25 Valleywood Drive, Unit 20
Markham, Ontario, L3R 5L9, Canada
Tel: (905) 943-9572 Fax: (905) 943-9197
Support@DrRobot.com

Dr Robot® C# Advance Sample Program General Function Introduction

For X80, Sputnik, Scout, Sentinel 2 Robot

Using Microsoft® Visual Studio® 2008

Version: 1.0.0
May 2008

Copyright Statement

Table of Contents

1 Introduction

Dr Robot provides these sample programs in order to help user to take advantage of powerful DrRobot products.

All these samples are written with C# program with Visual Studio 2008 Express under .Net 3.5 framework.

You can download the free development tools from Microsoft and run the source code.

The main purpose of these sample programs is to demonstrate the robot functions and how to use them. To control the robot as your desired behavior, you maybe need to write your code to call these functions.

Dr Robot always welcomes any improvements from user and encourage users write their own code to control the robot as well as their desired.

To use these sample programs, you need download some support programs from Dr Robot and install or register them. Here is the list:

1. DRROBOTSentinelCONTROL.OCX

   This ActiveX control component needs be copied to windows/system32 folder and use regsvr32 to register it.

2. WiRobotGatewayforWiFi.exe

   You can copy it to your working folder.

3. DrRobotSensorMapBuilder.dll

   This dll file provides some functions to build around sensor map for collision avoidance. It needs to be copied to "windows/system32" folder and use regsvr32 to register it.

4. DrRobotP2PSpeedDrive.dll

   This dll file provide some functions to make robot move form one point to another point. It needs to be copied to "windows/system32" folder and use regsvr32 to register it.

5. DrRobotConstellation.dll

   For Sentinel –II you will need this file. This dll file provides some functions to local the robot position with DrRobot Constellation system. It needs to be copied to "windows/system32" folder and use regsvr32 to register it.

6. VitaminCtrl.dll

   For I90 series, Sentinel –II and Sentinel-III you need to install camera control component. You can download VitaminDecoder.exe to install it.

7. Install DirectX9.0C for Joystick control in program.

Note: For DRROBOTSentinelCONTROL.OCX and Vitamin control, you can refer to relative documents.

Here we will list functions for DrRobotSensorMapBuilder.dll, DrRobotP2PSpeedDrive.dll, and DrRobotConstellation.dll.

2 DrRobot Sensor Map Builder

2.1 Introduction

Source code is also available to selected Dr Robot product owners; please contact Dr Robot Inc support team for detail (support@Drrobot.com).

This function will build an environment map based on infrared range sensors and ultrasonic sensors reading information and output the sensor map to P2P Drive service for collision avoidance.

You can write your own sensor map builder program to implement this function.

Before running this service, you need to configure the infrared range sensor and ultrasonic sensors.

2.2 Configure Sensors

You need to configure sensors first according the sensor mounted location on the robot.

