Dobot M1 Script Description

Issue: V1.4.3

Date: 2019-10-30

Shenzhen Yuejiang Technology Co., Ltd

Copyright © ShenZhen Yuejiang Technology Co., Ltd 2019. All rights reserved.

No part of this document may be reproduced or transmitted in any form or by any means without prior written consent of Yuejiang Technology Co., Ltd

Disclaimer

To the maximum extent permitted by applicable law, the products described (including its hardware, software and firmware, etc.) in this document are provided **AS IS**, which may have flaws, errors or faults. Yuejiang makes no warranties of any kind, express or implied, including but not limited to, merchantability, satisfaction of quality, fitness for a particular purpose and non-infringement of third party rights. In no event will Yuejiang be liable for any special, incidental, consequential or indirect damages resulting from the use of our products and documents.

Before using our product, please thoroughly read and understand the contents of this document and related technical documents that are published online, to ensure that the robotic arm is used on the premise of fully understanding the robotic arm and related knowledge. Please use this document with technical guidance from professionals. Even if follow this document or any other related instructions, Damages or losses will be happen in the using process, Dobot shall not be considered as a guarantee regarding to all security information contained in this document.

The user has the responsibility to make sure following the relevant practical laws and regulations of the country, in order that there is no significant danger in the use of the robotic arm.

Shenzhen Yuejiang Technology Co., Ltd

Address: Floor 9-10, Building 2, Chongwen Garden, Nanshan iPark, Liuxian Blvd, Nanshan District, Shenzhen, Guangdong Province, China

Website: www.dobot.cc

Preface

Purpose

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

Intended Audience

This document is intended for:

- Customer Engineer
- Installation and Commissioning Engineer
- Technical Support Engineer

Change History

Date	Change Description
2019/10/30	The first release

Symbol Conventions

The symbols that may be founded in this document are defined as follows.

Symbol	Description
⚠DANGER	Indicates a hazard with a high level of risk which, if not avoided, could result in death or serious injury
⚠WARNING	Indicates a hazard with a medium level or low level of risk which, if not avoided, could result in minor or moderate injury, robotic arm damage
⚠NOTICE	Indicates a potentially hazardous situation which, if not avoided, can result in robotic arm damage, data loss, or unanticipated result
🕮NOTE	Provides additional information to emphasize or supplement important points in the main text

 Copyright © Yuejiang Technology Co., Ltd

Contents

 Copyright © Yuejiang Technology Co., Ltd

1. Overview

You can control a Dobot M1 over scripting. Dobot M1 supports various script API, such as velocity/acceleration setting, motion mode setting, and I/O configuration, which uses Python language for secondary development.

 Copyright © Yuejiang Technology Co., Ltd

2. Script API Interface Description

2.1 Dobot Commands

Dobot controller supports two kind of commands: Immediate command and queue command:

- Immediate command: 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: When Dobot controller receives a command, this command will be pressed into the controller internal command queue. Dobot controller will execute commands in the order in which the commands were pressed into the queue.

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

📖NOTE

The script API supports synchronous and asynchronous modes. For details, please see as follows.

- Synchronous mode: After upper-computer software sends a synchronous command, the upper-computer software will not send the next command until the Dobot M1 system finishes executing the current synchronous command.
- Asynchronous mode: Namely, upper-computer software sends queue command or immediate command. After the upper-computer software sends a command, it will send the next command immediately regardless of whether the Dobot M1 system executes the current command.

2.2 PTP

PTP mode supports MOVJ, MOVL, and JUMP, which is point-to-point movement. The trajectory of playback depends on the motion mode.

- MOVJ: Joint movement. From point A to point B, each joint will run from initial angle to its target angle, regardless of the trajectory, as shown in Figure 2.1.

