2.18 RTDX Module

The RTDX modules manage the real-time data exchange settings.

RTDX Data Declaration Macros

- ❏ RTDX_CreateInputChannel
- ❏ RTDX_CreateOutputChannel

Function Macros

- ❏ RTDX_disableInput
- ❏ RTDX_disableOutput
- ❏ RTDX_enableInput
- ❏ RTDX_enableOutput
- ❏ RTDX_read
- ❏ RTDX_readNB
- ❏ RTDX_sizeofInput
- ❏ RTDX_write

Channel Test Macros

- ❏ RTDX_channelBusy
- ❏ RTDX_isInputEnabled
- ❏ RTDX_isOutputEnabled

Configuration Properties

The following list shows the properties that can be configured in a DSP/BIOS TextConf script, along with their types and default values. For details, see the RTDX Manager Properties and RTDX Object Properties headings. For descriptions of data types, see Section 1.4, *DSP/BIOS TextConf Overview*, page 1-4.

Module Configuration Parameters.

Name	Type	Default (Enum Options)
ENABLERTDX	Bool	true
MODE	EnumString	"JTAG" ("HSRTDX", "Simulator")
RTDXDATASEG	Reference	prog.get("IDRAM")
BUFSIZE	Int16	1032
INTERRUPTMASK	Int16	0x00000000

Instance Configuration Parameters.

Name	Type	Default (Enum Options)
comment	String	"<add comments here>"
channelMode	EnumString	"output" ("input")

Description

The RTDX module provides the data types and functions for:

- ❏ Sending data from the target to the host.
- ❏ Sending data from the host to the target.

Data channels are represented by global structures. A data channel can be used for input or output, but not both. The contents of an input or output structure are not known to the user. A channel structure has two states: enabled and disabled. When a channel is enabled, any data written to the channel is sent to the host. Channels are initially disabled.

The RTDX assembly interface, *rtdx.i*, is a macro interface file that can be used to interface to RTDX at the assembly level.

RTDX Manager Properties

The following target configuration properties can be set for the RTDX module in the RTDX Manager Properties dialog of the Configuration Tool or in a DSP/BIOS TextConf script:

- **Enable Real-Time Data Exchange (RTDX)**. This box should be checked if you want to link RTDX support into your application.

  TextConf Name: ENABLERTDX Type: Bool

  Example:
  ```
  RTDX.ENABLERTDX = true;
  ```

- **RTDX Mode**. Select the port configuration mode RTDX should use to establish communication between the host and target. The default is JTAG for most targets. Set this to simulator if you use a simulator. The HS-RTDX emulation technology is also available. If this property is set incorrectly, a message says “RTDX target application does not match emulation protocol“ when you load the program.

  TextConf Name: MODE Type: EnumString

  Options: "JTAG", "HSRTDX", "Simulator"

  Example:
  ```
  RTDX.MODE = "JTAG";
  ```

- **RTDX Data Segment (.rtdx_data)**. The memory segment used for buffering target-to-host data transfers. The RTDX message buffer and state variables are placed in this segment.

  TextConf Name: RTDXDATASEG Type: Ref

  Example:
  ```
  RTDX.RTDXDATASEG =
      prog.get("myMEM");
  ```

- **RTDX Buffer Size (MADUs)**. The size of the RTDX target-to-host message buffer, in minimum addressable data units (MADUs). The default size is 1032 to accommodate a 1024-byte block and two control words. HST channels using RTDX are limited by this value.

  TextConf Name: BUFSIZE Type: Int16

  Example:
  ```
  RTDX.BUFSIZE = 1032;
  ```

- **RTDX Interrupt Mask**. This mask identifies RTDX clients and protects RTDX critical sections. The mask specifies the interrupts to be temporarily disabled inside RTDX critical sections. This also temporarily disables other RTDX clients and prevents another RTDX function call. See the RTDX on-line help for details.

TextConf Name: INTERRUPTMASK Type: Int16

Example: `RTDX.INTERRUPTMASK = 0x00000000;`

RTDX Object Properties

To create an RTDX object in a configuration script, use the following syntax. The DSP/BIOS TextConf examples that follow assume the object has been created as shown here.

```
var myRtdx = RTDX.create("myRtdx");
```

The following properties can be set for an RTDX object in the RTDX Object Properties dialog of the Configuration Tool or in a DSP/BIOS TextConf script:

- **comment**. Type a comment to identify this RTDX object.