The configure data class define is

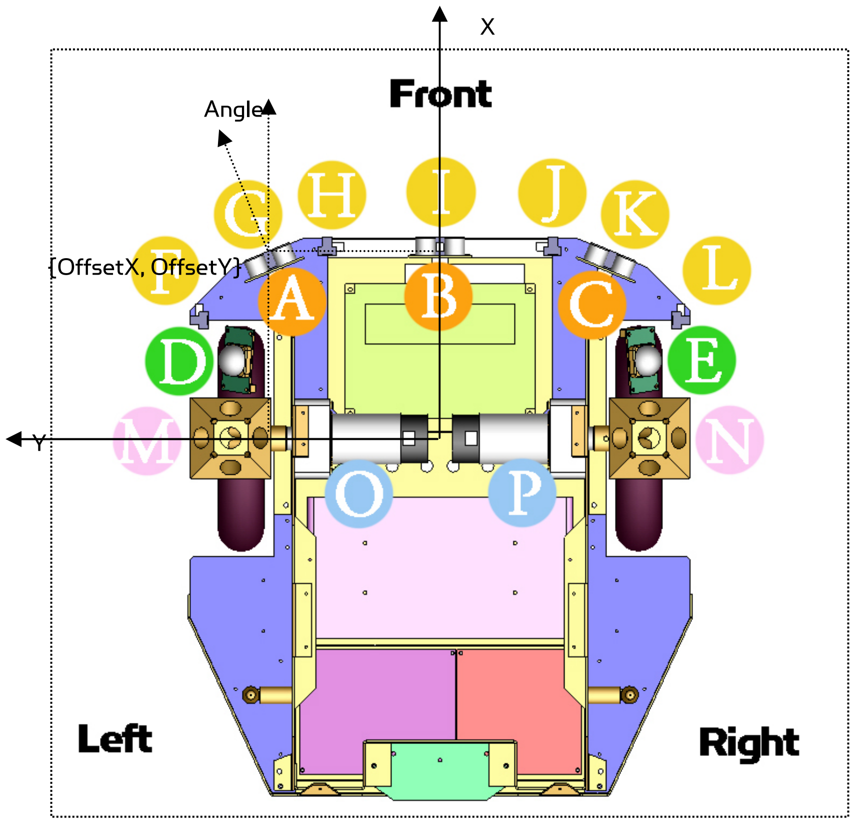
```
        public class ConfigData
                {
                public int IrNum = 0;
                public int UsNUm = 0;
                public IRConfigData[] IrConfigData = new IRConfigData[MAXIRNUM];
                                    public USConfigData[] UsConfigData = new
                   USConfigData[MAXUSNUM];

                   }
```

The MAXIRNUM is 20 and MAXUSNUM is 6.

Now for robots built by Dr Robot, the IrNum is 7 and UsNum is 3.

```
        public class IRConfigData
                 {
                public double OffsetX = 0;
                public double OffsetY = 0;
                public double Angle = 0;
                public double Weight = 0;
                        public int DisTag = 0;

                   }
        public class USConfigData
                {
                 public double OffsetX = 0;
                 public double OffsetY = 0;
                public double Angle = 0;
                public double Weight = 0;
                        public int DisTag = 0;
}
```

These class will define the locations of each sensor.



If the weight = 0 means this sensor is disabled. You can set different value to weight this sensor in sensor map builder

The DisTag is reserved now.

In sample programs, the program will read IrSensorConfig.xml to a dataset and UsSensorConfig.xml to another dataset, then in form load function, call sensorMapBuilder.ConfigSensor(sensorMapConfig) to send this configure data to sensor map builder service.

Now these conifgure xml file need to copy to the running program folder. They are different for different robot. If you change the sensor location, you need to modify this file too.

You can modify the source code to change the folder.

After configuration, this service need robot position information and sensor distance inforsation to builder sensor map.

In sample programm using tmrSensorMapBuilder to update these information to this service.

```
private void tmrSensorMapBuilder_Tick(object sender, EventArgs e)
{
        robotMapPosition.robotX = robotPosition.robotX;
        robotMapPosition.robotY = robotPosition.robotY;
        robotMapPosition.robotDir = robotPosition.robotDir;
        sensorMapBuilder.UpdateSensorInfo(sensorData, robotMapPosition);
}
```

The sensorData contains all infrared sensor distance and ultrasonic sensor information.

```
public class SensorData
```

```
            {
                        public double[] IrDis = new double[MAXIRNUM];
                        public double[] UsDis = new double[MAXUSNUM];
            }
```

All these distance information are got from motionControl in standard sensor event and custome sensor event .

2.3 Mapbuilder Service Output

This service output the sensor map by a call back function. It will fire this callback function after it updates the map with the latest sensor and robot position information.

