force dimension sdk

USER MANUAL

Robotic SDK
version 3.7.0

Force Dimension
Allée de la Petite Prairie 2
CH - 1260 Nyon
Switzerland

www.forcedimension.com

Contents

1 DRD - Robotic SDK Documentation 1

1.1 Disclaimer . . . 1

1.2 Introduction . . . 1

1.2.1 Regulation . . . 2

1.2.2 Trajectory Generation Parameters . . . 2

1.2.3 Non Real-Time Considerations . . . 2

1.2.4 Control Instability Detection . . . 2

1.3 SDK Function References . . . 2

1.4 Technical Support . . . 4

2 GLOSSARY 4

2.1 References . . . 4

3 File Index 5

3.1 File List . . . 5

4 File Documentation 5

4.1 drdc.h File Reference . . . 5

4.1.1 Detailed Description . . . 6

4.1.2 Function Documentation . . . 6

Index 30

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 `drdMove*` 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.

2.1 References

Please use the following image when referencing the Force Dimension SDK in your own documentation:

Please use the following website when referencing Force Dimension products and services in your own documentation:

3 File Index

3.1 File List

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

drdc.h
DRD header file 5

4 File Documentation

4.1 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)

4.1.1 Detailed Description

DRD header file.

4.1.2 Function Documentation

4.1.2.1 drdAutoInit()

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

Performs automatic initialization of that particular robot 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.

4.1.2.2 drdCheckInit()

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

Check the validity of that particular robot initialization by robotically sweeping all endstops and comparing their joint space position to expected values (stored in each device internal memory). If the robot is not yet initialized, this function will first perform the same initialization routine as drdAutoInit() before running the endstop check.

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.

4.1.2.3 drdClose()

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

Close 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.

4.1.2.4 drdEnableFilter()

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

Enable or disable 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.

4.1.2.5 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.

4.1.2.6 drdGetDeviceID()

```
int __SDK drdGetDeviceID ( )
```

Return the ID of the current **default device**.

Returns

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

4.1.2.7 drdGetEncDGain()

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

Retrieve the D term 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.

4.1.2.8 drdGetEnclGain()

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

Retrieve the I term 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.

4.1.2.9 drdGetEncMoveParam()

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

Retrieve 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.

4.1.2.10 drdGetEncPGain()

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

Retrieve the P term 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.

4.1.2.11 drdGetEncTrackParam()

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

Retrieve 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.

4.1.2.12 drdGetMotRatioMax()

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

Retrieve 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)

4.1.2.13 drdGetPositionAndOrientation()

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

Retrieve 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 X axis in [rad]
ob	**[out]** device orientation around the Y axis in [rad]
og	**[out]** device orientation around the Z axis 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 **`dhdGetPositionAndOrientationFrame()`**.

Returns

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

4.1.2.14 drdGetPosMoveParam()

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

Retrieve 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.

4.1.2.15 drdGetPosTrackParam()

```
int __SDK drdGetPosTrackParam (
              double * amax,
              double * vmax,
```

```
            double * jerk,
            char ID )
```

Retrieve 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.

4.1.2.16 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.

4.1.2.17 drdGetTime()

```
double __SDK drdGetTime ( )
```

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]

4.1.2.18 drdGetVelocity()

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

Retrieve 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 **dhdGetLinearVelocity()**.

Returns

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

4.1.2.19 drdHold()

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

Immediately make the robot 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.

4.1.2.20 drdIsFiltering()

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

Checks whether the particular robot 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.

4.1.2.21 drdIsInitialized()

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

Checks the initialization status of a particular robot. The initialization status reflects the status of the controller RESET LED. The robot can be (re)initialized by calling drdAutoInit().

Parameters

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

Returns

true if the robot is initialized, false otherwise.

4.1.2.22 drdIsMoving()

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

Checks whether the particular robot 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 robot is moving, false otherwise.

4.1.2.23 drdIsRunning()

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

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.

4.1.2.24 drdIsSupported()

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

Determine 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.

