|  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| Document No: | |  | | | 2 | | | Date: | | 10/04/2017 |
|  | | | | | | | | | | |
| Document Title: | | | | | | | | | | |
| USB FlexComms Interface FPGA Datasheet | | | | | | | | | | |
|  | | | | | | | | | | |
| Summary / Scope: | | |  | | | | | | | |
| Describes the register and DPR addresses in the FCI FPGA [1]. | | | | | | | | | | |
|  | | | | | | | | | | |
| Reason for Issue / Nature of Change: | | | | | | |  | | | |
| Updated for firmware version 1.3 | | | | | | | | | | |
|  | | | | | | | | | | |
| Distribution: | | | |  | | | | | | |
| Project file | | | | | | | | | | |
|  | | | | | | | | | | |
| Author |  | | | | | Reviewer | | |  | |
| Reviewer |  | | | | |  | | |  | |

Contents

1 Introduction 4

2 Memory map 5

3 USB Interface DSP streaming worked example 29

Revision sheet

|  |  |  |  |
| --- | --- | --- | --- |
| **Issue** | **Change History** | **Author** | **Date** |
| 1 | First Issue | Nick Schollar | 15/09/2016 |
| 2 | Updated write only registers to read/write (section 2)  Corrected FPGA version number representation (2.1)  Updated USB configuration registers (2.32-2.37) to correspond to latest USB Interface specification  Added registers 0x0010 and 0x0014  Corresponds to code version v1.3 | Richard Jones | 10/04/2017 |

References

1. VHDL on Git (<http://git-embedded:7990>) –3065-FCI/USB\_FlexComms\_Interface.
2. PD-3065-9071-07 Head Interface Card FPGA Datasheet ([\\renishaw\global\GB\PLC\CMMPD\Data\CMMProjects\3065 - Head Interface Card\Controlled Documents - BLUE\Project Docs PD-\PD-3065-9071-07 Head Interface Card FPGA datasheet.docx](file:///\\renishaw\global\GB\PLC\CMMPD\Data\CMMProjects\3065%20-%20Head%20Interface%20Card\Controlled%20Documents%20-%20BLUE\Project%20Docs%20PD-\PD-3065-9071-07%20Head%20Interface%20Card%20FPGA%20datasheet.docx))
3. PD-3065-9111 USB Module Specification ([\\renishaw\global\GB\PLC\CMMPD\Data\CMMProjects\3065 - Head Interface Card\Controlled Documents - BLUE\Project Docs PD-\FlexComms Interface\Source\PD-3065-9111-02 USB Module Specification.docx](file:///\\renishaw\global\GB\PLC\CMMPD\Data\CMMProjects\3065%20-%20Head%20Interface%20Card\Controlled%20Documents%20-%20BLUE\Project%20Docs%20PD-\FlexComms%20Interface\Source\PD-3065-9111-02%20USB%20Module%20Specification.docx))

1. [\\renishaw\global\GB\PLC\CMMPD\Data\CMMnwprodcon\Project Files\5208 (SPA2)\Project Team Members File - (RED)\DSP Design\Documents\SPA2 SDD 5.pdf]( \\\\renishaw\\global\\GB\\PLC\\CMMPD\\Data\\CMMnwprodcon\\Project Files\\5208 (SPA2)\\Project Team Members File - (RED)\\DSP Design\\Documents\\SPA2 SDD 5.pdf)
2. PD-3007-9131-07 Serial Communications Scheme ([\\renishaw\global\GB\PLC\CMMPD\Data\CMM\Projects\3007 - GYRO-PH20\Controlled Documents - BLUE\Project Docs PD-\FlexComms Interface\PD-3007-9131-07 Serial communications scheme.docx](file:///\\renishaw\global\GB\PLC\CMMPD\Data\CMM\Projects\3007%20-%20GYRO-PH20\Controlled%20Documents%20-%20BLUE\Project%20Docs%20PD-\FlexComms%20Interface\PD-3007-9131-07%20Serial%20communications%20scheme.docx))
3. PD-3065-9108-02 Flex Comms Interface Critical Design Intent ([\\renishaw\global\GB\PLC\CMMPD\Data\CMMProjects\3065 - Head Interface Card\Controlled Documents - BLUE\Project Docs PD-\FlexComms Interface\Source\PD-3065-9108-03 FlexComms Interface Card CDI.docx](\\\\renishaw\\global\\GB\\PLC\\CMMPD\\Data\\CMMProjects\\3065 - Head Interface Card\\Controlled Documents - BLUE\\Project Docs PD-\\FlexComms Interface\\Source\\PD-3065-9108-03 FlexComms Interface Card CDI.docx))
4. FlexComms Interface PCB schematic C-3065-0880-03 (Teamcenter)

# Introduction

The USB FlexComms Interface (“FCI”, A-3065-0880-03) provides an interface between a computer and any Renishaw FlexComms product. The FCI includes a Texas Instruments DSP, two integrated 3-phase motor drives, an Altera Cyclone IV FPGA and associated power supplies. The FPGA implements the USB interface [3], comms to the DSP [4] and comms to the a FlexComms node [5].

This document describes the registers and dual-port RAM in the FPGA. Refer to the CDI [] and schematic [] for detailed low-level information.

## Features

* USB interface to computer
* FlexComms Interface to CMM head or other compatible device
* Motor drivers for two head motors
* 20V supply for powering head electronics
* FPGA with logic capacity of 22,000 LUTs

## How to use the FlexComms Interface

1. Connect a FlexComms head or other FlexComms device to the 15-pin D-sub connector using an appropriate cable
2. If using a head, connect a 24V (PH20) or 48V (REVO-2) power supply to the power connector
3. Connect the FlexComms Interface to a computer using a USB cable
4. Launch software

|  |  |  |
| --- | --- | --- |
|  | USB supply | External supply |
| REVO-2 | 🗶 | ✓ |
| PH20 | 🗶 | ✓ |
| FlexComms Adapter Without Attachment | 🗶 | ✓ |
| FlexComms Adapter With Attachment | 🗶 | ✓ |

## Limitations

External 24V/48V power supply is required to power a FlexComms device; the FlexComms Interface cannot operate from USB power alone when powering external nodes.

