

RTI USB Flight Recorder Blockset

Reference

For RTI USB Flight Recorder Blockset 1.2.4

Release 2021-A – May 2021

How to Contact dSPACE

Mail:	dSPACE GmbH Rathenaustraße 26 33102 Paderborn Germany
Tel.:	+49 5251 1638-0
Fax:	+49 5251 16198-0
E-mail:	info@dspace.de
Web:	http://www.dspace.com

How to Contact dSPACE Support

If you encounter a problem when using dSPACE products, contact your local dSPACE representative:

- Local dSPACE companies and distributors: <http://www.dspace.com/go/locations>
- For countries not listed, contact dSPACE GmbH in Paderborn, Germany.
Tel.: +49 5251 1638-941 or e-mail: support@dspace.de

You can also use the support request form: <http://www.dspace.com/go/supportrequest>. If you are logged on to mydSPACE, you are automatically identified and do not need to add your contact details manually.

If possible, always provide the relevant dSPACE License ID or the serial number of the CmContainer in your support request.

Software Updates and Patches

dSPACE strongly recommends that you download and install the most recent patches for your current dSPACE installation. Visit <http://www.dspace.com/go/patches> for software updates and patches.

Important Notice

This publication contains proprietary information that is protected by copyright. All rights are reserved. The publication may be printed for personal or internal use provided all the proprietary markings are retained on all printed copies. In all other cases, the publication must not be copied, photocopied, reproduced, translated, or reduced to any electronic medium or machine-readable form, in whole or in part, without the prior written consent of dSPACE GmbH.

© 2014 - 2021 by:
dSPACE GmbH
Rathenaustraße 26
33102 Paderborn
Germany

This publication and the contents hereof are subject to change without notice.

AUTERA, ConfigurationDesk, ControlDesk, MicroAutoBox, MicroLabBox, SCALEXIO, SIMPHERA, SYNECT, SystemDesk, TargetLink and VEOS are registered trademarks of dSPACE GmbH in the United States or other countries, or both. Other brand names or product names are trademarks or registered trademarks of their respective companies or organizations.

Contents

About This Reference	5
General Information on the RTI USB Flight Recorder Blockset	9
Basics on USB Flight Recorder	9
MAT File Format for the USB Flight Recorder	13
Handling the Data of the USB Flight Recorder	14
Overview of the RTI USB Flight Recorder Blockset	15
Components of the RTI USB Flight Recorder Blockset	17
USB_FLIGHT_REC_SETUP	18
Block Description (USB_FLIGHT_REC_SETUP)	18
Parameters Page (USB_FLIGHT_REC_SETUP)	19
USB_FLIGHT_REC_BLx	20
Block Description (USB_FLIGHT_REC_BLx)	20
Parameters Page (USB_FLIGHT_REC_BLx)	22
USB_FLIGHT_REC_EJECT	23
Block Description (USB_FLIGHT_REC_EJECT)	23
Unit Page (USB_FLIGHT_REC_EJECT)	24
Index	25

About This Reference

Content

This RTI Reference is a complete description of the Real-Time Interface (RTI) blocks and their settings provided by the RTI USB Flight Recorder Blockset.





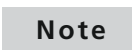



You can use this blockset to perform long-term data acquisition. The storage size is only restricted by the USB mass storage device.

The RTI USB Flight Recorder Blockset is supported by:

- DS1007 PPC Processor Board
- MicroAutoBox II
- MicroLabBox

Symbols

dSPACE user documentation uses the following symbols:

Symbol	Description
 DANGER	Indicates a hazardous situation that, if not avoided, will result in death or serious injury.
 WARNING	Indicates a hazardous situation that, if not avoided, could result in death or serious injury.
 CAUTION	Indicates a hazardous situation that, if not avoided, could result in minor or moderate injury.
 NOTICE	Indicates a hazard that, if not avoided, could result in property damage.
 Note	Indicates important information that you should take into account to avoid malfunctions.
 Tip	Indicates tips that can make your work easier.
	Indicates a link that refers to a definition in the glossary, which you can find at the end of the document unless stated otherwise.
	Precedes the document title in a link that refers to another document.

Naming conventions

dSPACE user documentation uses the following naming conventions:

%name% Names enclosed in percent signs refer to environment variables for file and path names.

