HPDI32

High Speed 32-bit Digital I/O,
with 39-bits of GPIO

PCI-HPDI32A
PCI64-HPDI32AL
PMC-HPDI32A
PMC64-HPDI32ALT

GPIO Library Reference Manual

Manual Revision: October 7, 2014

General Standards Corporation
8302A Whitesburg Drive
Huntsville, AL 35802
Phone: (256) 880-8787
Fax: (256) 880-8788
URL: http://www.generalstandards.com
E-mail: sales@generalstandards.com
E-mail: support@generalstandards.com

Preface

Additional copies of this manual or other literature may be obtained from:

General Standards Corporation
8302A Whitesburg Dr.
Huntsville, Alabama 35802
Phone: (256) 880-8787
FAX: (256) 880-8788
URL: http://www.generalstandards.com/
E-mail: sales@generalstandards.com

General Standards Corporation makes no warranty of any kind with regard to this material, including, but not limited to, the implied warranties of merchantability and fitness for a particular purpose. Although extensive editing and reviews are performed before release to ECO control, **General Standards Corporation** assumes no responsibility for any errors that may exist in this document. No commitment is made to update or keep current the information contained in this document.

General Standards Corporation does not assume any liability arising out of the application or use of any product or circuit described herein, nor is any license conveyed under any patent rights or any rights of others.

General Standards Corporation assumes no responsibility for any consequences resulting from omissions or errors in this manual or from the use of information contained herein.

General Standards Corporation reserves the right to make any changes, without notice, to this product to improve reliability, performance, function, or design.

The Purchaser of this software may use or modify in source form the subject software, but not to re-market or distribute it to outside agencies or separate internal company divisions. The software, however, may be embedded in the Purchaser's distributed software. In the event the Purchaser's customers require the software source code, then they would have to purchase their own copy of the software.

General Standards Corporation makes no warranty of any kind with regard to this software, including, but not limited to, the implied warranties of merchantability and fitness for a particular purpose and makes this software available solely on an "as-is" basis. **General Standards Corporation** reserves the right to make changes in this software without reservation and without notification to its users.

Table of Contents

1. Introduction

This document provides information on the HPDI32 GPIO Library, which is a library designed to facilitate use of the GPIO capabilities built into the HPDI32.

1.1. Purpose

The purpose of this document is to give a complete description of the HPDI32 GPIO Library interface.

1.2. Acronyms

The following is a list of commonly occurring acronyms used throughout this document.

Acronyms	Description
FIFO	First In/First Out memory
GPIO	General Purpose Input/Output
GSC	General Standards Corporation
PCI	Peripheral Component Interconnect

1.3. Definitions

The following is a list of commonly occurring terms used throughout this document.

Term	Definition
Application	Application means the user mode process, which runs in the user space with user mode privileges.
Driver	Driver means the executable program providing direct access to the HPDI32 hardware.
HPDI32	This is used as a general reference to any device supported by this library.

1.4. Software Overview

The HPDI32 GPIO Library is a statically linked library providing a GPIO centric interface to the HPDI32 API that is part of the HPDI32 SDK. The library is provided in source form and must be built before being used. The library is a thin software layer that sits between an HPDI32 application and the HPDI32 API. The interface provided by the library is GPIO specific. The library exists in parallel with the interface provided by the HPDI32 SDK API. Applications are free to use the GPIO library interface and the HPDI32 SDK API at will.

1.5. Hardware Overview

The HPDI32 is a high-performance 32-bit parallel digital I/O interface board. The host side connection is PCI based and is either 32-bit or 64-bit according to the model ordered. The external I/O interface varies per model ordered. The board is capable of transmitting or receiving data at up to 200 Mbytes per second over an external I/O interface, depending on the model ordered. Onboard transmit and receive FIFOs of up to 128k data values each, buffer transfer data between the PCI bus and the cable interface. This allows the HPDI32 to maintain maximum bursts on the cable interface (at least up to the depth of the FIFOs) independent of the PCI bus interface. The onboard FIFOs can also be used to buffer data between the cable interface and the PCI bus to maintain a sustained data throughput for real-time applications.

The HPDI32 offers a half-duplex external I/O interface. The board can either transmit or receive data, but it cannot do both simultaneously. In addition to the 32 synchronous data I/O lines, the external interface includes a set of configurable flow control signals. These can be configured as discrete I/O, as can the 32 data lines. The board accommodates a wide range of applications. This range extends from sending or receiving relatively small blocks of data on demand, to sending or receiving large continuous streams of data for an extended period. Once a data link is

established, the data is transferred to/from host memory by simply writing to or reading from the onboard FIFOs. The board has an advanced PCI interface engine, which provides for increased data throughput via DMA.

