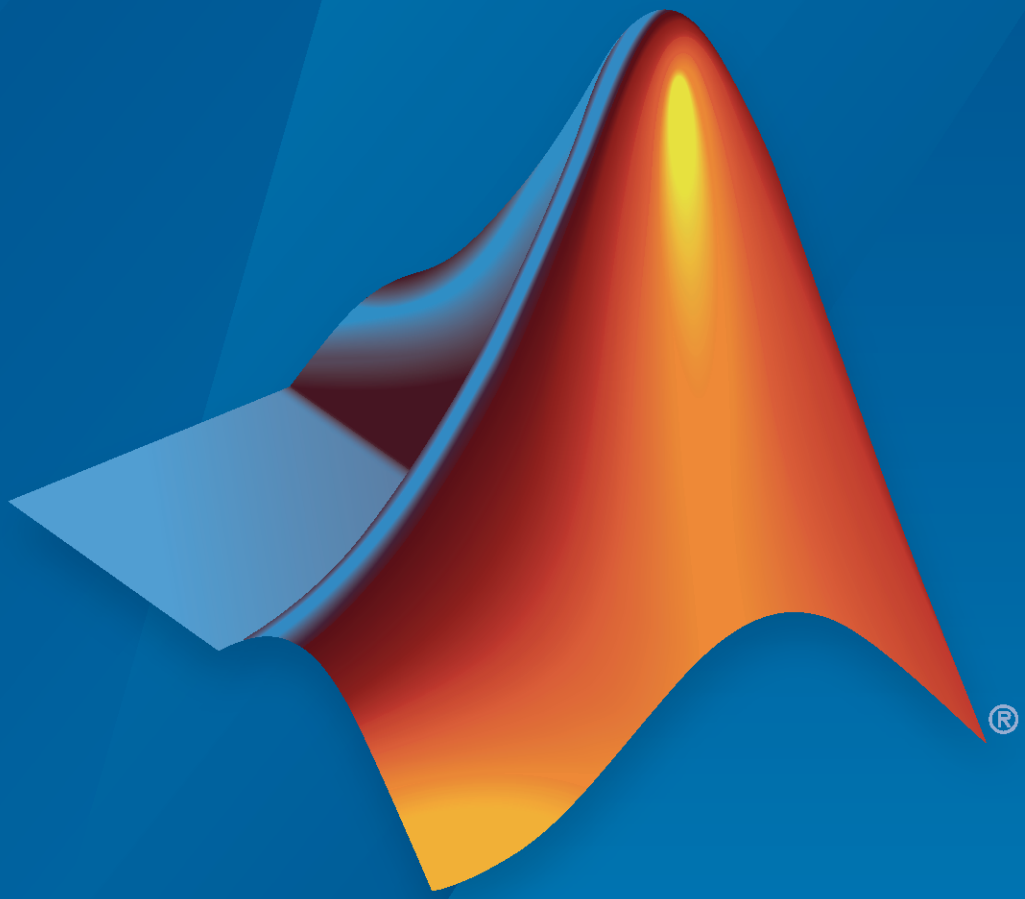
Wireless Testbench™

Reference



MATLAB®

R2022a

MathWorks®

How to Contact MathWorks

Latest news: `www.mathworks.com`

Sales and services: `www.mathworks.com/sales_and_services`

User community: `www.mathworks.com/matlabcentral`

Technical support: `www.mathworks.com/support/contact_us`

Phone: 508-647-7000

The MathWorks, Inc.
1 Apple Hill Drive
Natick, MA 01760-2098

Wireless Testbench™ Reference

Revision History

March 2022	Online only	New for Version 1.0 (Release 2022a)

Contents

Objects

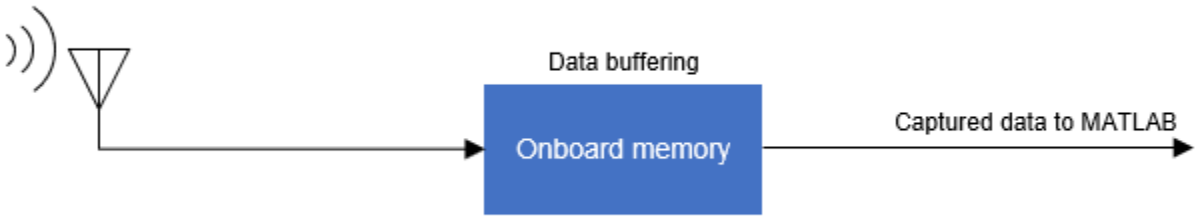
basebandReceiver

Configure SDR as baseband receiver

Description

Use the `basebandReceiver` object to configure the specified software-defined radio (SDR) as a baseband receiver to capture raw IQ data from the air.

This diagram shows a conceptual overview of capturing radio signals in Wireless Testbench™ using a radio that you configure with this object. The onboard data buffering ensures contiguous data capture.



Creation

Syntax

```
bbrx = basebandReceiver(radio)
bbrx = basebandReceiver(radio,Name=Value)
```

Description

`bbrx = basebandReceiver(radio)` creates a baseband receiver configuration object for the specified radio `radio`.

Note The object requires exclusive access to radio hardware resources. Before creating this object, clear any existing Wireless Testbench object associated with the specified radio from the workspace.

`bbrx = basebandReceiver(radio,Name=Value)` sets properties on page 1-3 using one or more name-value arguments. For example, `CaptureDataType="double"` sets the data type of the returned captured data to `double`.

Input Arguments

`radio` — Radio setup configuration

string scalar

Radio setup configuration, specified as a string scalar. To create a radio setup configuration, set up your radio and save your radio setup configuration using the Radio Setup wizard. Call the `radioConfigurations` function to list all saved radio setup configurations. For more information, see “Connect and Set Up NI USRP Radios”.

Example: `"MyRadio"` indicates that you saved a radio setup configuration under the name MyRadio in the Radio Setup wizard.

Properties

`CenterFrequency` — Radio center frequency in Hz

`2.4e9` (default) | positive numeric scalar | numeric array

Radio center frequency in Hz, specified as one of these options.

- Positive numeric scalar — The object applies this value by scalar expansion to each antenna in the `Antennas` property.
- Numeric array — The object applies the *i*th array element value to the *i*th antenna in the `Antennas` property.

The valid center frequency range depends on the radio device.

Radio Device	Center Frequency
USRP™ N310	1 MHz to 6 GHz
USRP N320	1 MHz to 6 GHz
USRP N321	1 MHz to 6 GHz

Note When setting this property for multiple antennas on the USRP N310 radio, consider these hardware characteristics.

- The antenna ports on the RF0 and RF1 radio channels use the same center frequency. Therefore, set identical center frequency values for the antennas specified as `"RF0:RX2"` and `"RF1:RX2"`.
- The antenna ports on the RF2 and RF3 radio channels use the same center frequency. Therefore, set identical center frequency values for the antennas specified as `"RF2:RX2"` and `"RF3:RX2"`.

Data Types: `double`

`Antennas` — Capture radio antennas

`"RF0:RX2"` (default) | string scalar | string array

Capture radio antennas, specified as one of these options.

- String scalar — Specify use of single antenna.
- String array — Specify use of multiple antennas.

Use this table to identify a supported radio antenna port on the radio device and the corresponding string constant that you can specify for this property.

Radio Device	Supported Antenna Port	Valid String Scalar
USRP N310	RF0 channel: RX2 port	`"RF0:RX2"` (default)
	RF1 channel: RX2 port	`"RF1:RX2"`
	RF2 channel: RX2 port	`"RF2:RX2"`
	RF3 channel: RX2 port	`"RF3:RX2"`
USRP N320	RF0 channel: RX2 port	`"RF0:RX2"` (default)
	RF1 channel: RX2 port	`"RF1:RX2"`
USRP N321	RF0 channel: RX2 port	`"RF0:RX2"` (default)
	RF1 channel: RX2 port	`"RF1:RX2"`

Note When you update this property, the execution time of the next object function call increases by a few seconds.

Example: `Antennas=["RF0:RX2","RF1:RX2"]` specifies two capture antennas.

Data Types: `string`

`RadioGain` — Capture radio gain in dB

`10` (default) | positive numeric scalar | numeric array

Capture radio gain in dB, specified as one of these options.

- Positive numeric scalar — The object applies this value by scalar expansion to each antenna that you specify in the `Antennas` property.
- Numeric array — The object applies the *i*th array element value to the *i*th antenna in the `Antennas` property.

The valid gain range depends on the radio device.

Radio Device	Capture Radio Gain
USRP N310	0 dB to 75 dB
USRP N320	0 dB to 60 dB
USRP N321	0 dB to 60 dB

Data Types: `double`

`SampleRate` — Baseband sample rate in Hz

highest device sample rate (default) | positive numeric scalar

Baseband sample rate in Hz, specified as a positive numeric scalar. The object automatically selects the master clock rate available for the radio based on the specified sample rate. Therefore, the valid range for the sample rate depends on the master clock rate available on the radio. The sample rate must be less than or equal to *MCR*/2 or equal to *MCR*, where *MCR* is the master clock rate that the object selects. For more information, see “Baseband Sample Rate in NI USRP Radios”.

Radio Device	Sample Rate	Master Clock Rate
USRP N310	120,471 Hz to 153.6 MHz `153.6e6` (default)	122.88 MHz 125.00 MHz 153.60 MHz
USRP N320	196,078 Hz to 250 MHz `250e6` (default)	200.00 MHz 245.76 MHz 250.00 MHz
USRP N321	196,078 Hz to 250 MHz `250e6` (default)	200.00 MHz 245.76 MHz 250.00 MHz

Note When you update this property, the execution time of the next object function call increases by a few seconds.

Data Types: `double`

`CaptureDataType` — Data type of captured data
`"int16"` (default) | `"double"` | `"single"`

Data type of the captured data, specified as `"int16"`, `"double"`, or `"single"`. Use this property to set the data type of the captured data that the `capture` object function returns.

Note When you update this property, the execution time of the next object function call increases by a few seconds.

Data Types: `string`

`DroppedSamplesAction` — Behavior upon dropped samples
`"error"` (default) | `"warning"` | `"none"`

Behavior of the `capture` object function upon dropped samples, specified as one of these values.

- `"error"` — The object function stops with an error message.
- `"warning"` — The object function displays a warning message.
- `"none"` — The object function ignores dropped samples.

