force dimension sdk

USER MANUAL

Robotic SDK

version 3.14.0

Force Dimension
Switzerland

www.forcedimension.com

1 DRD - Robotic SDK Documentation

1.1 Disclaimer

This release of the DRD is provided "as is", with no warranty of any kind. Force Dimension will accept no responsibility for any damage that result from using this software, including damage to any hardware involved.

Please note that the performance and safety of the robotic regulation provided by the DRD depends on the operating system, execution context and physical hardware used. A background in control theory and real-time programming is strongly recommended prior to writing applications using the DRD.

USE AT YOUR OWN RISK

1.2 Introduction

This document describes the DRD robotics software library. The DRD has been designed specifically to enable robotic control of Force Dimension haptic devices, as well as to make it possible to write applications that combine interactive (haptic) and automatic (robotic) capabilities. As a consequence, the DRD is built alongside the DHD haptic library and shares some fundamental resources with it. It is therefore possible to use DHD calls (`dhd*`) and DRD calls (`drd*`) in the same application. For the same reason, this documentation makes direct references to the DHD Haptic SDK documentation.

This section describes how the DRD is structured. Conceptually, the library manages axis regulation in its own high-priority thread, while all DRD function calls asynchronously control the regulation loop parameters. The DRD library is targeted at real-time platforms to guarantee the performance and safety of the regulation (as well as the users and hardware involved). However, the DRD has been implemented so that it can run with reasonable performance and acceptable safety on non-real time platforms, thanks in part to a control instability detection algorithm.

1.2.1 Regulation

At the heart of the library is the regulation thread, which constraints the position of each joint. The thread can be started with drdStart() (after the device has been initialized), and stopped with drdStop(). The refresh rate of the control thread can be retrieved by calling drdGetCtrlFreq(). Once the the thread is running (which can be assessed with drdIsRunning()), the device can only be moved by changing the regulation target of each joint. This can be achieved by calling one of the `drdMove*` or `drdTrack*` functions. The `drdMove*` functions are designed to send the device end-effector on a direct, smooth trajectory to any point in the workspace, while the `drdTrack*` calls should be used to smoothly constrain the device motion on a continuous trajectory along a set of points sent asynchronously to the control thread (see drdTrackPos() for more details). The key difference between the two sets of functions is that `drdMove*` calls do not guarantee continuity if a new call is made before an earlier call finishes. On the other hand, `drdTrack*` calls do guarantee continuity regardless of when they are invoked. However, `drdTrack*` trajectory generation is performed on each axis individually, while `drd↩ Move*` functions generate trajectories in 3D space. Outside of these different behaviors, both `drdMove*` and drdTrack* calls use a trajectory generation algorithm that guarantees continuous acceleration changes. For more details on the trajectory generation, see the section on trajectory generation parameters.

1.2.2 Trajectory Generation Parameters

The trajectory generation algorithm implemented in the DRD uses triangular acceleration constraints to guarantee smooth movements in both joint and cartesian spaces. The following parameters can be used to change the behavior of the generated trajectories:

- **Amax** - the maximal allowed acceleration [m/s^2]
- **Vmax** - the maximal allowed velocity [m/s]
- **Jerk** - the variation of acceleration over time [m/s^3]

1.2.3 Non Real-Time Considerations

On non real-time platforms, the periodicity of the regulation thread cannot be guaranteed. This has direct consequences on control stability and performance. In order to limit the performance degradation, the DRD implements a regulator that does not assume periodicity and can tolerate some jitter in the control loop. In order to optimize the performance of the control thread, drdSetPriorities() can be used to change the priority of both the calling and the regulation thread. It must however be emphasized that, by definition, no performance guarantee can be offered on non real-time operating systems, and unpredictable behaviors (including disastrous instability) may occur. In order to prevent hardware damage, the regulation thread uses an internal measure of its own stability. See Control Instability Detection for more details.

1.2.4 Control Instability Detection

During its execution, the regulation thread measures the jitter and delays of each iteration. Short of the thread being fully suspended by the system, these metrics allow the library to detect instability and exit gracefully, while applying the electro-magnetic brakes on the controlled device, in case of dangerous control performance degradation.

1.3 SDK Function References

- drdOpen()
- drdOpenID()
- drdSetDevice()
- drdGetDeviceID()
- drdClose()
- drdIsSupported()
- drdIsRunning()
- drdIsMoving()
- drdIsFiltering()
- drdGetTime()
- drdSleep()
- drdWaitForTick()
- drdIsInitialized()
- drdAutoInit()
- drdCheckInit()
- drdGetPositionAndOrientation()
- drdGetVelocity()
- drdGetCtrlFreq()
- drdStart()
- drdRegulatePos()
- drdRegulateRot()
- drdRegulateGrip()
- drdSetForceAndTorqueAndGripperForce()
- drdSetForceAndWristJointTorquesAndGripperForce()

- drdEnableFilter()
- drdMoveToPos()
- drdMoveToRot()
- drdMoveToGrip()
- drdMoveTo()
- drdMoveToEnc()
- drdMoveToAllEnc()
- drdTrackPos()
- drdTrackRot()
- drdTrackGrip()
- drdTrack()
- drdTrackEnc()
- drdTrackAllEnc()
- drdHold()
- drdLock()
- drdStop()
- drdGetPriorities()
- drdSetPriorities()
- drdSetEncPGain()
- drdGetEncPGain()
- drdSetEncIGain()
- drdGetEncIGain()
- drdSetEncDGain()
- drdGetEncDGain()
- drdSetMotRatioMax()
- drdGetMotRatioMax()
- drdSetEncMoveParam()
- drdSetEncTrackParam()
- drdSetPosMoveParam()
- drdSetPosTrackParam()
- drdGetEncMoveParam()
- drdGetEncTrackParam()
- drdGetPosMoveParam()
- drdGetPosTrackParam()

1.4 Technical Support

Please contact your distributor for any technical support inquiry.