1.6. Reference Material

The following reference material may be of particular benefit in using the HPDI32 and this driver. The specifications provide the information necessary for an in depth understanding of the specialized features implemented on this board.

- The applicable *HPDI32 User Manual* from General Standards Corporation.
- The *PCI9080 PCI Bus Master Interface Chip* data handbook from PLX Technology, Inc. (for 32-bit PCI interface boards) *
- The *PCI9656 PCI Bus Master Interface Chip* data handbook from PLX Technology, Inc. (for 64-bit PCI interface boards) *

* PLX data books are available from PLX at the following location.

PLX Technology Inc.
870 Maude Avenue
Sunnyvale, California 94085 USA
Phone: 1-800-759-3735
WEB: http://www.plxtech.com/

2. GPIO Capabilities

The HPDI32 has two banks of lines usable as GPIO. The first bank consists of seven control lines. In their default functionality these lines provide flow control capabilities when transmitting or receiving data over the 32 data lines. The second bank consists of the 32 data lines. In their default capacity these lines carry the data that is transferred during transmit or receive operations.

2.1. Control Lines

The HPDI32 cable interface includes seven control lines. In their default configuration each line supports flow control for data transmission and/or reception operations. Alternatively, each line can independently be configured as General Purpose Input or Output. Configuring any or all lines as GPIO does not interfere with data transmission or reception. GPIO and data transfer can both be used simultaneously. However, configuring any control line as GPIO reduces the flow control capabilities according to each line's flow control functionality. The following table summarizes the capabilities of the seven control lines. Refer to the board user manual for additional information.

NOTE: Control lines which are not to be used should be configured as GPIO inputs.

NOTE: Most HPDI32 boards support the use of all seven control lines for GPIO. Older boards or firmware may support only six GPIO lines or even none.

GPIO #	Flow Control	Flow Control Operation
GPIO 0	Line Valid	This signal is driven by the transmitting device and is a secondary flow control signal. It is driven high when data is being transmitted and is driven low when data is not being transmitted.
GPIO 1	Status Valid	This signal is driven by the transmitting device and is a secondary flow control signal. It is driven high when status data is being transmitted and is driven low when status data is not being transmitted.
GPIO 2	Rx Ready	This signal is driven by the receiving device. It is driven low to pause data transfer and is driven high to resume data transfer. This signal is used for remote throttling of the transfer process. To use remote throttling it must be enabled by software.
GPIO 3	Tx Data Ready	This signal is driven by the transmitting device. It is driven low when Tx Data is present and is driven high when no Tx Data is present.
GPIO 4	Tx Enabled	This is an output only signal. It is driven high when the transmitter is enabled and is driven low when the transmitter is disabled. *
GPIO 5	Rx Enabled	This is an output only signal. It is driven high when the receiver is enabled and is driven low when the receiver is disabled. *
GPIO 6	Frame Valid	This signal is driven by the transmitting device and is the primary flow control signal. It is driven high when data is being transmitted and is driven low when data is not being transmitted.

* The signal can be configured so that it is driven when high and tri-stated when low.

2.2. Data Lines

The 32 data lines used for parallel data transfers can also be used for GPIO. For either use, all 32 lines operate identically. The individual bits are not independently configurable. The HPDI32 GPIO Library provides two implementations, depending on the board's firmware capabilities. Both are described below.

2.2.1. Explicit GPIO Support

More recent firmware implementations include explicit support for using the 32 data lines as GPIO. If bit D9 of the Features Register is set, then the firmware includes this support. This support includes the Auxiliary Control Register, the GPIO Input Register and the GPIO Output Register. When this support is present, the 32 data lines can be used as GPIO independent of how any of the control lines are configured.

2.2.2. Simulated GPIO Support

With older firmware implementations, use of the 32 data lines as GPIO is possible, though indirectly. This operation is accomplished using the transmitter for GPIO output and the receiver for GPIO input, though each requires suitable control line configurations.

2.2.2.1. Simulated GPIO Output

GPIO output functionality is simulated by writing the GPIO output pattern to the transmit FIFO and immediately clocked it to the cable interface. That is essentially all there is to it. For output simulation purposes the library configures the device to facilitate GPIO use. This configuration should not be altered by applications. Applications are free to use the control lines as described in the table below.

