
# Document number 2007008

# VersaPHY GPIO Profile

## January 19, 2010

**Sponsored by:**

1394 Trade Association

**Accepted for publication by**

This draft specification has been accepted by the 1394 Trade Association Board of Directors

**Abstract**

This specification defines a standardized method to interface General Purpose Input and Output devices to IEEE-1394 networks using the VersaPHY protocols.

**Keywords**

IEEE 13994, Serial Bus, VersaPHY, GPIO, GPI, GPO

# 1394 Trade Association Specification

1394 Trade Association Specifications are developed within Working Groups of the 1394 Trade Association, a non-profit industry association devoted to the promotion of and growth of the market for IEEE 1394-compliant products. Participants in Working Groups serve voluntarily and without compensation from the Trade Association. Most participants represent member organizations of the 1394 Trade Association. The specifications developed within the working groups represent a consensus of the expertise represented by the participants.

Use of a 1394 Trade Association Specification is wholly voluntary. The existence of a 1394 Trade Association Specification is not meant to imply that there are not other ways to produce, test, measure, purchase, market or provide other goods and services related to the scope of the 1394 Trade Association Specification. Furthermore, the viewpoint expressed at the time a specification is accepted and issued is subject to change brought about through developments in the state of the art and comments received from users of the specification. Users are cautioned to check to determine that they have the latest revision of any 1394 Trade Association Specification.

Comments for revision of 1394 Trade Association Specifications are welcome from any interested party, regardless of membership affiliation with the 1394 Trade Association. Suggestions for changes in documents should be in the form of a proposed change of text, together with appropriate supporting comments.

Interpretations: Occasionally, questions may arise about the meaning of specifications in relationship to specific applications. When the need for interpretations is brought to the attention of the 1394 Trade Association, the Association will initiate action to prepare appropriate responses.

Comments on specifications and requests for interpretations should be addressed to:

Editor, 1394 Trade Association
315 Lincoln, Suite E
Mukilteo, WA 98275
USA



Published by

**1394 Trade Association**
**315 Lincoln, Suite E**
**Mukilteo, WA 98275 USA**



Printed in the United States of America

# Contents

## Tables



## Figures

## Annexes

## Foreword

This specification defines a standardized method to interface General Purpose Input and Output devices to IEEE-1394 networks using the VersaPHY protocols.

There are 3 annexes in this specification. Annex A and Annex C are normative and part of this specification. Annexes B and D are informative and are not considered part of this specification.

This specification was accepted by the Board of Directors of the 1394 Trade Association. Board of Directors acceptance of this specification does not necessarily imply that all board members voted for acceptance. At the time it accepted this specification, the 1394 Trade Association Board of Directors had the following members:

Max Bassler, Chair
Morten Lave, Vice-Chair
Dave Thompson, Secretary

| *Organization Represented* | *Name of Representative* |
|---|---|
| Littelfuse | Max Bassler |
| PLX Technology | Don Harwood |
| Texas Instruments | Toni Ray |
| LSI | Dave Thompson |
| TC Applied Technologies | Morten Lave |
| Aztek Corp. | Richard Mourn |

The VersaPHY Working Group, which developed and reviewed this specification, had the following members:

Richard Mourn, Chair

Garold Yurko
Mike Gardener
Max Bassler
Morten Lave
Toni Ray
David Thompson
Richard Mourn
Roumen Botev

## Revision history

**Revision Draft 0.3 (August 05, 2007)**

Initial draft circulated to Industrial Working Group for review.

**Revision Draft 0.4 (October 08, 2007)**

Clean up for formal review.

**Revision Draft 0.5 (January 16, 2008)**

Added Conformance requirements Annex.

Addressed editorial commits from working group vote.

**Revision Draft 0.6 (August 26, 2008)**

Corrected 1294 TA address.

Moved Bibliography to Annex D.

Set_Bits and Clear_Bits description completed.

Concatenate Timestamp description fixed to concatenate registers 4 and 6.

Update_Mask description clarified.

Timestamp overflow condition clarified.

Added Capability Discovery description in section 5.3 and the CD bit and description in section 9.1.

Changed the Continuous Watchdog Availability bit abbreviation to WA from CA in section 9.1.

Added the Concatenate All description in section 7.7 and the CA bit and description in section 9.3.

**Revision Draft 0.7 (January 19, 2010)**

Updated headers and footers.

# 1 Scope and purpose

## 1.1 Scope

This specification supplements VersaPHY Additions to IEEE-1394. It provides necessary details to develop hardware and software applications to monitor and control of a broad range of GPIO-type devices.

GPIO (General Purpose Input/Output) is not a standardized interface. It is likely some GPIO-type devices will not be supportable by this protocol.

The electrical characteristics of GPIO are very diverse and not defined in this document. This protocol only describes communication of on/off state information. Voltage and current levels and other details of the on and off states are left to individual implementations.

The intent of VersaPHY profiles is for a wide range of profiles to provide highly optimized interfaces. This GPIO profile is more generic and suited to a wide range of on/off signaling requirements.

It is possible that some low speed synchronous protocols can be supported by this VersaPHY GPIO profile. This use is not prohibited but is not the intent of the design. Most synchronous protocols would likely be better served by a dedicated VersaPHY profile.

## 1.2 Purpose

This document defines a standardized method of monitoring and controlling GPIO devices over IEEE-1394 networks via the VersaPHY protocols.

Defining a standard for the GPIO interface promotes cross compatibility and allows reuse of hardware devices, application software, and driver infrastructure between compliant implementations.

# 2 Normative references

## 2.1 Reference scope

The specifications and standards named in this section contain provisions, which, through reference in this text, constitute provisions of this 1394 Trade Association Specification. At the time of publication, the editions indicated were valid. All specifications and standards are subject to revision; parties to agreements based on this 1394 Trade Association Specification are encouraged to investigate the possibility of applying the most recent editions of the specifications and standards indicated below.

## 2.2 Approved references

The following approved specifications and standards may be obtained from the organizations that control them.

IEEE Std 1394-1995, Standard for a High Performance Serial Bus

IEEE Std 1394a-2000, Standard for a High Performance Serial Bus—Amendment 1

IEEE Std 1394b-2002, Standard for a High Performance Serial Bus—Amendment 2

Throughout this document, the term “IEEE 1394" shall be understood to refer to IEEE Std 1394-1995 as amended by IEEE Std 1394a-2000 and IEEE Std 1394b-2002.

## 2.3 References under development

At the time of publication, the following referenced specifications and standards were under development.

VersaPHY Additions to IEEE-1394

## 2.4 Reference acquisition

The references cited may be obtained from the organizations that control them:

1394 Trade Association, 315 Lincoln, Suite E, Mukilteo, WA 98275 USA; (817) 416-2200 / (817) 416-2256 (FAX); http://www.1394ta.org/

American National Standards Institute (ANSI), 11 West 42nd Street, New York, NY 10036, USA; (212) 642-4900 / (212) 398-0023 (FAX); http://www.ansi.org/

Institute of Electrical and Electronic Engineers (IEEE), 445 Hoes Lane, PO Box 1331, Piscataway, NJ 08855-1331, USA; (732) 981-0060 / (732) 981-1721 (FAX); http://www.ieee.org/

In addition, many of the documents controlled by the above organizations may also be ordered through a third party:

Global Engineering Documents, 15 Inverness Way, Englewood, CO 80112-5776; (800) 624-3974 / (303) 792-2192; http://www.global.ihs.com/

# 3 Definitions and notation

## 3.1 Definitions

### 3.1.1 Conformance

Several keywords are used to differentiate levels of requirements and optionality, as follows:

**3.1.1.1 expected:** A keyword used to describe the behavior of the hardware or software in the design models assumed by this specification. Other hardware and software design models may also be implemented.

**3.1.1.2 ignored:** A keyword that describes bits, bytes, quadlets, octlets or fields whose values are not checked by the recipient.

**3.1.1.3 may:** A keyword that indicates flexibility of choice with no implied preference.

**3.1.1.4 reserved:** A keyword used to describe objects (bits, bytes, quadlets, octlets and fields) or the code values assigned to these objects in cases where either the object or the code value is set aside for future standardization. Usage and interpretation may be specified by future extensions to this or other specifications. A reserved object shall be zeroed or, upon development of a future specification, set to a value specified by such a specification. The recipient of a reserved object shall ignore its value. The recipient of an object defined by this specification as other than reserved shall inspect its value and reject reserved code values.

**3.1.1.5 shall:** A keyword that indicates a mandatory requirement. Designers are required to implement all such mandatory requirements to assure interoperability with other products conforming to this specification.

**3.1.1.6 should:** A keyword that denotes flexibility of choice with a strongly preferred alternative. Equivalent to the phrase “is recommended.”

### 3.1.2 Glossary

The following terms are used in this specification:

**3.1.2.1 VersaPHY:** Using the protocols specified in VersaPHY Extensions to IEEE-1394

### 3.1.3 Abbreviations

The following are abbreviations that are used in this specification:

| | |
|---|---|
| GPO | General Purpose Output |
| GPI | General Purpose Input |
| GPIO | General Purpose Input / Output |

## 3.2 Notation

### 3.2.1 Numeric values

Decimal, hexadecimal, and binary are used within this specification. By editorial convention, decimal numbers are most frequently used to represent quantities or counts. Addresses are uniformly represented by hexadecimal numbers. Hexadecimal numbers are also used when the value represented has an underlying structure that is more

apparent in a hexadecimal format than in a decimal format. Binary are used to represent register fields with lengths not evenly divisible by four.

Decimal numbers are represented by Arabic numerals without subscripts or by their English names. Hexadecimal numbers are represented by digits from the character set 0 – 9 and A – F followed by the subscript 16. When the subscript is unnecessary to disambiguate the base of the number it may be omitted. For the sake of legibility hexadecimal numbers are separated into groups of four digits separated by spaces.

Binary numbers are represented by digits 0 and 1 followed by the subscript 2. When the subscript is unnecessary to disambiguate the base of the number it may be omitted. For the sake of legibility binary numbers are separated into groups of four digits separated by spaces.

As an example, 42, $2A_{16}$, and 10 $1010_2$ all represent the same numeric value.

### 3.2.2 Bit, byte and quadlet ordering

This specification uses the facilities of Serial Bus, IEEE 1394, and therefore uses the ordering conventions of Serial Bus in the representation of data structures. In order to promote interoperability with memory buses that may have different ordering conventions, this specification defines the order and significance of bits within bytes, bytes within quadlets and quadlets within octlets in terms of their relative position and not their physically addressed position.

Within a byte, the most significant bit, *msb*, is that which is transmitted first and the least significant bit, *lsb*, is that which is transmitted last on Serial Bus, as illustrated below. The significance of the interior bits uniformly decreases in progression from *msb* to *lsb*.

