**DRXP**

**Driver**

**Specification**

Elecs Industry Co.,Ltd

**Revision History**

|  |  |  |  |
| --- | --- | --- | --- |
| Version | date | Writer | Content |
| 1.0.0 | 2016/07/05 | K.Shimoyama | First edition. |
| 1.0.1 | 2016/07/14 | K.Shimoyama | Figure.1.1 Overview of the reception sequence  Data Buffer is changed.(4Group ->16, 32 or 64  Group[default:16Group])  Figure.1.2　Overview of the transmission sequence  Data Buffer is changed.(Delete unnecessary　memory area)  2.1. Structure of Receive Ring Buffer Mode  Data Buffer is changed.(4Group ->16, 32 or 64  Group[default:16Group])  2.2. Structure of Transmission Buffer  Data Buffer is changed.(Delete unnecessary　memory area)  3.2.Time Frame  Name is changed to”Information Frame”.  The contents also change.  4.4. Example of Reception  “ioctl(DRXPD\_RCVGROUPS)” is added.  5.5.7. Command :DRXPD\_GET\_RCVSTS  　　Structure of ST\_DRXPD\_RCVSTS is changed.  “GroupNum” is added.  5.5.17. Command :DRXPD\_RCVGROUPS  Added new.  6.1.11. GET\_DFR\_SN  Data is changed.  6.1.12. GET\_DFR\_VER  Added new.  Other  Reception completion notification is affected by the usage of the PCIe bus and system bus, there is a possibility that fluctuate with respect to "the meta- frame timing of 48ms". So, "It is synchronized to the meta- frame timing of 48ms" was deleted. |
| 1.0.2 | 2016/08/15 | K.Shimoyama | 3.2.Information　Frame  　　“UTC” is changed to “TAI”. |

　(Continued to the next page.)(Continued from the previous page.)

|  |  |  |  |
| --- | --- | --- | --- |
| 1.0.3 | 2016/08/24 | K.Shimoyama | 3.2.Information Frame  unused area size is changed to ‘20byte’from  ‘10byte’.  6.1.1. GET\_ALARM\_STATUS\_REG  　“Electrical OR”　is changed to “Logical add”.  Add the remarks column.  6.1.2. GET\_POWER\_ALARM\_REG  　“Electrical OR”　is changed to “Logical add”.  Add the remarks column.  6.1.5. GET\_DFR\_VOLTAGE\_STATUS  “+VDDR(for DDR)”　is changed to  “+(TBD)VDDR(for DDR)”.  6.2.4. DFR\_TEST\_DATA  Appended the details of Random number generatoron and Counting data.  6.2.5.INSERT\_SYNCPTN\_ERR  Added new.  6.2.6.SET\_CHKSUM\_ERR  Added new.  6.2.7.SET\_SEQCNT\_ERR  Added new.  7.1. SFP module alarm threshold  Added new. |
| 1.0.4 | 2016/08/30 | K.Shimoyama | 2.1. Structure of Receive Ring Buffer Mode  Added the comment about Linux memory  management.  3.2.Information Frame  Sequence Count Error bits are added into Frame　status.  6.1.Monitoring Items(The overall change)  　0 of bit value is normal, and 1 of bit value is alarm  (or failures).  6.1.1.GET\_ALARM\_STATUS\_REG  ‘bit 0: ALMINT’ is deleted.  　Warning bits is added. |

(Continued to the next page.)(Continued from the previous page.)

|  |  |  |  |
| --- | --- | --- | --- |
| 1.0.4 | 2016/08/30 | K.Shimoyama | 6.1.2.GET\_POWER\_ALARM\_REG  ‘bit 0: Power summary alarm’ is deleted.  Warning bits is added.  6.1.5. GET\_DFR\_VOLTAGE\_STATUS  Added the remarks column.  6.1.10. GET\_DFR\_METAFRAME\_DELAY  　Clock is changed to 200MHz from 125MHz.  Value of Data is changed to 32bit-value from  24bit-value.  Added the remarks column.  6.1.12. GET\_DFR\_VER  ItemID is changed because ID is duplicate.  7.1. SFP module warning and alarm threshold  Changed from “7.1. SFP module alarm threshold”.  Added referenced product type and comment. |
| 1.0.5 | 2016/09/06 | K.Shimoyama | 6.1.Monitoring Items(The overall change)  　Reserved bit is changed to “always 0” from  “always 1”.  6.2. Control Items(The overall change)  　Reserved bit is changed to “always 0” from  “always 1”. |
| 1.0.6 | 2016/10/06 | K.Shimoyama | 3.2.Information　Frame  Add a "Dummy data flag" to bit1 of Frame status.  5.5.15.Command:DRXPD\_GET\_I2C\_DATA  “result” member type of ST\_DRXPD\_I2CRD struct is changed to “char” from “unsigned char”.  5.5.16.Command:DRXPD\_SET\_I2C\_CMD  “result” member type of ST\_DRXPD\_I2CCMD struct is changed to “char” from “unsigned char”.  Remark is changed.  6.1.1.GET\_ALARM\_STATUS\_REG  Added I2C Access Error bit.  “CLR\_ALARM\_STATUS\_REG” is changed to “DFR\_RESET”. |

(Continued to the next page.)(Continued from the previous page.)

|  |  |  |  |
| --- | --- | --- | --- |
| 1.0.6 | 2016/10/06 | K.Shimoyama | 6.1.2.GET\_POWER\_ALARM\_REG  Added I2C Access Error bit.  “CLR\_ALARM\_STATUS\_REG” is changed to “DFR\_RESET”.  6.1.3.GET\_SIGNAL\_AVG\_nW  Added I2C Access Error bit.  　“two’s complement” is removed from Byte0-2.  The range of values is changed.    6.1.5.GET\_DFR\_VOLTAGE\_STATUS  0 of bit value is normal, and 1 of bit value is alarm  (leakage of modify).  6.1.4.GET\_DFR\_STATUS  0 of bit value is normal, and 1 of bit value is alarm  (leakage of modify).  　Byte2 is changed to “Sync error detected” from  “Sync pattern detected”.  Added the remarks column.  6.1.5.GET\_DFR\_VOLTAGE\_STATUS  “+(TBD)VDDR” is changed to “+1.35VDDR”.  6.1.7.GET\_DFR\_SYNC\_ERR\_CNT  Added DFR\_RESET to factors of reset.  6.1.9.GET\_DFR\_TEST\_DATA  “Refer to SET\_DFR\_TEST\_DATA” is changed to  “Refer to DFR\_TEST\_DATA”.  6.1.10.GET\_DFR\_METAFRAME\_DELAY  “two’s complement” is removed from Byte0-3.  Remark is changed.  6.2.4.DFR\_TEST\_DATA  Description is changed.  6.2.7.SET\_SQCNT\_ERR  ItemID is changed because ID is duplicate.  6.1.11.GET\_DFR\_SN  Data is changed.  7.1. SFP module warning and alarm threshold  The description of the threshold value column is modified. |

(Continued to the next page.)

(Continued from the previous page.)

|  |  |  |  |
| --- | --- | --- | --- |
| 1.0.7 | 2016/10/14 | K.Shimoyama | 4.4.Example of Reception  Also added the procedure when you want to use to continue without “close”.  4.5.Example of Transmission  Also added the procedure when you want to use to continue without “close”.  “O\_WRONLY” is changed to “O\_RDWR”.  5.1.open  “O\_WRONLY”is changed to “O\_RDWR”.  5.3.mmap  When　using　reception, Specify PROT\_READ.  5.5.8.Command:DRXPD\_CLR\_RCV  Range of “clr\_group” is corrected.  6.1.1.GET\_ALARM\_STATUS\_REG  6.1.2.GET\_POWER\_ALARM\_REG  6.1.3.GET\_SIGNAL\_AVG\_nW  6.1.4.GET\_DFR\_STATUS  6.1.5.GET\_VOLTAGE\_STATUS  Added the following comment to description.  “Also when driver is opened, resets alarm/warn bit to 0.”  6.1.6.GET\_DFR\_CHKSUM\_COUNTER  6.1.7.GET\_DFR\_SYNC\_ERR\_CNT  6.1.8.GET\_DFR\_SYNC\_STATUS  6.1.9.GET\_DFR\_TEST\_DATA  6.2.4.DFR\_TEST\_DATA  Added the following comment to description.  “Also when driver is opened, resets to 0.”  6.1.11.GET\_DFR\_VER  Data is changed. |
| 1.0.8 | 2016/11/8 | Y.Hayashi | 3.1.Normal Frame  Added the comment.  3.2.Information Frame  Updated the comment.  4.1. Note on the driver use  4.2. Installing Driver  Added the comment.  4.5. Example of Transmission  Corrected,  “DRXPD\_SNDSIZ”→“DRXPD\_SET\_SNDSIZ”  5.3. mmap  Updated the comment. |

(Continued to the next page.)(Continued from the previous page.)

|  |  |  |  |
| --- | --- | --- | --- |
| 1.0.8 | 2016/11/8 | Y.Hayashi | 5.5.12. Command :DRXPD\_SET SNDSIZ  Added the comment.  6. Monitoring and Control Items  Added the ItemID name.  6.1.1.GET\_ALARM\_STATUS\_REG  6.1.2.GET\_POWER\_ALARM\_REG  6.1.3.GET\_SIGNAL\_AVG\_nW  6.1.4.GET\_DFR\_STATUS  6.1.5.GET\_VOLTAGE\_STATUS  Removed the following comment from description.  “Also when driver is opened, resets alarm/warn bit to 0.”  Added the remarks.  6.1.6.GET\_DFR\_CHKSUM\_COUNTER  6.1.7.GET\_DFR\_SYNC\_ERR\_CNT  Removed the following comment from description.  “Also when driver is opened, resets to 0.”  Added the remarks.  6.1.8. GET\_DFR\_SYNC\_STATUS  Updated the description.  6.1.10. GET\_DFR\_METAFRAME\_DELAY  Corrected “TE related” , NO→YES  6.2.1. DFR\_RESET  Updated the description.  7.1. SFP module warning and alarm threshold  Added the module type and comment. |
| 1.0.9 | 2017/08/11 | K.Shimoyama | Proofreading version |
| 1.0.10 | 2017/09/20 | K.Takezawa | 3.2 Information Frame  Modified the type of software timestamp field (TAI->UTC).  Modified the definition of unused area and unused bit value in information frame.  (always0 -> undefined )  4.2 Installing Driver  Modified the maximum board number (2->4).  5.1 open  Added acceptable arguments to first parameter.  (“/dev/drxp2” and “/dev/drxp3” ) |

(Continued to the next page.)

(Continued from the previous page.)

|  |  |  |  |
| --- | --- | --- | --- |
| 1.0.10 | 2017/09/20 | K.Takezawa | 6.1.1 GET\_ALARM\_STATUS\_REG  6.1.2 GET\_POWER\_ALARM\_REG  6.1.3 GET\_SIGNAL\_AVG\_nW  Modified the definition of Reserved bit (byte) value in Data field. (always0 ->undefined).  Modified the behavior when device is closed..  6.1.4 GET\_DFR\_STATUS  6.1.5 GET\_DFR\_VOLTAGE\_STATUS  Modified the definition of Not used bit value in Data field. (always0 ->undefined).  6.1.6 GET\_DFR\_CHKSUM\_COUNTER  Modified the behavior when open(2) is called.  6.1.7 GET\_DFR\_SYNC\_ERR\_CNT  Modified the definition of Reserved Byte value in Data field. (always0 ->undefined).  Modified the behavior when open(2) is called.  6.1.8 GET\_DFR\_SYNC\_STATUS  6.1.9 GET\_DFR\_TEST\_DATA  6.1.10 GET\_DFR\_METAFRAME\_DELAY  Modified the definition of Reserved and Not used bit (byte) value in Data field. (always0 ->undefined).  6.1.11 GET\_DFR\_SN  Modified the definition of Reserved and Not used bit (byte) value in Data field. (always0 ->undefined).  Delete the specific value (“131763” or “131764”) from Byte0-2 (Serial number) in Data field.  6.1.12 GET\_DFR\_VER  Modified the definition of Reserved and Not used bit (byte) value in Data field. (always0 ->undefined).  6.1.13. GET\_FPGA\_TEMP  Added new monitoring function. |
| 1.0.11 | 2018/01/10 | K.Takezawa | 1.1 Scope of this document  Added new subsection.  This subsection describes the scope of this document and correspondence table between driver and FPGA. |

(Continued to the next page.)