GPIO #	Flow Control	Transmit Use Options
None	Tx Clock	The clock is active while the transmitter is enabled, and is otherwise not configurable.
GPIO 0	Line Valid	This signal can be configured and used as desired, without restriction.
GPIO 1	Status Valid	This signal can be configured and used as desired, without restriction.
GPIO 2	Rx Ready	This signal can be configured and used as GPIO. No attempt should be made to utilize the remote throttling feature.
GPIO 3	Tx Data Ready	This signal can be configured and used as desired, without restriction.
GPIO 4	Tx Enabled	This signal can be configured and used as desired, without restriction. *
GPIO 5	Rx Enabled	This signal can be configured and used as desired, without restriction. *
GPIO 6	Frame Valid	This signal can be configured and used as desired, without restriction.

* In its flow control configuration, this signal is also configured to be driven when high and tri-stated when low.

2.2.2.2. Simulated GPIO Input

GPIO input functionality is simulated by clocking in what currently appears at the cable interface. This is accomplished by resetting the Rx FIFO, waiting for data to appear in the FIFO, and then obtaining the GPIO input pattern by reading from the FIFO Data Register. (At the waiting stage, the library will wait for only a short period (see note) before concluding that the necessary cable interface signals have not been provided.) For input simulation purposes the library configures the device to facilitate GPIO use. This configuration should not be altered by applications. Using the receiver for GPIO input requires some functionality not provided by the HPDI32 itself as described in the table below.

GPIO #	Flow Control	Transmit Use Options
None	Rx Clock	A high speed clock must be provided by an external source.
GPIO 0	Line Valid	This signal should be configured as GPIO. If it cannot be configured as GPIO, then it should be driven high from an external source.
GPIO 1	Status Valid	This signal should be configured as GPIO. If it cannot be configured as GPIO, then it should be driven high from an external source.
GPIO 2	Rx Ready	This signal can be configured and used as desired, without restriction.
GPIO 3	Tx Data Ready	This signal should be configured as GPIO. If it cannot be configured as GPIO, then it should be driven low from an external source.
GPIO 4	Tx Enabled	This signal can be configured and used as desired, without restriction. *
GPIO 5	Rx Enabled	This signal can be configured and used as desired, without restriction. *
GPIO 6	Frame Valid	This signal should be configured as GPIO. If it cannot be configured as GPIO, then it should be driven high from an external source.

* In its flow control configuration, this signal is also configured to be driven when high and tri-stated when low.

> **NOTE:** The HPDI32 GPIO Library will wait at most 20ms or 20 system timer ticks, whichever is longer, while waiting for data to be clocked in at the cable interface. If no data is received during that time, then the GPIO input read request will return a failure status.

3. GPIO Library Interface

The GPIO Library interface is defined in the header file is `hpdi32_gpio.h`. The interface consists of data types, macros and functions.

> **NOTE**: Attempts to exercise a nonexistent capability will generally result in return of the `GSC_UNSUPPORTED_FUNCTION` status value.

3.1. Data Types

The GPIO Library interface uses a limited number of data types. The table below describes those used.

Macro	Description
`U8`	This is an unsigned 8-bit value. It is used by the library to report device capabilities, to express GPIO configuration, and to convey GPIO data relating to the cable interface control lines.
`U32`	This is an unsigned 32-bit value. It's most common use is return status from function calls. For the list of possible return values refer to the `gsc_status_t` enumeration from the header file `gsc_common.h`, which is located in the SDK's driver directory. This type is also used for the GPIO pattern passed across the 32-bit data lines.
`void*`	This type is used for the device handle used by the API Library for accessing HPDI32 boards.

3.2. Macros

3.2.1. Capability Macros

The interface reports the device's GPIO capabilities using the below list of macros. These macros are combined bitwise to report the device's overall GPIO capabilities, which are returned by calling the function `hpdi32_gpio_dev_cap_get()` (see section 3.4.1, page 9).

Macro	Description
`HPDI32_GPIO_CAP_CTRL_0_5`	The control lines support GPIO 0 to GPIO 5 only.
`HPDI32_GPIO_CAP_CTRL_0_6`	The control lines support GPIO 0 to GPIO 6.
`HPDI32_GPIO_CAP_CTRL_RX`	The control lines support GPIO input.
`HPDI32_GPIO_CAP_CTRL_TX`	The control lines support GPIO output.
`HPDI32_GPIO_CAP_DATA_EXP`	The data lines provide explicit GPIO support.
`HPDI32_GPIO_CAP_DATA_SIM`	The data lines provide simulated GPIO support.
`HPDI32_GPIO_CAP_DATA_RX`	The data lines support GPIO input.
`HPDI32_GPIO_CAP_DATA_TX`	The data lines support GPIO output.