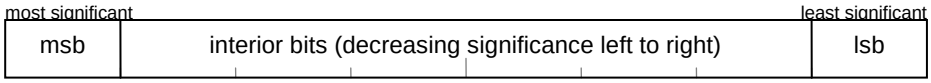


**Figure 1 – Bit ordering within a byte**

Within a quadlet, the most significant byte is that which is transmitted first and the least significant byte is that which is transmitted last on Serial Bus, as shown below.

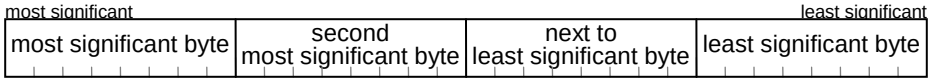


**Figure 2 – Byte ordering within a quadlet**

Within an octlet, which is frequently used to contain 64-bit Serial Bus addresses, the most significant quadlet is that which is transmitted first and the least significant quadlet is that which is transmitted last on Serial Bus, as the figure below indicates.

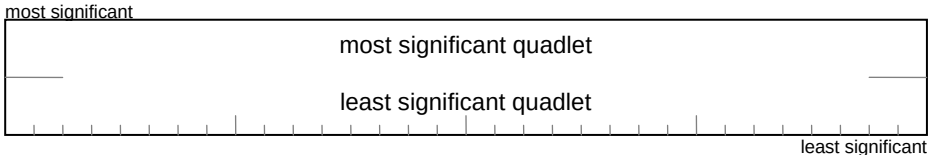


**Figure 3 – Quadlet ordering within an octlet**

When block transfers take place that are not quadlet aligned or not an integral number of quadlets, no assumptions can be made about the ordering (significance within a quadlet) of bytes at the unaligned beginning or fractional quadlet end of such a block transfer, unless an application has knowledge (outside of the scope of this specification) of the ordering conventions of the other bus.

# 4 Model (informative)

## 4.1 Model overview

The promise of the VersaPHY protocols is to easily enable access to IEEE-1394 networks for the simplest devices. The very simplest electronic devices are controlled by or provide information via steady state voltage levels labeled on and off (1 and 0).

Mechanical switches, comparators, timers, and other sensors can generate these signals. Relays, LED indicators, and motor controls use them. Though no universal standard exists, this interface is commonly called GPIO, General Purpose Input Output.

## 4.2 GPIO

Even in this age of dozens of powerful electronic interface protocols, many systems fall back to the lowest common denominator of an on/off signal. That seems unlikely to change because, at the end of the day, an on/off signal is all that is really necessary to monitor or control many physical devices. It can also be the least expensive.

The problem with these on/off (GPIO) signals is that they seem to multiply exponentially in sophisticated control systems, while fundamental GPIO control requires at least one conductor for each signal. This is where a sophisticated interface like IEEE-1394 can help by accumulating 1000's of GPIO signals over a network of four conductor interconnects. If the system has other requirements to move data, audio, or video, that same infrastructure may be used, making the cabling for GPIO devices essentially free.

The remaining issue is the cost of connecting GPIO signals to the IEEE-1394 network. The VersaPHY Protocol Extensions for IEEE-1394 and this VersaPHY GPIO Profile are optimized to provide the least complicated (and least expensive) method to convey GPIO information across IEEE-1394.

Standardizing a method further enables cost reduction through volume production of general purpose hardware solutions and providing off-the-shelf operating system drivers and application programming interfaces (APIs).

## 4.3 Minimum Requirements

This specification prefixes most "shall" requirements with an "if implemented" qualifier. The fundamental requirements are extremely minimal by design. Several presumably useful features are defined but not required. Typically, these enhancements are indicated by an availability bit or other method by which a controller can understand the implemented features.

The register set for a minimal VersaPHY GPIO device is shown below. This register set supports control of a single output bit write value (WV). This register set uses a static label and doesn't support any other VersaPHY or GPIO profile options. Many of the registers that are set to 0s are reserved for the extended feature set.

| Block | Offset | Bit | | | | | | | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | | 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 10 | 11 | 12 | 13 | 14 | 15 |
| 0 | 0 | 1 | 0 | $00\ 0000_2$ | | | | | | $00_2$ | | $11\ 1111_2$ | | | | | |
| 0 | 2 | 0 | 0 | $00_{16}$ | | | | | | | | $00\ 0111_2$ | | | | | |
| 0 | 4 | 1 | 1 | 0 | 0 | 0 | $01_2$ | | 0 | $00_{16}$ | | | | | | | |
| 0 | 6 | $80_{16}$ | | | | | | | | $00_{16}$ | | | | | | | |
| 0 | 8-E | $00_{16}$ | | | | | | | | $00_{16}$ | | | | | | | |
| 1 | 0 | $00_{16}$ | | | | | | | | $00_{16}$ | | | | | | | |
| 1 | 2 | $01_{16}$ | | | | | | | | $00_{16}$ | | | | | | | |
| 1 | 4-E | $00_{16}$ | | | | | | | | $00_{16}$ | | | | | | | |
| 2 | 0 | $00_{16}$ | | | | | | | | $00_{16}$ | | | | | | | |
| 2 | 2 | **WV** | $000\ 0000_2$ | | | | | | | $00_{16}$ | | | | | | | |
| 2 | 4-E | $00_{16}$ | | | | | | | | $00_{16}$ | | | | | | | |

**Figure 4 - Minimum GPO Register Set**

This implementation requires one functional register bit for the output bit. Of course additional registers are required for receiving and transmitting VersaPHY packets. This entire device can be implemented in 46-51 macro blocks in a CPLD connected to an IEEE-1394 PHY.

Obviously this minimal solution has very limited application, but additional features are offered a la carte, attempting to provide the required level of capability with minimum complexity. A custom IC developer may wish to implement all available options to reach the widest possible range of applications.

### 4.4 Controller Centric Systems

Many systems may use VersaPHY GPIO simply to collect data for a controller on the network and/or execute commands from a controller on the network. These systems can be implemented with simple generic VersaPHY GPIO devices that have dedicated GPI and GPO blocks as shown below.

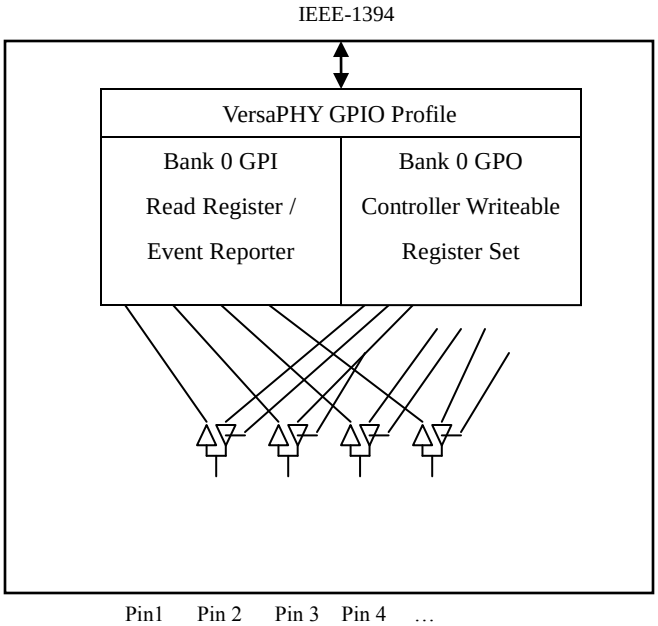


**Figure 5 - Controller Managed VersaPHY GPIO Device**

### 4.5 Advanced Systems

Other systems may need to replicate legacy controller-less GPIO systems. This requires GPO actions initiated directly from GPI responses.

The VersaPHY GPIO profile introduces a third concept in addition to standard GPI and GPO logic. The General Purpose Listener (GPL) is a logic block that is configured to listen for GPI information from another device across the network. This enables outputs on one VersaPHY GPIO device to respond to input information from another VersaPHY GPIO device across an IEEE-1394 network without any sort of controller.

Only the simplest systems link single inputs to single outputs. More typically, additional logic and state machines are required to collect several inputs an determine an appropriate output. The diagram below depicts how this can be implemented in advanced VersaPHY GPIO devices.

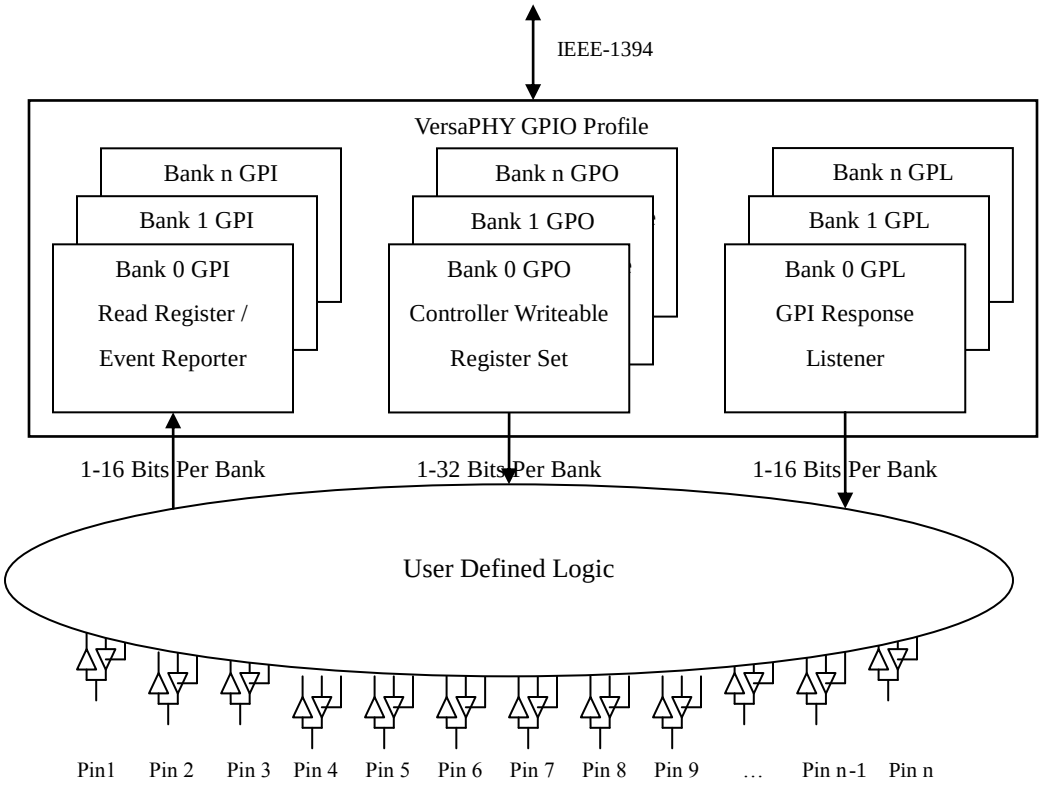


**Figure 6 – VersaPHY GPIO Device for Advanced Systems**

Developing and configuring the "User-Defined Logic" is outside the scope of this document. The logic may be fixed for a given implementation, or it may be a cross point switch providing only interconnect functionality. Field Programmable Gate Array (FPGA) devices may be used for VersaPHY GPIO allowing the User-Defined Logic to be compiled with the VersaPHY GPIO core logic and programmed into a custom device.