Data Types: `string`

Object Functions

capture Capture IQ data using baseband receiver or transceiver

Examples

Configure Baseband Receiver and Capture Data

Create a baseband receiver object, specifying a radio setup configuration previously saved in the Radio Setup wizard.

```
bbrx = basebandReceiver("MyRadio")
```

```
bbrx =
  basebandReceiver with properties:

               RadioGain: 10
         CenterFrequency: 2.4000e+09
              SampleRate: 250000000
                Antennas: "RF0:RX2"
    DroppedSamplesAction: "error"
         CaptureDataType: "int16"
```

Set the baseband sample rate and center frequency.

```
bbrx.SampleRate = 122.88e6;
bbrx.CenterFrequency = 2.2e9;
```

Capture 3 ms of IQ data with the radio associated with the baseband receiver object using the default antenna.

```
[data,~] = capture(bbrx,milliseconds(3));
```

See Also

Functions
`radioConfigurations`

Objects
`basebandTransceiver` | `basebandTransmitter`

Topics
“Supported Radio Devices”

Introduced in R2022a

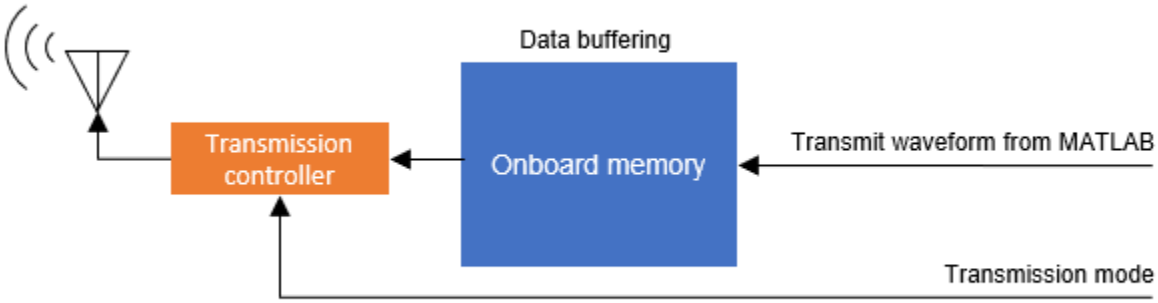
basebandTransmitter

Configure SDR as baseband transmitter

Description

Use the `basebandTransmitter` object to configure the specified software-defined radio (SDR) as a baseband transmitter to transmit IQ waveforms to the air.

This diagram shows the conceptual overview of transmitting radio signals in Wireless Testbench using a radio that you configure with this object. The onboard data buffering ensures contiguous data transmit. The transmission controller enables you to specify continuous or single-shot transmissions.



Creation

Syntax

```
bbtx = basebandTransmitter(radio)
bbtx = basebandTransmitter(radio,Name=Value)
```

Description

`bbtx = basebandTransmitter(radio)` creates a baseband transmitter object for the specified radio `radio`.

Note The object requires exclusive access to radio hardware resources. Before creating this object, clear any existing Wireless Testbench object associated with the specified radio from the workspace.

`bbtx = basebandTransmitter(radio,Name=Value)` sets properties on page 1-8 using one or more name-value arguments. For example, `CenterFrequency=2.2e9` sets the center frequency to 2.2 GHz.

Input Arguments

`radio` — Radio setup configuration

string scalar

Radio setup configuration, specified as a string scalar. To create a radio setup configuration, set up your radio and save your radio setup configuration using the Radio Setup wizard. Call the `radioConfigurations` function to list all saved radio setup configurations. For more information, see “Connect and Set Up NI USRP Radios”.

Example: `"MyRadio"` indicates that you saved a radio setup configuration under the name MyRadio in the Radio Setup wizard.

Properties

`CenterFrequency` — Radio center frequency in Hz

`2.4e9` (default) | positive numeric scalar | numeric array

Radio center frequency in Hz, specified as one of these options.

- Positive numeric scalar — The object applies this value by scalar expansion to each antenna in the `Antennas` property.
- Numeric array — The object applies the *i*th array element value to the *i*th antenna in the `Antennas` property.

The valid center frequency range depends on the radio device.

Radio Device	Center Frequency
USRP N310	1 MHz to 6 GHz
USRP N320	1 MHz to 6 GHz
USRP N321	1 MHz to 6 GHz

Note When setting this property for multiple antennas on the USRP N310 radio, consider these hardware characteristics.

- The antenna ports on the RF0 and RF1 radio channels use the same center frequency. Therefore, set identical center frequency values for the antennas specified as `"RF0:TX/RX"` and `"RF1:TX/RX"`.
- The antenna ports on the RF2 and RF3 radio channels use the same center frequency. Therefore, set identical center frequency values for the antennas specified as `"RF2:TX/RX"` and `"RF3:TX/RX"`.

Data Types: `double`

`Antennas` — Transmit radio antennas

`"RF0:TX/RX"` (default) | string scalar | string array

Transmit radio antennas, specified as one of these options.

- String scalar — Specify use of single antenna.
- String array — Specify use of multiple antennas.

Use this table to identify a supported radio antenna port on the radio device and the corresponding string constant that you can specify for this property.

Radio Device	Supported Antenna Port	Valid String Scalar
USRP N310	RF0 channel: TX/RX port	`"RF0:TX/RX"` (default)
	RF1 channel: TX/RX port	`"RF1:TX/RX"`
	RF2 channel: TX/RX port	`"RF2:TX/RX"`
	RF3 channel: TX/RX port	`"RF3:TX/RX"`
USRP N320	RF0 channel: TX/RX port	`"RF0:TX/RX"` (default)
	RF1 channel: TX/RX port	`"RF1:TX/RX"`
USRP N321	RF0 channel: TX/RX port	`"RF0:TX/RX"` (default)
	RF1 channel: TX/RX port	`"RF1:TX/RX"`

Note When you update this property, the execution time of the next object function call increases by a few seconds.

Data Types: `string`

`RadioGain` — Transmit radio gain in dB

`10` (default) | positive numeric scalar | numeric array

Transmit radio gain in dB, specified as one of these options.

- Positive numeric scalar — The object applies this value by scalar expansion to each antenna that you specify in the `Antennas` property.
- Numeric array — The object applies the *i*th array element value to the *i*th antenna in the `Antennas` property.

The valid gain range depends on the radio device.

Radio Device	Transmit Radio Gain
USRP N310	0 dB to 65 dB
USRP N320	0 dB to 60 dB
USRP N321	0 dB to 60 dB

Data Types: `double`

`SampleRate` — Baseband sample rate in Hz

highest device sample rate (default) | positive numeric scalar

Baseband sample rate in Hz, specified as positive numeric scalar. The object automatically selects the master clock rate available for the radio based on the specified sample rate. Therefore, the valid range for the sample rate depends on the master clock rate available on the radio. The sample rate must be less than or equal to *MCR*/2 or equal to *MCR*, where *MCR* is the master clock rate that the object selects. For more information, see “Baseband Sample Rate in NI USRP Radios”.

Radio Device	Sample Rate	Master Clock Rate
USRP N310	120,471 Hz to 153.6 MHz `153.6e6` (default)	122.88 MHz 125.00 MHz 153.60 MHz
USRP N320	196,078 Hz to 250 MHz `250e6` (default)	200.00 MHz 245.76 MHz 250.00 MHz
USRP N321	196,078 Hz to 250 MHz `250e6` (default)	200.00 MHz 245.76 MHz 250.00 MHz

Note To update this property, you must stop any ongoing transmission by calling the `stopTransmission` function on the object. When you update this property, the execution time of the next object function call increases by a few seconds.

Data Types: `double`

Object Functions

transmit	Transmit waveform using baseband transmitter or transceiver
stopTransmission	Stop transmission from baseband transmitter or transceiver

Examples

Configure Baseband Transmitter and Transmit Waveform

Create a baseband transmitter object, specifying a radio setup configuration previously saved in the Radio Setup wizard.

```
bbtx = basebandTransmitter("MyRadio")
```

```
bbtx =
  basebandTransmitter with properties:

          RadioGain: 10
    CenterFrequency: 2.4000e+09
         SampleRate: 250000000
           Antennas: "RF0:TX/RX"
```

Set the baseband sample rate and center frequency.

```
bbtx.SampleRate = 122.88e6;
bbtx.CenterFrequency = 2.2e9;
```

Generate random transmit waveform.

```
txWaveform = complex(randn(1000,1),randn(1000,1));
```

Transmit the generated waveform continuously with the radio associated with the baseband transmitter object using the default antenna.

```
transmit(bbtx,txWaveform,"continuous");
```

Stop the continuous transmission after 5 seconds.

```
pause(5);
stopTransmission(bbtx);
```

See Also

Functions
`radioConfigurations`

Objects
`basebandTransceiver` | `basebandReceiver`

Introduced in R2022a

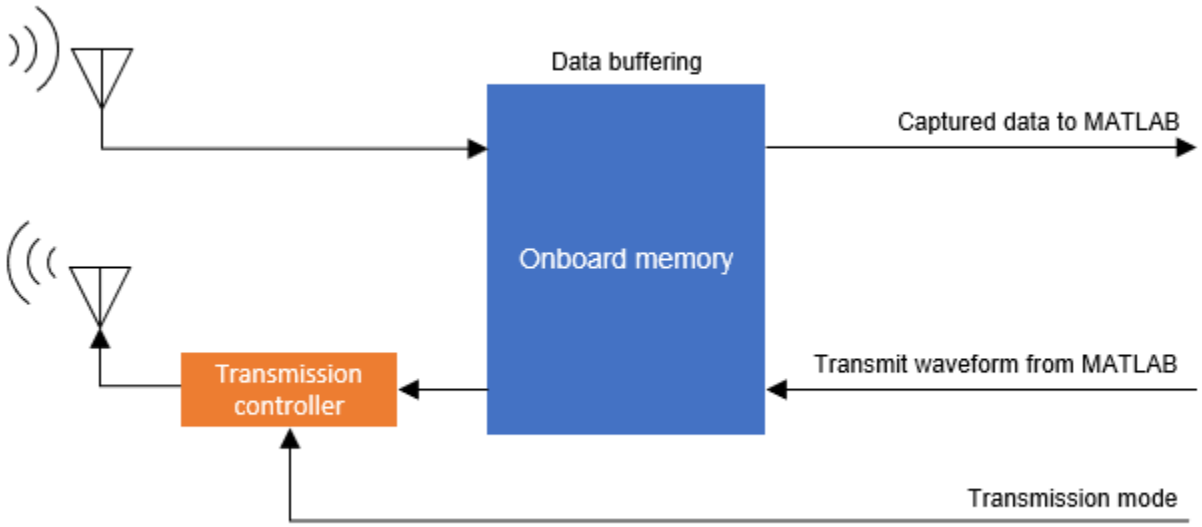
basebandTransceiver

Configure SDR as baseband transceiver

Description

Use the `basebandTransceiver` object to configure the specified software-defined radio (SDR) as a baseband transceiver to simultaneously transmit and capture IQ waveforms over the air.

This diagram shows a conceptual overview of capturing and transmitting radio signals in Wireless Testbench using a radio that you configure with this object. The onboard data buffering ensures contiguous data capture and transmit. The transmission controller enables you to specify continuous or single-shot transmissions.



