Dobot API

Dobot Magician

TN01010101 V1.0.0 Date: 2016/07/29

Shenzhen Yuejiang Technology Co.,Ltd

Revised History

Version	Date	Reason
V1.0.0	2016/07/29	Create a document
V1.0.1	2016/08/09	Modify protocol classification, description and so on
V1.0.2	2016/08/22	Modify the interface description of EndEffector settings parameters, also support quene way
V1.0.3	2016/08/26	Increase search Dobot function; modify EIO part of the definition
V1.0.4	2016/08/27	Modify searched API definition and so on
V1.0.5	2016/08/29	Amend the reset real-time poseAPI interface error
V1.0.6	2016/08/31	Add Wi-Fi configuration function
V1.0.7	2016/09/09	Modify the control connection of the EndEffector
V1.0.8	2016/09/13	Modify Home Parameter and maxJumpHeight
V1.0.9	2016/09/19	Add the connection of DNS
V1.1.0	2016/09/27	Add an API interface to detect whether the -Fi module is in place
V1.1.1	2016/10/24	Modify output parameters description in ConnectDobot

Contents

1. Application Scope

The document is aiming to have a detailed description of Dobot API and general process of Dobot API development program.

2. API Interface Description

2.1 Dobot Commands

There are two features of communication commands when communicating with Dobot controller:

1. All commands can be returned to the controller; regarding to setup commands, the controller can cut command parameter domain and then return; while for getting commands, the controller can fill the parameters got in parameter domain and then return.
2. Dobot controller supports two kind of commands: Immidinate instruction and queen instruction:
   - Immidinate instruction: Dobot controller will process the command once received regardless of whether there is the rest commands processing or not in the current controller;
   - Queue command: Dobot controller receives the queue instruction, the command is pressed into the controller internal instruction queue. Dobot controller will execute instructions in the order in which the instruction was pushed into the queue.

For more detailed information about Dobot commands, please refer to Dobot protocol.

2.2 API Background API

In order to use API properly, call API background task periodically at first. One can create timer or thread to call the task in different languages.

Figure 2.1 API description of background task API

Prototype	void PeriodicTask(void)
Description	API background task, recommend call 100ms periodically.
Parameter	Void
Return	Void

2.3 Connect/Disconnect

2.3.1 Search Dobot

Figure 2.2 Get number of Dobot interface

Prototype	int SearchDobot(char *dobotList, uint32_t maxLen)
Description	Search Dobot, dll will store connected Dobot information and use ConnectDobot.
Parameter	dobotList:array TPTR transmitted externally,dll will write serialport/UDP searched into dobotList, for example, a specific returned dobotList is "COM1 COM3 COM6 192.168.0.5", different interface should be separated by the space. maxLen: The max length that dobotList transmitted externally can support to avoid internalstorage overflow.
Return	Dobot number

2.3.2 Connect Dobot

Figure 2.3 Descripition of Connect Dobot controller interface

Prototype	int ConnectDobot(const char * portName, int baudrate)
Description	Connect Dobot controller, in this process, portName can be got from the dobotList oflast API. Note：If we don't call SearchDobot but call ConnectDobot directly (portName is empty), and then dll will connect the first found Dobot controller automatically.
Parameters	portName，Dobot port name;As for serial port, maybe is COM3; Whilefor UDP, maybe is 192.168.0.5 baudrate
Return	DobotConnect_NoError：Connect Dobot controller successfully DobotConnect_NotFound：Dobot controller port is not found DobotConnect_Occupied：Dobot controller port is occupied

Note：In order to make API recognize Dobot controller interface, please install the needed driver in advance, refer to Dobot user manual for detailed information.

2.3.3 Disconnect Dobot

Figure 2.4 Disconnect Dobot interface

Prototype	void DisconnectDobot(void)
Description	Disconnect Dobot controller
Parameter	Void
Return	Void

2.4 Instruction Timeout

As described in Section 2.1, all instructions sent to Dobot controller have Return. When an instruction error occurs due to a communication link interference or any other factors, the controller can be recognized by the Void method and the Void method Return. Therefore, each instruction issued to the controller has a time-out period. The instruction timeout period can be set by the following API.

Figure 2.5 Interface of setup command timeout

Prototype	void SetCmdTimeout(uint32_t cmdTimeout)
Descrption	Setup commands timeout
Parameter	cmdTimeout Unit：ms
Return	Void

2.5 Command queen controlling

Dobot controller support start, stop the execution of quene command.

2.5.1 SetQueuedCmdStartExec

Figure 2.6 Interface of start up commands quene

Prototype	int SetQueuedCmdStartExec(void)
Description	Operation of startup commands
Parameter	Void
Return	DobotCommunicate_NoError: The command is returned normally DobotCommunicate_BufferFull: The command quene is full.(The interface do not return the value) DobotCommunicate_Timeout: The command doesn't return, which results in timeout.

2.5.2 SetQueuedCmdStopExec

Figure 2.7 Operation of stop commands quene interface

Prototype	int SetQueuedCmdStopExec(void)
Description	Stop instruction queue operation. If the current instruction queue is running an instruction, it will stop the instruction queue after finishing running the instruction.
Parameter	Void
Return	DobotCommunicate_NoError: The command returns normally DobotCommunicate_BufferFull: The command quene is full (The interface will not return the value) DobotCommunicate_Timeout: The command has no return,which resultsin timeout.

2.5.3 SetQueuedCmdForceStopExec

Figure 2.8 The interface description of SetQueuedCmdForceStopExec

Prototype	int SetQueuedCmdForceStopExec(void)
Description	Stop instruction queue operation forcely.No matter whether the instruction quene is running an instruction or not,the controller will force it to stop running.
Parameter	Void
Return	DobotCommunicate_NoError: The instruction returns normally. DobotCommunicate_BufferFull: The instruction queue is full. (The interface does not return this value) DobotCommunicate_Timeout: The instruction did not return, resulting in a timeout.

2.5.4 SetQueuedCmdStartDownload

Dobot controller supports storing instructions in the controller's external Flash, and then it can be triggered by pressing the keys on the controller, that is, the offline function.

The general flow of instructions to download is:

1. Call the start instruction queue to download API. Where totalLoop is the total number of the instruction is run offline;

2. Send queue instructions;

3. Repeat until the queue instruction transmission is complete;

4. Call the Stop Instruction Queue to download API.

Figure 2.9 The interface description of SetQueuedCmdStartDownload

Prototype	int SetQueuedCmdStartDownload(uint32_t totalLoop, uint32_t linePerLoop)
Description	SetQueuedCmdStartDownload
Parameter	totalLoop: the total number of running offline linePerLoop: The number of per cycle instruction
Return	DobotCommunicate_NoError: The instruction returns normally. DobotCommunicate_BufferFull: The instruction queue is full. (The interface does not return this value) DobotCommunicate_Timeout: The instruction did not return, resulting in a timeout.

2.5.5 Complete the instruction queue download.

Figure 2.10 The interface instruction of completing the instruction queue download

Prototype	int SetQueuedCmdStopDownload(void)
Description	Complete the instruction queue download.
Parameter	Void
Return	DobotCommunicate_NoError: The instruction returns normally. DobotCommunicate_BufferFull: The instruction queue is full. (The interface does not return this value) DobotCommunicate_Timeout: The instruction did not return, resulting in a timeout.

2.5.6 SetQueuedCmdClear

This interface can clear the instruction queue buffered in Dobot controller.

Figure 2.11 SetQueuedCmdClear

Prototype	int SetQueuedCmdClear(void)
Description	Clear command quene
Parameter	Void
Return	DobotCommunicate_NoError: The instruction returns normally. DobotCommunicate_BufferFull: The instruction queue is full. (The interface does not return this value) DobotCommunicate_Timeout: The instruction did not return, resulting in a timeout.