(Continued from the previous page.)

|  |  |  |  |
| --- | --- | --- | --- |
| 1.0.11 | 2018/01/10 | K.Takezawa | 3.2 Information Frame  Corrected description. Before 1.0.10 describes the information frame is correspond to last frame in the group. We corrected the relation between information frame and normal frame (last frame -> first frame).  5.5.19. Command :DRXPD\_SET\_SEUEVT  Added new comamnd.  5.5.20. Command :DRXPD\_CLR\_SEUEVT  Added new command.  6 Monitoring and Control Items  Deleted the item of “suggested interval” from each monitoring/control function subsection.  6.1.4. GET\_DFR\_STATUS  Added description to the item of “Update Rate”  6.1.5. GET\_DFR\_VOLTAGE\_STATUS  Added description to the item of “Update Rate”  6.1.6. GET\_DFR\_CHKSUM\_COUNTER  Added description to the item of “Update Rate”  6.1.7. GET\_DFR\_SYNC\_ERR\_CNT  Added description to the item of “Update Rate”  6.1.8. GET\_DFR\_SYNC\_STATUS  Added description to the item of “Update Rate”  6.1.9. GET\_DFR\_TEST\_DATA  Added description to the item of “Update Rate”  6.1.10. GET\_DFR\_METAFRAME\_DELAY  Added description to the item of “Update Rate”  6.1.11. GET\_DFR\_SN  Added description to the item of “Update Rate”  6.1.12. GET\_DFR\_VER  Added description to the item of “Update Rate”  6.1.13. GET\_FPGA\_TEMP  Deleted space from subsection title.  Deleted space from the item of ItemID.  Added sample code.  Added description to the item of “Update Rate” |

(Continued to the next page.)

(Continued from the previous page.)

|  |  |  |  |
| --- | --- | --- | --- |
| 1.0.11 | 2018/01/10 | K.Takezawa | 6.1.14. GET\_SEU\_STATUS  Added new monitoring function.  6.1.15. GET\_SEU\_ERRCNT  Added new monitoring function.  6.1.16. GET\_SEU\_ELAPSED  Added new monitoring function.  6.2.1. DFR\_RESET  Added comment about the target register, when DRR\_RESET with status/voltage clear bit called.  6.2.8. SET\_SEU\_STATUS  Added new control function.  6.2.9. SET\_SEU\_INJERR  Added new control function.  6.2.10. SET\_SEU\_RSTERR  Added new control function.  7.3 SEU Mitigation  Added new subsection for FPGA SEU Mitigation f unction. |
| 2.0.0 | 2019/01/29,  2019/04/02 | H.Sano,  K.Takezawa | 6.1.17. GET\_CONSUMPTION  Added new monitoring function.  6.1.18. GET\_METAFRAME\_ERROR  Added new monitoring function.  5.1 – 6.2 All items,  Added “errors” filed  1.Overview  Added block diagram and description.  5.5.1. Command :DRXPD INIT  Added the description when it called during overheating state.  5.5.2. Command :DRXPD\_SET\_RCVEVT  Added the description when it called during overheating state.  6.1.5. GET\_DFR\_VOLTAGE\_STATUS  Added the description for production DRXP board.  6.1.18. GET\_OVH\_THRESHOLD  Added new function. |

(Continued to the next page.)

(Continued from the previous page.)

|  |  |  |  |
| --- | --- | --- | --- |
| 2.0.0 | 2019/01/29,  2019/04/02 | H.Sano,  K.Takezawa | 6.1.19. GET\_OVH\_FLG  6.2.11. SET\_OVH\_THRESHOLD  Added new function.  7.4. The overheating protection  Added new section. |
| 2.0.1 | 2019/05/30 | K.Takezawa | 5.5.18 Command :DRXPD\_SET\_SEUEVT  Fixed description of “Function”. It didn’t take into account of production DRXP boards. And added description about the handling of evnetfd.  Changed status of EVENT LIST. In the case of driver version 2.0.2 or later, correctable error and uncorrectable error can be detected.  5.5.19 Command :DRXPD\_CLR\_SEUEVT  Fixed description of “Function”. It didn’t take into account of production DRXP boards.  6.1.13 GET\_FPGA\_TEMP  7.4. The overheating protection  Fixed the orthographic variants about the FPGA junction temperature.  6.1.1 GET\_ALARM\_STATUS\_REG  6.1.2 GET\_POWER\_ALARM\_REG  Fixed description in the remarks. It didn’t take into account of production DRXP boards.  Added the description about source of data.  6.1.3.GET\_SIGNAL\_AVG\_nW  Fixed description in the remarks. It didn’t take into account of production DRXP boards.  6.1.10 GET\_DFR\_METAFRAME\_DELAY  Fixed description.  6.1.17 GET\_CONSUMPTION  Fixed sample code.  Rephrased the content of remarks.  Added description about the behavior that this function is used with prototype DRXP boards. |

(Continued to the next page.)

(Continued from the previous page.)

|  |  |  |  |
| --- | --- | --- | --- |
| 2.0.1 | 2019/05/30 | K.Takezawa | 6.1.18 GET\_OVH\_THRESHOLD  Rephrased the content of remarks.  Fixed description about the behavior of resetting threshold value.  Added description about the behavior that this function is used with prototype DRXP boards.  6.1.19 GET\_OVH\_FLG  Rephrased the content of remarks.  Added description about the behavior that this function is used with prototype DRXP boards.  6.2.1 DFR\_RESET  Added the explanation that it called with bit0 in byte0 and bit7 in byte0 are equal to zero.  6.2.2 RESET\_DFR\_CHKSUM  Fixed description.  6.2.11 SET\_OVH\_THRESHOLD  Rephrased the content of remarks.  Added description about the behavior that this function is used with prototype DRXP boards.  7.3 SEU Mitigation  Added description about production DRXP board.  7.3.1.1.6 Diagnostic Scan  Renew description. GET\_SEU\_DIAERRCNT\_ID is implemented to driver version 2.0.2 or later. |

(Continued to the next page.)

(Continued from the previous page.)

|  |  |  |  |
| --- | --- | --- | --- |
| 2.0.2 | 2019/06/18 | K.Takezawa | Document header  Fixed document header.  2.1 Structure of Receive Ring Buffer Mode  Fixed the description.  (before) “The last 2800592-Bytes data in **each** block shall be thrown away because it is meaningless.”  (after) “The last 2800592-Bytes data in **last** block shall be thrown away because it is meaningless.”  Fixed the figure of buffer structure. The description of the Group2 in this figure was wrong.  (before) 4MBx35block  (after) 4M**i**Bx35block  Fixed the description.  (before)The group 0 to **16** is repeated time sequentially and barrel-rolled.  (after)The group 0 to **15** is repeated time sequentially and barrel-rolled.  3.2 Information Frame  Fixed the description about the software timestamp field (UTC(struc timespec)) in the information frame.  Fixed the description about the frame status field in the information frame. The definition of bit0 in the frame status was wrong.  Fixed the description about the abstract.  4.1 Note on the driver use  Deleted the below sentence.  “While this driver is working, interruption by another driver access is inhibited.”  Added the description about the architecture this driver supports.  4.2 Installing Driver  Deleted the instruction. Added the reference instated of it.  4.2 Uninstalling Driver  Deleted the instruction. Added the reference instated of it. |

(Continued to the next page.)

(Continued from the previous page.)

|  |  |  |  |
| --- | --- | --- | --- |
| 2.0.2 | 2019/06/18 | K.Takezawa | 4.4 Example of Reception  Added new path related to overheat protection to the flow chart.  4.6 How to see the device driver logs  　Added　new  5.1 open  Added new errno (ECANCELED) to represent the hardware error.  Added the description how to identify the interface number.  5.5.3 DRXPD\_RCVEVTSTART  Fixed the description of errno(EBUSY).  Fixed the description of errno(EFAULT).  5.5.4 DRXPD\_RCVEVTSTOP  Fixed the description of errno(EFAULT).  5.5.5 DRXPD\_RCVSTART  Fixed the description of errno(EFAULT).  5.5.6 DRXPD\_RCVSTOP  Fixed the description of errno(EFAULT).  5.5.7 DRXPD\_GET\_RCVSTS  Fixed the description of errno(EFAULT).  5.5.8 DRXPD\_CLR\_RCV  Fixed the description of EINVAL.  (before) … argument is out of range (0- **64**).  (after) … argument is out of range (0- **63**).  5.5.9 DRXPD\_SNDSTART  Fixed the description of errno(EFAULT).  5.5.10 DRXPD\_SNDSTOP  Fixed the description of errno(EFAULT).  5.5.11 DRXPD\_GET\_SNDSTS  Fixed the description of errno(EFAULT).  5.5.15 DRXPD\_GET\_I2C\_DATA(for Debug)  Fixed the description of errno(EFAULT). |

(Continued to the next page.)

(Continued from the previous page.)

|  |  |  |  |
| --- | --- | --- | --- |
| 2.0.2 | 2019/06/18 | K.Takezawa | 5.5.21 DRXPD\_SEL\_EQ  5.5.22 DRXPD\_GET\_EQ  Added new.  6.1.1 GET\_ALARM\_STATUS\_REG  Fixed the description about the monitoring timing.  6.1.2 GET\_POWER\_ALARM\_REG  Delete the description below  “This status should be cleared after power-up by using DFR\_RESET.”  Added the desction of the method to reset the alarm bits.  6.1.4 GET\_DFR\_STATUS  Changed bit status definition.  Bit2, 1 and 0 in byte1:  (before) Data input clock PLL alarm (CHx-Bitx)  (after) **Internal** input clock PLL alarm (CHx-Bitx)  6.1.5 GET\_DFR\_VOLTAGE\_STATUS  Fixed the description about the usage of this register.  Added the description about the usage of this register with driver version 2.0.2 or older.  6.1.6 GET\_DFR\_CHKSUM\_COUNTER  Fixed the description about the monitoring timing in the remarks.  6.1.7 GET\_DFR\_SYNC\_ERR\_CNT  Fixed the description about the monitoring timing in the remarks.  6.1.8 GET\_DFR\_SYNC\_STATUS  Added the description for explaining the function behavior.  6.1.11 GET\_DFR\_SN  Added the new description of Data. (Added bit0 in the byte4 for indicating the index of interface.) |

(Continued to the next page.)

(Continued from the previous page.)

|  |  |  |  |
| --- | --- | --- | --- |
| 2.0.2 | 2019/06/18 | K.Takezawa | 6.1.13 GET\_FPGA\_TEMP  Added data endian information.  6.1.17 GET\_CONSUMPTION  Added data endian information.  6.1.18 GET\_OVH\_THRESHOLD  Added data endian information.  6.1.19 GET\_DFR\_FORCETX  Added new.  6.2.5 INSERT\_SYNCPTN\_ERR  Added the description of what data pattern is written to the sync pattern field of ALMA frame.  6.2.11 SET\_OVH\_THRESHOLD  Fixed the valid data range for setting the user –defined threshold.  Changed the description of initial threshold value.  6.2.12 SET\_DFR\_FORCETX\_ID  Added new  7.4.1 Log  Changed the description of the timing that device driver dumps the overheat logs to syslog.  Fixed the log examples.  Added the description of the maximum number of log entry.  Added the description about the condition of hardeare protection is enabled.  7.4.2 Overheat flag  　Added new |

(Continued to the next page.)