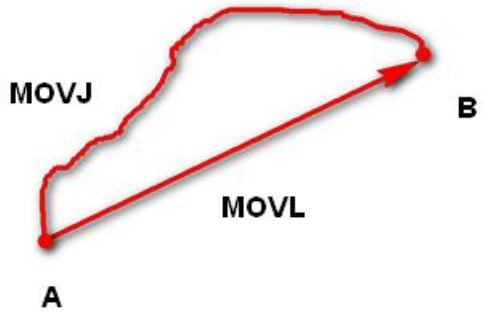


Figure 2.1 MOVL/MOVJ mode

- MOVL: Rectilinear movement. The joints will perform a straight line trajectory from point A to point B, as shown in Figure 2.1.

- JUMP: From point A to point B, The joints will move in MOVJ mode, of which the trajectory looks like a door, as shown in Figure 2.2.
    1. Move up to the lifting Height (**Height**) in MOVJ mode.
    2. Move up to the maximum lifting height (**Limit**).
    3. Move horizontally to a point that is above B by height.
    4. Move down to a point that is above B by height, which the height of the point is that of point B plus **Height**.
    5. Move down to Point B.

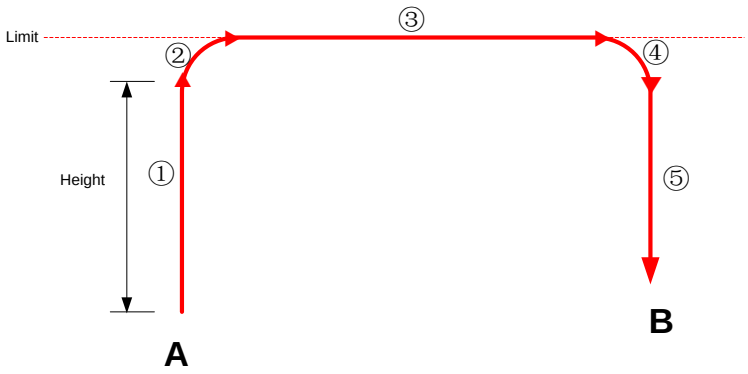
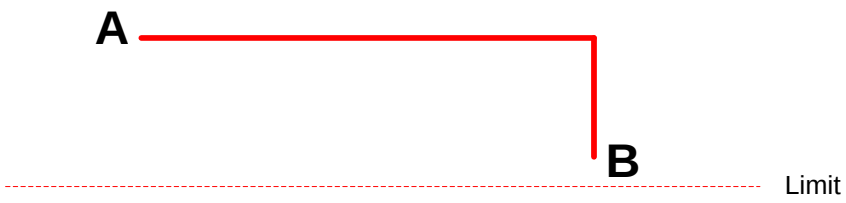


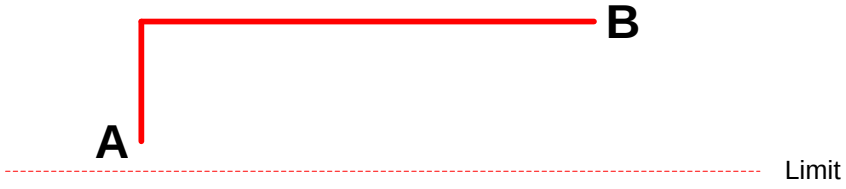
Figure 2.2 JUMP mode

In JUMP mode, if the starting point or the end point is higher than or equal to **Limit**, or the height that the end effector lifts upwards is higher than or equal to **Limit**, the trajectory is different to that of Figure 2.2. Assuming that point A is the starting point, point B is the end point, **Limit** is the maximum lifting height, and **Height** is the lifting height.

- Point A and point B are both higher than **Limit**, but point A is higher than point B.

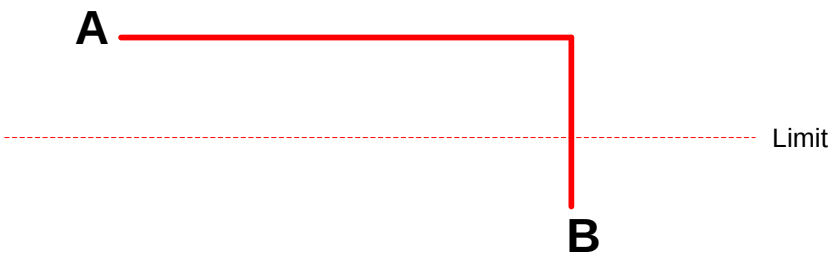


- Point A and point B are both higher than **Limit**, but point B is higher than point A.