2.5.7 GetQueuedCmdCurrentIndex

In Dobot controller instruction queue mechanism, there is a 64-bit internal count index. Whenever the controller executes a command, the counter is automatically incremented. With this internal index, you can query how many queue instructions the controller has executed and the instructions that have been executed. (Indicate the progress of the run)

Figure 2.12 The interface description of GetQueuedCmdCurrentIndex

Prototype	int GetQueuedCmdCurrentIndex(uint64_t *queuedCmdCurrentIndex)
Description	GetQueuedCmdCIndex Executed
Parameter	queuedCmdCurrentIndex:Quene command index variable pointers;
Return	DobotCommunicate_NoError: The instruction returns normally. DobotCommunicate_BufferFull: The instruction queue is full. (The interface does not return this value) DobotCommunicate_Timeout: The instruction did not return, resulting in a timeout.

2.6 Device Information

2.6.1 Set the device serial number

Figure 2.13 The interface instruction of setting the device serial number

Prototype	int SetDeviceSN(const char *deviceSN)
Description	Set the device serial number
Parameter	deviceSN:Device serial number string pointer.
Return	DobotCommunicate_NoError: The instruction returns normally. DobotCommunicate_BufferFull: The instruction queue is full. (The interface does not return this value) DobotCommunicate_Timeout: The instruction did not return, resulting in a timeout.

NOTE: The device serial number setting interface is only valid when shipped out (requires special password).

2.6.2 Get the device serial number

Figure 2.14 Get the device serial numberThe interface description of

Prototype	int GetDeviceSN(char *deviceSN, uint32_t maxLen)
Description	Get the device serial number
Parameter	deviceSN:Device serial number string pointer. maxLen:Incoming external buffer length to avoid overflow.
Return	DobotCommunicate_NoError: The instruction returns normally. DobotCommunicate_BufferFull: The instruction queue is full. (The interface does not return this value) DobotCommunicate_Timeout: The instruction did not return, resulting in a timeout.

2.6.3 Set the device name

Figure 2.15 The interface instruction of setting the device name

Prototype	int SetDeviceName(const char *deviceName)
Description	Set the device name. When there are multiple machines, you can use this interface to set the device name for distinction.
Parameter	deviceName: Device name string pointer
Return	DobotCommunicate_NoError: The instruction returns normally. DobotCommunicate_BufferFull: The instruction queue is full. (The interface does not return this value) DobotCommunicate_Timeout: The instruction did not return, resulting in a timeout.

2.6.4 GetDeviceName

Figure 2.16 The interface description of GetDeviceName

Prototype	int GetDeviceName(char *deviceName, uint32_t maxLen)
Description	GetDeviceName
Parameter	deviceName: Device name string pointer maxLen:Incoming external buffer length to avoid overflow.
Return	DobotCommunicate_NoError: The instruction returns normally. DobotCommunicate_BufferFull: The instruction queue is full. (The interface does not return this value) DobotCommunicate_Timeout: The instruction did not return, resulting in a timeout.

2.6.5 GetDeviceVersion

Figure 2.17 2.6.5 The interface instruction of getting the device version number

Prototype	int GetDeviceVersion(uint8_t *majorVersion, uint8_t *minorVersion, uint8_t *revision)
Description	Get equipment version information
Parameter	majorVersion minorVersion revision
Return	DobotCommunicate_NoError: The instruction returns normally. DobotCommunicate_BufferFull: The instruction queue is full. (The interface does not return this value) DobotCommunicate_Timeout: The instruction did not return, resulting in a timeout.

2.7 Real-time pose

In DobotV2.0, Dobot controller calculates the real-time pose initial value based on the following information:

- Encoder position (can be obtained by Home);
- Rear arm angle sensor (power on or by the arm when press UNLOCK button);

- Forearm angle sensor(power on or by the arm when press UNLOCK button).

Then in the later process of controlling Dobot, Dobot controller will update real-time pose based on real-time pose initial value.

2.7.1 Get real-time pose

Figure 2.18 The interface description of getting real-time pose

Prototype	int GetPose(Pose *pose)
Description	Get real-time pose
Parameter	Pose definition： typedef struct tagPose { float x; float y; float z; float r; float jointAngle[4]; }Pose; pose:Real-time pose pointer
Return	DobotCommunicate_NoError: The instruction returns normally. DobotCommunicate_BufferFull: The instruction queue is full. (The interface does not return this value) DobotCommunicate_Timeout: The instruction did not return, resulting in a timeout.

2.7.2 Reset real-time pose

The real-time pose (initial value) can be reset in the following cases:

- Angle sensor is damaged, which must rely on the external angle measurement means;
- Angle sensor accuracy is too poor, which need an external angle measurement means to confirm the exact value directly or indirectly.

Figure 2.19 The interface description of setting initial pose

Prototype	int ResetPose(bool manual, float rearArmAngle, float frontArmAngle)
Description	Reset pose
Parameter	manual:If got 0,reset the pose and don’t need incoming rearArmAngle and frontArmAngle；If got 1, need incoming rearArmAngle and frontArmAngle. rearArmAngle; frontArmAngle;
Return	DobotCommunicate_NoError: The instruction returns normally. DobotCommunicate_BufferFull: The instruction queue is full. (The interface does not return this value) DobotCommunicate_Timeout: The instruction did not return, resulting in a timeout.

2.7.3 GetKinematicsParams

Figure 2.20 The interface of getting kinematica parameter

Prototype	int GetKinematics(Kinematcis *kinematics)
Description	Get Kinematics Parameter
Parameter	typedef struct tagKinematics { float velocity; float acceleration; }Kinematics; kinematics:Kinematica Parameter of variable pointers
Return	DobotCommunicate_NoError: The instruction returns normally. DobotCommunicate_BufferFull: The instruction queue is full. (The interface does not return this value) DobotCommunicate_Timeout: The instruction did not return, resulting in a timeout.

2.8 ALARM

2.8.1 GetAlarmsState

Figure 2.21 The interface description of GetAlarmsState

Prototype	int GetAlarmsState(uint8_t *alarmsState, uint32_t *len, uint32_t maxLen)
Description	Gets the system alarm status
Parameter	alarmsState: The first address of the array used to receive the alarm bit. len: The byte occupied by the alarm. maxLen:Incoming external buffer length to avoid overflow.
Return	DobotCommunicate_NoError: The instruction returns normally. DobotCommunicate_BufferFull: The instruction queue is full. (The interface does not return this value) DobotCommunicate_Timeout: The instruction did not return, resulting in a timeout.

Note: For specific alarms and indexes, refer to Dobot User Manual. Each byte in the array alarmsState identifies the alarm status of the eight alarm items, with the MSB (Most Significant Bit) in the high order and LSB (Least Significant Bit) in the low

2.8.2 ClearAllAlarmsState

Figure 2.22 The interface description of ClearAllAlarmsState

Prototype	int ClearAllAlarmsState(void)
Description	Clear all system alarms
Parameter	Void
Return	DobotCommunicate_NoError: The instruction returns normally. DobotCommunicate_BufferFull: The instruction queue is full. (The interface does not return this value) DobotCommunicate_Timeout: The instruction did not return, resulting in a timeout.

2.9 HOME

2.9.1 Set Home Parameter

Figure 2.23 Set Home ParameterThe interface description of

Prototype	int SetHOMEParams(HOMEParams *homeParams, bool isQueued, uint64_t *queuedCmdIndex)
Description	Set Home Parameter
Parameter	HOMEParams typedef struct tagHOMEParams { float x; float y; float z; float r; }HOMEParams; homeParams:Hoem Parameter variable pointers isQueued:Whether to specify this instruction as a queue command. queuedCmdIndex:Quene command index (The Controller Return)
Return	DobotCommunicate_NoError: The instruction returns normally. DobotCommunicate_BufferFull: The instruction quene is full. DobotCommunicate_Timeout: The instruction did not return, resulting in a timeout.

