Communications System Toolbox™
Support Package for RTL-SDR Radio
User Guide

R2013b
Communications System Toolbox Support Package for RTL-SDR Radio Release Notes

1

Support Package for RTL-SDR Radio (R2013b v1.0) .... 1-2
   Key Features ..................................... 1-2
   Blocks and System Objects ...................... 1-2
   RTL-SDR Examples ............................. 1-2
   Hardware and Software Requirements .......... 1-3

2

Getting Started with Communications System Toolbox Support Package for RTL-SDR Radio

SDR Support Package Description .................... 2-2

SDR Support Package Requirements .................. 2-3
   Required Products ............................... 2-3
   Recommended Products .......................... 2-3
   Third-Party Software ............................ 2-3
   Hardware and Software Requirements .......... 2-3

Setup and Configuration ............................. 2-4
   Setup and Configuration Workflow ............... 2-4
   Install RTL-SDR Support Package ............... 2-4
   Install RTL-SDR USB Driver ..................... 2-5
   Verify Hardware Setup ........................... 2-7

Configure Multiple Radios ........................... 2-9
   About USB Port Addresses ...................... 2-9

Explore RTL-SDR Radio Featured Examples .......... 2-11
### Processing RTL-SDR Radio Errors and Fixes

- sdrinfo returns empty message after calling step .................................. 2-12
- sdrinfo returns only three radio settings ........................................... 2-12
- sdrinfo returns radio ID as “vm” ......................................................... 2-13
- Cannot find radio .................................................................................. 2-13
- Simulink hangs when radio is removed during simulation ..................... 2-14

### Communications System Toolbox Support Package for RTL-SDR Radio User Guide

#### Performance Optimization ................................................................. 3-2
- General Host Operations ........................................................................ 3-2
- Model Performance Optimization ......................................................... 3-2
- MATLAB Performance Improvements ................................................. 3-4
Communications System Toolbox Support Package for RTL-SDR Radio Release Notes
Support Package for RTL-SDR Radio (R2013b v1.0)

Design and prototype software-defined radio (SDR) systems using MATLAB® and Simulink® with the Communications System Toolbox™ Support Package for RTL-SDR Radio.

For full access to features and documentation, download the support package from the Hardware Support page. To get help for the RTL-SDR Radio support package after you install it, enter `help sdr` at the MATLAB command line.

<table>
<thead>
<tr>
<th>In this section...</th>
</tr>
</thead>
<tbody>
<tr>
<td>“Key Features” on page 1-2</td>
</tr>
<tr>
<td>“Blocks and System Objects” on page 1-2</td>
</tr>
<tr>
<td>“RTL-SDR Examples” on page 1-2</td>
</tr>
<tr>
<td>“Hardware and Software Requirements” on page 1-3</td>
</tr>
</tbody>
</table>

Key Features

- RTL-SDR radio as an I/O peripheral to receive streaming RF signals
- Configurable center frequency and sample rate
- NooElec™ NESDR Mini USB Stick (R820T) and NooElec NESDR Nano USB Stick (R820T) SDR devices with frequency range 30MHz – 1.8GHz
- Compatible with other RTL-SDR USB radios (for example, Terratec T-Stick E4000).
- Several application examples for getting started

Blocks and System Objects

- Simulink radio receiver block: RTL-SDR Receiver
- MATLAB radio System object™: `comm.SDRRTLReceiver`

RTL-SDR Examples

- Spectrum Analysis with RTL-SDR Radio for MATLAB and Simulink
- Frequency Offset Calibration with RTL-SDR Radio for MATLAB and Simulink
- FM Monophonic Receiver with RTL-SDR Radio for MATLAB and Simulink
- FM Stereo Receiver with RTL-SDR Radio for MATLAB and Simulink
- FRS/GMRS Walkie-Talkie Receiver with RTL-SDR Radio for MATLAB and Simulink

Enter `sdreexamples` at the MATLAB command prompt for a full index of SDR support package examples.

**Hardware and Software Requirements**

For both MathWorks® and third-party software and hardware requirements, see RTL-SDR Support from Communications System Toolbox.
Getting Started with Communications System Toolbox Support Package for RTL-SDR Radio

- “SDR Support Package Description” on page 2-2
- “SDR Support Package Requirements” on page 2-3
- “Setup and Configuration” on page 2-4
- “Configure Multiple Radios” on page 2-9
- “Explore RTL-SDR Radio Featured Examples” on page 2-11
- “Processing RTL-SDR Radio Errors and Fixes” on page 2-12
SDR Support Package Description

The Communications System Toolbox Support Package for RTL-SDR Radio allows you to design and prototype software-defined radio (SDR) systems using MATLAB and Simulink.

When you use Communications System Toolbox in conjunction with an RTL-SDR USB radio, you can design and prototype systems that process real-time wireless signals in MATLAB and Simulink.

Key Features

- RTL-SDR radio as an I/O peripheral to receive streaming RF signals
- Configurable center frequency and sample rate
- NooElec™ NESDR Mini USB Stick (R820T) and NooElec NESDR Nano USB Stick (R820T) SDR devices with frequency range 30MHz – 1.8GHz
- Compatible with other RTL-SDR USB radios (for example, Terratec T-Stick E4000).
- Several application examples for getting started
**SDR Support Package Requirements**

<table>
<thead>
<tr>
<th>In this section...</th>
</tr>
</thead>
<tbody>
<tr>
<td>“Required Products” on page 2-3</td>
</tr>
<tr>
<td>“Recommended Products” on page 2-3</td>
</tr>
<tr>
<td>“Third-Party Software” on page 2-3</td>
</tr>
<tr>
<td>“Hardware and Software Requirements” on page 2-3</td>
</tr>
</tbody>
</table>

**Required Products**

For all support package functionality, the following products are required:

- MATLAB
- Communications System Toolbox
- DSP System Toolbox™
- Signal Processing Toolbox™

**Recommended Products**

- Simulink

**Third-Party Software**

All required third-party software is installed by the Support Package Installer.

**Hardware and Software Requirements**

For both MathWorks and third-party software and hardware requirements, see RTL-SDR Support from Communications System Toolbox.
Setup and Configuration

Setup and Configuration Workflow

1 “Install RTL-SDR Support Package” on page 2-4
2 “Install RTL-SDR USB Driver” on page 2-5
3 “Verify Hardware Setup” on page 2-7

Install RTL-SDR Support Package

**Note** If you have already installed the Support Package for RTL-SDR Radio and the USB driver, you may skip to “Verify Hardware Setup” on page 2-7.

1 Open MATLAB.

2 Click **Add-Ons** in the MATLAB Home menu.

3 Select **Get Hardware Support Packages**.
4 Follow the support package installer prompts. At **Support package to install**, select **RTL-SDR Radio**.

5 Follow the remaining prompts to download and install the support package. At any time during this process, you can click **Help** for more information about downloading support packages.

6 During support package installation, you will be prompted to install the driver needed for the RTL-SDR Radio software. See “Install RTL-SDR USB Driver” on page 2-5.

**Install RTL-SDR USB Driver**

**Note** If you have already installed the Support Package for RTL-SDR Radio and the USB driver, you may skip to “Verify Hardware Setup” on page 2-7.