Advanced systems do not preclude the use of controllers. Controllers may be used to configure advanced devices or to manage or monitor selected processes.

### 4.6 GPIO Device/Function Discovery

Ad hoc automated discovery of GPIO devices is problematic because GPI reporting devices may only report their on/off state. GPO controlled devices may not report anything.

The VersaPHY GPIO interface may proxy identification information with a Static VersaPHY label or with the optional GUID and Model_ID fields. Actual device identification methods are left to the application developer, but, typically, pre-assignment for fixed networks or first-time manual identification on ad hoc networks will typically be used.

### 4.7 Latency

It is assumed that GPIO information is relayed between the GPIO device and the controller as quickly as possible, but exact delays cannot be specified. The total delay is the sum of controller action delays, IEEE-1394 network delays, and GPIO device delays.

IEEE-1394 network delays can vary greatly depending on the number of IEEE-1394 nodes on the network, the maximum size of their asynchronous packets, the amount of isochronous bandwidth allocated, the bit rate (S100-S3200), and the arbitration mechanism (1394a or 1394b).

The small size of VersaPHY packets enables latencies of less than 1µs in specific cases. The worst case for large numbers of nodes sending large packets with high isochronous utilization can be up to 62ms. The fair arbitration mechanisms of IEEE-1394 ensure that VersaPHY packets cannot be delayed indefinitely.

### 4.8 Error Handling

#### 4.8.1 GPIO Device Removed

If a GPIO device loses power or is otherwise removed from the IEEE-1394 bus, a bus reset is automatically issued by the PHY(s) to which it was connected. After a bus reset notification, the controller can poll for remaining devices. If the VersaPHY base register B bit is supported and set on the GPIO device, remaining GPIO devices will issue an unsolicited response packet indicating their presence.

#### 4.8.2 GPIO Device Inactive

The periodic update features of the GPIO profile may be used in conjunction with timers in the controller to spot inactive GPIO devices that have not left the bus.

Custom heart beat, health status, or custom counter logic may also be attached to input bits in a GPIO function bank. If automatic event updates are enabled, an unsolicited response will be issued every time these bits change.

#### 4.8.3 Controller Removed

The VersaPHY W bit enables a watchdog timer that begins after every bus reset and resets the GPIO function if it expires before a controller resets it.

#### 4.8.4 Controller Inactive

The GPIO profile offers an extension to the watchdog timer. The CW bit enables the watchdog function to continuously count after every time it is reset. This enables the GPIO profile bank to reset if a controller becomes inactive but remains on the bus. Refer to the watchdog timer feature for details.

#### 4.8.5 Packet Errors

Responses to all read and write requests provide confirmation to controllers that the GPIO device received and validated each request.

Each GPIO VersaPHY packet may be validated by an 8-bit CRC in the last byte of the packet. VersaPHY controllers may analyze response packets with invalid CRC's and attempt to re-read corrupt unsolicited responses.

#### 4.8.6 Unreported Transitions

It is possible that multiple transitions occur before each one is reported by the GPI function. The likelihood of this event depends on the switching frequency of the input, debounce characteristics, as well as many IEEE-1394 bus-related parameters.

The transition overflow may be used to indicate when more than one transition occurred between the last response (solicited or unsolicited) and the current response (solicited or unsolicited). The transition overflow feature description provides more details.

### 4.9 Response Queuing

VersaPHY GPIO profile implementations may choose their own method for queuing responses. The GPIO application developer is responsible for understanding how a given device queues responses and the effect of that method on the application.

The simplest GPIO devices may queue only one response, expecting to respond before an additional request arrives.

Other GPIO devices may only retain one request for each register along with the tlabel and request type. In this case if any combination of read requests, write requests, or internal unsolicited requests are made to the same register before a response opportunity is granted, only the last one will be sent and it will be sent with the current register value. All the requestors are allowed to use this single response. It may be important for some requestors to note if the tlabel does not match their request.

Another option for more complex GPIO devices is to queue all requests along with the register value at the time of the request into a FIFO. When a grant is received, the FIFO can be emptied into concatenated responses. This method will require management of potential FIFO overflow and may increase bus usage.

Other implementations are permissible.

# 5 GPIO Profile Features

## 5.1 Grouping

To effectively support multiple GPIO signals, the GPIO profile groups signals in two levels: top and second.

The top level is the VP-Function level. As noted previously, there are many options in the GPIO profile. A given set of options shall apply to all GPIO bits in a given VP-Function. Logically, all the settings and capabilities represented in the first two register blocks apply to all the bits in that GPIO function. Since a single VP-Device can support many VP-Functions, if an application requires different feature sets for its attached GPIO devices, multiple GPIO VP-Functions may be implemented.

The second level is the bank level. GPIO banks contain 1-16 bits. This enables monitoring and controlling several bits with a single 16-bit payload VP-Packet. A single GPIO VP-Function can support up to 254 banks. Most available GPIO features are controlled at the bank level. Logically, all the settings represented in the register block for each bank apply only to all the GPIO bits in that bank.

For example, a feature like time stamping (if implemented) must be available to all the banks in a VP-Function, but each bank will maintain its own timestamp value (if enabled). If two GPIO bits each require independent timestamps, they should be arranged in separate banks.

GPI banks, GPO banks, and device pins are not required to be linked together. For example, GPI Bank 0 and GPO Bank 0 are not required to be tied directly to I/O pins 1-16. They may be configured that way if it suits the application.

## 5.2 Custom Registers

There are several sections of register bits reserved for custom features.

The Vendor_Defined register should be used to report or implement features that apply to the entire GPIO function.

The VOBank_Options, VIBank_Options, and VLBank_Options registers should be used to report or implement features that apply to all the bits in their respective GPO, GPI, or GPL bank.

The VOBit_Option, VIBit_Option, and VLBit_Option registers should be used to report or control features that apply to each bit in the respective GPO, GPI or GPL bank bits.

The VIInfo field should be used to report a capability or status for the GPI bank.

No effort is made here to list all or exclude any vendor-defined possibilities. No custom features are required. Vendor_Defined or VBank_Options might be used to transfer register settings to non-volatile memory or link banks of GPIO. VBit_Option registers might be used to group bits or possibly XOR set bit fields if Set_Bits and Clear_Bits does not suffice.

Specific groupings of custom features should be linked to a specific Model_ID value to facilitate discovery and controller model driver selection.

## 5.3 Capability Discovery

The implementation of specific GPIO bits and features may be discovered by a controller if Capability Discovery is implemented. The CD bit in the GPIO register set controls the Capability Discovery mode. If Capability Discovery is not supported, the CD bit shall respond with a 0 after a write request setting it to 1.

When the CD bit is set, read requests to GPI, GPO, and GPL registers will produce responses indicating the availability of the bits and features in those registers, not the actual register values. A 1 shall indicate the bit is available, a 0 shall indicate the bit is not available. The definition of available is left to the developer. Available may only mean the register and its corresponding pin are implemented in hardware. In more sophisticated implementations it may mean the pin is actually controlling or under control of an attached device. When an availability bit is 0, it should be considered unusable in any event.

When Capability Discovery is not implemented, the actual GPIO bits and features in question may be written and read to discover if they are useable. Bits and features that are not implemented are read only and shall not respond to writes.

# 6 General Purpose Output Features

## 6.1 Output Enable

Each VersaPHY GPIO device pin may be an input, an output, or a bi-directional pin. This is managed by an output enable for each pin. When the GPO output enable mask is set to 1, the output driver is tri-stated and the external input signal may be registered in a GPI bank.

When the mask is set to 0, the output is enabled and may be fed back into the GPI register where it can be read as feedback.

The OE_Mask register can also be used to toggle between a driven state and a not driven state as required by some systems. Pull up/down requirements are left to specific applications.

Signal pin rich applications without bi-directional requirements may offer separate inputs and outputs without bi-directional capability.

## 6.2 Watchdog Timer

When the GPO bank watchdog timer expires, the OE_Mask field shall be set to the value contained in the Reset_Mask field and the Write_Value field shall be set to the value contained in the Reset_Value field.

Three register fields control the GPO Watchdog Timer.

The Watchdog Timer (W_Timer) field specifies the length of the watchdog counter. The field is 5 bits providing 31 timer lengths and a disabled state ($0\ 0000_2$). Table 5 -- Timer Delays lists the delays of each setting. The delays follow the bits used for the IEEE-1394 cycle timer and range from 40ns to 32 seconds, mostly in powers of two. Implementing all settings is not required. If settings are omitted, the register shall only accept values in the implemented range. Values outside of the range may be modified to the nearest acceptable value before the write response is issued. If only a single delay setting is implemented, the W_Timer should report the value nearest that fixed delay. The W_Timer is a configuration setting and shall not update as the timer counts.

If finer resolution of delays is required the Watchdog Count (WCnt) field may be implemented. WCnt represents the number of times the W_Timer field is repeated before the final trigger. Only 1x, 3x, 5x, and 7x multipliers are available for the WCnt field. These multipliers are coded into WCnt as represented in Table 6 -- Timer Multipliers. The WCnt field shall not change to reflect the current count state.

The Watchdog Mode (WM) bit sets the GPO bank watchdog mode. When WM is set to 0 and Watchdog bit (W) is set to 1, at the beginning of each bus reset process, W shall be set to 0 and the watchdog timer shall be initialized and started. If W is set back to 1 before the watchdog timer expires, the watchdog timer shall be stopped. This provides a GPO reset function if a controller is unplugged or a bus reset storm interrupts communication.

When WM is set to 1, the watchdog timer shall always be enabled and shall be restarted after every write to the GPO bank's Write_Value, Set_Bits, or Clear_Bits fields. This provides a GPO reset function if a frozen controller doesn't access the GPO block within the watchdog time. Reads shall not restart the watchdog; so a controller may read Write_Value to understand if the watchdog has expired.

WM may be fixed if both modes are not supported.

The watchdog timer may also be used to generate GPO pulses. If the "after the pulse" state is set in the Reset_Value field and the pulse length is set using the W_Timer and WCnt fields, when a pulse value is written to the Write_Value field it will last until the watchdog timer expires and then be replaced with the previously entered Reset_Value (provided Write_Value, Set_Bits, and Clear_Bits are not written before the watchdog expires).

Generating pulses might make the very short and very long timer values useful.

### 6.3 Set / Clear

If set and clear are implemented, any bits written to 1 in the Set_Bits register will set the corresponding bits in the Write_Value register to 1. Likewise, any bits written to 1 in the Clear_Bits register will clear the corresponding Write_Value bits to 0. Bits written to 0 in the Set_Bits or Clear_Bits registers shall have no effect on the Write_Value register.