Note: The homing function may not be supported on some models. For details, refer to the user manual.

2.9.2 Get Home Parameter

Figure 2.24 Get HomeParameterThe interface description of

Prototype	int GetHOMEParams(HOMEParams *homeParams)
Description	Get HomeParameter
Parameter	HOMEParams typedef struct tagHOMEParams { float x; float y; float z; float r; }HOMEParams; homeParams:回零 Parameter variable pointers
Return	DobotCommunicate_NoError: The instruction returns normally. DobotCommunicate_BufferFull: The instruction queue is full. (The interface does not return this value) DobotCommunicate_Timeout: The instruction did not return, resulting in a timeout.

Note: The homing function may not be supported on some models. For details, refer to the user manual.

2.9.3 Execute HOME function

Figure 2.25 The interface description of executing home function

Prototype	int SetHOMECmd(HOMECmd *homeCmd, bool isQueued, uint64_t *queuedCmdIndex)
Description	Execute HOME function

Parameter	HOMECmd
	typedef struct tagHOMECmd {
	uint32_t reserved; // Reserved for future use
	}HOMECmd;
	
	homeCmd:Home command variable pointers
	isQueued:Whether to specify this instruction as a queue command.
	queuedCmdIndex:Quene command index (The Controller Return)

Return	DobotCommunicate_NoError: The instruction returns normally. DobotCommunicate_BufferFull: The instruction quene is full. DobotCommunicate_Timeout: The instruction did not return, resulting in a timeout.

Note: The homing function may not be supported on some models. For details, refer to the user manual.

2.10 HHT

2.10.1 Set trigger mode

Figure 2.26 Set handhold teaching trigger mode

Prototype	int SetHHTTrigMode (HHTTrigMode hhtTrigMode)
Description	Set handhold teaching trigger mode of saved points
Parameter	typedef enum tagHHTTrigMode { TriggeredOnKeyReleased, // Update when releasing the key TriggeredOnPeriodicInterval // Timed trigger }HHTTrigMode; hhtTrigMode:After pressing UNLOCK, get handhold teaching trigger mode
Return	DobotCommunicate_NoError: The instruction returns normally. DobotCommunicate_BufferFull: The instruction queue is full. (The interface does not return this value) DobotCommunicate_Timeout: The instruction did not return, resulting in a timeout.

2.10.2 Get trigger mode

Figure 2.27 The interface description of getting handhold teaching trigger mode

Prototype	int GetHHTTrigMode (HHTTrigMode *hhtTrigMode)
Description	Get handhold teaching trigger mode of saved points
Parameter	typedef enum tagHHTTrigMode { TriggeredOnKeyReleased, // Update when releasing the key TriggeredOnPeriodicInterval // Timed trigger }HHTTrigMode; hhtTrigMode: After pressing UNLOCK, get handhold teaching trigger mode
Return	DobotCommunicate_NoError: The instruction returns normally. DobotCommunicate_BufferFull: The instruction queue is full. (The interface does not return this value) DobotCommunicate_Timeout: The instruction did not return, resulting in a timeout.

2.10.3 SetHHTTrigOutputEnabled/Disabled

Figure 2.28 The interface description of setting trigger output enabled or deiabled

Prototype	int SetHHTTrigOutputEnabled (bool isEnabled)
Description	The status of SetHHTTrigOutputEnabled
Parameter	isEnabled: The status of SetHHTTrigOutputEnabled
Return	DobotCommunicate_NoError: The instruction returns normally. DobotCommunicate_BufferFull: The instruction queue is full. (The interface does not return this value) DobotCommunicate_Timeout: The instruction did not return, resulting in a timeout.

2.10.4 GetHHTTrigOutputEnabled/Disabled

Figure 2.29 The interface description of GetHHTTrigOutputEnabled or disabled

Prototype	int GetHHTTrigOutputEnabled (bool *isEnabled)
Description	The status of GetHHTTrigOutputEnabled
Parameter	isEnabled: The status GetHHTTrigOutputEnabled
Return	DobotCommunicate_NoError: The instruction returns normally. DobotCommunicate_BufferFull: The instruction queue is full. (The interface does not return this value) DobotCommunicate_Timeout: The instruction did not return, resulting in a timeout.

2.10.5 Get trigger output

Figure 2.30 The interface description of getting trigger output

Prototype	int GetHHTTrigOutput(bool *isTriggered)
Description	Get the status of trigger output
Parameter	isTriggered:Trigger output status
Return	DobotCommunicate_NoError: The instruction returns normally. DobotCommunicate_BufferFull: The instruction queue is full. (The interface does not return this value) DobotCommunicate_Timeout: The instruction did not return, resulting in a timeout.

2.11 End-effector

2.11.1 Set EndEffectorParams

Figure 2.31 Set end-effector parameter

Prototype	int SetEndEffectorParams(EndEffectorParams *endEffectorParams, bool isQueued, uint64_t *queuedCmdIndex)
Description	Set end-effector parameter
Parameter	EndEffectorParams： typedef struct tagEndEffectorParams { float xBias; float yBias; float zBias; }EndEffectorParams; endEffectorParams:End-effector parameter pointer isQueued:Whether to specify this instruction as a queue command. queuedCmdIndex:Quene command index(The Controller Return)
Return	DobotCommunicate_NoError: The instruction returns normally. DobotCommunicate_BufferFull: The instruction quene is full. DobotCommunicate_Timeout: The instruction did not return, resulting in a timeout.

Note: When using a standard end-effector, refer to Dobot User Manual for the X-axis and Y-axis offset and call up the interface settings. In other cases, the end of the actuator Parameter, you need to confirm the structure Parameter.

2.11.2 GetEndEffectorParams

Figure 2.32 The interface description of GetEndEffectorParams

Prototype	int GetEndEffectorParams(EndEffectorParams *endEffectorParams)
Description	GetEndEffectorParams
Parameter	EndEffectorParams： typedef struct tagEndEffectorParams { float xBias; float yBias; float zBias; }EndEffectorParams; endEffectorParams:End-effector parameter pointer
Return	DobotCommunicate_NoError: The instruction returns normally. DobotCommunicate_BufferFull: The instruction queue is full. (The interface does not return this value) DobotCommunicate_Timeout: The instruction did not return, resulting in a timeout.

2.11.3 SetEndEffectorLaser Output

Figure 2.33 The interface description of SetEndEffectorLaser output

Prototype	int SetEndEffectorLaser(bool enableCtrl, bool on, bool isQueued, uint64_t *queuedCmdIndex)
Description	Set laser on/off
Parameter	enableCtrl:Enable to control on:If the laser is on isQueued:Whether to specify this instruction as a queue command. queuedCmdIndex:Quene command index(The Controller Return)
Return	DobotCommunicate_NoError: The instruction returns normally. DobotCommunicate_BufferFull: The instruction queue is full. (The interface does not return this value) DobotCommunicate_Timeout: The instruction did not return, resulting in a timeout.

2.11.4 GetEndEffectorLaser Output

Figure 2.34 The interface description of GetEndEffectorLaser output

Prototype	int GetEndEffectorLaser(bool *isCtrlEnabled, bool *isOn)
Description	Get status of laser on/off
Parameter	isCtrlEnabled: If the control is enabled isOn:If the laser is on.
Return	DobotCommunicate_NoError: The instruction returns normally. DobotCommunicate_BufferFull: The instruction queue is full. (The interface does not return this value) DobotCommunicate_Timeout: The instruction did not return, resulting in a timeout.