This step is a continuation of the installation process. During the support package installation, you are guided through using Zadig software to install the driver required for using the RTL-SDR device as a Software Defined Radio (SDR) receiver. The Zadig software is automatically downloaded as part of the support package installation. These instructions are for installing the USB driver for one RTL-SDR device. To install more than one RTL-SDR device for use with the Support Package for RTL-SDR Radio, see “Configure Multiple Radios” on page 2-9.
Before you begin, we recommend that you remove all other non-essential USB devices from your computer so that you do not accidentally replace the driver for a different device.

- “Insert RTL-SDR Radio into USB Port” on page 2-6
- “Install USB Driver with Zadig Software” on page 2-6
- “Check Radio” on page 2-7

**Insert RTL-SDR Radio into USB Port**

1. Insert the RTL-SDR radio into an available USB port on your computer. Your operating system (OS) may automatically install a driver for the device as a DVB-T receiver. If this happens, wait until the OS finishes the installation before continuing.

2. Click Next.

Note that in the future, you must use the same USB port for this particular RTL-SDR radio in the same MATLAB session. If you want to use a different port, you must re-run the installation for that port.

**Install USB Driver with Zadig Software**

As part of the installation process, the installer launches the Zadig software in a separate window. You might see a “User Account Control” prompt, as the Zadig software may require administrative privileges to install the USB driver. If you see this prompt, click Yes to continue.

Perform the following steps in the Zadig software window:

1. In the Options menu, check that **List All Devices** is selected.

2. In the pull-down menu, just below the menu bar, select the following device: **Bulk-In, Interface (Interface 0)**

3. In the Driver selector, at the green arrow, check that **WinUSB** is selected.
4 Click the button that reads “Replace Driver” (it may also read “Install Driver” or “Re-install Driver”).

If you see a Windows Security dialog with the message "Windows can’t verify the publisher of this driver software", click "Install this driver software anyway."

When the installation completes successfully, you will see a dialog with a message “The driver was installed successfully”. Click OK to close the dialog. You will also see “Driver Installation: SUCCESS” displayed in the status field at the bottom of the Zadig window. Exit the Zadig software window.

Check Radio

On this screen you can see all the RTL-SDR radios connected to your computer. You can click the "Refresh Radio List" button to refresh the list.

If you see at least one radio in the list, your setup has been successful. If you do not see any radios in the list, see “Processing RTL-SDR Radio Errors and Fixes” on page 2-12.

Verify Hardware Setup

You can check the RTL-SDR hardware connections to your computer using the function sdrinfo.

1 To get information for all radios connected to your computer, enter the function in this format at the MATLAB command prompt:

   hwinfo = sdrinfo

To get information on a specific radio, enter the function with this format:

   hwinfo = sdrinfo(Address)

Where Address is the radio ID, for example:

   hwinfo = sdrinfo('0');
• If `sdrinfo` is successful, MATLAB returns a hardware information structure. It may look similar to the following:

```matlab
ans =

    RadioName: 'ezcap USB 2.0 DVB-T/DAB/FM dongle'
    RadioAddress: '0'
    RadioIsOpen: 0
    TunerName: 'R820T'
    Manufacturer: 'Realtek'
    Product: 'RTL2838UHIDIR'
    GainValues: [29x1 double]
    RTLCrystalFrequency: 28800000
    TunerCrystalFrequency: 28800000
    SamplingMode: 'Quadrature'
    OffsetTuning: 'Disabled'
```

• If `sdrinfo` returns the message *There was no response at address Address*, or if you see the following message:

```matlab
>> sdrinfo('0')
ans =
{}
```

Check for:
- No radio connected
- An improperly specified radio ID
- USB driver is not installed properly

2 If the connection is successful, go on to “Explore RTL-SDR Radio Featured Examples” on page 2-11.
Configure Multiple Radios

To install more than one RTL-SDR device, perform the following steps for each additional device, and configure only one device at a time:

1. Plug the device into an available USB port on your computer. You must use a separate USB port for each RTL-SDR device.

2. Run `targetUpdater` in the MATLAB command window.

These steps will configure the USB ports to work with RTL-SDR radios without requiring you to download and install the support package each time.

About USB Port Addresses

USB address are not assigned in a manner that it is intuitive. If a device is plugged into the first USB port, it is always at address 0. But if there is a device plugged into a second or third port, the address varies depending on whether there is a device in any of the previous ports. See the following illustrations for a visual explanation.
This diagram illustrates how the USB port addresses are assigned when devices are added and then removed in FIFO order.

RTL devices at radio addresses 0 and 1.

Remove the RTL device from the first USB slot.

The second RTL device, previously at radio address 1, is now at radio address 0.

Insert the first RTL device back into the first USB slot.

The second RTL device is now back at radio address 1, and the first RTL device has been reassigned radio address 0.

This diagram illustrates how the USB port addresses are assigned when devices are added and then removed from port 0. Note that the radio address for the second RTL device changes based on whether or not the first port has a device in it. You must keep track of the plugged-in devices in order to access a radio with the correct radio address.
Explore RTL-SDR Radio Featured Examples

The Support Package for RTL-SDR Radio comes with several featured Examples, for both MATLAB and Simulink:

- Spectrum Analysis with RTL-SDR Radio
- Frequency Offset Calibration with RTL-SDR Radio
- FM Monophonic Receiver with RTL-SDR Radio
- FM Stereo Receiver with RTL-SDR Radio
- FRS/GMRS Walkie-Talkie Receiver with RTL-SDR Radio

These examples can help you get started writing your own applications and exploring all the features of the Support Package for RTL-SDR Radio. To access these examples, enter `sdreexamples` at the MATLAB command prompt.
Processing RTL-SDR Radio Errors and Fixes

<table>
<thead>
<tr>
<th>In this section...</th>
</tr>
</thead>
<tbody>
<tr>
<td>“sdrinfo returns empty message after calling step” on page 2-12</td>
</tr>
<tr>
<td>“sdrinfo returns only three radio settings” on page 2-12</td>
</tr>
<tr>
<td>“sdrinfo returns radio ID as &quot;vm&quot;” on page 2-13</td>
</tr>
<tr>
<td>“Cannot find radio” on page 2-13</td>
</tr>
<tr>
<td>“Simulink hangs when radio is removed during simulation” on page 2-14</td>
</tr>
</tbody>
</table>

sdrinfo returns empty message after calling step

Problem

Function sdrinfo returns an empty cell array

Possible Solution

The radio may not be connected or configured correctly. Follow these steps to reconfigure the port and radio:

1. Connect the radio
2. Install the USB driver using targetupdater
3. Run sdrsetup

sdrinfo returns only three radio settings

Problem

Function sdrinfo does not return all radio settings for the specified device.
Possible Solution

`sdrinfo` returns all valid radio fields that are applicable, as long as the System object or block is not open (unlocked). If the block or System object is currently running, `sdrinfo` returns only the top three fields: `RadioName`, `RadioAddress`, and `RadioIsOpen`.

If this is the case, you can do one of the following actions:

- Release the System object
- Stop the model simulation
- Close the block mask

**sdrinfo returns radio ID as “vm”**

Problem

In the Zadig software window, the USB driver is reported as RTL2838UHIDIR. When you hover your mouse over vendor ID field (“USB ID” row first text field), it reports the vendor as “VM”.

