

newton\_control

Generated by Doxygen 1.6.1

Tue Feb 11 16:38:59 2020



# Contents

<b>1</b>	<b>Newton Control Console Application and Newton Control API</b>	<b>1</b>
1.1	Newton Control Console Application . . . . .	1
1.2	Table of Contents . . . . .	1
1.3	Command Line Usage . . . . .	1
1.4	Command Line Examples . . . . .	2
1.5	Console Operation . . . . .	3
<b>2</b>	<b>Class Index</b>	<b>5</b>
2.1	Class List . . . . .	5
<b>3</b>	<b>File Index</b>	<b>7</b>
3.1	File List . . . . .	7
<b>4</b>	<b>Class Documentation</b>	<b>9</b>
4.1	revisionID_t Struct Reference . . . . .	9
<b>5</b>	<b>File Documentation</b>	<b>11</b>
5.1	newton_control.h File Reference . . . . .	11
5.1.1	Detailed Description . . . . .	15
5.1.2	Typedef Documentation . . . . .	15
5.1.2.1	adi_loadTargets_e . . . . .	15
5.1.3	Enumeration Type Documentation . . . . .	15
5.1.3.1	adi_attribute_e . . . . .	15
5.1.3.2	adi_data_type_codes_e . . . . .	15
5.1.3.3	adi_errorCodes_e . . . . .	15
5.1.3.4	adi_loadTargets . . . . .	16
5.1.3.5	adi_parameter_codes_e . . . . .	16
5.1.3.6	spiBitRates . . . . .	16
5.1.4	Function Documentation . . . . .	16
5.1.4.1	adi_clear_h2s_valid . . . . .	16

5.1.4.2	<code>adi_error_msg</code>	16
5.1.4.3	<code>adi_get_data</code>	17
5.1.4.4	<code>adi_is_spi_open</code>	17
5.1.4.5	<code>adi_load_hsp</code>	17
5.1.4.6	<code>adi_load_newton_ram</code>	17
5.1.4.7	<code>adi_newton_config</code>	18
5.1.4.8	<code>adi_read_burst</code>	18
5.1.4.9	<code>adi_read_burst_no_incr</code>	18
5.1.4.10	<code>adi_read_register</code>	19
5.1.4.11	<code>adi_send_command</code>	19
5.1.4.12	<code>adi_send_data</code>	19
5.1.4.13	<code>adi_soft_reset</code>	19
5.1.4.14	<code>adi_spi_close</code>	20
5.1.4.15	<code>adi_spi_open</code>	20
5.1.4.16	<code>adi_spi_read_word</code>	20
5.1.4.17	<code>adi_spi_read_word_multiple</code>	20
5.1.4.18	<code>adi_spi_write</code>	21
5.1.4.19	<code>adi_spi_write_word</code>	21
5.1.4.20	<code>adi_spi_write_word_multiple</code>	21
5.1.4.21	<code>adi_unload_hsp</code>	21
5.1.4.22	<code>adi_verify_hsp</code>	22
5.1.4.23	<code>adi_wait_for_h2s_valid</code>	22
5.1.4.24	<code>adi_wait_for_hsp_ready</code>	22
5.1.4.25	<code>adi_wait_for_s2h_not_valid</code>	22
5.1.4.26	<code>adi_write_burst</code>	23
5.1.4.27	<code>adi_write_burst_no_incr</code>	23
5.1.4.28	<code>adi_write_register</code>	23
5.1.5	Variable Documentation	23
5.1.5.1	<code>spiBitRates_e</code>	23

# Chapter 1

## Newton Control Console Application and Newton Control API

### 1.1 Newton Control Console Application

The purpose of the Newton Control Console Application is to provide a program for controlling and debugging Newton operation via the SPI slave interface.

Also provided is an API for building C programs to control Newton

### 1.2 Table of Contents

- [Command Line Usage](#)
- [Console Operation](#)
- [Command Line Examples](#)
  - newton\_control\_api\_sec

### 1.3 Command Line Usage

Usage:

- newton\_control.exe load target file\_name
- newton\_control.exe verify target file\_name
- newton\_control.exe unload target file\_name
- newton\_control.exe spi\_write address data
- newton\_control.exe spi\_read address
- newton\_control.exe reg\_write address data
- newton\_control.exe reg\_read address

- `newton_control.exe reg_dump address word_count`
- `newton_control.exe reg_fill address data word_count`
- `newton_control.exe console`

Where:

- `load` : Loads the contents of the specified file into the target memory.
- `verify` : Compares the contents of the specified file to the contents of the target memory.
- `unload` : Dumps the content of the target memory to the specified file.
- `target` is one of the following:
  - `useq_seq_ram` : Microsequencer Sequence RAM
  - `useq_map_ram` : Microsequencer MAP RAM
  - `useq_wave_ram` : Microsequencer Wave RAM
  - `datapath_ram` : Gain Correction RAM
  - `de_ram` : Dump Engine RAM
  - `lps1_ram` : LPS1
  - `lps2_ram` : LPS2
  - `hsp_rom` : HSP ROM (FPGA only)
  - `hsp_ram` : HSP RAM (FPGA only)
  - `efuse` : eFuse (FPGA only)
- `spi_write` : Write data to the address. For accessing HSP MBOX registers.
- `spi_read` : Read data at address. For accessing HSP MBOX registers.
- `reg_write` : Write data to the Newton register at address
- `reg_read` : Read the Newton register at address
- `reg_dump` : Read and display `word_count` Newton registers starting at address.
- `reg_fill` : Write `word_count` `word_count` Newton registers with data starting at address.
- `console` : Enter console mode

Options:

`--help` Shows this help message.

## 1.4 Command Line Examples

Here are a few examples of the command line interface:

- Loading new HSP ROM and eFuse data:
  - `newton_control.exe load hsp_rom ../../firmware/ADI_Firmware_Drop_0.4/ADI_HSP_ROM_-0.4.4.vhx`

- newton\_control.exe load efuse ../../efuse/hsp\_fuse\_file.vhx
- Writing the HSP mailbox S2H\_MBX\_VALID register:
  - spi\_write 8 1
- Reading the HSP mailbox S2H\_MBX\_FIFO\_CNT register:
  - spi\_read a
- Writing the newton microsequencer gprR0 register:
  - reg\_write 60 01234
- Reading the newton microsequencer gprR0 register:
  - reg\_read 50

## 1.5 Console Operation

Console mode supports the same commands and arguments at the command line interface.

```
{.c}  
FIXME: need new code example
```

See [newton\\_control.h](#) for detailed documentation.





# Chapter 2

## Class Index

### 2.1 Class List

Here are the classes, structs, unions and interfaces with brief descriptions:

<a href="#">revisionID_t</a> . . . . .	9
--	---



## Chapter 3

# File Index

### 3.1 File List

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

[newton\\_control.h](#) (Newton Control API) . . . . . 11



## Chapter 4

# Class Documentation

### 4.1 revisionID\_t Struct Reference

#### Public Attributes

- union {  
    struct {  
        int **major**:8  
        int **minor**:8  
        int **patch**:8  
        int **unused**:8  
    }  
    uint32\_t **value32**  
};

The documentation for this struct was generated from the following file:

- [newton\\_control.h](#)



## Chapter 5

# File Documentation

### 5.1 newton\_control.h File Reference

```
Newton Control API. #include <stdio.h>
#include <stdlib.h>
#include <stdint.h>
#include <stdbool.h>
#include <unistd.h>
#include <string.h>
#include <ctype.h>
#include <time.h>
#include <sys/ioctl.h>
#include <sys/stat.h>
#include <linux/types.h>
#include <linux/spi/spidev.h>
#include <fcntl.h>
#include "newton_addr_cdef.h"
#include "newton_typedefs.h"
#include "newton_memMap.h"
#include "fpga_backdoor_addr_cdef.h"
#include "fpga_backdoor_typedefs.h"
#include "hsp_regs_addr_cdef.h"
#include "hsp_regs_typedefs.h"
#include "json.h"
#include "cutils/str_parms.h"
#include <wiringPi.h>
```

## Classes

- struct [revisionID\\_t](#)

## Defines

- #define **ERROR\_CODE\_BASE** 0x0ad10000
- #define **BIT\_RATE\_COUNT** 6

## Typedefs

- typedef unsigned long long **u64**
- typedef unsigned int **u32**
- typedef unsigned short **u16**
- typedef unsigned char **u08**
- typedef void(\* **callback\_t**)(void)
- typedef struct str\_parms **str\_parms**
- typedef enum [adi\\_loadTargets](#) [adi\\_loadTargets\\_e](#)  
*Newton RAM Targets.*
- typedef enum [adi\\_hspMemBitWidths](#) [adi\\_hspMemBitWidths\\_e](#)