2.11.5 SetEndEffectorSuctionCup Output

Figure 2.35 The interface description of SetEndEffectorSuctionCup output

Prototype	int SetEndEffectorSuctionCup(bool enableCtrl, bool suck, bool isQueued, uint64_t *queuedCmdIndex)
Description	Set the suction cup is sucked or realeased
Parameter	enableCtrl:Enable to control suck:If it is sucked isQueued:Whether to specify this instruction as a queue command. queuedCmdIndex:Quene command index(The Controller Return)
Return	DobotCommunicate_NoError: The instruction returns normally. DobotCommunicate_BufferFull: The instruction queue is full. (The interface does not return this value) DobotCommunicate_Timeout: The instruction did not return, resulting in a timeout.

2.11.6 GetEndEffectorSuctionCup Output

Figure 2.36 The interface description of GetEndEffectorSuctionCup output

Prototype	int GetEndEffectorSuctionCup(bool *isCtrlEnabled, bool *isSucked)
Description	Get the status of the suction sucking or releasing
Parameter	isCtrlEnabled:If the control is enabled isSucked:If the suction is sucked
Return	DobotCommunicate_NoError: The instruction returns normally. DobotCommunicate_BufferFull: The instruction queue is full. (The interface does not return this value) DobotCommunicate_Timeout: The instruction did not return, resulting in a timeout.

2.11.7 Set gripper output

Figure 2.37 Set gripper output interface

Prototype	int SetEndEffectorGripper(bool enableCtrl, bool grip, bool isQueued, uint64_t *queuedCmdIndex)
Description	Set gripper to caught or release
Parameter	enableCtrl:Enable to control grip:If caught isQueued:Whether to specify this instruction as a queue command. queuedCmdIndex:Quene command index(The Controller Return)
Return	DobotCommunicate_NoError: The instruction returns normally. DobotCommunicate_BufferFull: The instruction queue is full. (The interface does not return this value) DobotCommunicate_Timeout: The instruction did not return, resulting in a timeout.

2.11.8 Get gripper output

Figure 2.38 Get gripper output interface

Prototype	int GetEndEffectorGripper(bool *isCtrlEnabled, bool *isGripped)
Description	Get the condition of gripper
Parameter	isCtrlEnabled:If the control is enabled isGripped:If the gripper is caught
Return	DobotCommunicate_NoError: The instruction returns normally. DobotCommunicate_BufferFull: The instruction queue is full. (The interface does not return this value) DobotCommunicate_Timeout: The instruction did not return, resulting in a timeout.

2.12 ARM Orientation

2.12.1 SetArmOrientation

Figure 2.39 The interface description of SetArmOrientation

Prototype	int SetArmOrientation(ArmOrientation armOrientation, bool isQueued, uint64_t *queuedCmdIndex)
Description	SetArmOrientation
Parameter	ArmOrientation： typedef enum tagArmOrientation { LeftyArmOrientation, RightyArmOrientation }ArmOrientation; armOrientation isQueued:Whether to specify this instruction as a queue command. queuedCmdIndex:Quene command index(The Controller Return)
Return	DobotCommunicate_NoError: The instruction returns normally. DobotCommunicate_BufferFull: The instruction quene is full. DobotCommunicate_Timeout: The instruction did not return, resulting in a timeout.

Note: This command is currently only applicable to SCARA models.

2.12.2 GetArmOrientation

Figure 2.40 The interface description of GetArmOrientation

Prototype	int GetArmOrientation(ArmOrientation *armOrientation)
Description	GetArmOrientation
Parameter	ArmOrientation: typedef enum tagArmOrientation { LeftyArmOrientation, RightyArmOrientation }ArmOrientation; armOrientation: Arm direction variable pointer
Return	DobotCommunicate_NoError: The instruction returns normally. DobotCommunicate_BufferFull: The instruction quene is full. DobotCommunicate_Timeout: The instruction did not return, resulting in a timeout.

2.13 JOG

2.13.1 SetJOGJointParams

Figure 2.41 The interface description of SetJOGJointParams

Prototype	int SetJOGJointParams(JOGJointParams *jogJointParams, bool isQueued, uint64_t *queuedCmdIndex)
Description	SetJOGJointParams
Parameter	JOGJointParams: typedef struct tagJOGJointParams { float velocity[4]; float acceleration[4]; }JOGJointParams; jogJointParams:Jogjoint Parameter variable pointer isQueued:Whether to specify this instruction as a queue command. queuedCmdIndex:Quene command index(The Controller Return)
Return	DobotCommunicate_NoError: The instruction returns normally. DobotCommunicate_BufferFull: The instruction quene is full. DobotCommunicate_Timeout: The instruction did not return, resulting in a timeout.

2.13.2 GetJOGJointParams

Figure 2.42 The interface description of GetJOGJointParams

Prototype	int GetJOGJointParams(JOGJointParams *jogJointParams)
Description	GetJOGJointParams
Parameter	JOGJointParams: typedef struct tagJOGJointParams { float velocity[4]; float acceleration[4]; }JOGJointParams; jogJointParams: Jogjoint Parameter variable pointer
Return	DobotCommunicate_NoError: The instruction returns normally. DobotCommunicate_BufferFull: The instruction queue is full. (The interface does not return this value) DobotCommunicate_Timeout: The instruction did not return, resulting in a timeout.

2.13.3 SetJOGCoordinateParams

Figure 2.43 The interface description of SetJOGCoordinateParams

Prototype	int SetJOGCoordinateParams(JOGCoordinateParams *jogCoordinateParams, bool isQueued, uint64_t *queuedCmdIndex)
Description	SetJOGCoordinateParams
Parameter	JOGCoordinateParams: typedef struct tagJOGCoordinateParams { float velocity[4]; float acceleration[4]; }JOGCoordinateParams; jogCoordinateParams:JogCoordinate Parameter variable pointer isQueued:Whether to specify this instruction as a queue command. queuedCmdIndex:Quene command index(The Controller Return)
Return	DobotCommunicate_NoError: The instruction returns normally. DobotCommunicate_BufferFull: The instruction quene is full. DobotCommunicate_Timeout: The instruction did not return, resulting in a timeout.

2.13.4 GetJOGCoordinateParams

Figure 2.44 ParameterThe interface description of GetJOGCoordinateParams

Prototype	int GetJOGCoordinateParams(JOGCoordinateParams *jogCoordinateParams)
Description	GetJOGCoordinateParams
Parameter	JOGCoordinateParams: typedef struct tagJOGCoordinateParams { float velocity[4]; float acceleration[4]; }JOGCoordinateParams; jogCoordinateParams: JogCoordinate Parameter variable pointer
Return	DobotCommunicate_NoError: The instruction returns normally. DobotCommunicate_BufferFull: The instruction queue is full. (The interface does not return this value) DobotCommunicate_Timeout: The instruction did not return, resulting in a timeout.

2.13.5 SetJOGCommonParams Parameter

Figure 2.45 The interface description of SetJOGCommonParams

Prototype	int SetJOGCommonParams(JOGCommonParams *jogCommonParams, bool isQueued, uint64_t *queuedCmdIndex)
Description	SetJOGCommonParams
Parameter	JOGCommonParams: typedef struct tagJOGCommonParams { float velocityRatio; float accelerationRatio; }JOGCommonParams; jogCommonParams:JOGCommonParams variable pointer isQueued:Whether to specify this instruction as a queue command. queuedCmdIndex:Quene command index(The Controller Return)
Return	DobotCommunicate_NoError: The instruction returns normally. DobotCommunicate_BufferFull: The instruction quene is full. DobotCommunicate_Timeout: The instruction did not return, resulting in a timeout.