Possible Solution

This issue can happen sometimes when the VMPlayer is running on Windows®. Turn off the VMplayer and try again. You can start the VMPlayer after your computer recognizes the RTL-SDR radio correctly.

**Cannot find radio**

Problem

The RTL-SDR radio is attached to your computer but when you run `sdrinfo`, it returns empty or the info method of an RTL-SDR System object returns "Cannot find radio".
**Possible Solution**

Unplug the radio and then plug the radio back in again. Be sure to use the same USB slot.

**Simulink hangs when radio is removed during simulation**

**Problem**

Simulink hangs if the RTL radio is removed while running a simulation.

**Possible Solution**

You must terminate the MATLAB session and restart.

In the future, do not remove an RTL radio while it is in use.
Communications System Toolbox Support Package for RTL-SDR Radio User Guide
Performance Optimization

In this section...

| “General Host Operations” on page 3-2 |
| “Model Performance Optimization” on page 3-2 |
| “MATLAB Performance Improvements” on page 3-4 |

General Host Operations

Tune your computing environment to improve performance:

- Turn off antivirus and firewall software.
- Turn off all nonessential background processes on your computer.

Model Performance Optimization

Performance improvements can help you achieve real-time execution. The following lists contains performance improvements you can make to models that contain the RTL-SDR Receiver block.

- “Acceleration” on page 3-2
- “Model Tuning” on page 3-3
- “Simulink Code Generation” on page 3-3

Acceleration

Run your application in Accelerator or Rapid Accelerator mode instead of Normal mode. Be aware that some scopes do not plot data when run in Rapid Accelerator mode.

When you use Accelerator or Rapid Accelerator mode, set Model Configuration Parameters > Optimization > Compiler optimization level to Optimizations on (faster runs).
Model Tuning

- Use frame-based processing. With frame-based processing, the model processes multiple samples during a single execution call to a block. Consider using frame sizes from roughly 100 to several thousand.

- In Model Configuration Parameters > Data Import/Export, turn off all logging.

- The model must be single-rate. If the model requires resampling, then choose rational coefficients that keep the model single-rate.

- Do not add any Buffer blocks to the model. If you want to create convenient frame sizes, do it in your data sources. Using a Buffer block typically degrades performance.

- Avoid feedback loops. Typically, such loops imply scalar processing, which slows the model considerably.

- Avoid using scopes. To visualize your data, send it to a workspace variable and post-process it.

- If the model has many Constant blocks, check Model Configuration Parameters > Optimization (Signals and Parameters) > Inline parameters. This setting causes the sample time of those Constant blocks to become inf, and Simulink gets the values only once during a run.

Simulink Code Generation

- If you are generating code from the model, set the Solver setting to Fixed-step/discrete. Set tasking mode to SingleTasking.

- To improve performance, you can generate a standalone executable for your Simulink model. The generated code runs without Simulink in the loop. To perform any code generation, you must have an appropriate compiler installed. See http://www.mathworks.com/support/compilers/ for compilers supported in the current release.

You can generate generic real-time target (GRT) code if you have a Simulink Coder™ license. To do so, set Model Configuration Parameters > Code Generation > System target file to grt.tlc (Generic Real-Time Target).
When you select the option to generate code for any target (not just GRT), clear the Model Configuration Parameters > Hardware Implementation > Test hardware > Test hardware is the same as production hardware check box. Then, set the Device type to MATLAB Host Computer.

You can create generated code with a smaller stack than the GRT code if you have an Embedded Coder® license. To do so, set Model Configuration Parameters > Code Generation > System target file to ert.tlc (Embedded Coder). Then, add the following lines to the Model Configuration Parameters > Code Generation > Custom Code > Include custom C code in generated: > Source file:

```c
#include <stdio.h>
#include <stdlib.h>
```

**MATLAB Performance Improvements**

- “Vector-Based Processing” on page 3-4
- “MATLAB Code Generation” on page 3-4

**Vector-Based Processing**

- Use vector-based processing. With vector-based processing, the program processes multiple samples during a single execution call to a System object. Consider using vectors from roughly 256 to several thousand. The default is 1024.
- Use large vectors of data to minimize function call overhead.

**MATLAB Code Generation**

You can accelerate your MATLAB algorithms by generating a MEX function using the MATLAB Coder function `codegen`.

Use `codegen` to generate a MEX function from MATLAB code. The following example generates a MEX file called `sdrrExMex` from the function `sdrrExample`:

```bash
codegen sdrrExample -args {ones(10,1)} -o sdrrExMex -g launchreport
```
For a full list of syntax options and input parameters, see the `codegen` reference page.

---

**Note** `sdrExample` is used only for illustrative purposes; it is not a function shipped with the RTL-SDR Radio support package. You must provide your own function for `codegen`. 
Blocks — Alphabetical List
RTL-SDR Receiver

**Purpose**  
Receive data from RTL-SDR

**Library**  
Communications System Toolbox Support Package for RTL-SDR

### Description

RTL-SDR Receiver block with default ports

RTL-SDR Receiver block with all ports enabled

The RTL-SDR Receiver block is a signal source that receives data from an RTL-SDR radio and outputs a column vector signal of fixed length specified by the samples per frame parameter.

The following block diagram illustrates how Simulink, the RTL-SDR Receiver block, and the USRP® hardware interface.
If your computer is not connected to any RTL-SDR hardware, you can still use this block to develop a model that propagates sample time and data type information. To propagate this information, select Edit > Update diagram; alternatively, you can press Ctrl + D.

To open the RTL-SDR Block library, enter the following at the MATLAB prompt:

`sdrrlib`

**Supported Data Types**

<table>
<thead>
<tr>
<th>Port</th>
<th>Supported Data Types</th>
</tr>
</thead>
<tbody>
<tr>
<td>Output</td>
<td>The output port supports these complex data types only:</td>
</tr>
<tr>
<td></td>
<td>• Double-precision floating point</td>
</tr>
<tr>
<td></td>
<td>• Single-precision floating point</td>
</tr>
<tr>
<td></td>
<td>• 16-bit signed integers</td>
</tr>
</tbody>
</table>
RTL-SDR Receiver

Dialog Box and Parameters

![Source Block Parameters: RTL-SDR Receiver](image)

**RTL-SDR Receiver**
Receive data from an RTL-SDR radio.

**Radio Connection**
Radio address: 0

**Radio Configuration**
Initialize the attached radio with block parameters and get device configuration values.

<table>
<thead>
<tr>
<th>Property Name</th>
<th>Source</th>
<th>Desired Value</th>
<th>Device Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>Center frequency (Hz)</td>
<td>Property</td>
<td>102.5e6</td>
<td>1.025e+08</td>
</tr>
<tr>
<td>Tuner gain (dB)</td>
<td>AGC</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Sampling rate (Hz)</td>
<td></td>
<td>250e3</td>
<td>250000</td>
</tr>
<tr>
<td>Frequency correction (ppm)</td>
<td>0</td>
<td></td>
<td>0</td>
</tr>
</tbody>
</table>

**Data Transfer Configuration**
- [ ] Lost samples output port
- [ ] Latency output port
- Output data type: int16
- Samples per frame: 1024

**Hardware Information**
- RadioName: 'ezcap USB 2.0 DVB-T/DAB/FM dongle'
- RadioAddress: '0'
- TunerName: 'R820T'
- Manufacturer: 'Realtek'
- Product: 'RTL2838UHIDIR'
- GainValues: [1x29 double]
- RTLCrystalFrequency: 28800000
- TunerCrystalFrequency: 28800000
- SamplingMode: 'Quadrature'
- OffsetTuning: 'Disabled'
Radio address
USB address of the RTL receiver. Must be a string containing a scalar, non-negative integer value. The default is '0'. 
**About USB port addresses** USB address are not assigned in a manner that it is intuitive. If a device is plugged into the first USB port, it is always at address 0. But if there is a device plugged into a second or third port, the address varies depending on whether there is a device in any of the previous ports. See the following illustrations for a visual explanation.

