 0 – 3.0 V
10 PFB3 Channel #3 position feedback data, , analog 0 – 3.0 V

III.4 Specifications

Table III.5 MDM5253 Specification
Parameter Conditions MIN TYP MAX Unit
Power Operating Voltage VCCM 5.0 28.0 V
Switch-off VCCM 4.15 4.4 4.65 V
Switch-on VCCM 4.5 4.75 5.0 V
Under-
Voltage
Shutdown Hysteresis 150 mV
Logic Operating Voltage VCC5 4.5 5.0 5.5 V
Position Sensor Power Supply VCC3 3.0 3.3 V
Standby Supply Current VENA = 0V, IOUT = 0A 65 mA
Control Input HIGH VENA, VCTL 3.5 V
Control Input LOW VENA, VCTL 1.4 V
Enable Input Current IENA 25 100 µA

PWM Input Current ICTL ± 1 µA
Output-on Resistance T = 25 °C 120
ROUT
T = 150 °C 300
mOhm
DC Load Current T < 150 °C 5.0 A
Over-current Protection 7.0 A
Thermal shutdown 175
Over-temperature Protection
Hysteresis 10 30
°C
PWM Frequency 20 KHz
Output ON Delay VCCM = 14V 18 µS
Output OFF Delay VCCM = 14V 18 µS
Output Rise Time VCCM = 14V, IOUT = 3A 2.0 8.0 µS
Output Fall Time VCCM = 14V, IOUT = 3A 2.0 8.0 µS
Disable Delay Time 8.0 µS
Protection Turn-off Time 4.0 µS
Power-off Delay Time 1.0 5.0 µS
Position Sensor Input Range With PMS5005
controller board
0.0 3.0 V
Current Feedback Sensitivity 533 mV/A
Current Feedback Accuracy IOUT > 1.5 A ± 10
IOUT < 1.5 A ± 20
%
Board Size 30 x 58 mm x mm

IV. WFS802b WiFI 802.11 Serial Module with antenna

IV.1 Introduction

The WFS802b WiFi (802.11b) serial module is the most compact, integrated solution available to add
802.11b wireless networking to your robots with a serial interface.
To enable access to a local network or the internet, the WFS802b integrates a fully developed

TCP/IP network stack and OS. The WFS802b also includes an embedded web server that can be
used to remotely configure, monitor, or troubleshoot the attached device.
The WFS802b is the most compact, integrated solution available to add 802.11b wireless networking
to any device with a serial interface. Using our highly integrated hardware and software platform, you
will add to your bottom line by significantly reducing product development time, risk, and cost.

IV.1.1 Features
. Serial to 802.11b conversion
. Dual serial ports up to 921.6kbps per port
. Integrated industry standard 802.11b wireless interface
. 128bit WEP Encryption for security
. Connect any serial device to a wireless network
. Stable, field proven TCP/IP protocol suite and Web-based application framework
. Easy configuration through a web interface
. Embedded web server
. High performance throughput

IV.1.2 Applications
. Robotic systems: both run-time and development-stage communication
. General-purpose wireless data communication

IV.2 Operations

IV.2.1 Protocol Support
The WFS802b uses the widely accepted 802.11b protocol to connect to a wireless access point or an
ad hoc network. It uses the Transmission Control Protocol (TCP) to ensure that no data is lost or
duplicated and everything sent to the connection arrives correctly at the target.
The WFS802b also supports User Datagram Protocol (UDP) for typical datagram applications in
which devices interact with other devices without maintaining a point-to-point connection.

IV.3 Connections

IV.3.1 Board Structure
Figure IV.1 illustrates the structure of the board

94
Figure IV.1 WFS802bStructure
IV.3.2 Connector Description
The WFS802b is connected to WiRobot system via an 8-pin 2.54 mm-pitch single row connector1
(COM1):
Table IV.1 Connector1 (COM1)

Pin Name Function
1 VCC +3.3 V
2 RXD Data receiving
3 TXD Data transmitting

4 RTS Request to send
5 CRTS Clear to send
6 GND Power supply ground
7 NC Reserved
8 NC Reserved
Table IV.2 Connector2 (COM2)
Pin Name Function
1 NC Reserved
2 RXD Data receiving
3 TXD Data transmitting
4 RTS Request to send
5 CRTS Clear to send
6 GND Power supply ground
7 NC Reserved
8 NC Reserved

95

IV.4 Specifications

Table IV.3 WFS802b Specification
Network Standard IEEE 802.11b
Frequency Range 2.412 – 2.484 GHz
Radio # of Selectable Channels 14 Channels
Security Password protection, locking features, WEP 64/128
Maximum Receive Level -10dBm (with PER < 8%)
Receiver Sensitivity ．-82dBm for 11Mbps
．-87dBm for 5.5Mbps
．-89dBm for 2.0Mbps
．-93dBm for 1.0Mbps
WLAN Power and Link LED Current Max: 4mA
Firmware Upgradeable via serial port
Serial Interface CMOS (Asynchronous) 3.3V-level signals
Rate is software selectable (300 bps to 921600 bps)
Serial Line Formats 7 or 8 data bits, 1-2 Stop bits,
Parity: odd, even, none
Modem Control DTR, DCD
Flow Control XON/XOFF (software), CTS/RTS (hardware), none
Network Interface Wireless 802.11b
Protocols Supported 802.11b, UDP, TCP, DHCP
Data Rates With Automatic Fallback ．11Mbps
．5.5Mbps
．2Mbps
．1Mbps
Media Access Control CSMA/CA with ACK
Frequency Range 2.412 – 2.484 GHz
Range Up to 328 feet indoors
Modulation Techniques ．CCK (11Mbps)
．CCK (5.5 Mbps)
．DQPSK (2 Mbps)
．DBPSK (1 Mbps)
Transmit Output Power 14dBm ± 1dBm
Average Power Consumption ．1280 mW (WLAN mode; maximum data rate)
．820 mW (WLAN mode; idle)

. 710 mW (Ethernet mode)
Peak Supply Current 460 mA
Management Internal web server
Weight with antenna 50 grams
Temperature Operating range, WLAN: -40°C to +70°C
Size (w/o antenna) 50 mm x 40 mm x 15 mm

IV.5 Configuration via Serial Mode or Telnet Port

Configure the unit so that it can communicate on a network with your serial device.
The WFS802b unit is configurable using a terminal program to access the serial port locally. Using this
terminal program to respond to prompts is referred to as the Setup Mode. A Telnet connection may also be
used to configure the unit over the network.
The unit's configuration is stored in nonvolatile memory and is retained without power. You can change the
configuration at any time. The unit performs a reset after the configuration has been changed and stored.
Note: The menus in this section show a typical device. Not all devices display information in the same
manner.
This chapter includes the following topics:
. Accessing Setup Mode
. Server Configuration
. Channel 1 and Channel 2 Configuration
. Email Configuration
. WLAN Settings
. Expert Settings
. Security Settings
. Factory Defaults
. Exit Configuration Mode
Figure IV.1 Connection WFS802b with MCR3210P RS232 Interface Module
IV.5.1 Accessing Setup Mode
Telnet Access
To configure the unit over the network, establish a Telnet connection to port 9999:
1. From the Windows Start menu, click Run.
2. From the Run dialogue box, type the following command (where x.x.x.x is the IP

address and 9999 is the unit's fixed network configuration port number):
Windows: telnet x.x.x.x 9999
UNIX: telnet x.x.x.x:9999
3. Click OK. The following information displays:
Figure IV.1 MAC Address
MAC address 00204AFFFF30
Software version 05.3(040129)WPT
Press Enter to go into Setup Mode
4. To enter the Setup Mode, press Enter within 5 seconds.

Note: Connection fails if Enter is not pressed within 5 seconds.
The configuration settings display, followed by the setup menu options:
Figure IV.2 Setup Menu Options
Change Setup:
0 Server
1 Channel 1
2 Channel 2
3 Email
4 WLAN
5 Expert
6 Security
7 Factory defaults
8 Exit without save
9 Save and exit Your choice ?
5. Select an option on the menu by entering the number of the option in the Your choice ? field and pressing Enter.
View the current configuration by pressing Enter from the Change Setup menu.
To enter a value for a parameter, type the value and press Enter. To confirm a current value, press Enter (without inputted parameters).
6. When finished, save the new configurations (9 Save and exit). The unit reboots.
Serial Port Access
To configure the unit through a serial connection:
1. Connect a console terminal or PC running a terminal emulation program to your unit's serial port. The default serial port settings are 9600 baud, 8 bits, no parity, 1 stop bit, no flow control.
2. Reset the WFS802b unit by cycling the unit's power (turning the power off and back on). Immediately upon resetting the device, enter three lowercase x characters

(xxx).
Note: The easiest way to enter Setup Mode is to hold down the x key at the terminal
(or emulation) while resetting the unit. This must be done within three seconds of resetting the WFS802b.
3. Upon connection, the following information displays:
Figure IV.3 MAC Address
MAC address 00204AFFFF30
Software version 05.3 (040129) WPT
Press Enter to go into Setup Mode
4. To enter the Setup Mode, press Enter within 5 seconds
Note: Connection fails if Enter is not pressed within 5 seconds.
The configuration settings display, followed by the setup menu options:
Figure IV.4 Setup Menu Options
Change Setup:
0 Server
1 Channel 1
2 Channel 2
3 Email

4 WLAN

5 Expert

6 Security

7 Factory defaults

8 Exit without save

9 Save and exit Your choice ?

5. Select an option on the menu by entering the number of the option in the Your choice ?

field and pressing Enter.

View the current configuration by pressing Enter from the Change Setup menu. To enter a value for a parameter, type the value and press Enter. To confirm a current value, press Enter (without inputted parameters).

