DIO24

Discrete 24-bit Digital I/O

All Form Factors
…-DIO24

API Library
Reference Manual

Manual Revision: May 4, 2023
Version 9.8.104.X.X

General Standards Corporation
8302A Whitesburg Drive
Huntsville, AL 35802
Phone: (256) 880-8787
Fax: (256) 880-8788
URL: 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 documentation and/or software, 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, **General Standards Corporation** assumes no responsibility for any errors, inaccuracies or omissions herein. This documentation, information and software are made available solely on an "as-is" basis. Nor is there any commitment to update or keep current this documentation.

General Standards Corporation does not assume any liability arising out of the application or use of documentation, software, 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 documentation, software or product, to improve accuracy, clarity, reliability, performance, function, or design.

GSC is a trademark of **General Standards Corporation**.

PLX and PLX Technology are trademarks of PLX Technology, Inc.

Table of Contents

1. Introduction **7**
1.1. Purpose 7
1.2. Acronyms 7
1.3. Definitions 7
1.4. Software Overview 7
  1.4.1. Basic Software Architecture 7
  1.4.2. API Library 8
  1.4.3. Device Driver 8
1.5. Hardware Overview 8
1.6. Reference Material 8
1.7. Licensing 9
2. Installation **10**
2.1. Host and Environment Support 10
2.2. Driver and Device Information 10
2.3. File List 10
2.4. Directory Structure 10
2.5. Installation 11
2.6. Removal 11
2.7. Overall Make Script 11
2.8. Environment Variables 11
3. Main Interface Files **12**
3.1. Main Header File 12
3.2. Main Library File 12
  3.2.1. Build 12
  3.2.2. Additional Libraries 12
4. API Library **13**
4.1. Files 13
4.2. Build 13
4.3. Library Use 13
4.4. Macros 13
  4.4.1. IOCTL Codes 13
  4.4.2. Registers 13
4.5. Data Types 14
4.6. Functions 14
  4.6.1. dio24_close() 14
  4.6.2. dio24_init() 15
  4.6.3. dio24_ioctl() 16

4.6.4. dio24_open()....16
4.6.5. dio24_read() 18

4.7. IOCTL Services 18
4.7.1. DIO24_IOCTL_CLOCK_COUNT_RX....18
4.7.2. DIO24_IOCTL_CLOCK_LATCH_FALL_RX 19
4.7.3. DIO24_IOCTL_CLOCK_LATCH_RISE_RX 19
4.7.4. DIO24_IOCTL_GPIO_DIR_OUT 19
4.7.5. DIO24_IOCTL_GPIO_RX 20
4.7.6. DIO24_IOCTL_GPIO_TX....20
4.7.7. DIO24_IOCTL_IRQ_FALL_ENABLE 20
4.7.8. DIO24_IOCTL_IRQ_RISE_ENABLE 21
4.7.9. DIO24_IOCTL_INITIALIZE....21
4.7.10. DIO24_IOCTL_QUERY....21
4.7.11. DIO24_IOCTL_REG_MOD 22
4.7.12. DIO24_IOCTL_REG_READ 22
4.7.13. DIO24_IOCTL_REG_WRITE....23
4.7.14. DIO24_IOCTL_WAIT_CANCEL 23
4.7.15. DIO24_IOCTL_WAIT_EVENT 24
4.7.16. DIO24_IOCTL_WAIT_STATUS 25

5. The Driver....27

5.1. Files....27

5.2. Build 27

5.3. Startup....27

5.4. Verification 27

5.5. Version....27

5.6. Shutdown 27

6. Document Source Code Examples....28

6.1. Files....28

6.2. Build 28

6.3. Library Use 28

7. Utility Source Code 29

7.1. Files....29

7.2. Build 29

7.3. Library Use 29

8. Operating Information 30

8.1. Debugging Aids 30
8.1.1. Device Identification 30
8.1.2. Detailed Register Dump 30

8.2. Digital Input Configuration....30

8.3. Digital Output Configuration 31

9. Sample Applications 32

Document History 33

Table of Figures

Figure 1 Basic architectural representation..8

1. Introduction

1.1. Purpose

The purpose of this document is to describe the interface to the DIO24 API Library and, to a lesser extent, the underlying device driver. The API Library software provides the interface between "Application Software" and the device driver. The driver software provides the interface between the API Library and the actual DIO24 hardware. The API Library and device driver interfaces are primarily IOCTL based.

1.2. Acronyms

The following is a list of commonly occurring acronyms which may appear throughout this document.

Acronyms	Description
API	Application Programming Interface
DIL	Driver Interface Library
DIO	Digital I/O
DLL	Dynamic Link Library
GSC	General Standards Corporation
PCI	Peripheral Component Interconnect
PCIe	PCI Express
PMC	PCI Mezzanine Card

1.3. Definitions

The following is a list of commonly occurring terms which may appear throughout this document.

Term	Definition
…	This is a shortcut representation of the DIO24 installation directory or any of its subdirectories.
API Library	This is a library that provides application-level access to DIO24 hardware.
Application	This is a user mode process, which runs in user space with user mode privileges.
DIO24	This is used as a general reference to any board supported by this driver.
Driver	This refers to the device driver. Depending on the OS, the driver may be a user space application, a kernel mode process, or something in between.
Library	This is usually a general reference to the API Library.
Linux	This refers to the Linux operating system. Refer to the *DIO24 Linux Driver User Manual*.
Windows	This refers to the Windows operating system. Refer to the *DIO24 Windows Driver User Manual*.

1.4. Software Overview

1.4.1. Basic Software Architecture

This section describes the general architecture for the basic components that comprise DIO24 applications. The overall architecture is illustrated in Figure 1 below.

