

Dobot API Reference and Developing Examples

1 Application Scope

The scope of this document is limited to use Dobot API to develop applications to control Dobot products. When using Dobot API to develop applications, you need the following files:

- Dobot DII.dII: Dobot API Dynamic-Link Library;
- DobotDII.h: Dobot API declarations:
- DobotType.h: All the related data structures and enumerations in Dobot API.

Dobot API Dynamic-Link Library iswritten in C++, which supports the software development of languages like C, C++, C#, Java, Python, JS and etc..

1.1 Developing Process

The processes of calling Dobot API to develop applications are as shown in figure 1:

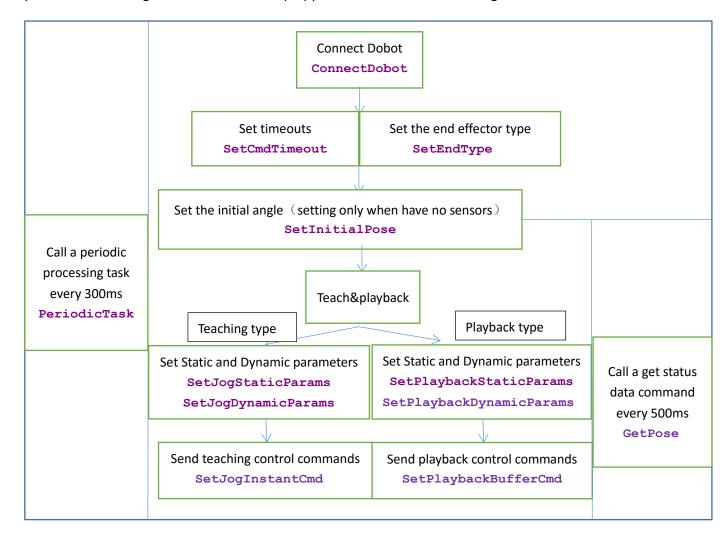


Figure1 developing flow chart



- Create a timer or a thread to call a periodic processing task at a fixed interval;
- Establish the connection with the robotic arm and determine whether the return type is correct or not. If not then the errors need to be handled.
- Set the initial position for the rear arm and fore arm, timeout threshold, end effector type, the static/dynamic parameters for teaching and playback.
- Send operation commands.

1.2 Descriptions of error type

		Error cause
Code value	Error flag	
0	DobotResult_NoError	No error
1	DobotResult_Sensor1Warning	Reararm angle greater than 95 degrees or less than -20 degrees
2	DobotResult_Sensor2Warning	Forearm angle greater than 100 degrees or less than -20 degrees
3	DobotResult_Sensor12Warning	Error 1 and Error 2 occur at the same time
4	DobotResult_NotFound	No available COM port is found,.
5	DobotResult_Occupied	The COM port is occupied
6	DobotResult_NoDataUploaded	No current status data is uploaded when establishing a connection
7	DobotResult_Timeout	Command execution timeout

1.3 Dobot coordinate system

The robotic arm is working under the World Coordinate System based on the intersection point of the three axes, the direction of X, Y, Z axis is shown as figure 2.

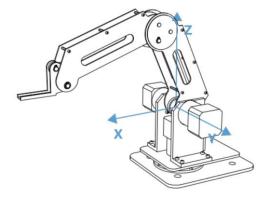


Figure 2 The corresponding world coordinate system of Dobot

2 API

All functions of this API library will be detailed in the following sections.



2.1 Connect and Disconnect

The upper computer is using serial connection, so please make sure you have installed the necessary serial drivers. Otherwise, the library cannot identify your COM port. And this API is responsible for handling upper computer with connection, disconnection and reset of Dobot.