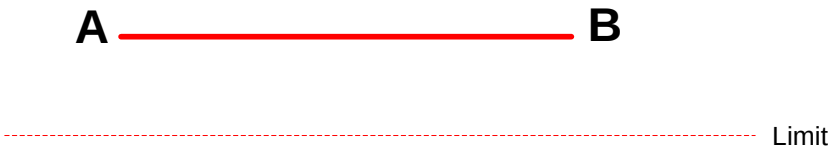


 Copyright © Yuejiang Technology Co., Ltd

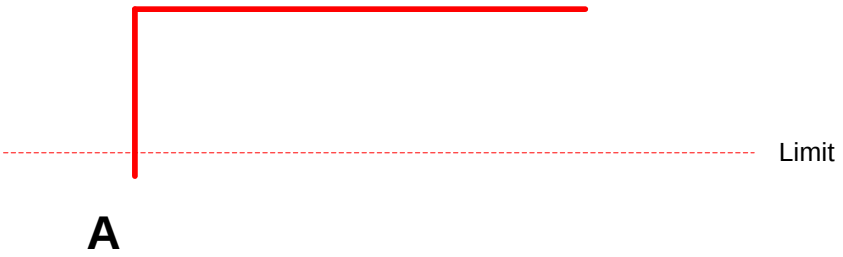
- Point A is higher than **Limit**, but point B is lower than **Limit**.



- The height of point A is the same as that of point B, but both are higher than **Limit**.



- Point A is lower than **Limit**, but point B is higher than **Limit**.



- The height of point A and point B are both the same as **Limit**.

- Point A and point B are both lower than **Limit**, but the height that the height of point A plus **Height** and that of point B plus **Height** is higher than **Limit**.

 Copyright © Yuejiang Technology Co., Ltd

NOTICE

If you use the motion command when writing a program in Cartesian coordinate system, please add the orientation command before this motion command, which indicates the arm orientation of Dobot M1.

2.2.1 Setting the Lifting Height and the Maximum Lifting Height in JUMP mode

Table 2.1 Set the lifting height and the maximum lifting height in JUMP mode

Prototype	dType.SetPTPJumpParams(api, jumpHeight, zLimit, isQueued)
Description	Set the lifting height and the maximum height in JUMP mode
Parameter	api: Dobot DLL object, cannot be modified jumpHeight: Lifting height zLimit: Maximum lifting height isQueued: Whether to add this command to the queue. Value range: 0, immediate command; 1: queue command
Return	If the command is immediate command , the return is 0 If the command is queue command, the return is the index of this command in the queue
Example	dType.SetPTPJumpParams(api, 50, 100, 0)

2.2.2 Setting the Lifting Height and the Maximum Lifting Height Synchronously in JUMP Mode

Table 2.2 Set the lifting height and the maximum lifting height synchronously in JUMP mode

Prototype	dType.SetPTPJumpParamsSync(api, jumpHeight, zLimit)
Description	In synchronous mode, set the lifting height and the maximum height in JUMP mode
Parameter	api: Dobot DLL object, cannot be modified jumpHeight: Lifting height zLimit: Maximum lifting height

Return	None
Example	dType.SetPTPJumpParamsSync (api, 50, 100)

2.2.3 Getting the Lifting Height and the Maximum Lifting Height in JUMP mode

Table 2.3 Get the lifting height and the maximum lifting height in JUMP mode

Prototype	dType.GetPTPJumpParams(api)
Description	Get the lifting height and the maximum height in JUMP mode
Parameter	api: Dobot DLL object, cannot be modified
Return	jumpHeight (List[0]): Lifting height zLimit (List[1]): Maximum lifting height
Example	None