< > Angle brackets contain wildcard characters or placeholders for variable file and path names, etc.

Examples:

- Where you find terms such as **rti<XXXX>** replace them by the RTI platform support you are using, for example, **rti1007**.
- Where you find terms such as **<model>** or **<submodel>** in this document, replace them by the actual name of your model or submodel. For example, if the name of your Simulink model is **smd_1007_s1.slx** and you are asked to edit the **<model>_usr.c** file, you actually have to edit the **smd_1007_s1_usr.c** file.

RTI block name conventions All I/O blocks have default names based on dSPACE's board naming conventions:

- Most RTI block names start with the board name.
- A short description of functionality is added.
- Most RTI block names also have a suffix.

Suffix	Meaning
B	Board number (for PHS-bus-based systems)
M	Module number (for MicroAutoBox II)
C	Channel number
G	Group number
CON	Converter number
BL	Block number
P	Port number
I	Interrupt number

A suffix is followed by the appropriate number. For example, **DS2201IN_B2_C14** represents a digital input block located on a DS2201 board. The suffix indicates board number 2 and channel number 14 of the block. For more general block naming, the numbers are replaced by variables (for example, **DS2201IN_Bx_Cy**).

Special folders

Some software products use the following special folders:

Common Program Data folder A standard folder for application-specific configuration data that is used by all users.

%PROGRAMDATA%\dSPACE\<InstallationGUID>\<ProductName>

or

%PROGRAMDATA%\dSPACE\<ProductName>\<VersionNumber>

Documents folder A standard folder for user-specific documents.

%USERPROFILE%\Documents\dSPACE\<ProductName>\<VersionNumber>

Local Program Data folder A standard folder for application-specific configuration data that is used by the current, non-roaming user.

%USERPROFILE%\AppData\Local\dSPACE\<InstallationGUID>\<ProductName>

Accessing dSPACE Help and PDF Files


After you install and decrypt dSPACE software, the documentation for the installed products is available in dSPACE Help and as PDF files.

dSPACE Help (local) You can open your local installation of dSPACE Help:

- On its home page via Windows Start Menu
- On specific content using context-sensitive help via **F1**

dSPACE Help (Web) You can access the Web version of dSPACE Help at www.dspace.com.

To access the Web version, you must have a *mydSPACE* account.

PDF files You can access PDF files via the  icon in dSPACE Help. The PDF opens on the first page.

General Information on the RTI USB Flight Recorder Blockset

Introduction	Provides basic information on the RTI USB Flight Recorder Blockset.
--------------	---

Where to go from here	Information in this section
-----------------------	-----------------------------

Basics on USB Flight Recorder.....	9
MAT File Format for the USB Flight Recorder.....	13
Handling the Data of the USB Flight Recorder.....	14
Overview of the RTI USB Flight Recorder Blockset.....	15

Basics on USB Flight Recorder

General information	<p>The USB Flight Recorder is used to store time histories of real-time variables. The values of real-time variables are written to an externally connected USB mass storage device during real-time simulation.</p> <p>A maximum of 250 different real-time variables can be recorded.</p> <p>The recorded data is written to a file in the root directory of the USB mass storage device. The file name is automatically generated and contains the name of the real-time application, the creation date and the creation time.</p> <p>On multicore platforms such as the DS1007 PPC Processor Board or MicroLabBox, the USB Flight Recorder is separately configured for each real-time application. A separate sequence of output files will be generated for each application.</p>
---------------------	---

The default maximum size of a single file is 32 MB. With RTLib, you can specify a maximum file size of up to 256 MB. If more than the specified maximum file size is recorded during one simulation, the data is split into several files. Consecutive files are created until the storage capacity of the USB device has been reached. Then the captured data is discarded or older files are overwritten, according to your setting (refer to [Memory overwrite mode](#) on page 10).

After the simulation has finished, the recorded data can be read out by the host PC, refer to [Handling the Data of the USB Flight Recorder](#) on page 14.

Before you power down the board, you have to stop flight recording. For information how to terminate a flight recorder session, refer to [Avoiding data loss](#) on page 11. With MicroAutoBox II's power-down feature, all currently executed applications are terminated before the hardware is shut down. The flight recording is also stopped in a definite state to avoid data loss.