2.13.6 GetJOGCommonParams

Figure 2.46 The interface description of GetJOGCommonParams

Prototype	int GetJOGCommonParams(JOGCommonParams *jogCommonParams)
Description	GetJOGCommonParams
Parameter	JOGCommonParams: typedef struct tagJOGCommonParams { float velocityRatio; float accelerationRatio; }JOGCommonParams; jogCommonParams: JOGCommonParams variable pointer
Return	DobotCommunicate_NoError: The instruction returns normally. DobotCommunicate_BufferFull: The instruction queue is full. (The interface does not return this value) DobotCommunicate_Timeout: The instruction did not return, resulting in a timeout.

2.13.7 Execute JOG command

Figure 2.47 The interface description of executing JOGCmd

Prototype	int SetJOGCmd(JOGCmd *jogCmd, bool isQueued, uint64_t *queuedCmdIndex)
Description	Execute JOG command
Parameter	JOGCmd typedef struct tagJOGCmd { uint8_t isJoint; uint8_t cmd; }JOGCmd; jogCmd:Jog command variable pointers isQueued:Whether to specify this instruction as a queue command. queuedCmdIndex:Quene command index(The Controller Return)
Return	DobotCommunicate_NoError: The instruction returns normally. DobotCommunicate_BufferFull: The instruction quene is full. DobotCommunicate_Timeout: The instruction did not return, resulting in a timeout.

2.14 PTP

2.14.1 SetPTPJointParams

Figure 2.48 The interface description of SetPTPJointParams

Prototype	int SetPTPJointParams(PTPJointParams *ptpJointParams, bool isQueued, uint64_t *queuedCmdIndex)
Description	SetPTPJointParams
Parameter	PTPJointParams: typedef struct tagPTPJointParams { float velocity[4]; float acceleration[4]; }PTPJointParams; ptpJointParams:PTPJointParams variable pointers isQueued:Whether to specify this instruction as a queue command. queuedCmdIndex:Quene command index(The Controller Return)
Return	DobotCommunicate_NoError: The instruction returns normally. DobotCommunicate_BufferFull: The instruction quene is full. DobotCommunicate_Timeout: The instruction did not return, resulting in a timeout.

2.14.2 GetPTPJointParams

Figure 2.49 The interface description of GetPTPJointParams

Prototype	int GetPTPJointParams(PTPJointParams *ptpJointParams)
Description	GetPTPJointParams
Parameter	PTPJointParams typedef struct tagPTPJointParams { float jointVelocity[4]; float jointAcceleration[4]; }PTPJointParams; ptpJointParams:PTPJointParams variable pointer
Return	DobotCommunicate_NoError: The instruction returns normally. DobotCommunicate_BufferFull: The instruction queue is full. (The interface does not return this value) DobotCommunicate_Timeout: The instruction did not return, resulting in a timeout.

2.14.3 SetPTPCoordinateParams

Figure 2.50 The interface description of SetPTPCoordinateParams

Prototype	int SetPTPCoordinateParams(PTPCoordinateParams *ptpCoordinateParams, bool isQueued, uint64_t *queuedCmdIndex)
Description	SetPTPCoordinateParams
Parameter	PTPSpeedParams: typedef struct tagPTPCoordinateParams { float xyzVelocity; float rVelocity; float xyzAcceleration; float rAcceleration; }PTPCoordinateParams; ptpCoordinateParams:PTPCoordinateParams variable pointers isQueued:Whether to specify this instruction as a queue command. queuedCmdIndex:Quene command index(The Controller Return)
Return	DobotCommunicate_NoError: The instruction returns normally. DobotCommunicate_BufferFull: The instruction quene is full. DobotCommunicate_Timeout: The instruction did not return, resulting in a timeout.

2.14.4 GetPTPCoordinateParams

Figure 2.51 The interface description of GetPTPCoordinateParams

Prototype	int GetPTPCoordinateParams(PTPCoordinateParams *ptpCoordinateParams)
Description	GetPTPCoordinateParams
Parameter	PTPCoordinateParams: typedef struct tagPTPCoordinateParams { float xyzVelocity; float rVelocity; float xyzAcceleration; float rAcceleration; }PTPCoordinateParams; ptpCoordinateParams: PTPCoordinateParams variable pointers
Return	DobotCommunicate_NoError: The instruction returns normally. DobotCommunicate_BufferFull: The instruction queue is full. (The interface does not return this value) DobotCommunicate_Timeout: The instruction did not return, resulting in a timeout.

2.14.5 SetPTPJumpMode

Figure 2.52 The interface description of SetPTPJumpParams

Prototype	int SetPTPJumpParams(PTPJumpParams *ptpJumpParams, bool isQueued, uint64_t *queuedCmdIndex)
Description	SetPTPJumpMode
Parameter	PTPJumpParams typedef struct tagPTPJumpParams { float jumpHeight; float zLimit; }PTPJumpParams; ptpJumpParams:PTPJumpMode Parameter variable pointers isQueued:Whether to specify this instruction as a queue command. queuedCmdIndex:Quene command index(The Controller Return)
Return	DobotCommunicate_NoError: The instruction returns normally. DobotCommunicate_BufferFull: The instruction quene is full. DobotCommunicate_Timeout: The instruction did not return, resulting in a timeout.

2.14.6 GetPTPJumpMode

Figure 2.53 The interface description of GetPTPJumpParams

Prototype	int GetPTPJumpParams(PTPJumpParams *ptpJumpParams)
Description	GetPTPJumpParams
Parameter	PTPJumpParams typedef struct tagPTPJumpParams { float jumpHeight; float zLimit; }PTPJumpParams; ptpJumpParams: PTP Jump Mode Parameter variable pointers
Return	DobotCommunicate_NoError: The instruction returns normally. DobotCommunicate_BufferFull: The instruction queue is full. (The interface does not return this value) DobotCommunicate_Timeout: The instruction did not return, resulting in a timeout.

2.14.7 SetPTPCommonParams

Figure 2.54 The interface description of SetPTPCommonParams

Prototype	int SetPTPCommonParams(PTPCommonParams *ptpCommonParams, bool isQueued, uint64_t *queuedCmdIndex)
Description	SetPTPCommonParams
Parameter	PTPCommonParams typedef struct tagPTPCommonParams { float velocityRatio; float accelerationRatio; }PTPCommonParams; ptpCommonParams:PTPCommonParameter variable pointers isQueued:Whether to specify this instruction as a queue command. queuedCmdIndex:Quene command index(The Controller Return)
Return	DobotCommunicate_NoError: The instruction returns normally. DobotCommunicate_BufferFull: The instruction quene is full. DobotCommunicate_Timeout: The instruction did not return, resulting in a timeout.

2.14.8 GetPTPCommonParams

Figure 2.55 The interface description of GetPTPCommonParams

Prototype	int GetPTPCommonParams(PTPCommonParams *ptpCommonParams)
Description	Get PTP speed parameter
Parameter	PTPCommonParams typedef struct tagPTPCommonParams { float velocityRatio; float accelerationRatio; }PTPCommonParams; ptpCommonParams:PTP Common Parameter variable pointers
Return	DobotCommunicate_NoError: The instruction returns normally. DobotCommunicate_BufferFull: The instruction queue is full. (The interface does not return this value) DobotCommunicate_Timeout: The instruction did not return, resulting in a timeout.