2.1.1 Establish a connection with the Dobot

Function Name	<pre>int ConnectDobot(void)</pre>	
Description	Establish a connection with the Dobot	
Parameters	Parameters NULL	
Return value enum DobotResult		
For detailed definition please check section 1.2		

2.1.2 Disconnect from the Dobot

Function Name	<pre>void DisconnectDobot(void);</pre>
Description Disconnect from Dobot	
Parameters	NULL
Return value	NULL

2.1.3 Reset Dobot

Function Name	<pre>void ResetDobot(void);</pre>
Description	Reset the Dobot controller
Parameters	NULL
Return value	NULL

2.2 Period Process Tasks

When using Dobot API, there are two functions must be periodically called. You can create a timer or a thread to call them separately in different languages. Among them, you can call the GetPose function from anywhere in order to get the latest status of the robotic arm instantly.

2.2.1 Period Process Task

Function Name	<pre>void PeriodicTask(void);</pre>
Description	Period task for sending instruction. This function should be called periodically . A cycle time
	less than 500ms is recommended.
Parameters	NULL
Return value	NULL



2.2.2 Get the status of the robotic arm

Function Name	<pre>int GetPose(Pose *pose);</pre>		
Description	Get the current status of the robotic arm A cycle time less than 500ms is recommended.		
Parameters	struct Pose:		
	f	float x	X axis absolute position
	f	float y	Y axis absolute position
	f	float z Z axis absolute position	
	f	float rHead R axis absolute position	
	f	float jointAngle[4] Base angle, Rear arm angle, forearm	
		angle, servo angle	
	b	bool isGrab Pump or Laser on/off status	
	float gripper Gripper angle		
Return value	For detailed definition , please check section 1.2		

2.3 Teaching function control

If you want to control the movement of the robotic arm, you can call the SetJogInstantCmd command to control the movement and the motion pattern. And the max velocity, max acceleration and velocity ratio setting can be achieved by calling SetJogStaticParams and SetJogDynamicParams.

2.3.1 Set the static parameters

Function Name	<pre>int SetJogStaticParams(JogStaticParams);</pre>		
Description	set the joint jog movement static parameters		
Parameters	struct JogStaticParams		
	parameter descriptions of st	ruct JogStaticParams	
	float jointMaxVelocity The max velocity of joint jog movement		
	float jointMaxAcceleration The max acceleration of joint jog movement		
	float servoMaxVelocity The max velocity of servo jog movement		
	float servoMaxAcceleration The max acceleration of servo jog movement		
	float linearMaxVelocity The max velocity of linear jog movement		
	float linearMaxAcceleration The max acceleration of linear jog movement		
Return value	For concrete definition please check section 1.2		

2.3.2 Set dynamic parameters

Function Name	<pre>int SetJogDynamicParams(JogDynamicParams *jogDynamicParams);</pre>	
Description	Description set the jog movement dynamic parameters	
Parameters	struct JogDynamicParams	
	Parameter descriptions of struct JogDynamicParams	
	float velocityRatio; // Velocity ratio	



Return value	For concrete definition please check section 1.2	
--------------	--	--

2.3.3 Control movement

Function Name	<pre>int SetJogInstantCmd(JogInstantCmd *jogInstantCmd);</pre>			
Description	Send the jog control command			
Parameters	1:joint	control, 0 li	near control	
	enum Jo	ogCmd Each defi	nition of <mark>enum</mark> JogCmd is as fo	ollows
	NO	Enumeration	Joint control	Linear control
		value		
	0	JogIdle	All buttons released	All buttons released
	1	JogAPPressed	Base Axis+(counterclockwise)	X axis+
	2	JogANPressed	Base Axis-(clockwise)	X axis-
	3	JogBPPressed	Rear arm	Y axis+
			Axis2+(counterclockwise)	
	4	JogBNPressed	Rear arm Axis2-(clockwise)	Y axis-
	5	JogCPPressed	Forearm	Z axis+
			Axis3+(counterclockwise)	
	6	JogCNPressed	Forearm Axis3-(clockwise)	Z axis-
	7	JogDPPressed	Servo rotates	Servo rotates forward
			forward(counterclockwise)	
	8	JogDNPressed	Servo rotates	Servo rotates backward
			backward(clockwise)	
	9	JogGrab	Suction cap or Gripper grab	Suction cap or Gripper gr
	10	JogRelease	Suction cap or Gripper	Suction cap or Gripper
			release	release
	11	JogGPPressed	Gripper rotates forward	Gripper rotates forward
	12	JogGNPressed	Gripper rotates backward	Gripper rotates backward
	13	JogLaserOn	Laser on	Laser on
	14	JogLaserOff	Laser off	Laser off
Return value	For detailed definition, please check section 1.2			