This diagram illustrates how the USB port addresses are assigned when devices are added and then removed in FIFO order.

[Diagrams showing USB port address assignment]

This diagram illustrates how the USB port addresses are assigned when devices are added and then removed from port 0. Note that the radio address for the second RTL device changes based on whether or not the first port has a device in it. You must keep track of the plugged-in devices in order to access a radio with the correct radio address.

[Diagrams showing USB port address assignment]
**Initialize**
Update block parameters in model. Clicking this button also causes the **Hardware Information** panel to be updated with current radio information.

**Center frequency (Hz)**
Specifies the center frequency of the input signal for the RTL-SDR radio. You can specify this parameter using the block dialog mask or a block input port. If you specify a value out of range for your device, you will get an error.

**Tuner gain (dB)**
Specify the desired tuner gain as coming from a block parameter setting (Property), a block input port (Input Port), or via automatic gain control (AGC). The default is AGC. The valid range of this property depends on the tuner chip of the RTL-SDR radio. However, if you specify a value out of range for the device, the actual tuner gain value is quantized to the minimum or maximum valid value.

**Sampling rate (Hz)**
Specify the desired sampling rate as a double-precision, nonnegative scalar. The default value is $250\times 10^3$ MHz. The maximum sampling rate is $3.2$ MHz.

**Frequency correction (ppm)**
Specify the frequency correction value in ppm as an integer. The valid range is $[-1\times 10^4, 1\times 10^4]$ ppm. The default value is 0. This parameter value is used to correct for frequency shifts that occur as a result of local oscillator offsets or clock rate inaccuracies.

**Lost samples output port**
Select this parameter to instruct the RTL-SDR Receiver block to output the number of lost samples during host-hardware data transfers.
- Zero indicates no data loss.
- A positive number indicates that overruns occurred.
The default value is not selected, which means that the port is not enabled and no information about dropped packets is displayed.

These ports are useful diagnostic tools for determining real-time operation of the block. If your model is not running in real time, try the techniques described in the section on Performance Maximization to approach or achieve real-time performance.

**Latency output port**
Select this parameter to instruct the RTL-SDR Receiver block to indicate latency during host-hardware data transfers in number of frames.

**Output data type**
Specify the complex output data type as double, single, or int16. When you select double or single for the output data type, the complex values are scaled to the range of [-1,1]. When you select int16, the complex values are the raw 16-bit I and Q samples from the board. The default is int16.

This block supports the following complex output data types:
- Double-precision floating point
- Single-precision floating point
- 16-bit signed integers

**Samples per frame**
Specify the number of samples in a frame for the block to output. This value must be a positive, scalar integer that is an integer multiple of 256. The default value is 1024.

**Usage**
- “Block Connectivity” on page 4-9
- “Desired vs. Actual Device Block Parameter Values” on page 4-9
- “Real Time Operation” on page 4-9
**Block Connectivity**

You can verify that your RTL-SDR Receiver block is connected to the RTL device with the **Initialize** button on the block.

1. Open the block mask.
2. Click the **Initialize** button.

If the block is not connected, the panel will show the following message:

No attached SDR hardware or unable to retrieve hardware information.

**Desired vs. Actual Device Block Parameter Values**

When you set block values for center frequency, tuner gain, sampling rate, and frequency correction, the block initially performs some rudimentary checks that the values are scalar and real. If your values pass those checks, you can still provide values that are out of range for the RTL-SDR radio. In that case, the hardware will make a best effort to set the requested value, and will report the actual value in the **Device value** column of the block mask.

**Real Time Operation**

You can check if your model runs in real time using the two optional outputs, **Lost samples output port** and **Latency output port**. If your model runs in real-time, the lost samples output is always zero. Even when your model runs in real-time, received signals may experience latency. You can observe latency in real-time using the latency output port.

These port are useful diagnostic tools for determining real time operation of the block. If your model is not running in real time, try the techniques described in “Performance Optimization” on page 3-2 to approach or achieve real-time performance.

**Examples**

See either of the following RTL-SDR Radio featured examples:

- FM Monophonic Receiver with RTL-SDR Hardware
RTL-SDR Receiver

- FM Stereo Receiver with RTL-SDR Hardware

For more examples featuring the RTL-SDR Receiver block, enter sdreexamples at the MATLAB command prompt.
System Objects —
Alphabetical List
Purpose

Receive data from RTL-SDR radio

Description

The SDRRTLReceiver System object supports communication between MATLAB and an RTL-SDR radio, by way of an RTL-SDR device attached to a USB port on the host computer, allowing simulation and development of various software-defined radio (SDR) applications. Although comm.SDRRTLReceiver receives data from an RTL-SDR radio, the object acts as a signal source that outputs a column vector signal of fixed length.

Supported Data Types

<table>
<thead>
<tr>
<th>Port</th>
<th>Supported Data Types</th>
</tr>
</thead>
<tbody>
<tr>
<td>Output</td>
<td>The output port supports these complex data types only:</td>
</tr>
<tr>
<td></td>
<td>• Double-precision floating point</td>
</tr>
<tr>
<td></td>
<td>• Single-precision floating point</td>
</tr>
<tr>
<td></td>
<td>• 16-bit unsigned integers</td>
</tr>
</tbody>
</table>

Construction

H = comm.SDRRTLReceiver creates an RTL-SDR radio receiver System object, H, that receives data from the RTL-SDR radio.

H = comm.SDRRTLReceiver(Name,Value) creates an RTL-SDR radio receiver object, H, with the specified property Name set to the specified Value. You can specify additional name-value pair arguments in any order as (Name1,Value1,...,NameN,ValueN).

H = comm.SDRRTLReceiver(RadioAddress,Name,Value) creates an RTL-SDR radio receiver System object, H, where RadioAddress is a
string containing a non-negative, scalar integer that refers to the USB ID of the RTL-SDR radio, and the other specified properties set to the specified values.

**Properties**

**RadioAddress**

USB address of the RTL-SDR device

Specify the USB address of the radio you want to communicate with as a string. The default value is '0'. Use the `sdrinfo` function to discover which radios are connected to your computer.
About USB port addresses  USB address are not assigned in a manner that it is intuitive. If a device is plugged into the first USB port, it is always at address 0. But if there is a device plugged into a second or third port, the address varies depending on whether there is a device in any of the previous ports. See the following illustrations for a visual explanation.

This diagram illustrates how the USB port addresses are assigned when devices are added and then removed in FIFO order.