Write requests to Set_Bits and Clear_Bits should respond with the value that was written. Read requests to Set_Bits and Clear_Bits should respond with all 0s. This enables controllers to change an individual GPO bit in a bank without first masking all the other bits. Separate controllers may each manage bits within a bank without concern for modifications of other bits by other controllers.

If set and clear are not implemented, a controller will have to use a read – modify – write process that takes more time and bandwidth and can also be less reliable.

# 7 General Purpose Input Features

## 7.1 GPI Update Notification

### 7.1.1 Methods

There are three methods for interested devices to receive GPI update notifications: Polling, Event Updates, and Periodic Updates.

### 7.1.2 Polling

All VersaPHY GPIO devices shall support polling via VersaPHY read request packets addressed to the Read_Value field. Any interested device may listen to the read responses resulting from requests issued by other devices. Polling frequency is managed by the application and controller.

### 7.1.3 Event Updates

If the EU bit is enabled in the GPIO function and set to 1 for a given bank, the GPIO function shall issue an unsolicited read response VersaPHY packet for the Read_Value register every time an input bit in the bank changes. The EU bit feature description provides more details.

If the trigger field is implemented and TM=0, changes to GPI bits corresponding to bits set to 1 in the Trigger field shall not initiate an unsolicited response.

If the Trigger field is implemented and TM=1, unsolicited response shall not be made unless the new value of Read_Value equals the value in the Trigger field.

### 7.1.4 Periodic Updates

If the Update Timer is enabled in the GPIO function and set for a given bank, the function shall issue an unsolicited response for the input bits of that bank every time the Update Timer expires.

The update timer shall run continuously when enabled. Each time the update timer expires the GPIO function bank shall issue an unsolicited response containing the value of the Read_Value Register. If the Read_Value register has changed since the last response it shall issue a write response; if Read_Value is unchanged since the last response it shall issue a read response.

The update timer U_Timer is configured the same way as the watchdog timer. The U_Timer field represents a delay as specified in Table 5 Timer Delays. There is no multiplier field specified for the update timer. The U_Timer is a setting and shall not change to reflect the timer state. Again, all settings are not required. Write values may be modified to the nearest permissible value. Implementations with a single value should represent the closest value within the U_Timer field.

The update timer may be used to periodically indicate the presence of the GPIO device to the bus or provide scheduled updates of current GPI values.

## 7.2 Timestamp

In some GPI applications, when the signal change information arrives is less important than knowing when the signal change occurred. This specification includes optional timestamping features to support these applications.

GPIO Timestamping is based on the IEEE-1394 cycle timer features used for isochronous data transmission. To use GPIO timestamping, an active cycle master-capable node must be present on the bus and sending cycle start packets.

Cycle master features may be present in a VersaPHY node, however the requirements are not included in this specification.

Timestamping-capable GPIO functions shall collect the 32 bit cycle time information and continue their own counts between updates. Cycle time counters are all based on a 24.576 MHz clock. When a change occurs to an input in a time stamping bank, the cycle time counter shall be registered in the bank's Timestamp register.

The Timestamp register can be read with a read request. If CT0 or CT1 are set to 1, their corresponding Timestamp registers shall be automatically concatenated to all Read_Value register responses. CT0 represents the most significant 16 Timestamp bits, CT1 represents the least significant 16 bits. Typically, timestamped unsolicited responses only require the least significant bits. The most significant bits can usually be recovered from the controller's own cycle time counter.

Timestamping of bit sets that do not all transition simultaneously may be gated with the debounce feature detailed in section 7.4.

### 7.3 Timestamp Overflow

If timestamp information is not broadcast soon enough, there is potential for the timestamp to become invalid. Optional timestamp overflow bits are specified to indicate this condition.

TSO bit 0 indicates an overflow of the most significant two Timestamp bytes (a 64 second overflow). TSO bit 1 indicates an overflow of the least significant two Timestamp bytes (a 1 ms overflow). The overflow is registered when the current time equals the Timestamp. This indicates the range of the Timestamp has been exceeded. The current time may pass the zero value without effecting the overflow bits.

Timestamp overflow information can be automatically reported using the concatenated metadata feature defined in section 7.7.

### 7.4 Debounce

Digital counter debouncing of input signals enables VersaPHY GPIO devices to connect directly to mechanical switches. It can also be used to qualify minimum pulse lengths.

Three register fields are defined to control input debounce. They are very much like the GPO watchdog timer fields.

The D_Timer field specifies the length of the debounce counter. The field is 5 bits, providing 31 timer lengths and a disabled state ($00000_2$). Table 5 -- Timer Delays lists the delays of each setting. The delays follow the bits used for the IEEE-1394 cycle timer and range from 40ns to 32 seconds, mostly in powers of two. Implementing all settings is not required. If settings are omitted, the register shall only accept values in the implemented range. Values outside of the range may be modified to the nearest acceptable value before the write response is issued. If only a single delay setting is implemented, the D_Timer should report the value nearest that fixed delay. The D_Timer is a configuration setting and shall not update as the timer counts.

If finer resolution of delays is required, the DCnt field may be implemented. DCnt represents the number of times the D_Timer field is repeated before the final trigger. DCnt does not directly represent the required count. Only 1x, 3x, 5x, and 7x multipliers are available. These multipliers are coded into DCnt as represented in Table 6 -- Timer Multipliers. The DCnt field shall not change to reflect the current count state.

The DM field sets the debounce mode. When DM is set to 0, the first detected transition of a bit will be recorded in the input register. When that bit is latched, the unsolicited write response process will proceed, if enabled. After the bit is latched in the register, the debounce timer will begin and no further updates will be allowed to that GPI register bank until the count is complete. This can prevent multiple unsolicited write responses being issued for bouncy switches or oscillating A/D values. It also reports the first transition as quickly as possible.

When DM is set to 1, the counter starts when a bit transition is detected. When the counter expires, the current state of the inputs is latched. If any bits have changed from the previous latched settings, an unsolicited write response shall proceed if EU=1. If all bits in the bank have retained their previous setting, no change is registered or processed.

This setting has several potential benefits. It can filter out glitches. It can detect pulse lengths. It can also allow time for bussed bits to settle, so that only one update is required for a group of bits. The change will be reported (and timestamped) later than DM=0, but the offset will be fixed.

Debouncing for 40 ns or 224 seconds may have no application. These ranges may have some benefit for glitch filtering or pulse measurement.

### 7.5 Input Overflow

It is possible that every GPI bit transmission will not be reported to the bus. To detect this condition, the OV bit may be implemented.

If implemented, the OV bit shall be set to one when more than one change is made between VersaPHY packet responses of the Read_Value register to bits within a bank. The OV bit shall be cleared after a Read_Value response is completed (including concatenated metadata).

The lost bit transitions are not identified, but an indication that unreported transitions occurred can be reported to the bus. Input overflow can be automatically reported using the CO concatenated metadata feature detailed in section 7.7.

### 7.6 Heartbeat

When implemented, the Heartbeat field shall increment by one for each response from that GPI block's Read_Value field. This provides an indication that the GPI block is functioning and can enable a controller to identify if responses made by the GPI block were missed by the controller.

The Heartbeat field is placed in a register with the overflow fields and is treated like an overflow indication for the purposes of concatenated metadata.

### 7.7 Concatenated Metadata

Several GPI register fields contain metadata about the GPI inputs that users may wish to have automatically transferred along with Read_Value Responses.

The CO, CT0, CT1 and CV fields control concatenation of the Overflow, Timestamp MSBs, Timestamp LSBs, and VIBit_Option registers, respectively.

If implemented, when these control bits are set to 1, the GPIO function shall concatenate the appropriate registers to all responses of the Read_Value field. If no overflow bits are set and the Heartbeat field has not changed, the overflow registers shall not be concatenated.

If implemented, then the CA field is set to 1, s Read_Value response (and their indicated metadata) from all other GPI blocks shall be concatenated after the current block Read_Value (and indicated metadata) response. This permits an event on a single block to initiate the automatic transfer of all GPI block Read_Values.

If implemented, when the CN field is set to 1, a Read_Value response (and its indicated metadata) from the next GPI block shall also be concatenated after the current block Read_Value (and indicated metadata) response. This permits an event on a single block to initiate the automatic transfer of multiple sequential GPI block Read_Values. If CA and CN functionality are both implemented, and both bits are set to 1, CN behavior is ignored.

# 8 General Purpose Listener Features

## 8.1 Listener Target

Each GPL bank listens for responses from another block in a VersaPHY GPIO function. The exact block to which it listens is configured with the CC, RR, Bus_ID/Cont_ID, Label, and Block fields.

All of the following conditions must be met for the GPL block to respond to a given response packet:

If RR is set to 1, the GPL bank shall respond to VP-Packets when the R_W field indicates Read (1) or Write (0).

If RR is set to 0, the GPL bank shall respond to VP-Packets only when the R_W field indicates Write (0).

GPL banks are not required to monitor the tlabel field of VP-Packets; so they are not required to distinguish between solicited and un-solicited response packets.

If CC is set to 1, the GPL bank shall respond to VP-Packets when the Bus_ID/Cont_ID field section of the Source_VP_Label in the VP-Packet matches the Bus_ID/Cont_ID section of the VP_Label of the GPIO function containing the GPL bank. This enables the GPL to listen to devices labeled under the control of the GPIO functions controller. Controller changes or bus changes need only be applied to the GPIO function VP-Label; GPLs with CC=1 need not be reconfigured.

If CC is set to 0, the GPL bank shall respond to VP-Packets when the Bus_ID/Cont_ID field section of the Source_VP_Label in the VP_Packet matches the Bus_ID/Cont_ID field of the GPL bank. This enables the GPL to listen to banks anywhere on the network, at the expense of more complicated configuration (and re-configuration when necessary).

The GPL bank shall respond to VP-Packets when the Label field section of the Source_VP_Label in the VP-Packet matches the Label field of the GPL block.

The GPL bank shall respond to VP-Packets when the Block field in the VP_Packet matches the Block field of the GPL Block. Blocks 0, 1, 2, and 3 are invalid as they are the profile registers for GPIO functions. A GPL may listen to responses from a GPI Bank, a GPO Bank, or another GPL Bank.

The GPL bank shall only act on responses from offset 0-1 of the specified bank. These offsets contain the Write_Value fields for GPO banks, the Read_Value fields for GPI banks, and the Current_Value fields for GPL banks.

## 8.2 Controller Writes

Controllers may write directly to the Current_Value field. GPL banks shall accept writes to their Current_Value field. This permits initialization and allows GPL banks to function as limited GPO banks.

## 8.3 Watchdog

GPL blocks shall implement Reset_Value and watchdog fields (WM,WCnt, and W_Timer) in the same manner as the identically labeled fields in the GPO watchdog section of this specification.

## 8.4 Update Mask

When any bits are set to 1 in the Update_Mask field, the corresponding Current_Value bits shall not be updated by listener-validated response packets.