Memory overwrite mode

You can use RTI or RTLib functions to define how to handle data when the USB mass storage device for flight recording is full:

Discard new data (blocked mode) When the USB mass storage device for flight recording is full, no further data is recorded.

The flight recording session is stopped, but the real-time application continues to run.

Replace old data (overwrite mode) When the USB mass storage device for flight recording is full, the oldest files are replaced.

Note

On multicore platforms such as the DS1007 PPC Processor Board or MicroLabBox, all real-time applications must use the same memory overwrite mode.

Requirements on the USB mass storage device

Any standard USB 2.0 mass storage device can be used, such as a USB memory stick or an external USB hard drive with or without separate power supply. The USB device must be formatted with the Microsoft FAT32 file system and must be directly connected to your hardware.

Note

A connection via a USB hub is not supported.

USB mass storage devices differ according to their rates for writing data. It is recommended to use a fast device for good performance.

The maximum supported file system size is 32 GB. Using a file system with a size greater than 32 GB might work but is neither recommended nor supported.

If you use an external USB hard drive with more than one partition, the flight recorder data is stored only in the first partition.

Avoiding data loss

The Windows FAT32 file system is not designed to operate in a fail-safe manner. For example, removing a USB memory stick while data is written to it can result in corrupted data.

Note

Risk of data loss

While a USB Flight Recorder session is active:

- Do not unplug the USB device from your hardware.
- Do not switch off your hardware.

You can recognize an active USB Flight Recorder session by the green flashing USB status LED.

To safely remove the USB device while an application is running, follow the instructions in this section.

To avoid partial data loss or corruption of the recorded data, you have to take the following precautions.

Removing the USB device while an application is running To safely remove the USB device while an application is running, apply one of the following methods:

- Stop the real-time application.

The USB device can be safely removed as soon as the real-time application is stopped, for example, by using ControlDesk.

- Eject by signal or button.

The method to eject the USB device differs for the board used, for example, you can eject by I/O pin, eject cable, or eject button. For detailed information, refer to the board-specific *Features* document.

- Eject by user-defined signal.

You can use the USB_FLIGHT_REC_EJECT block in your Simulink model or the **dsflrec_usb_eject** command in your handcoded application to unmount the USB device as a reaction to a signal, for example, a specific model variable.

To restart the USB Flight Recorder, you have to remove the USB device and reconnect it to your hardware.

Accessing USB Flight Recorder files via FTP

You can use any standard FTP client to retrieve USB Flight Recorder files without disconnecting the USB device from your hardware. The USB Flight Recorder files are stored in <FTP_Root>\usb. When the real-time application is not running, reading USB Flight Recorder files via FTP is safe, otherwise the following limitations apply:

- An FTP connection generates a CPU load that might result in partial data loss if the USB flight recording load is high.
- If a file is read via FTP while a flight recording session is running, incomplete data will be retrieved if the flight recorder data is written in overwrite mode.
- Any data capture session running via Ethernet might be disturbed by an FTP connection because the network bandwidth is shared.

Note

Do not download flight recorder data while the real-time application is running.

USB status LED

The status LED of the USB connector displays the current status of the USB device and the flight recorder.

LED Status	Meaning
Off	No USB device is connected.
Green	USB device is connected and flight recorder is not running.
Green blinking	USB device is connected and flight recorder is running.
Orange	USB device is full and the active flight recorder is specified not to overwrite old files.
Red	Write error when accessing the USB device, for example, if the device was removed while the flight recorder was running.

Time base

In the flight recorder, data captures are stored together with time stamps. Time stamps are measured in seconds relative to the time base 01/01/1970. Time stamps are interpreted appropriately by MATLAB or dSPACE experiment software. You can change the time base using M-program code. For an example, refer to [MAT File Format for the USB Flight Recorder](#) on page 13.

Each entry is stored together with a time stamp indicating an absolute date and time value with a resolution of 10.24 μ s.

Startup behavior and maximum data rate

The maximum data rate per application depends on the real-time platform, the USB mass storage device and the number of running applications.

Limitations using the USB Flight Recorder

Stopping the simulation Do not stop the simulation during recording, for example, by switching the simulation state from *Run* to *Stop*, and then to *Run* again. If the data is not continuously recorded, time-stamping might be corrupted.