Creation

Syntax

```
bbtrx = basebandTransceiver(radio)
bbtrx = basebandTransceiver(radio,Name=Value)
```

Description

`bbtrx = basebandTransceiver(radio)` creates a baseband transceiver object for the specified radio `radio`.

Note The object requires exclusive access to radio hardware resources. Before creating this object, clear any existing Wireless Testbench object associated with the specified radio from the workspace.

`bbtrx = basebandTransceiver(radio,Name=Value)` sets properties on page 1-13 using one or more name-value arguments. For example, `CaptureDataType="double"` sets the data type of the returned captured data to `double`.

Input Arguments

`radio` — Radio setup configuration

string scalar

Radio setup configuration, specified as a string scalar. To create a radio setup configuration, set up your radio and save your radio setup configuration using the Radio Setup wizard. Call the `radioConfigurations` function to list all saved radio setup configurations. For more information, see “Connect and Set Up NI USRP Radios”.

Example: `"MyRadio"` indicates that you saved a radio setup configuration under the name MyRadio in the Radio Setup wizard.

Properties

`TransmitCenterFrequency` — Transmit center frequency in Hz

`2.4e9` (default) | positive numeric scalar

Transmit center frequency in Hz, specified as a positive numeric scalar. The valid center frequency range depends on the radio device.

Radio Device	Center Frequency
USRP N310	1 MHz to 6 GHz
USRP N320	1 MHz to 6 GHz
USRP N321	1 MHz to 6 GHz

`TransmitAntennas` — Transmit radio antenna

`"RF0:TX/RX"` (default) | string scalar

Transmit radio antenna, specified as a string scalar. Use this table to identify a supported radio antenna port on the radio device and the corresponding string constant that you can specify for this property.

Radio Device	Supported Antenna Port	Valid String Scalar
USRP N310	RF0 channel: TX/RX port	`"RF0:TX/RX"` (default)
	RF1 channel: TX/RX port	`"RF1:TX/RX"`
	RF2 channel: TX/RX port	`"RF2:TX/RX"`
	RF3 channel: TX/RX port	`"RF3:TX/RX"`
USRP N320	RF0 channel: TX/RX port	`"RF0:TX/RX"` (default)
	RF1 channel: TX/RX port	`"RF1:TX/RX"`
USRP N321	RF0 channel: TX/RX port	`"RF0:TX/RX"` (default)

Radio Device	Supported Antenna Port	Valid String Scalar
	RF1 channel: TX/RX port	`"RF1:TX/RX"`

Note When you update this property, the execution time of the next object function call increases by a few seconds.

Data Types: `string`

`TransmitRadioGain` — Transmit radio gain in dB

`10` (default) | positive numeric scalar

Transmit radio gain in dB, specified as a positive numeric scalar. The valid gain range depends on the radio device.

Radio Device	Transmit Radio Gain
USRP N310	0 dB to 65 dB
USRP N320	0 dB to 60 dB
USRP N321	0 dB to 60 dB

Data Types: `double`

`CaptureCenterFrequency` — Capture center frequency in Hz

`2.4e9` (default) | positive numeric scalar

Capture center frequency in Hz, specified as a positive numeric scalar. The valid center frequency range depends on the radio device.

Radio Device	Center Frequency
USRP N310	1 MHz to 6 GHz
USRP N320	1 MHz to 6 GHz
USRP N321	1 MHz to 6 GHz

Data Types: `double`

`CaptureAntennas` — Capture radio antenna

`"RF0:RX2"` (default) | string scalar

Capture radio antenna, specified as a string scalar. Use this table to identify a supported radio antenna port on the radio device and the corresponding string constant that you can specify for this property.

Radio Device	Supported Antenna Port	Valid String Scalar
USRP N310	RF0 channel: RX2 port	`"RF0:RX2"` (default)
	RF1 channel: RX2 port	`"RF1:RX2"`
	RF2 channel: RX2 port	`"RF2:RX2"`
	RF3 channel: RX2 port	`"RF3:RX2"`

Radio Device	Supported Antenna Port	Valid String Scalar
USRP N320	RF0 channel: RX2 port	`"RF0:RX2"` (default)
	RF1 channel: RX2 port	`"RF1:RX2"`
USRP N321	RF0 channel: RX2 port	`"RF0:RX2"` (default)
	RF1 channel: RX2 port	`"RF1:RX2"`

Note When you update this property, the execution time of the next object function call increases by a few seconds.

Data Types: `string`

`CaptureRadioGain` — Capture radio gain in dB

`10` (default) | positive numeric scalar

Capture radio gain in dB, specified as a positive numeric scalar. The valid gain range depends on the radio device.

Radio Device	Capture Radio Gain
USRP N310	0 dB to 75 dB
USRP N320	0 dB to 60 dB
USRP N321	0 dB to 60 dB

Data Types: `double`

`CaptureDataType` — Data type of captured data

`"int16"` (default) | `"double"` | `"single"`

Data type of the captured data, specified as `"int16"`, `"double"`, or `"single"`. Use this property to set the data type of the captured data that the `capture` object function returns.

Note When you update this property, the execution time of the next object function call increases by a few seconds.

Data Types: `string`

`SampleRate` — Baseband sample rate in Hz

highest device sample rate (default) | positive numeric scalar

Baseband sample rate in Hz, specified as positive numeric scalar. The object automatically selects the master clock rate available for the radio based on the specified sample rate. Therefore, the valid range for the sample rate depends on the master clock rate available on the radio. The sample rate must be less than or equal to *MCR*/2 or equal to *MCR*, where *MCR* is the master clock rate that the object selects. For more information, see “Baseband Sample Rate in NI USRP Radios”.

Radio Device	Sample Rate	Master Clock Rate
USRP N310	120,471 Hz to 153.6 MHz `153.6e6` (default)	122.88 MHz 125.00 MHz 153.60 MHz
USRP N320	196,078 Hz to 250 MHz `250e6` (default)	200.00 MHz 245.76 MHz 250.00 MHz
USRP N321	196,078 Hz to 250 MHz `250e6` (default)	200.00 MHz 245.76 MHz 250.00 MHz

Note To update this property, you must stop any ongoing transmission by calling the `stopTransmission` function on the object. When you update this property, the execution time of the next object function call increases by a few seconds.

Data Types: `double`

`DroppedSamplesAction` — Behavior upon dropped samples
`"error"` (default) | `"warning"` | `"none"`

Behavior of the `capture` object function upon dropped samples, specified as one of these values.

- `"error"` — The object function stops with an error message.
- `"warning"` — The object function displays a warning message.
- `"none"` — The object function ignores dropped samples.

Data Types: `string`

Object Functions

capture	Capture IQ data using baseband receiver or transceiver
transmit	Transmit waveform using baseband transmitter or transceiver
stopTransmission	Stop transmission from baseband transmitter or transceiver

Examples

Configure Baseband Transceiver to Transmit and Capture Data

Create a baseband transceiver object, specifying a radio setup configuration previously saved in the Radio Setup wizard.

```
bbtrx = basebandTransceiver("MyRadio")
```

```
bbtrx =
  basebandTransceiver with properties:
```

```
       TransmitRadioGain: 10
  TransmitCenterFrequency: 2.4000e+09
         TransmitAntennas: "RF0:TX/RX"
         CaptureRadioGain: 10
   CaptureCenterFrequency: 2.4000e+09
          CaptureAntennas: "RF0:RX2"
          CaptureDataType: "int16"
    DroppedSamplesAction: "error"
              SampleRate: 250000000
```

Set the baseband sample rate.

```
bbtrx.SampleRate = 122.88e6;
```

Set the transmit and capture center frequencies.

```
bbtrx.TransmitCenterFrequency = 2.2e9;
bbtrx.CaptureCenterFrequency = 2.2e9;
```

Generate random transmit waveform.

```
txWaveform = complex(randn(1000,1),randn(1000,1));
```

Transmit the generated waveform continuously with the radio associated with the baseband transceiver object using the default transmit antenna.

```
transmit(bbtrx,txWaveform,"continuous");
```

Capture IQ data with the radio associated with the baseband transceiver object using the default capture antenna.

```
[data,~] = capture(bbtrx,milliseconds(3));
```

Stop the continuous transmission after data capture is complete.

```
stopTransmission(bbtrx);
```

See Also

Functions
`radioConfigurations`

Objects
`basebandTransmitter` | `basebandReceiver`

Introduced in R2022a

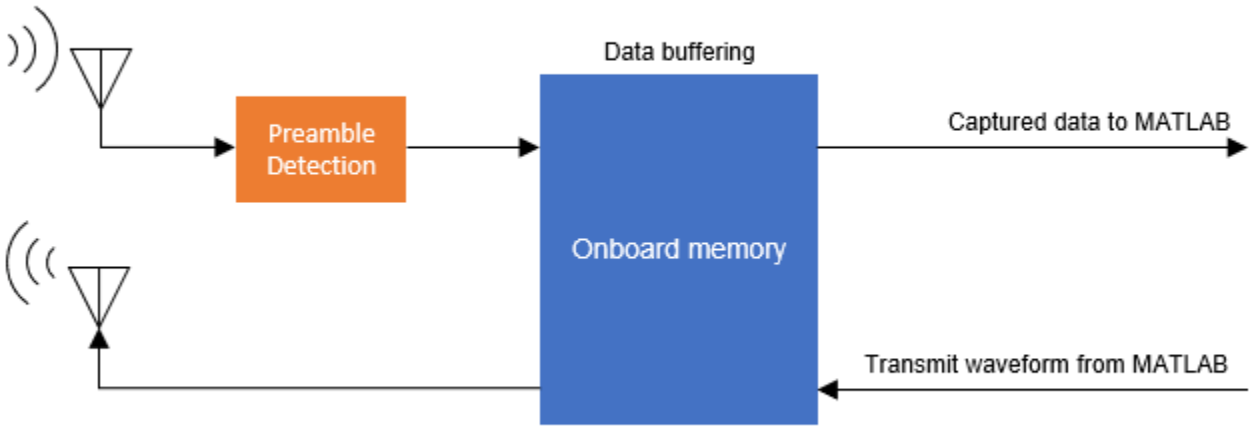
preambleDetector

Configure SDR as preamble detector

Description

Use the `preambleDetector` object to configure the specified software-defined radio (SDR) as a preamble detector, which you can use to detect and capture a signal of interest from the air using a correlation with a known preamble sequence. The object specifies the preamble sequence and the thresholding and triggering parameters. For more details on the internal architecture of the preamble detector, see “Algorithms” on page 1-24.

This diagram shows a conceptual overview of detecting and capturing radio signals in Wireless Testbench using a radio that you configure with this object. The object also enables you to send a test waveform for detection and capture. The onboard data buffering ensures contiguous data capture and transmit.