6. When finished, save the new configurations (9 Save and exit). The unit reboots

IV.5.2 Server Configuration

The unit's basic server (i.e. network) values display upon selecting Server (option 0 from the

Change Setup menu). The following sections describe the configurable parameters within the

Copyright © Dr Robot Inc. 2006 99

Server configuration menu.

Set the IP Address

If DHCP is not used to assign IP addresses, enter it manually. The IP address must be set to a

unique value in the network. Enter each octet and press Enter between each section inputted.

The current value is displayed in parentheses.

IP Address : (0) (0) (0) (0)

Set the Gateway IP Address

The gateway address, or router, allows communication to other LAN segments. The gateway address should be the IP address of the router connected to the same LAN segment as the unit. The gateway address must be within the local network.

The default is N (No), indicating the gateway address has not been set. To set the gateway

address, type Y. At the prompt, enter the gateway address.

Set Gateway IP Address (N) ? Y

Gateway IP addr (0) (0) (0) (0)

Set the Netmask

A netmask defines the number of bits taken from the IP address that are assigned for the

host part.

Netmask: Number of Bits for Host Part (0=default) (0)

The unit prompts for the number of host bits to be entered, then calculates the netmask, which

displays in standard decimal-dot notation when the saved parameters are displayed (for

example, 255.255.255.0).

Table IV.4 Standard IP Network Netmasks Representing Host Bits

Network Class Host Bits Netmask

A 24 255.0.0.0

B 16 255.255.0.0

C 8 255.255.255.0

Change Telnet Configuration Password

Setting the Telnet configuration password prevents unauthorized access to the setup
menu via a Telnet connection to port 9999 or via web pages. The password must have
4 characters.

Change telnet config password (N) ?

An enhanced password setting (for Telnet access only) of 16 characters is available
under

option 6 Security from the Change Setup menu.

Note: A password is not required to access the Setup Mode window via a serial

connection.

DHCP Name

There are three methods for assigning DHCP names to the unit.

. Default DHCP Name: If the DHCP name is not changed and the IP is 0.0.0.0, then
the

DHCP name defaults to CXXXXXX (XXXXXX is the last 6 digits of the MAC address
shown

on the label on the bottom/side of the unit). For example, if the MAC address is 00-
20-

4A-12-34-56, then the default DHCP name is C123456.

. Custom DHCP Name: Create your own DHCP name. If using an IP address of
0.0.0.0,

then the last option in Server configuration is Change DHCP device name. This
option

allows you to change the DHCP name to an alphanumeric name (LTX in the
example).

Change DHCP device name (not set) ? (N)

Enter new DHCP device name : LTX

. Numeric DHCP Name: Change the DHCP name by specifying the last octet of the
IP

address. When using this method, the DHCP name is LTXYY where YY is the last
octet of

the IP address. If the IP address specified is 0.0.0.12, then the DHCP name is
LTX12.

This method only works with 2 digit numbers (0-99).

IV.5.3 Channel 1 and Channel 2 Configuration

Select option 1 Channel 1 or 2 Channel 2 from the Change Setup menu to define
how the serial

port responds to network and serial communications. The following sections describe
the

configurable parameters within the Channel configuration menu.

Figure IV.5 Serial and Telnet Port Parameters

Baudrate

The unit and attached serial device, such as a modem, must agree on a speed or baud rate to
use for the serial connection. Valid baud rates are 300, 600, 1200, 2400, 4800, 9600 (default), 19200, 38400, 57600, 115200, 230400, 460800, or 921600. The current value is displayed in parentheses.
Baudrate (9600) ? _
I/F (Interface) Mode
The Interface (I/F) Mode is a bit-coded byte entered in hexadecimal notation. The current value is displayed in parentheses.
I/F Mode (4C) ? _

101

The following table displays available I/F Mode options:
Table IV.5 Interface Mode Options
*(1) 2 stop bits are implemented by the software. This might influence performance.
Note: If attempting to select an I/F Mode bit pertaining to RS-422/485, a
"WARNING: RS-422/485 I/F Modes not supported" message displays.
The following table demonstrates some common I/F Mode settings:
Table IV.6 Common Interface Mode Settings
Common I/F Mode Setting Binary Hex
RS-232C, 8-bit, No Parity, 1 stop bit 0100 1100 4C
RS-232C, 7-bit, Even Parity, 1 stop bit 0111 1000 78
Flow
Flow control sets the local handshaking method for stopping serial input/output. The current value is displayed in parentheses.
Flow (0) ?
Use the following table to select flow control options:
Table IV.7 Flow Control Options
Flow Control Option Hex
No flow control 00
XON/XOFF flow control 01
Hardware handshake with RTS/CTS lines 02
XON/XOFF pass characters to host 05
Port Number
The Port No setting represents the source port number in TCP connections. It is the number that identifies the channel for remote initiating connections. The port number functions as the TCP/UDP source port number for outgoing packets. Packets sent to the
I/F Mode Option 7 6 5 4 3 2 1 0
RS-232C (1) 0 0
7 Bit 1 0
8 Bit 1 1
No Parity 0 0
Even Parity 1 1
Odd Parity 0 1
1 stop bit 0 1
2 stop bits(1) 1 1

102

unit with this port number are received to this channel. The port number selected is the

Incoming TCP/UDP port and Outgoing TCP/UDP source port.

Port No (10001) ?

The current value is displayed in parentheses. The default setting for Port 1 is 10001. The

range is 1-65535, except for the following reserved port numbers:

Table IV.8 Reserved Port Numbers

Port Numbers Reserved for

1 – 1024 Reserved

9999 Telnet setup

14000-14009 Reserved for Redirector

30704 Reserved (77F0h)

30718 Reserved (77FEh)

Note: It is recommended to not use the reserved port numbers for

this setting as incorrect operation may result.

Use Port 0 for the outgoing local port to change with each connection. The port range is

50,000 to 59,999. Each subsequent connection increments the number by 1 (it wraps back

around to 50,000).

Only use this automatic port increment feature to initiate a connection using TCP. Set the port

to a non-zero value when the unit is in a passive mode or when using UDP instead of TCP.

Connect Mode

Connect Mode defines the unit's connection method and its reaction to incoming connections over the network. The current value is displayed in parentheses.

ConnectMode (C0) ?

Enter Connect Mode options in hexadecimal notation:

Table IV.9 Connect Mode Options

Connect Mode Option 7 6 5 4 3 2 1 0

a) Incoming Connection

Never accept incoming 0 0 0

Accept with modem-control_in Active 0 1 0

Always Accept 1 1 0

b) Response

Nothing (quiet) 0

Character response (C=connect,

D=disconnect, N=unreachable)

1

c) Active Startup

No active startup 0 0 0 0

With any character 0 0 0 1

With modem_control_in Active 0 0 1 0

With a specific start character 0 0 1 1

Manual connection 0 1 0 0
Autostart 0 1 0 1
Hostlist 0 0 1 0
d) Datagram Type
Directed UDP 1 1 0 0
e) Modem Mode
Full Verbose 1 0 1 1 0
Without Echo 0 0 1 1 0
Numeric modem result codes 1 0 1 1 1

a) Incoming Connection

Never Accept Incoming Rejects all external connection attempts
Accept with modem_control_in Active Accepts external connection requests only
when the modem_control_in input is asserted.
Cannot be used with Modem Mode
Always Accept Accepts any incoming connection when a
connection is not already established. Default
setting

b) Response

Character Response A single character is transmitted to the serial port when there
is a change in connection state:
C = connected, D = disconnected, N = host unreachable.
This option is overridden when the Active Start Modem Mode
or Active Start Host List is in effect. Default setting is
Nothing (quiet).
No Active Startup Does not attempt to initiate a connection. Default setting
With Any Character Attempts to connect when any character is received from the
serial port
Accept with
modem_control_in
Active
Attempts to connect when the modem_control_in input
changes from not asserted to asserted
With a Specific Start
Character
Attempts to connect when it receives a specific start character
from the serial port. The default start character is carriage
return
Manual Connection Attempts to connect when directed by a command string
received from the serial port. The first character of the
command string must be a C (ASCII 0x43), and the last
character must be either a carriage return (ASCII 0x0D) or a
line feed (0x0A). No blanks or space characters may be in the
command string. Between the first and last command string
characters must be a full or partial destination IP address and

can include a destination port number.
The IP address must be in standard dot-decimal notation and

may be a partial address, representing the least significant 1, 2, or 3 bytes of the remote IP address. The period is required between each pair of IP address numbers.

If present, the port number must follow the IP address, must be presented as a decimal number in the range 1-65535, and must be preceded by a forward slash (ASCII 0x2F). The slash separates the IP address and the port number. If you omit the port number from a command string, the internally stored remote port number starts a connection.

If a partial IP address is presented in a command string, it is interpreted to be the least significant bytes of the IP address and uses the internally stored remote IP address to provide the most significant bytes of the IP address. If the IP address entered is 0.0.0.0/0, the device server enters Monitor Mode.

For example, if the remote IP address already configured in the unit is 129.1.2.3, then an example command string would be C3/7. (This would connect to 129.1.2.3 and port 7.) You may also use a different ending for the connection string. For example, C50.1/23 would connect you to 129.1.50.1 and port 23.

Table IV.10 Manual Connection Address Example

Command String Result if remote IP is 129.1.2.3 and remote port is 1234
C121.2.4.5/1 Complete override; connection is started with host 121.2.4.5, port 1
C5 Connects to 129.1.2.5, port 1234
C28.10/12 Connects to 129.1.28.10, port 12
C0.0.0.0/0 Connects to 129.1.28.10, port 12; enters Monitor Mode

Autostart (Automatic
Connection)

The unit automatically attempts a connection to the remote IP address and port after booting up

Hostlist If this option is set to True, the device server scrolls through the host list until it connects to the first available device listed in the host list table. Once it connects, the unit stops further attempts. If this connection fails, the unit continues to scroll through the table until it is able to connect to the next available IP address in the host list.