  TextConf Name: comment Type: String

  Example: `myRtdx.comment = "my RTDX";`

- **Channel Mode**. Select output if the RTDX channel handles output from the DSP to the host. Select input if the RTDX channel handles input to the DSP from the host.

  TextConf Name: channelMode Type: EnumString

  Options: "input", "output"

  Example: `myRtdx.channelMode = "output";`

Examples

The `rtdx.xls` example is in the `c:\ti\examples\hostapps\rtdx` folder. (If you installed in a path other than `c:\ti`, substitute your appropriate path.) The examples are described below.

- **Ta_write.asm**. Target to Host transmission example. This example sends 100 consecutive integers starting from 0. In the `rtdx.xls` file, use the h_read VB macro to view data on the host.
- **Ta_read.asm**. Host to target transmission example. This example reads 100 integers. Use the h_write VB macro of the `rtdx.xls` file to send data to the target.
- **Ta_readNB.asm**. Host to target transmission example. This example reads 100 integers. Use the h_write VB macro of the `rtdx.xls` file to send data to the target. This example demonstrates how to use the non-blocking read, RTDX_readNB, function.

Note:

Programs must be linked with C run-time libraries and contain the symbol _main.

RTDX_channelBusy *Return status indicating whether data channel is busy*

C Interface

Syntax

```
int RTDX_channelBusy( RTDX_inputChannel *pichan );
```

Parameters

```
pichan          /* Identifier for the input data channel */
```

Return Value

```
int             /* Status:  0 = Channel is not busy. */
                /* non-zero = Channel is busy. */
```

Assembly Interface

Use C function calling standards.

Reentrant

yes

Description

RTDX_channelBusy is designed to be used in conjunction with RTDX_readNB. The return value indicates whether the specified data channel is currently in use or not. If a channel is busy reading, the test/control flag (TC) bit of status register 0 (STO) is set to 1. Otherwise, the TC bit is set to O.

Constraints and Calling Context

- ❑ RTDX_channelBusy cannot be called by an HWI function.

See Also

RTDX_readNB

RTDX_CreateInputChannel *Declare input channel structure*

C Interface

Syntax RTDX_CreateInputChannel(ichan);

Parameters ichan /* Label for the input channel */

Return Value none

Assembly Interface Use C function calling standards.

Reentrant no

Description This macro declares and initializes to 0, the RTDX data channel for input.

Data channels must be declared as global objects. A data channel can be used either for input or output, but not both. The contents of an input or output data channel are unknown to the user.

A channel can be in one of two states: enabled or disabled. Channels are initialized as disabled.

Channels can be enabled or disabled via a User Interface function. They can also be enabled or disabled remotely from Code Composer or its COM interface.

Constraints and Calling Context

- RTDX_CreateInputChannel cannot be called by an HWI function.

See Also RTDX_CreateOutputChannel

RTDX_CreateOutputChannel *Declare output channel structure*

C Interface

Syntax RTDX_CreateOutputChannel(ochan);

Parameters ochan /* Label for the output channel */

Return Value none

Assembly Interface Use C function calling standards.

Reentrant no

Description This macro declares and initializes the RTDX data channels for output.

Data channels must be declared as global objects. A data channel can be used either for input or output, but not both. The contents of an input or output data channel are unknown to the user.

A channel can be in one of two states: enabled or disabled. Channels are initialized as disabled.

Channels can be enabled or disabled via a User Interface function. They can also be enabled or disabled remotely from Code Composer Studio or its OLE interface.

Constraints and Calling Context

- RTDX_CreateOutputChannel cannot be called by an HWI function.

See Also RTDX_CreateInputChannel

RTDX_disableInput *Disable an input data channel*

C Interface

Syntax void RTDX_disableInput(RTDX_inputChannel *ichan);

Parameters ichan /* Identifier for the input data channel */

Return Value void

Assembly Interface Use C function calling standards.