# Memory map

Addresses are offsets from the FlexComms Interface PCB base address, when the card is accessed over the USB interface. A USB bus write to an address where there is no defined register has no effect on the FCI.

| address | data | r/w |
| --- | --- | --- |
| [0x00](#_0x00:_FPGA_version) | FPGA version number major | Ro |
| [0x04](#_0x00,_0x04:_FPGA) | FPGA version number minor | Ro |
| … | … | … |
| [0x0C](#_0x0C:_Reset_comms) | reset comms counters | Wo |
| [0x10](#_0x10:_Tx_/) | Tx / Rx frame count discrepancy | Ro |
| [0x14](#_0x14_Tx_frame) | Tx count multiplier | RW |
| … | … | … |
| [0x5C](#_0x5C,_0x64,_0x6C,) | RX time | Ro |
| [0x64](#_0x5C,_0x64,_0x6C,) | TX time | Ro |
| [0x6C](#_0x5C,_0x64,_0x6C,) | current time | Ro |
| [0x74](#_0x5C,_0x64,_0x6C,) | sync time | Ro |
| [0x7C](#_0x7C:_DSP_checksum) | DSP comms checksum | Ro |
| [0x80](#_0x80-0x88:_DSP_TAPs) | DSP TAP1\_1 | RW |
| [0x84](#_0x80-0x88:_DSP_TAPs) | DSP TAP1\_2 | RW |
| [0x88](#_0x80-0x88:_DSP_TAPs) | DSP TAP1\_3 | RW |
| [0x8C](#_0x80-0x88:_DSP_TAPs) | DSP TAP2\_1 | RW |
| [0x90](#_0x80-0x88:_DSP_TAPs) | DSP TAP2\_2 | RW |
| [0x94](#_0x80-0x88:_DSP_TAPs) | DSP TAP2\_3 | RW |
| … | … | … |
| [0xFC](#_0xFC:_DSP_comms) | DSP comms control/status | RW |
| … | … | … |
| [0x100](#_0x100:_destination_address) | destination address, source address | RW |
| [0x104](#_0x104:_frame_type,) | frame type, control status | RW |
| [0x108](#_0x108:_service_channel) | service channel address, data | RW |
| [0x10C](#_0x10C:_send_sync) | send sync frame | Wo |
| [0x110](#_0x110_bits_2-0:) | scheduler control | RW |
| [0x114](#_0x114:_scheduler_period) | scheduler period | RW |
| [0x118](#_0x118_bit_0:) | have received frame (bit 0 ) | Ro |
| [0x11C](#_0x11C_heads_comms) | head comms control | RW |
| [0x120](#_0x120_head_power) | head power control | RW |
| [0x124](#_0x124_DSP_service) | DSP service channel TX FIFO status | Ro |
| … | … | … |
| [0x200](#_0x200_–_0x23C:) | node source address array | RW |
| … | … |  |
| [0x23C](#_0x200_–_0x23C:) | node source address array | RW |
| [0x240](#_0x240:_Frames_received) | frames received register | RW |
| … | … | … |
| … | … | … |
| [0x300](#_0x300_–_0x37C:) | 1st received head comms service channel response | Ro |
| … | … | … |
| [0x37C](#_0x300_–_0x37C:) | 32nd received head comms service channel response | Ro |
| [0x380](#_0x0380_head_comms) | received frame CRC error array | Ro |
| [0x384](#_0x0384_transmit_frame) | transmit frame FIFO full |  |
| 0x388 | reserved | - |
| [0x38C](#_0x038C_force_error) | force error count | RW |
| [0x390](#_0x0390_–_0x398) | frame error count |  |
| [0x394](#_0x0390_–_0x398) | max frame error count |  |
| [0x398](#_0x0390_–_0x398) | expected number of response frames |  |
| [0x39C](#_0x39C_automatic_command) | automatic command frame control | RW |
| [0x3A0](#_0x3A0_automatic_command) | automatic command frame transmission delay | RW |
| [0x3A4](#_0x3A0_automatic_command) | automatic command frame status | Ro |
| … | … | … |
| [0x1000](#_0x1000,_bits_7-0:) | DSP service channel RX/TX FIFO | RW |
| [0x1004](#_0x1004:_Service_channel) | number of bytes in DSP RX FIFO | Ro |
| [0x1008](#_0x1008:_LTC4151_Config) | LTC4151 Config (Only bits 7 down to 0 writable) | RW |
| [0x100C](#_0x100C:_LTC4151_Sense) | LTC4151 Sense | Ro |
| [0x1010](#_0x1010:_LTC4151_VIN) | LTC4151 VIN | Ro |
| [0x1014](#_0x1014_MAX1286_Temperature) | MAX1286 Temperature | Ro |
| … | … | … |
| [0x8000](#_Head_comms_dual-port)- | start of head comms dual-port RAM | RW |
| … | … | RW |
| [0xBFFC](#_Head_comms_dual-port) | end of head comms DPR | RW |
| … | … | … |
| [0xFF00](#_0xFF00_USB_Interface) | USB Interface Control Byte | RW |
| [0xFF04](#_0xFF04_USB_Interface) | USB Interface Status Byte | Ro |
| [0xFF08](#_0xFF08_USB_Interface) | USB Interface Quantity Of Addresses To Stream | RW |
| [0xFF0C](#_0xFF0C_USB_Interface) | USB Interface Stream Interval | RW |
| [0xFF10](#_0xFF10-0xFF8C_USB_Interface) | USB Interface First Address To Stream | RW |
| … | … | … |
| 0xFF4C | USB Interface 16th Address To Stream | RW |
| … | … | … |
| 0xFF8C | USB Interface 32nd Address To Stream | RW |
| [0xFF90](#_0xFF90_Stream_Config) | Stream Config 1 | RW |
| [0xFF94](#_0xFF94_Stream_Config) | Stream Config 2 | RW |

## 0x00, 0x04: FPGA version number

Addresses 0x0000 and 0x0004 contain the FPGA version number major and minor respectively.

|  |  |
| --- | --- |
| **bits 31-16** | **bits 15-0** |
| 0x0000 | version number |

## 0x0C: Reset comms counters

A write of any value to address 0x0C resets the comms rx, tx and error counters to zero – see 2.31.8 and 2.21

## 0x10: Tx / Rx frame count discrepancy

Returns the difference between the multiplied Tx and Rx frame count objects (HIC addresses 0x803C – 0x8048) according to the following formula:

*Frame count discrepancy = (Tx\_frame\_count x Tx\_frame\_count\_multiplier) – Rx frame count*

Note that this value is not latched or synchronised to a FlexComms frame.

## 0x14 Tx frame count multiplier

This 16-bit number is used to multiply the Tx frame count when calculating the count discrepancy.

## 0x5C, 0x64, 0x6C, 0x74: timers

A 32-bit free running counter is implemented which increments every 100ns (i.e. at a frequency of 10MHz). The current value of the counter may be read directly at address **0x6C**. The counter value is latched into 3 separate registers on each of the following events:

* Address **0x5C**: complete reception of any comms frame addressed to the FCI.
* Address **0x64**: commencement of transmission of a comms frame.
* Address **0x74**: a ‘1’ to ‘0’ transition on the ‘SYNC’ input (i.e. latching the CMM scales).

## 0x7C: DSP checksum

Bits 15-0 at address 0x7C contain the 16-bit checksum received from the DSP in the most recent Two Axis Packet (TAP). The checksum is validated automatically by the FPGA – a mismatch in received and calculated checksums is indicated in DSP comms status register bit 5 (see 2.6.2).

## 0x80-0x94: DSP TAPs

Addresses 0x80-0x94 contain the contents of the single TAP received from or written to the DSP. Serial communication between the DSP and FPGA uses the same protocol as between the UCC-2 and SPA-2, etc. For more information on the FPGA-DSP serial communication protocol, see ref [4].