Using a USB mass storage device There are some limitations when working with a USB mass storage device:

- It is recommended to use a separate USB mass storage device for flight recording. Other files in the root folder of the device will be deleted by the USB Flight Recorder.
- Do not use a USB hub. The device must be directly connected to your hardware.

- Using MicroAutoBox II
 - Do not manually create or delete files while the USB device is used by an active flight recorder.
 - Do not create subfolders on the USB device.
- Using DS1007 or MicroLabBox
 - If you remove the USB device while data is written, there is not only the risk of data loss. In rare cases, the USB driver of the board fails to detect a reconnected USB device. To solve the problem, you have to restart the board.

RTI/RTLib support

Using RTI You can use the RTI blocks from the RTI USB Flight Recorder blockset to write flight recorder data to the USB mass storage device, refer to [Components of the RTI USB Flight Recorder Blockset](#) on page 17.

Using RTLib You can use the `dsf1rec_usb` RTLib functions to write flight recorder data to the USB mass storage device, refer to [USB Flight Recorder \(USB Flight Recorder RTLib Reference !\[\]\(e2376d476d06eb31946dc01a69a4403a_img.jpg\)](#)).

MAT File Format for the USB Flight Recorder

Introduction

The USB Flight Recorder uses the MAT file format.

MAT file format

MAT files generated by the USB Flight Recorder contain a separate x-axis (time stamp vector) for each y-axis. To find the correct x-axis, the XIndex element of the y-axis data struct must be evaluated. For an example, see the code below.

For instructions on how to access the data of a MAT file, see [Postprocessing Recorded Data With MATLAB \(ControlDesk Measurement and Recording !\[\]\(6bb0e4f14c4133b37d2887cb37e67ddd_img.jpg\)](#)). For details on the MAT file structure, see [Structure of MAT Files Generated by Exporting Recorded Data \(ControlDesk Measurement and Recording !\[\]\(5677a36a9444aca55c9ef7a9b7d8dd5c_img.jpg\)](#)).

Changing the time base

The following program code shows you how to convert flight recorder data based on the 01.01.1970 to data relative to an entered time base in MATLAB. You have to change <MATFILENAME> and <VARNAME> with your own names.

```
load <MATFILENAME>.mat
varValues = <MATFILENAME>;
base_date = '01-Jan-1970';
plot_date = varValues.FlightRec.StartDateTime;
n = 1;
figure;
plot_start = (datenum(plot_date) - datenum(base_date))*24*3600;
use_XIndex = varValues.Y(n).XIndex;
plot(varValues.X(use_XIndex).Data - plot_start, varValues.Y(n).Data);
xlabel(strcat('sec since [', datestr(plot_start/24/3600 + datenum(base_date)), ']'));
ylabel(varValues.Y(n).Name);
```

Let the index n increase to display all the recorded data.

Plotting flight recorder data

The following program code shows you how to plot flight recorder data. You have to change <MATFILENAME> and <VARNAME> with your own names.

```
load <MATFILENAME>.mat
varValues = <MATFILENAME>;
n = 1;
figure;
use_XIndex = varValues.Y(n).XIndex;
plot(varValues.X(use_XIndex).Data, varValues.Y(n).Data);
xlabel('Time/sec');
ylabel(varValues.Y(n).Name);
```

Let the index n increase to display all the recorded data.

Related topics

Basics

[Basics on USB Flight Recorder.....](#) 9

Handling the Data of the USB Flight Recorder

Introduction

The data recorded by the USB Flight Recorder can be handled via ControlDesk.

Loading data to the host PC

After the simulation has finished, the recorded data can be downloaded to the host PC.

If the USB device is connected to your hardware, you can use ControlDesk and its specific functions for USB Flight Recorder handling to access the recorded data. You can select several binary files to download and convert them to CSV or MAT file format and to delete the binary files.

For further information, refer to [How to Upload Flight Recorder Data Written to a USB Mass Storage Device \(ControlDesk Measurement and Recording\)](#).

Alternatively, you can use an FTP client or the File Explorer to download data from the USB mass storage device. Use `ftp://<IP_Address>` to connect to your real-time hardware.