2.2.4 Setting the Velocity Ratio and Acceleration Ratio in PTP Mode

Table 2.4 Set the velocity ratio and the acceleration ratio in PTP mode

Prototype	int SetPTPCommonParams(PTPCommonParams *ptpCommonParams, bool isQueued, uint64_t *queuedCmdIndex)
Description	Set the velocity ratio and acceleration ratio in PTP mode
Parameter	api: Dobot DLL object, cannot be modified velocityRatio: Velocity ratio in Cartesian coordinate system and Joint coordinate system accelerationRatio: Acceleration ratio in Cartesian coordinate system and Joint coordinate system isQueued: Whether to add this command to the queue. Value range: 0, immediate command; 1: queue command
Return	If the command is immediate command , the return is 0 If the command is queue command, the return is the index of this command in the queue
Example	dType.SetPTPCommonParams(api, 50, 50, 1)

2.2.5 Setting the Velocity Ratio and Acceleration Ratio Synchronously in PTP Mode

Table 2.5 Set the velocity ratio and the acceleration ratio synchronously in PTP mode

Prototype	dType.SetPTPCommonParamsSync(api, velocityRatio, accelerationRatio)
Description	In synchronous mode, set the velocity ratio and acceleration ratio in PTP mode

Parameter	api: Dobot DLL object, cannot be modified velocityRatio: Velocity ratio in Cartesian coordinate system and Joint coordinate system accelerationRatio: Acceleration ratio in Cartesian coordinate system and Joint coordinate system
Return	None
Example	dType.SetPTPCommonParamsSync(api, 30, 30)

2.2.6 Getting the Velocity Ratio and Acceleration Ratio in PTP Mode

Table 2.6 Get the velocity ratio and acceleration ratio in PTP mode

Prototype	dType.GetPTPCommonParams(api)
Description	Get the velocity ratio and acceleration ratio in PTP mode
Parameter	api: Dobot DLL object, cannot be modified
Return	velocityRatio (List[0]): Velocity ratio in Cartesian coordinate system and Joint coordinate system accelerationRatio (List[1]): Acceleration ratio in Cartesian coordinate system and Joint coordinate system
Example	None

2.2.7 Executing a PTP Command

Table 2.7 Execute a PTP command

Prototype	dType.SetPTPCmd(api, ptpMode, x, y, z, rHead, isqueued)
Description	Execute a PTP command. Please call this API after setting the related parameters in PTP mode to make the Dobot move to the target point
Parameter	api: Dobot DLL object, cannot be modified ptpMode: PTP mode. Value range: 0-9 0: JUMP mode. (x,y,z, rHead) is the target point in Cartesian coordinate system and the robot moves horizontally in MOVJ mode 1: MOVJmode. (x,y,z, rHead) is the target point in Cartesian coordinate system 2: MOVLmode. (x,y,z, rHead) is the target point in Cartesian coordinate system 3: JUMPmode. (x,y,z, rHead) is the target point in Joint coordinate system 4: MOVJmode. (x,y,z, rHead) is the target point in Joint coordinate system 5: MOVLmode. (x,y,z, rHead) is the target point in Joint coordinate system 6: MOVJmode. (x,y,z, rHead) is the angle increment in Joint coordinate system 7: MOVLmode. (x,y,z, rHead) is the Cartesian coordinate increment in Joint

	coordinate system 8: MOVJmode. (x,y,z, rHead) is the Cartesian coordinate increment in Cartesian coordinate system 9: JUMPmode. (x,y,z, rHead) is the target point in Cartesian coordinate system and the robot moves horizontally in MOVL mode x.y.z.rHead: Coordinate parameters in PTP mode. (x,y,z,rHead) can be set to Cartesian coordinate, joints angle, or increment of them isQueued: Whether to add this command to the queue. Value range: 0, immediate command; 1: queue command
Return	If the command is immediate command , the return is 0 If the command is queue command, the return is the index of this command in the queue
Example	Robot moves to (244, -136, 80,0) in MOVJ mode dType.SetPTPCommonParams(api, 50, 50, 1) dType.SetPTPCmd(api, 1, 244, -136, 80,0, 1)