## Enumerations

- enum [adi\\_attribute\\_e](#) { **no\_increment** = 1, **increment** = 2, **addr\_data\_pair** = 4 }  
*HSP mailbox attributes.*
- enum [adi\\_loadTargets](#) {  
    **USEQ\_SEQ\_RAM** = 0, **USEQ\_MAP\_RAM**, **USEQ\_WAVE\_RAM**, **DATAPATH\_RAM**,  
    **DE\_RAM**, **LPS1\_RAM**, **LPS2\_RAM**, **HSP\_ROM**,  
    **HSP\_RAM**, **EFUSE** }  
*Newton RAM Targets.*
- enum [adi\\_hspMemBitWidths](#) { **HSP\_ROM\_WIDTH** = 40, **HSP\_RAM\_WIDTH** = 40, **EFUSE\_WIDTH** = 32 }
- enum [adi\\_errorCodes\\_e](#) {  
    **ADI\_NO\_ERROR** = 0, **ADI\_JSON\_FILE\_NOT\_FOUND** = 0x0ad10001, **ADI\_JSON\_FILE\_OPEN\_ERROR** = 0x0ad10002, **ADI\_JSON\_PARSE\_ERROR** = 0x0ad10003,  
    **ADI\_JSON\_UNEXPECTED\_KEY** = 0x0ad10004, **ADI\_UNEXPECTED\_SPI\_BYTE\_COUNT** = 0x0ad10006, **ADI\_SPI\_DRIVER\_ERROR** = 0x0ad1000a, **ADI\_SPI\_XFER\_ERROR** = 0x0ad1000b,  
    **ADI\_SPI\_BIT\_RATE\_ERROR** = 0x0ad1000c, **ADI\_BAD\_POWER\_STATE** = 0x0ad10011, **ADI\_POWER\_ON\_TIMEOUT** = 0x0ad10012, **ADI\_D32\_FILE\_NOT\_FOUND** = 0x0ad10013,  
    **ADI\_D32\_FILE\_OPEN\_ERROR** = 0x0ad10014, **ADI\_MBI\_FILE\_NOT\_FOUND** = 0x0ad10015, **ADI\_MBI\_FILE\_OPEN\_ERROR** = 0x0ad10016, **ADI\_ERROR\_CODE\_MISSING** = 0x0ad10017,  
    **ADI\_UNEXPECTED\_ARGS** = 0x0ad10018, **ADI\_BAD\_RUNSTALL\_STATE** = 0x0ad10019, **ADI\_H2S\_VALID\_TIMEOUT** = 0x0ad10020, **ADI\_S2H\_NOT\_VALID\_TIMEOUT** = 0x0ad10021  
}



*Error code definitions.*

- enum [adi\\_parameter\\_codes\\_e](#) {  
**REQ\_BUF\_ADDR** = 0xecec0000, **REQ\_BUF\_SIZE** = 0xecec0001, **REQ\_TO\_SEND** = 0xecec0002, **ALLOCATE\_BUFFER** = 0xecec0003,  
**RDY\_TO\_RECV** = 0xecec0004, **START\_ADDRESS** = 0xecec0005, **BUFFER\_SIZE** = 0xecec0006, **BUFFER\_SENT** = 0xecec0007,  
**REQ\_BUF\_ADDR\_SIZE** = 0xecec0008, **START\_ADDRESS\_SIZE** = 0xecec0009, **PARAM\_VALID** = 0xfafa8000 }

*Enums for specifying the Set / Get Parameter name codes.*

- enum [adi\\_data\\_type\\_codes\\_e](#) { **DTYPE\_INT32** = 0, **DTYPE\_UINT32**, **DTYPE\_FLOAT** }

*Enums for specifying the data type codes.*

- enum [spiBitRates](#) {  
**SPI\_BIT\_RATE\_1M** = 1000, **SPI\_BIT\_RATE\_2M** = 2000, **SPI\_BIT\_RATE\_4M** = 4000, **SPI\_BIT\_RATE\_8M** = 8000,  
**SPI\_BIT\_RATE\_12M** = 12000, **SPI\_BIT\_RATE\_16M** = 16000 }

*Enums for specifying valid SPI Clock Rates.*

## Functions

- int [adi\\_spi\\_write](#) (int bytes\_out, u08 \*data\_out, int bytes\_in, u08 \*data\_in)  
*Write a 32-bit word to the Newton over the SPI Interface.*
- int [adi\\_spi\\_read\\_word](#) (u16 address, u16 \*data)  
*Read a 16-bit word from the Newton over the SPI Interface.*
- int [adi\\_spi\\_write\\_word](#) (u16 address, u16 data)  
*Write a 16-bit word to the Newton over the SPI Interface.*
- int [adi\\_spi\\_write\\_word\\_multiple](#) (u16 address, int dataLength, u16 \*dataWritePtr)  
*Write multiple 16-bit words to the Newton over the SPI Interface.*
- int [adi\\_spi\\_read\\_word\\_multiple](#) (u16 address, int dataLength, u16 \*dataReadPtr)  
*Read multiple 16-bit words from the Newton over the SPI Interface.*
- int [adi\\_spi\\_open](#) (int bitRate)  
*Open the SPI Device.*
- int [adi\\_spi\\_close](#) ()  
*Close the SPI Device.*
- bool [adi\\_is\\_spi\\_open](#) ()  
*Test if the SPI Device is open.*
- int [adi\\_wait\\_for\\_hsp\\_ready](#) ()  
*Wait for HSP ready.*