3.2.2. Configuration Macros

The interface uses the below list of macros both to report the device's current GPIO configuration and to change the device's GPIO configuration. These macros are combined bitwise to enable the designated lines and to set the designated lines as outputs. For device configuration refer to functions `hpdi32_gpio_config_get()` (section 3.5.1, page 10) and `hpdi32_gpio_config_set()` (section 3.5.2, page 10).

Macro	Description
`HPDI32_GPIO_CFG_0`	This refers to the GPIO configuration of GPIO 0 provided via the control lines.
`HPDI32_GPIO_CFG_1`	This refers to the GPIO configuration of GPIO 1 provided via the control lines.
`HPDI32_GPIO_CFG_2`	This refers to the GPIO configuration of GPIO 2 provided via the control lines.
`HPDI32_GPIO_CFG_3`	This refers to the GPIO configuration of GPIO 3 provided via the control lines.
`HPDI32_GPIO_CFG_4`	This refers to the GPIO configuration of GPIO 4 provided via the control lines.
`HPDI32_GPIO_CFG_5`	This refers to the GPIO configuration of GPIO 5 provided via the control lines.

HPDI32_GPIO_CFG_6	This refers to the GPIO configuration of GPIO 6 provided via the control lines.
HPDI32_GPIO_CFG_D32	This refers to the GPIO configuration of the 32 data lines.

3.3. Library Entry/Exit Functions

The following functions relate to use of the GPIO Library as a whole. All other GPIO Library calls should be made after the library initialize call and before the library release call.

3.3.1. hpdi32_gpio_lib_init()

This function is required in order to prepare the GPIO Library for operation. This must be the first call to the library.

Prototype

```
void hpdi32_gpio_lib_init(void);
```

Argument	Description
None	The function has no arguments.

Return Value	Description
None	No value is returned.

3.3.2. hpdi32_gpio_lib_release()

This function is required in order to release resources used by the GPIO Library when it will no longer be used. This must be the last call to the library.

Prototype

```
void hpdi32_gpio_lib_release(void);
```

Argument	Description
None	The function has no arguments.

Return Value	Description
None	No value is returned.

3.4. Device Entry/Exit Functions

The following functions relate to use of the GPIO Library for each device to be accessed. All other GPIO Library device access calls should be made after the device capabilities call and before the device release call.

3.4.1. hpdi32_gpio_dev_cap_get()

This function is the entry point to retrieving the GPIO capability information for the device being accessed. This should be the first call to the library for the referenced device.

Prototype

```
U32 hpdi32_gpio_dev_cap_get(void* dev, U8* cap);
```

Argument	Description
dev	This is the device handle returned from the API Library's hpdi32_open() function.
cap	The device capabilities are recorded here. If a bit is set, then the corresponding capability is

	present. If a bit is clear, then the capability is absent. *

* Refer to the capability macros in section 3.2.1 on page 8.

Return Value	Description
GSC_SUCCESS	This is returned when no errors are encountered.
else	Some other GSC_XXX macro value is returned in the event of an error.

3.4.2. hpdi32_gpio_dev_release()

This function is required in order to release resources used by the GPIO Library for the given device when the device will no longer be access via the GPIO Library. This should be the last call to the library for the referenced device.

Prototype

```
void hpdi32_gpio_dev_release(void* dev);
```

Argument	Description
dev	This is the device handle returned from the API Library's hpdi32_open() function.

Return Value	Description
None	No value is returned.

3.5. Device Configuration Functions

The following functions are used to either retrieve or adjust the board's current configuration.

3.5.1. hpdi32_gpio_config_get()

This function is the entry point to retrieving the current GPIO configuration for the device being accessed. This should be the first call to the library for the referenced device.

Prototype

```
U32 hpdi32_gpio_config_get(void* dev, U8* cfg_en, U8* cfg_dir);
```

Argument	Description
dev	This is the device handle returned from the API Library's hpdi32_open() function.
cfg_en	The library reports the enable configuration here. If a bit is set, then the corresponding GPIO resource is enabled. If a bit is clear, then it is disabled. *
cfg_dir	The library reports the GPIO direction configuration here. If a bit is set, then the corresponding GPIO resource is an output. If a bit is clear, then it is an input. *

* Refer to the configuration macros in section 3.2.2 on page 8.