Hostlist supports a minimum of 1 and a maximum of 12 entries. Each entry contains the IP address and the port number.

The hostlist is disabled for Manual Mode and for Modem Mode. The unit will not accept a data connection from a remote device when the hostlist option is enabled.

Figure IV.6 Hostlist Example

Baudrate (9600) ?
I/F Mode (4C) ?
Flow (00) ?
Port No (10001) ?
ConnectMode (C0) ?25

Hostlist :
No Entry !
Change Hostlist ? (N) Y
01. IP address : (000) 172.(000) 19.(000) 0.(000) 1 Port :
(0) ?23
02. IP address : (000) 172.(000) 19.(000) 0.(000) 2 Port :
(0) ?3001
03. IP address : (000) 172.(000) 19.(000) 0.(000) 3 Port :
(0) ?10001
04. IP address : (000) .(000) .(000) .(000)
Hostlist :
01. IP : 172.019.000.001 Port : 00023
02. IP : 172.019.000.002 Port : 03001
03. IP : 172.019.000.003 Port : 10001
Change Hostlist ? (N) N
Hostlist Retrycounter (3) ?
Hostlist Retrytimeout (250) ?
DisConnMode (00) ?
FlushMode (00) ?
DisConnTime (00:00) ?:
SendChar 1 (00) ?
SendChar 2 (00) ?
To enable the hostlist:
1. Enter a Connect Mode of 0x20. The menu shows a list of current entries already defined in the product.
2. To delete, modify, or add an entry, select Yes. If entering an IP address of 0.0.0.0, that
entry and all others after it are deleted.
3. After completing the hostlist, repeat the previous step if necessary to edit the hostlist again.
4. For Retrycounter, enter the number of times the Lantronix unit should try to make a
good network connection to a hostlist entry that it has successfully ARPed. The range is
1-15, with the default set to 3.
5. For Retrytimeout, enter the number of seconds the unit should wait before failing an
attempted connection. The time is stored as units of milliseconds in the range of 1-65535.
The default is 250.

c) Datagram Type
Directed UDP When selecting this option, the prompt requests the Datagram
type. Enter 01 for directed or broadcast UDP.
When the UDP option is in effect, the unit uses UDP
datagrams to send and receive data.

d) Modem Mode

In Modem (Emulation) Mode, the unit presents a modem interface to the attached serial device.

It accepts AT-style modem commands, and handles the modem signals correctly.

Normally, there is a modem connected to a local PC and a modem connected to a remote

machine. A user must dial from the local PC to the remote machine, accumulating phone

charges for each connection. Modem Mode allows you to replace modems with WFS802bs,

and to use an Ethernet connection instead of a phone call. By not having to change communications applications, you avoid potentially expensive phone calls.

To select Modem Mode, set the Connect Mode to C6 (no echo), D6 (echo with full verbose), or D7 (echo with 1-character response).

Note: If the unit is in Modem Mode, and the serial port is idle, the unit can still accept network

TCP connections to the serial port if Connect Mode is set to C6 (no echo), D6 (echo with full

verbose), or D7 (echo with 1-character response).

Without Echo In Modem Mode, echo refers to the echo of all of the

characters entered in command mode; it does not mean to

echo data that is transferred. Quiet Mode (without echo)

refers to the modem not sending an answer to the commands

received (or displaying what was typed).

Full Verbose The unit echoes modem commands and responds to a

command with a message string shown in the table below.

1-Character

Response

The unit echoes modem commands and responds to a

command with a single character response.

Table IV.11 Modem Mode Messages

Message Meaning

Full Verbose

OK Command was executed without error

CONNECT A network connection has been established

NO CARRIER A network connection has been closed

RING n.n.n.n

A remote device, having IP address n.n.n.n, is connecting to

this device.

1-Character Response

0 OK

1 Connected

2 Ring

3 No Carrier

4 Error

Received commands must begin with the two-character sequence AT and be terminated with a carriage return character.

The unit ignores any character sequence received not starting with AT, and only recognizes
and processes single AT-style commands. The unit treats compound AT commands as
unrecognized commands.
If the Full Verbose option is in effect, the unit responds to an unrecognized command string
that is otherwise formatted correctly (begins with AT and ends with carriage return) with the
OK message and takes no further action.
If the 1-Character Response option is in effect, the unit responds to an unrecognized command string that is otherwise formatted correctly with OK and takes no further action.

Copyright © Dr Robot Inc. 2006 107

When an active connection is in effect, the unit transfers data and does not process commands received from the serial interface.
When a connection is terminated or lost, the unit reverts to command mode. When an active connection is in effect, the unit terminates the connection if it receives the following sequence from the attached serial device:
. No serial data is received for one second.
. The character sequence +++ is received, with no more than one second between each two characters.
. No serial data is received for one second after the last + character. At this time, the unit responds affirmatively per the selected echo/response mode.
. The character string ATH is received, terminated with a carriage return. The unit responds affirmatively according to the selected echo/response mode and drops the network connection. The serial interface reverts to accepting command strings.
If this sequence is not followed, the unit remains in data transfer mode.
Table IV.12 Modem Mode Commands
Modem Mode Command Function
ATDTx.x.x.x,pppp or
ATDTx.x.x.x/pppp
Makes a connection to an IP address (x.x.x.x) and a
remote port number (pppp).
ATDTx.x.x.x Makes a connection to an IP address (x.x.x.x) and the
remote port number defined within the unit.
ATD0.0.0.0 Forces the unit into Monitor Mode. Uses remote IP
address and port settings to initiate a connection.
ATD or ATDT
Forces the unit into Monitor Mode. Uses remote IP
address and port settings to initiate a connection.
ATDx.x.x.x Makes a connection to an IP address (x.x.x.x) and the
remote port number defined within the unit.
ATH Hangs up the connection (Entered as +++ ATH).
ATS0=n Enables or disables connections from the network
going to the serial port.

n=0 disables the ability to make a connection from the
network to the serial port.
n=1-9 enables the ability to make a connection from the
network to the serial port.
n>9 is invalid
ATEn Enables or disables character echo and responses.
n=0 disables character echo and responses.
n=1 enables character echo and responses
ATVn Enables 1-character response or full verbose.
n=0 enables 1-character response.
n=1 enables full verbose.
Note: The unit recognizes these AT commands as single commands such as
ATE0 or ATV1; it

Copyright © Dr Robot Inc. 2006 108

does not recognize compound commands such as ATE0V.
Remote IP Address
This is the destination IP address used with an outgoing connection. The current value is
displayed in parentheses.
Remote IP Address : (0) (0) (0) (0)
Note: This option is not displayed when Hostlist is enabled from the
ConnectMode
prompt
Remote Port
Set the remote TCP port number for the unit to make outgoing connections. This
parameter defines the port number on the target host to which a connection is
attempted.
To connect an ASCII terminal to a host using the unit for login purposes, use the
remote port
number 23 (Internet standard port number for Telnet services)
Remote Port (0) ?
Note: This option is not displayed when Hostlist is enabled from the
ConnectMode
prompt
DisConnMode
Disconnect Mode (DisConnMode) determines the conditions under which the unit will
cause a
network connection to terminate. The current value is displayed in parentheses.
DisConnMode (0) ?
In DisConnMode, modem_control_in either drops the connection or is ignored. The
following table displays the available input options:
Table IV.13 Disconnect Mode Options
Disconnect Mode Option 7 6 5 4 3 2 1 0
Disconnect with
modem_control_in drop (6)
1
Ignore modem_control_in 0

Telnet mode and terminal type
setup (1)
1
Channel (port) password (2) 1
Hard disconnect (3) 0
Disable hard disconnect 1
State LED off with connection (4) 1
Disconnect with EOT (^D) (5) 1
(1) The WFS802b sends the "Terminal Type" upon an outgoing connection.
(2) A password is required for a connection to the serial port from the network.
(3) The TCP connection closes even if the remote site does not acknowledge the disconnection.
(4) When there is a network connection to or from the serial port, the state LED turns off

109

instead of blinking.
(5) When Ctrl D or Hex 04 is detected, the connection is dropped. Both Telnet mode and
Disconnect with EOT must be enabled for Disconnect with EOT to function properly. Ctrl D
is only detected going from the serial port to the network.
(6) When modem_control_in transitions from a high state to a low state, the network connection to or from the serial port drops.
Flush Mode
The FlushMode (buffer flushing) parameter controls line handling and network buffers with
connection startup and disconnect.
FlushMode (0) ?
Select between two different packing algorithms (the current configuration is displayed
within the parentheses). Available Flush Mode options are:
Table IV.14 Flush Mode Options
Function 7 6 5 4 3 2 1 0
Input Buffer (Serial to Network)
Clear with a connection that is
initiated from the device to the
network
1
Clear with a connection initiated from
the network to the device
1
Clear when the network connection
to or from the device is disconnected
1
Output Buffer (Network to Serial)
Clear with a connection that is
initiated from the device to the
network

1

Clear with a connection initiated from
the network to the device
1

Clear when the network connection
to or from the device is disconnected
1

Alternate Packing Algorithm (Pack Control)
Enable 1

Pack Control

The packing algorithm defines how and when packets are sent to the network. The standard algorithm is optimized for applications in which the unit is used in a local environment. The alternate packing algorithm minimizes the packet count on the network and is especially useful in applications in a routed Wide Area Network (WAN).

Adjusting parameters in this mode can economize the network data stream. Pack control settings are enabled in Flush Mode. Set this value to 00 if specific functions are not needed.

Table IV.15 Pack Control Options
Option 7 6 5 4 3 2 1 0

Packing Interval
Interval: 12ms 0 0
Interval: 52ms 0 1
Interval: 250ms 1 0

110

Interval: 5sec 1 1

Trailing Characters
None 0 0
One 0 1
Two 1 0

Send Characters
2-Byte Send Character Sequence 1
Send Immediately After Send chars 1

Packing Interval: Packing Interval defines how long the unit should wait before sending

accumulated characters. This wait period is between successive network segments containing data. For alternate packing, the default interval is 12 ms.