Creation

Syntax

```
pd = preambleDetector(radio)
pd = preambleDetector(radio,Name=Value)
```

Description

`pd = preambleDetector(radio)` creates a preamble detector configuration object for the specified radio `radio`.

Note The object requires exclusive access to radio hardware resources. Before creating this object, clear any existing Wireless Testbench object associated with the specified radio from the workspace.

`pd = preambleDetector(radio,Name=Value)` sets properties on page 1-19 using one or more name-value arguments. For example, `CaptureDataType="double"` sets the data type of the returned captured signal to `double`.

Input Arguments

`radio` — Radio setup configuration

string scalar

Radio setup configuration, specified as a string scalar. To create a radio setup configuration, set up your radio and save your radio setup configuration using the Radio Setup wizard. Call the `radioConfigurations` function to list all saved radio setup configurations. For more information, see “Connect and Set Up NI USRP Radios”.

Example: `"MyRadio"` indicates that you saved a radio setup configuration under the name MyRadio in the Radio Setup wizard.

Properties

`CenterFrequency` — Capture center frequency in Hz

`2.4e9` (default) | positive numeric scalar

Capture center frequency in Hz, specified as a positive numeric scalar. The valid center frequency range depends on the radio device.

Radio Device	Center Frequency
USRP N310	1 MHz to 6 GHz
USRP N320	1 MHz to 6 GHz
USRP N321	1 MHz to 6 GHz

Data Types: `double`

`RadioGain` — Capture radio gain in dB

`10` (default) | positive numeric scalar

Capture radio gain in dB, specified as a positive numeric scalar. The valid gain range depends on the radio device.

Radio Device	Capture Radio Gain
USRP N310	0 dB to 75 dB
USRP N320	0 dB to 60 dB
USRP N321	0 dB to 60 dB

Data Types: `double`

`Antennas` — Capture radio antenna

`"RF0:RX2"` (default) | string scalar

Capture radio antenna, specified as a string scalar. Use this table to identify a supported radio antenna port on the radio device and the corresponding string constant that you can specify for this property.

Radio Device	Supported Antenna Port	Valid String Scalar
USRP N310	RF0 channel: RX2 port	`"RF0:RX2"` (default)
	RF1 channel: RX2 port	`"RF1:RX2"`
	RF2 channel: RX2 port	`"RF2:RX2"`
	RF3 channel: RX2 port	`"RF3:RX2"`
USRP N320	RF0 channel: RX2 port	`"RF0:RX2"` (default)
	RF1 channel: RX2 port	`"RF1:RX2"`
USRP N321	RF0 channel: RX2 port	`"RF0:RX2"` (default)
	RF1 channel: RX2 port	`"RF1:RX2"`

Note When you update this property, the execution time of the next object function call increases by a few seconds.

Data Types: `string`

`SampleRate` — Baseband sample rate in Hz

highest device sample rate (default) | positive numeric scalar

Baseband sample rate in Hz, specified as positive numeric scalar. The object automatically selects the master clock rate available for the radio based on the specified sample rate. Therefore, the valid range for the sample rate depends on the master clock rate available on the radio. The sample rate must be less than or equal to *MCR*/2 or equal to *MCR*, where *MCR* is the master clock rate that the object selects. For more information, see “Baseband Sample Rate in NI USRP Radios”.

Radio Device	Sample Rate	Master Clock Rate
USRP N310	120,471 Hz to 153.6 MHz `153.6e6` (default)	122.88 MHz 125.00 MHz 153.60 MHz
USRP N320	196,078 Hz to 250 MHz `250e6` (default)	200.00 MHz 245.76 MHz 250.00 MHz
USRP N321	196,078 Hz to 250 MHz `250e6` (default)	200.00 MHz 245.76 MHz 250.00 MHz

Note To update this property, you must stop any ongoing transmission by calling the `stopTransmission` function on the object. When you update this property, the execution time of the next object function call increases by a few seconds.

Data Types: `double`

`CaptureDataType` — Data type of captured data

`"int16"` (default) | `"double"` | `"single"`

Data type of the captured data, specified as `"int16"`, `"double"`, or `"single"`. Use this property to set the data type of the captured data that the `capture` object function returns.

Note When you update this property, the execution time of the next object function call increases by a few seconds.

Data Types: `string`

`DroppedSamplesAction` — Behavior upon dropped samples

`"error"` (default) | `"warning"` | `"none"`

Behavior of the `capture` and `plotThreshold` object functions upon dropped samples, specified as one of these values.

- `"error"` — The object function stops with an error message.
- `"warning"` — The object function displays a warning message.
- `"none"` — The object function ignores dropped samples.

Data Types: `string`

`Preamble` — Preamble sequence

16-by-1 vector of all zeros (default) | numeric column vector

Preamble sequence, specified as a numeric column vector with vector elements in the range [– 1, 1).

The first object function call validates the preamble length against the maximum preamble length. The maximum preamble length is equal to `floor(`*MCR*`/SampleRate)×48`, where *MCR* is the master clock rate that the object automatically selects for the radio based on the sample rate. Lower sample rates enable longer preambles.

You can assign a `double`, `single`, or `fi` (requires Fixed-Point Designer™) data type to this property. To assign a fixed-point data type, use the `fi` object. The fixed-point data type must be signed with the `WordLength` property set to `16` and `FractionLength` property set to `15`.

Note To update this property, you must stop any ongoing transmission by calling the `stopTransmission` function on the object. When you update this property, the execution time of the next object function call increases by a few seconds.

Data Types: `double` | `single` | `fi`

ThresholdMethod — Threshold calculation method

"adaptive" (default) | "fixed"

Threshold calculation method to trigger data capture, specified as one of these values.

- "adaptive" — The threshold is the scaled signal power. To configure the scaled signal power, set the AdaptiveThresholdGain and AdaptiveThresholdOffset properties.
- "fixed" — The threshold is a constant, specified by the FixedThreshold property.

For more details, see “Internal Architecture of Preamble Detection” on page 1-24.

Note To update this property, you must stop any ongoing transmission by calling the stopTransmission function on the object. When you update this property, the execution time of the next object function call increases by a few seconds.

Data Types: double

FixedThreshold — Fixed threshold value

0 (default) | numeric scalar in the range [0, 4095]

Fixed threshold value, specified as a numeric scalar in the range [0, 4095]. For more details, see “Internal Architecture of Preamble Detection” on page 1-24.

Dependencies

This property is applicable only when ThresholdMethod is set to "fixed".

Data Types: double

AdaptiveThresholdOffset — Adaptive threshold offset

0 (default) | numeric scalar in the range [0, 2]

Adaptive threshold offset to avoid false trigger points, specified as a numeric scalar in the range [0, 2]. For more details, see “Internal Architecture of Preamble Detection” on page 1-24.

Dependencies

This property is applicable only when ThresholdMethod is set to "adaptive".

Data Types: double

AdaptiveThresholdGain — Adaptive threshold gain

0 (default) | numeric scalar in the range [0, 64]

Adaptive threshold gain value applied to the average input signal power before adaptive threshold calculation, specified as a numeric scalar in the range [0, 64]. For more details, see “Internal Architecture of Preamble Detection” on page 1-24.

Dependencies

This property is applicable only when ThresholdMethod is set to "adaptive".

Data Types: double

TriggerOffset — Trigger point offset

0 (default) | integer in the range [–3096, 4096]

Trigger point offset, specified as an integer in the range [–3096, 4096]. This value specifies the start of data capture relative to the trigger point. For more details, see “Thresholding and Triggering” on page 1-25.

Note To update this property, you must stop any ongoing transmission by calling the `stopTransmission` function on the object. When you update this property, the execution time of the next object function call increases by a few seconds.

Data Types: `double`

Object Functions

capture	Capture signal of interest from the air upon detection
plotThreshold	Plot preamble detection signals for triggering
transmit	Transmit waveform using preamble detector
stopTransmission	Stop transmission from preamble detector

Examples

Configure Preamble Detector and Capture Data

Define a preamble sequence with good correlation properties. For example, generate and normalize a Zadoff-Chu sequence of length 137.

```
seq = zadoffChuSeq(38,137);
preamble = seq/norm(seq,2);
```

Create and configure a preamble detector object, specifying a radio setup configuration previously saved in the Radio Setup wizard.

```
pd = preambleDetector("MyRadio")
```

```
pd =
  preambleDetector with properties:

            CenterFrequency: 2.4000e+09
                  RadioGain: 10
                   Antennas: "RF0:RX2"
                 SampleRate: 250000000
            CaptureDataType: "int16"
       DroppedSamplesAction: "error"
                   Preamble: 0
            ThresholdMethod: "adaptive"
             FixedThreshold: 0
    AdaptiveThresholdOffset: 0
      AdaptiveThresholdGain: 0
              TriggerOffset: 0
```

```
pd.SampleRate = 10.24e6;
pd.CenterFrequency = 2.2e9;
pd.CaptureDataType = "double";
```

Specify the preamble.

```
pd.Preamble = preamble;
```

Set the trigger offset to a negative value to capture 500 samples before the trigger point.

```
pd.TriggerOffset = -500;
```