- int [adi\\_send\\_command](#) (u16 command, u16 address, int word\_count, [adi\\_attribute\\_e](#) attribute)  
*Send command to HSP.*
- int [adi\\_clear\\_h2s\\_valid](#) ()  
*Clear H2S valid.*
- int [adi\\_wait\\_for\\_h2s\\_valid](#) ()  
*Wait for H2S Valid from HSP.*
- int [adi\\_wait\\_for\\_s2h\\_not\\_valid](#) ()  
*Wait for S2H not valid from HSP.*
- int [adi\\_send\\_data](#) (int word\_count, u16 \*wr\_data)  
*Send data to HSP.*
- int [adi\\_get\\_data](#) (int word\_count, u16 \*rd\_data)  
*Get data from HSP.*
- int [adi\\_write\\_register](#) (u16 addr, u16 wr\_data)  
*Perform register write through HSP.*
- int [adi\\_write\\_burst](#) (u16 addr, u16 word\_count, u16 \*wr\_data)  
*Perform write burst through HSP with incrementing addresses.*
- int [adi\\_write\\_burst\\_no\\_incr](#) (u16 addr, u16 word\_count, u16 \*wr\_data)  
*Perform write burst through HSP with non-incrementing addresses.*
- int [adi\\_read\\_register](#) (u16 addr, u16 \*rd\_data)  
*Perform register read through HSP.*
- int [adi\\_read\\_burst](#) (u16 addr, u16 word\_count, u16 \*rd\_data)  
*Perform read burst through HSP with incrementing addresses.*
- int [adi\\_read\\_burst\\_no\\_incr](#) (u16 addr, u16 word\_count, u16 \*rd\_data)  
*Perform read burst through HSP with non-incrementing addresses.*
- int [adi\\_load\\_newton\\_ram](#) ([adi\\_loadTargets\\_e](#) loadTarget, char \*fileName)  
*Load the Newton memory image contained in the specified file into the specified HSP memory.*
- int [adi\\_load\\_hsp](#) ([adi\\_loadTargets\\_e](#) loadTarget, char \*fileName)  
*Load the memory image contained in the specified file into the specified HSP memory.*
- int [adi\\_verify\\_hsp](#) ([adi\\_loadTargets\\_e](#) verifyTarget, char \*fileName)  
*Verify that the memory image contained in the specified file matches the content of the specified HSP memory.*
- int [adi\\_unload\\_hsp](#) ([adi\\_loadTargets\\_e](#) unloadTarget, char \*fileName)  
*Unload the memory image to the specified file from the specified HSP memory.*

- int [adi\\_soft\\_reset](#) ()  
*Issue a soft reset to the newton.*
- int [adi\\_newton\\_config](#) (int bitRateOverride)  
*Configure the newton control program.*
- char \* [adi\\_error\\_msg](#) (int returnCode)  
*Return error message string for given error code.*

## Variables

- enum [spiBitRates](#) [spiBitRates\\_e](#)  
*Enums for specifying valid SPI Clock Rates.*
- struct spi\_ioc\_transfer [global\\_tr](#)

### 5.1.1 Detailed Description

Newton Control API.

### 5.1.2 Typedef Documentation

#### 5.1.2.1 typedef enum [adi\\_loadTargets](#) [adi\\_loadTargets\\_e](#)

Newton RAM Targets. Newton RAM targets.

### 5.1.3 Enumeration Type Documentation

#### 5.1.3.1 enum [adi\\_attribute\\_e](#)

HSP mailbox attributes. Newton HSP mailbox attributes.

#### 5.1.3.2 enum [adi\\_data\\_type\\_codes\\_e](#)

Enums for specifying the data type codes. Set / Get Parameter data type codes.

#### 5.1.3.3 enum [adi\\_errorCodes\\_e](#)

Error code definitions. These error codes are used in function return values.

#### Enumerator:

***ADI\_NO\_ERROR*** No error.  
***ADI\_JSON\_FILE\_NOT\_FOUND*** JSON file not found error.  
***ADI\_JSON\_FILE\_OPEN\_ERROR*** JSON file open error.  
***ADI\_JSON\_PARSE\_ERROR*** JSON parsing error.