2.2.8 Executing a PTP Command Synchronously

Table 2.8 Execute a PTP command synchronously

Prototype	dType.SetPTPCmdSync(api, ptpMode, x, y, z, rHead)
Description	In synchronously mode, Execute a PTP command. Please call this API after setting the related parameters in PTP mode to make the Dobot move to the target point
Parameter	api: Dobot DLL object, cannot be modified ptpMode: PTP mode. Value range: 0-9 0: JUMP mode. (x,y,z, rHead) is the target point in Cartesian coordinate system and the robot moves horizontally in MOVJ mode 1: MOVJmode. (x,y,z, rHead) is the target point in Cartesian coordinate system 2: MOVLmode. (x,y,z, rHead) is the target point in Cartesian coordinate system 3: JUMPmode. (x,y,z, rHead) is the target point in Joint coordinate system 4: MOVJmode. (x,y,z, rHead) is the target point in Joint coordinate system 5: MOVLmode. (x,y,z, rHead) is the target point in Joint coordinate system 6: MOVJmode. (x,y,z, rHead) is the angle increment in Joint coordinate system 7: MOVLmode. (x,y,z, rHead) is the Cartesian coordinate increment in Joint coordinate system 8: MOVJmode. (x,y,z, rHead) is the Cartesian coordinate increment in Cartesian coordinate system 9: JUMPmode. (x,y,z, rHead) is the target point in Cartesian coordinate system and the robot moves horizontally in MOVL mode

	x.y.z.rHead: Coordinate parameters in PTP mode. (x,y,z,rHead) can be set to Cartesian coordinate, joints angle, or increment of them
Return	None
Example	Robot moves to (244, -136, 80,0) in MOVJ mode dType.SetPTPCommonParams(api, 50, 50, 1) dType.SetPTPCmd(api, 1, 244, -136, 80,0)

2.2.9 Executing the ARC Command

Table 2.9 Execute the ARC command

Prototype	dType.SetARCCmd(api, cirPoint, toPoint, isQueued)
Description	Execute the ARC command. Please call this API after setting the related parameters in ARC mode to make Dobot move to the target point. In ARC mode, it is necessary to confirm the three points with other motion modes.
Parameter	api: Dobot DLL object, cannot be modified cirPoint: Middle point. (x,y,z,r) can be set to Cartesian coordinate toPoint: End point. (x,y,z,r) can be set to Cartesian coordinate isQueued: Whether to add this command to the queue. Value range: 0, immediate command; 1: queue command
Return	If the command is immediate command , the return is 0 If the command is queue command, the return is the index of this command in the queue
Example	dType.SetARCCmd(api, [62,265,120,50], [-58,266,120,76], 1)

2.2.10 Executing the ARC Command Synchronously

Table 2.10 Execute the ARC command

Prototype	dType.SetARCCmdSync(api, cirPoint, toPoint)
Description	In synchronous mode, execute the ARC command. Please call this API after setting the related parameters in ARC mode to make Dobot move to the target point. In ARC mode, it is necessary to confirm the three points with other motion modes.
Parameter	api: Dobot DLL object, cannot be modified cirPoint: Middle point. (x,y,z,r) can be set to Cartesian coordinate toPoint: End point. (x,y,z,r) can be set to Cartesian coordinate
Return	None
Example	None

 Copyright © Yuejiang Technology Co., Ltd

2.2.11 Executing the CIRCLE Command

The CIRCLE mode is similar to the ARC mode, where the trajectory is a circle.

Table 2.11 Execute the CIRCLE command

Prototype	dType.SetCircleCmd(api, cirPoint, toPoint, count, isQueued)
Description	Execute the CIRCLE command. Please call this API after setting the related parameters of playback in CIRCLE mode to make Dobot move to the target point. In CIRCLE mode, it is necessary to confirm the three points with other motion modes.
Parameter	api: Dobot DLL object, cannot be modified cirPoint: Middle point. (x,y,z,r) can be set to Cartesian coordinate toPoint: End point. (x,y,z,r) can be set to Cartesian coordinate count: Circle number isQueued: Whether to add this command to the queue. Value range: 0, immediate command; 1: queue command
Return	If the command is immediate command , the return is 0 If the command is queue command, the return is the index of this command in the queue
Example	dType.SetCircleCmd (api, [62,265,120,50], [-58,266,120,76], 1, 1)