2.4 Playback function control

If you want to make the robotic arm cycles at a fixed pattern rather than controlling it manually every time, you can call this kind of function. The basic parameters are similar to manual control modes.

2.4.1 Set static parameters

Function Name	int SetPlaybackStaticParams (PlaybackStaticParams		
	*playbackStaticParams);		
Description	set playback joint jog static parameters		
Parameters	struct PlaybackStaticParams		
	Parameter descriptions of struct PlaybackStaticParams		
	float jointMaxVelocity The max velocity of joint jog movement		



	float jointMaxAcceleration	The max acceleration of joint jog movement
	float servoMaxVelocity	The max velocity of servo jog movement
	float servoMaxAcceleration	The max acceleration of servo jog movement
	float linearMaxVelocity	The max velocity of linear jog movement
	float linearMaxAcceleration	The max acceleration of linear jog movement
	float pauseTime	Pause time s(deprecated parameter)
	float jumpHeight	JUMP height
Return value	For concrete definition please check section 1.2	

2.4.2 Set dynamic parameters

Function Name	int SetPlaybackDynamicParams (PlaybackDynamicParams	
	*playbackDynamicParams);	
Description	set playback joint jog dynamic parameters	
Parameters	Parameters velocityRatio, accelerationRatio	
Return value	Return value For concrete definition please check section 1.2	

2.4.3 Control movement

Function Name	<pre>int SetPlaybackBufferCmd(PlaybackBufferCmd *playbackBufferCmd);</pre>		
Description	set palyback control instruction		
Parameters	struct PlaybackBufferCmd		
	Parameter descriptions of struct PlaybackBufferCmd		
	uint8_t motionStyle	Motion style, 0:jump, 1: Movj, 2: Movl	
	uint8_t isGrab	Suction cap open/close	
	<pre>float x;float y;float z;float rHead;</pre>	X、Y、Z、R axis coordinate	
	float gripper	Gripper angle	
	float pauseTime	Pause time(s)	
Return value	For concrete definition please check section 1.2		

2.5 Other functions

If not set, this kind of API will use the default values.

2.5.1 Timeout setup

Function Name	<pre>int SetCmdTimeout(uint32_t cmdTimeout);</pre>	
Description	Timeout of every instruction	
Parameters Timeout unit: ms (default value: 3000 ms)		
Return value For concrete definition please check section 1.2		



2.5.2 Set initial position of the Reararm and Forearm

Function Name	<pre>int SetInitialPose(InitialPose *initialPose);</pre>	
Description	Set the initial position of the Rear arm and Forearm. You can check the file DobotType.h for	
	the definition of the InitialPose.	
Parameters	Parameters joint2Angle: Reararm, joint3Angle: Forearm	
Return value	Return value For detailed definition please check section 1.2	

2.5.3 Set end effector type

Function Name	<pre>int SetEndType (EndType endType);</pre>	
Description	Set the end effector type. You can check file DobotType.h for definition of EndType.	
Parameters	Parameters 0: Suction cap, 1: Gripper, 2: Laser	
Return value For detailed definition please check section 1.2		

3 Dobot Demo

Dobot offers three demo applications, respectively using WPF(c#), Python and java. There is a detailed instruction of the demo.

Note: In all demos below, we called SetInitialPose() API to initialize the joint angle of forearm and rear-arm, while in real application development, if the angle sensors are not connected or the sensor value cannot be read successfully, using function SetInitialPose() to set initial value for the joint 2 and joint 3 value is required, otherwise, this process is not necessary.