(Continued from the previous page.)

|  |  |  |  |
| --- | --- | --- | --- |
| 2.0.3 | 2019/06/28 | K.Takezawa | The index of figure and table  1. Overview  Moved a summery to the head of the Overview chapter.  Fixed description of the JTAG block in the figure1-1.  (Before) JTAG connector **(for debug)**  (After) JTAG connector  1.1 Scope of this document  Updated the document scope.  (from 1.0.0 to 2.0.0 -> 2.0.3 )  1.1.1 Correspondence table between driver version and FPGA (prototype)  Added information about the driver version 1.0.5, 1.0.6, 1.0.7, 2.0.1, 2.0.2 and 2.0.3.  1.1.2 Correspondence table between driver version and FPGA (production)  Added information about the driver version 2.0.1, 2.0.2 and 2.0.3.  2.2 Structure of transmission Buffer  Fixed the figure 2-2. (MB -> M**i**B).  Added the description about the ring buffer allocation when DRXP board is installed to NUMA architecture server.  3.1 Normal frame  Added the description about the assuming condition of production DRXP board. Added following sentence. “DRXP board assumes that host machine is little endian machine.”  Added the description about the old normal frame format and the new normal frame format. The old one is named DRXP F1 format, the new one is named DRXP F2 format.  3.1.1 The kinds of normal frame  Added new. |

(Continued to the next page.)

(Continued from the previous page.)

|  |  |  |  |
| --- | --- | --- | --- |
| 2.0.3 | 2019/06/28 | K.Takezawa | 3.2 Information frame  Added the description about the frame synchronization bit in the Frame status bit.  Added the description about the Hardware 48ms tick count. Added the value range and the condition of resetting this counter.  Fixed the description about the UTC field in the Frame status. (Added the description about the nano second and value range of tv\_nsec. )  4.4 Example of reception  4.5 Example of transmission  Fixed the flow chart graph.  5.3 mmap  Fixed the description of return value. The return value is changed from -1 to MAP\_FAILED when it encounters the error.  5.5.1 Command :DRXPD\_INIT  Fixed the description of Function field. Added the instruction for production DRXP board.  5.5.2 Command: DRXPD\_SET\_RCVEVT  Added the description about the return value of eventfd. (To be able to determine which threshold (user-defined or hardware ) causes overheating state.)  5.5.16 Command :DRXPD\_SET\_I2C\_CMD (for Debug)  Fixed the description of caution in the remarks. Added a specific model name of SFP+ module.  5.5.19 Command :DRXPD FLASH\_OVHLOG  Fixed subsection number.  (before) 5.5.**19**  (after) 5.5.**20**  6.1.1 GET\_ALARM\_STATUS\_REG  Deleted the masking rule of Data  6.1.6 GET\_DFR\_CHKSUM\_COUNTER  Fixed the description about the monitoring timing in the remarks. |

(Continued to the next page.)

(Continued from the previous page.)

|  |  |  |  |
| --- | --- | --- | --- |
| 2.0.3 | 2019/06/28 | K.Takezawa | 6.1.7 GET\_DFR\_SYNC\_ERR\_CNT  Fixed the description about the monitoring timing in the remarks.  6.1.8 GET\_DFR\_SYNC\_STATUS  Fixed the description of bit6 in the Byte0 in the Data field.  (before) There is no description about the bit6.  (after) bit**6**-1: Not used(undefined)  Fixed the description about sync lock bit in the Description field.  6.1.9 GET\_DFR\_TEST\_DATA  Added the errors field.  6.1.11 GET\_DFR\_SN  6.1.12 GET\_DFR\_VER  Added the description about the behavior when this function is used both production DRXP board IFs.  6.1.13 GET\_FPGA\_TEMP  6.1.14 GET\_SEU\_STATUS  6.1.15 GET\_SEU\_ERRCNT  6.1.16 GET\_SEU\_ELAPSED  Added the description about the behavior when this function is used both production DRXP board IFs.  6.1.17 GET\_CONSUMPTION  6.1.18 GET\_OVH\_THRESHOLD  Added the description about the behavior when this function is used both production DRXP board IFs.  6.1.19 GET\_OVH\_FLG  Added the description about the behavior when this function is used both production DRXP board IFs.  Added the description about the bit0 and bit1 in Byte0 of Data field. (To be able to determine which threshold (user-defined or hardware ) causes overheating state.)  6.1.20 GET\_DFR\_FORCETX  Fixed the description of Byte4 in the Data field. |

(Continued to the next page.)

(Continued from the previous page.)

|  |  |  |  |
| --- | --- | --- | --- |
| 2.0.3 | 2019/06/28 | K.Takezawa | 6.2.1 DFR\_RESET  Added the note of condition for resetting DFR\_STATUS register and DFR\_VOLTAGE\_STATUS.  Added the description about the behavior when this function is used both production DRXP board IFs.  Added the instruction when user calls DRR\_RESET with master reset flag during transmission/reception.  6.2.9 SET\_SEU\_STATUS  6.2.9 SET\_SEU\_INJERR  6.2.10 SET\_SEU\_RSTERR  Added the description about the behavior when this function is used both production DRXP board IFs.  6.2.11 SET\_OVH\_THRESHOLD  Added the description of a behavior when this function is used both production DRXP board IFs.  Added the description about the value range of Byte0-3 in the Data field.  Added data endian information.  6.2.12 SET\_DFR\_FIRCETX\_ID  Fixed this subsection title.  (before) SET\_DFR\_FIRCETX**\_ID**  (after) SET\_DFR\_FIRCETX  Deleted the whilte space from ItemID field.  (before) SET\_OVH\_THRESHOLD\_ ID  (after) SET\_OVH\_THRESHOLD\_ID  Added the description about the call timing.  7.2 Acronym and meanings of abbreviation  Added new acronyms. (DFR and SEU).  7.4 The overheating protection  Changed the description about the return value of eventfd when overheat event is occurred.  7.4.2 Oveahea flag  Added new. |

(Continued to the next page.)

(Continued from the previous page.)

|  |  |  |  |
| --- | --- | --- | --- |
| 2.0.4 | 2019/07/12 | H.Sano,  K.Takezawa | 1.1 Introduction  Added about how to determine the number at the end of the device files.  1.3.2 Correspondence table between driver version and FPGA (production)  Fixed the description of footnote (\*2).  (before) … has no **timestamp** by …  (after) … has no **version** by …  3.2 Information frame  Added the description to the explanation of Frame status.  4.1 Note on the driver use  Added header file to include for errno.  5.1 open  Added error code (EBUSY).  5.4 munmap  Added labels “Remarks”.  5.5 ioctl  Added labels “Synchronization”.  5.5.4 DRXPD\_RCVEVTSTOP  Added error code (EAGAIN).  5.5.5 DRXPD\_RCVSTART  Added more describe about EAGAIN.  5.5.20. Command :DRXPD FLASH\_OVHLOG  Added more describe about this command.  6.5 Monitoring and Control Items  Added labels “Synchronization”.  6.1. Monitoring Items  In the section outline, stated that each item is cleared at the time of probe and added labels item of “Data”.  6.1.3 GET\_SIGNAL\_AVG\_nW  Fixed the description of the Description field.  (before) …Optical Power. This returns …  (after) …Optical Power **from SFP+ module** … |

(Continued to the next page.)

(Continued from the previous page.)

|  |  |  |  |
| --- | --- | --- | --- |
| 2.0.4 | 2019/07/12 | H.Sano,  K.Takezawa | 6.1.20 GET\_DFR\_FORCETX  Deleted white space form ItemID.  Fixed the description about Byte4 in the data field.  7.4.1 Log  Added “no log entry” description.  Added the log output example when there is no log message in the entry. |
| 2.0.5 | 2019/08/16  -  2019/09/19 | H.Sano,  K.Takezawa | 1.1 Introduction  Fixed the terminology error. (interface -> IF).  (before) …number of X is imparted **interface1**.  (after) …number of X is imparted **IF1**.  (before)…device file, **interface**, and…  (after)…device file, **IF**, and…  (before)…explanation of data flow for one **interface**  (after)…explanation of data flow for one IF  Changed the name of ring buffer on the server in the Figure 1-1, 1-2. (Data Buffer -> Host Data Buffer ).  1.2 The kinds of DRXP board  Fixed the terminology error in the Figure 1-3. (interface -> IF).  3 Format of Frame in Data Buffer  Changed the title. (Data Buffer -> Host Data Buffer).  3.1 Normal frame(DRXP F2 format)  Added production DRXP board bit assign to the table  Fixed the index number of each bit ALMA frame.  (M17, M18, M19 ->M16, M17, M18 )  (L17, L18, L19 ->L16, L17, L18 )  3.2 Information frame  Added the Parity field to the information frame.  4.6.1 Log format  Fixed the terminology error. (interface -> IF ).  5.1 open  Fixed the terminology error. (interface->IF).  Added the errno. ( EBUSY and ENOMEM). |

(Continued to the next page.)

(Continued from the previous page.)

|  |  |  |  |
| --- | --- | --- | --- |
| 2.0.5 | 2019/08/16  -  2019/09/19 | H.Sano,  K.Takezawa | 5.2 close  Changed the behavior when an error is occurred.  5.3 mmap  Changed the data buffer name. ( the data buffer -> the host data buffer ).  Deleted error code (EIO).  Added error code(ENOMEM).  5.5.1 Command : DRXPD\_INIT  Fixed the terminology error. (interface -> IF ).  Changed the behaivior of the secound argument (\*p\_result) in Parameter field. And also changed the description of return field.  Added new the error codes for errno.  5.5.2 DRXPD\_SET\_RCVEVT  Added error code(EBUSY).This is insufficient contents from driver version 2.0.0.  Fixed the description about the error codes (EBADFD and EINVAL).  5.5.3 DRXPD\_RCVEVTSTART  Added error code(EAGAIN).This is insufficient contents from driver version 2.0.0.  Changed the behaivior of the secound argument (\*p\_result) in Parameter field. And also changed the description of return field.  5.5.4 DRXPD\_RCVEVTSTOP  Changed the behaivior of the secound argument (\*p\_result) in Parameter field. And also changed the description of return field.  5.5.5 DRXPD\_RCVSTART  Changed describe about DRXPD\_NG\_INIT.  Changed the behaivior of the secound argument (\*p\_result) in Parameter field. And also changed the description of return field. |

(Continued to the next page.)

(Continued from the previous page.)

|  |  |  |  |
| --- | --- | --- | --- |
| 2.0.5 | 2019/08/16  -  2019/09/19 | H.Sano,  K.Takezawa | 5.5.6 DRXPD\_RCVSTOP  Changed the behaivior of the secound argument (\*p\_result) in Parameter field. And also changed the description of return field.  5.5.8 DRXPD\_CLR\_RCV  Deleted the remarks field.  5.5.9 DRXPD\_SNDSTART  Changed the behaivior of the secound argument (\*p\_result) in Parameter field. And also changed the description of return field.  5.5.10 DRXPD\_SNDSTOP  Changed the behaivior of the secound argument (\*p\_result) in Parameter field. And also changed the description of return field.  5.5.12 DRXPD\_SET\_SNDSIZ  Added the error code ( EINVAL, EAGAIN, and EBUSY).  Deleted the error code (EIO).  5.5.13 DRXPD\_GET\_MONSTS  Fixed the description of errors item.  5.5.14 DRXPD\_SET\_CONTROL  Fixed the description of errors item.  5.5.16 DRXPD\_SET\_I2C\_CMD(for Debug)  Fixed the description of EFAULT.  5.5.18 DRXPD\_SET\_SEUEVT  Added the description for explaining what SEU means.  Added cross-reference to the section 7.3 SEU Mitigation.  5.5.19 DRXPD\_CLR\_SEUEVT  Added cross-reference to the section 7.3 SEU Mitigation. |

(Continued to the next page.)