Return Value	Description
GSC_SUCCESS	This is returned when no errors are encountered.
else	Some other GSC_XXX macro value is returned in the event of an error.

3.5.2. hpdi32_gpio_config_set()

This function is the entry point to adjusting the GPIO configuration for the device being accessed.

Prototype

```
U32 hpdi32_gpio_config_set(void* dev, U8 cfg_en, U8 cfg_dir,
    U8 ctrl_io_bin, U32 data_io_bin);
```

Argument	Description
dev	This is the device handle returned from the API Library's hpdi32_open() function.
cfg_en	This argument specifies the enable configuration. If a bit is set, then the corresponding GPIO resource is enabled. If a bit is clear, then it is disabled. Unsupported GPIO resources are ignored. *
cfg_dir	This argument specifies the GPIO direction configuration. If a bit is set, then the corresponding GPIO resource is an output. If a bit is clear, then it is an input. Resources that are disabled or unsupported are ignored. *
ctrl_io_bin	This is the GPIO output pattern to apply to the control lines. A pattern bit is ignored if the corresponding resource is unsupported, disabled or an input.
data_io_bin	This is the GPIO output pattern to apply to the 32 data lines. The value is ignored if the resource is unsupported, disabled or an input.

* Refer to the configuration macros in section 3.2.2 on page 8.

Return Value	Description
GSC_SUCCESS	This is returned when no errors are encountered.
else	Some other GSC_XXX macro value is returned in the event of an error.

3.6. Input/Output Functions

The following functions are used to either read from the GPIO inputs or write to the GPIO outputs.

3.6.1. hpdi32_gpio_ctrl_rx()

This function is the entry point to read the state of the GPIO control lines configured as inputs.

Prototype

```
U32 hpdi32_gpio_ctrl_rx(void* dev, U8* io_bin);
```

Argument	Description
dev	This is the device handle returned from the API Library's hpdi32_open() function.
io_bin	The state of the GPIO control lines is reported here. A bit is set if the corresponding GPIO resource is present, enabled, an input and is driven high. Bits are otherwise clear. *

* Refer to the configuration macros in section 3.2.2 on page 8.

Return Value	Description
GSC_SUCCESS	This is returned when no errors are encountered.
else	Some other GSC_XXX macro value is returned in the event of an error.

3.6.2. hpdi32_gpio_ctrl_tx()

This function is the entry point to updating the state of the GPIO control lines configured as outputs.

Prototype

```
U32 hpdi32_gpio_ctrl_tx(void* dev, U8 io_bin);
```

Argument	Description
dev	This is the device handle returned from the API Library's hpdi32_open() function.
io_bin	This is the GPIO output pattern to apply to the control lines. A pattern bit is ignored if the corresponding resource is unsupported, disabled or an input. *

* Refer to the configuration macros in section 3.2.2 on page 8.

Return Value	Description
GSC_SUCCESS	This is returned when no errors are encountered.
else	Some other GSC_XXX macro value is returned in the event of an error.

3.6.3. hpdi32_gpio_data_rx()

This function is the entry point to read the state of the GPIO data lines when configured as inputs.

Prototype

```
U32 hpdi32_gpio_data_rx(void* dev, U32* io_bin);
```

Argument	Description
dev	This is the device handle returned from the API Library's hpdi32_open() function.
io_bin	The state of the GPIO data lines is reported here. When the GPIO resource is present, enabled and configured as an input, the value reported is according to how the lines are driven. The value is otherwise zero.

Return Value	Description
GSC_SUCCESS	This is returned when no errors are encountered.
else	Some other GSC_XXX macro value is returned in the event of an error.

3.6.4. hpdi32_gpio_data_tx()

This function is the entry point to updating the state of the GPIO data lines configured as outputs.

Prototype

```
U32 hpdi32_gpio_data_tx(void* dev, U32 io_bin);
```

Argument	Description
dev	This is the device handle returned from the API Library's hpdi32_open() function.
io_bin	This is the GPIO output pattern to apply to the data lines. The pattern is ignored if the resource is unsupported, disabled or an input.

Return Value	Description
GSC_SUCCESS	This is returned when no errors are encountered.
else	Some other GSC_XXX macro value is returned in the event of an error.

Document History

Revision	Description
October 7, 2014	Initial library release.