2 GLOSSARY

This page describes some of the technical expressions commonly used in the documentation. Make sure to check out the **DHD glossary** as well.

Initialization
The device must be initialized before drdStart() can be called. This can be achieved in two ways: either by manually placing the device calibration pole into the calibration pit, or by calling drdAutoInit() or drdCheckInit() to perform automatic, robotically controlled initialization. The current initialization status can be checked by calling drdIsInitialized(). Initialization can also be performed manually as described in the **DHD initialization** section.

Motion Filter
When calling drdTrackPos() or drdTrackEnc(), the trajectory generation algorithm is used to to guarantee the continuity of the acceleration. Because `drdTrack*` functions can be called asynchronously, the motion generation algorithm takes the acceleration and velocity at time of calling into account and generates a smooth transition to the new regulation target (for each axis independently).

Thread Priority
The DRD allows the programmer to control the priority of the regulation thread, as well as that of the calling thread. However, thread priority and its meaning is system dependent. Please refer to your platform documentation to determine the priority level scale, as well as safety and performance consideration involved in changing thread priorities.

3 Deprecated List

Member drdGetEncDGain (char ID=-1)

Member drdGetEncIGain (char ID=-1)

Member drdGetEncPGain (char ID=-1)

Member drdSetEncDGain (double gain, char ID=-1)

Member drdSetEncIGain (double gain, char ID=-1)

Member drdSetEncPGain (double gain, char ID=-1)

4 File Index

4.1 File List

Here is a list of all documented files with brief descriptions:

5 File Documentation

5.1 drdc.h File Reference

DRD header file.

Functions

- int __SDK drdOpen ()
- int __SDK drdOpenID (char ID)
- int __SDK drdSetDevice (char ID)
- int __SDK drdGetDeviceID ()
- int __SDK drdClose (char ID=-1)
- bool __SDK drdIsSupported (char ID=-1)
- bool __SDK drdIsRunning (char ID=-1)
- bool __SDK drdIsMoving (char ID=-1)
- bool __SDK drdIsFiltering (char ID=-1)
- double __SDK drdGetTime ()
- void __SDK drdSleep (double sec)
- void __SDK drdWaitForTick (char ID=-1)
- bool __SDK drdIsInitialized (char ID=-1)
- int __SDK drdAutoInit (char ID=-1)
- int __SDK drdCheckInit (char ID=-1)
- int __SDK drdGetPositionAndOrientation (double ∗px, double ∗py, double ∗pz, double ∗oa, double ∗ob, double ∗og, double ∗pg, double matrix[3][3], char ID=-1)
- int __SDK drdGetVelocity (double ∗vx, double ∗vy, double ∗vz, double ∗wx, double ∗wy, double ∗wz, double ∗vg, char ID=-1)
- double __SDK drdGetCtrlFreq (char ID=-1)
- int __SDK drdStart (char ID=-1)
- int __SDK drdRegulatePos (bool on, char ID=-1)
- int __SDK drdRegulateRot (bool on, char ID=-1)
- int __SDK drdRegulateGrip (bool on, char ID=-1)
- int __SDK drdSetForceAndTorqueAndGripperForce (double fx, double fy, double fz, double tx, double ty, double tz, double fg, char ID=-1)
- int __SDK drdSetForceAndWristJointTorquesAndGripperForce (double fx, double fy, double fz, double t0, double t1, double t2, double fg, char ID=-1)
- int __SDK drdEnableFilter (bool on, char ID=-1)
- int __SDK drdMoveToPos (double px, double py, double pz, bool block=true, char ID=-1)
- int __SDK drdMoveToRot (double oa, double ob, double og, bool block=true, char ID=-1)
- int __SDK drdMoveToGrip (double pg, bool block=true, char ID=-1)
- int __SDK drdMoveTo (double p[**DHD_MAX_DOF**], bool block=true, char ID=-1)
- int __SDK drdMoveToEnc (int enc0, int enc1, int enc2, bool block=true, char ID=-1)
- int __SDK drdMoveToAllEnc (int enc[**DHD_MAX_DOF**], bool block=true, char ID=-1)
- int __SDK drdTrackPos (double px, double py, double pz, char ID=-1)
- int __SDK drdTrackRot (double oa, double ob, double og, char ID=-1)
- int __SDK drdTrackGrip (double pg, char ID=-1)
- int __SDK drdTrack (double p[**DHD_MAX_DOF**], char ID=-1)
- int __SDK drdTrackEnc (int enc0, int enc1, int enc2, char ID=-1)
- int __SDK drdTrackAllEnc (int enc[**DHD_MAX_DOF**], char ID=-1)
- int __SDK drdHold (char ID=-1)
- int __SDK drdLock (unsigned char mask, bool init, char ID=-1)
- int __SDK drdStop (bool frc=false, char ID=-1)
- int __SDK drdGetPriorities (int ∗prio, int ∗ctrlprio, char ID=-1)
- int __SDK drdSetPriorities (int prio, int ctrlprio, char ID=-1)
- int __SDK drdSetEncPGain (double gain, char ID=-1)
- double __SDK drdGetEncPGain (char ID=-1)
- int __SDK drdSetEncIGain (double gain, char ID=-1)
- double __SDK drdGetEncIGain (char ID=-1)
- int __SDK drdSetEncDGain (double gain, char ID=-1)
- double __SDK drdGetEncDGain (char ID=-1)
- int __SDK drdSetMotRatioMax (double scale, char ID=-1)
- double __SDK drdGetMotRatioMax (char ID=-1)
- int __SDK drdSetEncMoveParam (double amax, double vmax, double jerk, char ID=-1)