This diagram illustrates how the USB port addresses are assigned when devices are added and then removed from port 0. Note that the radio address for the second RTL device changes based on whether or not the first port has a device in it. You must keep track of the plugged-in devices in order to access a radio with the correct radio address.
CenterFrequency

Desired center frequency in Hz

Specify the desired center frequency as a double precision, nonnegative scalar. The default value is 102.5 MHz. The valid range of this property depends on the tuner chip of the RTL-SDR radio. You can get a list of tuner chips and their frequency ranges at http://sdr.osmocom.org/trac/wiki/rtl-sdr#SupportedHardware and http://sdr.osmocom.org/trac/wiki/rtl-sdr#Specifications.

EnableTunerAGC

Turn the tuner automatic gain control (AGC) on (true) or off (false).

When set to true, this property will enable the tuner AGC to provide almost constant amplitude signals. The default value is true.

TunerGain

Desired tuner gain in dB

Specify the desired tuner gain as a double precision scalar. The default is 0 dB. The valid range of this property depends on the tuner chip of the RTL-SDR radio. Use the info method to get valid values. This property applies and appears only when EnableTunerAGC is set to false. This property is tunable.

SampleRate

Desired ADC sample rate in Hz

Specify the desired sample rate of the output samples of the step method, in Hertz, as a double-precision, positive, scalar value. The default is 250 KHz. The maximum ADC sample rate is 3.2 MHz.

OutputDataType

Data type of output

Specify the output data type as one of double, single, or int16. When you select double or single data type, the complex values...
are scaled to the range of [-1,1]. When selecting int16, the complex values are the raw 16-bit I and Q samples from the board. The default value is int16.

**SamplesPerFrame**

Number of samples in a frame

Specify the number of samples in a frame for the step method to output. This value must be a positive, scalar integer that is an integer multiple of 256. The default value is 1024.

**FrequencyCorrection**

Frequency correction value in ppm

Specify the frequency correction value in ppm as an integer. The valid range is [-1e4 1e4] ppm. The default value is 0. This property value is used to correct for frequency shifts that occur as a result of local oscillator offsets or clock rate inaccuracies.

**Methods**

- info: Obtain RTL-SDR radio information
- isLocked: Locked status (logical)
- release: Allow property value and input characteristics changes
- reset: Reset the internal states of the RTL-SDR radio System object
- step: Receive data from RTL-SDR radio

**Usage**

- “Object Connectivity” on page 5-7
- “Desired vs. Actual Device System Object Parameter Values” on page 5-8
Object Connectivity

Verify that your SDRRTLReceiver System object is connected to an RTL-SDR radio by using the info method.

1 Construct an SDRRTLReceiver System object.

```matlab
h = comm.SDRRTLReceiver
h =

System: comm.SDRRTLReceiver

Properties:
    RadioAddress: '0'
    CenterFrequency: 102500000
    EnableTunerAGC: true
    SampleRate: 250000
    OutputDataType: 'int16'
    SamplesPerFrame: 1024
    FrequencyCorrection: 0
```

2 Use the info method:

```matlab
S = info(h)

The function returns the hardware information for object h in structure S. Note that the values returned show the actual current radio settings, not the property values of the RTL-SDR receiver System object.

S =

    RadioName: 'ezcap USB 2.0 DVB-T/DAB/FM dongle'
    RadioAddress: '0'
    TunerName: 'R820T'
    Manufacturer: 'Realtek'
    Product: 'RTL2838UHIDIR'
    GainValues: [1x29 double]
**comm.SDRTLReceiver**

- **RTLCrystalFrequency**: 28800000
- **TunerCrystalFrequency**: 28800000
- **SamplingMode**: 'Quadrature'
- **OffsetTuning**: 'Disabled'
- **CenterFrequency**: 102500000
- **SampleRate**: 250000
- **FrequencyCorrection**: 0

Note that **TunerGain** does not appear if **EnableAGC** is true (which, in this example, it is).

The info() method first initializes the radio with the System object properties, and then shows the actual values of center frequency, sample rate, and tuner gain. Note that the System object properties show user requested values and may be different than the values reported by the info() method due to quantization.

**Desired vs. Actual Device System Object Parameter Values**

You can set the desired values in the receiver System object for center frequency, gain, sample rate, and frequency correction. However, due to quantization or range issues, the actual values can differ from your desired values.

The System object properties always shows the requested values. For example, h.CenterFrequency displays the requested center frequency set by you. To see the actual center frequency set at the radio, call the info method, \( r = \text{info}(h) \): r.CenterFrequency shows the actual center frequency. The System object sends the requested values to the radio and reads the actual values back when either the info method is called or when step() is called.

<table>
<thead>
<tr>
<th>Parameter to Set</th>
<th>Actual Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>h.CenterFrequency</td>
<td>r.CenterFrequency</td>
</tr>
<tr>
<td>h.TunerGain</td>
<td>r.TunerGain</td>
</tr>
<tr>
<td>h.SampleRate</td>
<td>r.SampleRate</td>
</tr>
<tr>
<td>h.FrequencyCorrection</td>
<td>r.FrequencyCorrection</td>
</tr>
</tbody>
</table>
Examples

See either of the following featured examples for some basic radio functionality with the RTL-SDR Radio support package:

- FM Monophonic Receiver with RTL-SDR Hardware
- FM Stereo Receiver with RTL-SDR Hardware

For more examples featuring the RTL-SDR radio receiver System object, enter `sdrexamples` at the MATLAB command prompt.
**Purpose**
Obtain RTL-SDR radio information

**Syntax**
\[ S = \text{info}(H) \]

**Description**
\( S = \text{info}(H) \) returns a structure, \( S \), containing characteristic information for the \text{SDRRTLReceiver} System object, \( H \). If \( S \) has only two fields, \text{RadioName}, which has the value “Cannot find radio”, and \text{RadioAddress}, then the \text{info} method cannot read the desired radio information. Otherwise, \( S \) contains the radio information described in “Output Arguments” on page 5-10.

Note that the values returned show the actual current radio settings, not the property values of the RTL-SDR receiver System object. The \text{info()} method first initializes the radio with the System object properties, and then shows the actual values of center frequency, sample rate, and tuner gain. Note that the System object properties show user requested values and may be different than the values reported by the \text{info()} method due to quantization.

**Input Arguments**
\( H \)
Instance of \text{SDRRTLReceiver}

**Output Arguments**
\( S \)
Structure containing characteristic information for the radio connected to the \text{SDRRTLReceiver} System object.

<table>
<thead>
<tr>
<th>Field</th>
<th>Contains</th>
</tr>
</thead>
<tbody>
<tr>
<td>\text{RadioName}</td>
<td>Device name</td>
</tr>
<tr>
<td>\text{RadioAddress}</td>
<td>USB address of the RTL-SDR radio</td>
</tr>
<tr>
<td>\text{RadioIsOpen}</td>
<td>Flag for open devices</td>
</tr>
<tr>
<td>\text{TunerName}</td>
<td>Tuner chip name</td>
</tr>
<tr>
<td>\text{Manufacturer}</td>
<td>Manufacturer name</td>
</tr>
<tr>
<td>Field</td>
<td>Contains</td>
</tr>
<tr>
<td>--------------------------</td>
<td>--------------------------------------------------------------------------</td>
</tr>
<tr>
<td>Product</td>
<td>Product name</td>
</tr>
<tr>
<td>GainValues</td>
<td>Tuner chip supported gain values, in dB</td>
</tr>
<tr>
<td>RTLCrystalFrequency</td>
<td>RTL chip crystal oscillator frequency, in Hz</td>
</tr>
<tr>
<td>TunerCrystalFrequency</td>
<td>Tuner chip crystal oscillator frequency, in Hz</td>
</tr>
<tr>
<td>SamplingMode</td>
<td>Direct sampling mode. Values:</td>
</tr>
<tr>
<td></td>
<td>• Disabled</td>
</tr>
<tr>
<td></td>
<td>• In-phase</td>
</tr>
<tr>
<td></td>
<td>• Quadrature</td>
</tr>
<tr>
<td>OffsetTuning</td>
<td>Offset tuning mode. Values:</td>
</tr>
<tr>
<td></td>
<td>• Disabled</td>
</tr>
<tr>
<td></td>
<td>• Enabled</td>
</tr>
<tr>
<td>CenterFrequency</td>
<td>Actual current value of the center frequency, in Hz</td>
</tr>
<tr>
<td>SampleRate</td>
<td>Actual current sample rate as samples per second</td>
</tr>
<tr>
<td>TunerGain</td>
<td>Actual current AGC value, in dB. This property is displayed only when EnableTunerAGC is set to false.</td>
</tr>
<tr>
<td>FrequencyCorrection</td>
<td>Actual current frequency correction value, in ppm</td>
</tr>
</tbody>
</table>
Purpose

Locked status (logical)

Syntax

L = isLocked(H)

Description

L = isLocked(H) returns the locked status, L, of the SDRRTLReceiver System object, H.

The isLocked method returns a logical value that indicates whether input attributes and nontunable properties for the object are locked. The object performs an internal initialization the first time the step method is executed. This initialization locks nontunable properties and input specifications, such as dimensions, complexity, and data type of the input data. After locking, the isLocked method returns a true value.

Input Arguments

H

Instance of SDRRTLReceiver
Purpose
Allow property value and input characteristics changes

Syntax
release(H)

Description
release(H) releases system resources (such as memory, file handles, or hardware connections), allowing you to change System object properties and input characteristics. This method releases the RTL-SDR radio for use by other clients. It stops streaming data and closes the connection to the radio.

Note  You can use the `release` method on a System object in code generated from MATLAB, but once you release its resources, you cannot use that System object again.

Input
Arguments
H
Instance of `SDRRTLReceiver`
### Purpose
Reset the internal states of the RTL-SDR radio System object

### Syntax
```
reset(H)
```

### Description
`reset(H)` resets the System object’s internal states to their initial values.

### Input Arguments
- **H**
  Instance of `SDRRTLReceiver`
Purpose
Receive data from RTL-SDR radio

Syntax
Y = step(H)
[Y,LEN] = step(H)
[Y,LEN,LOST] = step(H)
[Y,LEN,LOST,LATE] = step(H)

Description
Y = step(H) receives signal and control data from an RTL-SDR radio as represented by an SDRRTLReceiver System object, H. Output signal, Y, is a column vector of complex double precision, single precision, or 16-bit integer values.

[Y,LEN] = step(H) receives signal and control data from the RTL-SDR radio. Output signal, Y, is a column vector of complex 16-bit signed integer values in the [-128 127] range. The software converts the samples to double or single, if requested, as Y/128, where 128 = 2^7, to scale the output to [-1 1]. LEN is the number of valid samples in Y, and can be zero if no valid samples were returned from the radio.

[Y,LEN,LOST] = step(H) outputs the number of lost samples. If LOST = 0, then no samples were lost.

[Y,LEN,LOST,LATE] = step(H) outputs the latency, LATE. Output, LATE, is an integer valued scalar representing the latency in number of frames.

Note
The object performs an initialization the first time the step method is executed. This initialization locks nontunable properties and input specifications, such as dimensions, complexity, and data type of the input data. If you change a nontunable property or an input specification, the System object issues an error. To change nontunable properties or inputs, you must first call the release method to unlock the object.
Examples

Receive Streaming Data