### 0x80

|  |  |
| --- | --- |
| **bits 31-16** | **bits 15-0** |
| TAP 1 data word 1 | TAP 1 data word 2 |

### 0x84

|  |  |
| --- | --- |
| **bits 31-16** | **bits 15-0** |
| TAP 1 data word 3 | TAP 1 data word 4 |

### 0x88

|  |  |  |
| --- | --- | --- |
| **bits 31-16** | **bits 15-8** | **bits 7-0** |
| Not used | TAP 1 control / status | TAP 1 service channel |

### 0x8C

|  |  |
| --- | --- |
| **bits 31-16** | **bits 15-0** |
| TAP 2 data word 1 | TAP 2 data word 2 |

### 0x90

|  |  |
| --- | --- |
| **bits 31-16** | **bits 15-0** |
| TAP 2 data word 3 | TAP 2 data word 4 |

### 0x94

|  |  |  |
| --- | --- | --- |
| **bits 31-16** | **bits 15-8** | **bits 7-0** |
| Not used | TAP 2 control / status | TAP 2 service channel |

When the data for both TAPs has been written (i.e. writes have occurred to addresses 0x80 – 0x94 inclusive) the FPGA automatically sends the TAP to the DSP.

When the DSP has sent its response to the FPGA, the TAPs received by the FPGA are read from addresses 0x80 – 0x94. Bit 4 in the DSP comms status register (2.6.2) indicates that new TAP data is available.

## 0xFC: DSP comms control/status register

Address 0xFC contains the *DSP serial comms control* and *status* registers.

### 0xFC: DSP comms control (write)

Writing to address 0xFC writes the *DSP comms control register*.

|  |  |  |
| --- | --- | --- |
| **bit** | **description** | **default** |
| 31 | reset DSP comms  Write ‘1’ to synchronously reset the serial comms logic | 0 |
| 30-0 | unused |  |

### 0xFC: DSP comms status register (read)

Reading from address 0xFC reads the *DSP comms status register*.

|  |  |
| --- | --- |
| **bit** | **description** |
| 31-6 | unused |
| 5 | receiver error detected (CRC error in last TAP) |
| 4 | receiver full (i.e. message received since last read) |
| 3-1 | unused |
| 0 | transmitter not busy  ‘0’ indicates transmission in progress |

## 0x100-0x108: head comms frame

Reading addresses 0x100-0x108 reads data from the most recently received head comms frame.

Writing to addresses 0x100-0x108 writes data which is presented at the input of a 60-bit wide by 32 deep FIFO. The data is written to the FIFO when address 0x104 (frame type, control, status) is written.

When the *automatically send command frame* bit in the *automatic command frame control* register (see 2.22) is ‘0’, the transmit behaviour is as follows:

The frame transmitter sends frames sequentially from the FIFO – i.e. as soon as the frame transmitter detects a frame in the transmit FIFO it sends the frame. When the frame transmission has completed, the transmitter waits for *20 clock cycles* (the transmit frame gap, equal to 400ns) and then checks the FIFO for further frames. The FIFO therefore allows the host controller to write up to 32 frames in succession without having to wait between writes for previous frame transmission to complete. The host controller can check whether the FIFO is full by reading bit 0 at address 0x384 – see 2.19.

When the *automatically send command frame* bit is ‘1’, no frames are transmitted from the transmit FIFO until the *automatic command frame transmission delay* (see 2.23) has elapsed, after which time, if there is a frame in the transmit FIFO it is immediately transmitted. If the *automatically send command frame* bit is ‘1’ and no frames have been written to the transmit FIFO when the *automatic command frame transmission delay* has elapsed, the next frame written to the FIFO in this comms cycle – i.e. before the next sync frame is transmitted, is discarded. See 2.23. (This functionality is designed to support REVO Vision Probe data transfer.)

### 0x100: destination address and source address

|  |  |
| --- | --- |
| bits 31-16 | bits 15-0 |
| destination address | (source address) |

The source address is ignored when writing frames to transmit: all transmitted frames have a source address of 1.

### 0x104: frame type, control and status

|  |  |
| --- | --- |
| bits 15-8 | bits 7-0 |
| frame type | control/status |

A write to address 0x104 performs a write of the destination address, frame type, control and status and service channel address and data to the transmit FIFO

### 0x108: service channel address and data

|  |  |
| --- | --- |
| bits 31-16 | bits 15-0 |
| service channel address | service channel data |

## 0x10C: send sync frame

Writing any value to address 0x10C causes the FCI FPGA to send a sync frame to the head. See ref [5].

## 0x110 bits 2-0: scheduler control

Writing to bits 2–0 at address 0x110 writes the *scheduler control register*.

The scheduler allows head comms sync frames and DSP comms messages to be sent periodically without the intervention of the host controller.

|  |  |
| --- | --- |
| bit | description |
| 2 | DSP service channel ‘do not send’ |
| 1 | enable DSP comms scheduler |
| 0 | enable head comms scheduler |

If no data is sent to the DSP, SSC Comm Fault will be active in MDM Setup [4]. Enabling the scheduler solves this problem.

If the scheduler is enabled, it runs continually at a period defined by the *scheduler period* register at address 0x114 (2.10).

The *head comms scheduler* transmits a sync frame to the head.

The *DSP comms scheduler* constructs a TAP and sends it to the DSP:

* The TAP data payload is the four 16-bit words at head comms DPR locations 0x0920, 0x0921, 0x0930 and 0x0931:

|  |  |  |
| --- | --- | --- |
| data word 1 from DPR 0x0930 | data word 2 from DPR 0x0931 | |
| data word 3 from DPR 0x0960 | data word 4 from DPR 0x0961 | |
|  | control byte | service channel byte |

* The TAP control byte *drive enable* bit is as written to the TAP control byte at address 0x88 (2.5.3).
* If a byte is present in the DSP service channel transmit FIFO, the byte is put in the TAP service channel byte and the TAP control byte sync bit is toggled.
* The host controller may write bit 2 (DSP service channel ‘do not send’), which is written to the TAP control byte ‘SC do not send’ bit, to prevent the DSP sending further service channel bytes – e.g. in case the DSP service channel receive FIFO is full.

## 0x114: scheduler period

Writing to bits 31-0 writes the *scheduler period*. The scheduler period defines the time between scheduler ticks in units of 50MHz clock periods (20ns). If the relevant scheduler is enabled in the scheduler control register (see 2.9), when the tick occurs the scheduler will send a message to the head and/or the DSP.

## 0x118 bit 0: ‘have received frame’ bit

Bit 0 at address 0x118 is the *have received frame* bit. The bit is set to ‘1’ when a head comms frame is received without CRC error, and reset to ‘0’ when address 0x118 is read over the USB bus.

## 0x11C head comms control

Writing to address 0x11C writes to the *head comms control register*.

|  |  |  |
| --- | --- | --- |
| bit | description | default |
| 31-1 | unused | 0 |
| 0 | include process data in transmit frame | 0 |

If bit 0, *include process data in transmit frame*, is set to ‘1’ then a frame sent by the FCI will include process data, assuming that a process data map has been configured. If bit 0 is ‘0’ then transmitted frames will not include any process data even if a map has been configured.

## 0x120 head power control

Writing to bit 0 at address 0x120 writes to the *head power control register*.

|  |  |  |
| --- | --- | --- |
| bit | description | default |
| 31-2 | unused | 0 |
| 1 | enable USB to head power | 0 |
| 0 | enable head power | 1 |

If bit 0, *enable head power*, is set to ‘1’ then the isolated 20V head power supply is enabled and 20V is supplied to the head. If bit 0 is set to ‘0’ the head supply voltage is switched off.