***ADI\_JSON\_UNEXPECTED\_KEY*** Unexpected JSON Key.  
***ADI\_UNEXPECTED\_SPI\_BYTE\_COUNT*** Unexpected SPI count.  
***ADI\_SPI\_DRIVER\_ERROR*** SPI Driver Error.  
***ADI\_SPI\_XFER\_ERROR*** SPI Transfer Error.  
***ADI\_SPI\_BIT\_RATE\_ERROR*** Unexpected SPI Bit Rate.  
***ADI\_BAD\_POWER\_STATE*** Unexpected power state.  
***ADI\_POWER\_ON\_TIMEOUT*** Timeout waiting for power state == ON.  
***ADI\_D32\_FILE\_NOT\_FOUND*** D32 file not found error.  
***ADI\_D32\_FILE\_OPEN\_ERROR*** D32 file open error.  
***ADI\_MBI\_FILE\_NOT\_FOUND*** MBI file not found error.  
***ADI\_MBI\_FILE\_OPEN\_ERROR*** MBI file open error.  
***ADI\_ERROR\_CODE\_MISSING*** Missing error code.  
***ADI\_UNEXPECTED\_ARGS*** Unexpected arguments.  
***ADI\_BAD\_RUNSTALL\_STATE*** Unexpected RunStall state.  
***ADI\_H2S\_VALID\_TIMEOUT*** H2s valid timeout.  
***ADI\_S2H\_NOT\_VALID\_TIMEOUT*** S2H not valid timeout.

#### 5.1.3.4 enum adi\_loadTargets

Newton RAM Targets. Newton RAM targets.

#### 5.1.3.5 enum adi\_parameter\_codes\_e

Enums for specifying the Set / Get Parameter name codes. Set / Get Parameter name codes.

#### 5.1.3.6 enum spiBitRates

Enums for specifying valid SPI Clock Rates. Valid SPI Clock Rates.

### 5.1.4 Function Documentation

#### 5.1.4.1 int adi\_clear\_h2s\_valid ()

Clear H2S valid. This task clears H2S Valid.

##### Returns:

status value. Indicates success or failure of the function.

#### 5.1.4.2 char\* adi\_error\_msg (int *returnCode*)

Return error message string for given error code.

##### Parameters:

*errorCode* error code value.

**Returns:**

error message string.

**5.1.4.3 int adi\_get\_data (int word\_count, u16 \* rd\_data)**

Get data from HSP. This task gets read data from HSP.

**Parameters:**

*word\_count* Word Count.

*rd\_data* Read Data Array.

**Returns:**

status value. Indicates success or failure of the function.

**5.1.4.4 bool adi\_is\_spi\_open ()**

Test if the SPI Device is open.

**Returns:**

status value. Indicates success or failure of the function.

**5.1.4.5 int adi\_load\_hsp (adi\_loadTargets\_e loadTarget, char \* fileName)**

Load the memory image contained in the specified file into the specified HSP memory. This function performs the following:

- Loads the memory image contained in the specified file into specified HSP memory. This function is only valid for the newton FPGA.

**Parameters:**

*loadTarget* HSP memory to load.

*fileName* name of the file containing the HSP ROM to be loaded.

**Returns:**

status value. Indicates success or failure of the function.

**5.1.4.6 int adi\_load\_newton\_ram (adi\_loadTargets\_e loadTarget, char \* fileName)**

Load the Newton memory image contained in the specified file into the specified HSP memory. This function performs the following:

- Loads the memory image contained in the specified file into specified Newton memory.

**Parameters:**

*loadTarget* Newton memory to load.

*fileName* name of the file containing the Newton RAM to be loaded.

**Returns:**

status value. Indicates success or failure of the function.

**5.1.4.7 int adi\_newton\_config (int bitRateOverride)**

Configure the newton control program.

**Parameters:**

*bitRateOverride* use this SPI bit rate instead of the default.

**Returns:**

status value. Indicates success or failure of the function.

**5.1.4.8 int adi\_read\_burst (u16 addr, u16 word\_count, u16 \* rd\_data)**

Perform read burst through HSP with incrementing addresses. This task performs read burst through HSP with incrementing addresses.

**Parameters:**

*addr* Read address.

*word\_count* Word Count.

*rd\_data* Read Data Array.

**Returns:**

status value. Indicates success or failure of the function.

**5.1.4.9 int adi\_read\_burst\_no\_incr (u16 addr, u16 word\_count, u16 \* rd\_data)**

Perform read burst through HSP with non-incrementing addresses. This task performs read burst through HSP with non-incrementing addresses.

**Parameters:**

*addr* Read address.

*word\_count* Word Count.

*rd\_data* Read Data Array.

**Returns:**

status value. Indicates success or failure of the function.

**5.1.4.10 int adi\_read\_register (u16 addr, u16 \* rd\_data)**

Perform register read through HSP. This task performs register read through HSP.

**Parameters:**

*addr* Read address.

*rd\_data* Read Data.

**Returns:**

status value. Indicates success or failure of the function.