4.1.2.25 drdLock()

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

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

- move the device to its park position and engage 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

Upon success, the robotic regulation is running when the function returns. If the device has just been parked, it is recommended to call drdStop() to disable regulation.

Parameters

mask	0 to disengage the locks, any other value to park the device and engage the locks
init	if **true** and `mask` is 0, 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.

4.1.2.26 drdMoveTo()

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

Send the robot 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.

4.1.2.27 drdMoveToAllEnc()

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

Send the robot 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.

4.1.2.28 drdMoveToEnc()

```
int __SDK drdMoveToEnc (
            int enc0,
            int enc1,
            int enc2,
```

```
            bool block,
            char ID )
```

Send the robot 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.

4.1.2.29 drdMoveToGrip()

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

Send the robot 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.

4.1.2.30 drdMoveToPos()

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

Send the robot 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.

4.1.2.31 drdMoveToRot()

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

Send the robot 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 X axis in [rad]
ob	target orientation around the Y axis in [rad]
og	target orientation around the Z axis 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.

4.1.2.32 drdOpen()

```
int __SDK drdOpen ( )
```

Open 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

4.1.2.33 drdOpenID()

```
int __SDK drdOpenID (
            char ID )
```

Open a connection to one particular **compatible device** connected to the system. The order in which devices are opened is predictable and remains the same until system configuration changes.

Parameters

ID	the device ID (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

4.1.2.34 drdRegulateGrip()

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

Enable/disable 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.

4.1.2.35 drdRegulatePos()

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

Enable/disable 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.

4.1.2.36 drdRegulateRot()

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

Enable/disable 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.

4.1.2.37 drdSetDevice()

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

Select 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.

4.1.2.38 drdSetEncDGain()

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

Set the D term 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)

4.1.2.39 drdSetEncIGain()

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

Set the I term 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)

4.1.2.40 drdSetEncMoveParam()

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

Set 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.

4.1.2.41 drdSetEncPGain()

```
int __SDK drdSetEncPGain (
            double gain,
```

```
            char ID )
```

Set the P term 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)

4.1.2.42 drdSetEncTrackParam()

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

Set 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.

4.1.2.43 drdSetForceAndTorqueAndGripperForce()

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

Apply 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]

Parameters

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.

4.1.2.44 drdSetForceAndWristJointTorquesAndGripperForce()

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

Apply 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.

4.1.2.45 drdSetMotRatioMax()

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

Set 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)

4.1.2.46 drdSetPosMoveParam()

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

Set 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.

4.1.2.47 drdSetPosTrackParam()

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

Set 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.

4.1.2.48 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.

4.1.2.49 drdSleep()

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

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

Parameters

sec	sleep duration in [s]

4.1.2.50 drdStart()

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

Start the robotic control loop for the given robot. The robot 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.

4.1.2.51 drdStop()

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

Stop the robotic control loop for the given robot.

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.

4.1.2.52 drdTrack()

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

Send the robot 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.

4.1.2.53 drdTrackAllEnc()

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

Send the robot 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.

4.1.2.54 drdTrackEnc()

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

Send the robot 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.

4.1.2.55 drdTrackGrip()

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

Send the robot 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.

4.1.2.56 drdTrackPos()

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

Send the robot 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.

4.1.2.57 drdTrackRot()

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

Send the robot 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 X axis in [rad]
ob	target orientation around the Y axis in [rad]
og	target orientation around the Z axis in [rad]
ID	**[default=-1]** device ID (see **multiple devices** section for details)

Returns

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

4.1.2.58 drdWaitForTick()