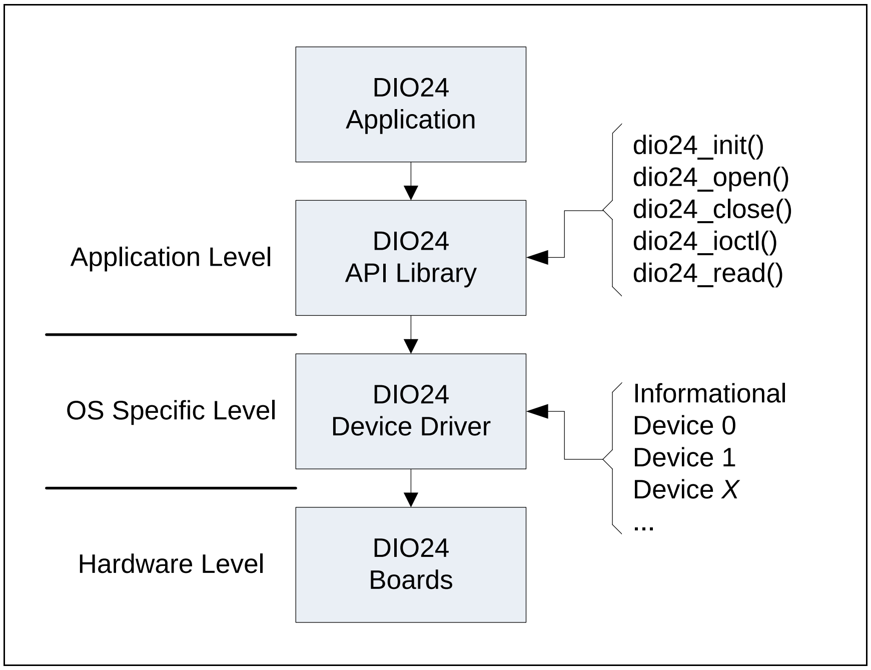

Figure 1 Basic architectural representation.

1.4.2. API Library

The primary means of accessing DIO24 hardware is via the DIO24 API Library. This library is application-level code that sits between an DIO24 application and the DIO24 device driver. With the library, applications are able to open and close a device and, while open, perform I/O control operations, and read data from the driver. For additional information refer to section 4 (page 13).

1.4.3. Device Driver

The device driver is the host software that provides a means of communicating directly with DIO24 hardware. Depending on the OS, the driver may be a user space application, a kernel mode process, or something in between. The software interface to the device driver is analogous to that of the API Library.

1.5. Hardware Overview

The DIO24 is a simple 24-bit discrete I/O interface board. The host side connection is PCI based and the external I/O varies with the transceivers ordered. The external interface includes 24 pins that can be arbitrarily programmed as either input or output. The 24 programmable pins are divided into three groups of eight pins each; Port A, Port B and Port C. Ports A and B are each programmable as all inputs or all outputs. The Port C pins are individually programmable. Each of the 24 I/O pins can also be configured to generate an interrupt on rising and/or falling edges. Depending on ordering options, the cable interface may include a 25th dedicated input or a reference ground connection.

1.6. Reference Material

The following reference material may be of particular benefit in using the DIO24, the API Library and the device driver. The specifications provide the information necessary for an in depth understanding of the specialized features implemented on this device.

- The applicable *DIO24 Device Driver User Manual* for your operating system from General Standards Corporation.

- The applicable *DIO24 User Manual* from General Standards Corporation.
- The *PCI9080 PCI Bus Master Interface Chip* data handbook from PLX Technology, Inc.

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

1.7. Licensing

For licensing information please refer to the text file `LICENSE.txt` in the root installation directory.

2. Installation

For additional information on driver installation refer to this same section number in the OS specific DIO24 driver user manual.

2.1. Host and Environment Support

For information on host and environment support refer to this same section number in the OS specific DIO24 driver user manual.

2.2. Driver and Device Information

Each driver implements an OS specific means of obtaining generic, high-level information about the driver and the installed devices. The information is given in textual format. Each line of text begins with an entry name, which is followed immediately by a colon, a space character, and an entry value. Below is an example of what is provided, followed by descriptions of each entry. This information is accessed by passing a device index value of `-1` to the API open service (section 4.6.4, page 16).

```
version: 9.8.104.47
32-bit support: yes
boards: 1
models: DIO24
ids: 0x3
```

Entry	Description
`version`	This gives the driver version number in the form `x.x.x.x`.
`32-bit support`	This reports the driver's support for 32-bit applications. This will be either "`yes`" or "`no`" for 64-bit driver builds and "`yes (native)`" for 32-bit builds.
`boards`	This identifies the total number of DIO24 boards the driver detected.
`models`	This gives a comma separated list of the basic model number for each board the driver detected. The model numbers are listed in the same order that the boards are accessed via the API Library's open function. For this driver all model numbers should be `DIO24`.
`ids`	This is a list identifying the values read from each board's user jumpers. The id numbers are listed in the same order that the boards are accessed via the API Library's open function.

The API's source for the text provided is as follows.

OS	Source
Linux	The file "`/proc/dio24`".
Windows	The Driver Interface Library DLL.

2.3. File List

For the list of primary files included with each release refer to this same section number in the OS specific DIO24 driver user manual.

2.4. Directory Structure

The following table is representative of the directory structure utilized by each DIO24 driver installation. During the installation process the directory structure is created and populated with the respective files.

NOTE: Additional or alternate directories may be installed, depending on the OS. For additional information refer to this same section number in the OS specific DIO24 driver user manual.

Directory	**Description**
`dio24/`	This is the driver root directory. It contains the documentation, the Overall Make Script (section 2.7, page 11) and the below listed subdirectories.
…`/api/`	This directory contains the DIO24 API Library sources (section 4, page 13).
…`/docsrc/`	This directory contains the code samples from the reference manual (section 6, page 28).
…`/driver/`	This directory contains the driver sources (section 5, page 27).
…`/include/`	This directory contains the header files for the various libraries.
…`/lib/`	This directory contains all of the libraries built from the driver archive sources.
…`/samples/`	This directory contains the sample applications sources (section 9, page 32).
…`/utils/`	This directory contains utility sources used by the sample applications (section 7, page 29).