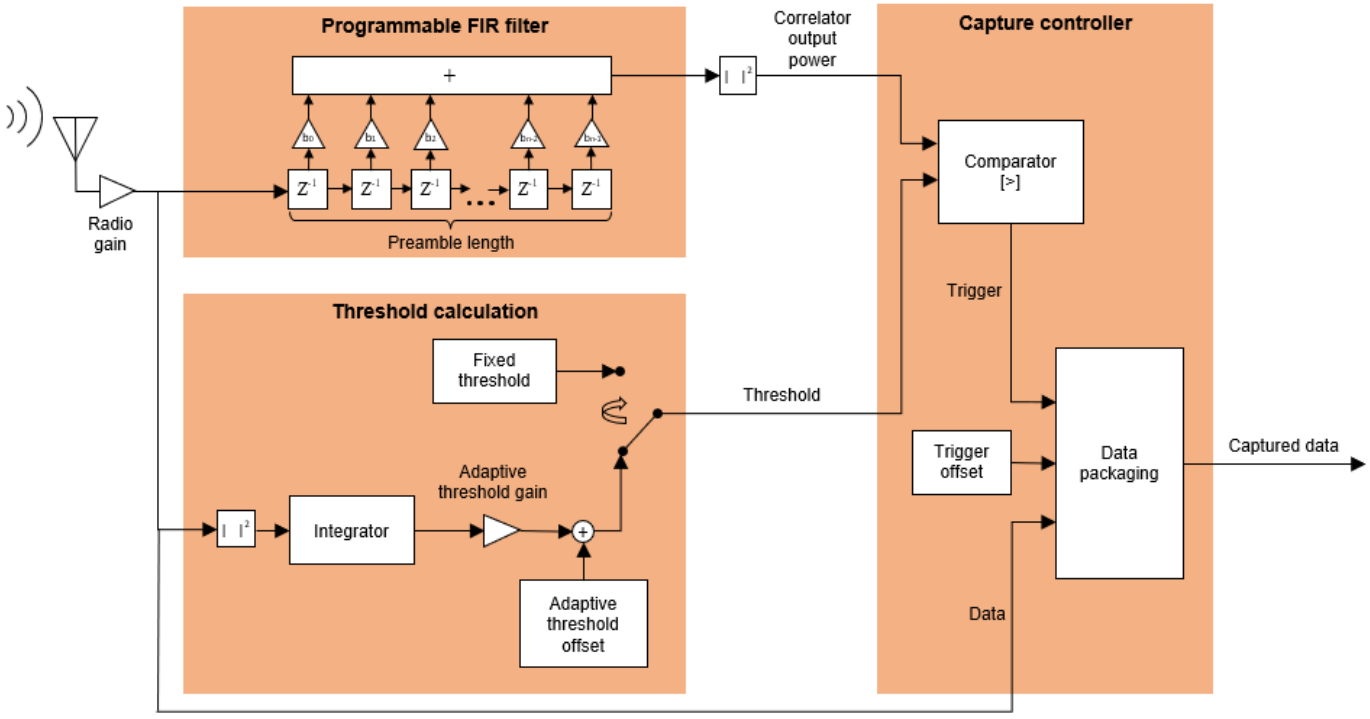
Capture 10 ms of data with a timeout of 1 second.

```
[data,timestamp,droppedSamples,status] = capture(pd,milliseconds(10),seconds(1));
```

Algorithms

Internal Architecture of Preamble Detection

This diagram shows a conceptual overview of triggered capture based on preamble detection. The input is an RF signal received from the RF board of the radio.



- The **Programmable FIR filter** correlates the input signal with a known preamble sequence. The FIR filter has 48 complex-number MAC elements that the detector can reuse to lengthen the filter. The achievable number of taps depends on the `SampleRate` property. To specify the preamble sequence, use the `Preamble` property.
- The **Threshold calculation** component provides the threshold for the trigger point generation. For more information on how to adjust the threshold, see “Thresholding and Triggering” on page 1-25.
- The **Capture controller** component controls the triggered capture.
  - The **Comparator** component generates the trigger point, which is the first data sample for which the correlator output power is greater than the threshold.

- The **Data packaging** component starts the data capture at the trigger point. To adjust the start of the data capture relative to the trigger point, use the `TriggerOffset` property.

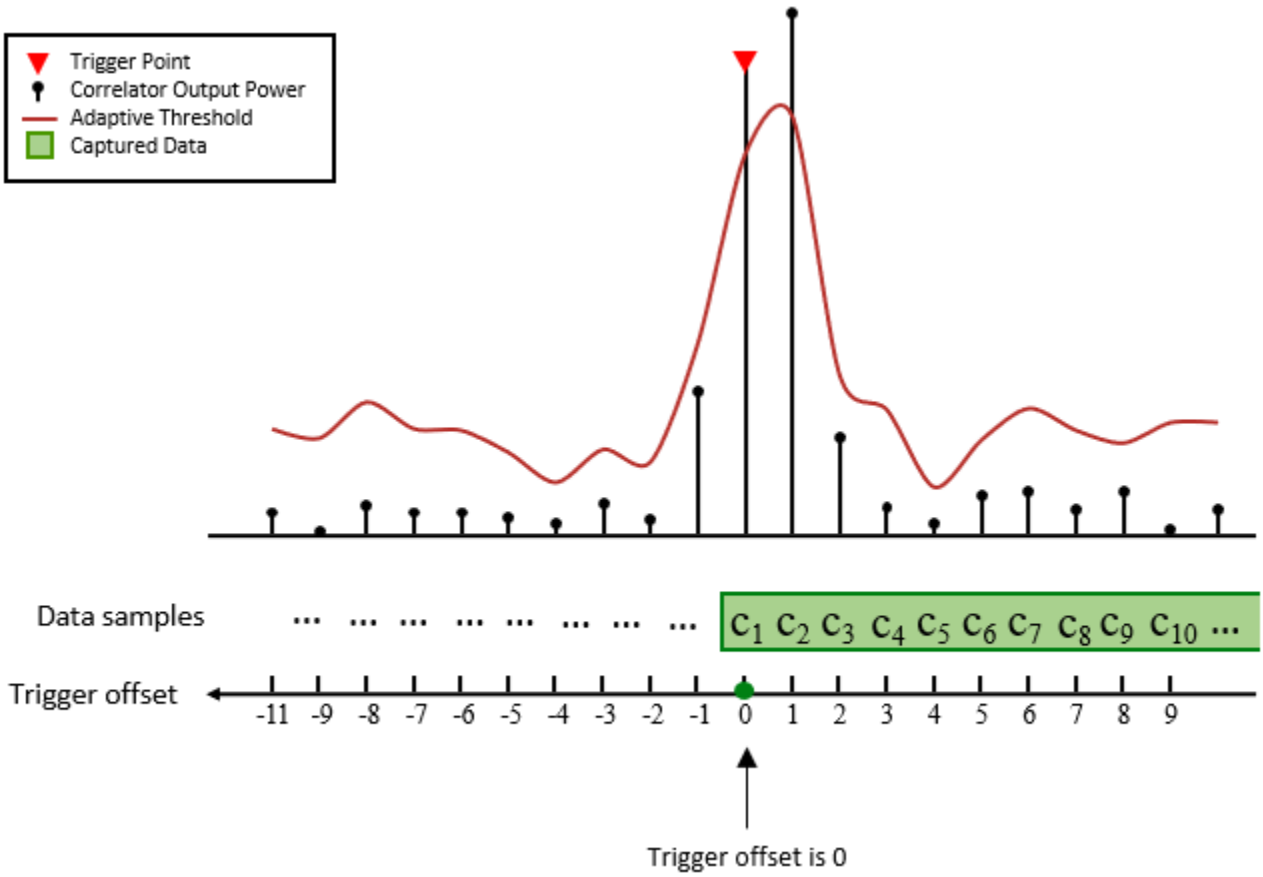
Thresholding and Triggering

The trigger point is the first data sample for which the correlator output power is greater than the threshold. To calibrate the thresholding and triggering operation:

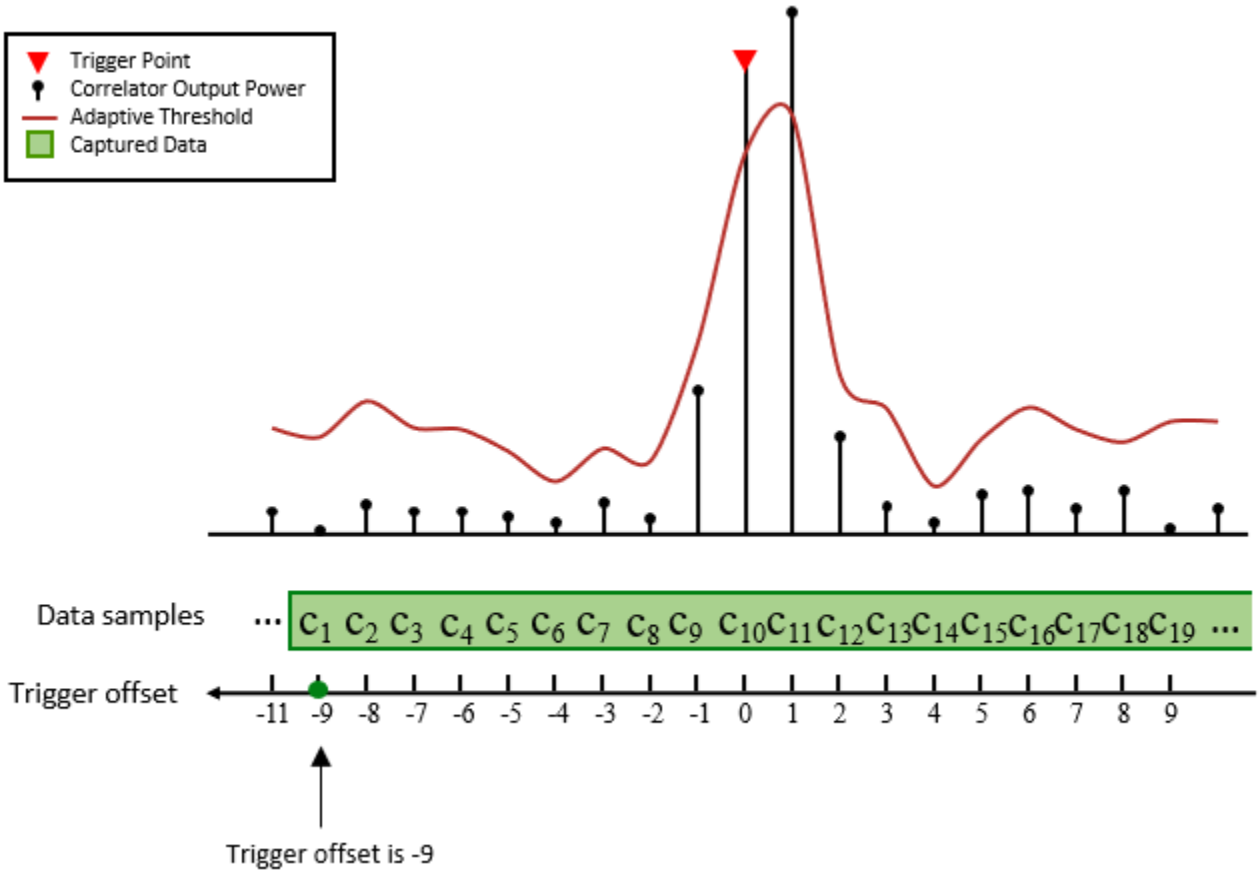
1. Adjust the gain on the RF signal by using the `RadioGain` property.
2. Adjust the threshold. You can specify an adaptive threshold or a fixed threshold.
   - For adaptive threshold, set the `ThresholdMethod` property to `"adaptive"`, then use the `AdaptiveThresholdGain` property to set the adaptive threshold gain value. To avoid false triggers, adjust the `AdaptiveThresholdOffset` property.
   - For fixed threshold, set the `ThresholdMethod` property to `"fixed"`, then use the `FixedThreshold` property to set the fixed threshold value.
3. Call the `plotThreshold` object function to display and analyze the correlator output power, threshold, and corresponding trigger points.

By default, data capture stars at the trigger point. To adjust the start of the data capture relative to the trigger point, use the `TriggerOffset` property.

For example, this figure shows a scenario in which the trigger offset is 0. Hence, data capture starts at the trigger point.