2.2.12 Executing the CIRCLE Command Synchronously

Table 2.12 Execute the CIRCLE command synchronously

Prototype	dType.SetCircleCmdSync(api, cirPoint, toPoint, count)
Description	In synchronous mode, execute the CIRCLE command. Please call this API after setting the related parameters of playback in CIRCLE mode to make Dobot move to the target point. In CIRCLE mode, it is necessary to confirm the three points with other motion modes.
Parameter	api: Dobot DLL object, cannot be modified cirPoint: Middle point. (x,y,z,r) can be set to Cartesian coordinate toPoint: End point. (x,y,z,r) can be set to Cartesian coordinate count: Circle number
Return	None
Example	None

2.2.13 Setting Arm Orientation

 Copyright © Yuejiang Technology Co., Ltd

Table 2.13 Set arm orientation

Prototype	dType.SetArmOrientation(api, armOrientation, isQueued)
Description	Set arm orientation
Parameter	api: Dobot DLL object, cannot be modified armOrientation: Arm orientation. 0: Lefty hand; 1: Righty hand isQueued: Whether to add this command to the queue. Value range: 0, immediate command; 1: queue command
Return	If the command is immediate command , the return is 0 If the command is queue command, the return is the index of this command in the queue
Example	dType.SetArmOrientation(api, 0, 1)

2.2.14 Setting Arm Orientation Synchronously

Table 2.14 Set arm orientation synchronously

Prototype	dType.SetArmOrientationSync(api, armOrientation)
Description	In synchronous mode, set arm orientation
Parameter	api: Dobot DLL object, cannot be modified armOrientation: Arm orientation. 0: Lefty hand; 1: Righty hand
Return	None
Example	None

2.2.15 Getting the Real-time Pose of the Dobot M1

Table 2.15 Get the real-time pose of Dobot

Prototype	int GetPose(Pose *pose)
Description	Get the real-time pose of the Dobot
Parameter	api: Dobot DLL object, cannot be modified
Return	X (List[0]): X-axis coordinate in Cartesian coordinate system Y (List[1]): Y-axis coordinate in Cartesian coordinate system Z (List[2]): Z-axis coordinate in Cartesian coordinate system R (List[3]): R-axis coordinate in Cartesian coordinate system J1 (List[4]): J1 coordinate in Joint coordinate system J2 (List[5]): J2 coordinate in Joint coordinate system J3 (List[6]): J3 coordinate in Joint coordinate system

 Copyright © Yuejiang Technology Co., Ltd

	J4 (List[7])）: J4coordinate in Joint coordinate system
Example	None

2.3 WAITING

2.3.1 Executing the Waiting Command

Table 2.16 Execute the Waiting command

Prototype	dType.SetWAITCmd(api, waitTime, isQueued)
Description	Execute the Waiting command. If you need to set the pause time between the two commands, please call this API This command must be added to the command queue, namely, **isQueued** must be set to **1**. If not, the parameter **waitTime** of **Waiting** command in the command queue being executed may be changed because the **WAITCmd** memory is shared
Parameter	api: Dobot DLL object, cannot be modified waitTime: Wait time. Unit: ms isQueued: Whether to add this command to the queue. Value range: 0, immediate command; 1: queue command
Return	If the command is immediate command , the return is 0 If the command is queue command, the return is the index of this command in the queue
Example	dType.SetWAITCmd(api, 1000, 1)

2.3.2 Executing the Waiting Command Synchronously

Table 2.17 Execute the Waiting command synchronously

Prototype	dType.SetWAITCmd Sync(api, waitTime)
Description	In synchronous mode, execute the Waiting command. If you need to set the pause time between the two commands, please call this API
Parameter	api: Dobot DLL object, cannot be modified waitTime: Wait time. Unit: ms
Return	None
Example	None

2.3.3 Delay Command

 Copyright © Yuejiang Technology Co., Ltd

Table 2.18 Delay command

Prototype	dType.dSleep(ms)
Description	Delay command, sent by upper-computer software
Parameter	ms: Delay time. Unit: ms
Return	None
Example	None