Note: In all demos below, we called SetInitialPose() API to initialize the joint angle of forearm and reararm, while in real application development, this is only necessary when the angle sensors are not connected or the sensor value cannot be read successfully.

3.1 WPF Demo

This demo uses developer kits of VS2013 based on .NET 4.0 and DobotDII.dll encapsulated by C#. CPlusDII/DobotDII.cs encapsulates the functions and CPlusDII/DobotDIIType.cs encapsulates the data types, which correspond to the function and data type defined in DobotType.h.

Notice: it is required to put the dll library into the same folder as the program.

3.1.1 Brief Introduction

The GUI(Graphic User Interface) is shown as figure 3 when the program running. The Jog area consists of buttons for jog operation or settings for jog, e.g., set jog mode and velocity ratio. EndType area includes end type settings, pump on/off control, grab or release and so on. In Playback area, you can set the motion mode, and manually set the target coordinate value and run to the target point. The Tip zone shows the current operation information.



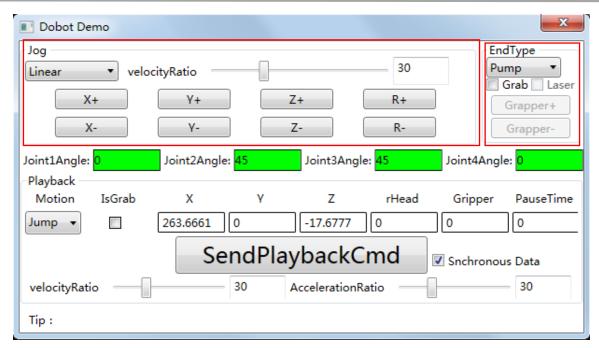


Figure 1 Dobot demo procedure interface

3.1.2 EndType Area Introduction

We can change the end effector type using the EndType combo box. The end type control is only available when the corresponding end effector type is selected. As shown in Figure 4.

- 1. Pump: Change the pump state so that the suction cap can take something up or drop it somewhere.
- 2. Gripper: Control the gripper angles
- 3. Laser: You can turn on the laser here, then you can manually adjust the height of laser in order to make it focused on the target plan.



Figure 2 Dobot Endtype

3.1.3 Jog Region

Jog control methods include Joint Jog and coordinate liner jog. VelocityRatio indicates that moving velocity can be adjusted by dragging the sliding bar up and down. When Dobot runs, its four angles will be shown on textbox from Joint1Angle to Joint4Angle, respectively angle of base, forearm, reararm, and servo. Joint Jog control and Liner Jog are as shown in Figure 5 and Figure 6. You can control x, y, z, Joint1+ and others to make Dobot work and check trajectory coordinates.

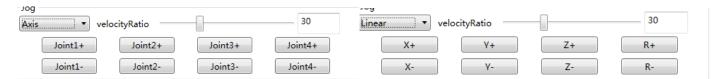


Figure 5 Joint Jog

Figure 6 Liner Jog



3.1.4 Playback

The playback region, among which state data will be shown when "Synchronize Data" is selected. VelocityRatio and AccelerationRatio indicates velocity rate, please review chapter 3.4.2 for more information. There are three MotionStyle options: JUMP、MOVJ、MOVJ、MOVL.

1. JUMP: from point A to point B, the trajectory is shown below, the end effector will lift upwards by amount of Height (in mm) and move horizontally to a point that is above B by Height and then move down to Point B. Value of Height can be configured in the Playback tab of the Config Dobot Module, of which the default value is 20 mm. Click send to configure Dobot after changing the value, as shown in Figure 7.



Figure 7 JUMP trajectory illustration

- 2. MOVJ: Joint movements. From point A to point B, each joint will run from initial angle to its target angle, regardless of the trajectory. The motion time for all joints are the same which means all joints will start and finish at the same time. (Figure 8)
- 3. MOVL: The joints will cooperate in order to perform a line trajectory from point A to point B, as shown in Figure 8.