2.5. Installation

For installation instructions refer to this same section number in the OS specific DIO24 driver user manual.

2.6. Removal

For removal instructions refer to this same section number in the OS specific DIO24 driver user manual.

2.7. Overall Make Script

Each DIO24 installation includes an OS specific means of building all of the build targets included in the installation. For additional information refer to this same section number in the OS specific DIO24 driver user manual.

2.8. Environment Variables

The use of any environment variables is OS specific. Refer to the OS specific driver user manual.

3. Main Interface Files

This section gives general information on the suggested device interface files to use when developing DIO24 based applications.

3.1. Main Header File

Throughout the remainder of this document references are made to various header files included as part of the DIO24 installation. For ease of use it is suggested that applications include only the single header file shown below rather than individually including those headers identified separately later in this document. Including this header file pulls in all other pertinent DIO24 specific header files. Therefore, sources may include only this one DIO24 header and make files may reference only this one DIO24 include directory.

Description	File	Location	OS
Header File	`dio24_main.h`	`…/include/`	All

3.2. Main Library File

Throughout the remainder of this document references are made to various statically linkable libraries included as part of the DIO24 installation. For ease of use it is suggested that applications link only the single static library file shown below rather than individually linking those static libraries identified elsewhere in this document. Linking this static library file pulls in all other pertinent DIO24 specific static libraries. Therefore, make files may reference only this one DIO24 static library and only this one DIO24 library directory.

Description	File	Location	OS
Library File	`dio24_main.a` `dio24_multi.a`	`…/lib/`	Linux
	`dio24_main.lib` `dio24_multi.lib`	`…\lib\…`	Windows

> **NOTE**: For applications using the DIO24 and no other GSC devices, link the `dio24_main.xxx` library. For applications using multiple GSC device types, link the `xxxx_main.xxx` library for one of the devices and the `xxxx_multi.xxx` library for the others. Linking multiple `xxxx_main.xxx` libraries may likely produce link errors due to duplicate symbols being defined. While it may make little or no difference, it is recommended that one choose the `xxxx_main.xxx` library from the driver with the largest number in positions three (x.x.X.x.x) and/or four (x.x.x.X.x) in the driver release version number.

3.2.1. Build

For information on building the Main Library refer to this same section number in the OS specific DIO24 driver user manual.

3.2.2. Additional Libraries

For information on any additional required libraries refer to this same section number in the OS specific DIO24 driver user manual.

4. API Library

The DIO24 API Library is the software interface between user applications and the DIO24 device driver. The interface is accessed by including the header file `dio24_api.h`.

> **NOTE:** Contact General Standards Corporation if additional library functionality is required.

4.1. Files

The API Library is built into a library linkable with DIO24 applications. The pertinent files are identified in the following table. Some source files are specific only to the DIO24, some are specific only to the OS and some are DIO24 and OS independent.

Description	**Files**	**Location**	**OS**
Source Files	`*.c`, `*.h`	…`/api/`	All
Header File	`dio24_api.h`	…`/include/`	All
Library File	`libdio24_api.so` †	…`/lib/` `/usr/lib/`	Linux
	`dio24_api.lib` `dio24_api.dll` ‡	…`\lib\`…	Windows

† The Linux run time executable is implemented as a shared object file.
‡ The Windows run time executable is implemented as a DLL.

4.2. Build

For build instructions refer to this same section number in the OS specific DIO24 driver user manual.

4.3. Library Use

For Library usage information refer to this same section number in the OS specific DIO24 driver user manual.

4.4. Macros

The Library interface includes the following macros, which are defined in `dio24.h`.

4.4.1. IOCTL Codes

The IOCTL macros are documented in section 4.7 (page 18).

4.4.2. Registers

The following gives the complete set of DIO24 registers.

4.4.2.1. GSC Registers

The following table gives the DIO24 firmware register macros used by the DIO24 API Library interface.

Macros	**Description**
`DIO24_GSC_BCR`	Board Control Register (BCR)
`DIO24_GSC_BSR`	Board Status Register (BSR)
`DIO24_GSC_DDIR`	Discrete Data Input Register (DDIR)
`DIO24_GSC_DDOR`	Discrete Data Output Register (DDOR)
`DIO24_GSC_FRR`	Firmware Revision Register (FRR)

`DIO24_GSC_IFCR`	Interrupt Falling Control Register (IFCR) *
`DIO24_GSC_IFSR`	Interrupt Falling Status Register (IFSR) *
`DIO24_GSC_IRCR`	Interrupt Rising Control Register (IRCR) *
`DIO24_GSC_IRSR`	Interrupt Rising Status Register (IRSR) *
`DIO24_GSC_IOCR`	I/O Control Register (IOCR)

* These registers are present only on boards supporting interrupts.

4.4.2.2. PCI Configuration Registers

Access to the PCI registers is seldom required so these registers are not listed here. For the complete list of the PCI register identifiers refer to header file `gsc_pci9080.h`, which is automatically included via `dio24_api.h`.

4.4.2.3. PLX Feature Set Registers

Access to the PLX registers is seldom required so these registers are not listed here. For the complete list of the PLX register identifiers refer to header file `gsc_pci9080.h`, which is automatically included via `dio24_api.h`.

4.5. Data Types

The data types used by the API Library are described with the IOCTL services with which they are used. For additional information refer to section 4.7 (page 18).

4.6. Functions

The interface includes the following functions. The return values reflect the completion status of the requested operation. A return value less than zero always reflects an error condition. The table below summarizes the error status values. For the I/O function, read, non-negative return values reflect the number of bytes transferred between the application and the interface. A value equal to the requested transfer size indicates complete success. Return values less than the requested transfer size indicate that the I/O timeout expired. For the other API function calls a return value of zero indicates success.

Return Value	Description	OS
`-1` to `-499`	This is the value "`(-errno)`" (see `errno.h`).	All
`-500` to `-999`	This is the value returned from the Driver Interface Library. *	Windows
`<= -1000`	This is "`(int)(GetLastError()+1000)`" forced to a negative value.	