Trailing Characters: In some applications, CRC, Checksum, or other trailing characters

follow the end-of-sequence character; this option helps to adapt frame transmission to

the frame boundary.

Send Characters:

. If 2-Byte Send Character Sequence is enabled, the unit interprets the sendchars as a

2-byte sequence; if this option is not enabled, the unit interprets them independently.

. If Send Immediately After Characters is not set, any characters already in the serial
buffer are included in the transmission after a "transmit" condition is found. If this
option is set, the unit sends immediately after recognizing the transmit condition
(sendchar or timeout).
Note: A transmission might occur if status information needs to be
exchanged or an acknowledgment needs to be sent.
DisConnTime (Inactivity Timeout)
Use this parameter to set an inactivity timeout. The unit drops the connection if there is no
activity on the serial line before the set time expires. Enter time in the format mm:ss, where m is
the number of minutes and s is the number of seconds.
DisConnTime (0: 0) ?:
To disable the inactivity timeout, enter 00:00. Range is 0 (disabled) to 5999 seconds
(99 minutes, 59 seconds). The default is 0.
SendChar 1 and SendChar2
Enter up to two characters in hexadecimal representation
SendChar 1 (0) ?
SendChar 2 (0) ?
If the unit receives a character on the serial line that matches one of these characters, it
sends the character immediately, along with any awaiting characters, to the TCP connection.
This action minimizes the response time for specific protocol characters on the serial line
(for example, ETX, EOT). Setting the first SendChar to 00 disables the recognition of the
characters. Alternatively, the unit can interpret two characters as a sequence.
Telnet Terminal Type

111
This parameter displays only if the terminal type option is enabled in Disconnect Mode. If
this option is enabled, use the terminal name for the Telnet terminal type. Enter only one
name.
If the terminal type option is enabled, the unit also reacts to the EOR (end of record) and
binary options, which can be used for applications such as terminal emulation to
UNIX hosts.
Channel (Port) Password
This parameter appears only if the channel (port) password option is enabled in Disconnect
Mode. If the option is enabled, set a password on the serial port.
IV.5.4 Email Configuration
Reserved
IV.5.5 WLAN Settings

Without adequate protection, a wireless LAN is susceptible to access by unauthorized
users. As such, WFS802b includes the Wired Equivalent Privacy (WEP) encryption
standard as an additional means of security.
To modify WLAN and WEP settings, select 4 WLAN from the Change Setup menu.
Enable WLAN
The current value is displayed in parentheses. By default, WLAN is enabled on
WFS802b.
Enable WLAN (Y) ?
Find Network Name
Enter the name of the network in which the WFS802b unit resides. The current value is
displayed in parentheses.
Find network name (DRI_IBSS) ?
Enable Ad Hoc Network Creation
The current value is displayed in parentheses. By default, Ad Hoc network creation is enabled
on WFS802b.
Enable Ad Hoc network creation (Y) ?
Name (DRI_IBSS) ?
Country 0=US, 1=FR, 2=JP, 3=Other (0) ?
Channel (11) ?
Enter Y to enable Ad Hoc network creation and display configurable parameters:
1. At the Name prompt, enter the network name as text and hit Enter. The default name
displays in parentheses.
2. Select a Country by entering 0, 1, or 3. By default, 0 (United States) is selected. Press
Enter.
3. At the Channel prompt, enter the WFS802b's channel setting.

Copyright © Dr Robot Inc. 2006 112

Security
As an additional security measure, enable WEP on the WFS802b. The current value is
displayed in parentheses. By default, WEP is disabled on WFS802b.
Security 0=none, 1=WEP (0) ?
Data Rate
WFS802b permits the control of the transmission rate. The default is a data rate up to
11Mbps. The current value is displayed in parentheses.
Data rate, Only : 0=1, 1=2, 2=5.5, 3=11 Mbps or
Up to: 4=2, 5=5.5, 6=11 Mbps (6) ?
Power Management
Power management reduces the overall power consumption of the WFS802b unit. Enabling
power management increases the response time. The current value is displayed in parentheses.
Enable power management (N) ?

IV.5.6 Expert Settings

Note: Change these settings via Telnet or serial connections only.

Caution: Only an expert should change these parameters. These changes hold serious consequences.

TCP Keepalive Time

TCP Keepalive time defines how many seconds the unit waits during a silent connection before

checking whether the currently connected network device is still on the network. If the unit

does not receive a response, it drops that connection.

TCP Keepalive time in s (1s – 65s; 0s=disable): (45)?

ARP Cache Timeout

When the unit communicates with another device on the network, it adds an entry into its ARP

table. ARP Cache timeout defines the number of seconds (1-600) the unit waits before timing

out this table.

ARP Cache timeout in s (1s – 65s; 0s=disable): (600)?

IV.5.7 Security Settings

Note: As recommended, set security over the dedicated network or over the serial setup. If the parameters are set over the network (Telnet 9999), someone else could capture these settings.

Caution: Disabling both Telnet Setup and Port 77FE prevent users from accessing the setup menu from the network.

113

Disable SNMP

Reserved

SNMP Community Name

Reserved

Disable Telnet Setup

Note: If this option is disabled, note that disabling both Telnet Setup and Port 77FE

prevents users from accessing the setup menu from the network.

This setting defaults to the N (No) option. The Y (Yes) option disables access to Setup

Mode by Telnet (port 9999). It only allows access locally via the web pages and the serial

port of the unit.

Disable Telnet Setup (N) ?

Disable TFTP Firmware Upgrade

Reserved

Disable Port 77FE (Hex)

Reserved

Disable Web Server

The Y (Yes) option disables the web server. This setting defaults to the N (option).

Disable Web Server (N) ?

Disable Web Setup

The Y (Yes) option disables configuration via the Web-Manager. This setting defaults to the N
(option).
Disable Web Setup (N) ?
Disable ECHO Ports
This setting controls whether the serial port echoes characters it receives. The current value
is displayed in parent.
Disable ECHO ports (Y) ?
Enable Enhanced Password
This setting defaults to the N (option), which permits a 4-character password protecting Setup
Mode by means of Telnet and web pages.
Enable Enhanced Password (Y) ?
The Y (Yes) option allows an extended security password of 16-characters for

protecting Telnet access.
Disable Port 77F0 (Hex)
Port 77F0 is a setting that allows a custom application to query or set the eleven
WFS802b configurable pins when they are functioning as general purpose I/O (GPIO).
Disable this capability, if desired, for security purposes.
Disable Port 77F0h ?
The default setting, the N (No) option, enables GPIO control. The Y (Yes) option disables the GPIO control interface.
IV.5.8 Factory Defaults
Select 7 Factory Defaults from the Change Setup menu to reset the unit's Channel 1
configuration, Channel 2 configuration, E-mail settings, and Expert settings to the factory
default settings. The server configuration settings for IP address, gateway IP address, and
netmask remain unchanged. The configurable pins' settings also remain unchanged. The
specific settings that this option changes are listed below:
Channel 1 Configuration
Baudrate 9600
I/F Mode 4C (1 stop bit, no parity, 8 bit, RS-232C)
Port No 10001
Connect Mode C0 (always accept incoming connection; no
active connection startup)
Hostlist Retry Counter 3
Hostlist Retry Timeout 250 (msec)
Send Character 0x0D (CR)
All other parameters 0
Channel 2 Configuration
Baudrate 9600
I/F Mode 4C (1 stop bit, no parity, 8 bit, RS-232C)
Port No 10002

Connect Mode C0 (always accept incoming connection; no
active connection startup)
Hostlist Retry Counter 3
Hostlist Retry Timeout 250 (msec)
Send Character 0x0D (CR)
All other parameters 0
WLAN Settings
Enable WLAN (Y) Yes
Find Network Name LTRX_IBSS
Enable Ad Hoc Network Creation (Y) Yes

115

Name LTRX_IBSS
Country (0) United States
Channel 11
Security (0) None
Data Rate 11Mbps
Expert Settings
TCP keepalive 45 (seconds)
ARP cache timeout 600 (seconds)
Security Settings
Disable SNMP (N) No
SNMP Community Name public
Disable Telnet Setup (N) No
Disable TFTP Firmware Update (N) No
Disable Port 77FEh (N) No
Disable Web Server (N) No
Disable ECHO ports (Y) Yes
Enable Enhanced password (N) No
Disable Port 77F0h (N) No
Email Settings
Trigger Priority L
Min. notification interval 1 second
All other parameters 0 (e.g. Email notification and triggers are
disabled)
IV.5.9 Exit Configuration Mode
To exit setup mode:
. Select option 9 Save and exit from the Change Setup menu to save all changes
and reboot the device. All values are stored in nonvolatile memory.
or
. Select option 8 Exit without save from the Change Setup menu to exit the
configuration mode without saving any changes or rebooting.