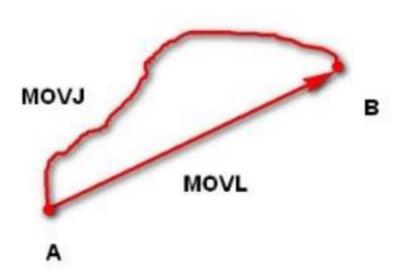


Figure 8 Illustration of MOVJ and MOVL

When using playback, please choose motion trajectory, whether to grab or not, write the value of X, Y, Z, rHead, Gripper, pauseTime. Generally, plus or minus a value based on the original value to get the aimed value you want. Check "whether to syn state display or not" when you want to get the current value, and uncheck it when you want to send playback command, to prevent the values covered by the returned data.



3.2 Python Demo

The demo is a terminal procedure written in Python(no GUI built). You can open DobotControl.py in the demo folder with python-2.7.11 to look into detailed implementations. When the program is running, Dobot will swing from side to side repeatedly. A screenshot of the running program is shown in following figure.

```
D:\AppInstall\oython\python.exe
Pose: 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0
Pose: 0.0 0.0 0.0 0.0 0.0 0.0 0.0
Pose: 0.0 0.0 0.0 0.0 0.0 0.0 0.0
!!!Start checking 9600bps: QDateTime<2016-05-24 09:24:36.471 中国标准时间 Qt::Ti
meSpec(LocalTime))
Pose: 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0
!!!Stop checking 9600bps: QDateTime<2016-05-24 09:24:36.929 中国标准时间 Qt::Tim
eSpec(LocalTime))
Real baudrate is 9600bps
Both long & short arm angle not good!
Pose: 38.8804588318 260.783691406 -17.6777038574 81.5199966431 81.5199966431 45.
 45.0 0.0
Pose: 38.8804588318 260.783691406 -17.6777038574 81.5199966431 81.5199966431 45.
 45.0 0.0
Pose: 107.242042542 240.871276855 -17.6777038574 66.0 66.0 45.0 45.0 0.0
Pose: 167.782852173 203.393066406 -17.6777038574 50.4800033569 50.4800033569 45.
0 45.0 0.0
Pose: 167.782852173 203.393066406 -17.6777038574 50.4800033569 50.4800033569 45
0 45.0 0.0
Pose: 38.8804588318 260.783691406 -17.6777038574 81.5199966431 81.5199966431 45.
 45.0
```

Figure 9 Python demo operation interface

3.2.1 Introduction

The demo's basic processing flow is shown in Figure 10. Dobot API library is written in C++. When used in Python, we use byref instead of pointer, and struct is declared by inhabiting the structure class, e.g., for C++, it is 'struct JogStaticParams', while in Python it is 'class JogStaticParams(Structure)'.

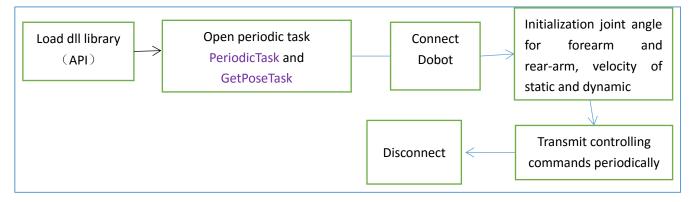


Figure 10 python demo operation flow chart



3.3 Java Demo

The following java demo terminal procedure need import Jna-4.2.2.jar in DobotDemo folder, in order to call Dobot API librarywritten in C++. The demo is compiled by Java EE of Eclipse. Of course, you can use other java developer kits. The demo is a terminal procedure without UI interface, which controls the robot to swing from side to side repeatedly as the python demo. The demo running screenshot is shown as python procedure in chapter 3.2.

3.3.1 Brief Introduction

The demo operational process is same as shown in Figure 9. It uses jna approach to call dlllibrary written in C++. According to JNA grammar, you need to load JNA library, and inherit the Structure to declare a struct:

import com.sun.jna.Library; import com.sun.jna.Native; import com.sun.jna.Structure;

when you compile API interface (DobotDII), please inherit Library class, implement all function interface under DobotDII, and then using

DobotDII instance = (DobotDII) Native.loadLibrary("DobotDII", DobotDII.class);

to load the interface, all API operations are implemented in the file DobotDII.java .