Reentrant yes

Description A call to a disable function causes the specified input channel to be disabled.

Constraints and Calling Context

- RTDX_disableInput cannot be called by an HWI function.

See Also

RTDX_disableOutput
RTDX_enableInput
RTDX_read

RTDX_disableOutput *Disable an output data channel*

C Interface

Syntax

```
void RTDX_disableOutput( RTDX_outputChannel *ochan );
```

Parameters

```
ochan                    /* Identifier for an output data channel */
```

Return Value

void

Assembly Interface

Use C function calling standards.

Reentrant

yes

Description

A call to a disable function causes the specified data channel to be disabled.

Constraints and Calling Context

- RTDX_disableOutput cannot be called by an HWI function.

See Also

RTDX_disableInput
RTDX_enableOutput
RTDX_read

RTDX_enableInput *Enable an input data channel*

C Interface

Syntax void RTDX_enableInput(RTDX_inputChannel *ichan);

Parameters

```
ochan    /* Identifier for an output data channel */
ichan    /* Identifier for the input data channel */
```

Return Value void

Assembly Interface Use C function calling standards.

Reentrant yes

Description A call to an enable function causes the specified data channel to be enabled.

Constraints and Calling Context

- RTDX_enableInput cannot be called by an HWI function.

See Also

RTDX_disableInput
RTDX_enableOutput
RTDX_read

RTDX_enableOutput *Enable an output data channel*

C Interface

Syntax

```
void RTDX_enableOutput( RTDX_outputChannel *ochan );
```

Parameters

```
ochan                    /* Identifier for an output data channel */
```

Return Value

void

Assembly Interface

Use C function calling standards.

Reentrant

yes

Description

A call to an enable function causes the specified data channel to be enabled.

Constraints and Calling Context

- RTDX_enableOutput cannot be called by an HWI function.

See Also

RTDX_disableOutput
RTDX_enableInput
RTDX_write

RTDX_isInputEnabled *Return status of the input data channel*

C Interface

Syntax

```
RTDX_isInputEnabled( ichan );
```

Parameter

```
ichan          /* Identifier for an input channel. */
```

Return Value

```
0              /* Not enabled. */
non-zero       /* Enabled. */
```

Assembly Interface

Use C function calling standards.

Reentrant

yes

Description

The RTDX_isInputEnabled macro tests to see if an input channel is enabled and sets the test/control flag (TC bit) of status register 0 to 1 if the input channel is enabled. Otherwise, it sets the TC bit to 0.

Constraints and Calling Context

- RTDX_isInputEnabled cannot be called by an HWI function.

See Also

RTDX_isOutputEnabled

RTDX_isOutputEnabled *Return status of the output data channel*

C Interface

Syntax RTDX_isOutputEnabled(ohan);

Parameter

ochan	/* Identifier for an output channel. */

Return Value

0	/* Not enabled. */
non-zero	/* Enabled. *

Assembly Interface Use C function calling standards.

Reentrant yes

Description The RTDX_isOutputEnabled macro tests to see if an output channel is enabled and sets the test/control flag (TC bit) of status register 0 to 1 if the output channel is enabled. Otherwise, it sets the TC bit to 0.

Constraints and Calling Context

- ❏ RTDX_isOutputEnabled cannot be called by an HWI function.

See Also RTDX_isInputEnabled

RTDX_read *Read from an input channel*

C Interface

Syntax

```
int RTDX_read( RTDX_inputChannel *ichan, void *buffer, int bsize );
```

Parameters

```
ichan       /* Identifier for the input data channel */
buffer      /* A pointer to the buffer that receives the data */
bsize       /* The size of the buffer in address units */
```

Return Value

```
> 0               /* The number of address units of data */
                  /* actually supplied in buffer. */
0                 /* Failure. Cannot post read request */
                  /* because target buffer is full. */
RTDX_READ_ERROR   /* Failure. Channel currently busy or
                     not enabled. */
```

Assembly Interface

Use C function calling standards.

Reentrant

yes

Description

RTDX_read causes a read request to be posted to the specified input data channel. If the channel is enabled, RTDX_read waits until the data has arrived. On return from the function, the data has been copied into the specified buffer and the number of address units of data actually supplied is returned. The function returns RTDX_READ_ERROR immediately if the channel is currently busy reading or is not enabled.