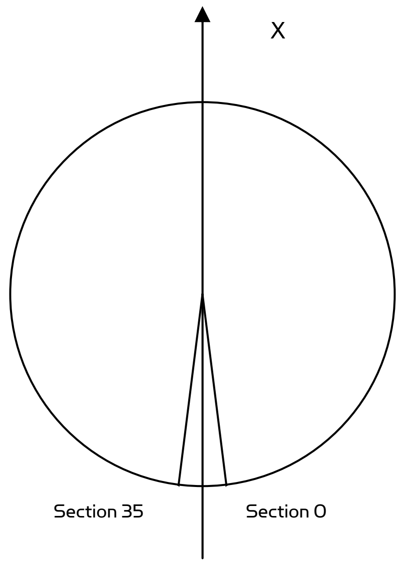
The sample will first register the call back functions by calling sensorMapBuilder. RegisterSensorMapCallback(UpdateSensorMap).

You can do your own work in UpdateSensorMap function.

The output data is defined by

```
            public class SensorMapData

                        {

                                    public int[] Polar_Value_Map = new int[MAPSECTIONNUM];

                                    Pblic int[] Polar_Bin_Map = new int[MAPSECTIONNUM];

                        }
```

The MAPSECTIONNUM is 36.



The value in Polar_Value_Map represents the histogram data in every section.

Polar_Bin_Map only has two values 1 or 0. “1” means that there is object in this direction. “0” means no object in this direction.

P2P Drive service will use these data to determine the robot move direction.

3 DrRobotP2PSpeed Drive Service

3.1 Introduction

Source code is also available to selected Robot owners, and please contacts Dr Robot Inc for detail (support@Drrobot.com).

This service dll provides the function to drive robot from one point to another point with or without collision avoidance.

3.2 Initialize

To initialize the service you need call DrRobotP2PSpeedDrive().

After that you need call below function to set robot some parameters.

```
Public bool SetRobotInfo(RobotInfo data)

public class RobotInfo
    {
                public double WheelRadius = 0.085;     //unit:m
                public double WheelDis = 0.32;       //unit:m
                public int OneCircleCount = 800;      //encoder count
                public int MotorDir = 1;

  }
```

Robot Series	WheelRadius	WheelDis	OneCircleCount	MotorDir
I90, sentinel	0.085	0.32	800	-1
X80,Sputnik	0.085	0.26	1200	1

You can call below function to set collision avoidance parameters.

```
public bool SetP2PCAParameter(P2PCAParameter data)

public class P2PCAParameter
    {
                 public int Distance2ObjectTH;
                public int MinOpenTH;
                public int CATH;
                //here is the default value
                 public P2PCAParameter()
                {
                            Distance2ObjectTH = 6;     //6 section
                            MinOpenTH = 4;          //minimum open 4 section
                            CATH = 100;
                }

    }
```

You also can call below function to set P2P drive parameters.

```
public bool SetP2PDriveParameter(P2PDriveParameter request)

public class P2PDriveParameter
    {
            public double ToleranceStep;
            public double SlowGainTurnRatio;
            public double SlowGain;
            public double MinTurnSpeed;
            public double MinForwardSpeed;
            public double MoveMoment;
            //here is the default value
            public P2PDriveParameter()
            {
                    ToleranceStep = 0.01;     //1 cm
                    SlowGainTurnRatio = 0.1;
                    SlowGain = 0.05;
                    MinTurnSpeed = 0.3;      //around 34 degree
                    MinForwardSpeed = 0.05;
                    MoveMoment = 0;
            }

    }
```

Usually you don't need to modify these parameters. You'd better contact Dr Robot before you modify these parameters.

3.3 Update Robot Position and Sensor Map Information

This service need robot current position and sensor map information to drive the robot.

public bool UpdateSensorMap_Position(RobotPosition posData, SensorMapData sensorMap)

Here is the parameters define:

```
public class RobotPosition
{
        public double robotX = 0;
        public double robotY = 0;
        public double robotDir = 0;
        public int LeftWheelEncoderCnt = 0;
        public int RightWheelEncoderCnt = 0;
}

public class SensorMapData
{
        public int[] Polar_Value_Map = new int[MAPSECTIONNUM];
        public int[] Polar_Bin_Map = new int[MAPSECTIONNUM];

                }
```

You can get RobotPosition by motioncontrol encoder sensor information and DrRobot Constellation system(for Sentinel^2) or StarGazer syste(for Sentinel^3).