If bit 1, *enable USB to head power*, is set to '1' then the POWER\_CONTROLLER entity automatically switches between the USB to 20V converter and the external to 20V converter. If bit 1 is set to '0' then the USB to 20V converter is disabled and no power will be supplied from the head connector unless an external power supply is connected.

### Power Controller

The power controller entity controls the USB to head power, external supply to head power, external supply to 5V, external supply to 12V, powerpath controller and head supply enable MOSFET.

When connecting the external supply, the external to 5V supply is enabled when the external supply is good (determined from the inrush current limiter), then the external to 12V supply (for powering the motor driver logic) is turned on once the external to 5V supply is good, then finally after a short delay (1ms) to ensure the external to 5V supply is stable the powerpath controller is switched to use the external supply. When disconnecting the external supply, the powerpath controller is immediately switched to use the USB supply, and the external supplies are all disabled.

If the head supply is enabled but the *enable USB to head power* bit is not set, the head supply is only enabled when the external supply is connected. If *enable USB to head power* is set, the head supply is also enabled when only USB power is present. The USB to head supply is adversely affected by voltage drops in the cable and therefore the shortest, heaviest gauge USB cable available should be used.

When the Power Controller turns on the USB to head power supply, the head power MOSFET will initially be turned off as the head power capacitors need to be charged before connecting the head to meet the inrush current requirements of the PH20 head. The head power MOSFET is turned on after 500ms, and then if at any time after a further 500ms the USB to head supply is not good (according to the LTC4151 I2C controller), the head power MOSFET will be turned off for another 500ms then turned on again whilst the supply itself remains enabled.

If the external supply is connected when the USB supply is active, the Power Controller will immediately switch to the external to head power supply and disable the USB to head supply. The USB to head supply will be immediately re-enabled after disconnecting the external supply.

## 0x124 DSP service channel TX FIFO status

Bit 0 of the DSP service channel TX FIFO status register indicates whether the DSP service channel transmit FIFO is full. If bit 0 is ‘1’ the FIFO is full, and further writes to the FIFO risk corrupting its contents.

## 0x200–0x23C: Node source address array

The node source address list is an array of up to 32 16-bit source addresses. The FlexComms Interface PCB FPGA ‘listens’ for incoming frames from addresses in the array and when a frame is received with a source address that matches a source address in the node source address array, a corresponding bit is set in the *frames received* register (read at address 0x240).

Table 1: node source address array

|  |  |  |
| --- | --- | --- |
| address | bits | description |
| 0x200 | 15 - 0 | node source address array[ 0 ] |
|  | 31 – 16 | node source address array[ 1 ] |
| 0x204 | 15 - 0 | node source address array[ 2 ] |
|  | 31 – 16 | node source address array[ 3 ] |
| 0x208 | 15 - 0 | node source address array[ 4 ] |
|  | 31 – 16 | node source address array[ 5 ] |
| 0x20C | 15 - 0 | node source address array[ 6 ] |
|  | 31 – 16 | node source address array[ 7 ] |
| 0x210 | 15 - 0 | node source address array[ 8 ] |
|  | 31 – 16 | node source address array[ 9 ] |
| 0x214 | 15 - 0 | node source address array[ 10 ] |
|  | 31 – 16 | node source address array[ 11 ] |
| 0x218 | 15 - 0 | node source address array[ 12 ] |
|  | 31 – 16 | node source address array[ 13 ] |
| 0x21C | 15 - 0 | node source address array[ 14 ] |
|  | 31 – 16 | node source address array[ 15 ] |
| 0x220 | 15 - 0 | node source address array[ 16 ] |
|  | 31 – 16 | node source address array[ 17 ] |
| 0x224 | 15 - 0 | node source address array[ 18 ] |
|  | 31 – 16 | node source address array[ 19 ] |
| 0x228 | 15 - 0 | node source address array[ 20 ] |
|  | 31 – 16 | node source address array[ 21 ] |
| 0x22C | 15 - 0 | node source address array[ 22 ] |
|  | 31 – 16 | node source address array[ 23 ] |
| 0x230 | 15 - 0 | node source address array[ 24 ] |
|  | 31 – 16 | node source address array[ 25 ] |
| 0x234 | 15 - 0 | node source address array[ 26 ] |
|  | 31 – 16 | node source address array[ 27 ] |
| 0x238 | 15 - 0 | node source address array[ 28 ] |
|  | 31 – 16 | node source address array[ 29 ] |
| 0x23C | 15 - 0 | node source address array[ 30 ] |
|  | 31 – 16 | node source address array[ 31 ] |

**Example**: the host controller writes 0x00000002 to address 0x200, and 0x00030000 to address 0x22C.

When the FlexComms Interface PCB FPGA receives a frame addressed to itself with a source address of 0x02, it sets bit 0 in the frames received register to ‘1’.

When the FlexComms Interface PCB FPGA receives a frame addressed to itself with a source address of 0x03, it sets bit 23 in the frames received register to ‘1’.

## 0x240: Frames received

The *frames received* register is a 32-bit register whose bits are set when a frame is received whose source address matches a source address in the node source address array – see 2.11.

**Example**: if a frame is received with source address *A*, and *node source array[n]* *=* *A*, then bit *n* in the frames received register is set to ‘1’.

A write of any value to address 0x240 resets the frames received register to 0x00000000.

## 0x300–0x37C: head comms service channel response

Reading addresses 0x300 – 0x37C reads the head comms service channel address and data (and some status word bits) from the 32 most recently received head comms frames since the last sync frame. When the number of head comms frames received since the last sync frame exceeds 32, the index rolls over to zero – so the service channel data from the 33rd response frame is at address 0x300.

|  |  |  |
| --- | --- | --- |
| address | bits | description |
| 0x300 | 31 | 1st received status bit 4 (bulk sequence bit) |
|  | 30 | 1st received status bit 2 (CRC error in last frame) |
|  | 29 | 1st received status bit 1 (SC write) |
|  | 28 | 1st received status bit 0 (SC read) |
|  | 27 – 16 | 1st received service channel address |
|  | 15 - 0 | 1st received service channel data |
| 0x304 |  | 2nd received service channel data, etc… |
|  |  |  |
| 0x37C |  | 32nd received service channel data, etc… |
|  |  |  |

**Example**: the manager node sends a sync frame to a head comprising a stator node and a rotor node. The stator responds first: its service channel address and data are stored at address 0x300, along with the status bits. Next the rotor responds: its service channel address and data are stored at address 0x304. (The times at which each node responds is determined by the node’s cyclic response delay – see ref [5].) Following transmission of the next sync frame, the stator’s service channel response is again stored at address 0x300 and the rotor’s service channel response is stored at 0x304.

## 0x0380 head comms CRC errors

Reading address 0x380 reads the head comms CRC errors register. Bit 0 in the register is ‘1’ if the most recent frame received since transmission of the last sync frame had a CRC error. Bit 1 is ‘1’ if the second most recent frame received since the last sync frame had a CRC error. And so on.