Note

Do not delete any files on the USB mass storage device while a flight recorder session is still running.
For further information, refer to *Accessing USB Flight Recorder files via FTP* in [Basics on USB Flight Recorder](#) on page 9.

If the USB device is directly connected to your PC, you can use ControlDesk's functions to load and convert a single binary file. You can also use a standard file manager, for example, the File Explorer, to copy the recorded binary files to a local drive or to delete them from the USB device.

Note

See the section *Avoiding data loss* in [Basics on USB Flight Recorder](#) on page 9 for information on how to safely remove the USB device from your hardware.

For the handling of a great amount of binary files on the USB mass storage device, or if there is no ControlDesk installed on the PC used for postprocessing the flight recorder data, you can use a command line tool for merging, extracting and converting several binary files, refer to [Merging, Extracting and Converting BIN Files of a Flight Recorder](#) ([ControlDesk Measurement and Recording](#) ).

Related topics

Basics

[Basics on USB Flight Recorder](#).....9

Overview of the RTI USB Flight Recorder Blockset

Introduction

With the USB Flight Recorder Blockset you can perform long-term data acquisition. The values of selectable variables are written to the connected USB mass storage device during simulation. The storage size is only restricted by the USB mass storage device.

Supported hardware

The RTI USB Flight Recorder blockset can be used with:

- PHS-bus-based system containing a DS1007 PPC Processor Board (DS1007 modular system)

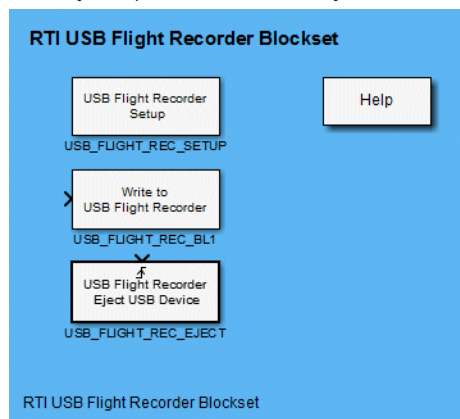
- MicroAutoBox II (any variant)
- MicroLabBox

Library access

The library can be opened with the following methods:

- Enter `rtiusbflightrec` in the MATLAB Command Window.
- Navigate to the dSPACE RTI USB Flight Recorder Blockset folder in the Simulink Library Browser to access the RTI blocks of the library separately.
- Use a platform-specific access.
 - If you use a PHS-bus-based system containing a DS1007 PPC Processor Board:
 - Click Blocksets - USB FL REC Blockset in the DS1007 blockset.
 - If you use MicroAutoBox II:
 - Click the USB Flight Recorder button in the Base Board II blockset of a MicroAutoBox II variant.
 - Click Blocksets - USB FL REC Blockset in the MicroAutoBox II blockset.
 - If you use MicroLabBox:
 - Click Blocksets - USB FL REC Blockset in the DS1202 blockset.

When you open the block library, the blockset is displayed.



Library components

The library provides the following RTI blocks:

- [USB_FLIGHT_REC_SETUP](#) on page 18
To initialize and configure the connected USB mass storage device for flight recording.
- [USB_FLIGHT_REC_BLx](#) on page 20
To save lengthy data histories of real-time variables to a USB mass storage device.
- [USB_FLIGHT_REC_EJECT](#) on page 23
To safely eject the connected USB mass storage device.

Components of the RTI USB Flight Recorder Blockset

Introduction

The RTI USB Flight Recorder Blockset provides RTI blocks that you use in the Simulink model to implement the handling of a USB mass storage device for long-term data acquisition.

Where to go from here

Information in this section

USB_FLIGHT_REC_SETUP.....	18
To initialize and configure the connected USB mass storage device for flight recording.	
USB_FLIGHT_REC_BLx.....	20
To save lengthy data histories of real-time variables to a USB mass storage device.	
USB_FLIGHT_REC_EJECT.....	23
To safely eject the connected USB mass storage device.	