(Continued from the previous page.)

|  |  |  |  |
| --- | --- | --- | --- |
| 2.0.5 | 2019/08/16  -  2019/09/19 | H.Sano,  K.Takezawa | 2.1 Structure of Receive Ring Buffer  5.3 mmap  5.5.17 DRXPD\_RCVGROUPS  Changed command name. (Maintains compatibility with previous implementations for prototype DRXP.)  (before) DRXPD\_RCVGROUPS  (after) DRXPD\_**SET\_**RCVGROUPS  Whole document.  Changed command name. (Maintains compatibility with previous implementations for prototype DRXP.)  (before) DFR\_RESET/DFR\_RST\_xxx  (after) RESET\_DFR/RST\_DFR \_xxx  5.5.18 DRXPD\_SET\_SEUEVT  Fixed the description about the error codes (EBADFD and EINVAL).  6.1 Monitoring Items  Added the description about monitoring item.  6.1.3 GET\_SIGNAL\_AVG\_nW  Fixed the description of the Latched field. Added following description. “These are latched value.”.  6.1.11 GET\_DFR\_SN  Fixed the terminology error. (interface -> IF ).  6.1.14 GET\_SEU\_STATUS  6.1.15 GET\_SEU\_ERRCNT  6.1.16 GET\_SEU\_ELAPSED  Added cross-reference to the section 7.3 SEU Mitigation.  6.1.22 GET\_PARITY\_RANGE  Added new.  6.2.1 RESET\_DFR  Fixed the terminology error. (interface->IF).  6.2.4 DFR\_TEST\_DATA  Changed command name. (Maintains compatibility with previous implementations for prototype DRXP.)  6.2.6 SET\_CHKSUM\_ERR  Deleted the error code(EINVAL). |

(Continued to the next page.)

(Continued from the previous page.)

|  |  |  |  |
| --- | --- | --- | --- |
| 2.0.5 | 2019/08/16  -  2019/09/19 | H.Sano,  K.Takezawa | 6.2.7 SET\_SQCNT\_ERR  Deleted the error code(EINVAL).  6.2.8 SET\_SEU\_STATUS  6.2.9 SET\_SEU\_INJERR  6.2.10 SET\_SEU\_RSTERR  Added cross-reference to the section 7.3 SEU Mitigation.  6.2.13 SET\_PARITY\_RANGE  Added new.  5.5.1 DRXPD INIT  Added the description about the behavior when this command is used.  4.1 Note on the driver use  Fixed grammer of sentence about errno.  3.2 Information frame  Added more describe about Frame status.  7.4.1 Log  Changed overheat log message.  6.1.17 GET\_SEU\_DIAERRCNT  Added command (subsequent monitoring items have different item numbers)  5.5.20. FLASH\_OVHLOG  Fixed error code EINVAL to ENOSYS.  6.2.8. SET\_SEU\_STATUS  Added error code(EIO and ECANCELED). This is insufficient contents from driver version 2.0.0.  5.5.1. DRXPD\_INIT  5.5.3 RCVEVTSTART  5.5.4 RCVEVTSTOP  5.5.5 RCVSTART  5.5.6 RCVSTOP  5.5.9 SNDSTART  5.5.10 SNDSTOP  5.5.21 SEL\_EQ  6.1.4 GET\_DFR\_STATUS  Fixed synchronization asynchronous to synchronous |

(Continued to the next page.)

(Continued from the previous page.)

|  |  |  |  |
| --- | --- | --- | --- |
| 2.0.5 | 2019/08/16  -  2019/09/19 | H.Sano,  K.Takezawa | 6.1.5 GET\_DFR\_VOLTAGE\_STATUS  6.1.19 GET\_OVH\_THRESHOLD  6.1.20 GET\_OVH\_FLG  6.2.2 RESET\_DFR\_CHKSUM  6.2.3 RESET\_DFR\_SYNC  Fixed synchronization asynchronous to synchronous  5.5.22 GET\_EQ  Corrected misspelling in Errors.(EAGAIN)  7.4 The overheating protection  Fixed the terminology error. (interface -> IF ).  7.5 The parity for checking FPGA data integrity  Added new. |

(Continued to the next page.)

(Continued from the previous page.)

|  |  |  |  |
| --- | --- | --- | --- |
| 2.0.6 | 2019/10/06 | K.Takezawa | Entire this document  Corrected the description errors, grammatical errors.  1.1 Introduction  Fixed the terminology mistake.  (before) …of X is imparted interface1. If…  (after) …of X is imparted IF1. If…  1.4 Notation for command instructions  Added new.  4.1 The note on the driver use  Added the description about the rule about creating device file.  4.6 How to see the device driver logs  Deleted the description about a command authority level. Instead of this, we added new section 1.4.  6.1.5 GET\_DFR\_VOLTAGE\_STATUS  Added the criteria of voltage alarm.  6.2.1 RESET\_DFR  Added the description about the IF reset. What logic is reset by RESET\_DFR with master\_reset bit.  Added the common\_logic\_reset bit.  6.2.8 SET\_SEU\_STATUS  Added description about the diagnostic scan mode to the remarks field.  6.2.9 SET\_SEU\_INJERR  Added description about the error injection about production DRXP board. And fixed the address for injection test.  6.2.8 SET\_SEU\_STATUS  Added the description to “Synchronization” field.  6.2.9 SET\_SEU\_INJERR  Added the description to “Synchronization” field.  6.2.13 SET\_PARITY\_RANGE  Added new errno(EBUSY).  Added the description to “Synchronization” field.  7.3.1.7 Diagnostic Scan  Fixed the description of note. |

(Continued to the next page.)

(Continued from the previous page.)

|  |  |  |  |
| --- | --- | --- | --- |
| 2.0.6 | 2019/10/06 | K.Takezawa | 7.5 The parity for checking FPGA data integrity  Added hyperlink to subsection SET\_PARITY\_RANGE.  7.4 The overheating protection  Added a state transition diagram. |
| 2.0.7 | 2020/04/03 | K.Takezawa | 2.1 Structure of Receive Ring Buffer  Fixed the section title  (before) Structure of Receive Ring Buffer  (after) Structure of **Host** Receive Ring Buffer  Fixed the name of ring buffer name.  (before) Note: …this ring buffer is allocated  (after) Note: …this **host** ring buffer is allocated  (before) Figure2-1: …receive ring buffer  (after) Figure2-1: …**host** receive ring buffer  2.2 Structure of Transmission Buffer  Fixed the section Title.  (before) Structure of Transmission Buffer  (after) Structure of **Host** Transmission Buffer  Fixed the name of ring buffer name.  (before) Note: …this ring buffer is allocated  (after) Note: …this **host** ring buffer is allocated  (before) Figure2-2: …transmission buffer  (after) Figure2-2: …**host** transmission buffer  3.2 Information frame  Added reference information of ALMA frame de-formatter to the note (\*3), (\*4) and (\*5).  Added incomplete bit.  4.4 Example of reception  Fixed the name of ring buffer in the diagram.  (before) Call "mmap" to get the address of the receive ring buffer.  (after) Call "mmap" to get the address of the **host** receive ring buffer.  (before) Call "munmap" to free the address of the receiving ring buffer |

(Continued to the next page.)

(Continued from the previous page.)

|  |  |  |  |
| --- | --- | --- | --- |
| 2.0.7 | 2020/04/03 | K.Takezawa | 4.4 Example of reception  (after) Call "munmap" to free the address of the **host** receiving ring buffer  5.1 open  Fixed the description about the restrictions on sending and receiving functions. (Added the description about SET/GET\_DFR\_FORCETX.).  5.5.2 DRXPD\_SET\_RCVEVT  Fixed the name of ring buffer name in the [description].  (before) group basis in the Receive Ring Buffer …  (after) group basis in the **Host** Receive Ring Buffer …  Added the explanation about the eventfd resource.  5.5.3 DRXPD\_RCVEVTSTART  Fixed the terminology.  (DFR\_RESET -> RESET\_DFR in the timeout field)  5.5.4 DRXPD\_RCVEVTSTOP  Fixed the terminology.  (DFR\_RESET -> RESET\_DFR in the timeout field)  5.5.5 DRXPD\_RCVSTART  Fixed the terminology.  (DFR\_RESET -> RESET\_DFR in the timeout field)  5.5.6 DRXPD\_RCVSTOP  Fixed the terminology.  (DFR\_RESET -> RESET\_DFR in the timeout field)  5.5.7 DRXPD\_GET\_RCVSTS  Fixed the name of ring buffer name in the [parameter].  (before) the receive ring buffer state(group…  (after) the **host** receive ring buffer state(group… |

(Continued to the next page.)

(Continued from the previous page.)

|  |  |  |  |
| --- | --- | --- | --- |
| 2.0.7 | 2020/04/03 | K.Takezawa | 5.5.8 DRXPD\_CLR\_RCV  Fixed the name of ring buffer name in the [function], [paramter].  (before) This command sets the state of the receive ring buffer to DRXPD\_IDLE.  (after) This command sets the state of the **host** receive ring buffer to DRXPD\_IDLE.  (before) specifies a group of the receive r ing buffer. The specified number of the ring buffer is set to the IDLE state.  (after) specifies a group of the **host** receive  ring buffer. The specified number of the **host** ring buffer is set to the IDLE state.  5.5.9 DRXPD\_SNDSTART  Fixed the terminology.  (DFR\_RESET -> RESET\_DFR in the timeout field)  5.5.10 DRXPD\_SNDSTOP  Fixed the terminology.  (DFR\_RESET -> RESET\_DFR in the timeout field)  5.5.18 DRXPD\_SET\_SEUEVT  Added the explanation about the eventfd resource.  Fixed typo. (“Signal Event …” -> “Single Event …” in the Description field.  5.5.20 DRXPD\_FLASH\_OVHLOG  Fixed the terminology.  (DFR\_RESET -> RESET\_DFR in the timeout field)  5.5.21 DRXPD\_SEL\_EQ  Fixed the terminology.  (DFR\_RESET -> RESET\_DFR in the timeout field)  5.5.21 DRXPD\_SEL\_EQ  Fixed description of remarks for production DRXP.  5.5.22 DRXPD\_GET\_EQ  Fixed description of remarks for production DRXP.  5.5.23 DRXPD\_GET\_OVRFLW  Added new. |

(Continued to the next page.)

(Continued from the previous page.)

|  |  |  |  |
| --- | --- | --- | --- |
| 2.0.7 | 2020/04/03 | K.Takezawa | 6.1 Monitoring Items  Added note for explaining the mean of latched value.  Added item’s categories.  6.1.2 GET\_POWER\_ALARM\_REG  Added explanation about bit0 in Byte1of date.  6.1.3 GET\_SIGNAL\_AVG\_nW  Delete below description from Latch field.  “This value is latched value.”  6.1.9 GET\_DFR\_TEST\_DATA  Fixed typo. (DFR\_TEST\_DATA -> SET\_DFR\_TEST\_DATA) and added explanation about Byte0 in the value field.  6.1.14 GET\_SEU\_STATUS  Fixed the terminology.  (DFR\_RESET -> RESET\_DFR in the reset field)  7.4 The overheating protection  Add detail description of the implementation of shutdown threshold.  Add detail description of the log message.  7.6 Device file attribution  Added new.  7.7 State transition table  Added new. |
| 2.0.8 | 2020/12/01 | K.Takezawa | 1.2.1. Comparison table between prototype DRXP and production DRXP (hardware)  Fix table content. (Table 1-1)  1.2.2. Comparison table between prototype DRXP and production DRXP (software)  Fixed table content. (Table 1-2)  The prototype DRXP can use some features in devce driver version 2.0.8 or later.  1.3.1. Correspondence table between driver version and FPGA (prototype)  Added new FPGA information. |

(Continued to the next page.)

(Continued from the previous page.)

|  |  |  |  |
| --- | --- | --- | --- |
| 2.0.8 | 2020/12/01 | K.Takezawa | 1.3.2. Correspondence table between driver version and FPGA (production)  Added new FPGA information.  3.1.1. The kinds of normal frame (DRXP F1 format)  　Changed description.  3.2 Information frame  Added description to the note (\*7) for explaing the checksum error bit and sequence error bit behavior when it detects meta-frame bit earlier than expected.  5.5.23. Command :DRXPD\_GET\_OVRFLW  Fixed typo in the format field.  (DRXPD\_GET\_EQ -> DRXPD\_GET\_OVRFLW,  I\_eq\_type -> i\_flag)  Deleted version information from EFAULT.des  Added description to the function field.  6.1.4. GET\_DFR\_STATUS  Added remarks. DDR clock PLL1/2 alarm sometimes meaning less.  6.1.19. GET\_OVH\_THRESHOLD  Fixed the description of Remarks related to the prototype DRXP.  Fixed the description of Errors related to the prototype DRXP.  6.1.20. GET\_OVH\_FLG  Fixed the description of Remarks related to the prototype DRXP.  Fixed the description of Errors related to the prototype DRXP.  6.1.22. GET\_PARITY\_RANGE  Fixed the description of ENOSYS.  6.2.1. RESET\_DFR  　Fixed typo in the note (\*1). (DFR\_RST\_ID\_ALL ->RST\_DFR\_ID\_ALL).  Added note for bug behavior when using the device driver in the 2.0.7 or older and production DRXP. |

(Continued to the next page.)