2.14.9 SetPTPCmd

Figure 2.56 The interface description of SetPTPCmd

Prototype	int SetPTPCmd(PTPCmd *ptpCmd, bool isQueued, uint64_t *queuedCmdIndex)
Description	SetPTPCmd
Parameter	PTPCmd typedef struct tagPTPCmd { uint8_t ptpMode; float x; float y; float z; float r; }PTPCmd; ptpCmd:PTP 命令 variable pointers isQueued:Whether to specify this instruction as a queue command. queuedCmdIndex:Quene command index(The Controller Return)
Return	DobotCommunicate_NoError: The instruction returns normally. DobotCommunicate_BufferFull: The instruction quene is full. DobotCommunicate_Timeout: The instruction did not return, resulting in a timeout.

2.15 CP

2.15.1 SetCPParams

Figure 2.57 The interface description of SetCPParams

Prototype	int SetCPParams(CPParams *cpParams, bool isQueued, uint64_t *queuedCmdIndex)
Description	SetCPParams

Parameter	CPParams
	typedef struct tagCPParams {
	float planAcc;
	float junctionVel;
	union {
	float acc; // realTimeTrack = false
	float period; // realTimeTrack = true
	};
	uint8_t realTimeTrack;
	}CPParams;
	
	cpParams:CPParams variable pointers
	isQueued:Whether to specify this instruction as a queue command.
	queuedCmdIndex:Quene command index(The Controller Return)

Return	DobotCommunicate_NoError: The instruction returns normally. DobotCommunicate_BufferFull: The instruction quene is full. DobotCommunicate_Timeout: The instruction did not return, resulting in a timeout.

2.15.2 GetCPParams

Figure 2.58 The interface description of GetCPParams

Prototype	int GetCPParams(CPParams *cpParams)
Description	GetCPParams

Parameter	CPParams
	typedef struct tagCPParams {
	float planAcc;
	float junctionVel;
	union {
	float acc; // realTimeTrack = false
	float period; // realTimeTrack = true
	};
	uint8_t realTimeTrack;
	}CPParams;
	
	cpParams: CPParams variable pointers

Return	DobotCommunicate_NoError: The instruction returns normally. DobotCommunicate_BufferFull: The instruction queue is full. (The interface does not return this value) DobotCommunicate_Timeout: The instruction did not return, resulting in a timeout.

2.15.3 SetCPCmd

Figure 2.59 The interface description of SetCPCmd

Prototype	int SetCPCmd(CPCmd *cpCmd, bool isQueued, uint64_t *queuedCmdIndex)
Description	SetCPCmd
Parameter	CPCmd typedef struct tagCPCmd { uint8_t cpMode; float x; float y; float z; float velocity; }CPCmd; cpCmd: CPCmd variable pointers isQueued:Whether to specify this instruction as a queue command. queuedCmdIndex:Quene command index(The Controller Return)
Return	DobotCommunicate_NoError: The instruction returns normally. DobotCommunicate_BufferFull: The instruction quene is full. DobotCommunicate_Timeout: The instruction did not return, resulting in a timeout.

注：当指令队列中有多条连续的 CP 指令时，Dobot 控制器将自动前瞻。前瞻的条件是，队列中这些 CP 指令之间没有 JOG、PTP、ARC、WAIT、TRIG 等指令。

2.16 ARC

1. Set circular interpolation parameter

Figure 2.60 The interface description of circular interpolation parameter

Prototype	int SetARCParams(ARCParams *arcParams, bool isQueued, uint64_t *queuedCmdIndex)
Description	Set circular interpolation Parameter
Parameter	ARCParams typedef struct tagARCParams { float xyzVelocity; float rVelocity; float xyzAcceleration; float rAcceleration; }ARCParams; arcParams:Circular interpolation Parameter variable pointers isQueued:Whether to specify this instruction as a queue command. queuedCmdIndex:Quene command index(The Controller Return)
Return	DobotCommunicate_NoError: The instruction returns normally. DobotCommunicate_BufferFull: The instruction quene is full. DobotCommunicate_Timeout: The instruction did not return, resulting in a timeout.

2.16.2 Get circular interpolation parameter

Figure 2.61 The interface description of getting circular interpolation parameter

Prototype	int GetARCParams(ARCParams *arcParams)
Description	Get circular interpolation parameter
Parameter	ARCParams typedef struct tagARCParams { float xyzVelocity; float rVelocity; float xyzAcceleration; float rAcceleration; }ARCParams; arcParams:Circular interpolation functionParameter variable pointers
Return	DobotCommunicate_NoError: The instruction returns normally. DobotCommunicate_BufferFull: The instruction queue is full. (The interface does not return this value) DobotCommunicate_Timeout: The instruction did not return, resulting in a timeout.

2.16.3 Perform circular interpolation function

Figure 2.62 The interface description of performing circular interpolation function

Prototype	int SetARCCmd(ARCCmd *arcCmd, bool isQueued, uint64_t *queuedCmdIndex)
Description	Perform circular interpolation function

Parameter	ARCCmd
	typedef struct tagARCCmd {
	struct {
	float x;
	float y;
	float z;
	float r;
	}cirPoint;
	struct {
	float x;
	float y;
	float z;
	float r;
	}toPoint;
	}ARCCmd;
	
	arcCmd:Circular interpolation function variable pointers
	isQueued:Whether to specify this instruction as a queue command.
	queuedCmdIndex:Quene command index(The Controller Return)

Return	DobotCommunicate_NoError: The instruction returns normally. DobotCommunicate_BufferFull: The instruction quene is full. DobotCommunicate_Timeout: The instruction did not return, resulting in a timeout.

2.17 WAIT

2.17.1 Perform wait function

Figure 2.63 The interface description of performing wait function

Prototype	int SetWAITCmd(WAITCmd *waitCmd, bool isQueued, uint64_t *queuedCmdIndex)
Description	Perform wait function

Parameter	WAITCmd：
	typedef struct tagWAITCmd {
	uint32_t timeout; // Unit:ms
	}WAITCmd;
	
	waitCmd:Variable pointers of wait function
	isQueued:Whether to specify this instruction as a queue command.
	queuedCmdIndex:Quene command index(The Controller Return)

Return	DobotCommunicate_NoError: The instruction returns normally. DobotCommunicate_BufferFull: The instruction quene is full. DobotCommunicate_Timeout: The instruction did not return, resulting in a timeout.

Note: This instruction can only be used as a queue instruction, isQueued must be set to true. Setting this command to immediate command may cause a time-out change in the WAIT queue instruction being executed.

2.18 TRIG

2.18.1 Execute the trigger function

Figure 2.64 The interface description of executing the trigger function

Prototype	int SetTRIGCmd(TRIGCmd *trigCmd, bool isQueued, uint64_t *queuedCmdIndex)
Description	Execute the trigger function
Parameter	TRIGCmd typedef struct tagTRIGCmd { uint8_t address; uint8_t mode; uint16_t threshold; }TRIGCmd; trigCmd:Trigger variable pointers isQueued:Whether to specify this instruction as a queue command. queuedCmdIndex:Quene command index(The Controller Return)
Return	DobotCommunicate_NoError: The instruction returns normally. DobotCommunicate_BufferFull: The instruction quene is full. DobotCommunicate_Timeout: The instruction did not return, resulting in a timeout.

Note: This instruction can only be used as a queue instruction, isQueued must be set to true. Setting this command to immediate command may cause a time-out change in the Trigger queue instruction being executed.。

2.19 EIO

In Dobot controller, all extensible I/O is addressed, here you can see as follows:

- High-low level output;
- PWM output;
- Read High-low level output;
- Read analog-digital conversion value output.

Some I/O may have all the functions above. You need configure I/O multiplex when use different functions.