**5.1.4.11 int adi\_send\_command (u16 command, u16 address, int word\_count, adi\_attribute\_e attribute)**

Send command to HSP. This task sends the command to the HSP.

**Parameters:**

*command* Command Code.

*address* Address.

*word\_count* Word Count.

*attribute* Attributes.

**Returns:**

status value. Indicates success or failure of the function.

**5.1.4.12 int adi\_send\_data (int word\_count, u16 \* wr\_data)**

Send data to HSP. This task sends the data to the HSP.

**Parameters:**

*wr\_data* Write Data Array.

*word\_count* Word Count.

**Returns:**

status value. Indicates success or failure of the function.

**5.1.4.13 int adi\_soft\_reset ()**

Issue a soft reset to the newton. Issue a soft reset.

**Parameters:**

*channel\_id* the channel ID of the to be written.

**Returns:**

status value. Indicates success or failure of the function.

status value. Indicates success or failure of the function.

**5.1.4.14 int adi\_spi\_close ()**

Close the SPI Device.

**Returns:**

status value. Indicates success or failure of the function.

**5.1.4.15 int adi\_spi\_open (int *bitRate*)**

Open the SPI Device.

**Parameters:**

*bitRate* the bit rate that the SPI interface should operate at.

**Returns:**

status value. Indicates success or failure of the function.

**5.1.4.16 int adi\_spi\_read\_word (u16 *address*, u16 \* *data*)**

Read a 16-bit word from the Newton over the SPI Interface.

**Parameters:**

*address* the address of the 32-bit word.

*data32* a pointer to the data read from the Newton.

**Returns:**

status value. Indicates success or failure of the function.

**5.1.4.17 int adi\_spi\_read\_word\_multiple (u16 *address*, int *dataLength*, u16 \* *dataReadPtr*)**

Read multiple 16-bit words from the Newton over the SPI Interface.

**Parameters:**

*address* the address of the data to be read.

*dataLength* the number of words to be read.

*dataPtr* a pointer to the data read from the Newton.

**Returns:**

status value. Indicates success or failure of the function.



**5.1.4.18 int adi\_spi\_write (int bytes\_out, u08 \* data\_out, int bytes\_in, u08 \* data\_in)**

Write a 32-bit word to the Newton over the SPI Interface.

**Parameters:**

*bytes\_out* the number of bytes to be sent over the SPI interface to the Newton.

*data\_out* the data to be sent over the SPI interface to the Newton.

*bytes\_in* the number of bytes to be received from the Newton over the SPI interface.

*data\_in* the data to be received from the Newton over the SPI interface.

**Returns:**

status value. Indicates success or failure of the function.

**5.1.4.19 int adi\_spi\_write\_word (u16 address, u16 data)**

Write a 16-bit word to the Newton over the SPI Interface.

**Parameters:**

*bytes\_out* the number of bytes to be sent over the SPI interface to the Newton.

*data\_out* the data to be sent over the SPI interface to the Newton.

*bytes\_in* the number of bytes to be received from the Newton over the SPI interface.

*data\_in* the data to be received from the Newton over the SPI interface.

**Returns:**

status value. Indicates success or failure of the function.

**5.1.4.20 int adi\_spi\_write\_word\_multiple (u16 address, int dataLength, u16 \* dataWritePtr)**

Write multiple 16-bit words to the Newton over the SPI Interface.

**Parameters:**

*address* the address of the data to be written.

*dataLength* the number of words to be written.

*dataPtr* a pointer to the data written to the Newton.

**Returns:**

status value. Indicates success or failure of the function.

**5.1.4.21 int adi\_unload\_hsp (adi\_loadTargets\_e unloadTarget, char \* fileName)**

Unload the memory image to the specified file from the specified HSP memory. This function performs the following:

- Unloads the memory to the specified file from specified HSP memory. This function is only valid for the newton FPGA.

**Parameters:**

*unloadTarget* HSP memory to unload.

*fileName* name of the file containing the HSP ROM to be loaded.

**Returns:**

status value. Indicates success or failure of the function.

**5.1.4.22 int adi\_verify\_hsp (adi\_loadTargets\_e verifyTarget, char \*fileName)**

Verify that the memory image contained in the specified file matches the content of the specified HSP memory. This function performs the following:

- Verifies that the memory image contained in the specified file matches the contents of the specified HSP memory. This function is only valid for the newton FPGA.

**Parameters:**

*verifyTarget* HSP memory to verify.

*fileName* name of the file containing the HSP ROM to be verified.

**Returns:**

status value. Indicates success or failure of the function.

**5.1.4.23 int adi\_wait\_for\_h2s\_valid ()**

Wait for H2S Valid from HSP. This task polls the h2s valid bit until it is set.