- int __SDK drdSetEncTrackParam (double amax, double vmax, double jerk, char ID=-1)
- int __SDK drdSetPosMoveParam (double amax, double vmax, double jerk, char ID=-1)
- int __SDK drdSetPosTrackParam (double amax, double vmax, double jerk, char ID=-1)
- int __SDK drdGetEncMoveParam (double ∗amax, double ∗vmax, double ∗jerk, char ID=-1)
- int __SDK drdGetEncTrackParam (double ∗amax, double ∗vmax, double ∗jerk, char ID=-1)
- int __SDK drdGetPosMoveParam (double ∗amax, double ∗vmax, double ∗jerk, char ID=-1)
- int __SDK drdGetPosTrackParam (double ∗amax, double ∗vmax, double ∗jerk, char ID=-1)

5.1.1 Detailed Description

DRD header file.

5.1.2 Function Documentation

5.1.2.1 drdAutoInit()

```
int __SDK drdAutoInit (
        char ID )
```

This function performs automatic initialization of that particular device by robotically moving to a known position and reseting encoder counters to their correct values.

Parameters

ID	**[default=-1]** device ID (see **multiple devices** section for details)

Note

In order to make regulation as stable as possible, drdAutoInit() automatically increases the priority level of the regulation thread to a higher value than the normal system process priority.

See also

drdCheckInit()

Returns

0 on success, -1 otherwise.
See **error management** for details.

5.1.2.2 drdCheckInit()

```
int __SDK drdCheckInit (
        char ID )
```

This function checks the validity of that particular device initialization by robotically sweeping all endstops and comparing their joint space position to expected values (stored in each device internal memory). If the device is not yet initialized, this function will first perform the same initialization routine as drdAutoInit() before running the endstop check. If drdCheckInit() has already been called on the device (possibly by another application), and it has not been reset or powered down, the call returns immediately and reports success. To force the device to check again regardless of previous calls, run drdAutoInit() or **dhdReset()** prior to calling drdCheckInit().

Parameters

ID	**[default=-1]** device ID (see **multiple devices** section for details)

Note

In order to make regulation as stable as possible, drdCheckInit() automatically increases the priority level of the regulation thread to a higher value than the normal system process priority.

See also

drdAutoInit()

Returns

0 on success, -1 otherwise.
See **error management** for details.

5.1.2.3 drdClose()

```
int __SDK drdClose (
        char ID )
```

This function closes the connection to a particular device.

Parameters

ID	**[default=-1]** device ID (see **multiple devices** section for details)

Returns

0 on success, -1 otherwise.
See **error management** for details.

5.1.2.4 drdEnableFilter()

```
int __SDK drdEnableFilter (
        bool on,
        char ID )
```

This function controls the motion filtering for subsequent calls to drdTrackPos() or drdTrackEnc().

Parameters

on	true to enable motion filtering, false to disable it
ID	**[default=-1]** device ID (see **multiple devices** section for details)

Returns

0 on success, -1 otherwise.
See **error management** for details.

5.1.2.5 drdGetCtrlFreq()

```
double __SDK drdGetCtrlFreq (
        char ID )
```

This function returns the average refresh rate of the control loop (in kHz) since the function was last called.

Parameters

ID	**[default=-1]** device ID (see **multiple devices** section for details)

Returns

0 on success, -1 otherwise.
See **error management** for details.

5.1.2.6 drdGetDeviceID()

```
int __SDK drdGetDeviceID ( )
```

This function returns the ID of the current **default device**.

Returns

0 on success, -1 otherwise.
See **error management** for details.

5.1.2.7 drdGetEncDGain()

```
double __SDK drdGetEncDGain (
        char ID )
```

This function retrieves the D gain of the PID controller that regulates the base joint positions.

Parameters

ID	**[default=-1]** device ID (see **multiple devices** section for details)

Returns

The P gain of the PID regulator.

Deprecated

See also

drdGetDGain()

5.1.2.8 drdGetEncIGain()

```
double __SDK drdGetEncIGain (
        char ID )
```

This function retrieves the I gain of the PID controller that regulates the base joint positions.

Parameters

ID	**[default=-1]** device ID (see **multiple devices** section for details)

Returns

The P gain of the PID regulator.

Deprecated

See also

drdGetIGain()

5.1.2.9 drdGetEncMoveParam()

```
int __SDK drdGetEncMoveParam (
        double * amax,
        double * vmax,
        double * jerk,
        char ID )
```

This function retrieves encoder positioning trajectory generation parameters.

Parameters

amax	**[out]**max acceleration (m/s2)
vmax	**[out]**max velocity (m/s)
jerk	**[out]**jerk (m/s3)
ID	**[default=-1]** device ID (see **multiple devices** section for details)

Returns

0 on success, -1 otherwise.
See **error management** for details.

5.1.2.10 drdGetEncPGain()

```
double __SDK drdGetEncPGain (
        char ID )
```

This function retrieves the P gain of the PID controller that regulates the base joint positions.

Parameters

ID	**[default=-1]** device ID (see **multiple devices** section for details)

Returns

The P gain of the PID regulator.

Deprecated

See also

drdGetPGain()

5.1.2.11 drdGetEncTrackParam()

```
int __SDK drdGetEncTrackParam (
        double * amax,
        double * vmax,
        double * jerk,
        char ID )
```

This function retrieves encoder tracking trajectory generation parameters.