Information in other sections

Basics on USB Flight Recorder.....	9
Handling the Data of the USB Flight Recorder.....	14

USB_FLIGHT_REC_SETUP

Purpose	To initialize and configure the connected USB mass storage device for flight recording.
----------------	---

Where to go from here**Information in this section**

Block Description (USB_FLIGHT_REC_SETUP).....	18
To give information about the appearance and purpose of the block.	
Parameters Page (USB_FLIGHT_REC_SETUP).....	19
To initialize the USB mass storage device for the USB Flight Recorder.	

Information in other sections

USB_FLIGHT_REC_BLx.....	20
To save lengthy data histories of real-time variables to a USB mass storage device.	

Block Description (USB_FLIGHT_REC_SETUP)

Block	To give information about the appearance and purpose of the block.
--------------	--



Purpose	To initialize and configure the connected USB mass storage device for flight recording.
----------------	---

Description	The block is used to initialize the USB Flight Recorder in your model and to specify the overwrite behavior. A new file is created on the USB mass storage device for each data acquisition that is started. If a file has reached a file size of 32 MB, it is closed and a new file is automatically opened.
--------------------	---

Note


Only one USB_FLIGHT_REC_SETUP block is allowed within one model. If you use an MP model, each submodel has to contain a USB_FLIGHT_REC_SETUP block.

Dialog pages

The dialog settings can be specified on the following pages:

- Parameters page (refer to [Parameters Page \(USB_FLIGHT_REC_SETUP\)](#) on page 19)

Related RTLib functions

This RTI block is implemented by using the RTLib functions, which are described in the [USB Flight Recorder RTLib Reference](#) .

- dsflrec_usb_initialize

Parameters Page (USB_FLIGHT_REC_SETUP)

Purpose

To initialize the USB mass storage device for the USB Flight Recorder.

Dialog settings**Overwrite mode**

Lets you select the flight recorder mode.

Mode	Meaning
Discard new data	When the USB mass storage device is full, no further data will be recorded.
Replace old data	When the USB mass storage device has reached 90% of its capacity, the oldest files will be overwritten by new files.

Related topics**References**

[Block Description \(USB_FLIGHT_REC_SETUP\)](#)..... 18

USB_FLIGHT_REC_BLx

Purpose To save lengthy data histories of real-time variables to a USB mass storage device.

Where to go from here

Information in this section

[Block Description \(USB_FLIGHT_REC_BLx\)](#)..... 20

To give information about the appearance and purpose of the block.

[Parameters Page \(USB_FLIGHT_REC_BLx\)](#)..... 22

To specify the flight recorder parameters.

Information in other sections

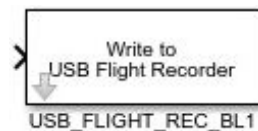
[USB_FLIGHT_REC_SETUP](#)..... 18

To initialize and configure the connected USB mass storage device for flight recording.

Block Description (USB_FLIGHT_REC_BLx)

Block

To give information about the appearance and purpose of the block.



Purpose

To save data histories of real-time variables over a longer period to a USB mass storage device.

Description

During the simulation, the values of the selected real-time variable are written to the connected USB mass storage device. After the simulation has finished, the acquired data can be read out by the host PC. For detailed information, refer to [How to Upload Flight Recorder Data Written to a USB Mass Storage Device \(ControlDesk Measurement and Recording !\[\]\(d3e32d099174a7c248ec1f564ee4f69c_img.jpg\)](#)). You can insert as many USB_FLIGHT_REC_BLx blocks into the model as desired. A maximum of 250 different real-time variables can be recorded at the same time.

Note

- Only scalar input values are allowed.
- The block requires the USB_FLIGHT_REC_SETUP block in the model.
For further information, refer to [Basics on USB Flight Recorder](#) on page 9.

I/O characteristics

The following table describes the ports of the block:


Port	Description
Input	
Write to USB Flight Recorder	Saves the recorded real-time variables to the connected USB mass storage device. Data type: Selectable (Single, Int8, UInt8, Int16, UInt16, Int32, UInt32) Range: Depends on the selected data type

Dialog pages

The dialog settings can be specified on the following pages:

- Parameters page (refer to [Parameters Page \(USB_FLIGHT_REC_BLx\)](#) on page 22)

Related RTLib functions

This RTI block is implemented by using the RTLib functions, which are described in the [USB Flight Recorder RTLib Reference](#) .