(Continued from the previous page.)

|  |  |  |  |
| --- | --- | --- | --- |
| 2.0.8 | 2020/12/01 | K.Takezawa | 6.2.1. RESET\_DFR  Fixed the Remarks for the production DRXP boards. It moved DFR\_STATUS from the shared register category to the independent register category.  Changed timeout value. (100ms -> 800ms).  6.2.8. SET\_SEU\_STATUS  Fixed the description of remarks related to the prototype DRXP.  6.2.10. SET\_SEU\_RSTERR  Fixed typo.  6.2.11. SET\_OVH\_THRESHOLD  Fixed the description of Remarks related to the prototype DRXP.  Fixed the description of Errors related to the prototype DRXP.  6.2.12. SET\_DFR\_FORCETX  Fixed the description of ENOSYS and remarks.  Fixed the typo in the Data field. (Byte3-7 ->Byte5-7)  6.2.13. SET\_PARITY\_RANGE  Fixed the description of ENOSYS and remarks.  6.2.1. RESET\_DFR  Fixed typo in Data field.  ((before)SEM logic -> SEU logic)  7.3.1.7. Diagnostic Scan  Fixed note for prototype DRXP.  7.3.2.3. Dignostic scan  Fixed note related to the driver version information.  7.7.1. State table (reception mode)  7.7.2. State table (transmission mode)  Added information of DRXPD\_GET\_OVRFLW |

(Continued to the next page.)

(Continued from the previous page.)

|  |  |  |  |
| --- | --- | --- | --- |
| 2.0.9 | 2021/02/01 | K.Takezawa | 3.1.1 The kinds of normal frame (DRXP F1 format)  Fixed the typo about the version information of prototype DRXP.  3.2 Information frame  Added the explanation to the notes (\*2) and (\*7).  6.1.21 GET\_DFR\_FORCETX  6.2.12 SET\_DFR\_FORCETX  Added new options.  - DFRFORCETX\_PRBS31  - DFRFORCETX\_PRBS63  - DFRFORCETX\_PRBS31SR  - DFRFORCETX\_PRBS31S  - DFRFORCETX\_PRBS63S  6.2.12.1 DFR\_FORCETX data format  Added new |

(Continued to the next page.)

**Table　of contents**

[1. Overview 39](#_Toc63179380)

[1.1. Introduction 39](#_Toc63179381)

[1.2. The kinds of DRXP board 42](#_Toc63179382)

[1.2.1. Comparison table between prototype DRXP and production DRXP (hardware) 44](#_Toc63179383)

[1.2.2. Comparison table between prototype DRXP and production DRXP (software) 45](#_Toc63179384)

[1.3. Scope of this document 47](#_Toc63179385)

[1.3.1. Correspondence table between driver version and FPGA (prototype) 47](#_Toc63179386)

[1.3.2. Correspondence table between driver version and FPGA (production) 48](#_Toc63179387)

[1.4. Notation for command instructions 49](#_Toc63179388)

[1.5. The notation for the level of sample 49](#_Toc63179389)

[2. Structure of Data Buffer 50](#_Toc63179390)

[2.1. Structure of Host Receive Ring Buffer 50](#_Toc63179391)

[2.2. Structure of Host Transmission Buffer 52](#_Toc63179392)

[3. Format of Frame in Host Data Buffer 53](#_Toc63179393)

[3.1. Normal frame (DRXP F2 format) 53](#_Toc63179394)

[3.1.1. The kinds of normal frame (DRXP F1 format) 54](#_Toc63179395)

[3.2. Information frame 55](#_Toc63179396)

[4. Usage 60](#_Toc63179397)

[4.1. The note on the driver use 60](#_Toc63179398)

[4.2. Installation of the Driver 60](#_Toc63179399)

[4.3. Uninstallation of the Driver 60](#_Toc63179401)

[4.4. Example of reception 61](#_Toc63179402)

[4.5. Example of transmission 64](#_Toc63179403)

[4.6. How to see the device driver logs 66](#_Toc63179404)

[4.6.1. Log format 67](#_Toc63179405)

[5. API detailed description 68](#_Toc63179406)

[5.1. open 68](#_Toc63179407)

[5.2. close 70](#_Toc63179408)

[5.3. mmap 71](#_Toc63179409)

[5.4. munmap 72](#_Toc63179410)

[5.5. ioctl 73](#_Toc63179411)

[5.5.1. Command :DRXPD INIT 75](#_Toc63179412)

[5.5.2. Command :DRXPD\_SET\_RCVEVT 77](#_Toc63179413)

[5.5.3. Command :DRXPD RCVEVTSTART 79](#_Toc63179414)

[5.5.4. Command :DRXPD RCVEVTSTOP 81](#_Toc63179415)

[5.5.5. Command :DRXPD RCVSTART 83](#_Toc63179416)

[5.5.6. Command :DRXPD RCVSTOP 85](#_Toc63179417)

[5.5.7. Command :DRXPD\_GET\_RCVSTS 87](#_Toc63179418)

[5.5.8. Command :DRXPD\_CLR\_RCV 89](#_Toc63179419)

[5.5.9. Command :DRXPD SNDSTART 90](#_Toc63179420)

[5.5.10. Command :DRXPD SNDSTOP 92](#_Toc63179421)

[5.5.11. Command :DRXPD\_GET\_SNDSTS 94](#_Toc63179422)

[5.5.12. Command :DRXPD\_SET\_SNDSIZ 95](#_Toc63179423)

[5.5.13. Command :DRXPD\_GET\_MONSTS 96](#_Toc63179424)

[5.5.14. Command :DRXPD\_SET\_CONTROL 97](#_Toc63179425)

[5.5.15. Command :DRXPD\_GET\_I2C\_DATA (for Debug) 98](#_Toc63179426)

[5.5.16. Command :DRXPD\_SET\_I2C\_CMD (for Debug) 100](#_Toc63179427)

[5.5.17. Command :DRXPD\_SET\_RCVGROUPS 102](#_Toc63179428)

[5.5.18. Command :DRXPD\_SET\_SEUEVT 103](#_Toc63179429)

[5.5.19. Command :DRXPD\_CLR\_SEUEVT 105](#_Toc63179430)

[5.5.20. Command :DRXPD\_FLASH\_OVHLOG 106](#_Toc63179431)

[5.5.21. Command :DRXPD\_SEL\_EQ 108](#_Toc63179432)

[5.5.22. Command :DRXPD\_GET\_EQ 110](#_Toc63179433)

[5.5.23. Command :DRXPD\_GET\_OVRFLW 112](#_Toc63179434)

[6. Monitoring and Control Items 114](#_Toc63179435)

[6.1. Monitoring Items 114](#_Toc63179436)

[6.1.1. GET\_ALARM\_STATUS\_REG 115](#_Toc63179437)

[6.1.2. GET\_POWER\_ALARM\_REG 117](#_Toc63179438)

[6.1.3. GET\_SIGNAL\_AVG\_nW 119](#_Toc63179439)

[6.1.4. GET\_DFR\_STATUS 121](#_Toc63179440)

[6.1.5. GET\_DFR\_VOLTAGE\_STATUS 123](#_Toc63179441)

[6.1.6. GET\_DFR\_CHKSUM\_COUNTER 125](#_Toc63179442)

[6.1.7. GET\_DFR\_SYNC\_ERR\_CNT 126](#_Toc63179443)

[6.1.8. GET\_DFR\_SYNC\_STATUS 127](#_Toc63179444)

[6.1.9. GET\_DFR\_TEST\_DATA 129](#_Toc63179445)

[6.1.10. GET\_DFR\_METAFRAME\_DELAY 130](#_Toc63179446)

[6.1.11. GET\_DFR\_SN 131](#_Toc63179447)

[6.1.12. GET\_DFR\_VER 132](#_Toc63179448)

[6.1.13. GET\_FPGA\_TEMP 133](#_Toc63179449)

[6.1.14. GET\_SEU\_STATUS 134](#_Toc63179450)

[6.1.15. GET\_SEU\_ERRCNT 137](#_Toc63179451)

[6.1.16. GET\_SEU\_ELAPSED 139](#_Toc63179452)

[6.1.17. GET\_SEU\_DIAERRCNT 141](#_Toc63179453)

[6.1.18. GET\_CONSUMPTION 142](#_Toc63179454)

[6.1.19. GET\_OVH\_THRESHOLD 144](#_Toc63179455)

[6.1.20. GET\_OVH\_FLG 145](#_Toc63179456)

[6.1.21. GET\_DFR\_FORCETX 147](#_Toc63179457)

[6.1.22. GET\_PARITY\_RANGE 149](#_Toc63179458)

[6.2. Control Items 150](#_Toc63179459)

[6.2.1. RESET\_DFR 150](#_Toc63179460)

[6.2.2. RESET\_DFR\_CHKSUM 154](#_Toc63179461)

[6.2.3. RESET\_DFR\_SYNC 155](#_Toc63179462)

[6.2.4. SET\_DFR\_TEST\_DATA 156](#_Toc63179463)

[6.2.5. INSERT\_SYNCPTN\_ERR 158](#_Toc63179464)

[6.2.6. SET\_CHKSUM\_ERR 158](#_Toc63179465)

[6.2.7. SET\_SQCNT\_ERR 159](#_Toc63179466)

[6.2.8. SET\_SEU\_STATUS 160](#_Toc63179467)

[6.2.9. SET\_SEU\_INJERR 162](#_Toc63179468)

[6.2.10. SET\_SEU\_RSTERR 164](#_Toc63179469)

[6.2.11. SET\_OVH\_THRESHOLD 165](#_Toc63179470)

[6.2.12. SET\_DFR\_FORCETX 166](#_Toc63179471)

[6.2.13. SET\_PARITY\_RANGE 175](#_Toc63179472)

[7. Appendix 176](#_Toc63179473)

[7.1. SFP module warning and alarm threshold 176](#_Toc63179474)

[7.2. Acronym and meanings of abbreviation 177](#_Toc63179475)

[7.3. SEU Mitigation 178](#_Toc63179476)

[7.3.1. Function detail 179](#_Toc63179477)

[7.3.2. Usage 182](#_Toc63179478)

[7.4. The overheating protection 185](#_Toc63179479)

[7.4.1. Log (version 2.0.6 or older) 188](#_Toc63179480)

[7.4.2. Log (version 2.0.7 or later) 189](#_Toc63179481)

[7.4.3. Confirm shutdown threshold (version 2.0.7 or later) 191](#_Toc63179482)

[7.4.4. Oveaheat flag 192](#_Toc63179483)

[7.5. The parity for checking FPGA data integrity 193](#_Toc63179484)

[7.6. Device file attribution 196](#_Toc63179485)

[7.6.1. Udev rule 197](#_Toc63179486)

[7.7. State transition table 199](#_Toc63179487)

[7.7.1. State table (reception mode) 199](#_Toc63179488)

[7.7.2. State table (transmission mode) 202](#_Toc63179489)

# Overview

## Introduction

The DRXP driver provides the interface to monitor and control the DRXP board. This document gives the hardware information related to software development and how to use driver.

The DRXP board is a PCI Express add-in card to acquire observed data from optical interfaces in ALMA frame format. The observed data is quantized with 3 bits(D/C/B bit) and transmitted separately. Therefore, you need three optical interfaces to receive/transmit. The DRXP board receives these three data and extracts, rearranges payload. After converting them, it transmits rearranged data to the Host Ring Buffer allocated on the host server memory (\*1) via PCI Express interface. Your application can read the observed data from this host ring buffer (Figure 1‑1: **Overview of the reception sequence** ). And your application can transmit data via this host ring buffer (Figure 1‑2: **Overview of the transmission sequence**). The DRXP driver also provides the access interface to the other board functions (e.g. the optical module status, the onboard power status, the 48ms time event and onboard FPGA status.). Your application monitors and controls these functions via the driver interface.