Parameters

amax	**[out]**max acceleration (m/s2)
vmax	**[out]**max velocity (m/s)
jerk	**[out]**jerk (m/s3)
ID	**[default=-1]** device ID (see **multiple devices** section for details)

Returns

0 on success, -1 otherwise.
See **error management** for details.

5.1.2.12 drdGetMotRatioMax()

```
double __SDK drdGetMotRatioMax (
        char ID )
```

This function retrieves the maximum joint torque applied to all regulated joints expressed as a fraction of the maximum torque available for each joint.

Parameters

ID	**[default=-1]** device ID (see **multiple devices** section for details)

Returns

The maximum joint torque ratio (between 0.0 and 1.0)

5.1.2.13 drdGetPositionAndOrientation()

```
int __SDK drdGetPositionAndOrientation (
        double * px,
        double * py,
        double * pz,
        double * oa,
        double * ob,
        double * og,
        double * pg,
        double matrix[3][3],
        char ID )
```

This function retrieves the position of the end-effector in Cartesian coordinates. Please refer to your device user manual for more information on your device coordinate system.

Parameters

px	**[out]** device position on the X axis in [m]
py	**[out]** device position on the Y axis in [m]
pz	**[out]** device position on the Z axis in [m]
oa	**[out]** device orientation around the first joint in [rad]
ob	**[out]** device orientation around the second joint in [rad]
og	**[out]** device orientation around the third joint in [rad]
pg	**[out]** device gripper opening gap in [m]
matrix	**[out]** orientation matrix frame
ID	**[default=-1]** device ID (see **multiple devices** section for details)

Note

This function differs from **`dhdGetPositionAndOrientationFrame()`** in that it is synchronized with the robotic control loop. As a result, it does not impact control performance and returns faster than **`dhdGet←PositionAndOrientationFrame()`**.

Returns

0 on success, -1 otherwise.
See **error management** for details.

5.1.2.14 drdGetPosMoveParam()

```
int __SDK drdGetPosMoveParam (
        double * amax,
        double * vmax,
        double * jerk,
        char ID )
```

This function retrieves Cartesian positioning trajectory generation parameters.

Parameters

amax	**[out]**max acceleration (m/s2)
vmax	**[out]**max velocity (m/s)
jerk	**[out]**jerk (m/s3)
ID	**[default=-1]** device ID (see **multiple devices** section for details)

Returns

0 on success, -1 otherwise.
See **error management** for details.

5.1.2.15 drdGetPosTrackParam()

```
int __SDK drdGetPosTrackParam (
        double * amax,
        double * vmax,
        double * jerk,
        char ID )
```

This function retrieves Cartesian tracking trajectory generation parameters.

Parameters

amax	**[out]**max acceleration (m/s2)
vmax	**[out]**max velocity (m/s)
jerk	**[out]**jerk (m/s3)
ID	**[default=-1]** device ID (see **multiple devices** section for details)

Returns

0 on success, -1 otherwise.
See **error management** for details.

5.1.2.16 drdGetPriorities()

```
int __SDK drdGetPriorities (
        int * prio,
        int * ctrlprio,
        char ID )
```

This function makes it possible to retrieve the priority of the control thread and the calling thread. Thread priority is system dependent, as described in thread priorities.

Parameters

prio	**[out]** calling thread priority level (value is system dependent)
ctrlprio	**[out]** control thread priority level (value is system dependent)
ID	**[default=-1]** device ID (see **multiple devices** section for details)

Note

The first argument (`prio`) is always set to the calling thread priority, even when the call returns an error.

See also

drdSetPriorities()

Returns

0 on success, -1 otherwise.
See **error management** for details.

5.1.2.17 drdGetTime() `double __SDK drdGetTime ( )`

This function returns the current value from the high-resolution system counter in [s]. The resolution of the system counter may be machine-dependent, as it usually derived from one of the CPU clocks signals. The time returned is guaranteed to be monotonous.

Returns

The current time in [s]

5.1.2.18 drdGetVelocity()

```
int __SDK drdGetVelocity (
            double * vx,
            double * vy,
            double * vz,
            double * wx,
            double * wy,
            double * wz,
            double * vg,
            char ID )
```

This function retrieves the velocity of the end-effector position in Cartesian coordinates. Please refer to your device user manual for more information on your device coordinate system.

Parameters

vx	**[out]** velocity along the X axis [m/s]
vy	**[out]** velocity along the Y axis [m/s]
vz	**[out]** velocity along the Z axis [m/s]
wx	**[out]** angular velocity around the X axis [rad/s]
wy	**[out]** angular velocity around the Y axis [rad/s]
wz	**[out]** angular velocity around the Z axis [rad/s]
vg	**[out]** gripper linear velocity [m/s]
ID	**[default=-1]** device ID (see **multiple devices** section for details)

Note

This function differs from **`dhdGetLinearVelocity()`** and its relatives in that it is synchronized with the robotic control loop. As a result, it does not impact control performance and returns faster than **`dhdGetLinear←Velocity()`**.

Returns

0 on success, -1 otherwise.
See **error management** for details.

5.1.2.19 drdHold() `int __SDK drdHold ( char ID )`

This function immediately makes the device hold its current position. All motion commands are abandoned.

Parameters

ID	**[default=-1]** device ID (see **multiple devices** section for details)

Returns

0 on success, -1 otherwise.
See **error management** for details.

5.1.2.20 drdIsFiltering() `bool __SDK drdIsFiltering ( char ID )`

This function checks whether the particular device control thread is applying a motion filter while tracking a target using drdTrackPos() or drdTrackEnc().