In this second scenario, the trigger point is the same, but the trigger offset is –9. Hence, data capture starts at the specified trigger offset.



For an example of how to calibrate the thresholding and triggering operation, see “Triggered Capture Using Preamble Detection”.

See Also

Functions
`radioConfigurations`

Topics
“Triggered Capture Using Preamble Detection”
“Supported Radio Devices”
“Connect and Set Up NI USRP Radios”

Introduced in R2022a

Functions

capture

Capture signal of interest from the air upon detection

Syntax

```
[data,timestamp,droppedSamples,status] = capture(pd,length,timeout)
```

Description

`[data,timestamp,droppedSamples,status] = capture(pd,length,timeout)` captures a signal of interest of length `length` from the air using the preamble detector `pd` with the detection timeout `timeout`. The function returns captured signal `data`, capture request timestamp `timestamp`, dropped samples status `droppedSamples`, and capture outcome status `status`.

Examples

Configure Preamble Detector and Capture Data

Define a preamble sequence with good correlation properties. For example, generate and normalize a Zadoff-Chu sequence of length 137.

```
seq = zadoffChuSeq(38,137);
preamble = seq/norm(seq,2);
```

Create and configure a preamble detector object, specifying a radio setup configuration previously saved in the Radio Setup wizard.

```
pd = preambleDetector("MyRadio")
```

```
pd =
  preambleDetector with properties:

           CenterFrequency: 2.4000e+09
                 RadioGain: 10
                  Antennas: "RF0:RX2"
                SampleRate: 250000000
           CaptureDataType: "int16"
       DroppedSamplesAction: "error"
                  Preamble: 0
           ThresholdMethod: "adaptive"
            FixedThreshold: 0
    AdaptiveThresholdOffset: 0
      AdaptiveThresholdGain: 0
              TriggerOffset: 0
```

```
pd.SampleRate = 10.24e6;
pd.CenterFrequency = 2.2e9;
pd.CaptureDataType = "double";
```

Specify the preamble.

```
pd.Preamble = preamble;
```

Set the trigger offset to a negative value to capture 500 samples before the trigger point.

```
pd.TriggerOffset = -500;
```

Capture 10 ms of data with a timeout of 1 second.

```
[data,timestamp,droppedSamples,status] = capture(pd,milliseconds(10),seconds(1));
```

Input Arguments

pd — Preamble detector

`preambleDetector` object

Preamble detector, specified as a `preambleDetector` object.

Note The first object function call in which you specify this object as an input requires a few extra seconds to load the application onto the hardware.

length — Capture length

integer number of samples | `duration`

Capture length, specified as an integer number of samples or a `duration` value in time units. The function converts `length` into *N* samples based on the `SampleRate` property of the `pd` input and captures `ceil`(*N*) number of data samples.

Specify the capture length relative to the onboard radio memory buffer size. The buffer has a capacity of 2 GB, which amounts to a total of 2^{29} data samples.

Note

- Capture and transmit data samples are buffered in the onboard radio memory. Therefore, when specifying the capture length, you must also take into account the length of the transmit waveform of any continuous transmission that you specify when calling the `transmit` object function with the `pd` input.
- If your host computer does not have enough free memory to receive the captured data from the radio buffer, the function call can hang or error out. To free up memory space on your host computer, try closing other software or reduce the capture length.

Example: `seconds(5)`

Data Types: `double` | `duration`

timeout — Preamble signal detection timeout

`duration`

Preamble signal detection timeout, specified as a `duration` value in time units.

Example: `seconds(5)`

Data Types: duration

Output Arguments

data — Captured signal

column vector of complex values

Captured signal, returned as a column vector of complex values. Use the CaptureDataType property of the pd input to specify the output data type of the captured data. If you specify the return data type as single or double, the function scales the captured data sample values to the range [–1, 1].

Data Types: int16 | single | double
Complex Number Support: Yes

timestamp — Capture request timestamp

datetime value

Capture request timestamp, returned as a datetime value. The function creates this timestamp just before requesting data capture from the hardware.

Data Types: datetime

droppedSamples — Status of dropped samples

1 | 0

Status of dropped samples, returned as one of these logical values.

- 1 — Samples are dropped during capture.
- 0 — Samples are not dropped during capture.

Use the DroppedSamplesAction property of the pd input to specify the behavior of the capture function upon dropped samples.

Data Types: logical

status — Capture outcome status

1 | 0

Capture outcome status, returned as one of these logical values.

- 1 — Capture operation was successful and the captured signal of interest is returned in data.
- 0 — Detect and capture operation timed out.

Data Types: logical

See Also

Functions

duration | plotThreshold

Introduced in R2022a

capture

Capture IQ data using baseband receiver or transceiver

Syntax

```
[data,timestamp,droppedSamples] = capture(bba,length)
```

Description

`[data,timestamp,droppedSamples] = capture(bba,length)` captures IQ data of length `length` from the air using the specified baseband receiver or baseband transceiver `bba`. The function returns the captured signal `data`, capture request timestamp `timestamp`, and dropped samples status `droppedSamples`.

Examples

Configure Baseband Receiver and Capture Data

Create a baseband receiver object, specifying a radio setup configuration previously saved in the Radio Setup wizard.

```
bbrx = basebandReceiver("MyRadio")
```

```
bbrx =
  basebandReceiver with properties:

              RadioGain: 10
         CenterFrequency: 2.4000e+09
              SampleRate: 250000000
                Antennas: "RF0:RX2"
    DroppedSamplesAction: "error"
         CaptureDataType: "int16"
```

Set the baseband sample rate and center frequency.

```
bbrx.SampleRate = 122.88e6;
bbrx.CenterFrequency = 2.2e9;
```

Capture 3 ms of IQ data with the radio associated with the baseband receiver object using the default antenna.

```
[data,~] = capture(bbrx,milliseconds(3));
```

Configure Baseband Transceiver to Transmit and Capture Data

Create a baseband transceiver object, specifying a radio setup configuration previously saved in the Radio Setup wizard.

```
bbtrx = basebandTransceiver("MyRadio")
```

```
bbtrx =
  basebandTransceiver with properties:

          TransmitRadioGain: 10
    TransmitCenterFrequency: 2.4000e+09
           TransmitAntennas: "RF0:TX/RX"
           CaptureRadioGain: 10
     CaptureCenterFrequency: 2.4000e+09
            CaptureAntennas: "RF0:RX2"
            CaptureDataType: "int16"
       DroppedSamplesAction: "error"
                 SampleRate: 250000000
```

Set the baseband sample rate.

```
bbtrx.SampleRate = 122.88e6;
```

Set the transmit and capture center frequencies.

```
bbtrx.TransmitCenterFrequency = 2.2e9;
bbtrx.CaptureCenterFrequency = 2.2e9;
```

Generate random transmit waveform.

```
txWaveform = complex(randn(1000,1),randn(1000,1));
```

Transmit the generated waveform continuously with the radio associated with the baseband transceiver object using the default transmit antenna.

```
transmit(bbtrx,txWaveform,"continuous");
```

Capture IQ data with the radio associated with the baseband transceiver object using the default capture antenna.

```
[data,~] = capture(bbtrx,milliseconds(3));
```

Stop the continuous transmission after data capture is complete.

```
stopTransmission(bbtrx);
```

Input Arguments

bba — Baseband application

`basebandReceiver` object | `basebandTransceiver` object

Baseband application, specified as a `basebandReceiver` object or `basebandTransceiver` object.

Note The first object function call in which you specify this object as an input requires a few extra seconds to load the application onto the hardware.

length — Capture length

integer number of samples | `duration`

Capture length, specified as an integer number of samples or a `duration` value in time units. The function converts `length` into *N* samples based on the `SampleRate` property of the `bba` input and captures `ceil`(*N*) number of data samples.

Specify the capture length relative to the onboard radio memory buffer size. The buffer has a capacity of 2 GB, which amounts to a total of 2^{29} data samples.

Note

- Transmit and capture data samples on the baseband transceiver are buffered in the onboard radio memory. Therefore, if the `bba` input is a baseband transceiver, you must also take into account the length of the transmit waveform of any continuous transmission that you specify when calling the `transmit` object function with the `bba` input.
- If your host computer does not have enough free memory to receive the captured data from the radio buffer, the function call can hang or error out. To free up memory space on your host computer, try closing other software or reduce the capture length.

Example: `seconds(5)`

Data Types: `double` | `duration`

Output Arguments

`data` — Captured signal

complex-valued column vector | complex-valued matrix

Captured signal, returned as one of these options.

- Complex-valued column vector — The vector contains data that is captured on a single capture antenna.
- Complex-valued matrix — The matrix contains data that is captured on multiple capture antennas. This option applies only when `bba` is a baseband receiver. The number of antennas specified by the `Antennas` property of the `bba` input determines the number of matrix columns.

Use the `bba`.`CaptureDataType` property to specify the data type of the returned data. If you specify the return data type as `single` or `double`, the function scales the captured data sample values to the range [–1, 1].

Note The first data samples of the captured signal can contain transient values from the radio data path.

Data Types: `int16` | `single` | `double`
Complex Number Support: Yes

`timestamp` — Capture request timestamp

`datetime` value

Capture request timestamp, returned as a `datetime` value. The function creates this timestamp just before requesting data capture from the hardware.

Data Types: `datetime`

`droppedSamples` — Status of dropped samples
`1` | `0`

Status of dropped samples, returned as one of these logical values.

- `1` — Samples are dropped during capture.
- `0` — Samples are not dropped during capture.

Use the `DroppedSamplesAction` property of the `bba` input to specify the behavior of the function upon dropped samples.

Data Types: `logical`

See Also

Functions
`transmit`

Objects
`basebandReceiver` | `basebandTransceiver`

Introduced in R2022a

radioConfigurations

List saved radio setup configurations

Syntax

```
radios = radioConfigurations
```

Description

`radios = radioConfigurations` lists all radio setup configurations that you saved using the Radio Setup wizard. Use this function to identify your radio when you create the `preambleDetector`, `basebandReceiver`, `basebandTransceiver`, or `basebandTransmitter` objects.

Examples

List Radio Setup Configurations

```
radios = radioConfigurations;
```

Specify the name of a saved radio setup configuration when creating an application object, for example, a preamble detector object.

```
radioName = radios(1).Name;
pd = preambleDetector(radioName);
```

Output Arguments

`radios` — Radio setup configurations

structure array

Radio setup configurations, returned as a structure array. Each structure contains the name of a radio setup configuration and the associated radio device and IP address information.