116

IV.6 Configuration using Web-Manager

This chapter describes how to configure the WFS802b using Web-Manager,
DrRobot's
browser-based configuration tool. The unit's configuration is stored in nonvolatile
memory

and is retained without power. The unit performs a reset after the configuration is changed
and stored.
This chapter includes the following topics:
. Accessing WFS802b using Web-Manager
. Network Configuration
. Server Configuration
. Host List Configuration
. Channel 1 and Channel 2 Configuration
. WLAN Configuration
. OEM Pin Configuration
. Updating Settings
IV.6.1 Accessing WFS802b using Web-Manager
Follow the instructions to configure the unit's MAC address
1. Using Ser i a l Port to s e t the IP. For more information on the Serial
Port Access, see "Server" on page 98.
2. Configure WLAN parameter. For more information on the Serial Port Access,
see "WLAN" on page 111.
Figure IV.1 Web-Manager
The main menu is displayed in the left side of the Web-Manager window.
Note: Alternatively, access the WFS802b's Web-Manager if it is
connected to the network by entering its IP address in a web browser.
Copyright © Dr Robot Inc. 2006 117
IV.6.2 Network Configuration
The unit's network values display upon selecting Network from the main menu. The
following sections describe the configurable parameters within the Network configuration
menu.
Note: The IP address is assigned via DHCP (on DHCP-enabled networks). Assign
a static IP address only if necessary.
Figure IV.2 Network Settings
Automatic IP Address Configuration
To automatically assign an IP address and its network configuration:
1. Click Network from the main menu.
2. Select Obtain IP address automatically.
3. Enter the following (as necessary):
Disable BOOTP Leave the checkbox empty to enable Bootstrap Protocol(BOOTP).
The BOOTP server automatically assigns the IP address from a pool of
addresses.
Disable DHCP Leave the checkbox empty to enable Dynamic Host Configuration
Protocol (DHCP). DHCP automatically assigns a leased IP address to
the WFS802b unit.
Disable Auto-IP The WFS802b generates an IP in the 169.254.x.x address range with
a Class B subnet. Select the checkbox to disable this feature.
Host Name Enter the name of the host on the network.

Note: Disabling BOOTP, DHCP, and Auto-IP (i.e. all three checkboxes) is not advised as the only available IP assignment method will then be ARP or serial port.

4. Click the OK button when finished.

Static IP Address Configuration

To manually assign an IP address and its network configuration:

Copyright © Dr Robot Inc. 2006 118

1. Click Network from the main menu.

2. Select Use the following IP configuration.

3. Enter the following (as necessary):

IP Address If DHCP is not used to assign IP addresses, enter it manually. The IP address must be set to a unique value in the network.

Subnet Mask A subnet mask defines the number of bits taken from the IP address that are assigned for the host part.

Default Gateway The gateway address, or router, allows communication to other LAN

segments. The gateway address should be the IP address of the router connected to the same LAN segment as the unit. The gateway address must be within the local network.

4. Click the OK button when finished.

IV.6.3 Server Configuration

The unit's server values display upon selecting Server from the main menu. The following sections describe the configurable parameters within the Server configuration

menu.

Figure IV.3 Server Settings

To configure the WFS802b's device server settings:

1. Click Server from the main menu.

2. Configure or modify the following fields:

Server Configuration

Telnet Password Enter the password required for Telnet access.

Retype Password Re-enter the password required for Telnet access.

Advanced

ARP Cache

Timeout

When the unit communicates with another device on the network, it

adds an entry into its ARP table. ARP Cache timeout defines the

number of seconds (1-600) before it refreshes this table.

TCP Keepalive TCP Keepalive time defines how many seconds the unit waits during

Copyright © Dr Robot Inc. 2006 119

an inactive connection before checking its status. If the unit does not

receive a response, it drops that connection. Enter a value between 0

and 60 seconds. 0 disables keepalive.

IV.6.4 Host List Configuration

The WFS802b scrolls through the host list until it connects to a device listed in the host list

table. After a successful connection, the unit stops trying to connect to any others. If this

connection fails, the unit continues to scroll through the table until the next successful

connection.

The host list supports a minimum of 1 and a maximum of 12 entries. Each entry contains

an IP address and a port number.

Note: The host list is disabled for Manual and Modem Mode. The unit will not accept

a data connection from a remote device when the hostlist option is enabled.

To configure the WFS802b's host list:

1. From the main menu, click the Hostlist tab.

Figure IV.4 Hostlist Settings

2. Enter or modify the following fields from the Hostlist Settings window:

Retry Settings

Retry Counter Enter the value for the number of times the WFS802b should attempt to retry connecting to the host list.

Retry Timeout Enter the duration (in seconds) the WFS802b should abandon attempting a connection to the host list.

Host Information

Host Address Enter or modify the host's IP address.

Port Enter the target port number.

Copyright © Dr Robot Inc. 2006 120

IV.6.5 Channel 1 and Channel 2 Configuration

Channel 1 and Channel 2 configurations define how the serial ports respond to network and serial communication.

Serial Settings

To configure a channel's serial settings:

1. From the main menu, click Serial Settings for either Channel 1 or Channel 2 to display

the Serial Settings page for the selected channel.

Figure IV.5 Channel Serial Settings

2. In the available fields, enter the following information:

Channel 1

Disable Serial Port Available on Channel 1 settings only. When selected, disables communication through the serial port.

Port Settings

Protocol Select the protocol type from the pull down menu for the selected channel.

Flow Control Flow control manages data flow between devices in a network to ensure it is processed efficiently. Too much data arriving before a

device is prepared to manage it causes lost or retransmitted data.

Baud Rate The unit and attached serial device, such as a modem, must agree on

Copyright © Dr Robot Inc. 2006 121

a speed or baud rate to use for the serial connection. Valid baud rates

are 300, 600, 1200, 2400, 4800, 9600 (default), 19200, 38400,

57600, 115200, 230400, 460800, or 921600.

Data Bits Indicates the number of bits in a transmitted data package.

Parity Checks for the parity bit. The default is None.

Stop Bits The stop bit follows the data and parity bits in serial communication. It indicates the end of transmission.

Port Settings

Enable Packing Select the checkbox to enable packing on the WFS802b. Two firmware-selectable packing algorithms define how and when packets are sent to the network. The standard algorithm is optimized for applications in which the unit is used in a local environment, allowing for very small delays for single characters, while keeping the packet count low. The alternate packing algorithm minimizes the packet count on the network and is especially useful in applications in a routed Wide Area Network (WAN). Adjusting parameters in this mode can economize the network data stream.

Idle Gap Time Select the maximum time for inactivity. The default time is 12 milliseconds.

Match 2 Byte

Sequence

Use to indicate the end of a series of data to be sent as one group. The sequence must occur sequentially to indicate to the WFS802b end of the data collection.

Match Bytes Use to indicate the end of a series of data to be sent as one group. Set this value to 00 if specific functions are not needed.

Send Frame Only After the detection of the byte sequence, indicates whether to send

the data frame or the entire buffer. Select True to send only the data frame.

Send Trailing

Bytes

Select the number of bytes to send after the end-of-sequence characters.

Flush Input Buffer (Serial to Network)

With Active

Connect

Select Yes to clear the input buffer with a connection that is initiated from the device to the network.

With Passive

Connect

Select Yes to clear the input buffer with a connection initiated from the network to the device.

At Time of

Disconnect

Select Yes to clear the input buffer when the network connection to or from the device is disconnected.

Flush Output Buffer (Network to Serial)

With Active

Connect
Select Yes to clear the output buffer with a connection that is initiated
from the device to the network.
With Passive
Connect
Select Yes to clear the output buffer with a connection initiated from
the network to the device.
At Time of
Disconnect
Select Yes to clear the output buffer when the network connection to
or from the device is disconnected.
Connection Settings - TCP
To configure a channel's TCP settings:
1. From the main menu, click Connection for either Channel 1 or Channel 2 to
display

122

the Connection Settings page for the selected channel.
2. In the available fields, enter the following information:
Connect Protocol
Protocol Select TCP from the pull down menu.
Figure IV.6 TCP Connection Settings
3. In the available fields, enter the following information:
Connect Mode: Passive Connection
Accept Incoming Select Yes to accept incoming connections.
Password
Required
Determines whether a password is required for an incoming passive
connection. Field is not available when a password is set for Telnet
mode.
Password If Password Required was set to Yes, enter the password for passive
connections.
Connect Mode: Active Connection Port Settings
Active Connect Select None to disable Active Connect. Otherwise, indicate the
connection type from the available list. Never Accept Incoming rejects
all external connection attempts. Accept with modem_control_in
Active accepts external connection requests only when the
modem_control_in input is asserted. Cannot be used with Modem Mode.
Always Accept accepts any incoming connection when a connection is

123

not already established.
Start Character If Active Connect is set to With Start Character, enter the start
character in this field.
Modem Mode Indicates the on-screen response type when in Modem Mode
(if enabled).
Endpoint Configuration
Local Port Enter the local port number.
Auto increment

local port number
Select to auto-increment the local port number for new outgoing
connections. The range of auto-incremented port numbers is 50,000
to 59,999 and loops back to the beginning when the maximum range is
reached.
Remote Port Enter the remote port number.
Remote Host the IP address of the remote device.

Common Options

Telnet Mode This field is available for configuration only when Active Connection is
not set to None. Select Enable to permit Telnet communication to the
WFS802b unit
Terminal Name This field is available for configuration only when Telnet Mode is set
to
Enable.
Use the terminal name for the Telnet terminal type. Enter only one
name. When this option is enabled, the unit also reacts to the EOR
(end of record) and binary options, which can be used for applications
such as terminal emulation to IBM hosts.
Connect Response A single character is transmitted to the serial port when there is a
change in connection state. Default setting is None.
Use Hostlist If this option is set to True, the device server scrolls through the host
list until it connects to a device listed in the host list table. Once it
connects, the unit stops trying to connect to any others. If this
connection fails, the unit continues to scroll through the table until it is
able to connect to another IP in the host list.
The host list is disabled for Manual Mode and for Modem Mode. The
unit will not accept a data connection from a remote device when the
host list option is enabled.
LED Select Blink for the status LEDs to blink upon connection or
None for no LED output.