Parameters

ID	**[default=-1]** device ID (see **multiple devices** section for details)

Returns

true if the motion filter is enabled, false otherwise.

5.1.2.21 drdIsInitialized() `bool __SDK drdIsInitialized ( char ID )`

This function checks the initialization status of a particular device. The initialization status reflects the status of the controller RESET LED. The device can be (re)initialized by calling drdAutoInit().

Parameters

ID	**[default=-1]** device ID (see **multiple devices** section for details)

Returns

true if the device is initialized, false otherwise.

5.1.2.22 drdIsMoving() `bool __SDK drdIsMoving ( char ID )`

This function checks whether the particular device is moving (following a call to drdMoveToPos(), drdMoveToEnc(), drdTrackPos() or drdTrackEnc()) as opposed to holding the target position after successfully reaching it.

Parameters

ID	**[default=-1]** device ID (see **multiple devices** section for details)

Returns

true if the device is moving, false otherwise.

5.1.2.23 drdIsRunning()

```
bool __SDK drdIsRunning (
        char ID )
```

This function checks the state of the robotic control thread for a particular device.

Parameters

ID	**[default=-1]** device ID (see **multiple devices** section for details)

Returns

true if the control thread is running, false otherwise.

5.1.2.24 drdIsSupported()

```
bool __SDK drdIsSupported (
        char ID )
```

This function determines if the device is supported out-of-the-box by the DRD. The regulation gains of supported devices are configured internally so that such devices are ready to use. Unsupported devices can still be operated with the DRD, but their regulation gains must first be configured using the drdSetEncPGain(), drdSetEncIGain() and drdSetEncDGain() functions.

Parameters

ID	**[default=-1]** device ID (see **multiple devices** section for details)

Returns

true if the device is configured internally, false otherwise.

5.1.2.25 drdLock()

```
int __SDK drdLock (
        unsigned char mask,
        bool init,
        char ID )
```

Depending on the value of the `mask` parameter, This function either:

- moves the device to its park position and engages the locks, or
- removes the locks.

This function only applies to devices equipped with mechanical locks, and will return an error when called on other devices.

Note

It is recommended to set the **init** argument to **true** when engaging the lock.

When the function returns, the robotic regulation is running and the device is held in its current position (locked or unlocked). Unless the device is to be used in robotic mode (e.g. to move to a desired position after unlocking), drdStop() should be called to disable regulation.

Parameters

mask	0 to disengage the locks, any other value to park the device and engage the locks
init	if **true**, then the device will auto-initialize before locks are engaged or after locks are disengaged
ID	**[default=-1]** device ID (see **multiple devices** section for details)

Returns

0 on success, -1 otherwise.
See **error management** for details.

5.1.2.26 drdMoveTo()

```
int __SDK drdMoveTo (
        double p[DHD_MAX_DOF],
        bool block,
        char ID )
```

This function sends the device end-effector to a desired Cartesian 7-DOF configuration. The motion uses smooth acceleration/deceleration. The acceleration and velocity profiles can be controlled by adjusting the trajectory generation parameters.

Parameters

p	target positions/orientations in [m] or as described in **rad**
block	if true, the call blocks until the destination is reached. If false, the call returns immediately.
ID	**[default=-1]** device ID (see **multiple devices** section for details)

Returns

0 on success, -1 otherwise.
See **error management** for details.

5.1.2.27 drdMoveToAllEnc()

```
int __SDK drdMoveToAllEnc (
        int enc[DHD_MAX_DOF],
        bool block,
        char ID )
```

This function sends the device end-effector to a desired encoder position. The motion follows a straight line in the encoder space, with smooth acceleration/deceleration. The acceleration and velocity profiles can be controlled by adjusting the trajectory generation parameters.

Parameters

enc	target encoder positions
block	if true, the call blocks until the destination is reached. If false, the call returns immediately.
ID	**[default=-1]** device ID (see **multiple devices** section for details)

Returns

0 on success, -1 otherwise.
See **error management** for details.

5.1.2.28 drdMoveToEnc()

```
int __SDK drdMoveToEnc (
        int enc0,
        int enc1,
        int enc2,
        bool block,
        char ID )
```

This function sends the device end-effector to a desired encoder position. The motion follows a straight line in the encoder space, with smooth acceleration/deceleration. The acceleration and velocity profiles can be controlled by adjusting the trajectory generation parameters.

Parameters

enc0	target encoder position on axis 0 []
enc1	target encoder position on axis 1 []
enc2	target encoder position on axis 2 []
block	if true, the call blocks until the destination is reached. If false, the call returns immediately.
ID	**[default=-1]** device ID (see **multiple devices** section for details)

Returns

0 on success, -1 otherwise.
See **error management** for details.

5.1.2.29 drdMoveToGrip()

```
int __SDK drdMoveToGrip (
        double pg,
        bool block,
        char ID )
```

This function sends the device gripper to a desired opening distance. The motion is executed with smooth acceleration/deceleration. The acceleration and velocity profiles can be controlled by adjusting the trajectory generation parameters.

Parameters

pg	target gripper opening distance in [m]
block	if true, the call blocks until the destination is reached. If false, the call returns immediately.
ID	**[default=-1]** device ID (see **multiple devices** section for details)

Returns

0 on success, -1 otherwise.
See **error management** for details.

5.1.2.30 drdMoveToPos()

```
int __SDK drdMoveToPos (
        double px,
        double py,
        double pz,
        bool block,
        char ID )
```

This function sends the device end-effector to a desired Cartesian position. The motion follows a straight line, with smooth acceleration/deceleration. The acceleration and velocity profiles can be controlled by adjusting the trajectory generation parameters.