2.4 TRIGGERING

2.4.1 Executing the Triggering Command

Table 2.19 Execute the Triggering command

Prototype	dType.SetTRIGCmd(api, address, mode, condition, threshold, isQueued)
Description	Execute the triggering command. This command must be added to the command queue, namely, **isQueued** must be set to **1**. If not, the parameter **condition** of the **Triggering** command in the queue command being executed may be changed because the **TRIGCmd** memory is shared
Parameter	api: Dobot DLL object, cannot be modified address：I/O address: If **mode** is set to 0, the value range is 1 to 24. If **mode** is set to 1, the value range is 1 to 6 mode: Triggering mode. 0: Level trigger.1:A/D trigger condition: Triggering condition **Level**: 0, equal. 1, unequal **A/D**: 0, less than. 1,less than or equal. 2, greater than or equal. 3, greater than threshold: Triggering threshold. **Level** : 0,1 .**A/D** : 0-4095 isQueued: Whether to add this command to the queue. Value range: 0, immediate command; 1: queue command
Return	If the command is immediate command , the return is 0 If the command is queue command, the return is the index of this command in the queue
Example	None

2.4.2 Executing the Triggering Command Synchronously

Table 2.20 Execute the Triggering command synchronously

Prototype	dType.SetTRIGCmdSync(api, address, mode, condition, threshold)

Description	In synchronous mode, execute the triggering command.
Parameter	api: Dobot DLL object, cannot be modified address：I/O address: If **mode** is set to 0, the value range is 1 to 24. If **mode** is set to 1, the value range is 1 to 6 mode: Triggering mode. 0: Level trigger.1:A/D trigger condition: Triggering condition **Level**: 0, equal. 1, unequal **A/D**: 0, less than. 1,less than or equal. 2, greater than or equal. 3, greater than threshold: Triggering threshold. **Level** : 0,1 .**A/D** : 0-4095
Return	None
Example	None

2.5 I/O Command

In the Dobot controller, the addresses of the I/O interfaces are unified. Here, you can see as follows:

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

For more details, please see *Dobot M1 User Guide.*

2.5.1 Setting Digital Output

Table 2.21 Set digital output

Prototype	dType.SetIODO(api, address, level, isQueued)
Description	Set digital output
Parameter	api: Dobot DLL object, cannot be modified address: Digital output address. Value range: 1-22 level: Output level. 0: Low level; 1: High level isQueued: Whether to add this command to the queue. Value range: 0, immediate command; 1: queue command
Return	If the command is immediate command , the return is 0 If the command is queue command, the return is the index of this command in the queue
Example	dType.SetIODO(api, 1,0, 0)

2.5.2 Setting Digital Output Synchronously

Table 2.22 Set ditial output synchronously

Prototype	dType.SetIODOSync(api, address, level)
Description	In synchronous mode, set the digital output
Parameter	api: Dobot DLL object, cannot be modified address: Digital output address. Value range: 1-22 level: Output level. 0: Low level; 1: High level
Return	None
Example	dType.SetIODO(api, 1, 1)

2.5.3 Getting Digital Output

Table 2.23 Get digital output

Prototype	dType.GetIODO(api, addr)
Description	Get digital output
Parameter	api: Dobot DLL object, cannot be modified address: Digital output address. Value range: 1-22
Return	Level value of the right digital output address. 0: Low level; 1: High level
Example	dType.SetIODOSync(api, 1, 1) level = dType.GetIODO(api, 1)

2.5.4 Getting Digital Input

Table 2.24 Get Digital input

Prototype	dType.GetIODI(api, addr)
Description	Get digital input
Parameter	api: Dobot DLL object, cannot be modified address: Digital input address. Value range: 1-24
Return	Level value of the right digital input address. 0: Low level; 1: High level
Example	None

2.5.5 Getting Digital Inputs in Succession

Table 2.25 Get Digital inputs in succession

Prototype	dType.GetIODIs(api, startAddr, endAddr)