* Applicable error codes, if any, are defined in the header `os_common.h`.

4.6.1. dio24_close()

This function is the entry point to close a connection to an open DIO24. The board is put in an initialized state before this call returns.

Prototype

```
int dio24_close(int fd);
```

Argument	Description
`fd`	This is the file descriptor of the device to be closed.

Return Value	Description
`0`	The operation succeeded.
`< 0`	An error occurred. See error value description above.

Example

```
#include <stdio.h>

#include "dio24_dsl.h"

int dio24_close_dsl(int fd)
{
    int errs;
    int ret;

    ret = dio24_close(fd);

    if (ret)
        printf("ERROR: dio24_close() returned %d\n", ret);

    errs    = ret ? 1 : 0;
    return(errs);
}
```

4.6.2. dio24_init()

This function is the entry point to initializing the DIO24 API Library and must be the first call into the Library. This function may be called more than once, but only the first successful call actually initializes the library. Subsequent calls perform no operation at all. All other API calls return a failure status when the API Library is not initialized.

Prototype

```
int dio24_init(void);
```

Return Value	Description
`0`	The operation succeeded.
`< 0`	An error occurred. See error value description above.

Example

```
#include <stdio.h>

#include "dio24_dsl.h"

int dio24_init_dsl(void)
{
    int errs;
    int ret;

    ret = dio24_init();

    if (ret)
        printf("ERROR: dio24_init() returned %d\n", ret);

    errs    = ret ? 1 : 0;
    return(errs);
}
```

4.6.3. dio24_ioctl()

This function is the entry point to performing setup and control operations on a DIO24 board. This function should only be called after a successful open of the respective device. The specific operation performed varies according to the `request` argument. The `request` argument also governs the use and interpretation of the `arg` argument. The set of supported options for the `request` argument consists of the IOCTL services supported by the driver, which are defined in section 4.7 beginning on page 18. The `arg` argument varies according to the specified IOCTL service. When not used `arg` should be NULL.

NOTE: IOCTL operations are not supported for an open on device index `-1`.

Prototype

```
int dio24_ioctl(int fd, int request, void* arg);
```

Argument	Description
`fd`	This is the file descriptor of the device to access.
`request`	This specifies the desired operation to be performed.
`arg`	This is specific to the requested IOCTL operation.

Return Value	Description
`0`	The operation succeeded.
`< 0`	An error occurred. See error value description above.

Example

```
#include <stdio.h>

#include "dio24_dsl.h"

int dio24_ioctl_dsl(int fd, int request, void *arg)
{
    int errs;
    int ret;

    ret = dio24_ioctl(fd, request, arg);

    if (ret)
        printf("ERROR: dio24_ioctl() returned %d\n", ret);

    errs    = ret ? 1 : 0;
    return(errs);
}
```

4.6.4. dio24_open()

This function is the entry point to open a connection to a DIO24 board. Before returning, the initialize IOCTL service is called to reset all hardware and software settings to their defaults.

Prototype

```
int dio24_open(int device, int share, int* fd);
```

<table>
<tr><th>Argument</th><th>Description</th></tr>
<tr><td>device</td><td>This is the zero-based index of the DIO24 to access. *</td></tr>
<tr><td>share</td><td>Open the device in Shared Access Mode? If non-zero the device is opened in Shared Access Mode (see below). If zero the device is opened in Exclusive Access Mode (see below).</td></tr>
<tr><td>fd</td><td>The device handle is returned here. The pointer cannot be NULL. Values returned are as follows.
<table>
<tr><th>Value</th><th>Description</th></tr>
<tr><td>>= 0</td><td>This is the handle to use to access the device in subsequent calls.</td></tr>
<tr><td>-1</td><td>There was an error. The device is not accessible.</td></tr>
</table>
</td></tr>
</table>

* The index value `-1` can also be given to acquire driver information (section 2.2, page 10).

Return Value	Description
0	The operation succeeded.
< 0	An error occurred. See error value description above.

Example

```
#include <stdio.h>

#include "dio24_dsl.h"

int dio24_open_dsl(int device, int share, int* fd)
{
    int errs;
    int ret;

    ret = dio24_open(device, share, fd);

    if (ret)
        printf("ERROR: dio24_open() returned %d\n", ret);

    errs    = ret ? 1 : 0;
    return(errs);
}
```

4.6.4.1. Access Modes

The value of the `share` argument determines the device access mode, as follows.

Shared Access Mode:

Shared Access Mode allows multiple applications to access the same device simultaneously. In this mode, the first successful open request returns with the device in an initialized state. Subsequent successful Shared Access Mode open requests do not affect the state of the device. Once opened in Shared Access Mode, the device access remains in this mode until all Shared Access Mode accesses release the device with a close request.

Exclusive Access Mode:

Exclusive Access Mode allows a single application to acquire exclusive access to a device. In this mode, a successful open request returns with the device in an initialized state. While open in this mode all subsequent open requests will fail regardless of the access mode requested. Once opened in Exclusive Access Mode, the device access remains in this mode until released by the application with a close request.

4.6.5. dio24_read()

This function is the entry point to reading data from an open performed on device index `-1`. This function should only be called after a successful open. The function reads up to `bytes` bytes. The return value is the number of bytes actually read.

> **NOTE:** The read service has no functionality for reading from DIO24 devices. Attempts to read from DIO24 devices will return an error.

Prototype

```
int dio24_read(int fd, void *dst, size_t bytes);
```

Argument	Description
`fd`	This is the file descriptor of the device to access.
`dst`	The data read is put here.
`bytes`	This is the desired number of bytes to read.

Return Value	Description
`0` to `bytes`	The operation succeeded.
`< 0`	An error occurred. See error value description above.