### 8.5 Automatic Updates

When EU=1, the GPL bank shall issue an unsolicited response when listener changes are made to the Current_Value register. This provides verification that appropriate packets were heard and provides a chain reaction process for other GPL banks listening for responses from the current GPL.

# 9 Register Definitions

## 9.1 GPIO Base Register Set

| Block | Offset | Bit | | | | | | | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | | 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 10 | 11 | 12 | 13 | 14 | 15 |
| n | 0 | E | I | VP-Label | | | | | | | | | | | | | |
| n | 2 | W | B | reserved | | | | | | | | Power | | | | | |
| n | 4 | V | S | r | r | O | ID_type | | G | Instance_nOffset | | | | | | | |
| n | 6 | $80_{16}$ | | | | | | | | GPIO_Revision | | | | | | | |
| n | 8 | GUID [0:7] | | | | | | | | GUID [8:15] | | | | | | | |
| n | A | GUID [16:23] | | | | | | | | GUID [24:31] | | | | | | | |
| n | C | GUID [31:39] | | | | | | | | GUID [40:47] | | | | | | | |
| n | E | GUID [48:55] | | | | | | | | GUID [56:63] | | | | | | | |
| n+1 | all | Additional blocks Physical ID access register blocks are permitted but not required.<br>If implemented, additional blocks shall match the corresponding VP-Label access blocks. | | | | | | | | | | | | | | | |

**Figure 7 – GPIO Base Registers for Physical_ID Access**

| Block | Offset | Bit | | | | | | | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | | 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 10 | 11 | 12 | 13 | 14 | 15 |
| 0 | 0 | E | I | Owner | | | | | | 0 | 0 | Physical_ID | | | | | |
| 0 | 2 | W | B | reserved | | | | | | | | Power | | | | | |
| 0 | 4 | V | S | r | r | O | ID_type | | G | Instance_nOffset | | | | | | | |
| 0 | 6 | $80_{16}$ | | | | | | | | GPIO_Revision | | | | | | | |
| 0 | 8 | GUID [0:7] | | | | | | | | GUID [8:15] | | | | | | | |
| 0 | A | GUID [16:23] | | | | | | | | GUID [24:31] | | | | | | | |
| 0 | C | GUID [31:39] | | | | | | | | GUID [40:47] | | | | | | | |
| 0 | E | GUID [48:55] | | | | | | | | GUID [56:63] | | | | | | | |
| 1 | 0 | Model_ID [0:7] | | | | | | | | Model_ID [8:15] | | | | | | | |
| 1 | 2 | GPOBanks [0:7] | | | | | | | | GPIBanks [0:7] | | | | | | | |
| 1 | 4 | GPLBanks [0:7] | | | | | | | | CD | r | OA | UA | WA | UC | TS | SC |
| 1 | 6 | Reserved | | | | | | | | Reserved | | | | | | | |
| 1 | 8 | Vendor_Defined [0:7] | | | | | | | | Vendor_Defined [8:15] | | | | | | | |
| 1 | A | Vendor_Defined [16:23] | | | | | | | | Vendor_Defined [24:31] | | | | | | | |
| 1 | C | Vendor_Defined [31:39] | | | | | | | | Vendor_Defined [40:47] | | | | | | | |
| 1 | E | Vendor_Defined [48:55] | | | | | | | | Vendor_Defined [56:63] | | | | | | | |

**Figure 8 -- GPIO Base Registers for VersaPHY Label Access**

**Table 1 -- Base Register Definitions**

| Field | Bus Access | Value | Definition |
|---|---|---|---|
| E | * | | Enabled – As defined in *VersaPHY Extensions to IEEE-1394*. May be fixed to 1 if device can always be enabled. |
| I | * | | Immediate Response Enable– As defined in *VersaPHY Extensions to IEEE-1394*. May be fixed to 0 if device is unable to make immediate responses to VP-Label based requests. |
| VP-Label | * | | VersaPHY Label– As defined in *VersaPHY Extensions to IEEE-1394*. May be any valid VP-Label; static, orphan, or controller assigned. |
| Physical_ID | RO | | IEEE-1394 Physical_ID value of the node containing the function as collected during Self ID phase of last bus reset. Simple devices with static VP-Labels in pre-configured systems not requiring Physical_ID based discovery may use a constant $111111_2$ as their Physical_ID. These devices shall not respond to Physical_ID based requests. |
| W | * | | Watchdog bit– As defined in *VersaPHY Extensions to IEEE-1394*, except when modified by the CW field bit described in the VP-Label access register set. May be fixed to 0 if watchdog timers are not supported. |
| B | * | | Bus Reset Response– As defined in *VersaPHY Extensions to IEEE-1394*. May be fixed to 0 if bus reset responses are not supported. |
| Power | * | | Power field– As defined in *VersaPHY Extensions to IEEE-1394* Note: Some GPI devices requiring only power state signaling may be implemented with just the Power field functionality. May be fixed at $00111_2$ if power control functionality is not supported. |
| V | RO | | Valid – As defined in *VersaPHY Extensions to IEEE-1394*. May be fixed to 1 if function is always valid. |
| S | RO | | Self Enabled– As defined in *VersaPHY Extensions to IEEE-1394*. If V is fixed to 1, S shall also be fixed to 1. |
| O | RO | | Owner Field Available -- As defined in *VersaPHY Extensions to IEEE-1394*. Fixed to 0 if Owner field is not supported. |
| ID_Type | RO | $01_2$ | Identification Type– As defined in *VersaPHY Extensions to IEEE-1394*, set to $01_2$ for all GPIO devices. This invokes the shortest identification fields to simplify GPIO devices. |
| G | RO | | GUID available – As defined in *VersaPHY Extensions to IEEE-1394*. Set to 1 if a valid GUID is available in reserved GUID field. |
| Instance _nOffset | RO | | Instance ID and Offset of next VP-Function– As defined in *VersaPHY Extensions to IEEE-1394*. Set to $00_{16}$ for single function VP-Devices. |
| Short_ Config_Ver | RO | $80_{16}$ | Short Configuration Version– As defined in *VersaPHY Extensions to IEEE-1394*. $80_{16}$ is reserved to identify VersaPHY GPIO functions. |
| GPIO_ Revision | RO | $00_{16}$ | GPIO Revision – Set to $00_{16}$ for devices compliant with this revision of the GPIO profile specification. |
| GUID | RO | | Globally Unique Identifier – As defined in *VersaPHY Extensions to IEEE-1394*. Combination of a 32 bit IEEE assigned Organizational Unique Identifier and a 32 bit unique identifier assigned by that organization. |

| Field | Bus Access | Value | Definition |
|---|---|---|---|
| Model_ID | RO | | Model Identifier – Used for discovery of specific models of VersaPHY GPIO devices. Assigned by the organization identified by the OUI field of the GUID. |
| GPOBanks | RO | | GPOBanks – Indicates the number of available GPO banks. The first GPO bank will be located in the third block (2) if GPO is implemented. Additional GPO banks shall be located in contiguous following blocks. |
| GPIBanks | RO | | GPIBanks – Indicates the number of available GPI banks. The first GPI bank will be located at block (2 + GPOBanks) if implemented. Additional GPI banks shall be located in contiguous following blocks. |
| GPLBanks | RO | | GPLBanks – Indicates the number of available GPL banks. The first GPL bank shall be located at block (2 + GPOBanks + GPIBanks) if implemented. Additional GPL banks shall be located in contiguous following blocks. |
| CD | RW | | Capability Display Mode – If implemented, when C is set to one, all bits in all following GPIO registers will respond to read requests with an indication of their availability. C must be set to 0 for normal read response functionality. See Capability feature. |
| OA | RO | | Overflow Indication Available – Indicates an operational overflow bit is available for all banks when set to 1. See Overflow feature. |
| UA | RO | | Unsolicited Responses Available – Indicates all banks can produce unsolicited updates when set to 1. EU bit for individual banks must also be set to 1 to enable updates for that bank. See Unsolicited Responses feature. |
| WA | RO | | Continuous Watchdog Available – Indicates watchdog timers are available for all banks when set to 1. See Watchdog feature. |
| UT | RO | | Update Timer Available – Indicates automatic update timers are available for all banks when set to 1. See Update Timer feature. |
| TS | RO | | Timestamps Available – Indicates time stamp registers are available for all banks when set to 1. See Timestamp feature. |
| SC | RO | | Set and Clear Register Available –Indicates set and clear registers are operational for all banks when set to 1. |
| Vendor_ Defined | * | | Vendor-Defined Function Registers – Reserved to allow developers space to enable custom features or report custom status information for the entire GPIO VP-Function. See custom registers feature. |

### 9.2 General Purpose Output Register Set

| Block | Offset | Bit | | | | | | | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | | 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 10 | 11 | 12 | 13 | 14 | 15 |
| 2 | 0 | Write_Value [0:7] | | | | | | | | Write_Value [8:15] | | | | | | | |
| 2 | 2 | OE_Mask [0:7] | | | | | | | | OE_Mask [8:15] | | | | | | | |
| 2 | 4 | Reset_Value [0:7] | | | | | | | | Reset_Value [8:15] | | | | | | | |
| 2 | 6 | Reset_Mask [0:7] | | | | | | | | Reset_Mask [8:15] | | | | | | | |
| 2 | 8 | Set_Bits [0:7] | | | | | | | | Set_Bits [8:15] | | | | | | | |
| 2 | A | Clear_Bits [0:7] | | | | | | | | Clear_Bits [8:15] | | | | | | | |
| 2 | C | VOBit_Option [0:7] | | | | | | | | VOBit_Option [8:15] | | | | | | | |
| 2 | E | VOBank_Options[0:3] | | | | Reserved | | | | WM | WCnt[0:1] | | W_Timer [0:4] | | | | |

**Figure 9 – GPO Registers**

**Table 2 -- GPO Register Definitions**

| Field | Bus Access | Definition |
|---|---|---|
| Write_Value | R/W | Write Value – Sets the output level of the corresponding GPIO bit. See Input / Output feature. |
| OE_Mask | R/W | Output Enable Mask – Disables the output driver for the corresponding GPIO bit when set to 1. See Input / Output feature. |
| Reset_Value | R/W | Write Value Applied on Reset – If the watchdog timer is operational and expires, the value of Reset_Value is copied to the Write_Value register. See watchdog feature. |
| Reset_Mask | R/W | Input / Output Mask Applied on Reset – If the watchdog timer is operational and expires, the value of Reset_Mask is copied to the IO_Mask register. See watchdog feature. |
| Set_Bits | W | Set Output Bits – If Set_Bits is implemented, when any bit of this field is written to 1, the corresponding output bit shall be set to 1. Bits written to 0 shall have no effect on the output bits. See Set / Clear feature section 6.3. |
| Clear_Bits | W | Clear Output Bits – If Clear_Bits is implemented, when any bit of this field is written to 1, the corresponding output bit shall be set to 0. Bits written to 0 shall have no effect on the output bits. See Set / Clear feature section 6.3. |
| VOBit_ Option | * | Vendor-Defined Bit Option 1 – This field is reserved for developers to implement a custom bit-by-bit feature. See Custom Registers. |
| VOBank_ Options | * | Vendor-Defined Bank Options -- This field is reserved for developer to implement custom bank features or report optional bank information. See Custom Registers. |
| WM | R/W | Watchdog Mode – See Watchdog Timer |
| WCnt | R/W | Watchdog Timer Multiplier -- Offers finer resolution for the Watchdog timer by repeating the timer a specified number of times. See Table 6 -- Timer Multipliers for count values. All settings are not required. See Watchdog Timer. |
| W_Timer | R/W | Watchdog Timer – Specifies the base delay for watchdog actions. See Table 5 -- Timer Delays for actual delays. All settings are not required. Set to 0 to disable watchdog actions. See Watchdog Timer . |