Parameters

px	target position on the X axis in [m]
py	target position on the Y axis in [m]
pz	target position on the Z axis in [m]
block	if true, the call blocks until the destination is reached. If false, the call returns immediately.
ID	**[default=-1]** device ID (see **multiple devices** section for details)

Returns

0 on success, -1 otherwise.
See **error management** for details.

5.1.2.31 drdMoveToRot()

```
int __SDK drdMoveToRot (
        double oa,
        double ob,
        double og,
        bool block,
        char ID )
```

This function sends the device end-effector to a desired Cartesian rotation. The motion follows a straight curve, with smooth acceleration/deceleration. The acceleration and velocity profiles can be controlled by adjusting the trajectory generation parameters.

Parameters

oa	target orientation around the first joint in [rad]

Parameters

ob	target orientation around the second joint in [rad]
og	target orientation around the third joint in [rad]
block	if true, the call blocks until the destination is reached. If false, the call returns immediately.
ID	**[default=-1]** device ID (see **multiple devices** section for details)

Returns

0 on success, -1 otherwise.
See **error management** for details.

5.1.2.32 drdOpen() `int __SDK drdOpen ( )`

This function opens a connection to the first **compatible device** connected to the system. To open connections to multiple devices, use the drdOpenID() call.

Note

If this call is successful, the **default device ID** is set to the newly opened device. See the **multiple device** section for more information on using multiple devices on the same computer.

Returns

The device ID on success, -1 otherwise.
See **error management** for details.

See also

drdOpenID

5.1.2.33 drdOpenID()

```
int __SDK drdOpenID (
            char index )
```

This function opens a connection to one particular **compatible device** connected to the system. The order in which devices are opened persists until devices are added or removed. If the device at the specified index is already opened, its device ID is returned.

Parameters

index	the device enumeration index, as assigned by the underlying operating system (must be between 0 and the number of devices connected to the system)

Note

If this call is successful, the **default device ID** is set to the newly opened device. See the **multiple device** section for more information on using multiple devices on the same computer.

Returns

The device ID on success, -1 otherwise.
See **error management** for details.

See also

drdOpen

5.1.2.34 drdRegulateGrip()

```
int __SDK drdRegulateGrip (
        bool on,
        char ID )
```

This function toggles robotic regulation of the device gripper. If regulation is disabled, the gripper can move freely and will display any force set using drdSetForceAndTorqueAndGripperForce(). If it is enabled, gripper orientation is locked and can be controlled by calling all robotic functions (e.g. drdMoveTo()). By default, gripper regulation is enabled.

Parameters

on	true to enable gripper regulation, false to disable it
ID	**[default=-1]** device ID (see **multiple devices** section for details)

Returns

0 on success, -1 otherwise.
See **error management** for details.

5.1.2.35 drdRegulatePos()

```
int __SDK drdRegulatePos (
        bool on,
        char ID )
```

This function toggles robotic regulation of the device delta base, which provides translations. If regulation is disabled, the base can move freely and will display any force set using drdSetForceAndTorqueAndGripperForce(). If it is enabled, base position is locked and can be controlled by calling all robotic functions (e.g. drdMoveToPos()). By default, delta base regulation is enabled.

Parameters

on	true to enable base regulation, false to disable it
ID	**[default=-1]** device ID (see **multiple devices** section for details)

Returns

0 on success, -1 otherwise.
See **error management** for details.

5.1.2.36 drdRegulateRot()

```
int __SDK drdRegulateRot (
        bool on,
        char ID )
```

This function toggles robotic regulation of the device wrist. If regulation is disabled, the wrist can move freely and will display any torque set using drdSetForceAndTorqueAndGripperForce(). If it is enabled, wrist orientation is locked and can be controlled by calling all robotic functions (e.g. drdMoveTo()). By default, wrist regulation is enabled.

Parameters

on	true to enable wrist regulation, false to disable it
ID	**[default=-1]** device ID (see **multiple devices** section for details)

Returns

0 on success, -1 otherwise.
See **error management** for details.

5.1.2.37 drdSetDevice()

```
int __SDK drdSetDevice (
        char ID )
```

This function selects the **default device** that will receive the API commands. The API supports **multiple devices**. This routine allows the programmer to decide which device the API **dhd_single_device_call** single-device calls will address. Any subsequent API call that does not specifically mention the device ID in its parameter list will be sent to that device.

Parameters

ID	**[default=-1]** device ID (see **multiple devices** section for details)

Returns

0 on success, -1 otherwise.
See **error management** for details.

5.1.2.38 drdSetEncDGain()

```
int __SDK drdSetEncDGain (
        double gain,
        char ID )
```

This function sets the D gain of the PID controller that regulates the base joint positions. In practice, this affects the velocity of the regulation.

Parameters

gain	D parameter of the PID regulator
ID	**[default=-1]** device ID (see **multiple devices** section for details)

Deprecated

See also

drdSetDGain()

5.1.2.39 drdSetEncIGain()

```
int __SDK drdSetEncIGain (
        double gain,
        char ID )
```

This function sets the I gain of the PID controller that regulates the base joint positions. In practice, this affects the precision of the regulation.

Parameters

gain	I parameter of the PID regulator
ID	**[default=-1]** device ID (see **multiple devices** section for details)

Deprecated

See also

drdSetIGain()

5.1.2.40 drdSetEncMoveParam()

```
int __SDK drdSetEncMoveParam (
        double amax,
        double vmax,
        double jerk,
        char ID )
```

This function sets encoder positioning trajectory generation parameters.