 Copyright © Yuejiang Technology Co., Ltd

Description	Get digital inputs in succession
Parameter	api: Dobot DLL object, cannot be modified startAddr : Start address of digital inputs. Value range: 1-24 endAddr : End address of digital inputs. Value range: 1-24
Return	Level values of the right digital input addresses. 0: Low level; 1: High level
Example	None

2.5.6 Getting Analog Input

Table 2.26 Get analog Input

Prototype	dType.GetIOADC(api, addr)
Description	Get analog input
Parameter	api: Dobot DLL object, cannot be modified address: Analog input address. Value range: 1-6
Return	Value of the right analog input address. Value range: 0-4095

2.6 Other functions

2.6.1 Loading Playback Data

Table 2.27 Load playback data

Prototype	playbackData.load(fileName)
Description	Load data from playback file The playback file must be saved in **Installation directory\M1Studio\config\pbstore** directory. Otherwise, loading data fails.
Parameter	fileName: Playback file name. Format: string
Return	None
Example	Please see Program 2.1

2.6.2 Getting Playback Data

Table 2.28 Getting playback data

Prototype	playbackData.get(rowName)
Description	Get the right row data from the playback file This command must be used with playbackData.load(fileName) function

Parameter	rowName: Row name in playback file. Format: string
Return	Point data (x,y,z,r) in the right row
Example	Please see Program 2.1

Program 2.1 Load playback data demo

```
playbackData.load("playbackDemo")
for i in range(0,5):
pose = playbackData.get("first")
dType.SetPTPCmdSync(api, 1, pose[0], pose[1], pose[2], pose[3], 1)
pose = playbackData.get("second")
dType.SetPTPCmdSync(api, 1, pose[0], pose[1], pose[2], pose[3], 1)
```

2.6.3 Setting Pallet Name

Table 2.29 Set pallet name

Prototype	matrixPallet.setMatrixPallet(name)
Description	Set pallet name The pallet name in this command must be same as that when creating a pallet on the script page
Parameter	name: Pallet name. Format: string
Return	None
Example	Please see Program 2.2

2.6.4 Getting Preparation Point Data

Table 2.30 Getting preparation point data

Prototype	matrixPallet.getReadyPoint()
Description	Get preparation point data Before calling this command, please create a pallet
Parameter	None
Return	Preparation point data
Example	Please see Program 2.2

2.6.5 Getting Transition Point Data

Table 2.31 Getting transition point data

Prototype	matrixPallet.getTransPoint()
Description	Get transition point data Before calling this command, please create a pallet
Parameter	None
Return	Transition point data
Example	Please see Program 2.2

2.6.6 Getting Pallet Points Data

Table 2.32 Getting pallet points data

Prototype	matrixPallet.getPalletPoint()
Description	Get pallet points data Before calling this command, please create a pallet
Parameter	None
Return	Pallet points data
Example	Please see Program 2.2

NOTICE

If you need to operate a pallet program in the offline mode, please make sure that the pallet parameters are set on the network condition. Namely, when setting the pallet parameters, you must use the network cable to connect the Dobot M1 and the PC. Otherwise, the pallet information cannot be loaded into the Dobot M1 system.

Program 2.2 Pallet demo

```
#set matrix pallet config
matrixPallet.setMatrixPallet("matrixTest")
#get matrix pallet ready point
readyPoint = matrixPallet.getReadyPoint()
#get matrix transition ready point
transPoint = matrixPallet.getTransPoint()
#get matrix pallet point list
palletPoint = matrixPallet.getPalletPoint()
```

```
for point in palletPoint:
#go to ready point
print("ready Point:", readyPoint)
dType.SetPTPCmd(api, 1, readyPoint[0], readyPoint[1], readyPoint[2], readyPoint[3], isQueued=1)
#go to transition point
print("trans Point:", transPoint)
dType.SetPTPCmd(api, 1, transPoint[0], transPoint[1], transPoint[2], transPoint[3], isQueued=1)
#go to pallet point
print("pallet point:", point)
dType.SetPTPCmd(api, 1, point[0], point[1], point[2], point[3], isQueued=1)
```

 Copyright © Yuejiang Technology Co., Ltd