## 0x0384 transmit frame FIFO full

If bit 0 at address 0x0384 is ‘1’ the transmit frame FIFO is full – see 0. Writing to the FIFO when it is full may corrupt data in the FIFO.

## 0x038C force error count

The *force error count* register gives the host controller a way of writing to the comms *force error count* register (2.31.16), which is otherwise only accessible via the comms service channel.

## 0x0390 – 0x398 frame error counting

The comms error count object (2.31.15) provides a method for counting the number of frames received with CRC errors. A different type of comms error occurs if an expected frame never arrives at the receiver, or if it arrives but has been corrupted to such an extent that the receiver does not recognize it as a frame. The registers at addresses 0x390 – 0x398 allow this second type of comms error to be counted.

The *expected number of response frames* (address 0x398) is a value written to the FCI by the host controller, and representing the number of response frames expected between two sync frames.

When a sync frame is transmitted, if the number of response frames received since the last sync frame is not equal to *expected number of response frames*, then 8 is added to the value of *frame error count* (address 0x390). If the number of response frames received since the last sync frame is equal to *expected number of response frames*, then 1 is subtracted from the value of *frame error count*.

*Max frame error count* (address 0x394) keeps track of the maximum value reached by *frame error count*. *Max frame error count* is reset to 0 automatically after it is read by the host controller – i.e. the action of performing the USB-bus read resets the max count.

Frame error count is reset to 0 by writing to *reset comms counters* (address 0x00C) – see 2.2.

## 0x39C automatic command frame control

The *automatic command frame control* register has 1 meaningful bit:

|  |  |  |
| --- | --- | --- |
| bit | description | default |
| 31-1 | unused | 0 |
| 0 | automatically send command frame | 0 |

### Bit 0: automatically send command frame

Bit 0 enables the automatic transmission of command frames.

When bit 0 is ‘1’, if, at the point in time *automatic command frame transmission delay* clock cycles after the sync frame begins to be transmitted, there is a command frame in the transmit FIFO then the command frame is automatically transmitted now.

Note that setting this bit to ‘1’ enables the automatic transmission functionality straight away (the *automatic command frame transmission delay* timer runs continually, whether automatic command frame transmission is enabled or not). So when this bit is set to ‘1’, if the timer has expired, the next frame written to the transmit FIFO in this cycle (i.e. before the next sync frame is sent) will be discarded.

## 0x3A0 automatic command frame transmission delay

The *automatic command frame transmission delay* is the number of clock cycles after the start of transmission of a sync frame that a command frame is automatically transmitted, provided there is a frame in the transmit FIFO and the *automatically send command frame* bit in the *automatic command frame control* register is ‘1’.

A new value written to this register takes effect the next time a sync frame is sent – i.e. in the next cycle.

Note that the FPGA does not distinguish between frame types in the FIFO: if there is *any* frame in the transmit FIFO at the end of the delay and the appropriate bits are set it will be transmitted, irrespective of its type.

If the *automatically send command frame* bit is ‘1’, if there wasn’t a frame in the transmit FIFO ready to be transmitted after *automatic command frame transmission delay* clock cycles, then the first frame (of any type) written to the transmit FIFO after the delay has expired will be discarded and not transmitted. The *frame discarded* bit of the *automatic command frame status* register will be set to ‘1’ for the duration of the cycle. Subsequent frames written to the transmit FIFO in the cycle (i.e. before the next sync frame) will be transmitted normally.

## 0x3A4 automatic command frame status

The *automatic command frame status* register provides information regarding the automatic transmission of command and response frames.

|  |  |
| --- | --- |
| bit | description |
| 31-4 | unused (‘0’) |
| 3 | frame discarded |
| 2 | first frame received after command had CRC error |
| 1 | response frame received with no error |
| 0 | command frame pending transmission |

### Bit 0: command frame pending transmission

Bit 0 is set when a command frame is written to the transmit FIFO. Bit 0 is set even if the command frame is not sent – for example, when it is written to the transmit FIFO after the *automatic command frame transmission delay* has elapsed. Bit 0 is cleared when the command frame is transmitted, and when the *automatic command frame status* register is read (over the USB bus).

### Bit 1: response frame received with no error

Bit 1 is set when a response frame is received with no CRC error, and cleared when the *automatic command frame status* register is read (over the USB bus).

### Bit 2: first frame received after command had CRC error

Bit 2 is set when a CRC error is detected in the first frame received after transmitting a command frame. It is cleared when the *automatic command frame status* register is read (over the USB bus).

### Bit 3: frame discarded

Bit 3 is set when a frame is discarded from the transmit FIFO and not transmitted. This happens when the *automatically send command frame* bit in the *automatic command frame control* register is ‘1’, but a command frame hasn’t been written to the transmit FIFO before the *automatic command frame transmission delay* timer has elapsed. Bit 3 is cleared when the next sync frame is sent.

## 0x1000, bits 7-0: DSP service channel FIFO

Writing a byte to 0x1000 bits 7-0 puts the byte in the *DSP service channel transmit FIFO*. Reading from 0x1000 bits 7-0 reads a byte from the *DSP service channel receive FIFO*.

When a byte is received on the DSP service channel (i.e. the sync bit toggles in the incoming TAP status byte), the byte is automatically written to the DSP service channel read FIFO.

When the DSP scheduler is enabled (see 2.9), bytes in the DSP service channel transmit FIFO are automatically sent to the DSP – one byte is sent on every scheduler tick. The TAP control word sync bit is toggled automatically by the scheduler.

## 0x1004: Service channel number of bytes in FIFO

Address 0x1004 contains the number of bytes in the DSP service channel receive FIFO.

## 0x1008: LTC4151 Config and Status

This register configures and reports the status of the LTC4151. There is no need to write this register for normal use.

Note – the LTC4151 won’t work at all when the 20V isolated power supply is turned off, meaning the Configured bit won’t ever go high until the Enable bit is set and the 20V isolated supply is switched on. The Enable bit is set by default.

|  |  |  |
| --- | --- | --- |
| **bit** | **description** | **default** |
| 8 | Configured (Read only) | 0 |
| 7 | Reconfigure | 0 |
| 6 | Enable | 1 |
| 5 | ADC Snapshot Mode Enable (0 recommended) | 0 |
| 4 | ADC Channel Label for snapshot mode | 0 |
| 3 | ADC Channel Label for snapshot mode | 0 |
| 2 | Test mode (0 recommended) | 0 |
| 1 | Page read/write enable (1 recommended) | 1 |
| 0 | Stuck-bus timer enable (1 recommended) | 1 |

When ADC Snapshot Mode is enabled, only the ADC channel selected by bits 4 and 3 is measured by the ADC instead of all channels being measured. For normal operation, ADC Snapshot Mode should be disabled.

|  |  |  |
| --- | --- | --- |
| **Bit 4** | **Bit 3** | **ADC Channel** |
| 0 | 0 | SENSE (Default) |
| 0 | 1 | VIN |
| 1 | 0 | ADIN |

If the configuration needs to be changed after power-on, the Reconfigure bit must be set.