```matlab
h = comm.SDRRTLReceiver('0', 'CenterFrequency', 102.5e6, 'SampleRate', 250000,...
'SamplesPerFrame', 2048, 'EnableTunerAGC', true, 'OutputDataType', 'double')
radioInfo = info(h)
radioInfo.CenterFrequency
radioInfo.SampleRate
for p=1:1000, x = step(h); end
release(h)

h =

System: comm.SDRRTLReceiver

Properties:
    RadioAddress: '0'
    CenterFrequency: 102500000
    EnableTunerAGC: true
    SampleRate: 250000
    OutputDataType: 'double'
    SamplesPerFrame: 2048
    FrequencyCorrection: 0

radioInfo =

The following values show radio settings, not the property values of RTL-SDR receiver System object. For more information, type 'help comm.SDRRTLReceiver'.

    RadioName: 'ezcap USB 2.0 DVB-T/DAB/FM dongle'
    RadioAddress: '0'
    TunerName: 'R820T'
    Manufacturer: 'Realtek'
    Product: 'RTL2838UHIDIR'
    GainValues: [1x29 double]
    RTLCrystalFrequency: 28800000
```

5-16
Receive Streaming Data on Two Frequencies

```matlab
h = comm.SDRRTLReceiver('0', 'CenterFrequency', 102.5e6, 'SampleRate', 350000,...
    'SamplesPerFrame', 2048,'EnableTunerAGC', true)
for p=1:1000, x = step(h); end
h.CenterFrequency = 103e6;
for p=1:1000, x = step(h); end
release(h)
```

h =

```
System: comm.SDRRTLReceiver
```

Properties:

```
RadioAddress: '0'
CenterFrequency: 102500000
EnableTunerAGC: true
SampleRate: 350000
OutputDataType: 'int16'
SamplesPerFrame: 2048
FrequencyCorrection: 0
```
Check Frame Loss and Latency

```matlab
h = comm.SDRRTLReceiver('0', 'CenterFrequency', 102.5e6, 'SampleRate', 250000,...
   'SamplesPerFrame', 2048,'EnableTunerAGC', true)
for p=1:100, [x, ~, lost(p), latency(p)] = step(h); end
if any(lost > 0), error('Lost frames'), end
if any(latency > 3), error('Latency is greater than 3 frames'), end
release(h)
```