Disconnect Mode

On Mdm_Ctrl_In
Drop
Set to Yes for the network connection to or from the serial port to drop
when modem_control_in transitions from a high state to a low state.
Hard Disconnect When set to Yes, the TCP connection closes even if the remote site
does not acknowledge the disconnect request.
With EOT Choose Yes to drop the connection when Ctrl-D or Hex 04 is detected.
Both Telnet mode and Disconnect with EOT must be enabled for
Disconnect with EOT to function properly. Ctrl D is only detected going
from the serial port to the network.
Inactivity Timeout Use this parameter to set an inactivity timeout. The unit drops the
connection if there is no activity on the serial line before the set time
expires. Enter time in the format mm:ss, where m is the number of
minutes and s is the number of seconds. To disable the inactivity
timeout, enter 00:00.

Connection Settings - UDP

To configure a channel's UDP settings:

1. From the main menu, click Connection for either Channel 1 or Channel 2 to display

the Connection Settings page for the selected channel.

2. In the available fields, enter the following information:

Connect Protocol

Protocol Select UDP from the pull down menu.

Figure IV.7 UDP Connection Settings

Datagram Mode

Datagram Type Configures remote IP or network broadcast address and the remote port. Enter 01 for directed or broadcast UDP.

Accept Incoming Select Yes to accept incoming UDP datagrams.

Endpoint Configuration

Local Port Enter the local port number.

Remote Port Enter the port number of the remote device.

Remote Host Enter the IP address of the remote device.

Change Address

Table

Field enabled when Datagram Type is set to FD. Enter values between

1-255 to identify units on the local network of device servers.

Copyright © Dr Robot Inc. 2006 125

IV.6.6 WLAN Configuration

Without adequate protection, a wireless LAN is susceptible to access by unauthorized

users. As such, WFS802b includes the Wired Equivalent Privacy (WEP) encryption standard as an additional means of security.

To configure the WFS802b's WLAN settings:

1. Select WLAN from the main menu to open the WLAN Settings window.

Figure IV.8 WLAN Settings

2. Enter or modify the following fields:

Network Interface Use the pull down menu to select a WLAN interface or an Ethernet interface.

Network Name Enter the name of the network where the WFS802b is located.

Ad Hoc Settings Ad

Hoc Network

Creation

Select the checkbox when using a client (such as a wireless card) to communicate with the WFS802b instead of an Access Point.

Ad Hoc Network

Name

Enter the network name for the Ad Hoc network.

Ad Hoc Network Select from the pull down menu the radio channel for the Ad Hoc

Copyright © Dr Robot Inc. 2006 126

Channel network. The default value is 11.

Ad Hoc Country From the pull down menu, select a country for the Ad Hoc network. The

default is United States.

Wireless Network Security

Security As an additional security measure, enable WEP on the WFS802b. By default, WEP is disabled on WFS802b.

Authentication Select an authentication scheme (None or Shared) from the drop down

menu.

Encryption Select the encryption type from the pull down menu. 128 bits is the default encryption.

Encryption Key Field is enabled when WEP is selected as the Security type. Enter the

Encryption Key in hexadecimal value

Advanced Settings

Data Rate WFS802b permits the control of the transmission rate. Select the data rate (in Mbps) from the pull down menu.

Radio Power

Management

Power management reduces the overall power consumption of the

WFS802b unit. Selecting Enable increases the response time.

IV.6.7 OEM Pin Configuration

There are 11 configurable hardware pins on the WFS802b unit. For each pin, configure the

pin function, communication direction, and its activity level.

To configure the WFS802b's OEM Configurable Pins:

1. Click Configurable Pins from the main menu to open the Configurable Pins window.

Figure IV.9 Configurable Pins Settings

Copyright © Dr Robot Inc. 2006 127

2. Configure or modify the following fields for each pin:

Function From the pull down menu, select the purpose of the specified

pin.

Active Level Select the signal active level (Low or High).

Direction Select whether the pin inputs or outputs.

IV.6.8 Updating Settings

Click the Apply Settings button from the main menu to save and apply the configuration changes.

Copyright © Dr Robot Inc. 2006 128

V. MCB3100 WiRobot Serial Bluetooth Wireless Module

V.1 Introduction

The MCB3100 Serial Bluetooth Wireless Module is a class II Bluetooth module with on-board

communication stack. This device can be plugged into any UART or RS232 compatible serial port

(requires MCR3210P RS232 Interface Module for signal change) on almost any devices without

needing to install drivers. It can be considered as a "wireless cable" to replacement for any RS232

serial cable and can be used in applications for wireless audio, still image, sensing and control data
communications.

V.1.1 Features

. Class 2 Bluetooth operation
. On-board communication stack
. Effective range: 15 meters indoor, 45 meters outdoor
. Support UART data rate: 921.6/460.8/115.2 kbps
. Plug-and-play in the WiRobot system

V.1.2 Applications

. Robotic systems: both run-time and development-stage communication
. General-purpose wireless data communication

V.2 Operations

V.2.1 Theory of Operation

The MCB3100 Serial Bluetooth Wireless Module is designed to run as part of the WiRobot system. It
can be directly plugged on to the PMB5010 Robot Multimedia Controller board or the PMS5005
Robot Sensing and Motion Controller board. When connected to the MCR3210P RS232 Interface
board through a cable, it can also serve as wireless links for any systems that have a standard RS232
interface (PC for example). By default, the UART data rate is pre-programmed to 115.2kbps with
hardware flow control and can be adjusted according to the customer's preferred setting at the time of
purchase. All wireless firmware has been embedded into the module and user simply needs to issue a
"CONNECT" command to the MCB3100 in order to establish a connection with another MCB3100
wireless module.

V.2.2 Configuration (PC-PC for Sample)

1. Connect MCB3100 Bluetooth modules and MCR3210P RS232 interface modules with
8pin flat cable (provided by Dr Robot), red line should be first Pin.
2. Use null-modem RS232 cable connect MCR3210P RS232 interface module to PC serial
port (such as Com1), and use one USB cable to connect MCR3210P RS232 interface module
to one USB port. It just provides power to RS232 module.
3. Same connection to another PC.
4. Launch Hyper Terminal program, choose a port (just connected on step2), and set port
settings as:
Bits per second: 115200,
Data bits: 8,
Parity: none,

129
Stop bits: 1,
Flow control: hardware.
5. Plug USB cable again to reset Bluetooth module, you can get a message from
HyperTerminal, AT-ZV -CommandMode-, AT-ZV BDAddress xxxxxxxxxxxx.
6. At another PC, you need launch same configuration of HyperTerminal. Get same
result, but
BDaddress should be different, it just like 00043e01xxxx.
7. At one PC, type command AT+ZV SPPConnect xxxxxxxxxxxx. Here xxxxxxxxxxxx
is
another Bluetooth module'BDAddress. If you can get AT-ZV ConnectionUp, AT-ZV -
BypassMode-, the connection between PCs is setup. You can type anything or
transfer a file
to another PC.
The connection command is AT+ZV SPPConnect xxxxxxxxxxxx.
The change baudrate command is AT+ZV ChangeBaud 460800.

V.3 Connections
V.3.1 Board Structure
Figure V.1 illustrates the structure of the board
Figure V.1 MCB3100 Structure
V.3.2 Connector Description
The MCB3100 is connected to WiRobot system via an 8-pin 2.54 mm-pitch single
row connector:
Table V.1 Connectors
Pin Name Function
1 VCC +3.3 V
2 TXD Data transmitting
3 RXD Data receiving
4 CTS Clear to send
5 RTS Request to send
6 GND Power supply ground
7 COMRST Reserved
8 BTIN Reserved

130

V.4 Specifications
Table V.2 MCB3100 Specification
Parameter Conditions MIN TYP MAX Unit
Power Supply Voltage (VCC) 3.0 3.3 3.6 V
Signal Pin Voltage 3.3 V
RF Frequency 2400 2483.5 MHz
Antenna Load 50 Ohm
Low-level Input Voltage VCC = 3.3V 0.8 V
High-level Input Voltage VCC = 3.3V 2.0 V
Low-level Output Voltage VCC = 3.3V, IOL =
2mA
0.4 V
High-level Output Voltage VCC = 3.3V, IOH =
2mA
2.4 V

Low-level Output Current VCC = 3.3V, VOL =
0.4V
2.2 mA
High-level Output Current VCC = 3.3V, VOH =
2.4V
3.1 mA
Board Size 30 x 40 mm x mm

VI. MAC5310 Audio Codec and Audio Power Amplifier Module

VI.1 Introduction

The MAC5310 Audio Codec and Audio Power Amplifier Module can be used as audio input/output
interface in the WiRobot system by plugging into the PMB5010 Multimedia Controller board. The onboard
codec provides high resolution signal conversion from digital-to-analog (D/A) and from analogto-
digital (A/D) using over-sampling sigma-delta technology. With the on-board audio output power
amplifier and the microphone preamp in the codec, the external speaker and microphone can be
directly connected to the MAC5310 board.

VI.1.1 Features

. 16-bit over-sampling sigma-delta A/D, D/A converter
. Maximum output conversion rate:
- 16 ksps with on-chip FIR filter
- 64 ksps with FIR bypassed
. Codec built-in FIR produces 84-db SNR for ADC and 85-db SNR for DAC over 11-kHz BW
. 2s-complement data format
. Codec built-in functions including PGA, anti-aliasing analog filter, and operational amplifiers
for general-purpose interface (such as MIC interface and hybrid interface)
. On-board audio output power amplifier can support up to 1.5 W power to the external speaker
. On-board oscillator
. Plug-and-play in the WiRobot system

VI.1.2 Applications

. Audio input/output for robotic systems
. Voice and speech recognition
. Voice and audio playback

VI.2 Operations

VI.2.1 Theory of Operation

The MAC5310 Module is designed to run as part of WiRobot system. It can be directly plugged on to
the PMB5010 Robot Multimedia Controller board. No configuration procedure is needed. Once

connected, the PMB5010 on-board firmware and the audio input/output device driver will take care
of the low level operations of the voice/speech capturing and audio output.

VI.3 Connections

VI.3.1 Board Structure

Figure VI.1 shows the board structure, locations and functions of the connectors on the MAC5310
module board.