## 0x100C: LTC4151 Current Sense

12-Bit Data of Current Sense Voltage with 20μV LSB and 81.92mV Full-Scale stored in the least significant 12 bits of this register. This corresponds to 1mA per LSB.

## 0x1010: LTC4151 VIN

This register contains the 12-Bit Data of VIN Voltage with 25mV LSB and 102.4V Full-Scale.

## 0x1014 Temperature

This register contains the temperature measured by the MAX1286 which is physically positioned between the FPGA and FPGA power supply.

|  |  |
| --- | --- |
| **bit** | **description** |
| 31-13 | Reserved (‘0’) |
| 12 | Sign |
| 11-0 | 12-bit temperature reading (0.0625°C/LSB) |

## Head comms dual-port RAM (0x8000-…)

The head comms dual-port RAM (DPR) is located at address 0x8000 onwards. The DPR is 16-bits wide, but is addressed in 32-bit mode, the 16 bits of DPR data being located in the least-significant 2 bytes of the four data bytes returned over the USB bus. To read from address 0x0000 in the DPR, read bits 0 – 15 from address 0x8000. To read from address 0x0001 in the DPR, read bits 0 – 15 from address 0x8004. To write to address 0x0002, write bits 0 – 15 to address 0x8008. And so on.

The DPR contains objects received from or sent to a remote node – e.g. a PH20 head, a REVO, a REVO probe – and objects relating to the comms link. The format of the DPR is fixed for all nodes, with some addresses reserved for particular uses: see ref [5] for full details.

### Head comms object address memory map

Table 2 shows the DPR addresses of the head comms objects. Access type is in the r/w column: ro = read only, rw = read/write.

Table 2: Head comms object address map

|  |  |  |  |
| --- | --- | --- | --- |
| object address | HIC address | object name | r/w |
| [0x0000](#_Node_type_(object_1) | 0x8000 | node type = 1 | ro |
| [0x0001](#_FPGA_version) | 0x8004 | FPGA version major | ro |
| [0x0002](#_FPGA_version_(object) | 0x8008 | FPGA version minor | ro |
| [0x0003](#_FPGA_process_delays) | 0x800C | FPGA delay d0 | ro |
| [0x0004](#_FPGA_process_delays) | 0x8010 | FPGA delay d1 | ro |
| [0x0005](#_FPGA_process_delays) | 0x8014 | FPGA delay d2 | ro |
| [0x0006](#_FPGA_process_delays) | 0x8018 | FPGA delay d3 | ro |
| [0x0007](#_Comms_receive_buffer) | 0x801C | comms receive buffer size | ro |
| [0x0008](#_Comms_transmit_buffer) | 0x8020 | comms transmit buffer size | ro |
| [0x0009](#_Clock_frequency_(object) | 0x8024 | clock frequency high word | ro |
| [0x000A](#_Clock_frequency_(object) | 0x8028 | clock frequency low word | ro |
| [0x000B](#_Cyclic_response_delay) | 0x802C | cyclic response delay (clock cycles) high word | rw |
| [0x000C](#_Receive_count_(object) | 0x8030 | cyclic response delay (clock cycles) low word | rw |
| [0x000D](#_Sync_acquisition_delay_1) | 0x8034 | sync acquisition delay | rw |
| [0x000E](#_Comms_control_(object) | 0x8038 | comms control | rw |
| [0x000F](#_Receive_count_(object) | 0x803C | receive count high word | ro |
| [0x0010](#_Receive_count_(object) | 0x8040 | receive count low word | ro |
| [0x0011](#_Transmit_count_(object) | 0x8044 | transmit count high word | ro |
| [0x0012](#_Transmit_count_(object) | 0x8048 | transmit count low word | ro |
| [0x0013](#_Error_count_(object) | 0x804C | error count high word | ro |
| [0x0014](#_Error_count_(object) | 0x8050 | error count low word | ro |
| 0x0015 | 0x8054 | not used |  |
| 0x0016 | 0x8058 | not used |  |
| 0x0017 | 0x805C | not used |  |
| [0x0018](#_Force_error_count) | 0x8060 | force error count | rw |
| …. |  |  |  |
| [0x001A](#_Transmit_block_size) | 0x8068 | transmit block size | rw |
| [0x001B](#_Receive_block_buffer) | 0x806C | receive block buffer address | rw |
| [0x001C](#_Transmit_block_buffer) | 0x8070 | transmit block buffer address | rw |
| …. |  |  |  |
|  |  | **transmit process data node address list** |  |
| 0x0020 | 0x8080 | transmit node 0 node address | rw |
| 0x0021 | 0x8084 | transmit node 1 node address | rw |
| … | … |  |  |
| 0x005F | 0x817C | transmit node 63 node address | rw |
| 0x0060 | 0x8180 | transmit node 0 transmit process data map DPR address | rw |
| 0x0061 | 0x818C | transmit node 1 transmit process data map DPR address | rw |
| … | … |  |  |
| 0x009F | 0x827C | transmit node 63 transmit process data map DPR address | rw |
|  |  |  |  |
|  |  |  |  |
|  |  | **transmit process data maps** |  |
|  |  | **transmit node 0 process data map** |  |
| 0x00A0 | 0x8280 | transmit process data object 0 | rw |
| 0x00A1 | 0x8284 | transmit process data object 1 | rw |
| … | … | …. |  |
| 0x00AF | 0x82BC | transmit process data object 15 | rw |
| 0x00B0… | 0x82C0 | **transmit node 1 process data map…** |  |
| 0x00C0… | 0x8300 | **transmit node 2 process data map…** |  |
| … | **…** |  |  |
| 0x0490… | 0x9240 | **transmit node 63 process data map…** |  |
|  |  |  |  |
|  |  | **receive process data node address list** |  |
| 0x04A0 | 0x9280 | receive node 0 node address | rw |
| 0x04A1 | 0x9284 | receive node 1 node address | rw |
| … | … |  |  |
| 0x04DF | 0x937C | receive node 63 node address | rw |
| 0x04E0 | 0x9380 | receive node 0 receive process data map DPR address | rw |
| 0x04E1 | 0x9384 | receive node 1 receive process data map DPR address | rw |
| … | … |  |  |
| 0x051F | 0x947C | receive node 63 receive process data map DPR address | rw |
|  |  |  |  |
|  |  | **receive process data maps** |  |
|  |  | **receive node 0 process data map** |  |
| 0x0520 | 0x9480 | receive process data object 0 | rw |
| 0x0521 | 0x9484 | receive process data object 1 | rw |
| … | … |  |  |
| 0x052F | 0x94BC | receive process data object 15 | rw |
| 0x0530… | 0x94C0 | **receive node 1 process data map…** |  |
| 0x0540… | 0x9500 | **receive node 2 process data map…** |  |
| … | … |  |  |
| 0x0910… | 0xA440 | **receive node 63 process data map…** |  |
|  |  |  |  |
| 0x0920… | 0xA480… | ***this address onwards free for process data…*** |  |
| … | … |  |  |
| 0x0FFF | 0xBFFC | **end of dual-port RAM** |  |

### Node type

The *node type* is a value which uniquely identifies the type of node. The FCI has a node type of 0x0001.