### 9.3 General Purpose Input Register Set

| Block | Offset | Bit | | | | | | | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | | 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 10 | 11 | 12 | 13 | 14 | 15 |
| o+2 | 0 | Read_Value [0:7] | | | | | | | | Read_Value [8:15] | | | | | | | |
| o+2 | 2 | Heartbeat[0:3] | | | | VIInfo[0:3] | | | | reserved | | | | | TSO[0:1] | | OV |
| o+2 | 4 | Timestamp [0:7] | | | | | | | | Timestamp [8:15] | | | | | | | |
| o+2 | 6 | Timestamp [16:23] | | | | | | | | Timestamp [24:31] | | | | | | | |
| o+2 | 8 | Trigger [0:7] | | | | | | | | Trigger [8:15] | | | | | | | |
| o+2 | A | VIBit_Option [0:7] | | | | | | | | VIBit_Option [8:15] | | | | | | | |
| o+2 | C | reserved | | | | | | TM | EU | res | | CA | CN | CV | CO | CT [0:1] | |
| o+2 | E | VIBank_Options[0:3] | | | | U_Timer [0:4] | | | | DM | DCnt[0:1] | | D_Timer[0:4] | | | | |

**Figure 10 -- GPI Registers**

**Table 3 -- GPI Register Definitions**

| Field | Bus Access | Definition |
|---|---|---|
| Read_Value | RO | Read Value – Represents the current state of the GPIO input pins. See input / output feature. |
| Heartbeat | RO | Heartbeat – Increments after every Read_Value response. See Heartbeat. |
| VI_Info | RO | Custom GPI Information – See Custom Registers. |
| TSO | RO | Timestamp Overflow – Indicates the timestamp register has overflowed since the last response from Read_Value. See Timestamp Overflow. |
| OV | RO | Input Change Overflow – Set to 1 by the GPIO function if the Read_Value has changed more than once since the previous Read_Value response. Set to 0 after each Read_Value response. See Input Overflow. |
| Timestamp | RO | Timestamp -- Indicates system time of last input change. See Timestamp. |
| Trigger | R/W | Trigger – Provides bit masking and bit matching features for automatic unsolicited responses. See Event Updates. |
| VIBit_ Option | * | Vendor-Defined Custom Bit Option – This field is reserved for developers to implement a second custom bit-by-bit feature. See Custom Registers. |
| TM | R/W | Trigger Mode – Configures the trigger field to mask or match. See Event Updates |
| EU | R/W | Enable Unsolicited Updates – Enables automatic unsolicited responses to be issued in response to input changes. See Event Updates. |
| CA | R/W | Concatenate All Read_Value(s) – Configures Current GPI block to concatenate GPI Read_Value fields from all GPI blocks to all responses from current GPI block. See Concatenated Metadata. |
| CN | R/W | Concatenate Next Read_Value – Configures current GPI block to concatenate Read_Value field from next GPI block to all responses. See Concatenated Metadata. |
| CV | R/W | Concatenate VIBit_Option field – Configures current GPI block to concatenate VIBit_Option field to all Read_Value responses. See Concatenated Metadata. |
| CO | R/W | Concatenate Overflow – Configures current GPI block to concatenate the overflow registers to all Read_Value responses. See Concatenated Metadata. |
| CT | R/W | Concatenate Timestamp – Configures current GPI block to concatenate the timestamp registers to all Read_Value responses. See Concatenated Metadata. |
| VIBank_ Options | * | Vendor-Defined Bank Options -- This field is reserved for developer to implement custom bank features or report optional bank information. See Custom Registers. |
| U_Timer | R/W | Update Timer – Specifies the delay for periodic responses. See Table 5 -- Timer Delays for actual delays. See Periodic Updates. |
| DM | R/W | Debounce Mode – When set to 1, changes to GPI bits will not be registered until D_Timer * DCnt expires (glitch filter). When set to 0, changes to GPI bits will be registered immediately, but further changes will not be accepted until D_Timer * DCnt expires (bouncing lockout). See Debounce. |
| DCnt | R/W | Debounce Timer Multiplier – Offers finer resolution for the Debounce timer by repeating the timer a specified number of times. See Table 6 -- Timer Multipliers for count values. See Debounce. |

| CT | |
|---|---|
| $00_2$ | Do not concatenate time stamp |
| $01_2$ | Concatenate Register 4 (MSBs) |
| $10_2$ | Concatenate Register 6 (LSBs) |
| $11_2$ | Concatenate Registers 4 and 6 |

| | | |
|---|---|---|
| D_Timer | R/W | Debounce Timer – Specifies the base delay for debounce actions. See Table 5 -- Timer Delays for actual delays. See Debounce. |

### 9.4 GPI Transaction Listener Register Set

| Block | Offset | Bit | | | | | | | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | | 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 10 | 11 | 12 | 13 | 14 | 15 |
| o+i+2 | 0 | Current_Value [0:7] | | | | | | | | Current_Value [8:15] | | | | | | | |
| o+i+2 | 2 | Reset_Value [0:7] | | | | | | | | Reset_Value [8:15] | | | | | | | |
| o+i+2 | 4 | VLBit_Option [0:7] | | | | | | | | VLBit_Option [0:7] | | | | | | | |
| o+i+2 | 6 | reserved | | | | | | | | reserved | | | | | | | |
| o+i+2 | 8 | CC | RR | Bus_ID/Cont_ID | | | | | | Label | | | | | | | |
| o+i+2 | A | reserved | | | | | | | | Block | | | | | | | |
| o+i+2 | C | Update_Mask [0:7] | | | | | | | | Update_Mask [8:15] | | | | | | | |
| o+i | E | VLBank_Options [0:3] | | | | reserved | | | EU | WM | WCnt[0:1] | | W_Timer [0:4] | | | | |

**Figure 11 -- GPL Registers**

**Table 4 --GPL Register Definitions**

| Field | Bus Access | Definition |
|---|---|---|
| Current_Value | R/W | Current Value – Reports or sets the current value of the GPL outputs. See Controller Writes. |
| Reset_Value | R/W | Value Applied on Watchdog Reset – Identical to GPO Reset_Value functionality. See Watchdog Timer |
| VLBit_Option | * | Vendor-Defined Listener Bit Option – This field is reserved for developers to implement a custom bit by bit function. See custom registers feature. |
| EU | R/W | Enable Unsolicited Responses – Identical to GPI EU function. See Event Updates. |
| CC | R/W | Listen to Current Controller – Indicates which Bus_ID/Cont_ID to use to filter packets. See Listener Target. |
| RR | R/W | Listen to Read Requests – Indicate if Read Responses should be listened or filtered. See Listener Target. |
| Bus_ID/ Cont_ID | R/W | Bus_ID/Cont_ID Filter – Set to define the required Bus_ID/Cont_ID of listened packets. See Listener Target. Unused when CC=1. |
| Label | R/W | Label Filter – Set to define the required Dest_VP_Label of listened packets. See Listener Target. |
| Block | R/W | Bank Filter – Set to define the required Block of listened packets. See Listener Target. |
| Update_Mask | R/W | Mask Applied to Bits in Monitored Response Packets – Masks bits from the monitored response packets so they do not affect the Current_Value bits. See Update Mask |
| VLBank_ Options | * | Vendor Defined Bank Options -- This field is reserved for developers to implement custom bank features or report optional bank information. See Custom Registers. |
| WM | R/W | Watchdog Mode – Identical to GPO WM functionality. See Watchdog Timer |
| WCnt | R/W | Watchdog Timer Multiplier – Identical to GPO WCnt functionality. See Watchdog Timer |

| | | |
|---|---|---|
| W_Timer | R/W | Watchdog Timer –Identical to GPO W_Timer functionality. See Watchdog Timer |

**Table 5 -- Timer Delays**

| Timer Setting | Delay (seconds) | Timer Setting | Delay (seconds) | Timer Setting | Delay (seconds) | Timer Setting | Delay (seconds) |
|---|---|---|---|---|---|---|---|
| $1\ 1111_2$ | 40.7E-9 | $1\ 0111_2$ | 10.4E-6 | $0\ 1111_2$ | 2E-3 | $0\ 0111_2$ | 512E-3 |
| $1\ 1110_2$ | 81.4E-9 | $1\ 0110_2$ | 20.8E-6 | $0\ 1110_2$ | 4E-3 | $0\ 0110_2$ | 1 |
| $1\ 1101_2$ | 162.8E-9 | $1\ 0101_2$ | 41.7E-6 | $0\ 1101_2$ | 8E-3 | $0\ 0101_2$ | 2 |
| $1\ 1100_2$ | 325.5E-9 | $1\ 0100_2$ | 83.3E-6 | $0\ 1100_2$ | 16E-3 | $0\ 0100_2$ | 4 |
| $1\ 1011_2$ | 651.0E-9 | $1\ 0011_2$ | 125E-6 | $0\ 1011_2$ | 32E-3 | $0\ 0011_2$ | 8 |
| $1\ 1010_2$ | 1.3E-6 | $1\ 0010_2$ | 250E-6 | $0\ 1010_2$ | 64E-3 | $0\ 0010_2$ | 16 |
| $1\ 1001_2$ | 2.6E-6 | $1\ 0001_2$ | 500E-6 | $0\ 1001_2$ | 128E-3 | $0\ 0001_2$ | 32 |
| $1\ 1000_2$ | 5.2E-6 | $1\ 0000_2$ | 1E-3 | $0\ 1000_2$ | 256E-3 | $0\ 0000_2$ | Disabled |

**Table 6 -- Timer Multipliers**

| Multiplier Setting | Multiplier |
|---|---|
| $00_2$ | 1 |
| $01_2$ | 3 |
| $10_2$ | 5 |
| $11_2$ | 7 |

# 10 Packets

## 10.1 Physical_ID Based Packets

Physical_ID-based packets used with the GPIO profile follow exactly the Physical_ID-based packets defined in VersaPHY Extensions to IEEE-1394.

Physical_ID packets are primarily used for device discovery and VP-Label management. Many GPIO devices will be used in pre-configured networks that do not require discovery. They might also use static labels and not require VP-Label management capabilities. Devices not requiring Physical_ID discovery capability and VP-Label management capability are not required to respond to VersaPHY Physical_ID addressed requests.