The DRXP driver supports Linux. The driver creates device file (e.g. /dev/drxpd0) as a monitoring/controlling interface. It is created for each set of three optical fibers (D/C/B bit). The driver creates two device files per one production DRXP board and these names are (/dev/drxpdX). X is assigned from zero in order and smaller number of X is imparted IF1. If two production DRXP boards are installed in server, the order in which the boards are recognized depends on the order in which the LIINUX OS recognizes them. You should call the driver API to know the correspondence between the device file, IF, and the production DRXP board. The user also uses udev mechanism for creating a symbolic link of user-specified device file automatically. User can specify the device file by using board serial number, IF number, board type and bus number. For the detail of udev, please refer **“7.6 Device file attribution”**.

At first, your application gets the file descriptor by open system call with DRXP device file as the argument. Then, you can control/monitor the DRXP board by calling API with this file descriptor.

(\*1) The production DRXP board has on-board ring buffer. In this document, we name this ring buffer Device Ring Buffer for distinguishing from the Host Ring Buffer allocated on the host server memory.

The following figure is an explanation of data flow for one IF.

**Your Linux Software**

**(User Module)**

**DATA**

**Control/Check**

**via API**

**(Copy via mmap’s**

**virtual address)**

eventfd

**Elecs’s Driver**

**(Kernel Module)**

Reception completion notification

**Host Data Buffer ※1.1**

**(Receive Ring Buffer Mode)**

4MiB×35block×16Groups[default]

= 2.1875GiB

(48ms×16Groups=768ms）

［You can extend to 32 or 64Groups.］

Control

/Check

**DATA**

**(DMA via PCIe)**

**System-Memory**

**(Kernel Space)**

**DRXP-Board**

※1.1

The data cannot be sent from the host data buffer to the DRXP-board in the reception mode

because the data transmission function shares the host data buffer with the data reception

function. But the DRXP-board can transmit ALMA frames with a fixed payload by using

SET\_DFR\_FORCETX\_ID even in the reception mode.

See **“6.1.21 GET\_DFR\_FORCETX”** and **“6.2.12 SET\_DFR\_FORCETX”** for more details.

Figure 1‑1: Overview of the reception sequence

**Your Linux Software**

**(User Module)**

**DATA**

**Control/Check**

**via API**

**(Copy via mmap’s**

**virtual address)**

**Elecs’s Driver**

**(Kernel Module)**

**Host Data Buffer ※1.1、1.2**

**(Transmission Buffer Mode)**

MaxSize: 4MiB×35blocks=140MiB

(48ms)

**DATA**

Control

/Check

**(DMA via PCIe)**

**System-Memory**

**(Kernel Space)**

**DRXP-Board**

※1.1

Transmission function send the contents of the

transmission buffer from first to last repeatedly.

Figure 1‑2: Overview of the transmission sequence

## The kinds of DRXP board

There are two kinds of DRXP board. One is a prototype DRXP board. The other is a production DRXP board. The prototype DRXP board has three optical interfaces (Figure 1‑4: **The Hardware block diagram of the prototype DRXP board**). It can receive/transmit one observed data quantized with 3 bits. The production DRXP board has six optical interfaces (Figure 1‑3: **The Hardware block diagram of the production DRXP board**). So it can receive/transmit two observed data. A device file is created per one set of three optical interfaces (D/C/B bit). Therefore, two device files are created for one production board. One device file is created for one prototype DRXP board. In the case that one prototype DRXP and one production DRXP are installed, 3 device files are created for controlling and monitoring each IF.

DDR for

IF2

DDR for

IF1

IF2

**RS485**

48ms In

　　FPGA

SFP＋Module

（Dbit）

**ALMA Frame(Tx)**

48ms Out

**ALMA Frame(Rx)**

**I2C**

SFP＋Module

（Cbit）

JTAG

Connector

**I2C**

**ALMA Frame(Rx)**

**ALMA Frame(Tx)**

**JTAG**

Powers on Board

+0.85V(for FPGA)

+1.8V(for FPGA)

+3.3V(for FPGA and SFP+)

+0.9V\_MGTAVCC(for FPGA)

+1.2V\_MGTAVTT(for FPGA)

+1.8V\_MGTVCCAUX

(for FPGA)

+1.35VDDR(for DDR)

SFP＋Module

（Bbit）

**I2C**

**ALMA Frame(Rx)**

**ALMA Frame(Tx)**

Same as

IF2

signals

IF1

(Same as IF2)

**12V**

Power

Connecter

12V

(For Debug)

**PCIE 3.0**

**(Gen3)**

**x 16 lane**

**＜To PCIE　Slot＞**

Figure 1‑3: The Hardware block diagram of the production DRXP board

**ALMA Frame(Tx)**

　　DDR

JTAG

Connector

SFP＋Module

（Dbit）

**ALMA Frame(Rx)**

**I2C**

**JTAG**

**ALMA Frame(Tx)**

Powers on Board

+0.95V(for FPGA core)

+1.8V(for FPGA)

+3.3V(for FPGA-IO and SFP+)

+1.0V\_MGTAVCC(for FPGA)

+1.2V\_MGTAVTT(for FPGA)

+1.8V\_MGTVCCAUX(for FPGA)

+1.35VDDR(for DDR)

SFP＋Module

（Bbit）

　　FPGA

SFP＋Module

（Cbit）

**ALMA Frame(Rx)**

**I2C**

**I2C**

**ALMA Frame(Tx)**

**ALMA Frame(Rx)**

**12V**

48ms

In

**Not connected**

**RS485**

Power

Connecter

１２Ｖ

(For Debug)

**PCIE 3.0**

**(Gen3)**

**x 8 lane**

**＜To PCIE　Slot＞**

Figure 1‑4: The Hardware block diagram of the prototype DRXP board

### Comparison table between prototype DRXP and production DRXP (hardware)

The below table shows the hardware differences between prototype DRXP and production DRXP. Some functions can’t use with prototype DRXP. For the detail of restriction, please refer to each function’s explanation.

|  |  |  |
| --- | --- | --- |
| Items | prototype DRXP | production DRXP |
| (A) PCI Express | PCI Express Gen3.0 8lane | PCI Express Gen3.0 16lane |
| (B) Measuring power consumption circuit | It doesn’t have this circuit. | It has this circuit. |
| (C) 48ms TE | The prototype DRXP has one D-sub interface for input RS485 signal. | The production DRXP has two RJ45 interface(48ms TE bracket) for input and output RS485 singnal. |
| (D) EEPROM | It is not implemented. | It is implemented. |
| (E) JTAG | The prototype DRXP uses DF130-10P-1.25H(21) as JTAG interface. (The user needs a special cable to connect Xilinx Platform Cable to this connector. For the detail, please refer to the user's manual.) | The production DRXP uses 087821420 as JTAG interface. ( Xilinx Platform Cable can connect to this connector) |
| (F) Optical interface | It has 3 optical interfaces (=1IF) | It has 6 optical interfaces (=2IF) |

Table 1‑1: Comparison table between prototype DRXP and production DRXP (hardware)

### Comparison table between prototype DRXP and production DRXP (software)

The software of the device driver is designed to be used for both prototype and production DRXPs. However, because of the differences of hardware design of both DRXPs shown in **”1.2.1 Comparison table between prototype DRXP and production DRXP (hardware)”**, some functions of the device driver are not available for prototype DRXPs. Table 1‑2 shows the differences in the software interface between prototype DRXP and production DRXP. For the detail of restriction, please refer each function’s explanation.

|  |  |  |
| --- | --- | --- |
| Items | prototype DRXP | production DRXP |
| (A) DRXPD\_FLASH\_OVHLOG | This function is not available with prototype DRXP.  The prototype DRXP doesn’t have on-board EEPROM for logging overheating event. Therefore, this function can’t use with prototype DRXP board. | Available. |
| (B) GET\_CONSUMPTION | This function is not available with prototype DRXP.  The prototype DRXP doesn’t have circuit for measuring power consumption. | Available. |
| (D) GET\_OVH\_THRESHOLD  GET\_OVH\_FLG  GET\_DFR\_FORCETX  GET\_PARITY\_RANGE  GET\_SEU\_DIAERRCNT  SET\_OVH\_THRESHOLD  SET\_DFR\_FORCETX  SET\_PARITY\_RANGE | These functions are available in device driver version 2.0.8 or later. | Available. |
| (C) Others | Available. (\*1) | Available. (\*1) |

Table 1‑2: Comparison table between prototype DRXP and production DRXP (software)

(\*1) It may not be available with old device driver or FPGA.

If you use DRXP board with the driver and FPGA are not latest,

please confirm each function’s explanation.

## Scope of this document

This document describes how to use DRXP driver version from 1.0.0 to 2.0.9. Following table is the correspondence between driver version and FPGA version.

### Correspondence table between driver version and FPGA (prototype)

|  |  |  |
| --- | --- | --- |
| Driver version | FPGA version (GET\_DFR\_VER)(\*1) | FPGA timestamp |
| 1.0.0 | 7520CA9C | 2016/10/14 12:42:28 |
| 1.0.1 | 90A32AD0 | 2017/01/18 18:43:16 |
| 1.0.2 | FC22F037 | 2017/08/31 15:00:55 |
| 1.0.3 | FC22F037 | 2017/08/31 15:00:55 |
| 1.0.4 | 28A4B761 | 2018/01/05 11:29:33 |
| 1.0.5 | 28A4B761 | 2018/01/05 11:29:33 |
| 1.0.6 | 7524B073 | 2018/10/14 11:01:51 |
| 1.0.7 | 7524B073 | 2018/10/14 11:01:51 |
| 2.0.0 | - | - |
| Form 2.0.1 to 2.0.7 | 7524B073 | 2018/10/14 11:01:51 |
| 2.0.8 | A5A89944 | 2020/11/20 09:37:04 |
| 2.0.9 | - | - |

Table 1‑3: Correspondence table betwenn driver and FPGA(prototype)

(\*1) Your application can get the FPGA version by calling GET\_DFR\_VER.

The above versions are read from data Bytes[7][6][5][4].

Note: FPGA timestamp in the above table are slightly different from the file’s one.

These items are noted for reference.

### Correspondence table between driver version and FPGA (production)

|  |  |  |
| --- | --- | --- |
| Driver version | FPGA version (GET\_DFR\_VER)(\*1) | FPGA timestamp |
| 2.0.0 | - (\*2) | 2019/01/31 09:40 |
| 2.0.1 | 7A27129B | 2019/04/15 17:10:27 |
| 2.0.2 | 7AA73ACC | 2019/05/15 19:43:12 |
| 2.0.3 | A426A8F8 | 2019/08/20 10:35:55 |
| From 2.0.4 to 2.0.7 | 52291486 | 2020/04/10 17:18:06 |
| 2.0.8 | A5A8A504 | 2020/11/20 10:20:04 |
| 2.0.9 | 7A962A09 | 2021/02/01 09:25:58 |

Table 1‑4: Correspondence table between driver version and FPGA (production)

(\*1) Your application can get the FPGA version by calling GET\_DFR\_VER.

The above versions are read from data Bytes[7][6][5][4].

(\*2) The first released FPGA has no version by mistake.

Note: FPGA timestamp in the above table are slightly different from the file’s one.

These items are noted for reference.

## Notation for command instructions

In this document, there are some command instructions. And in these instructions the prompt symbol ‘#’and ‘$’ represent the authority level. ‘$’is for normal user. ‘#’ is for root user. If command instruction is written with ‘#’, please execute this command with root privilege.

Notation for command instructions

## The notation for the level of sample

In this document, we use two notations for representing bit level of ALMA frame data sample.

|  |  |  |
| --- | --- | --- |
| HML notation | DCB notation | Description |
| H bit | D bit | The MSB bit of sample. |
| M bit | C bit | The Middle bit of sample. |
| L bit | B bit | The LSB bit of sample. |

Table 1‑5: The notation for bit level of ALMA frame data sample.