Parameters

amax	max acceleration (m/s2)
vmax	max velocity (m/s)
jerk	jerk (m/s3)
ID	**[default=-1]** device ID (see **multiple devices** section for details)

Returns

0 on success, -1 otherwise.
See **error management** for details.

5.1.2.41 drdSetEncPGain()

```
int __SDK drdSetEncPGain (
        double gain,
        char ID )
```

This function sets the P gain of the PID controller that regulates the base joint positions. In practice, this affects the stiffness of the regulation.

Parameters

gain	P parameter of the PID regulator
ID	**[default=-1]** device ID (see **multiple devices** section for details)

Deprecated

See also

drdSetPGain()

5.1.2.42 drdSetEncTrackParam()

```
int __SDK drdSetEncTrackParam (
        double amax,
        double vmax,
        double jerk,
        char ID )
```

This function sets encoder tracking trajectory generation parameters.

Parameters

amax	max acceleration (m/s2)
vmax	max velocity (m/s)
jerk	jerk (m/s3)
ID	**[default=-1]** device ID (see **multiple devices** section for details)

Returns

0 on success, -1 otherwise.
See **error management** for details.

5.1.2.43 drdSetForceAndTorqueAndGripperForce()

```
int __SDK drdSetForceAndTorqueAndGripperForce (
        double fx,
        double fy,
        double fz,
        double tx,
        double ty,
        double tz,
        double fg,
        char ID )
```

This function applies force, torques and gripper force to all non-regulated, actuated DOFs of the device. The regulated DOFs can be selected using drdRegulatePos(), drdRegulateRot() and drdRegulateGrip(). The requested force is ignored for all regulated DOFs. You must use this function instead of all **`dhdSetForce()`** calls if the robotic regulation thread is running to prevent interfering with the regulation commands.

Parameters

fx	translation force along X axis [N]
fy	translation force along Y axis [N]
fz	translation force along Z axis [N]
tx	torque around the X axis in [Nm]
ty	torque around the Y axis in [Nm]
tz	torque around the Z axis in [Nm]
fg	gripper force in [N]
ID	**[default=-1]** device ID (see **multiple devices** section for details)

Returns

0 on success, -1 otherwise.
See **error management** for details.

5.1.2.44 drdSetForceAndWristJointTorquesAndGripperForce()

```
int __SDK drdSetForceAndWristJointTorques↩
AndGripperForce (
            double fx,
            double fy,
            double fz,
            double t0,
            double t1,
            double t2,
            double fg,
            char ID )
```

This function applies force, wrist joint torques and gripper force to all non-regulated, actuated DOFs of the device. The regulated DOFs can be selected using drdRegulatePos(), drdRegulateRot() and drdRegulateGrip(). The requested force is ignored for all regulated DOFs. You must use this function instead of all **`dhdSetForce()`** calls if the robotic regulation thread is running to prevent interfering with the regulation commands.

Parameters

fx	translation force along X axis [N]
fy	translation force along Y axis [N]
fz	translation force along Z axis [N]
t0	wrist axis 0 torque command [Nm]
t1	wrist axis 1 torque command [Nm]
t2	wrist axis 2 torque command [Nm]
fg	gripper force in [N]
ID	**[default=-1]** device ID (see **multiple devices** section for details)

Returns

0 on success, -1 otherwise.
See **error management** for details.

5.1.2.45 drdSetMotRatioMax()

```
int __SDK drdSetMotRatioMax (
            double scale,
            char ID )
```

This function sets the maximum joint torque applied to all regulated joints expressed as a fraction of the maximum torque available for each joint.

In practice, this limits the maximum regulation torque (in joint space), making it potentially safer to operate in environments where humans or delicate obstacles are present.

Parameters

scale	the joint torque scaling factor (must be between 0.0 and 1.0)
ID	**[default=-1]** device ID (see **multiple devices** section for details)

5.1.2.46 drdSetPosMoveParam()

```
int __SDK drdSetPosMoveParam (
            double amax,
            double vmax,
            double jerk,
            char ID )
```

This function sets Cartesian positioning trajectory generation parameters.

Parameters

amax	max acceleration (m/s2)
vmax	max velocity (m/s)
jerk	jerk (m/s3)
ID	**[default=-1]** device ID (see **multiple devices** section for details)

Returns

0 on success, -1 otherwise.
See **error management** for details.

5.1.2.47 drdSetPosTrackParam()

```
int __SDK drdSetPosTrackParam (
            double amax,
            double vmax,
            double jerk,
            char ID )
```

This function sets Cartesian tracking trajectory generation parameters.

Parameters

amax	max acceleration (m/s2)
vmax	max velocity (m/s)
jerk	jerk (m/s3)
ID	**[default=-1]** device ID (see **multiple devices** section for details)

Returns

0 on success, -1 otherwise.
See **error management** for details.

5.1.2.48 drdSetPriorities()

```
int __SDK drdSetPriorities (
        int prio,
        int ctrlprio,
        char ID )
```

This function makes it possible to adjust the priority of the control thread and the calling thread. Thread priority is system dependent, as described in thread priorities.

Parameters

prio	calling thread priority level (value is system dependent)
ctrlprio	control thread priority level (value is system dependent)
ID	**[default=-1]** device ID (see **multiple devices** section for details)

Note

Please keep in mind that administrator/superuser access is required on many platforms in order to increase thread priority.

The first argument (`prio`) is always applied to the calling thread, even when the call returns an error.

Returns

0 on success, -1 otherwise.
See **error management** for details.

5.1.2.49 drdSleep()

```
void __SDK drdSleep (
        double sec )
```

This function suspends the calling thread for a given duration specified in [s]-> The sleep resolution is machine and OS dependent.