```
h =

System: comm.SDRRTLReceiver

Properties:
    RadioAddress: '0'
    CenterFrequency: 102500000
    EnableTunerAGC: true
    SampleRate: 250000
    OutputDataType: 'int16'
    SamplesPerFrame: 2048
    FrequencyCorrection: 0
```
Functions — Alphabetical List
**sdrexamples**

<table>
<thead>
<tr>
<th><strong>Purpose</strong></th>
<th>Open index to featured examples for all SDR support packages</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Syntax</strong></td>
<td>sdrexamples</td>
</tr>
<tr>
<td><strong>Description</strong></td>
<td>sdrexamples opens an index of all MATLAB and Simulink examples for SDR support packages.</td>
</tr>
</tbody>
</table>
**Purpose**

Report information about attached radios

**Syntax**

HARDWAREINFO = sdrinfo()
HARDWAREINFO = sdrinfo(DEVADDR)

**Description**

HARDWAREINFO = sdrinfo() returns a cell array of structures with information about all known radios attached to the host. If the function does not find any radios, the array is empty.

HARDWAREINFO = sdrinfo(DEVADDR) returns a structure with information about a radio attached to the host at the specified address. For devices attached via a Network Interface Card (NIC), DEVADDR is a dotted-quad character array such as 192.168.0.2. For devices attached via a USB port, DEVADDR is a char containing a scalar non-negative integer such as '0').

- The structure of information is specific to the type of radio attached.
- If you do not specify a device address, the function attempts to find all attached radios.
- If the function finds an incompatible radio at the specified address, it returns a warning.
- If you are using an FPGA programming file generated by the Xilinx® FPGA-based Radio workflow in the HDL Coder™ Workflow Advisor, the IP address matches the one specified in Step 4.1 Set SDR Options in "Board Addresses" (Ethernet connections only).

**Input Arguments**

DEVADDR - Specifies the particular Ethernet or USB-connected radio that you want to get information about

string

DEVADDR is either a dotted-quad character array such as 192.168.0.2, or a char containing a scalar non-negative integer such as '0', depending on the type of radio connection.

Example: '192.168.0.2'
## HARDWAREINFO - Contains radio information structure

For Ethernet-connected radios, HARDWAREINFO contains the following radio information:

<table>
<thead>
<tr>
<th>Field</th>
<th>Contains</th>
</tr>
</thead>
<tbody>
<tr>
<td>Version</td>
<td>Support package version</td>
</tr>
<tr>
<td>Motherboard</td>
<td>FPGA development board name</td>
</tr>
<tr>
<td>RFBoard</td>
<td>RF daughter board name</td>
</tr>
<tr>
<td>HasDDC</td>
<td>Indicates if bitstream (fixed or targeted) has a DDC (digital down converter)</td>
</tr>
<tr>
<td>HasRxUserDUT</td>
<td>Indicates if bitstream contains user Receiver DUT (targeting only)</td>
</tr>
<tr>
<td>HasRxPath</td>
<td>Indicates if bitstream contains the Receiver path (targeting only)</td>
</tr>
<tr>
<td>HasDUC</td>
<td>Indicates if bitstream has a DUC (digital up converter)</td>
</tr>
<tr>
<td>HasTxUserDUT</td>
<td>Indicates if bitstream contains user Transmitter DUT (targeting only)</td>
</tr>
<tr>
<td>HasTxPath</td>
<td>Indicates if bitstream contains the Transmitter path</td>
</tr>
<tr>
<td>DUTTargetFrequency</td>
<td>DUT target frequency, in Hz (targeting only)</td>
</tr>
<tr>
<td>BuildTimestamp</td>
<td>Bitstream build time stamp, for fixed or targeted</td>
</tr>
</tbody>
</table>

For USB-connected radios, HARDWAREINFO contains the following radio information:
<table>
<thead>
<tr>
<th>Field</th>
<th>Contains</th>
</tr>
</thead>
<tbody>
<tr>
<td>RadioName</td>
<td>Device name</td>
</tr>
<tr>
<td>RadioAddress</td>
<td>USB address of the RTL-SDR radio</td>
</tr>
<tr>
<td>RadioIsOpen</td>
<td>Flag for open devices</td>
</tr>
<tr>
<td>TunerName</td>
<td>Tuner chip name</td>
</tr>
<tr>
<td>Manufacturer</td>
<td>Manufacturer name</td>
</tr>
<tr>
<td>Product</td>
<td>Product name</td>
</tr>
<tr>
<td>GainValues</td>
<td>Tuner chip supported gain values, in dB. These values are displayed only when the System object property EnableTunerAGC is set to false or the block parameter Tuner Gain source is set to Dialog or Property.</td>
</tr>
<tr>
<td>RTLCrystalFrequency</td>
<td>RTL chip crystal oscillator frequency, in Hz</td>
</tr>
<tr>
<td>TunerCrystalFrequency</td>
<td>Tuner chip crystal oscillator frequency, in Hz</td>
</tr>
<tr>
<td>SamplingMode</td>
<td>Sampling mode. Values:</td>
</tr>
<tr>
<td></td>
<td>• disabled</td>
</tr>
<tr>
<td></td>
<td>• I-ADC</td>
</tr>
<tr>
<td></td>
<td>• Q-ADC</td>
</tr>
<tr>
<td>OffsetTuning</td>
<td>Offset tuning mode. Values:</td>
</tr>
<tr>
<td></td>
<td>• disabled</td>
</tr>
<tr>
<td></td>
<td>• enabled</td>
</tr>
</tbody>
</table>
**sdrinfo**

**Note** With an RTL-SDR radio, `sdrinfo` returns all valid radio fields that are applicable, as long as the System object or block is not open (unlocked). If the block or System object is currently running, `sdrinfo` returns only the top three fields: `RadioName`, `RadioAddress`, and `RadioIsOpen`.

**Examples**

Get radio information for a Xilinx FPGA-based radio.

```matlab
hwinfo = sdrinfo('192.168.2.2')
```

```
hwinfo =

    Version: 'SDRf R2013b.3.81'
    Motherboard: [1x39 char]
    RFBoard: [1x126 char]
    HasDDC: 'Yes'
    HasRxUserDUT: 'No'
    HasRxPath: 'Yes'
    HasDUC: 'Yes'
    HasTxUserDUT: 'No'
    HasTxPath: 'Yes'
    DUTTargetFrequency: '125000000 Hz'
    BuildTimestamp: '2013-08-08 22:35:50'
```

Get radio information for an RTL-SDR radio.