2.19.1 Set I/O multiplex

Figure 2.65 Set I/O multiplex

Prototype	int SetIOMultiplexing(IOMultiplexing ioMultiplexing, bool isQueued, uint64_t *queuedCmdIndex)
Description	Set I/O multiplex

Parameter	IOMultiplexing
	typedef struct tagIOMultiplexing {
	uint8_t address;
	uint8_t multiplex;
	}IOMultiplexing;
	
	Among them, the values supported by mutiplex are as follows:
	typedef enum tagIOFunction {
	IOFunctionDO,
	IOFunctionPWM,
	IOFunctionDI,
	IOFunctionADC
	}IOFunction;
	
	ioMultiplexing:I/O multiplex variables
	isQueued:Whether to specify this instruction as a queue command.
	queuedCmdIndex:Quene command index(The Controller Return)

Return	DobotCommunicate_NoError: The instruction returns normally. DobotCommunicate_BufferFull: The instruction quene is full. DobotCommunicate_Timeout: The instruction did not return, resulting in a timeout.

2.19.2 Read I/O multiplex

Figure 2.66 The interface description of reading I/O multiplex

Prototype	int GetIOMultiplexing(IOMultiplexing *ioMultiplexing)
Description	Read I/O multiplex

Parameter	IOMultiplexing
	typedef struct tagIOMultiplexing {
	uint8_t address;
	uint8_t multiplex;
	}IOMultiplexing;
	
	Among them, the values supported by mutiplex are as follows:
	typedef enum tagIOFunction {
	IOFunctionDO,
	IOFunctionPWM,
	IOFunctionDI,
	IOFunctionADC
	}IOFunction;
	
	ioMultiplexing:I/O multiplex variable pointers

Return	DobotCommunicate_NoError: The instruction returns normally. DobotCommunicate_BufferFull: The instruction queue is full. (The interface does not return this value) DobotCommunicate_Timeout: The instruction did not return, resulting in a timeout.

2.19.3 Set I/O level output

Figure 2.67 The interface description of setting I/O level output

Prototype	int SetIODO(IODO *ioDO, bool isQueued, uint64_t *queuedCmdIndex)
Description	Set I/O level output
Parameter	IODO typedef struct tagIODO { uint8_t address; uint8_t level; }IODO; ioDO:I/O output of level structure variable pointers isQueued:Whether to specify this instruction as a queue command. queuedCmdIndex:Quene command index(The Controller Return)
Return	DobotCommunicate_NoError: The instruction returns normally. DobotCommunicate_BufferFull: The instruction quene is full. DobotCommunicate_Timeout: The instruction did not return, resulting in a timeout.

2.19.4 Read I / O output level

Figure 2.68 Read I / O output level

Prototype	int GetIODO(IODO *ioDO)
Description	Reads I / O output level
Parameter	IODO typedef struct tagIODO { uint8_t address; uint8_t level; }IODO; ioDO:I/O ouput level structure variable pointers
Return	DobotCommunicate_NoError: The instruction returns normally. DobotCommunicate_BufferFull: The instruction queue is full. (The interface does not return this value) DobotCommunicate_Timeout: The instruction did not return, resulting in a timeout.

2.19.5 Set PWM Output

Figure 2.69 The interface description of setting PWM output

Prototype	int SetIOPWM(IOPWM *ioPWM, bool isQueued, uint64_t *queuedCmdIndex)
Description	Set I/O PWM output
Parameter	IOPWM typedef struct tagIOPWM { uint8_t address; float frequency; float dutyCycle; }IOPWM; ioPWM:I/O PWM output structure variable pointers isQueued:Whether to specify this instruction as a queue command. queuedCmdIndex:Quene command index(The Controller Return)
Return	DobotCommunicate_NoError: The instruction returns normally. DobotCommunicate_BufferFull: The instruction quene is full. DobotCommunicate_Timeout: The instruction did not return, resulting in a timeout.

2.19.6 Read PWM output

Figure 2.70 The interface description of reading PWM output

Prototype	int GetIOPWM(IOPWM *ioPWM)
Description	Read PWM output
Parameter	IOPWM typedef struct tagIOPWM { uint8_t address; float frequency; float dutyCycle; }IOPWM; ioPWM:I/O PWM output structure variable pointers
Return	DobotCommunicate_NoError: The instruction returns normally. DobotCommunicate_BufferFull: The instruction queue is full. (The interface does not return this value) DobotCommunicate_Timeout: The instruction did not return, resulting in a timeout.

2.19.7 Read I/O output level

Figure 2.71 The interface description of reading I/O output level

Prototype	int GetIODI(IODI *ioDI)
Description	Read I/O output level
Parameter	IODI typedef struct tagIODI { uint8_t address; uint8_t level; }IODI; ioDI:I/O output level structure variable pointers
Return	DobotCommunicate_NoError: The instruction returns normally. DobotCommunicate_BufferFull: The instruction queue is full. (The interface does not return this value) DobotCommunicate_Timeout: The instruction did not return, resulting in a timeout.

2.19.8 Read I/O analog-digital conversion values

Figure 2.72 Read I/O analog-digital conversion values

Prototype	int GetIOADC(IOADC *ioADC)
Description	Read I/O analog-digital conversion values
Parameter	IOADC： typedef struct tagIOADC { uint8_t address; uint16_t value; }IOADC; ioADC:I/O ADC variable point
Return	DobotCommunicate_NoError: The instruction returns normally. DobotCommunicate_BufferFull: The instruction queue is full. (The interface does not return this value) DobotCommunicate_Timeout: The instruction did not return, resulting in a timeout.

2.20 CAL

2.20.1 Set angle sensor static error

Angle sensors on forearm and rear arm may have a static error due to angle sensor welding, machine status, and so on. We can get this static error through various means (such as leveling, compared with the standard source), and write into the device through this API.

Figure 2.73 Set angle sensor static error

Prototype	int SetAngleSensorStaticError(float rearArmAngleError, float frontArmAngleError)
Description	Set angle sensor static error of reararm and forearm
Parameter	rearArmAngleError frontArmAngleError
Return	DobotCommunicate_NoError: The instruction returns normally. DobotCommunicate_BufferFull: The instruction queue is full. (The interface does not return this value) DobotCommunicate_Timeout: The instruction did not return, resulting in a timeout.

2.20.2 Get angle sensor static error

Figure 2.74 Read angle sensor static error

Prototype	int SetAngleSensorStaticError(float rearArmAngleError, float frontArmAngleError)
Description	Set angle sensor static error of reararm and forearm
Parameter	rearArmAngleError frontArmAngleError
Return	DobotCommunicate_NoError: The instruction returns normally. DobotCommunicate_BufferFull: The instruction queue is full. (The interface does not return this value) DobotCommunicate_Timeout: The instruction did not return, resulting in a timeout.

2.21 WIFI

2.21.1 Set WIFI configuration mode

Figure 2.75 Set WIFI configuration mode

Prototype	int SetWIFIConfigMode(bool enable)
Description	Set WIFI configuration mode
Parameter	enable: Indicates whether the configuration mode is enabled
Return	DobotCommunicate_NoError: The instruction returns normally. DobotCommunicate_BufferFull: The instruction queue is full. (The interface does not return this value) DobotCommunicate_Timeout: The instruction did not return, resulting in a timeout.

2.21.2 Get the current WIFI configuration mode if it is enabled

Figure 2.76 2.21.2 Get the current WIFI configuration mode if it is enabled

Prototype	int GetWIFIConfigMode(bool *isEnabled)
Description	Get the current WIFI configuration mode if it is enabled
Parameter	isEnabled:Incoming variable pointer
Return	DobotCommunicate_NoError: The instruction returns normally. DobotCommunicate_BufferFull: The instruction queue is full. (The interface does not return this value) DobotCommunicate_Timeout: The instruction did not return, resulting in a timeout.