**Returns:**

status value. Indicates success or failure of the function.

**5.1.4.24 int adi\_wait\_for\_hsp\_ready ()**

Wait for HSP ready. This task waits for the HSP to be ready.

**Returns:**

status value. Indicates success or failure of the function.

**5.1.4.25 int adi\_wait\_for\_s2h\_not\_valid ()**

Wait for S2H not valid from HSP. This task polls the S2h valid bit until it is cleared.

**Returns:**

status value. Indicates success or failure of the function.

#### 5.1.4.26 int adi\_write\_burst (u16 addr, u16 word\_count, u16 \* wr\_data)

Perform write burst through HSP with incrementing addresses. This task performs write burst through HSP with incrementing addresses.

##### Parameters:

*addr* Write address.  
*word\_count* Word Count.  
*wr\_data* Write Data Array.

##### Returns:

status value. Indicates success or failure of the function.

#### 5.1.4.27 int adi\_write\_burst\_no\_incr (u16 addr, u16 word\_count, u16 \* wr\_data)

Perform write burst through HSP with non-incrementing addresses. This task performs write burst through HSP with non-incrementing addresses.

##### Parameters:

*addr* Write address.  
*word\_count* Word Count.  
*wr\_data* Read Data Array.

##### Returns:

status value. Indicates success or failure of the function.

#### 5.1.4.28 int adi\_write\_register (u16 addr, u16 wr\_data)

Perform register write through HSP. This task performs register write through HSP.

##### Parameters:

*addr* Write address.  
*wr\_data* Write Data.

##### Returns:

status value. Indicates success or failure of the function.

### 5.1.5 Variable Documentation

#### 5.1.5.1 enum spiBitRates spiBitRates\_e

Enums for specifying valid SPI Clock Rates. Valid SPI Clock Rates.

# Index

ADI\_BAD\_POWER\_STATE  
    newton\_control.h, [16](#)

ADI\_BAD\_RUNSTALL\_STATE  
    newton\_control.h, [16](#)

ADI\_D32\_FILE\_NOT\_FOUND  
    newton\_control.h, [16](#)

ADI\_D32\_FILE\_OPEN\_ERROR  
    newton\_control.h, [16](#)

ADI\_ERROR\_CODE\_MISSING  
    newton\_control.h, [16](#)

ADI\_H2S\_VALID\_TIMEOUT  
    newton\_control.h, [16](#)

ADI\_JSON\_FILE\_NOT\_FOUND  
    newton\_control.h, [15](#)

ADI\_JSON\_FILE\_OPEN\_ERROR  
    newton\_control.h, [15](#)

ADI\_JSON\_PARSE\_ERROR  
    newton\_control.h, [15](#)

ADI\_JSON\_UNEXPECTED\_KEY  
    newton\_control.h, [15](#)

ADI\_MBI\_FILE\_NOT\_FOUND  
    newton\_control.h, [16](#)

ADI\_MBI\_FILE\_OPEN\_ERROR  
    newton\_control.h, [16](#)

ADI\_NO\_ERROR  
    newton\_control.h, [15](#)

ADI\_POWER\_ON\_TIMEOUT  
    newton\_control.h, [16](#)

ADI\_S2H\_NOT\_VALID\_TIMEOUT  
    newton\_control.h, [16](#)

ADI\_SPI\_BIT\_RATE\_ERROR  
    newton\_control.h, [16](#)

ADI\_SPI\_DRIVER\_ERROR  
    newton\_control.h, [16](#)

ADI\_SPI\_XFER\_ERROR  
    newton\_control.h, [16](#)

ADI\_UNEXPECTED\_ARGS  
    newton\_control.h, [16](#)

ADI\_UNEXPECTED\_SPI\_BYTE\_COUNT  
    newton\_control.h, [16](#)

adi\_attribute\_e  
    newton\_control.h, [15](#)

adi\_clear\_h2s\_valid  
    newton\_control.h, [16](#)

adi\_data\_type\_codes\_e  
    newton\_control.h, [15](#)

adi\_error\_msg  
    newton\_control.h, [16](#)

adi\_errorCodes\_e  
    newton\_control.h, [15](#)

adi\_get\_data  
    newton\_control.h, [17](#)

adi\_is\_spi\_open  
    newton\_control.h, [17](#)

adi\_load\_hsp  
    newton\_control.h, [17](#)

adi\_load\_newton\_ram  
    newton\_control.h, [17](#)

adi\_loadTargets  
    newton\_control.h, [16](#)

adi\_loadTargets\_e  
    newton\_control.h, [15](#)

adi\_newton\_config  
    newton\_control.h, [18](#)

adi\_parameter\_codes\_e  
    newton\_control.h, [16](#)