SensorMapbuilder will update the SensorMapData.

Sample program will update these information at 10Hz in tmrP2PUpdate.

3.4 SetTarget Point and Send P2P Drive Command

3.4.1 Set Target Point

Before you drive robot, you need to set the destination point information and drive behavior.

```
public bool SetTargetPosition(SetTargetPointRequest request)

public class SetTargetPointRequest
{
   public double TargetX;
   public double TargetY;
   public double TargetDir;
   public int StopTime;
   public double ForwardSpeed;
   public bool Forgetable;
   public bool NonStop;
   public bool FinalPosture;
   public int TargetTime;
   public double TargetTolerance;
   public double MaxTurnSpeed;
   public bool CAEnable;
   public bool ReverseDrive;
   public double TargetDirTolerance;

   public SetTargetPointRequest()
   {
      TargetX = 0;
      TargetY = 0;
      TargetDir = 0;
      StopTime = 0;
      ForwardSpeed = 0;
      Forgetable = false ;
      NonStop = false;
      FinalPosture = false;
      TargetTime = 0;
      TargetTolerance = 0;
      MaxTurnSpeed = 0;
      CAEnable = false;
      ReverseDrive = false;
      TargetDirTolerance = 0;
   }
}
```

In sample program, you can choose the a xml path to read the setting data.

Below is a sample xml file describing the P2P drive request.

```
<PointConfigTable>
        <TargetX>0.8</TargetX>
        <TargetY>0.6</TargetY>
         <TargetDir>90.0</TargetDir>
        <StopTime>50</StopTime>
        <ForwardSpeed>0.2</ForwardSpeed>
        <Forgetable>false</Forgetable>
        <NonStop>false</NonStop>
        <FinalPosture>true</FinalPosture>
        <TargetTime>200</TargetTime>
        <TargetTolerance>0.05</TargetTolerance>
        <MaxTurnSpeed>35</MaxTurnSpeed>
        <CAEnable>false</CAEnable>
        <ReverseDrive>false</ReverseDrive>
        <TargetDirTolerance>5</TargetDirTolerance>
</PointConfigTable>
```

It will drive robot from current position to target position (0.8m, 0.6m). On arrival, it will turn to direction at 90(degree) within 5 degree tolerance.(with TargetDir = 90, FinalPoature = true, and TargetTolerance = 5). It will stay at this point for 50 seconds(StopTime = 50, and NonStop = false). The maximum forward speed is 0.2m/s. The robot must arrive this target wihin the given tolreance 0.05m. (Forgetable = false, TargetTolerance = 0.05m). The given task time is 20 second(TargetTime = 200). The maximum turn speed is 35 degree/s. The collison avoidance feature is disabled (CAEnable = false). Robot drives toward this target with forward motion.(ReverseDrive = false).

3.4.2 Send P2P Drive Command:

Sending command to P2P drive service to drive robot or stop robot.

```
public bool SendP2PCmd(P2PCtrlCmd cmd)

public enum P2PCtrlCmd : byte
 {
        Null = 0x0,
        StopP2P = 0x01,
        SuspendP2P = 0x2,
        ResumeP2P = 0x3,
        P2PGo = 0x04,
        P2PSkip = 0x5,
}
```

StopP2P: stop the robot and this time drive task,

SuspendP2P: stop the robot and pause this time drive task.

ResumeP2P: resume this time drive task.

P2PGo: start this time drive task

P2PSkip: to set the stoptime to 0. (Robot will not stop and wait at the destine point).

3.5 Output the Motor Command

The service will generate drive command to drive the motor every 100ms if the task is running.