These devices shall report their Physical_ID as 11 $1111_2$ in VP-Label based requests to block 0 byte 1.

## 10.2 VP-Label Based Packets

The address, data, and CRC blocks for GPIO VP-Label based packets follow the Physical_ID-based packet format.

Note that CRC generation and validation are not required for all GPIO devices. If CRC generation is used, it shall follow the specified method.

## 10.3 GPIO Profile Read Request VersaPHY Packet with VP-Label Addressing

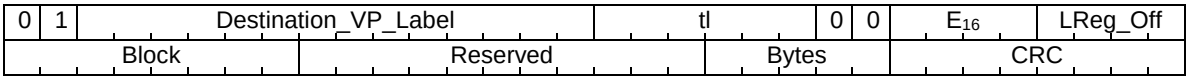


**Figure 12 -- GPIO Profile Read Request Using VP-Label**

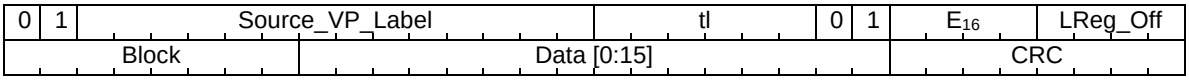


**Figure 13 -- GPIO Profile Read Response Using VP-Label**

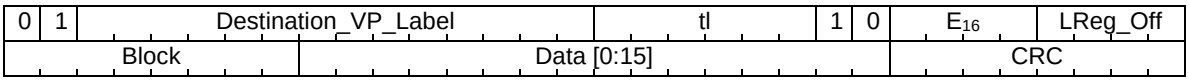


**Figure 14 -- GPIO Profile Write Request Using VP-Label**

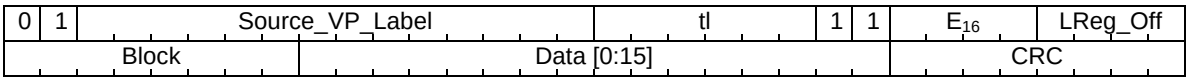


**Figure 15 -- GPIO Profile Write Response Using VP-Label**

**Table 7-- GPIO Packet Field Definitions**

| Field | Description |
|---|---|
| Dest_VP_Label | VersaPHY Label of the packet's destination. |
| Source_VP_Label | VersaPHY Label of the device sending the packet. |
| tl | Transaction label used to associate a request with a response. If required by the application, the requester must ensure no outstanding tl's have the same value. The tl value of $111111_2$ is reserved for use by unsolicited responses. |
| LReg_Off | Register Offset Low is the lowest four bits of the (first) VersaPHY register being accessed. |
| Block | Register block address are the most significant address bits of the register being accessed. |
| Data | Current register data value (response) or to be written (write request) |
| Bytes | Length of read request in bytes. Not all GPIO devices are required to respond to multi-register reads. Devices that only support single-register reads shall respond with the first register requested. |
| CRC | The CRC covers the first 56-bits of the packet. It is an 8 bit CRC: poly = x^8 + x^2 + x^1 + x^0, initial value = $FF_{16}$, final result inverted. See *VersaPHY Extenstions to IEEE-1394* for example code.<br><br>CRC processing is not required for all GPIO devices. Some applications may not require CRC protection. These implementations shall always report $5A_{16}$ in the CRC field. Controllers are responsible for recognizing this repeating value and inferring that valid CRCs are not supported. |

# Annex A
(normative)

# Complete List of Calculated Timer Delays

**Table A-1 Timer Delays**

| Timer | Cnt | Seconds | Timer | Cnt | Seconds | Timer | Cnt | Seconds | Timer | Cnt | Seconds |
|---|---|---|---|---|---|---|---|---|---|---|---|
| 1 1111 | 00 | 40.7E-9 | 1 0110 | 00 | 20.8E-6 | 1 0001 | 11 | 3.5E-3 | 0 1000 | 01 | 0.768 |
| 1 1110 | 00 | 81.4E-9 | 1 1000 | 10 | 26.0E-6 | 0 1110 | 00 | 4.0E-3 | 0 1001 | 11 | 0.896 |
| 1 1111 | 01 | 122.1E-9 | 1 0111 | 01 | 31.3E-6 | 1 0000 | 10 | 5.0E-3 | 0 0110 | 00 | 1 |
| 1 1101 | 00 | 162.8E-9 | 1 1000 | 11 | 36.5E-6 | 0 1111 | 01 | 6.0E-3 | 0 1000 | 10 | 1.28 |
| 1 1111 | 10 | 203.5E-9 | 1 0101 | 00 | 41.7E-6 | 1 0000 | 11 | 7.0E-3 | 0 0111 | 01 | 1.536 |
| 1 1110 | 01 | 244.1E-9 | 1 0111 | 10 | 52.1E-6 | 0 1101 | 00 | 8.0E-3 | 0 1000 | 11 | 1.792 |
| 1 1111 | 11 | 284.8E-9 | 1 0110 | 01 | 62.5E-6 | 0 1111 | 10 | 10.0E-3 | 0 0101 | 00 | 2 |
| 1 1100 | 00 | 325.5E-9 | 1 0111 | 11 | 72.9E-6 | 0 1110 | 01 | 12.0E-3 | 0 0111 | 10 | 2.56 |
| 1 1110 | 10 | 406.9E-9 | 1 0100 | 00 | 83.3E-6 | 0 1111 | 11 | 14.0E-3 | 0 0110 | 01 | 3 |
| 1 1101 | 01 | 488.3E-9 | 1 0110 | 10 | 104.2E-6 | 0 1100 | 00 | 16.0E-3 | 0 0111 | 11 | 3.584 |
| 1 1110 | 11 | 569.7E-9 | 1 0011 | 00 | 125.0E-6 | 0 1110 | 10 | 20.0E-3 | 0 0100 | 00 | 4 |
| 1 1011 | 00 | 651.0E-9 | 1 0101 | 01 | 125.0E-6 | 0 1101 | 01 | 24.0E-3 | 0 0110 | 10 | 5 |
| 1 1101 | 10 | 813.8E-9 | 1 0110 | 11 | 145.8E-6 | 0 1110 | 11 | 28.0E-3 | 0 0101 | 01 | 6 |
| 1 1100 | 01 | 976.6E-9 | 1 0101 | 10 | 208.3E-6 | 0 1011 | 00 | 32.0E-3 | 0 0110 | 11 | 7 |
| 1 1101 | 11 | 1.1E-6 | 1 0010 | 00 | 250.0E-6 | 0 1101 | 10 | 40.0E-3 | 0 0011 | 00 | 8 |
| 1 1010 | 00 | 1.3E-6 | 1 0100 | 01 | 250.0E-6 | 0 1100 | 01 | 48.0E-3 | 0 0101 | 10 | 10 |
| 1 1100 | 10 | 1.6E-6 | 1 0101 | 11 | 291.7E-6 | 0 1101 | 11 | 56.0E-3 | 0 0100 | 01 | 12 |
| 1 1011 | 01 | 2.0E-6 | 1 0011 | 01 | 375.0E-6 | 0 1010 | 00 | 64.0E-3 | 0 0101 | 11 | 14 |
| 1 1100 | 11 | 2.3E-6 | 1 0100 | 10 | 416.7E-6 | 0 1100 | 10 | 80.0E-3 | 0 0010 | 00 | 16 |
| 1 1001 | 00 | 2.6E-6 | 1 0001 | 00 | 500.0E-6 | 0 1011 | 01 | 96.0E-3 | 0 0100 | 10 | 20 |
| 1 1011 | 10 | 3.3E-6 | 1 0100 | 11 | 583.3E-6 | 0 1100 | 11 | 112.0E-3 | 0 0011 | 01 | 24 |
| 1 1010 | 01 | 3.9E-6 | 1 0011 | 10 | 625.0E-6 | 0 1001 | 00 | 128.0E-3 | 0 0100 | 11 | 28 |
| 1 1011 | 11 | 4.6E-6 | 1 0010 | 01 | 750.0E-6 | 0 1011 | 10 | 160.0E-3 | 0 0001 | 00 | 32 |
| 1 1000 | 00 | 5.2E-6 | 1 0011 | 11 | 875.0E-6 | 0 1010 | 01 | 192.0E-3 | 0 0011 | 10 | 40 |
| 1 1010 | 10 | 6.5E-6 | 1 0000 | 00 | 1.0E-3 | 0 1011 | 11 | 224.0E-3 | 0 0010 | 01 | 48 |
| 1 1001 | 01 | 7.8E-6 | 1 0010 | 10 | 1.3E-3 | 0 1000 | 00 | 256.0E-3 | 0 0011 | 11 | 56 |
| 1 1010 | 11 | 9.1E-6 | 1 0001 | 01 | 1.5E-3 | 0 1010 | 10 | 320.0E-3 | 0 0010 | 10 | 80 |
| 1 0111 | 00 | 10.4E-6 | 1 0010 | 11 | 1.8E-3 | 0 1001 | 01 | 384.0E-3 | 0 0001 | 01 | 96 |
| 1 1001 | 10 | 13.0E-6 | 0 1111 | 00 | 2.0E-3 | 0 1010 | 11 | 448.0E-3 | 0 0010 | 11 | 112 |
| 1 1000 | 01 | 15.6E-6 | 1 0001 | 10 | 2.5E-3 | 0 0111 | 00 | 512.0E-3 | 0 0001 | 10 | 160 |
| 1 1001 | 11 | 18.2E-6 | 1 0000 | 01 | 3.0E-3 | 0 1001 | 10 | 640.0E-3 | 0 0001 | 11 | 224 |