Example

```
#include <stdio.h>

#include "dio24_dsl.h"

int dio24_read_dsl(int fd, void* dst, size_t bytes, size_t* qty)
{
    int errs;
    int ret;

    ret = dio24_read(fd, dst, bytes);

    if (ret < 0)
        printf("ERROR: dio24_read() returned %d\n", ret);

    if (qty)
        qty[0]  = (ret < 0) ? 0 : (size_t) ret;

    errs    = (ret < 0) ? 1 : 0;
    return(errs);
}
```

4.7. IOCTL Services

The DIO24 API Library and device driver implement the following IOCTL services. Each service is described along with the applicable `dio24_ioctl()` function arguments.

4.7.1. DIO24_IOCTL_CLOCK_COUNT_RX

This service retrieves the current clock counter value, which is a 32-bit value driven by the board’s 33.0MHz clock.

Usage

Argument	Argument
`request`	`DIO24_IOCTL_CLOCK_COUNT_RX`
`arg`	`u32*`

The value returned is from `0x0` through `0xFFFFFFFF`, if the counter feature is supported. If the feature is unavailable, the value returned is `0xFFFFFFFF`.

4.7.2. DIO24_IOCTL_CLOCK_LATCH_FALL_RX

This service retrieves the clock counter value latched on most recent falling edge of the specified I/O pin.

Usage

Argument	Argument
`request`	`DIO24_IOCTL_CLOCK_LATCH_FALL_RX`
`arg`	`u32*`

The value passed in is the I/O pin to query and must be from zero through three. The value returned is from `0x0` through `0xFFFFFFFF`, if the counter feature is supported. If the feature is unavailable, the value returned is `0xFFFFFFFF`.

4.7.3. DIO24_IOCTL_CLOCK_LATCH_RISE_RX

This service retrieves the clock counter value latched on most recent rising edge of the specified I/O pin.

Usage

Argument	Argument
`request`	`DIO24_IOCTL_CLOCK_LATCH_RISE_RX`
`arg`	`u32*`

The value passed in is the I/O pin to query and must be from zero through three. The value returned is from `0x0` through `0xFFFFFFFF`, if the counter feature is supported. If the feature is unavailable, the value returned is `0xFFFFFFFF`.

4.7.4. DIO24_IOCTL_GPIO_DIR_OUT

This service configures the direction of the configurable input/output ports.

Usage

Argument	Argument
`request`	`DIO24_IOCTL_GPIO_DIR_OUT`
`arg`	`s32*`

Valid argument values include any bitwise combination of the below options, as well as `-1`, which requests the current setting. Each value included from below configures the corresponding port selections as outputs. Options not included are configured as inputs.

Value	Description
`DIO24_GPIO_PORT_ABC`	This refers to all 24 bits of Port A, B and C.
`DIO24_GPIO_PORT_A`	This refers to all eight bits of Port A.

`DIO24_GPIO_PORT_B`	This refers to all eight bits of Port B.
`DIO24_GPIO_PORT_C`	This refers to all eight bits of Port C.
`DIO24_GPIO_PORT_C0`	This refers to pin D0 of Port C.
`DIO24_GPIO_PORT_C1`	This refers to pin D1 of Port C.
`DIO24_GPIO_PORT_C2`	This refers to pin D2 of Port C.
`DIO24_GPIO_PORT_C3`	This refers to pin D3 of Port C.
`DIO24_GPIO_PORT_C4`	This refers to pin D4 of Port C.
`DIO24_GPIO_PORT_C5`	This refers to pin D5 of Port C.
`DIO24_GPIO_PORT_C6`	This refers to pin D6 of Port C.
`DIO24_GPIO_PORT_C7`	This refers to pin D7 of Port C.

4.7.5. DIO24_IOCTL_GPIO_RX

This service returns the current state of the Port A, Port B, Port C and the Dedicated Input port pins.

Usage

Argument	Description
`request`	`DIO24_IOCTL_GPIO_RX`
`arg`	`s32*`

Valid argument values returned are from zero through `0x1FFFFFF`. Port A is reported in bits D0 through D7. Port B is reported in bits D8 through D15. Port C is reported in bits D16 through D23. The Dedicated Input is reported in bit D24. The value reported for an input port is the state imposed by equipment attached at the cable interface. The value reported for an output port is the state imposed by the DIO24's corresponding output latch.

4.7.6. DIO24_IOCTL_GPIO_TX

This service configures the values of the 24 output port latches. The values written to the output latches appear at the cable interface only for those corresponding pins configured as outputs.

Usage

Argument	Description
`request`	`DIO24_IOCTL_GPIO_TX`
`arg`	`s32*`

Valid argument values are from zero through `0xFFFFFF`, and `-1`, which returns the current latched value. Port A is configured via bits D0 through D7. Port B is configured via bits D8 through D15. Port C is configured via bits D16 through D23.

4.7.7. DIO24_IOCTL_IRQ_FALL_ENABLE

This service enables and disables Falling Edge interrupt generation for the 24 I/O ports.

Usage

Argument	Description
`request`	`DIO24_IOCTL_IRQ_FALL_ENABLE`
`arg`	`s32*`

Valid argument values are from zero through `0xFFFFFF`, and `-1`, which returns the current state. If a bit position is set, then the Falling Edge interrupt is enabled for the corresponding cable I/O signal. If the bit position is clear, then the interrupt is disabled. The Port A Falling Edge interrupts are configured via bits D0 through D7. The Port B

Falling Edge interrupts are configured via bits D8 through D15. The Port C Falling Edge interrupts are configured via bits D16 through D23. The value `-1` is returned when the interrupt feature is not supported.

4.7.8. DIO24_IOCTL_IRQ_RISE_ENABLE

This service enables and disables Rising Edge interrupt generation for the 24 I/O ports.