- dsflrec_usb_add_variable
- dsflrec_usb_write_int32
- dsflrec_usb_write_float32

Parameters Page (USB_FLIGHT_REC_BLx)

Purpose To specify the flight recorder parameters.

Dialog settings

Variable name Lets you enter the name of the real-time variable to be recorded (name of the input data). The recorded data is referenced by variable names.

Note

- You must specify a different variable name for each USB_FLIGHT_REC_BLx block in your model.
- A valid variable name consists of letters, digits and underscores. There are the following naming restrictions for the variable name:
 - The name must not exceed 63 characters.
 - The first character must be a letter.
 - The name must not be a keyword, such as `while` or `if`.

Data type Lets you select the recording format of the input data.

Data Type	Meaning
Single	32-bit float values
Int8	8-bit integer values
UInt8	8-bit integer values (unsigned)
Int16	16-bit integer values
UInt16	16-bit integer values (unsigned)
Int32	32-bit integer values
UInt32	32-bit integer values (unsigned)

Sample time Lets you enter the sample time in seconds for the recording.

Sample Time	Meaning
-1	Inherited sample time
0	Continuous sample time
> 0	Discrete sample time as specified

Related topics

References

[Block Description \(USB_FLIGHT_REC_BLx\)..... 20](#)

USB_FLIGHT_REC_EJECT

Purpose To safely eject the connected USB mass storage device.

Where to go from here

Information in this section

- [Block Description \(USB_FLIGHT_REC_EJECT\)..... 23](#)
To give information about the appearance and purpose of the block.
- [Unit Page \(USB_FLIGHT_REC_EJECT\)..... 24](#)
To specify the parameters for ejecting the USB mass storage device.

Information in other sections

- [USB_FLIGHT_REC_BLx..... 20](#)
To save lengthy data histories of real-time variables to a USB mass storage device.

Block Description (USB_FLIGHT_REC_EJECT)

Block To give information about the appearance and purpose of the block.



Purpose To safely eject a USB mass storage device.

Description This block can be used to avoid data loss when you unplug the USB storage device while an application is running. The block can be triggered by a rising edge, a falling edge, or both. You can also specify a function-call to trigger ejecting the device.

Note

Only one USB_FLIGHT_REC_EJECT block is allowed in your model.

I/O characteristics

The following table describes the ports of the block:


Port	Description
Trigger input	Specifies the trigger signal to eject the USB mass storage device from your hardware. Data type: Trigger input or Function-call On the Unit page, you can specify the trigger type to be used.

Dialog pages

The dialog settings can be specified on the following pages:

- Unit page (refer to [Unit Page \(USB_FLIGHT_REC_EJECT\)](#) on page 24)

Related RTLib functions

This RTI block is implemented by using the RTLib functions, which are described in the [USB Flight Recorder RTLib Reference](#) .

- dsflrec_usb_eject

Unit Page (USB_FLIGHT_REC_EJECT)

Purpose

To specify the parameters for ejecting the USB mass storage device.

Dialog settings

Trigger type Lets you select the trigger type to be used for ejecting the connected USB mass storage device.

Trigger Type	Meaning
Rising	Specifies to eject the USB mass storage device if a rising edge is detected on the trigger input.
Falling	Specifies to eject the USB mass storage device if a falling edge is detected on the trigger input.
Either	Specifies to eject the USB mass storage device if a rising edge or a falling edge is detected on the trigger input.
Function-call	Specifies to eject the USB mass storage device by function-call.

Related topics**References**

[Block Description \(USB_FLIGHT_REC_EJECT\)](#)..... 23

C

Common Program Data folder 6

D

Documents folder 7

F

flight recording
USB 9

L

Local Program Data folder 7

M

MAT format 14

R

RTI USB Flight Recorder Blockset 9
components 16, 17
hardware support 15
library access 16
overview 15

U

USB Flight Recorder 9
avoiding data loss 11
basics 9
FTP connection 11
handling the data 14
limitations 12
max. data rate 12
overwrite mode 10
USB mass storage device 10
USB status LED 12
USB_FLIGHT_REC_BLK 20
block description 20
Parameters page 22
USB_FLIGHT_REC_EJECT 23
block description 23
Unit page 24
USB_FLIGHT_REC_SETUP 18
block description 18
Parameters page 19