2.21.3 Set SSID

Figure 2.77 Set SSID

Prototype	int SetWIFISSID(const char *ssid)
Description	Set network SSID
Parameter	ssid：Network SSID string pointer
Return	DobotCommunicate_NoError: The instruction returns normally. DobotCommunicate_BufferFull: The instruction queue is full. (The interface does not return this value) DobotCommunicate_Timeout: The instruction did not return, resulting in a timeout.

2.21.4 Get the current SSID settings

Figure 2.78 2.21.4 Get the current SSID settings

Prototype	int GetWIFISSID(char *ssid, uint32_t maxLen)
Description	Get the current SSID settings
Parameter	ssid： Network SSID string pointer maxLen：Incoming external buffer length to avoid overflow.
Return	DobotCommunicate_NoError: The instruction returns normally. DobotCommunicate_BufferFull: The instruction queue is full. (The interface does not return this value) DobotCommunicate_Timeout: The instruction did not return, resulting in a timeout.

2.21.5 Set the network password

Figure 2.79 Network password string pointer

Prototype	int SetWIFIPassword(const char *password)
Description	Set the network password
Parameter	password：Network password string pointer
Return	DobotCommunicate_NoError: The instruction returns normally. DobotCommunicate_BufferFull: The instruction queue is full. (The interface does not return this value) DobotCommunicate_Timeout: The instruction did not return, resulting in a timeout.

2.21.6 Get the current network password

Figure 2.80 Get the current network password

Prototype	int GetWIFIPassword(char *password, uint32_t maxLen)
Description	Get the current network password
Parameter	password： Network password string pointer maxLen：Incoming external buffer length to avoid overflow.
Return	DobotCommunicate_NoError: The instruction returns normally. DobotCommunicate_BufferFull: The instruction queue is full. (The interface does not return this value) DobotCommunicate_Timeout: The instruction did not return, resulting in a timeout.

2.21.7 Set IP address

Figure 2.81 The interface description of setting IP address

Prototype	int SetWIFIIPAddress(WIFIIPAddress *wifiIPAddress)
Description	Set IP address
Parameter	typedef struct tagWIFIIPAddress { uint8_t dhcp; uint8_t addr[4]; }WIFIIPAddress; wifiIPAddr：IP address structure pointer
Return	DobotCommunicate_NoError: The instruction returns normally. DobotCommunicate_BufferFull: The instruction queue is full. (The interface does not return this value) DobotCommunicate_Timeout: The instruction did not return, resulting in a timeout.

2.21.8 GetWIFIIPAddress

Figure 2.82 The interface description of GetWIFIIPAddress

Prototype	int GetWIFIIPAddress(WIFIIPAddress *wifiIPAddress)
Description	GetWIFIIPAddress
Parameter	typedef struct tagWIFIIPAddress { uint8_t dhcp; uint8_t addr[4]; }WIFIIPAddress; wifiIPAddr：IP address structure pointer
Return	DobotCommunicate_NoError: The instruction returns normally. DobotCommunicate_BufferFull: The instruction queue is full. (The interface does not return this value) DobotCommunicate_Timeout: The instruction did not return, resulting in a timeout.

2.21.9 Set WIFI Sub Netmask

Figure 2.83 The interface description of setting WIFI Sub Netmask

Prototype	int SetWIFINetmask(WIFINetmask *wifiNetmask)
Description	Set WIFI Sub Netmask
Parameter	typedef struct tagWIFINetmask { uint8_t addr[4]; }WIFINetmask; wifiNetmask：Sub Netmask structure pointer
Return	DobotCommunicate_NoError: The instruction returns normally. DobotCommunicate_BufferFull: The instruction queue is full. (The interface does not return this value) DobotCommunicate_Timeout: The instruction did not return, resulting in a timeout.

2.21.10 GetWIFINetmask

Figure 2.84 The interface description of GetWIFINetmask

Prototype	int GetWIFINetmask(WIFINetmask *wifiNetmask)
Description	GetWIFINetmask
Parameter	typedef struct tagWIFINetmask { uint8_t addr[4]; }WIFINetmask; wifiNetmask：Netmask structure pointer
Return	DobotCommunicate_NoError: The instruction returns normally. DobotCommunicate_BufferFull: The instruction queue is full. (The interface does not return this value) DobotCommunicate_Timeout: The instruction did not return, resulting in a timeout.

2.21.11 SetWIFIGateway

Figure 2.85 The interface description of setting WIFI Sub Netmask

Prototype	int SetWIFIGateway(WIFIGateway *wifiGateway)
Description	SetWIFIGateway
Parameter	typedef struct tagWIFIGateway { uint8_t addr[4]; }WIFIGateway; wifiGateway： WiFi gateway pointer
Return	DobotCommunicate_NoError: The instruction returns normally. DobotCommunicate_BufferFull: The instruction queue is full. (The interface does not return this value) DobotCommunicate_Timeout: The instruction did not return, resulting in a timeout.

2.21.12 Gets the current settings of gateway

Figure 2.86 Gets the current settings of subnet mask

Prototype	int GetWIFIGateway(WIFIGateway *wifiGateway)
Description	Gets the current settings for the gateway
Parameter	typedef struct tagWIFIGateway { uint8_t addr[4]; }WIFIGateway; wifiGateway：WiFi gateway pointer
Return	DobotCommunicate_NoError: The instruction returns normally. DobotCommunicate_BufferFull: The instruction queue is full. (The interface does not return this value) DobotCommunicate_Timeout: The instruction did not return, resulting in a timeout.

2.21.13 Set DNS

Figure 2.87 The interface deacription of setting DNS

Prototype	int SetWIFIDNS(WIFIDNS *wifiDNS)
Description	Set DNS
Parameter	typedef struct tagWIFIDNS { uint8_t addr[4]; }WIFIDNS; wifiDNS：DNS structure pointer
Return	DobotCommunicate_NoError: The instruction returns normally. DobotCommunicate_BufferFull: The instruction queue is full. (The interface does not return this value) DobotCommunicate_Timeout: The instruction did not return, resulting in a timeout.

2.21.14 Get the current settings of DNS

Figure 2.88 The interface description of getting the current DNS settings

Prototype	int GetWIFIDNS(WIFIDNS *wifiDNS)
Description	Get the current DNS settings
Parameter	typedef struct tagWIFIDNS { uint8_t addr[4]; }WIFIDNS; wifiDNS：DNS structure pointer
Return	DobotCommunicate_NoError: The instruction returns normally. DobotCommunicate_BufferFull: The instruction queue is full. (The interface does not return this value) DobotCommunicate_Timeout: The instruction did not return, resulting in a timeout.

2.21.15 Get Wi-Fi connect status

Figure 2.89 Get Wi-Fi connect status

Prototype	int GetWIFIConnectStatus(bool *isConnected)
Description	Get Wi-Fi connect status
Parameter	isConnected：Wi-Fi connect status variable pointer
Return	DobotCommunicate_NoError: The instruction returns normally. DobotCommunicate_BufferFull: The instruction queue is full. (The interface does not return this value) DobotCommunicate_Timeout: The instruction did not return, resulting in a timeout.

2.22 Other functions

2.22.1 Event loop

In some languages, the application exits directly after calling API interface because there is no event loop, thus causing the instruction not to be issued to the Dobot controller. To avoid this, we provide an event loop interface, which is called before the application exits (currently known, Python need to follow this).

Figure 2.90 The interface description of an event loop

Prototype	void DobotExec(void)
Description	Event loop
Parameter	Void
Return	Void

深圳市越疆科技有限公司

邮编：510630

网址：www.dobot.cc

电话：(0755)38730916

地址：深圳市南山区学苑大道 1001 号南山智园 c2 栋 18 楼