Usage

Argument	**Description**
`request`	`DIO24_IOCTL_IRQ_RISE_ENABLE`
`arg`	`s32*`

Valid argument values are from zero through `0xFFFFFF`, and `-1`, which returns the current state. If a bit position is set, then the Rising Edge interrupt is enabled for the corresponding cable I/O signal. If the bit position is clear, then the interrupt is disabled. The Port A Rising Edge interrupts are configured via bits D0 through D7. The Port B Rising Edge interrupts are configured via bits D8 through D15. The Port C Rising Edge interrupts are configured via bits D16 through D23. The value `-1` is returned when the interrupt feature is not supported.

4.7.9. DIO24_IOCTL_INITIALIZE

This service performs an initialization of all DIO24 settings. This include both hardware and software settings. This puts the driver and the device in the same state it was in when initially opened.

Usage

Argument	**Description**
`request`	`DIO24_IOCTL_INITIALIZE`
`arg`	Not used.

4.7.10. DIO24_IOCTL_QUERY

This service queries the driver for various pieces of information about the board and the driver.

Usage

Argument	**Description**
`request`	`DIO24_IOCTL_QUERY`
`arg`	`s32*`

Valid argument values are as follows.

Value	**Description**
`DIO24_QUERY_COUNT`	This returns the number of query options supported by the IOCTL service.
`DIO24_QUERY_DEVICE_TYPE`	This returns the identifier value for the board's type. The value should be `GSC_DEV_TYPE_DIO24`.
`DIO24_QUERY_DI`	This indicates if the board includes the cable interface Dedicated Input option. The feature is present only if the return value is non-zero.
`DIO24_QUERY_IRQ_QTY`	This option reports on the firmware support for interrupts. The value is 48 when the firmware supports rising and falling interrupt on each of the 24 I/O pins. The value is otherwise zero.
`DIO24_QUERY_JUMPERS`	This returns the numeric value of the user jumpers.

`DIO24_QUERY_JUMPER_QTY`	This returns the number of user jumpers.
`DIO24_QUERY_JUMPER_SENSE`	This returns the bit value of a jumper that is installed.

Valid return values are as indicated in the above tables and as given in the below table.

Value	**Description**
`DIO24_IOCTL_QUERY_ERROR`	Either there was a processing error or the query option is unrecognized.

4.7.11. DIO24_IOCTL_REG_MOD

This service performs a read-modify-write of an DIO24 register. This includes only the GSC firmware registers. The PCI and PLX Feature Set Registers are read-only. Refer to `dio24.h` for a complete list of the GSC firmware registers.

Usage

Argument	**Description**
`request`	`DIO24_IOCTL_REG_MOD`
`arg`	`gsc_reg_t*`

Definition

```
typedef struct
{
    u32 reg;
    u32 value;
    u32 mask;
} gsc_reg_t;
```

Fields	**Description**
`reg`	This is set to the identifier for the register to access.
`value`	This contains the value for the register bits to modify.
`mask`	This specifies the set of bits to modify. If a bit here is set, then the respective register bit is modified. If a bit here is zero, then the respective register bit is unmodified.

4.7.12. DIO24_IOCTL_REG_READ

This service reads the value of a DIO24 register. This includes the PCI registers, the PLX Feature Set Registers and the GSC firmware registers. Refer to `dio24.h` and `gsc_pci9080.h` for the complete list of accessible registers.

Usage

Argument	**Description**
`request`	`DIO24_IOCTL_REG_READ`
`arg`	`gsc_reg_t*`

Definition

```
typedef struct
{
    u32 reg;
    u32 value;
    u32 mask;
} gsc_reg_t;
```

Fields	Description
`reg`	This is set to the identifier for the register to access.
`value`	This is the value read from the specified register.
`mask`	This is ignored for read request.

4.7.13. DIO24_IOCTL_REG_WRITE

This service writes a value to a DIO24 register. This includes only the GSC firmware registers. The PCI and PLX Feature Set Registers are read-only. Refer to `dio24.h` for a complete list of the GSC firmware registers.

Usage

Argument	Description
`request`	`DIO24_IOCTL_REG_WRITE`
`arg`	`gsc_reg_t*`

Definition

```
typedef struct
{
    u32 reg;
    u32 value;
    u32 mask;
} gsc_reg_t;
```

Fields	Description
`reg`	This is set to the identifier for the register to access.
`value`	This is the value to write to the specified register.
`mask`	This is ignored for write request.

4.7.14. DIO24_IOCTL_WAIT_CANCEL

This service resumes all threads blocked via `DIO24_IOCTL_WAIT_EVENT` IOCTL calls (section 4.7.15, page 24), according to the provided criteria. When a blocked thread is waiting for any event specified in the structure, then the thread is resumed.

Usage

Argument	Description
`request`	`DIO24_IOCTL_WAIT_CANCEL`
`arg`	`dio24_wait_t*`

Definition

```
typedef struct
{
   u32  flags;
   u32  main;
   u32  rise;
   u32  fall;
   u32  reserved;
   u32  timeout_ms;
```

```
    u32  count;
} dio24_wait_t;
```

Fields	Description
`flags`	This is unused by wait cancel operations.
`main`	This specifies the set of `GSC_WAIT_MAIN_*` events whose wait requests are to be cancelled. Refer to section 4.7.15.2 on page 25.
`rise`	This specifies the set of `DIO24_WAIT_GSC_*` events whose wait requests are to be cancelled. Refer to section 4.7.15.3 on page 25.
`fall`	This specifies the set High-to-Low interrupt events whose wait requests are to be cancelled. Refer to section 4.7.15.4 on page 25.
`reserved`	This field is reserved and must be zero.
`timeout_ms`	This is unused by wait cancel operations.
`count`	Upon return this indicates the number of waits that were cancelled.

4.7.15. DIO24_IOCTL_WAIT_EVENT

This service blocks a thread until any one of a specified set of events occurs, or until a timeout lapses, whichever occurs first. The set of possible events to wait for are specified in the structure's `main`, `rise`, and `fall` fields. All field values must be valid and at least one event must be specified. If the thread is resumed because one of the referenced events has occurred, then the bit for the respective event is the only event bit that will be set. All other event bits and fields will be zero. (Multiple event bits will be set only if the events occur simultaneously.)