First you need to register a call back function to receive this command.

```
public void RegisterP2PDriveCmdCallback(DrRobot_P2PDrive_Command_CALLBACK_FUNC func)

public delegate void DrRobot_P2PDrive_Command_CALLBACK_FUNC(MotorControlCmd cmd);

public class MotorControlCmd
   {
        public P2PServiceStatus P2PDriveStatus = P2PServiceStatus.P2POver;
        public P2PDriveMethod P2PDriveMethod = P2PDriveMethod.SpeedControl;
```

```
                public int LeftWheelCmd = 0;
                public int RightWheelCmd = 0;
                public int RunTime = 0;
                public double TestVar = 0;
                public MotorControlCmd()
                {

                }

        }


        public enum P2PDriveMethod:byte
        {
                SpeedControl = 0x0,
                PositionControl = 0x02,
        }


        public enum P2PServiceStatus : byte
        {
                Null = 0x0,
                P2PGo = 0x01,          //P2P Drive running
                P2POver = 0x02,         // P2P Task over
                P2PCAStuck = 0x03,       // With Collision Avoidance, it can not find a open direction
                P2PWait = 0x04,         // If you set stop time for a point, it will wait setting time after achiving the
                                                   point
                P2PTurn = 0x05,         // if you set final direction, the robot will turn to settting direction after
                                       achiveing the point
                P2PSuspend = 0x06,       // you can send command to suspend the P2P task
                P2PAntiStuck = 0x07,      // with motor protection, it means robot  detect stuck, it will auto backwards
                P2PMotorHeatUp = 0x08,     // it means robot detect motor heat up, the task has to wait some time
        }
```

In call back function, you need send out motor command according P2P service status.

```
        private void ExeP2PCmd(DrRobotP2PSpeedDrive.MotorControlCmd cmd)
        {


                lblP2PStatus.Text = cmd.P2PDriveStatus.ToString();
                robotP2PStatus = cmd.P2PDriveStatus;


                if (robotP2PStatus == DrRobotP2PSpeedDrive.P2PServiceStatus.P2POver)
                {

                        motionControl.DisableDcMotor(0);
                        motionControl.DisableDcMotor(1);
                        return;
                }


                //always exe the wheel control cmd, 0 means stop robot

                 if (checkBoxMotorProtect.Checked)
                 {
                         if ((leftMotor.heatUpProtect) || (rightMotor.heatUpProtect))
                        {
                              //heat up, need stop
                              motionControl.DisableDcMotor(0);
                              motionControl.DisableDcMotor(1);
                              return;
                    }
                    else if ((leftMotor.stuckProtect) || (rightMotor.stuckProtect))
                    {
                              cmd.LeftWheelCmd = -cmd.LeftWheelCmd;
                              cmd.RightWheelCmd = -cmd.RightWheelCmd;
                    }

          }

          lblP2PCmd.Text = cmd.P2PDriveMethod.ToString();
          lblLeftWheelCmd.Text = cmd.LeftWheelCmd.ToString();
          lblRightWheelCmd.Text = cmd.RightWheelCmd.ToString();
          lblP2PCmdTime.Text = cmd.RunTime.ToString();


          if (cmd.P2PDriveMethod == DrRobotP2PSpeedDrive.P2PDriveMethod.SpeedControl)
          motionControl.DcMotorVelocityNonTimeCtrAll(short.Parse(lblLeftWheelCmd.Text),
          short.Parse(lblRightWheelCmd.Text), NOCONTROL, NOCONTROL, NOCONTROL, NOCONTROL);

          else if (cmd.P2PDriveMethod == DrRobotP2PSpeedDrive.P2PDriveMethod.PositionControl)
          motionControl.DcMotorPositionTimeCtrAll(short.Parse(lblLeftWheelCmd.Text),
          short.Parse(lblRightWheelCmd.Text), NOCONTROL, NOCONTROL, NOCONTROL, NOCONTROL,
          short.Parse(lblP2PCmdTime.Text));
}
```

4 DrRobot Constellation

4.1 Introduction

You could subscribe to this service to get the estimation of robot position.