3.4 JS Development examples

It is a remote control Demo with an HTML interface using JS's Ajax communication, which uses Qt to implement a TCP server to communicate with Dobot API. Node.js is in charge of transmitting the data received from the remote client HTTP POST to the TCP server through TCP protocols. As the interface is based on the common JQuery library and the Server side on the basis of Node.js, Node.js library (you can download it from here: https://nodejs.org/en/) needs to be installed locally or on the server side. Turn on the TCP server (run DobotJsServer), also turn on the HTTP service of the Node.js (run CMD ,enter the DobotJsClient directory provided by the demo and input "npm start" command), input http://127.0.0.1:8000/users in the browser (If it is a remote installed node.js server then just need to input the remote IP or domain name, for example, http://www.dobot.cc/users). The running interface is shown as figure11.



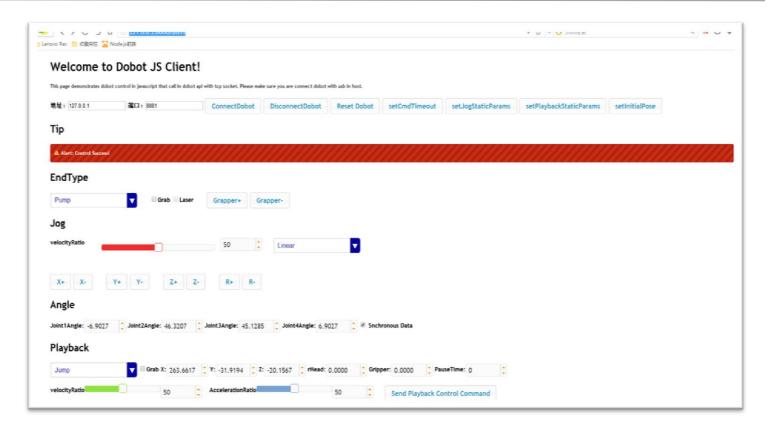
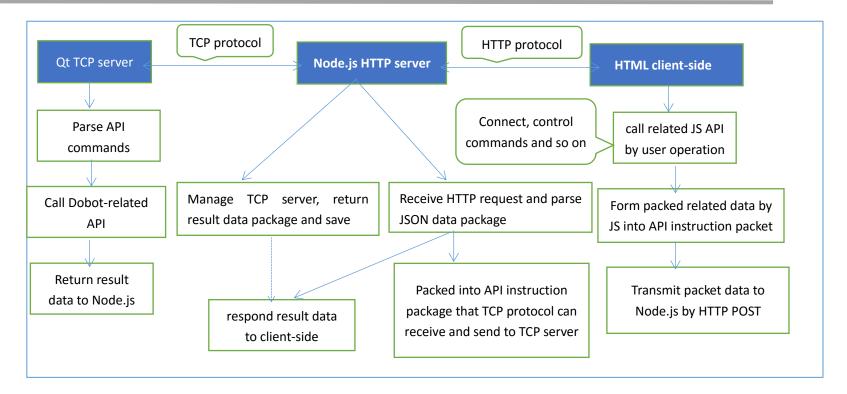


Figure 3 Js Demo operation interface

3.4.1 Demo Framework

In this demo, we realize the HTTP server using Node.js, and realize an application to call the Dobot API using Qt, while use TCP communication between HTTP server and the application, in which way we can unleash the strong processing capability of Node.js and avoid the problem that JS can neither process TCP communication nor call Dobot API DLL directly.





4 Version Description

Version	Date	Modify Description
V1.0.0	2016-05-17	create a document
V1.0.1	2016-05-19	add instruction of demo procedure and library files modification
V1.0.2	2016-05-24	Add instruction of python and java demo