### FPGA version

One 16-bit word represents the FPGA firmware version number. The normal FlexComms version numbering scheme, where node FPGA versions are represented by major and minor version numbers is not used here, in order to retain backward compatibility with software. Instead, a single version number is used.

Also read at FCI address 0x0000 – see .

### FPGA process delays d0, d1, d2, d3

*FPGA process delay* *d0* (object address 0x0003) is the number of clock cycles required by the FPGA to construct an outgoing sync frame.

*FPGA process delay* *d1* (object address 0x0004) is the number of clock cycles required by the FPGA to construct an outgoing standard frame, assuming no process data is included in the outgoing frame. If process data is included in the frame, *d1* is extended by a number of clock cycles: the number of clock cycles depends on the location of the destination node address in the transmit process data node address list (the lower the address of the destination node in the address list, the faster the search logic will locate the transmit process data map), and the number of process data objects included in the outgoing frame.

*FPGA process delay* *d2* (object address 0x0005) is the number of clock cycles required by the FPGA to decode an incoming sync frame.

*FPGA process delay* *d3* (object address 0x0006) is the number of clock cycles required by the FPGA to decode an incoming standard frame, assuming no process data is included in the incoming frame. If process data is included in the frame, *d3* is extended by a number of clock cycles: the number of clock cycles depends on the location of the receive node address in the receive process data node address list (the lower the address of the receive node in the address list, the faster the search logic will locate the receive process data map), and the number of process data objects included in the incoming frame.

shows the FCI FPGA delays, in FPGA clock cycles. The standard frame delays shown in assume that the transmit FIFO () is empty: if the host controller writes to address 0x104 to send a standard frame and there is data to transmit in the transmit FIFO, the frame will not be transmitted until the preceding frames in the FIFO have been transmitted. When more than one frame is present in the FIFO, the frames will be transmitted consecutively with an inter-frame gap of *d1* + 20 clock cycles.

TX\_TIME and RX\_TIME are the transmit and receive frame timestamps (see ).

Delays *t0* and *t1* represent the time it takes the FPGA to recognise a bus write, and vary depending on the bus (USB), the host controller and the relationship between the host controller clock and the FPGA clock.

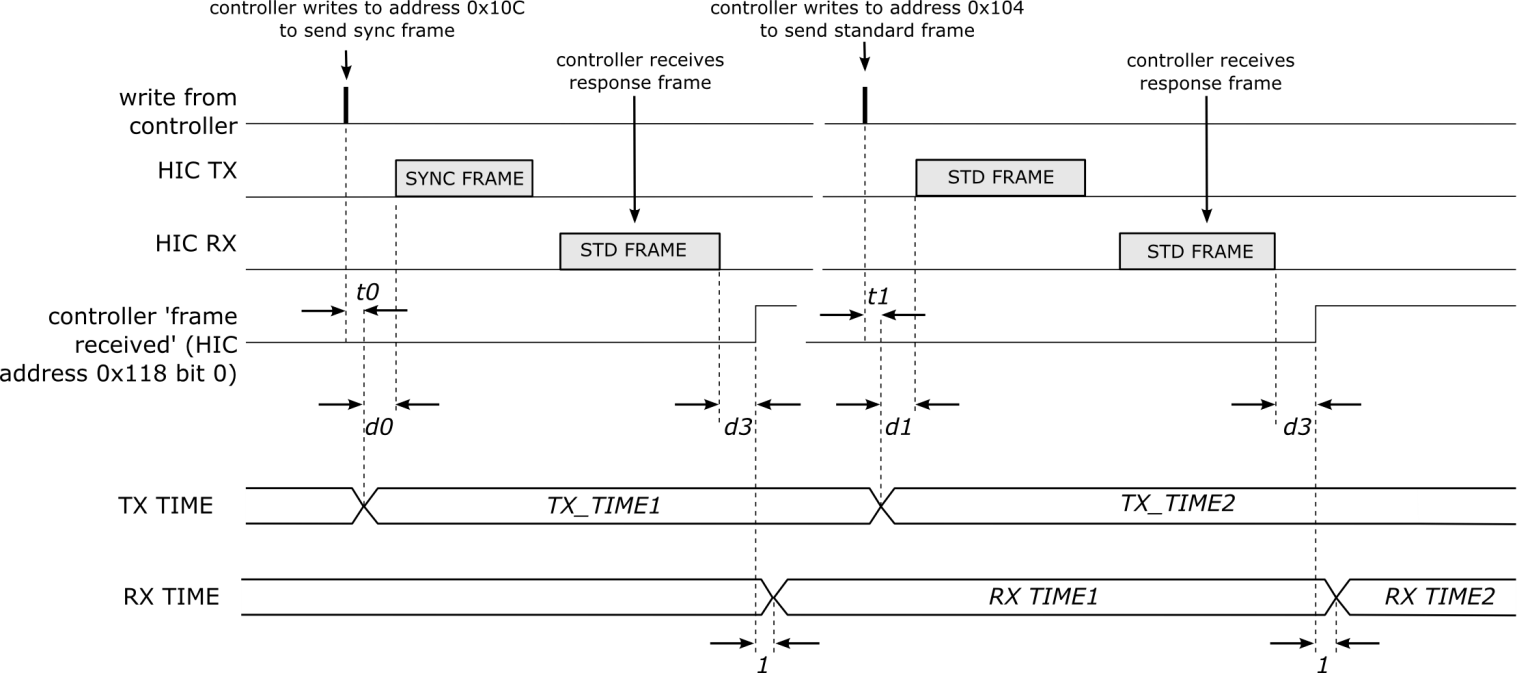


Figure 1: FPGA delays, in clock cycles

### Comms receive buffer size

The *comms receive buffer size* is the size, in 16-bit words, of the buffer used to store incoming frame data. In other words, the comms receive buffer size is the maximum number of process data objects that may be sent to the FCI in a single frame; and also the maximum number of data words in a bulk data frame that the FCI may receive.

### Comms transmit buffer size

The *comms transmit buffer size* is the size, in 16-bit words, of the buffer used to store outgoing frame data. In other words, the comms transmit buffer size is the maximum number of process data objects that may be sent from the FCI in a single frame; and also the maximum number of data words in a bulk data frame that the FCI may transmit.

### Clock frequency

This is a 32-bit value representing the FPGA’s main system *clock frequency*, in Hz. The Gyro/PH20 stator clock frequency is 50MHz = 0x02FAF080.

Table 3: FCI FPGA clock frequency objects

|  |  |
| --- | --- |
| clock frequency high word (0x0006) | clock frequency low word (0x0007) |
| 0x02FA | 0xF080 |

### Cyclic response delay

A 32-bit unsigned value representing the *number of clock cycles - 1* between the reception of a sync frame and the transmission of a response, after FPGA processing delays have been considered.

**Note:** The minimum cyclic response delay is 1 clock cycle. If the value of cyclic response delay stored in the FPGA is zero, the cyclic response delay will be 1 clock cycle.

Typically the FCI manager node won’t use the cyclic response delay objects.

### Sync acquisition delay

The *sync acquisition delay* specifies the *number of clock cycles – 1* between the reception of a sync frame and the latching of input data.