Data Types: `struct`

See Also

Objects

`preambleDetector` | `basebandReceiver` | `basebandTransceiver` | `basebandTransmitter`

Topics

“Supported Radio Devices”
“Connect and Set Up NI USRP Radios”

Introduced in R2022a

plotThreshold

Plot preamble detection signals for triggering

Syntax

```
plotThreshold(pd,length)
droppedSamples = plotThreshold(pd,length)
```

Description

`plotThreshold(pd,length)` plots preamble detection signals while detecting a preamble in live data of length `length` from the air using the preamble detector `pd`. The plot shows the correlator output power, threshold, and trigger points of the detection. Use the displayed information to calibrate the threshold in the preamble detector for triggering.

`droppedSamples = plotThreshold(pd,length)` returns the status information of dropped samples, in addition to plotting preamble detection signals.

Examples

Configure Preamble Detector and Plot Detection Threshold

Define a preamble sequence with good correlation properties. For example, generate and normalize a Zadoff-Chu sequence of length 137.

```
seq = zadoffChuSeq(38,137);
preamble = seq/norm(seq,2);
```

Create and configure a preamble detector object, specifying a radio setup configuration previously saved in the Radio Setup wizard.

```
pd = preambleDetector("MyRadio")
```

```
pd =
  preambleDetector with properties:

           CenterFrequency: 2.4000e+09
                 RadioGain: 10
                  Antennas: "RF0:RX2"
                SampleRate: 250000000
           CaptureDataType: "int16"
       DroppedSamplesAction: "error"
                  Preamble: 0
           ThresholdMethod: "adaptive"
            FixedThreshold: 0
    AdaptiveThresholdOffset: 0
      AdaptiveThresholdGain: 0
              TriggerOffset: 0
```

```
pd.SampleRate = 30.72e6;
pd.CenterFrequency = 2.45e9;
pd.RadioGain = 45;
```

Specify the preamble.

```
pd.Preamble = preamble;
```

Specify a fixed threshold.

```
pd.ThresholdMethod = "Fixed";
pd.FixedThreshold = 5;
```

Generate test waveform for detection.

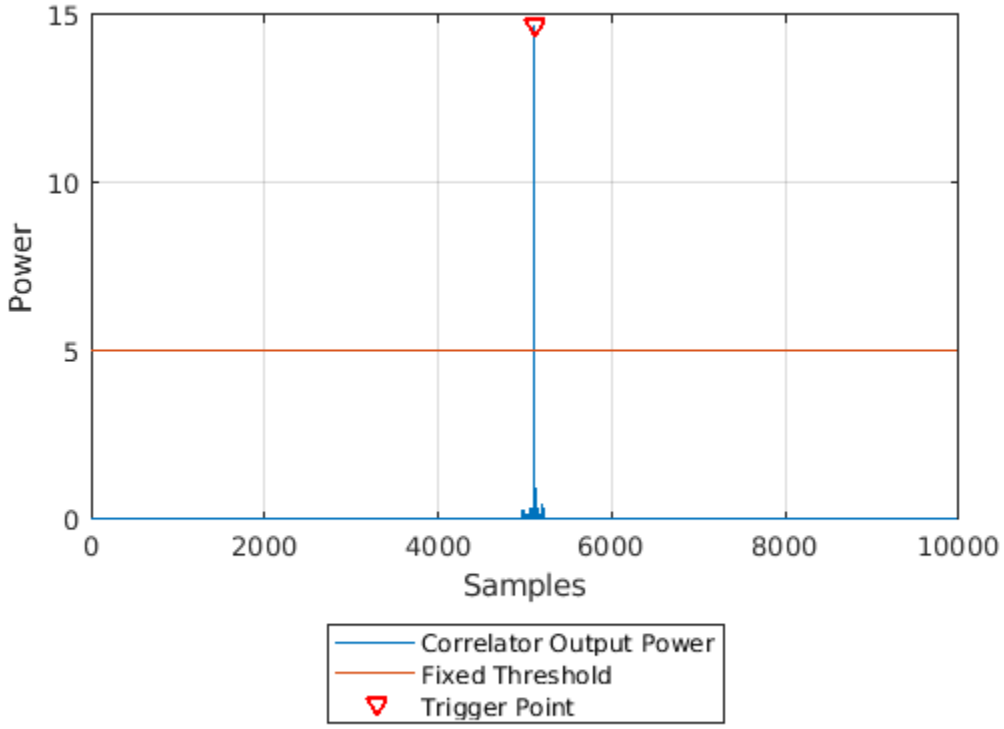
```
prePadLen = 2501;
postPadLen = 2500;
headSignal = complex(zeros(prePadLen,1),zeros(prePadLen,1));
rearSignal = complex(zeros(postPadLen,1),zeros(postPadLen,1));
testWaveform = [headSignal; preamble; rearSignal];
```

Send the test waveform continuously.

```
transmit(pd,testWaveform,"continuous", ...
    TransmitGain=45, ...
    TransmitCenterFrequency=2.45e9, ...
    TransmitAntennas="RF0:TX/RX");
```

Plot the specified number of samples of the correlator output power. Use the displayed information to calibrate thresholding parameters.

```
plotThreshold(pd,10000);
```



Stop test waveform transmission.

```
stopTransmission(pd);
```

Input Arguments

pd — Preamble detector

`preambleDetector` object

Preamble detector, specified as a `preambleDetector` object.

Note The first object function call in which you specify this object as an input requires a few extra seconds to load the application onto the hardware.

`length` — Plotted data length

integer number of samples | `duration`

Plotted data length, specified as an integer number of samples or a `duration` value in time units. The function converts `length` into N samples based on the `pd.SampleRate` property and plots `ceil`(N) number of data samples.

Specify the plotted data length relative to the onboard radio memory buffer size. The buffer has a capacity of 2 GB, which amounts to a total of 2^{29} data samples.

Note Plotted and transmit data samples are buffered in the onboard radio memory. Therefore, when specifying the plotted data length, you must also take into account the length of the transmit waveform of any continuous transmission that you specify when calling the `transmit` object function with the `pd` input.

Example: `seconds(5)`

Data Types: `double` | `duration`

Output Arguments

`droppedSamples` — Status of dropped samples

`1` | `0`

Status of dropped samples, returned as one of these logical values.

- `1` — Samples are dropped while plotting the signals.
- `0` — Samples are not dropped while plotting the signals.

Use the `DroppedSamplesAction` property of the `pd` input to specify the behavior of the `plotThreshold` function upon dropped samples.

Data Types: `logical`

See Also

Functions

`capture`

Introduced in R2022a

stopTransmission

Stop transmission from preamble detector

Syntax

```
stopTransmission(pd)
```

Description

`stopTransmission(pd)` stops continuous IQ waveform transmission to the air from the preamble detector `pd`.

Examples

Configure Preamble Detector and Capture Test Waveform

Configure a preamble detector to continuously send a test waveform with a known preamble sequence and capture data samples of the transmitted waveform.

Define a preamble sequence with good correlation properties. For example, generate and normalize a Zadoff-Chu sequence of length 137.

```
seq = zadoffChuSeq(38,137);
preamble = seq/norm(seq,2);
```

Generate a test waveform.

```
prePadding = complex(zeros(999,1),zeros(999,1));
postPadding = complex(zeros(1000,1),zeros(1000,1));
testWaveform = [prePadding;preamble;postPadding;];
```

Create and configure a preamble detector object, specifying a radio setup configuration previously saved in the Radio Setup wizard.

```
pd = preambleDetector("MyRadio")
```

```
pd =
  preambleDetector with properties:

           CenterFrequency: 2.4000e+09
                 RadioGain: 10
                  Antennas: "RF0:RX2"
                SampleRate: 250000000
           CaptureDataType: "int16"
       DroppedSamplesAction: "error"
                  Preamble: 0
           ThresholdMethod: "adaptive"
            FixedThreshold: 0
    AdaptiveThresholdOffset: 0
      AdaptiveThresholdGain: 0
              TriggerOffset: 0
```

```
pd.SampleRate = 10.24e6;
```

Specify the preamble.

```
pd.Preamble = preamble;
```

Send the test waveform continuously.

```
transmit(pd,testWaveform,"continuous", ...
    TransmitGain=40, ...
    TransmitCenterFrequency=2.45e9, ...
    TransmitAntennas="RF0:TX/RX");
```

Capture 10 ms of data of the transmitted waveform.

```
[data,timestamp,droppedSamples,status] = capture(pd,milliseconds(10),seconds(1));
```

Stop test waveform transmission.

```
stopTransmission(pd);
```

Input Arguments

pd — Preamble detector

`preambleDetector` object

Preamble detector, specified as a `preambleDetector` object.

Note The first object function call in which you specify this object as an input requires a few extra seconds to load the application onto the hardware.

See Also

Functions

`transmit`

Introduced in R2022a

stopTransmission

Stop transmission from baseband transmitter or transceiver

Syntax

```
stopTransmission(bba)
```

Description

`stopTransmission(bba)` stops continuous IQ waveform transmission from the specified baseband transmitter or baseband transceiver `bba`.

Examples

Configure Baseband Transmitter and Transmit Waveform

Create a baseband transmitter object, specifying a radio setup configuration previously saved in the Radio Setup wizard.

```
bbtx = basebandTransmitter("MyRadio")
```

```
bbtx =
  basebandTransmitter with properties:

          RadioGain: 10
    CenterFrequency: 2.4000e+09
         SampleRate: 250000000
           Antennas: "RF0:TX/RX"
```

Set the baseband sample rate and center frequency.

```
bbtx.SampleRate = 122.88e6;
bbtx.CenterFrequency = 2.2e9;
```

Generate random transmit waveform.

```
txWaveform = complex(randn(1000,1),randn(1000,1));
```

Transmit the generated waveform continuously with the radio associated with the baseband transmitter object using the default antenna.

```
transmit(bbtx,txWaveform,"continuous");
```

Stop the continuous transmission after 5 seconds.

```
pause(5);
stopTransmission(bbtx);
```

Configure Baseband Transceiver to Transmit and Capture Data

Create a baseband transceiver object, specifying a radio setup configuration previously saved in the Radio Setup wizard.

```
bbtrx = basebandTransceiver("MyRadio")
```

```
bbtrx =
  basebandTransceiver with properties:

          TransmitRadioGain: 10
    TransmitCenterFrequency: 2.4000e+09
           TransmitAntennas: "RF0:TX/RX"
           CaptureRadioGain: 10
     CaptureCenterFrequency: 2.4000e+09
            CaptureAntennas: "RF0:RX2"
            CaptureDataType: "int16"
       DroppedSamplesAction: "error"
                 SampleRate: 250000000
```

Set the baseband sample rate.

```
bbtrx.SampleRate = 122.88e6;
```

Set the transmit and capture center frequencies.

```
bbtrx.TransmitCenterFrequency = 2.2e9;
bbtrx.CaptureCenterFrequency = 2.2e9;
```

Generate random transmit waveform.

```
txWaveform = complex(randn(1000,1),randn(1000,1));
```

Transmit the generated waveform continuously with the radio associated with the baseband transceiver object using the default transmit antenna.

```
transmit(bbtrx,txWaveform,"continuous");
```

Capture IQ data with the radio associated with the baseband transceiver object using the default capture antenna.

```
[data,~] = capture(bbtrx,milliseconds(3));
```

Stop the continuous transmission after data capture is complete.

```
stopTransmission(bbtrx);
```

Input Arguments

bba — Baseband application

`basebandTransmitter` object | `basebandTransceiver` object

Baseband application, specified as a `basebandTransmitter` object or `basebandTransceiver` object.