Figure VI.1 MAC5310 Connector Locations

Microphone
Connector

Speaker
Connector

System Interface
Connector

132

VI.3.2 Connector Description

The definitions of the MAC5310 connector signals are listed in the following tables.

Table VI.1 Speaker Connector SPEAKER

Pin Name Function
1 SO1 Speaker output 1
2 SO2 Speaker output 2

Table VI.2 Microphone Connector MICROPHONE

Pin Name Function
1 NC No connection
2 MIP Microphone input +
3 MIM Microphone input -

Table VI.3 System Interface Connector CODEC

Pin Name Function
1 DOUT Data output
2 VCC5 + 5.0V
3 FS Frame sync
4, 6 GND Power supply ground
5 DIN Data input
7 SCK Shift clock
8 MCK NC
9 RESET Reset input
10 PDN Power down input
11 FC Request input for secondary communication
12 VCC3 + 3.3V

VI.4 Specifications

Table VI.4 MAC5310 Specification

Parameter Conditions MIN TYP MAX Unit
Power Supply Voltage VCC5 4.5 5.0 5.5 V
Power Supply Voltage VCC3 3.0 3.3 3.6 V
THD = 0.5%(max),
f = 1 kHz,
RL = 8 Ohm
1.0
Output Power

THD + N = 0.5%,
f = 1 kHz,
1.5
W

133

RL = 8 Ohm
Analog input voltage,
peak-to-peak
VCC3 = 3.3 V
2 V
ADC or DAC conversion
rate
16 kHz
On-board oscillator 8.1920 MHz
Board Size 30 x 40 mmxmm
Note:
THD + N = Total Harmonic Distortion + Noise

134

VII. DUR5200 Ultrasonic Range Sensor Module

VII.1 Introduction

The DUR5200 Ultrasonic Range Sensor Module can detect the range information from 4 cm to 340
cm. It transmits an ultrasonic "ping" when instructed by your program and returns a signal when it
receives an echo. The distance data is precisely presented by the time interval between the instant
when the measurement is enabled and the instant when the echo signal is received. There is an onboard
oscillator that significantly reduces the burden of the controller to transmit signal with the
required frequency. The DUR5200 is very easy to use and can be simply plug-in to the WiRobot
PMS5005 Sensing and Motion Controller board. The PMS5005 (shipped with WiRobot SDK for PC)
will handle the critical timing functions and distance calculation.

VII.1.1 Features

. On-board oscillator
. 4 cm to 340 cm effective range
. 40 KHz working frequency
. Plug-and-play in the WiRobot system

VII.1.2 Applications

. Mobile robot environment map building
. Obstacle detection, collision avoidance
. Robot range finder
. General-purpose distance detection

VII.2 Operations

VII.2.1 Theory of Operation

The DUR5200 works by means of ultrasonic wave (40 KHz) that is beyond the range of human
hearing. Sound wave propagation speed in the air is 343.5 m/s, when the ambient air temperature is
20°C. By detecting the propagation time of the sonic wave between the sensor and the object (if any)
in the path of the wave, the controller is able to calculate the distance.
VII.2.2 Running as Part of WiRobot System
When using the DUR5200 with the WiRobot system, user can simply connect the module to one of
the ultrasonic sensor module connectors on the PMS5005 controller board and the PMS5005 builtin
sensor device driver will take care of the range data acquisition. Users can simply call a function
offered by the WiRobot SDK software on PC (requires Microsoft platform) or send a data request
packet (platform independent) to obtain the data. Note that DUR5200 can measure from 4 to 255 cm
in WiRobot system since PMS5005 only uses one byte to represent the distance.
The sound wave propagation speed in the air depends on the temperature. If you also got the
temperature sensor module in your WiRobot system, you can measure the distance more precisely by
adding up the temperature compensation. The sound wave propagation speed (v) with temperature
compensation can be calculated by the following formula:
$v = 331.5 + 0.6 * T$ [m/sec]
where T is the air temperature (°C).
VII.2.3 Running as a General Purpose Ultrasonic Range Sensor Module

135

When using the DUR5200 with the third party controller, the power supply and the input/output
signals should be connected properly (please refer to Section VII.3 Connections). The basic operation
is illustrated in Figure VII.1.
Range measurement starts from the rising edge of TE. Then the controller set TE to low (logic 0) after
t1 (250 µsec). The controller should measure the time interval td from the rising edge of TE to the first
rising edge of RS, which is the returned sound wave. td is equal to two times of the traveling time
between the sensor to the object (transmitting and echoing). The time period between two
measurements should be no less than 20 msec. The minimum distance that the DUR5200 can
measure is 4 cm. This means that if the range is less than 4 cm, it will be reported as 4 cm.
Figure VII.1 Basic Operation Timing

The distance to object (in meter) can be obtained as follows:
Distance to object (in meter) = td (in second) * v (in meter/second) / 2

VII.3 Connections

VII.3.1 Board Structure
Figure VII.1 illustrates the structure of the board.
Figure VII.1 DUR5200 Structure

VII.3.2 Connector Description
The DUR5200 can be connected to the controller system via a 4-pin 2.54 mm-pitch single row
connector:
Table VII.1 Ultrasonic Range Sensor Connectors
Pin Name Function
1 Vcc Positive power source, 5 V DC
2 RS Ultrasonic echo receiving signal, active rising edge output
3 TE Ultrasonic transmitting enable, active high input

4 GND Power supply ground

VII.4 Specifications
Table VII.2 DUR5200 Specification
Parameter Conditions MIN TYP MAX Unit
Power Supply Voltage (Vcc) 4.9 5.0 5.1 V
Current Consumption Vcc = 5 V 45 50 mA
Working Frequency 40 KHz
Effective Range 25°C 4 340 cm
Directivity ±30 °
Board Size 30 x 48 mm x mm

VIII. DTA5102 Two-Axis Tilt and Acceleration Sensor Module

VIII.1 Introduction
The DTA5102 Tilt and Acceleration Sensor Module is capable of measuring both the static
acceleration (tilt or orientation) caused by the Earth's gravity or the shock caused by an impact. The
module uses a CMOS micro-machined accelerometer IC combined with on-board low-pass-filters and
signal amplifiers. The measurement range of the DTA5102 module is ±1 g.

VIII.1.1 Features
. ±1 g tilt and shock detection
. On-board low-pass-filters and signal amplifiers
. Linear output
. Plug-and-play in the WiRobot system

VIII.1.2 Applications
. Robotic application
. Vibration monitoring
. Impact/Acceleration measurement
. Tilt, orientation and posture measurement
. Handheld appliance control

. Virtual reality input devices
. Electronic diagnostic system

VIII.2 Operations

VIII.2.1 Theory of Operation

The structure of the micro-machined accelerometer is shown as Figure VIII.1. The sensing cell is a

micro-machined variable capacitive device. The center plate moves with the acceleration and hence

the values of the capacitors will change according to the distance between the plates. The change of

the value is then measured, converted, amplified and outputted.

Figure VIII.1 Equivalent accelerometer model

The acceleration sensing directions of the DTA5102 are shown as Figure VIII.2. The output signals

are basically consisting of static or low frequency data of tilt or orientation information and high

frequency data of vibration or impact information. Either analog or digital filter or both can be used to

extract relevant data for specific applications.

+ Y
- Y
- X + X

Figure VIII.2 Acceleration sensing directions

To measure the tilt or orientation of an object, the DTA5102 should be mounted in such a way that the

axes of sensitivity are parallel to the surface of the Earth. In this configuration, the relationship

between the output voltage and the tilt angle of each axis is shown by the following equation

$$V_{OUT} = V_{ZEROG} + (\frac{\Delta V}{\Delta G} \times G \times \sin\theta)$$

Where

V_{OUT} = Output voltage of each axis

V_{ZEROG} = Voltage at zero g

$\frac{\Delta V}{\Delta G}$ = Sensitivity

G = Earth's gravity (~9.81)

θ = Tilting angle

$\frac{\Delta V}{\Delta G}$ can be obtained by experiment by placing the sensor level (so that the gravity vector is

perpendicular to the measured sensor axis) to take the V_{ZEROG} reading. Then, you should rotate the

sensor so that the gravity vector is parallel with the measured axis and take the V_{ONEG} reading. The

equation then can be rewritten as:

$V_{OUT} = V_{ZEROG} + [(V_{ONEG} - V_{ZEROG}) \times \sin\theta]$

The tilting angle then can be calculated by ArcSin () function or by Taylor polynomial approximation

$\theta = \sin_{-1}(z) = z +$

40

3

6

$z_3 + z_5 + ... z < 1$

To detect the high frequency data, the sampling rate must be at least twice of the signal frequency

according to Nyquist Sampling Criterion. As a rule of thumb, using 5 to 10 time higher sampling rate

will get good results for data recovering. Using some digital filter may require even higher sampling

rate. However, sampling rate higher than 20 KHz is generally not recommended for the DTA5102

module. Also, be aware of the signal aliasing effects.

The relationship between the output voltage and the acceleration in each axis direction is shown by

the following equation

$V_{OUT} = V_{ZEROG} + ($

G

V

Δ

Δ

$\times$ Acc)

Where

Acc = value of the acceleration.

VIII.2.2 Running as part of WiRobot System

When using the DTA5102 with the WiRobot system, user can simply connect the module to the tilt

sensor module connector on the PMS5005 controller board and the PMS5005 built-in sensor device

driver will take care of the data acquisition. Users can simply call a function offered by the WiRobot

SDK software on PC (requires Microsoft platform) or send a data request packet (platform

independent) to obtain the data.

VIII.2.3 Running as a General Purpose Tilt and Acceleration Sensor Module

When using the DTA5102 with the third party controller, the power supply and the input/output

signals should be connected properly (please refer to Section VIII.3). The controller can get the tilt

and acceleration data via an analog to digital converter. The value of the angle or the acceleration can

be calculated according to the equations in Section VIII.1.