# Structure of Data Buffer

## Structure of Host Receive Ring Buffer

Offset(Virtual) address)

0x0

**Group 0**

4MiB×35block

=140MiB(48ms)

**Frame（48Ｂ = 384bit）**

**See “3.1 Normal frame”**

**Block(4MiB)**

≈1.398ms

0x8C00000

**Group 1**

4MiB×35block

=140MiB(48ms)

0x18400000

**← The　last　Block**

**Block(4MiB)**

**Information Frame(48B)**

**See “3.2 Information frame”**

0x11800000

**Group 2**

4MiB×35block

=140MiB(48ms)

**Padding**

**2800592byte.**

**(unused area)**

：

Only End Block of Group has the Information frame.

**Group 15(Default)**

4MiB×35block

=140MiB(48ms)

You can extend to 32Groups or 64Groups by using ioctl(DRXPD\_SET\_RCVGROUPS).

0x8C000000

Data are written from the offset 0x0.

Data are written in Group0 → Group1 → … → Group15→ Group0→ Group1　・・order.

Note: If you use the DRXP board with NUMA architecture server, this host ring buffer is allocated according to the node where the board belongs.

Figure 2‑1 : The structure of host receive ring buffer

An astronomy data quantized with three bits at a rate of 8GHz are transferred to the DRXP board through an optical fiber cable for each bit. The time sequential data is mapped onto the memory of 16 groups. The each group has 35 blocks of each 4MiB\* (2^20 Bytes) in the memory capacity. The total memory size of each group is 146800640 Byte. So if the address starts from 0x0, the next group starts from 8C00000 in hexadecimal (146800640 in decimal). Each group has 3x10^6 data frames of 48 Byte. The total memory size of the data frames is 144000000 Bytes, which corresponds to a total amount of data for 48 milliseconds (8GHzx3bits makes 24Gbit/s (3GB/s) and 3GB/s for 48 milliseconds makes 144MB.).

The remaining 2800640-Byte data is attached to data frames and completes data area of the group. In the 2800640-Byte data, 48 Bytes is used for an information frame. The 2800592 Bytes are padded with 0, as unused area. (\*Note: MiB is an unit of a data size of 2^20 Bytes. MB (Mega Byte) is 10^6 Bytes.).

The group 0 to 15 is repeated from first to last time-sequentially.

If the GPU interface uses DMA, your application might need to issue a copy instruction on a block-by-block basis in a group. The last 2800592-Bytes data in last block is disposed because it is meaningless.

In the Linux memory management system, you can allocate up to 4MiB to the DMA space (physical addresses are consecutive) at once.

This 4MiB is an unit called ‘Block’ in this document. Driver allocates 35 blocks to configure a group.

Physical addresses in a group are not necessarily consecutive.

## Structure of Host Transmission Buffer

Offset(Virtual)

address)

0x0

MaxSize:

4MiB×35block

=140MiB(48ms)

**Frame（48Ｂ = 384bit）**

**See “3.1 Normal frame”**

**Block(4MiB)**

≈1.398ms

0x8C00000

**←End Bloｃk！**

**Block(4MiB)**

**Padding**

**2800640byte.**

**(unused area)**

**No information frame**

Data are sent from the offset 0x0.

Note: If you use the DRXP board with NUMA architecture server, this host ring buffer is allocated according to the node where the board belongs.

Figure 2‑2: The structure of host transmission buffer

Physical　addresses are consecutive in block as noted in the previous section. But physical addresses in a group are not necessarily consecutive.　If your application interface intends to use the DMA, it might need to issue a copy command on a block-by-block basis.

# Format of Frame in Host Data Buffer

## Normal frame (DRXP F2 format)

The following tables represent one ALMA frame payload for each bit (Hbit/Mbit/Lbit). The index represents

the time order. The Alphabet of the following tables represents the sample Level.

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| H0 | H1 | H2 | ・・・・・・・・・・・・・ | H13 | H14 | H15 |
| H16 | H17 | H18 | ・・・・・・・・・・・・・ | H29 | H30 | H31 |
| H32 | H33 | H34 | ・・・・・・・・・・・・・ | H45 | H46 | H47 |
| H48 | H49 | H50 |  | H61 | H62 | H63 |
| ・・・・・・・・・・・・・ | | | | | | |
| H96 | H97 | H98 |  | H109 | H110 | H111 |
| H112 | H113 | H114 | ・・・・・・・・・・・・・ | H125 | H126 | H127 |

Hbit ALMA frame

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| M0 | M1 | M2 | ・・・・・・・・・・・・・ | M13 | M14 | M15 |
| M16 | M17 | M18 | ・・・・・・・・・・・・・ | M29 | M30 | M31 |
| M32 | M33 | M34 | ・・・・・・・・・・・・・ | M45 | M46 | M47 |
| M48 | M49 | M50 |  | M61 | M62 | M63 |
| ・・・・・・・・・・・・・ | | | | | | |
| M96 | M97 | M98 |  | M109 | M110 | M111 |
| M112 | M113 | M114 | ・・・・・・・・・・・・・ | M125 | M126 | M127 |

Mbit ALMA frame

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| L0 | L1 | L2 | ・・・・・・・・・・・・・ | L13 | L14 | L15 |
| L16 | L17 | L18 | ・・・・・・・・・・・・・ | L29 | L30 | L31 |
| L32 | L33 | L34 | ・・・・・・・・・・・・・ | L45 | L46 | L47 |
| L48 | L49 | L50 | ・・・・・・・・・・・・・ | L61 | L62 | L63 |
| ・・・・・・・・・・・・・ | | | | | | |
| L96 | L97 | L98 | ・・・・・・・・・・・・・ | L109 | L110 | L111 |
| L112 | L113 | L114 | ・・・・・・・・・・・・・ | L125 | L126 | L127 |

Lbit ALMA frame

One frame has 48Bytes (=384bits). The production DRXP board organizes the above ALMA frame payloads

as the following table. DRXP board assumes that host machine is a little endian machine.

|  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- |
|  | Bit7 | Bit6 | Bit5 | Bit4 | Bit3 | Bit2 | Bit1 | Bit0 |
| Byte0 | M1 | L1 | H16 | M16 | L16 | H0 | M0 | L0 |
| Byte1 | L18 | H2 | M2 | L2 | H17 | M17 | L17 | H1 |
| Byte2 | H19 | M19 | L19 | H3 | M3 | L3 | H18 | M18 |
| Byte3 | M5 | L5 | H20 | M20 | L20 | H4 | M4 | L4 |
| Byte4 | L22 | H6 | M6 | L6 | H21 | M21 | L21 | H5 |
| Byte5 | H23 | M23 | L23 | H7 | M7 | L7 | H22 | M22 |
| Byte6 | M9 | L9 | H24 | M24 | L24 | H8 | M8 | L8 |
| Byte7 | L26 | H10 | M10 | L10 | H25 | M25 | L25 | H9 |
| ・・・・・・・・・・・・・ | | | | | | | | |
| Byte43 | L122 | H106 | M106 | L106 | H121 | M121 | L121 | H105 |
| Byte44 | H123 | M123 | L123 | H107 | M107 | L107 | H122 | M122 |
| Byte45 | M109 | L109 | H124 | M124 | L124 | H108 | M108 | L108 |
| Byte46 | L126 | H110 | M110 | L110 | H125 | M125 | L125 | H109 |
| Byte47 | H127 | M127 | L127 | H111 | M111 | L111 | H126 | M126 |

Figure 3‑1: The structure of normal frame (DRXP F2 format)

### The kinds of normal frame (DRXP F1 format)

There are two kinds of normal frame formats, DRXP F1 format and DRXP F2 format. The prototype DRXP version 7524B073 or older uses DRXP F1 format. The production DRXP board and the prototype DRXP (version A5A89944 or later) uses DRXP F2 format. The following figure is the structure of DRXP F1 format. The prototype DRXP board also assumes that host machine is a little endian machine.

Offset

0x2F

0x1

0x0

3bit

M

H

H

H

M

・・・・・・・・・・・・・

L

L

M

H

L

M

L

H

L

M

L

M

H

M

H

L

L

M

H

Msb

Lsb

1Byte

1Byte

1Byte

3bit×128＝384bit=48Byte

Figure 3‑2: The structure of normal frame (DRXP F1 format)

The new data are offset 0x2F, The old data are offset 0.

The bit assignment of the ALMA frame is preserved,

128 samples with 3bits in a frame.

## Information frame

An information frame is appended to the end of each group. It contains a status of the frames in a group and the reception time of the first frame of a group.

Offset

0xC

0x2F

0x8

0x0

unsigned int

Frame Status

Parity

Unused

area

(undefined)

struct timespec

software UTC timestamp

unsigned int

Frame Status

unsigned long

hardware 48ms tick count

8Byte

14Byte

6Byte

4Byte

16Byte

48Byte

Figure 3‑3: The structure of information frame

**Hardware 48ms tick count (64bit)** : Free run counter. The counter on the DRXP board counts up at each received metaframe. This data is in little endian format. The value range is [0, 264 -1 ]. This counter is reset when the device file is opened or RESET\_DFR with master reset bit is called.

**Frame status (32bit)** : If any of them is 1, the data of the group should be discarded because it has an error.

These bits represent real-time statuses of the frames in a group.

Bit[31:20] unused bits(undefined)

Bit[19] unused bits(undefined)

Bit[18] 1… Couldn’t find the meta frame bit in ALMA Frame[Bbit], 0…OK

Bit[17] 1… Couldn’t find the meta frame bit in ALMA Frame[Cbit], 0…OK

Bit[16] 1… Couldn’t find the meta frame bit in ALMA Frame[Dbit], 0…OK (\*6)

Bit[15] unused bits(undefined)

Bit[14] 1…ALMA　Frame Sequence Count Error[Bbit], 0…OK

Bit[13] 1…ALMA　Frame Sequence Count Error[Cbit], 0…OK

Bit[12] 1…ALMA　Frame Sequence Count Error[Dbit], 0…OK (\*5)

Bit[11] unused bit(undefined)

Bit[10]　　1…ALMA　Frame CheckSum Error[Bbit], 0…OK

Bit[9]　　　1…ALMA　Frame CheckSum Error[Cbit], 0…OK

Bit[8]　　　1…ALMA　Frame CheckSum Error[Dbit], 0…OK (\*4)

Bit[7] unused bit(undefined)

Bit[6]　　　1…ALMA　Frame Sync Error[Bbit], 0…OK

Bit[5]　　　1…ALMA　Frame Sync Error[Cbit], 0…OK

Bit[4]　　　1…ALMA　Frame Sync Error[Dbit], 0…OK (\*3)

Bit[3] unused bits(undefined)

Bit[2] 1…Incomplete flag (\*7)

Bit[1] 1…Dummy data is filled. (\*1)

Bit[0] 0… Bbit/Cbit/Dbit are synchronized by meta bit on each ALMA frame (\*2),

1… not synchronized (\*2).

(\*1) When production DRXP board is in reception mode, it transfers the dummy data to user memory

until ALMA frame logic receives the first valid data. The valid data means that ALMA frame has no

checksum error, frame sync error and B/C/D bits are synchronized.

(\*2) In the following cases, it sets this bit to 1.

(1) When a metaframe can’t be detected 48ms after the last detected metaframe at B/C/D bit,

if the timing is earlier or later than this, it will set this bit. When it detects metaframe earlier than 48ms after the last detected metaframe, it also sets ALMA Frame CheckSum Error B/C/D bits and ALMA Frame Sequence Count Error B/C/D bits. These bits represent the data in the group buffer includes undefined data.

(2) When a metaframe is detected 48ms after the last detected metaframe at B/C/D bit. But group has (some) check sum error frame(s).

(\*3)(\*4)(\*5)(\*6)

The data in the group is extracted payload of 3,000,000 ALMA frames.

If the ALMA frame de-formater detects errors in these frames,

it sets error bit in the information frame.

When the ALMA frame de-formatter detects an error in the sequence counter area of ALMA frame,

"ALMA frame Sequence Count Error" bit is set to 1.

Normally, sequence counter of ALMA frame is increased by 1.

This error bit represents some ALMA frames in group have wrong sequence number.

When the ALMA frame de-formatter detects an error in the checksum area of ALMA frame,

the "ALMA frame Checksum Error" bit is set to 1.