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

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

Index

drdAutoInit
drdc.h, 6
drdCheckInit
drdc.h, 7
drdClose
drdc.h, 7
drdEnableFilter
drdc.h, 7
drdGetCtrlFreq
drdc.h, 8
drdGetDeviceID
drdc.h, 8
drdGetEncDGain
drdc.h, 8
drdGetEncIGain
drdc.h, 8
drdGetEncMoveParam
drdc.h, 9
drdGetEncPGain
drdc.h, 9
drdGetEncTrackParam
drdc.h, 10
drdGetMotRatioMax
drdc.h, 10
drdGetPosMoveParam
drdc.h, 11
drdGetPosTrackParam
drdc.h, 11
drdGetPositionAndOrientation
drdc.h, 10
drdGetPriorities
drdc.h, 12
drdGetTime
drdc.h, 12
drdGetVelocity
drdc.h, 13
drdHold
drdc.h, 13
drdIsFiltering
drdc.h, 14
drdIsInitialized
drdc.h, 14
drdIsMoving
drdc.h, 14
drdIsRunning
drdc.h, 14
drdIsSupported
drdc.h, 15
drdLock
drdc.h, 15
drdMoveTo
drdc.h, 16
drdMoveToAllEnc
drdc.h, 16
drdMoveToEnc
drdc.h, 16
drdMoveToGrip
drdc.h, 17
drdMoveToPos
drdc.h, 17
drdMoveToRot
drdc.h, 18
drdOpen
drdc.h, 18
drdOpenID
drdc.h, 19
drdRegulateGrip
drdc.h, 19
drdRegulatePos
drdc.h, 19
drdRegulateRot
drdc.h, 20
drdSetDevice
drdc.h, 20
drdSetEncDGain
drdc.h, 20
drdSetEncIGain
drdc.h, 21
drdSetEncMoveParam
drdc.h, 21
drdSetEncPGain
drdc.h, 21
drdSetEncTrackParam
drdc.h, 22
drdSetForceAndTorqueAndGripperForce
drdc.h, 22
drdSetForceAndWristJointTorquesAndGripperForce
drdc.h, 23
drdSetMotRatioMax
drdc.h, 23
drdSetPosMoveParam
drdc.h, 24
drdSetPosTrackParam
drdc.h, 24
drdSetPriorities
drdc.h, 24
drdSleep
drdc.h, 25
drdStart
drdc.h, 25
drdStop
drdc.h, 26
drdTrack
drdc.h, 26
drdTrackAllEnc
drdc.h, 26
drdTrackEnc
drdc.h, 27
drdTrackGrip
drdc.h, 27
drdTrackPos
drdc.h, 28
drdTrackRot
drdc.h, 28
drdWaitForTick
drdc.h, 29

drdc.h, 5
drdAutoInit, 6
drdCheckInit, 7
drdClose, 7
drdEnableFilter, 7
drdGetCtrlFreq, 8
drdGetDeviceID, 8
drdGetEncDGain, 8
drdGetEncIGain, 8
drdGetEncMoveParam, 9
drdGetEncPGain, 9
drdGetEncTrackParam, 10
drdGetMotRatioMax, 10
drdGetPosMoveParam, 11
drdGetPosTrackParam, 11
drdGetPositionAndOrientation, 10
drdGetPriorities, 12
drdGetTime, 12
drdGetVelocity, 13
drdHold, 13
drdIsFiltering, 14
drdIsInitialized, 14
drdIsMoving, 14
drdIsRunning, 14
drdIsSupported, 15
drdLock, 15
drdMoveTo, 16
drdMoveToAllEnc, 16
drdMoveToEnc, 16
drdMoveToGrip, 17
drdMoveToPos, 17
drdMoveToRot, 18
drdOpen, 18
drdOpenID, 19
drdRegulateGrip, 19
drdRegulatePos, 19
drdRegulateRot, 20
drdSetDevice, 20
drdSetEncDGain, 20
drdSetEncIGain, 21
drdSetEncMoveParam, 21
drdSetEncPGain, 21
drdSetEncTrackParam, 22
drdSetForceAndTorqueAndGripperForce, 22
drdSetForceAndWristJointTorquesAndGripperForce, 23
drdSetMotRatioMax, 23
drdSetPosMoveParam, 24
drdSetPosTrackParam, 24
drdSetPriorities, 24
drdSleep, 25
drdStart, 25
drdStop, 26
drdTrack, 26
drdTrackAllEnc, 26
drdTrackEnc, 27
drdTrackGrip, 27
drdTrackPos, 28
drdTrackRot, 28
drdWaitForTick, 29