For premium performance, several cautions need to be taken into account when operating the system:

. The power supply voltage should be 5 VDC nominal.

. The length of the cable connecting the DTA5102 and controller should be as short as possible.

. The DTA5102 module and the controller should not be in a high current path.

. If using switching power supply, be aware of the switching frequency may interfere with the

DTA5102 module.

VIII.3 Connections

VIII.3.1 Board Structure

Figure VIII.3 shows the board structure.

Figure VIII.3 DTA5102 Structure

VIII.3.2 Connector Description

The DTA5102 can be connected to the controller system via a 4-pin 2.54 mm-pitch single row

connector:

Table VIII.1 Tilt and Acceleration Sensor Connectors

Pin Name Function

1 VCC Positive power source, 5 V DC

2 YOUT Y direction signal, analog output

3 XOUT X direction signal, analog output

4 GND Power supply ground

VIII.4 Specifications

Table VIII.2 DTA5102 Specification

Parameter	Conditions	MIN	TYP	MAX	Unit
Power Supply Voltage	VCC	4.75	5.0	5.25	V
Current Consumption	VCC = 5 V		10	15	mA
Acceleration Measuring Range			± 1		g
Nonlinearity		-2.0		+ 2.0	%
Bandwidth Response			30		Hz
Board Size			30 x 48		mm x mm

IX. DHM5150 Human Motion Sensor Module

IX.1 Introduction

The DHM5150 Human Motion Sensor Module incorporates a pyroelectric infrared sensor to detect

infrared energy radiation from human body. The DHM5150 is able to detect human presence (like

security alarm) in the range up to 500 cm. With the use of two modules, human moving direction can

also be detected in the range up to 150 cm. Typical applications include a general-purpose security

alarm and human presence and motion sensing in a robot system.

IX.1.1 Features

. Human infrared radiation detection
. On-board signal conditioning
. Human presence detection up to 5 meters
. Human motion direction up to 1.5 meters
. Plug-and-play in the WiRobot system
. Applications
. Security alarm, human presence detection
. Human moving direction measurement
. Human-following devices
. Human avoidance and security robot

IX.2 Operations

IX.2.1 Theory of Operation

Infrared radiation exists in the electromagnetic spectrum at a wavelength that is longer than visible

light. Objects that generate heat also generate infrared radiation including animals and the human

body. The infrared radiation generated by human is strongest at a wavelength of 9.4 µm.

The sensor used in the DHM5150 module has two sensing elements. Together with a Fresnel lens, the

behavior of the sensor is shown in Figure IX.1.

Figure IX.1 Typical Sensor Behavior

IX.2.2 Running as Part of WiRobot System

142

When using the DHM5150 with the WiRobot system, user can simply connect the module to one of

the human sensor module connectors on the PMS5005 controller board and the PMS5005 built-in

sensor device driver will take care of the data acquisition. Users can simply call a function offered by

the WiRobot SDK software on PC (requires Microsoft platform) or send a data request packet

(platform independent) to obtain the data.

IX.2.3 Running as a General Purpose Human Motion Sensor Module

When using the DHM5150 with a third party controller, the power supply and the input/output signals

should be connected properly (please refer to Section IX.3). The controller can get the human

information data via an analog to digital converter.

There are analog outputs, one is the human motion (MS), and the other one is the human alarm (AS).

When no human presents, output voltage of MS and AS is 1.5V. The change of AS is basically 5 times

larger than MS due to the on-board amplifier. The closer the human and the faster the motion will

cause the longer voltage change shown in MS and AS.

Note that when using two sets of the DHM5150 for human motion detection, the moving direction

information can be identified by analyzing the pattern, timing and magnitude of two sensor output

signals.

IX.3 Connections

IX.3.1 Board Structure

Figure IX.2 illustrates the structure of the board.

Figure IX.2 DHM5150 Structure

IX.3.2 Connector Description

The DHM5150 can be connected to the controller system via a 4-pin 2.54 mm-pitch single row

connector:

Table IX.1 DHM5150 Connector

Pin Name Function

1 Vcc Positive power source, 5 V DC

2 MS Human motion signal, analog output

3 AS Human presence alarm, analog output

4 GND Power ground

IX.4 Specifications

Table IX.2 DHM5150 Specification

Parameter	Conditions	MIN	TYP	MAX	Unit
Power Supply Voltage	VCC	2.2	3.3	5.0	V
Current Consumption	VCC = 5 V		10		mA
Wavelength		5		14	µm
Human Motion Range			150		cm
Human Presence Range			500		cm
Directivity - Horizontal			100		°
Directivity - Vertical			60		°
Output Signal Voltage			VCC		V
Board Size			30 x 48		mm x mm

IX.5 Fresnel Lens

X. DAT5280 Ambient Temperature Sensor Module

X.1 Introduction

The DAT5280 Ambient Temperature Sensor Module uses high-precision CMOS temperature sensor

to generate linear voltage signal according to the ambient air temperature. With a temperature

coefficient of 25.5 mV/°C and nonlinearity of ±0.5 %, the DAT5280 is superior in the functionality

over conventional temperature sensors like thermometers. Typical applications include robotic system

and high-precision thermal control.

X.1.1 Features
. High linearity: $\pm$ 0.5%
. Standard output: 0 – 3.3V
. High temperature accuracy
. Plug-and-play in the WiRobot system

X.1.2 Applications
. Robotic system
. Temperature sensing
. Thermal control

X.2 Operations

X.2.1 Typical performance characteristics
The performance characteristics are shown in the figures X.1 and X.2. If needed, the temperature
accuracy error could be removed by calibration on the individual module basis.
Figure X.1 Output Voltage vs Ambient Temperature

145

Figure X.2 Accuracy vs Temperature

X.2.2 Running as part of WiRobot System
When using the DAT5280 with WiRobot system, user can simply connect the module to the
temperature sensor module connector on the PMS5005 controller board and the PMS5005 built-in
sensor device driver will take care of the data acquisition. Users can simply call a function offered by
the WiRobot SDK software on PC (requires Microsoft platform) or send a data request packet
(platform independent) to obtain the data.

X.2.3 Running as a General Purpose Temperature Sensor Module
When using the DAT5280 with a third party controller, the power supply and the input/output signals
should be connected properly (please refer to Section IX.3). The controller can get the temperature
data via an analog to digital converter. The temperature reading can then be calculated according to
Figure IX.1 or the following equation
TV (in V) = 0.92 + 0.0255 * (Temperature)

X.3 Connections

X.3.1 Board Structure
Figure X.3 illustrates the structure of the board.
Figure X.3 DAT5280 Structure

146

X.3.2 Connector Description
The DAT5280 can be connected to the controller system via a 3-pin 2.54 mm-pitch single row
connector.
Table X.1 Temperature Sensor Connectors
Pin Name Function

1 VCC Positive power source, 5 V DC
2 TV Temperature voltage, analog signal output
3 GND Power ground

X.4 Specifications

Table X.2 DAT5280 Specification
Parameter Conditions MIN TYP MAX Unit
Power Supply Voltage 4.9 5.0 5.1 V
Current Consumption Vcc = 5 V 5.0 mA
Nonlinearity -20°C ~ +80°C ±0.5 %
Temperature Sensitivity -20°C ~ +100°C 4.66 25.5 26.34 mV/°C
Board Size 30 x 24 mm x mm

147

Chapter V. TROUBLE SHOOTING

148

Q: How to start the robot?
A: Push the red button behind the robot's head.
Q: Why the robot does not start after I push the red button?
A: Please check the power whether the battery has been Plugged-in. If the robot does not start after
plugging-in the battery, please charge the battery and make sure it's voltage is higher than 7.5V.
Q: Why I can't connect the robot with my PC?
A: Please follow the step below:
1 Connecting the Bluetooth cables
 . Connect Serial cable, make sure the serial cable is connected to the COM1 socket of your PC
at one end, and the other end should be connected to the RS232 interface module with
power connector which is assembled together with a serial Bluetooth wireless module.
 . Connect USB cable, plug one end in your PC, and plug the other end in the RS232 interface
module with power connector which is assembled together with a serial Bluetooth wireless
module.
2. Turn on the robot. Check the LED lights on the socket board, and find out if they are flashing
on the socket board. There should be 2 LED lights keep flashing fast on the upper board
PMS5005 in the right rear corner of the robot and 1 LED light keep flashing on the lower
board in the right front corner of the robot. If these 3 LED lights are flashing, the robot is
started completely.
3. Run the WiRobot Gateway. (It can be found at the desktop after installing the WiRobot
System.)

. Wireless Connection

. Input the Robot Address which you can find at the bottom of the robot and on the serial

Bluetooth wireless module which had already been plugged in the lower socket board

PMB5010 of the robot.

. Click the "Connect" button when you are sure that the robot is completely started.

4. Waiting 1 to 3 minutes for the PC connecting to robot, the WiRobot Gateway will minimize

automatically when connected.

5. If it is not connected, close the WiRobot Gateway and turn off the robot. Go back to 2 and try

it again 10 seconds later.

6. If it is still not connected please check whether you have complete and unrestricted access to

the computer and if you have plugged the serial cable in the COM1 serial port of your PC.

Detailed information can be found on page 10 of the user manual.

Q: How to connect the robot directly to PC using the serial cable?

A: Please check page 11.

Q: How to charge the battery?

A: User can simply take out the battery at the lowest deck of the robot to recharge. It will normally

take about 20 hours to fully recharge the 2100mAh battery if slow charging is chosen. Fast

charge would take about 1-2 hours.

25 Valleywood Dr. Unit#20, Markham,

ON L3R 5L9 CANADA

T: (905) 943-9572 F: (905) 943-9197

Email: info@drrobot.com

www.DRROBOT.com