```matlab
hwinfo = sdrinfo('0');
```

```
hwinfo =

    RadioName: 'ezcap USB 2.0 DVB-T/DAB/FM dongle'
    RadioAddress: '0'
    RadioIsOpen: 0
    TunerName: 'R820T'
```
Manufacturer: 'Realtek'
Product: 'RTL2838UHIDIR'
GainValues: [29x1 double]
RTLCrystalFrequency: 28800000
TunerCrystalFrequency: 28800000
SamplingMode: 'Quadrature'
OffsetTuning: 'Disabled'

Call sdrinfo using a USB port address with no radio plugged into the USB port.

>> sdrinfo('0')

ans =

{}

The function returns an empty cell array.
<table>
<thead>
<tr>
<th><strong>Purpose</strong></th>
<th>Report root folder for RTL-SDR radio support package installation</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Syntax</strong></td>
<td><code>sdrrroot</code></td>
</tr>
<tr>
<td><strong>Description</strong></td>
<td><code>sdrrroot</code> returns the top-level directory of the RTL-SDR radio installation.</td>
</tr>
<tr>
<td><strong>Examples</strong></td>
<td>Locate RTL-SDR Radio Installation Folder</td>
</tr>
<tr>
<td></td>
<td><code>sdrrroot</code></td>
</tr>
<tr>
<td></td>
<td><code>ans = C:\MATLAB\SupportPackages\R2014a\sdrr</code></td>
</tr>
</tbody>
</table>
**Purpose**
Set up MATLAB to work with radios and third-party tools

**Syntax**
```
sdrsetup
sdrsetup(Name,Value)
```

**Description**
`sdrsetup` modifies system PATH and LD_LIBRARY_PATH variables in the current MATLAB session for using features in any installed SDR support packages.

`sdrsetup(Name,Value)` modifies system PATH and LD_LIBRARY_PATH variables in the current MATLAB session for using features in any installed SDR support packages and to allow running third-party tools. Specify this path for Linux® installations that do not have the libusb path already in the LD_LIBRARY_PATH environment variable.

All changes to system environment variables using `sdrsetup` are performed for the current MATLAB session only. You must run this command at the start of every MATLAB session in which you intend to use any of the installed SDR support packages. You can automate this step for future sessions by adding `sdrsetup` to your `startup.m` file or by adding a shortcut to the MATLAB shortcut bar.

For instructions on creating or modifying a startup script, see the MATLAB documentation for “Specifying Startup Options in MATLAB Startup File”.

 Optionally, you can add any PATH and LD_LIBRARY_PATH modifications to the system environment using your OS.

**RTL-SDR Radio and sdrsetup**
The function `sdrsetup` is automatically run when you first install the RTL-SDR Radio support package, as well as each time you call RTL-SDR Radio support package functions. You do not normally need to call `sdrsetup`; this reference page is for informational purposes only.
**Input Arguments**

**Name-Value Pair Arguments**

Specify optional comma-separated pairs of Name, Value arguments. Name is the argument name and Value is the corresponding value. Name must appear inside single quotes (' '). You can specify several name and value pair arguments in any order as Name1, Value1, ..., NameN, ValueN.

Example: 'ToolName', 'Xilinx iMPACT'

**‘ToolName’ - Name of third-party tool**

Xilinx ISE | Xilinx iMPACT

Name of the third-party tool. If you have multiple tools in a specific directory then you do not need to setup each one separately. Valid values include:

- Xilinx ISE
- Xilinx iMPACT

**‘ToolPath’ - Full path to the third-party tool executable**

string

Full path to the third-party tool executable. Enter as a string, for example: '/hdltools/Xilinx/14.6/ISE_DS/ISE/bin/lin64/impact'.

**‘LibUSBPath’ - Full path to where libusb is located (Linux)**

string

Specify this path for Linux installations that do not have the libusb path already in the LD_LIBRARY_PATH environment variable. Enter as a string, for example: '/hdltools/lib/libusb/64bit'.

**Examples**

Set up environment variables to use the Xilinx installation in the specified folder.

```matlab
sdrsetup('ToolName', 'Xilinx ISE', 'ToolPath', 'C:\Xilinx\14.6\ISE_DS\ISE\bin\nt64\ise.exe');
```
Set up environment variables to use the Xilinx installation in the specified folder and the libusb file in the specified path (for a Linux installation).

```
sdrsetup('ToolName', 'Xilinx iMPACT', 'ToolPath',
    '/hdltools/Xilinx/14.6/ISE_DS/ISE/bin/lin64/impact',
    'LibUSBPath', '/hdltools/lib/libusb/64bit');
```
**Purpose**

Start Support Package Installer and install support for third-party hardware or software

**Syntax**

`supportPackageInstaller`

**Description**

The `supportPackageInstaller` function opens *Support Package Installer*. Support Package Installer can install *support packages*, which add support for specific third-party hardware or software to specific MathWorks products. To see a list of available support packages, run Support Package Installer and advance to the second screen.

You can also start Support Package Installer in one of the following ways:

- On the MATLAB toolstrip, click **Add-Ons > Get Hardware Support Packages**.

- Double-click a support package installation file (*.mlpkginstall).
Support Package General Topics

- “Install This Support Package on Other Computers” on page 7-2
- “Open Support Package Examples” on page 7-4
- “Support Packages and Support Package Installer” on page 7-6
Install This Support Package on Other Computers

You can download a support package to one computer, and then install it on other computers. You can use this approach to:

- Save time when installing support packages on multiple computers.
- Install support packages on computers that are not connected to the Internet.

Before starting, select a computer to use for downloading. This computer must have the same base product license and platform as the computers upon which you are installing the support package. For example, suppose you want to install a Simulink support package on a group of computers that are running 64-bit Windows. To do so, you must first download the support package using a computer that has a Simulink license and is running 64-bit Windows.

Download the support package to one computer:

1 In the MATLAB Command Window, enter `supportPackageInstaller`.

2 In Support Package Installer, on the Select an action screen, choose Install from Internet or Download from Internet. Click Next.

3 On the following screen, select only one support package.

   Notice the path of the Download folder. For example, `C:\MATLAB\SupportPackages\R2013b\downloads`.

4 Using the file manager on your computer, open the downloads folder and observe its contents.

5 Using Support Package Installer, complete the installation or download process.

   This process creates a folder within the Download folder. In some cases, if the support package requires another support package, this process creates an additional folder.

Prepare and share the support package files:
1 In the file manager, check how many folders were created during the installation or download process.

2 If more than one folder was created, combine the contents of the folders into the folder named after the support package.

For example, C:\MATLAB\SupportPackages\R2013b\downloads\support_package_name.

3 Make that folder available to other computers by sharing it on the network, or copying it to portable media, such as a USB flash drive.

Note Some support packages require that you install third-party software separately before completing the support package installation process. In that case, also make the third-party software available for installation on the other computers.

Install the support package on the other computers:

1 Run Support Package Installer on the other computer or computers.

2 On the Install or update support package screen, select the Folder option.

3 Use Browse to specify the location of the support package folder on the network or portable media.

4 Complete the instructions provided by Support Package Installer.
Open Support Package Examples

Open Example Index

- “Using the Function sdrexamples” on page 7-4
- “Using Support Package Installer” on page 7-4

Using the Function sdrexamples

In the MATLAB command window, type the following:

    sdrexamples

The command opens an index of all installed SDR support package examples.

Using Support Package Installer

Support Package Installer (supportPackageInstaller) automatically displays the support package examples when you complete the process of installing and setting up a support package.

On the last screen in Support Package Installer, leave Show support package examples enabled and click Finish.
Install/update complete
Communications System Toolbox Support Package for Xilinx FPGA-Based Radio has been successfully installed.

☑ Show support package examples
Support Packages and Support Package Installer

What Is a Support Package?

A support package is an add-on that enables you to use a MathWorks product with specific third-party hardware and software.

Support packages can include:

- Simulink block libraries
- MATLAB functions, classes, and methods
- Firmware updates for the third-party hardware
- Automatic installation of third-party software
- Examples and tutorials

A support package file has a * .zip extension. This type of file contains MATLAB files, MEX files, and other supporting files required to install the support package. Use Support Package Installer to install these support package files.

A support package installation file has a *.mlpkginstall extension. You can double click this type of file to start Support Package Installer, which preselects a specific support package for installation. You can download these files from MATLAB Central File Exchange and use them to share support packages with others.

What Is Support Package Installer?

Support Package Installer is a wizard that guides you through the process of installing support packages.

Support packages may be available for your release of installed MathWorks products. If so, you can use Support Package Installer to:

- Display available support packages.
- Install or update a support package.
• Update the firmware on specific third-party hardware.

If third-party software is included, Support Package Installer displays a list of the software and licenses for you to review before continuing.

You can start Support Package Installer in one of the following ways:

• On the MATLAB toolstrip, click Add-Ons > Get Hardware Support Packages.

• In the MATLAB Command Window, enter supportPackageInstaller.

• Double-click a support package installation file (*.mlpkginstall).