NOTE: The service waits only for the first of the specified events, not for all specified events.

NOTE: A wait timeout is reported via the `gsc_wait_t` structure's `flags` field having the `GSC_WAIT_FLAG_TIMEOUT` flag set, rather than via an `ETIMEDOUT` error.

Usage

Argument	Description
`request`	`DIO24_IOCTL_WAIT_EVENT`
`arg`	`dio24_wait_t*`

Definition

```
typedef struct
{
    u32  flags;
    u32  main;
    u32  rise;
    u32  fall;
    u32  reserved;
    u32  timeout_ms;
    u32  count;
} dio24_wait_t;
```

Fields	Description
`flags`	This must initially be zero. Upon return this indicates the reason that the thread was resumed. Refer to section 4.7.15.1on page 25.
`main`	This specifies any number of `GSC_WAIT_MAIN_*` events that the thread is to wait for. Refer to section 4.7.15.2 on page 25.

`rise`	This specifies the set Rising Edge interrupts that the thread is to wait for. Each bit, zero through 23, refers to the corresponding input port. (D0 corresponds to Port A, bit D0.) Valid values are from zero to 0xFFFFFF.
`fall`	This specifies the set Falling Edge interrupts that the thread is to wait for. Each bit, zero through 23, refers to the corresponding input port. (D0 corresponds to Port A, bit D0.) Valid values are from zero to 0xFFFFFF.
`reserved`	This field is reserved and must be zero.
`timeout_ms`	This specified the maximum amount of time, in milliseconds, that the thread is to wait for any of the referenced events. A value of zero means do not timeout at all. If non-zero, then upon return the value will be the approximate amount of time actually waited.
`count`	This is unused by wait event operations and must be zero.

4.7.15.1. `dio24_wait_t.flags` Options

Upon return from a wait request the wait structure's `flags` field will indicate the reason that the thread was resumed. Only one of the below options will be set.

Fields	**Description**
`GSC_WAIT_FLAG_CANCEL`	The wait request was cancelled.
`GSC_WAIT_FLAG_DONE`	One of the referenced events occurred.
`GSC_WAIT_FLAG_TIMEOUT`	The timeout period lapsed before a referenced event occurred.

4.7.15.2. `dio24_wait_t.main` Options

The wait structure's `main` field may specify any of the below primary interrupt options. These interrupt options are supported by the DIO24 and other General Standards products.

Fields	**Description**
`GSC_WAIT_MAIN_GSC`	This refers to any of the Interrupt Control/Status Register interrupts.
`GSC_WAIT_MAIN_OTHER`	This generally refers to an interrupt generated by another device sharing the same interrupt as the DIO24.
`GSC_WAIT_MAIN_PCI`	This refers to any interrupt generated by the DIO24.
`GSC_WAIT_MAIN_SPURIOUS`	This refers to board interrupts which should never be generated.
`GSC_WAIT_MAIN_UNKNOWN`	This refers to board interrupts whose source could not be identified.

4.7.15.3. `dio24_wait_t.rise` Options

This field may have any value from zero through 0xFFFFFF. No predefined macros are defined for this field.

4.7.15.4. `dio24_wait_t.fall` Options

This field may have any value from zero through 0xFFFFFF. No predefined macros are defined for this field.

4.7.16. DIO24_IOCTL_WAIT_STATUS

This service counts all threads blocked via the `DIO24_IOCTL_WAIT_EVENT` IOCTL service (section 4.7.15, page 24), according to the provided criteria. A match is made when a waiting thread's wait criteria matches any of the criteria specified in the structure passed to this service.

Usage

Argument	**Description**
`request`	`DIO24_IOCTL_WAIT_STATUS`
`arg`	`dio24_wait_t*`

Definition

```
typedef struct
{
   u32  flags;
   u32  main;
   u32  rise;
   u32  fall;
   u32  reserved;
   u32  timeout_ms;
   u32  count;
} dio24_wait_t;
```

Fields	**Description**
`flags`	This is unused by wait status operations.
`main`	This specifies the set of `GSC_WAIT_MAIN_*` events whose wait requests are to be counted. Refer to section 4.7.15.2 on page 25.
`rise`	This specifies the set of Rising Edge interrupt events whose wait requests are to be counted. Refer to section 4.7.15.3 on page 25.
`fall`	This specifies the set of Falling Edge interrupt events whose wait requests are to be counted. Refer to section 4.7.15.4 on page 25.
`reserved`	This field is reserved and must be zero.
`timeout_ms`	This is unused by wait status operations.
`count`	Upon return this indicates the number of waits that met any of the specified criteria.

5. The Driver

NOTE: Contact General Standards Corporation if additional driver functionality is required.

5.1. Files

The device driver files are summarized in the table below. The driver executable is an OS specific file.

Description	Files	Location	OS
Source Files	`*.c`, `*.h`	…`/driver/`	Linux
Header File	`dio24.h`	…`/driver/`	Linux
Driver File	`dio24.ko` †	…`/driver/`	Linux (kernels version 2.6 and later)
	`dio24.o` †	…`/driver/`	Linux (kernels version 2.4 and earlier)
	`dio24_dil.lib` `dio24_dil.dll`	…`\lib\`…	Windows
	`dio24_9056.sys` ‡	…`\driver\`…	

† The Linux run time executable is implemented as a loadable kernel module.
‡ The Windows run time executable is implemented as a driver `.sys` file.

5.2. Build

For instructions on building the driver refer to this same section number in the OS specific DIO24 driver user manual.

5.3. Startup

For instructions on starting the driver executable refer to this same section number in the OS specific DIO24 driver user manual.

5.4. Verification

For instructions on verifying that the driver has been loaded and is running refer to this same section number in the OS specific DIO24 driver user manual.

5.5. Version

For instructions on obtaining the driver version number refer to this same section number in the OS specific DIO24 driver user manual.

5.6. Shutdown

For instructions on terminating the driver executable refer to this same section number in the OS specific DIO24 driver user manual.