About DrRobot constellation system and transponder, you can read oher documents about sentinel robot.

First you need initialize the service and set the communication .

On robot we use WiFi module channel2 to connect with constellation system and use TCP connection.

```
          public bool Init(string IPAddr, int PortNum)
```

You need to set the IP to robot consellation IP and port number.

You also can call function to set some parameters.

public bool SetConstellaton(double speed, double receiveDis, double checkbound).

By defult, the receiveDis is the distance between the receive moudle on robot.

Checkboound is the reliable check threshold in estimate algorithm.

Speed is the sound speed used to measure the distance between recive module and transponder sensor .

4.2 Configure Transponder

You need set the transponder position first.

```
public bool SetTransponder(DrRobotTransponderConfig configData)

public class DrRobotTransponderPosition
{
        public double PosX = 0;
        public double PosY = 0;
        public double PosZ = 0;
        public double PosDir = 0;
}

 public class DrRobotTransponderConfig
 {
        public DrRobotTransponderPosition[] transponderSet = new
        DrRobotTransponderPosition[128];

         public DrRobotTransponderConfig()
         {
                for (int i = 0; i < 128; i++)
                {
                        transponderSet[i] = new DrRobotTransponderPosition();
                        transponderSet[i].PosDir =0;
                        transponderSet[i].PosX = 0;
                        transponderSet[i].PosY = 0;
                        transponderSet[i].PosZ = 0;
                }
        }

}
```

In demo program, you need read TransponderPositionData.xml to get the transponder information. The file need to copy in running program folder. You can modify the source code to change the file's location.

4.3 Output the Estimate position

First you need to register the call back function to receive the estimate position data and the sensor data.

```
public void RegisterConstellationDataCallback(DrRobotConstellation_DATA_CALLBACK_FUNC func)

public delegate void DrRobotConstellation_DATA_CALLBACK_FUNC(DrRobotConstellationCommMsg Msg);

public class DrRobotConstellationCommMsg
{
   public double gpsX = 0;
   public double gpsY = 0;
   public double gpsDir = 0;
   public int[] transponderID = new int[4];
   public double[] sensorDistance = new double[8];
   public double gpsCompass = 0;
   public bool dataFlag = false;
   public DrRobotConstellationCommMsg()
   {
      gpsX = 0;
      gpsY = 0;
      gpsDir = 0;
```

```
                for (int i = 0; i < 4; i++)
                {
                    transponderID[i] = 0;
                }
                for (int i = 0; i < 8; i++)
                {
                    sensorDistance [i] = 0;
                }

                gpsCompass = 0;
                dataFlag = false;
            }
        }
```

gpsCompass is optional device data.

dataFlag = true means estimate data is latest and reliable.
In call back function, you can do more check to use this estimate position.
Another callback is sending back some information about this service.

```
public void RegisterConstellationServiceCallback(DrRobotConstellation_Service_CALLBACK_FUNC func)
public delegate void DrRobotConstellation_Service_CALLBACK_FUNC(string Msg)
```

5 Some Important Functions in Demo Program

5.1 Encoder Estimate Position

If the robot does not have any constellation, the robot only can estimate the position by encoder information like X80, Sputnik, Scout and i90 series.

In motion control ActiveX control motor sensor event, you can get the latest encoder count information. By this information, you can estimate the wheel running distance for each wheel and estimate the position of robot.

The algorithm is implemented in demo program function encoderGps().

5.2 Motor protect

There are two motor protect functions.

One is anti-stuck function. If the function detects the wheel stuck, the program will send out reverse direction command to let robot move back.

One is heat protection. If the function detects a bigger motor current (exceed set threshold) lasts a while, the program will set motor stop for a moment to protect the motor.

All algorithms are implemented in function StucKHeatDetect().