Note The first object function call in which you specify this object as an input requires a few extra seconds to load the application onto the hardware.

See Also

Functions
`transmit`

Introduced in R2022a

transmit

Transmit waveform using preamble detector

Syntax

```
transmit(pd,waveform,"continuous")
transmit( ___ ,Name=Value)
```

Description

`transmit(pd,waveform,"continuous")` continuously transmits the IQ waveform `waveform` to the air using the preamble detector `pd`. Use this function to send a test waveform to the air while trying to detect and capture a signal of interest using the same preamble detector. To stop the transmission, call `stopTransmission(pd)`.

`transmit( ___ ,Name=Value)` specifies optional name-value arguments for the transmission, in addition to the input arguments from the previous syntax.

Examples

Configure Preamble Detector and Capture Test Waveform

Configure a preamble detector to continuously send a test waveform with a known preamble sequence and capture data samples of the transmitted waveform.

Define a preamble sequence with good correlation properties. For example, generate and normalize a Zadoff-Chu sequence of length 137.

```
seq = zadoffChuSeq(38,137);
preamble = seq/norm(seq,2);
```

Generate a test waveform.

```
prePadding = complex(zeros(999,1),zeros(999,1));
postPadding = complex(zeros(1000,1),zeros(1000,1));
testWaveform = [prePadding;preamble;postPadding;];
```

Create and configure a preamble detector object, specifying a radio setup configuration previously saved in the Radio Setup wizard.

```
pd = preambleDetector("MyRadio")
```

```
pd =
  preambleDetector with properties:

            CenterFrequency: 2.4000e+09
                  RadioGain: 10
                   Antennas: "RF0:RX2"
                 SampleRate: 250000000
            CaptureDataType: "int16"
       DroppedSamplesAction: "error"
```

```
                  Preamble: 0
            ThresholdMethod: "adaptive"
             FixedThreshold: 0
    AdaptiveThresholdOffset: 0
      AdaptiveThresholdGain: 0
              TriggerOffset: 0
```

```
pd.SampleRate = 10.24e6;
```

Specify the preamble.

```
pd.Preamble = preamble;
```

Send the test waveform continuously.

```
transmit(pd,testWaveform,"continuous", ...
    TransmitGain=40, ...
    TransmitCenterFrequency=2.45e9, ...
    TransmitAntennas="RF0:TX/RX");
```

Capture 10 ms of data of the transmitted waveform.

```
[data,timestamp,droppedSamples,status] = capture(pd,milliseconds(10),seconds(1));
```

Stop test waveform transmission.

```
stopTransmission(pd);
```

Input Arguments

pd — Preamble detector

`preambleDetector` object

Preamble detector, specified as a `preambleDetector` object.

Note The first object function call in which you specify this object as an input requires a few extra seconds to load the application onto the hardware.

`waveform` — IQ waveform to transmit

column vector of complex values

IQ waveform to transmit, specified as a column vector of complex values. The waveform length must be relative to the onboard radio memory buffer size. The buffer has a capacity of 2 GB, which amounts to a total of 2^29 data samples.

Note Transmit data samples and capture or plotted data samples are buffered in the onboard radio memory. Therefore, when specifying the transmit waveform, take also into account the data capture length or plotted data length that you specify when calling the `capture` or `plotThreshold` object functions, respectively, with the `pd` input.

If you specify a waveform with a `single` or `double` data type, you must scale the transmit data sample values to the range [–1, 1).

Data Types: `int16` | `single` | `double`
Complex Number Support: Yes

Name-Value Pair Arguments

Specify optional pairs of arguments as `Name1=Value1,...,NameN=ValueN`, where `Name` is the argument name and `Value` is the corresponding value. Name-value arguments must appear after other arguments, but the order of the pairs does not matter.

Example: `TransmitGain=50` applies a gain of 50 dB to the transmit waveform.

`TransmitGain` — Transmit gain in dB

10 (default) | positive numeric scalar

Transmit gain in dB, specified as a positive numeric scalar. The valid gain range depends on the radio device.

Radio Device	Transmit Radio Gain
USRP N310	0 dB to 65 dB
USRP N320	0 dB to 60 dB
USRP N321	0 dB to 60 dB

Data Types: `double`

`TransmitCenterFrequency` — Transmit center frequency in Hz

`pd.CenterFrequency` (default) | positive numeric scalar

Transmit center frequency in Hz, specified as a positive numeric scalar. The valid center frequency range depends on the radio device.

Radio Device	Center Frequency
USRP N310	1 MHz to 6 GHz
USRP N320	1 MHz to 6 GHz
USRP N321	1 MHz to 6 GHz

Data Types: `double`

`TransmitAntennas` — Transmit radio antenna

`"RF0:TX/RX"` (default) | string constant

Transmit radio antenna, specified as a string constant. Use this table to identity a supported radio antenna port on the radio device and the corresponding string constant that you can specify for this input.

Radio Device	Supported Antenna Port	Valid String Scalar
USRP N310	RF0 channel: TX/RX port	`"RF0:TX/RX"` (default)
	RF1 channel: TX/RX port	`"RF1:TX/RX"`
	RF2 channel: TX/RX port	`"RF2:TX/RX"`

Radio Device	Supported Antenna Port	Valid String Scalar
	RF3 channel: TX/RX port	`"RF3:TX/RX"`
USRP N320	RF0 channel: TX/RX port	`"RF0:TX/RX"` (default)
	RF1 channel: TX/RX port	`"RF1:TX/RX"`
USRP N321	RF0 channel: TX/RX port	`"RF0:TX/RX"` (default)
	RF1 channel: TX/RX port	`"RF1:TX/RX"`

Data Types: `string`

See Also

Functions
`stopTransmission`

Introduced in R2022a

transmit

Transmit waveform using baseband transmitter or transceiver

Syntax

```
transmit(bba,waveform,mode)
```

Description

`transmit(bba,waveform,mode)` transmits the IQ waveform `waveform` to the air using the specified baseband transmitter or baseband transceiver `bba`. The input `mode` specifies continuous or single-shot transmission. To stop a continuous transmission, call `stopTransmission(bba)`.

Examples

Configure Baseband Transmitter and Transmit Waveform

Create a baseband transmitter object, specifying a radio setup configuration previously saved in the Radio Setup wizard.

```
bbtx = basebandTransmitter("MyRadio")
```

```
bbtx =
  basebandTransmitter with properties:

          RadioGain: 10
    CenterFrequency: 2.4000e+09
         SampleRate: 250000000
           Antennas: "RF0:TX/RX"
```

Set the baseband sample rate and center frequency.

```
bbtx.SampleRate = 122.88e6;
bbtx.CenterFrequency = 2.2e9;
```

Generate random transmit waveform.

```
txWaveform = complex(randn(1000,1),randn(1000,1));
```

Transmit the generated waveform continuously with the radio associated with the baseband transmitter object using the default antenna.

```
transmit(bbtx,txWaveform,"continuous");
```

Stop the continuous transmission after 5 seconds.

```
pause(5);
stopTransmission(bbtx);
```

Configure Baseband Transceiver to Transmit and Capture Data

Create a baseband transceiver object, specifying a radio setup configuration previously saved in the Radio Setup wizard.

```
bbtrx = basebandTransceiver("MyRadio")
```

```
bbtrx =
  basebandTransceiver with properties:

          TransmitRadioGain: 10
    TransmitCenterFrequency: 2.4000e+09
           TransmitAntennas: "RF0:TX/RX"
           CaptureRadioGain: 10
     CaptureCenterFrequency: 2.4000e+09
            CaptureAntennas: "RF0:RX2"
            CaptureDataType: "int16"
       DroppedSamplesAction: "error"
                 SampleRate: 250000000
```

Set the baseband sample rate.

```
bbtrx.SampleRate = 122.88e6;
```

Set the transmit and capture center frequencies.

```
bbtrx.TransmitCenterFrequency = 2.2e9;
bbtrx.CaptureCenterFrequency = 2.2e9;
```

Generate random transmit waveform.

```
txWaveform = complex(randn(1000,1),randn(1000,1));
```

Transmit the generated waveform continuously with the radio associated with the baseband transceiver object using the default transmit antenna.

```
transmit(bbtrx,txWaveform,"continuous");
```

Capture IQ data with the radio associated with the baseband transceiver object using the default capture antenna.

```
[data,~] = capture(bbtrx,milliseconds(3));
```

Stop the continuous transmission after data capture is complete.

```
stopTransmission(bbtrx);
```

Input Arguments

bba — Baseband application

`basebandTransmitter` object | `basebandTransceiver` object

Baseband application, specified as a `basebandTransmitter` object or `basebandTransceiver` object.

Note The first object function call in which you specify this object as an input requires a few extra seconds to load the application onto the hardware.

`waveform` — IQ waveform to transmit

complex-valued column vector | complex-valued matrix

IQ waveform to transmit, specified as one of these options.

- Complex-valued column vector with even number of rows — Use this option to send an IQ waveform on a single transmit antenna.
- Complex-valued matrix with even number of rows — Use this option to send IQ waveforms on multiple transmit antennas. This option applies only when `bba` is a baseband transmitter. The number of antennas specified by the `Antennas` property of the `bba` input must match the number of matrix columns.

The waveform length across all applicable antennas must be relative to the onboard radio memory buffer size. The buffer has a capacity of 2 GB, which amounts to a total of 2^{29} data samples.

Baseband Transceiver Transmit and capture data samples on the baseband transceiver are buffered in the onboard radio memory. Therefore, if the `bba` input is a baseband transceiver, you must also take into account the data capture length that you specify when calling the `capture` object function with the `bba` input.

If you specify a waveform with a `single` or `double` data type, you must scale the transmit data sample values to the range [–1, 1].

Data Types: `int16` | `single` | `double`
Complex Number Support: Yes

`mode` — Transmission mode

`"continuous"` | `"once"`

Transmission mode, specified as one of these options.

- `"continuous"` — The function transmits the waveform `waveform` continuously to the air by repeating the data samples until you call `stopTransmission(bba)`.
- `"once"` — The function transmits the waveform `waveform` once in a single-shot transmission.

Data Types: `string`

See Also

Functions
`stopTransmission`

Introduced in R2022a