adi\_read\_burst  
    newton\_control.h, [18](#)

adi\_read\_burst\_no\_incr  
    newton\_control.h, [18](#)

adi\_read\_register  
    newton\_control.h, [18](#)

adi\_send\_command  
    newton\_control.h, [19](#)

adi\_send\_data  
    newton\_control.h, [19](#)

adi\_soft\_reset  
    newton\_control.h, [19](#)

adi\_spi\_close  
    newton\_control.h, [19](#)

adi\_spi\_open  
    newton\_control.h, [20](#)

adi\_spi\_read\_word  
    newton\_control.h, [20](#)

adi\_spi\_read\_word\_multiple  
    newton\_control.h, [20](#)

adi\_spi\_write  
    newton\_control.h, [20](#)

adi\_spi\_write\_word  
    newton\_control.h, [21](#)

- adi\_spi\_write\_word\_multiple
  - newton\_control.h, 21
- adi\_unload\_hsp
  - newton\_control.h, 21
- adi\_verify\_hsp
  - newton\_control.h, 22
- adi\_wait\_for\_h2s\_valid
  - newton\_control.h, 22
- adi\_wait\_for\_hsp\_ready
  - newton\_control.h, 22
- adi\_wait\_for\_s2h\_not\_valid
  - newton\_control.h, 22
- adi\_write\_burst
  - newton\_control.h, 22
- adi\_write\_burst\_no\_incr
  - newton\_control.h, 23
- adi\_write\_register
  - newton\_control.h, 23
- newton\_control.h
  - ADI\_BAD\_POWER\_STATE, 16
  - ADI\_BAD\_RUNSTALL\_STATE, 16
  - ADI\_D32\_FILE\_NOT\_FOUND, 16
  - ADI\_D32\_FILE\_OPEN\_ERROR, 16
  - ADI\_ERROR\_CODE\_MISSING, 16
  - ADI\_H2S\_VALID\_TIMEOUT, 16
  - ADI\_JSON\_FILE\_NOT\_FOUND, 15
  - ADI\_JSON\_FILE\_OPEN\_ERROR, 15
  - ADI\_JSON\_PARSE\_ERROR, 15
  - ADI\_JSON\_UNEXPECTED\_KEY, 15
  - ADI\_MBI\_FILE\_NOT\_FOUND, 16
  - ADI\_MBI\_FILE\_OPEN\_ERROR, 16
  - ADI\_NO\_ERROR, 15
  - ADI\_POWER\_ON\_TIMEOUT, 16
  - ADI\_S2H\_NOT\_VALID\_TIMEOUT, 16
  - ADI\_SPI\_BIT\_RATE\_ERROR, 16
  - ADI\_SPI\_DRIVER\_ERROR, 16
  - ADI\_SPI\_XFER\_ERROR, 16
  - ADI\_UNEXPECTED\_ARGS, 16
  - ADI\_UNEXPECTED\_SPI\_BYTE\_COUNT, 16
- newton\_control.h, 11
  - adi\_attribute\_e, 15
  - adi\_clear\_h2s\_valid, 16
  - adi\_data\_type\_codes\_e, 15
  - adi\_error\_msg, 16
  - adi\_errorCodes\_e, 15
  - adi\_get\_data, 17
  - adi\_is\_spi\_open, 17
  - adi\_load\_hsp, 17
  - adi\_load\_newton\_ram, 17
  - adi\_loadTargets, 16
  - adi\_loadTargets\_e, 15
  - adi\_newton\_config, 18
  - adi\_parameter\_codes\_e, 16
  - adi\_read\_burst, 18
  - adi\_read\_burst\_no\_incr, 18
  - adi\_read\_register, 18
  - adi\_send\_command, 19
  - adi\_send\_data, 19
  - adi\_soft\_reset, 19
  - adi\_spi\_close, 19
  - adi\_spi\_open, 20
  - adi\_spi\_read\_word, 20
  - adi\_spi\_read\_word\_multiple, 20
  - adi\_spi\_write, 20
  - adi\_spi\_write\_word, 21
  - adi\_spi\_write\_word\_multiple, 21
  - adi\_unload\_hsp, 21
  - adi\_verify\_hsp, 22
  - adi\_wait\_for\_h2s\_valid, 22
  - adi\_wait\_for\_hsp\_ready, 22
  - adi\_wait\_for\_s2h\_not\_valid, 22
  - adi\_write\_burst, 22
  - adi\_write\_burst\_no\_incr, 23
  - adi\_write\_register, 23
  - spiBitRates, 16
  - spiBitRates\_e, 23
- revisionID\_t, 9
- spiBitRates
  - newton\_control.h, 16
- spiBitRates\_e
  - newton\_control.h, 23