Parameters

sec	sleep duration in [s]

5.1.2.50 drdStart()

```
int __SDK drdStart (
        char ID )
```

This function starts the robotic control loop for the given device. The device must be initialized (either manually or with drdAutoInit()) before drdStart() can be called successfully.

Parameters

ID	**[default=-1]** device ID (see **multiple devices** section for details)

Note

In order to make regulation as stable as possible, drdStart() automatically increases the priority level of the regulation thread to a higher value than the normal system process priority. The priority level of the regulation thread can be changed by calling drdSetPriorities().

Returns

0 on success, -1 otherwise.
See **error management** for details.

5.1.2.51 drdStop()

```
int __SDK drdStop (
        bool frc,
        char ID )
```

This function stops the robotic control loop for the given device.

Parameters

frc	if false, puts the device in **BRAKE** mode upon exiting, otherwise leaves the device in FORCE mode. See the DHD documentation for device mode details.
ID	**[default=-1]** device ID (see **multiple devices** section for details)

Returns

0 on success, -1 otherwise.
See **error management** for details.

5.1.2.52 drdTrack()

```
int __SDK drdTrack (
        double p[DHD_MAX_DOF],
        char ID )
```

This function sends the device end-effector to a desired Cartesian 7-DOF configuration. If motion filters are enabled, the motion follows a smooth acceleration/deceleration constraint on each Cartesian axis. The acceleration and velocity profiles can be controlled by adjusting the trajectory generation parameters.

Note

WARNING - If motion filters are disabled, the target position is immediately applied as the new regulation target, leading to potential discontinuities.

Parameters

p	target positions/orientations in [m] or as described in **rad**
ID	**[default=-1]** device ID (see **multiple devices** section for details)

Returns

0 on success, -1 otherwise.
See **error management** for details.

5.1.2.53 drdTrackAllEnc()

```
int __SDK drdTrackAllEnc (
        int enc[DHD_MAX_DOF],
        char ID )
```

This function sends the device end-effector to a desired encoder position. If motion filters are enabled, the motion follows a smooth acceleration/deceleration constraint on each encoder axis. The acceleration and velocity profiles can be controlled by adjusting the trajectory generation parameters.

Note

WARNING - If motion filters are disabled, the target position is immediately applied as the new regulation target, leading to potential discontinuities.

Parameters

enc	target encoder positions
ID	**[default=-1]** device ID (see **multiple devices** section for details)

Returns

0 on success, -1 otherwise.
See **error management** for details.

5.1.2.54 drdTrackEnc()

```
int __SDK drdTrackEnc (
        int enc0,
        int enc1,
        int enc2,
        char ID )
```

This function sends the device end-effector to a desired encoder position. If motion filters are enabled, the motion follows a smooth acceleration/deceleration constraint on each encoder axis. The acceleration and velocity profiles can be controlled by adjusting the trajectory generation parameters.

Note

WARNING - If motion filters are disabled, the target position is immediately applied as the new regulation target, leading to potential discontinuities.

Parameters

enc0	target encoder position on axis 0 []
enc1	target encoder position on axis 1 []
enc2	target encoder position on axis 2 []
ID	**[default=-1]** device ID (see **multiple devices** section for details)

Returns

0 on success, -1 otherwise.
See **error management** for details.

5.1.2.55 drdTrackGrip()

```
int __SDK drdTrackGrip (
            double pg,
            char ID )
```

This function sends the device gripper to a desired opening distance. If motion filters are enabled, the motion follows a smooth acceleration/deceleration. The acceleration and velocity profiles can be controlled by adjusting the trajectory generation parameters.

Note

WARNING - If motion filters are disabled, the target opening distance is immediately applied as the new regulation target, leading to potential discontinuities.

Parameters

pg	target gripper opening distance in [m]
ID	**[default=-1]** device ID (see **multiple devices** section for details)

Returns

0 on success, -1 otherwise.
See **error management** for details.

5.1.2.56 drdTrackPos()

```
int __SDK drdTrackPos (
            double px,
            double py,
            double pz,
            char ID )
```

This function sends the device end-effector to a desired Cartesian position. If motion filters are enabled, the motion follows a smooth acceleration/deceleration constraint on each Cartesian axis. The acceleration and velocity profiles can be controlled by adjusting the trajectory generation parameters.

Note

WARNING - If motion filters are disabled, the target position is immediately applied as the new regulation target, leading to potential discontinuities.

Parameters

px	target position on the X axis in [m]
py	target position on the Y axis in [m]
pz	target position on the Z axis in [m]
ID	**[default=-1]** device ID (see **multiple devices** section for details)

Returns

0 on success, -1 otherwise.
See **error management** for details.

5.1.2.57 drdTrackRot()

```
int __SDK drdTrackRot (
        double oa,
        double ob,
        double og,
        char ID )
```

This function sends the device end-effector to a desired Cartesian orientation. If motion filters are enabled, the motion follows a smooth acceleration/deceleration curve along each Cartesian axis. The acceleration and velocity profiles can be controlled by adjusting the trajectory generation parameters.

Note

WARNING - If motion filters are disabled, the target orientation is immediately applied as the new regulation target, leading to potential discontinuities.

Parameters

oa	target orientation around the first joint in [rad]
ob	target orientation around the second joint in [rad]
og	target orientation around the third joint in [rad]
ID	**[default=-1]** device ID (see **multiple devices** section for details)

Returns

0 on success, -1 otherwise.
See **error management** for details.

5.1.2.58 drdWaitForTick()

```
void __SDK drdWaitForTick (
        char ID )
```

Synchronization function: calling this function will block until the next iteration of the control loop begins.

Index