6. Document Source Code Examples

The source code examples included in this document are built into a statically linkable library usable with console applications. The purpose of these files is to verify that the documentation samples compile and to provide a library of working sample code to assist in a user's learning curve and application development effort.

6.1. Files

The library files are summarized in the table below.

Description	**Files**	**Location**	**OS**
Source Files	`*.c`, `*.h` …	…`/docsrc/`	All
Header File	`dio24_dsl.h`	…`/include/`	All
Library File	`dio24_dsl.a`	…`/lib/`	Linux
	`dio24_dsl.lib`	…`\lib\`…	Windows

6.2. Build

For library build instructions refer to this same section number in the OS specific DIO24 driver user manual.

6.3. Library Use

For library usage information refer to this same section number in the OS specific DIO24 driver user manual.

7. Utility Source Code

The DIO24 API Library includes a body of utility source code designed to aid in the understanding and use of all API calls and all IOCTL services. The essence of these utilities is to implement visual wrappers around the corresponding services. Utility sources are also included for device independent and common, general-purpose services. The utility services are used extensively by the sample applications. The utility sources are compiled and linked into two static libraries to simplify their use. The primary files are identified in the following subsection. Some source files are specific only to the DIO24, some are specific only to the OS and some are DIO24 and OS independent.

For each API function there is a corresponding utility source file with a corresponding utility service. As an example, for the API function `dio24_open()` there is the utility file `open.c` containing the utility function `dio24_open_util()`. The naming pattern is as follows: API function `dio24_xxxx()`, utility file name `xxxx.c`, utility function `dio24_xxxx_util()`. Additionally, for each IOCTL code there is a corresponding utility source file with a corresponding utility service. As an example, for IOCTL code `DIO24_IOCTL_QUERY` there is the utility file `util_query.c` containing the utility function `dio24_query()`. The naming pattern is as follows: IOCTL code `DIO24_IOCTL_XXXX`, utility file name `util_xxxx.c`, utility function `dio24_xxxx()`.

7.1. Files

The library files are summarized in the table below.

Description	**Files**	**Location**	**OS**
Source Files	`*.c`, `*,h`, `makefile` …	…`/utils/`	All
Header File	`dio24_utils.h`	…`/include/`	All

Library File	`dio24_utils.a`
	`gsc_utils.a`
	`os_utils.a`
	`plx_utils.a`
		…`/lib/`	Linux

	`dio24_utils.lib`
	`gsc_utils.lib`
	`os_utils.lib`
	`plx_utils.lib`
		…`\lib\`…	Windows

7.2. Build

For library build instruction refer to this same section number in the OS specific DIO24 driver user manual.

7.3. Library Use

For library usage information refer to this same section number in the OS specific DIO24 driver user manual.

8. Operating Information

This section explains some basic operational procedures for using the DIO24. This is in no way intended to be a comprehensive guide. This is simply to address a very few issues relating to their use. For additional operating information refer to this same section number in the OS specific *DIO24 Driver User Manual*.

8.1. Debugging Aids

The driver package includes the following items useful for development and/or debugging aids.

8.1.1. Device Identification

When communicating with technical support complete device identification is virtually always necessary. The *id* example application is provided for this specific purpose. This is a text only console application. The output can be piped to a file, which can then be emailed to GSC technical support when requested. Locate the application as follows.

Description	File	Location	OS
Application	`id`	…`/id/`	Linux
	`id.exe`	…`\id\`…	Windows

8.1.2. Detailed Register Dump

Among the utility services provided is a function to generate a detailed listing of the device registers to the console. When used, the function is typically used to verify the device configuration. In these cases, the function should be called after complete board configuration. When intended for sending to GSC tech support, please set the *detail* argument to `1`. The function arguments are as follows. The utility location is given in the subsequent table.

Argument	Description
`fd`	This is the file descriptor used to access the device.
`detail`	If non-zero the GSC register dump will include details of each register field.

Description	File/Name	Location	OS
Function	`dio24_reg_list()`	Source File	All
Source File	`util_reg.c`	…`/utils/`	All
Header File	`dio24_utils.h`	…`/include/`	All
Library File	`dio24_utils.a`	…`/lib/`	Linux
	`dio24_utils.lib`	…`\lib\`…	Windows

8.2. Digital Input Configuration

The basic steps for digital input configuration are illustrated in the utility function noted below. The table also gives the location of the source file, the header file and the corresponding library containing the executable code.

Description	File/Name	Location	OS
Function	`dio24_config_di()`	Source File	All
Source File	`util_config_di.c`	…`/utils/`	All
Header File	`dio24_utils.h`	…`/include/`	All
Library File	`dio24_utils.a`	…`/lib/`	Linux
	`dio24_utils.lib`	…`\lib\`…	Windows

8.3. Digital Output Configuration

The basic steps for digital output configuration are illustrated in the utility function noted below. The table also gives the location of the source file, the header file and the corresponding library containing the executable code.

Item	**Name/File**	**Location**	**OS**
Function	`dio24_config_do()`	Source File	All
Source File	`util_config_do.c`	…`/utils/`	All
Header File	`dio24_utils.h`	…`/include/`	All
Library File	`dio24_utils.a`	…`/lib/`	Linux
	`dio24_utils.lib`	…`\lib\`…	Windows

9. Sample Applications

For information on the sample applications refer to this same section number in the OS specific DIO24 driver user manual.

Document History

Revision	**Description**
May 5, 2023	Updated release date. Version 9.8.104.x.x. Numerous editorial changes.
December 15, 2022	Updated release date. Minor editorial changes.
November 23, 2022	Updated release date. Version 9.7.101.x.x. Minor editorial updates.
October 3, 2022	Updated release date. Version 9.6.101.x.x. Updated the information for the open and close calls. Discontinued the SDK.
March 8, 2021	Updated release date. Version 9.5.93.x.x.
March 5, 2021	Initial release. Version 9.5.93.x.x.