The sync acquisition delay is designed to allow multiple nodes which receive sync frames at different times to latch inputs simultaneously. Typically the FCI manager node won’t use this object.

### Comms control

The comms control object contains the following bits:

|  |  |
| --- | --- |
| bit | description |
| 2-15 | not used |
| 1 | send command response on sync |
| 0 | cyclic data enable |

### Comms control bit 0: cyclic data enable

The *cyclic data enable* bit is used to enable or disable cyclic data. When the cyclic data enable bit is ‘1’, the node will send a frame in response to reception of a sync frame; when ‘0’, the node does nothing following reception of a sync frame.

The default value of the cyclic data enable bit following reset is ‘1’.

### Comms control bit 1: send command response on sync

When the *send command response on sync* bit is ‘0’, a command response frame is sent *command-response delay* clock cycles after reception of a command frame.

When ‘1’, a command response frame is sent *command-response delay* clock cycles after reception of the next sync frame following reception of the command frame.

### Receive count

A 32-bit value representing the total number of comms frames (including sync frames) received. The count is reset by writing to address 0x0C (2.2).

### Transmit count

A 32-bit value representing the total number of comms frames (including sync frames) transmitted. The count is reset by writing to address 0x0C (2.2).

### Error count

A 32-bit value representing the total number of comms frames (including sync frames) received with CRC errors. The count is reset by writing to address 0x0C (2.2).

### Force error count

The *force error count* (FEC) is a 16-bit value used for used for testing comms error detection. When the FEC is greater than zero, a CRC error will be forced in the transmitted frame once every FEC frames. For example, if FEC is 100, every 100th transmitted frame will have a CRC error.

### Transmit block size

The *transmit block size* is a 16-bit value representing the number of 16-bit words contained in the data section of a command and command-response frame. See ref [6] for details of the command-response transfer mechanisms.

### Receive block buffer address

The *receive block buffer address* is the 16-bit DPR address of the start of the receive buffer for command and command-response frame data. Data from an incoming command or command-response frame is automatically unpacked into DPR starting at the address specified by the receive block buffer address. See ref [6] for details of the command-response transfer mechanisms.

### Transmit block buffer address

The *transmit block buffer address* is the 16-bit DPR address of the start of the transmit buffer for command and command-response frames. Data in an outgoing command or command-response frame is automatically copied into the frame from DPR starting at the address specified by the transmit block buffer address. See ref [6] for details of the command-response transfer mechanisms.

### Comms process data objects

Objects at addresses 0x0020 to 0x091F control how the FCI maps process data into frames it transmits, and from frames it receives. These objects are common to all nodes participating in process data transfer, and are described in detail in ref [6].

## 0xFF00 USB Interface Control Byte

|  |  |  |
| --- | --- | --- |
| **Bit** | **Description** | **Default** |
| 31..2 | Reserved |  |
| 2 | Stream mode | 0 |
| 1 | Reserved | 0 |
| 0 | Streaming Enable | 0 |

### Bit 2: Stream mode

0: USB stream interval is in units of milliseconds, i.e. a stream interval of 10 will result in a stream being sent every 10 milliseconds. Use this mode when streamed data is required at constant intervals.

1: USB stream interval is in units of number of frames received from the head, i.e. a stream interval of 10 will result in a stream being sent every 10 times a rising edge is detected on the HAVE\_RECEIVED\_FRAME bit. Use this mode when streamed data must be synchronized with data received from the head.

### Bit 0: Streaming Enable

1: The DPR and DSP streaming mode is enabled.

0: The DPR and DSP streaming mode is disabled.

## 0xFF08 USB Interface Quantity of Addresses to Stream

This register sets how many words of data are transmitted via the USB interface every streaming interval. 0 means one byte will be streamed and 15 means 16 bytes will be streamed. The addresses of the words to stream are defined in 0xFF10-0xFF4C and the USB Interface Control Byte is used to enable streaming.

Default value: 0

## 0xFF0C USB Interface Stream Interval

The value in this register defines how often the data read from the addresses set in 0xFF10-0xFF8C are sent in milliseconds (stream mode 0) or number of frames received from the head, up to once every 65535 milliseconds/frames.

Default value: 1000

## 0xFF10-0xFF8C USB Interface Addresses to Stream

The values in these registers are the addresses from which the data to stream is read. The first word streamed is that addressed by 0xFF10, then the next 0xFF14 and so on.

Default values: 0

## 0xFF90 Stream Config 1

The value in this register determines which streamed addresses are streamed from the DPR and which are streamed from the DSP. The DSP comms scheduler must be enabled before streaming any data from the DSP. A 1 in a bit means the corresponding register will be streamed from the DSP; otherwise it will be streamed from the DPR.

Default value: 0

|  |  |
| --- | --- |
| **Bit** | **Description** |
| 31 | 32nd word/ DSP selection |
| 30 | 31st word/ DSP selection |
| … | … |
| 1 | 2nd word/ DSP selection |
| 0 | 1st word/ DSP selection |

## 0xFF94 Stream Config 2

The value in this register determines which DSP reads are two byte reads and which are four byte reads. These bits must be configured correctly for all DSP words to be streamed, it is not sufficient to set them all to 1 (4 byte reads) when some are 2 byte reads. The bits have no effect for DPR reads.

Default value: 0

|  |  |
| --- | --- |
| **Bit** | **Description** |
| 31 | 32nd word/ 4 byte selection |
| 30 | 31st word/ 4 byte selection |
| … | … |
| 1 | 2nd word/ 4 byte selection |
| 0 | 1st word/ 4 byte selection |

# USB Interface DSP streaming worked example

The following two words will be streamed every 100 milliseconds:

* Current time (0x6C) from FPGA/DPR
* Servo Counter (P-0-259) from DSP (4 bytes)

To enable streaming of these registers, the following writes must be done:

1. Set number of addresses to stream to 2: Write 0x00000001 to 0xFF08.
2. Set stream interval to 100 milliseconds: Write 0x00000064 to 0xFF0C.
3. Set address of word read from FPGA/DPR: Write 0x0000006C to 0xFF10.
4. Set address of word read from DSP:

Desired address is P-0-259. This is converted to a 16-bit address according to the below table:

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| **Bit number** | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
| **Description** | Range  P=1, S=0 | Parameter Set  (0-7) | | | Parameter Number  (0-4095) | | | | | | | | | | | |
| **Value** | 1 | 0 | 0 | 0 | 0 | 0 | 0 | 1 | 0 | 0 | 0 | 0 | 0 | 0 | 1 | 1 |
| **Value (Hex)** | 8 | | | | 1 | | | | 0 | | | | 3 | | | |

The address of the DSP register is therefore 0x8103.

Write 0x8103 to 0xFF14.

1. Set Stream Config 1: Word 0 is streamed from the FPGA/DPR and Word 1 is streamed from the DSP, so bit 0 must be 0 and bit 1 must be 1.

Write 0x00000002 to 0xFF90.

1. Set Stream Config 2: Word 1 is streamed from the DSP and is four bytes long so but 1 must be 1. Word 0 is streamed from the FPGA/DSP so its bit in this register doesn't matter.

Write 0x00000002 to 0xFF94.

1. Enable streaming: Write 0x00000001 to 0xFF00.