When RTDX_read is used, the target application notifies the RTDX Host Library that it is ready to receive data and then waits for the RTDX Host Library to write data to the target buffer. When the data is received, the target application continues execution.

The specified data is to be written to the specified output data channel, provided that channel is enabled. On return from the function, the data has been copied out of the specified user buffer and into the RTDX target buffer. If the channel is not enabled, the write operation is suppressed. If the RTDX target buffer is full, failure is returned.

When RTDX_readNB is used, the target application notifies the RTDX Host Library that it is ready to receive data, but the target application does not wait. Execution of the target application continues immediately. Use RTDX_channelBusy and RTDX_sizeofInput to determine when the RTDX Host Library has written data to the target buffer.

Constraints and Calling Context

- ❏ RTDX_read cannot be called by an HWI function.

See Also

RTDX_channelBusy
RTDX_readNB

RTDX_readNB *Read from input channel without blocking*

C Interface

Syntax

```
int RTDX_readNB( RTDX_inputChannel *ichan, void *buffer, int bsize );
```

Parameters

```
ichan          /* Identifier for the input data channel */
buffer         /* A pointer to the buffer that receives
                  the data */
bsize          /* The size of the buffer in address units */
```

Return Value

```
RTDX_OK        /* Success.*/
0 (zero)       /* Failure. The target buffer is full. */
RTDX_READ_ERROR /*Channel is currently busy reading. */
```

Assembly Interface

Use C function calling standards.

Reentrant

yes

Description

RTDX_readNB is a nonblocking form of the function RTDX_read. RTDX_readNB issues a read request to be posted to the specified input data channel and immediately returns. If the channel is not enabled or the channel is currently busy reading, the function returns RTDX_READ_ERROR. The function returns 0 if it cannot post the read request due to lack of space in the RTDX target buffer.

When the function RTDX_readNB is used, the target application notifies the RTDX Host Library that it is ready to receive data but the target application does not wait. Execution of the target application continues immediately. Use the RTDX_channelBusy and RTDX_sizeofInput functions to determine when the RTDX Host Library has written data into the target buffer.

When RTDX_read is used, the target application notifies the RTDX Host Library that it is ready to receive data and then waits for the RTDX Host Library to write data into the target buffer. When the data is received, the target application continues execution.

Constraints and Calling Context

- ❏ RTDX_readNB cannot be called by an HWI function.

See Also

RTDX_channelBusy
RTDX_read
RTDX_sizeofInput

RTDX_sizeofInput *Return the number of MADUs read from a data channel*

C Interface

Syntax

```
int RTDX_sizeofInput( RTDX_inputChannel *pichan );
```

Parameters

```
pichan    /* Identifier for the input data channel */
```

Return Value

```
int       /* Number of sizeof units of data actually */
          /* supplied in buffer */
```

Assembly Interface

Use C function calling standards.

Reentrant

yes

Description

RTDX_sizeofInput is designed to be used in conjunction with RTDX_readNB after a read operation has completed. The function returns the number of sizeof units actually read from the specified data channel into the accumulator (register A).

Constraints and Calling Context

- RTDX_sizeofInput cannot be called by an HWI function.

See Also

RTDX_readNB

RTDX_write

Write to an output channel

C Interface

Syntax

```
int RTDX_write( RTDX_outputChannel *ochan, void *buffer, int bsize );
```

Parameters

```
ochan     /* Identifier for the output data channel */
buffer    /* A pointer to the buffer containing the data */
bsize     /* The size of the buffer in address units */
```

Return Value

```
int       /* Status: non-zero = Success. 0 = Failure. */
```

Assembly Interface

Use C function calling standards.

Reentrant

yes

Description

RTDX_write causes the specified data to be written to the specified output data channel, provided that channel is enabled. On return from the function, the data has been copied out of the specified user buffer and into the RTDX target buffer. If the channel is not enabled, the write operation is suppressed. If the RTDX target buffer is full, Failure is returned.

Constraints and Calling Context

- RTDX_write cannot be called by an HWI function.

See Also

RTDX_read