# Annex B
(informative)

# Minimum GPO Example Code

```
library IEEE;
library UNISIM;
use IEEE.STD_LOGIC_1164.ALL;
use IEEE.STD_LOGIC_ARITH.ALL;
use IEEE.STD_LOGIC_UNSIGNED.ALL;

entity minGPIO is Port (
    Cntl    : inout std_logic_vector(0 to 1);
    Data    : inout std_logic_vector(0 to 7);
    PClk    : in    std_logic;
    LClk    : out   std_logic;
    LReq    : out   std_logic;
    GPO1    : out   std_logic);
end minGPIO;

architecture Behavioral of minGPIO is
  Constant VPLabel  : std_logic_vector(0 to 13) := "111111" & x"05";  --Static VersaPHY Label
  Constant LReqBits : std_logic_vector(0 to 3)  := '1' & "010";--"0" & '0' & "0000" --LReq Stream
                                                --start|current fair |rfmt| speed --always S100
  type state_type is (Idle_Speed,Drive,Byte0,Byte1,Byte2,Byte3,Byte4,Byte5,Byte6,Byte7,Done);

  signal state, nState          : state_type := Idle_Speed;          --State machine for transmit and receive
  signal CntlOut,nCntlOut       : std_logic := '0';                  --Control output bit 1 (bit 0 = 0)
  signal inData                 : std_logic_vector(0 to 7);          --Input data buffer
  signal DataOut,nDataOut       : std_logic_vector(0 to 7);          --Output data buffer
  signal inCntl                 : std_logic_vector(0 to 1);          --Input control buffer
  signal GPOi,nGPO              : std_logic;                         --Internal GPO register
  signal DrivePL,nDrivePL       : std_logic := '0';                  --PHY/Link ouput enable
  signal ByteClk                : std_logic := '0';                  --Enable for multi clock PHY/Link bytes
  signal ClkCount               : std_logic_vector(0 to 2) := "111"; --Counter for multi clock PHY/Link bytes
  signal ValidPkt,nValidPkt     : std_logic;                         --Valid Header status
  signal LReqState              : std_logic_vector(0 to 1) := "11";  --Counter for LReq stream
  signal iLReq                  : std_logic;                         --Internal LReq register
  signal ReqW_R,nReqW_R         : std_logic;                         --Write_Read bit captured from request
  signal ResW_R,nResW_R         : std_logic;                         --Write_Read bit for next response
  signal ReqAddress,nReqAddress : std_logic_vector(0 to 4);          --Address bits captured from request
  signal ResAddress,nResAddress : std_logic_vector(0 to 4);          --Address bits for next response

begin
  Cntl(0) <= 'Z'        when DrivePL='0' else '0';                   --Assign Internal signals to pins
  Cntl(1) <= 'Z'        when DrivePL='0' else CntlOut;
  Data    <= "ZZZZZZZZ" when DrivePL='0' else DataOut;
  LClk    <= PClk;
  LReq    <= iLReq;
  GPO1    <= GPOi;

  Clk1_PROC: process(PClk)                --Counter for 8 clock S100 bytes
  begin                                   --Enabled when speed detected or non-idle states
    if PClk'event and PClk='1' then       --Disabled on last clock of byte 7
      if (state=Byte7 and ByteClk='1') or (state=idle_speed and not(inCntl="10" and inData(0)='0')) then
        ClkCount  <= "111";
      else
        ClkCount  <= ClkCount + "001";
      end if;
    end if;
  end process;

  ByteClk <= '1' when ClkCount="111" else '0'; --Register enable for 8 clock S100 bytes

  Latch_Proc: process (PClk)
  begin
    if (PClk'event and PClk='1') then
      if ByteClk='1' then             --Latch all "8 clock" enabled registers
        state     <= nState;
        DataOut   <= nDataOut;
        CntlOut   <= nCntlOut;
        DrivePL   <= nDrivePL;
```

```
      ValidPkt  <= nValidPkt;
      ReqW_R    <= nReqW_R;
      ResW_R    <= nResW_R;
      ReqAddress<= nReqAddress;
      ResAddress<= nResAddress;
      GPOi      <= nGPO;
    end if;
    inData      <= Data;
    inCntl      <= Cntl;
  end if;
end process;

Request_Decode: process (state,inData,ValidPkt,ReqW_R,ReqAddress,ResW_R,ResAddress,GPOi)
begin                                  --Decode all request driven registers
  nValidPkt   <= ValidPkt;                        --Maintain previous values as default
  nReqW_R     <= ReqW_R;
  nReqAddress <= ReqAddress;
  nResW_R     <= ResW_R;
  nResAddress <= ResAddress;
  nGPO        <= GPOi;

  case (state) is
    when Idle_Speed =>
        if (inData(4 to 5)  /=  "00") then        --S100 Packets only
          nValidPkt <= '0';
        else
          nValidPkt <= '1';
        end if;
    when Byte0    =>
      if (inData/=("01" & VPLabel(0 to 5))) then  --Valid header and high VPLabel
        nValidPkt <= '0';
      end if;
    when Byte1    =>
      if (inData  /=  VPLabel(6 to 13)) then      --Valid low VPLabel
        nValidPkt <= '0';
      end if;
    when Byte2    =>
      if (inData(7) /=  '0')  then                --Requests only
        nValidPkt <= '0';
      end if;                                     --not collecting tlabel
      nReqW_R     <= inData(6);                   --Collect Write_Read bit.
    when Byte3    =>
      if (inData(0 to 3)  /=  x"E") then          --verify tCode E
        nValidPkt <= '0';
      end if;
      nReqAddress(2 to 4) <= inData(4 to 6);      --Collect Low Address bits
    when Byte4  =>
      nReqAddress(0 to 1) <= inData(6 to 7);      --Collect High Address bits
    when Byte5  =>
      if  ReqW_R='1' and                          --Write Request and
          ValidPkt='1' and                        --Valid Header bits and
          ReqAddress="10001" then                 --GPO register
            nGPO  <= inData(0);                   --  Collect GPO data
      end if;
    when Byte6    =>
        if (ValidPkt='1') then                    --If header bits were valid,
          nResW_R     <= ReqW_R;                  --  tranfer W_R and Address to Response registers
          nResAddress <= ReqAddress;
        end if;
    when Others   => null;
  end case;
end process;

Response_Decode: process (state,ResAddress,ResW_R,GPOi,DrivePL,CntlOut,inCntl)
begin                               --decode all response driven registers
  nDataOut  <= x"00";
  nDrivePL  <= DrivePL;
  nCntlOut  <= CntlOut;
  case (state) is --output bytes must be calculated the state before they are transmitted.
    when Idle_Speed =>    if inCntl="11" then
                            nDrivePL  <= '1';
                          end if;
    when Drive  =>        nCntlOut  <= '1';
                          nDataOut  <= "01" & VPLabel(0 to 5);
    when Byte0  =>        nDataOut  <= VPLabel(6 to 13);
    when Byte1  =>        nDataOut <= "000000" & ResW_R & '1';          --tlabel, W_R, '1'
```

```
      when Byte2  =>        nDataOut <= "1110" & ResAddress(2 to 4) & '0';  --"1110" & "0" & Add
      when Byte3  =>        nDataOut  <= "000000" & ResAddress(0 to 1) ;
      when Byte4  =>
        case (ResAddress) is
          when "00000"  =>  nDataOut <= "10000000";
          when "00010"  =>  nDataOut <= "11000010";
          when "00011"  =>  nDataOut <= x"80";
          when "10001"  =>  nDataOut  <= GPOi & "0000000";
          when others   =>  null;
        end case;
      when Byte5 =>
        case (ResAddress) is
          when "00000"  =>  nDataOut <= "00111111";
          when "00001"  =>  nDataOut <= "00000111";
          when others   =>  null;
        end case;
      when Byte6    =>      nDataOut  <= x"5A";                         --should be crc
      when Byte7    =>      nCntlOut  <= '0';
      when Done     =>      nDrivePL  <= '0';
      when others   =>      null;
    end case;
  end process;

  state_decode: process (inData(0),state,inCntl)
  begin
      case (state) is
        when Idle_Speed =>  if inData(0)='0' and inCntl="10" then --wait for speed indication
                              nState <= Byte0;
                            elsif inCntl="11" then
                              nState <= Drive;
                            else
                              nState <= Idle_Speed;
                            end if;
        when Drive  =>    nState <= Byte0;
        when Byte0  =>    nState <= Byte1;
        when Byte1  =>    nState <= Byte2;
        when Byte2  =>    nState <= Byte3;
        when Byte3  =>    nState <= Byte4;
        when Byte4  =>    nState <= Byte5;
        when Byte5  =>    nState <= Byte6;
        when Byte6  =>    nState <= Byte7;
        when Byte7  =>    nState <= Done;
        when Done   =>  if inCntl="00" then
                          nState <= Idle_Speed;
                        else
                          nState <= Done;
                        end if;
      end case;
  end process;
-------------------------------------------------------LReq Handler-----------------------------------------
  LReq_Sync: process(PClk)
  begin
    if (PClk'event and PClk='1') then
      if state=Byte6 and ValidPkt='1' and ByteClk='1' then
        LReqState <= "00";
      elsif LReqState/="11" then
        LReqState <= LReqState + '1';
      end if;
    end if;
  end process;

  LReq_PROC: process(PClk)
  begin
    if PClk'event and PClk='1' then
      iLReq     <= LReqBits(CONV_INTEGER(LReqState));
    end if;
  end process;
end Behavioral;
```

# Annex C
# (normative)
# Conformance requirements

This annex is intended to assist VersaPHY GPIO device designers and implementers as well as developers of conformance tests; it provides a concise summary of mandatory and optional features and, for each feature, reference to the governing normative clauses.

If optional features are implemented the associated sub-features may or may not Mandatory.

| **Feature** | **Sub-feature** | **Implementation** | **Reference** |
|---|---|---|---|
| **Grouping** | | | 5.1 |
| | VP-Function level | Mandatory | 5.1 |
| | Bank level | Mandatory | 5.1 |
| **Custom Registers** | | Optional | 5.2 |
| General Purpose Output Features | | | 6 |
| | Output Enable | Mandatory | 6.1 |
| | Watchdog Timer | Optional | 6.2 |
| | Set/Clear | Optional | 6.3 |
| | Bus Label Space | Optional | 6.5 |
| General Purpose Input Features | | | 7 |
| | At least one of the following three are mandatory | | 7.1 |
| | Polling | Optional | 7.1.2 |
| | Event Updates | Optional | 7.1.3 |
| | Periodic Updates | Optional | 7.1.4 |
| | Timestamp | Optional | 7.2 |
| | Timestamp Overflow | Optional | 7.3 |
| | Debounce | Optional | 7.4 |
| | Input Overflow | Optional | 7.5 |

| | | | |
|---|---|---|---|
| | Heartbeat | Optional | 7.6 |
| | Concatenated Meta data | Optional | 7.7 |
| **General Purpose Listener Features** | | | 8 |
| | Listener Target | Optional | 8.1 |
| | Controller Writes | Optional | 8.2 |
| | Watchdog | Optional | 8.3 |
| | Update Mask | Optional | 8.4 |
| | Automatic Updates | Optional | 8.5 |
| **Register Definitions** | | | 9 |
| | GPIO Base Register Set | Mandatory | 9.1 |
| | At least one of the following three are mandatory | | |
| | General Purpose Output Register Set | Optional | 9.2 |
| | General Purpose Input Register Set | Optional | 9.3 |
| | General Purpose Listener Register Set | Optional | 9.4 |
| **Packets** | | | 10 |
| | Physical_ID Based Packets | Optional | 10.1 |
| | VP-Label Based Packets | Mandatory | 10.2 |
| | GPIO Profile Read Request | Mandatory | 10.3 |
| | | | |

# Annex D
(informative)

# Bibliography

[B1] IEEE Std 1394-1995, Standard for a High Performance Serial Bus

[B2] IEEE Std 1394a-2000, Standard for a High Performance Serial Bus—Amendment 1

[B3] IEEE Std 1394b-2002, Standard for a High Performance Serial Bus—Amendment 2