This error bit represents the caluculated checksum of some frames in group   
is not much each ALMA frame's checksum value.

When the ALMA frame de-formatter fails to find ALMA frame's sync pattern received data.

The "ALMA frame Sync Error" bit is set to 1.

The ALMA frame has sync pattern for finding the ALMA frame from received bit data.

This error bit represents the ALMA frame de-formatter can't find sync pattern at expected position.

If there is no error, this sync pattern appears every 160 bits (The length of ALMA frame is 160bits).

”Couldn’t find the meta frame bit in ALMA Frame” represents a frame de-formatter status.

When the ALMA frame de-formatter cannot find “MetaFrameBit” in the ALMA frame in group,

”Couldn’t find the meta frame bit in ALMA Frame” is set to 1.

If it detects an earlier meta-frame bit, it sets all checksum error bits and all sequence error bits in the information frame. And it also sets an incomplete flag. For the detail of incomplete flag, please refer to the note (\*7).

For the detail of the ALMA frame de-formatter, please refer to the user's manual (de-formatter).

(\*7) When writing target group moves to the next before filling all frames, this bit is asserted.

It means metaframet is detected earlier than 48ms after the last detected metaframe. When it detects

metaframe earlier than 48ms after the last detected metaframe, it also sets ALMA Frame CheckSum Error B/C/D bits and ALMA Frame Sequence Count Error B/C/D bits. These bits represent the state of the group buffer data includes undefined data. This flag is available with device driver version 2.0.8 or later. And it needs a newer FPGA version than “52291486”. (In the case of prototype DRXP, it needs a newer FPGA than 7524B073).

**Parity:**

Bit[47：32] Data parity [Bbit]

Bit[31：15] Data parity [Cbit]

Bit[15：0] Data parity [Dbit]

The FPGA logic generates the parity from the data that the de-formatter logic receives. Your application can confirm the data integrity in the FPGA logic with this parity. The logic starts to generate the parity when it detects the metaframe bit. Your application can set the range of parity calculation from 1 to 3,000,000 ALMA frames. For more details please refer to **“7.5 The parity for checking FPGA data integrity”**..

**UTC (struct timespec)** : software timestamp is set when the DRXP board writes a group to the Host Ring Buffer completely.

struct timespec {

time\_t tv\_sec; /\* seconds(64bit) \*/

long tv\_nsec; /\* nanoseconds(64bit) \*/

};

These data (tv\_sec and tv\_nsec) are in little endian format.

When a DRXP board receives the first frame which contains a metaframe bit, it interrupts the CPU and the driver stores the current time in a certain place in the system memory.

Right after all frames (blocks) of the group are transferred to the system memory, the driver copies the time to UTC(struct timespec). To get the current time, the driver calls getnstimeofday() which typically returns the time of the wall clock that can be adjusted using NTP or some other methods. You need to adjust the wall clock of the system using NTP (or another appropriate method) to set the correct timestamp in the information frame. This time stamp is the number of seconds since Epoch, 1970/01/01/00:00:00(UTC). And the value range of the tv\_nsec is [0, 999999999]. For hardware derived data timestamps, please use the tick count field at every 48msec. DRXP driver version 1.0.1 or older uses TAI for representing software timestamp.

# Usage

## The note on the driver use

When device driver is installed to system, device driver tries to find prototype/production DRXP board. If it detects the prototype/production DRXP board, it creates device file, /dev/drxpX (X=0,1,2,…). Your application can control/monitor DRXP boards via these device file. The device file index(X) is assigned from zero in order. When it detects prototype DRXP board, it creates one device file. (e.g. /dev/drxpX). When it detects production DRXP board, it creates two device files. (e.g. /dev/drxpX, /dev/drxp(X+1)) . The smaller number is imparted IF1. For example, one prototype DRXP board and one production DRXP board are in one server. And assume that the device driver finds prototype DRXP board at first. In this case, /dev/drxp0 is created for prototype DRXP board, /dev/drxp1 is created for production DRXP board IF1, /dev/drxp1 is created for production DRXP board IF2.

This driver does not allow a device file to open multiply. This driver is not reentrant. When you want to use single device file in many applications, you need to put them together so that single application access to single device file. And this driver supports only x86-x64 architecture.

C/C++ programs include the header file “drxpd.h” to use the data structures and constants in your application. The header file currently contains two sets of constants, the first is for kernel space (the driver itself), and the other is for the user-software. Control and monitoring items which contain ‘\*\_ID\_\*’are latter. For example, RESET\_DFR is for kernel space only, and you should use RST\_DFR\_ID\_ALL (or RST\_DFR\_ID\_D, RST\_DFR\_ID\_C, or RST\_DFR\_ID\_B).

We recommend your application should include <errno.h> and check the errno when DRXP API fails. For the detail of errno(3), please execute man command and read manual of errno(3). (e.g. $man errno.) Your application should include “/usr/include/asm-generic/errno.h” and “/usr/include/asm-generic/errno-base.h” to include the same errno header that the device driver uses explicitly.

## Installation of the Driver

About the installation driver, please refer to the user’s　manual.

## Uninstallation of the Driver

About the uninstallation driver, please refer to the user’s manual.

## Example of reception

START

“Open” with O\_RDONLY flag.

Use "ioctl (DRXPD\_SET\_RCVGROUPS)" to change the number of groups.

Not required if it is not necessary to change the number of group from the nominal value of 16.

Call "mmap" to get the address of the host receive ring buffer.

Use "ioctl (DRXPD\_SET\_RCVEVT)" to register the reception completion event.

4 (\*1)

Use "ioctl (DRXPD\_INIT)" to initialize.

3

Use "ioctl (DRXPD\_RCVEVTSTART)"

to start a reception event.

Use "ioctl (DRXPD\_RCVSTART)"

to start reception.

The event does

not occur

Wait for the reception

completion event

Event occurs

2

1

The Event does

not occur

Wait for the reception

completion event

The event occurs

To the next page

(\*1) If you call RESET\_DFR with master reset flag, please restart operation from 4.

From the previous page

Other values

Check the value

read from eventfd

5

The value is 1

Use "ioctl (DRXPD\_GET\_RCVSTS)",

to confirm that the reception of a group

is completed.

Your application processes the data in a valid group.

The process finishes. Clear the status of the group is no longer needed by calling "ioctl (DRXPD\_CLR\_RCV)".

No

1

Exit the reception?

Yes

Use "ioctl (DRXPD\_RCVSTOP)"

to stop reception.

Restart Reception

2

Restart reception?

Use "ioctl (DRXPD\_RCVEVTSTOP)"

to stop the event.

Restart Reception

3

Restart reception?

Call "munmap" to free the address of the host receiving ring buffer

Call "Close".

END

From the previous page

5

Use "ioctl (DRXPD\_RCVSTOP)"

to stop reception.

Use "ioctl (DRXPD\_RCVEVTSTOP)"

to stop the event.

Cooling board

and

check overheatflag

The overheat flag is not

cleared

Setup reception again

4

## Example of transmission

START

“Open”with O\_RDWR flag.

Change Data and

Restart the Transmission

Call "mmap" to get the address of the transmission buffer.

2 (\*1)

Your application writes data to the transmission buffer.

Transmission function transmits the data you write here repeatedly.

Use "ioctl (DRXPD\_SET\_SNDSIZ)" to set the transmission size.

Yes

Change

Data?

Use "ioctl (DRXPD\_INIT)” to initialize.

No

Use "ioctl (DRXPD\_SNDSTART)"

to start transmission.

1

Check overheat flag

Overheat is detected

No

Exit the transmission?

Use "ioctl (DRXPD\_SNDSTOP)"

to stop the transmission.

Restart

Transmission

Restart

Transmission ?

Call "munmap" to free the address of the transmission buffer

Call "Close".

END

From the previous page

1

Use "ioctl (DRXPD\_SNDSTOP)"

to stop the transmission.

Cooling board

and

check overheatflag

The overheat flag

Is not cleared.

Setup transmission again

2

(\*1) If you call RESET\_DFR with master reset flag, please restart operation from 2.

## How to see the device driver logs

Logs of DRXP driver are output to syslog. You can see the log with Linux command. Following command dumps the temporary buffer of syslog. You can check syslog about recent event.

$ dmesg

or

$ journalctl –k

Please add “ | grep drxpd” to the tail of command to check the log entry related to the DRXP driver as below.

$ dmesg | grep drxpd

or

$ journalctl –k | grep drxpd

The syslog recorded to the server is placed at /var/log/messages. This file is plain text. Therefore, you can check this file with Linux command, for example “less”. You need the root privilege to refer to this file.

$ su (please input the root user password)

# less /var/log/messages

or

$ sudo less /var/log/messages

(please input the sudo password)

### Log format

The DRXP driver has three log formats as below.

* The generic log message.

Ex. Jun 18 16:54:27 localhost.localdomain kernel: drxpd : start loading module.

This log message type is for logging the general event.

* The log message for the ALMA frame ID.

Ex. Jun 18 16:54:27 localhost.localdomain kernel: drxpd{#0} IF{#01} : create character device

file. (drxpd0).

This log message type is for logging the event especially related to ALMA frame IF. The word drxpd{**#0**} in the above message means this log message is related to device file /dev/drxpd**0**. The word IF{**#01**} in the above message means this log message is related to IF1.

* The log message for the specific bit of the ALMA frame IF.

Ex. Jun 18 20:38:28 localhost.localdomain kernel: drxpd{#0} IF{#01} Bit{#D} : SFP S/N: 3YT004400219

This log message type is for logging the event especially related to specific bit in ALMA frame IF. The word drxpd{**#0**} in the above message means this log message is related to device file /dev/drxpd**0**. The word IF{**#01**} in the above message means this log message is related to IF1. The word Bit{**#D**} in the above message means this log message is related to the **D** bit in the IF data.

# API detailed description

## open

**【Function】**

　　 This API opens a device file.

**【Format】**

int open(const char \*pathname, int flags, mode\_t mode);

**【Parameter】**

const char \*pathname…

Specify　a device file ( /dev/drxpdX). X is assigned from zero in order.

About the available value for X, please refer the section “4.2 Installation of the Driver”.

Please use the serial number acquisition function i.e. Use GET\_DFR\_SN via ioctl(DRXPD\_GET\_MONSTS).

You can also identify an IF number by using GET\_DFR\_SN via ioctl(DRXPD\_GET\_MONSTS) with the driver version 2.0.3 or later.

int flags…

When　you receive data, specify O\_RDONLY.

When　you transmit data, specify O\_RDWR.

O\_WRONLY is invalid.

The DRXP-board can not receive ALMA frames in the transmission mode and the DRXP-board can

not transmit ALMA frames in the reception mode. But the DRXP-board can transmit ALMA frames

with a fixed payload by using SET\_DFR\_FORCETX\_ID even in the reception mode, namely when it

opens with O\_RDONLY. See **“6.1.21 GET\_DFR\_FORCETX”** and **“6.2.12 SET\_DFR\_FORCETX”** for

more details.

mode\_t mode…

Not used. Specify 0

**【Return】**

A file descriptor is returned. If an error occurs, -1 is returned in which case, an errno is set.

**【Errors】**

This function can fail with the following errors.

EBUSY (driver version 2.0.3 or later)

When the device file is already opened, this command sets EBUSY to errno.

Or when the device file is referred by some other function.

For example, if user application has not released the mmaped memory yet,

please release it. After that, please try again.

ECANECLED (driver version 2.0.3 or later)

When this command is canceled by hardware error while driver tries to reset hardware,

This function sets ECANCELED to errno.

If this error occurrs often, it may be hardware error is happend.

Please execute cold-reboot and try again.

ENOMEM (driver version 2.0.3 or later)

When this command fails to allocate memory, this function sets ENOMEM to errno.

Please check the size of free memroy on your system.

EAGAIN (driver version 2.0.2 or older)

When this command fails to reset FPGA, this function sets EAGAIN to errno.

And also, the device file is already opened, this function sets EAGAIN to errno.

EINVAL

When the flag is not O\_RDONLY or O\_RDWR, this function sets EINVAL to errno.

EIO

When this function fails to read or write internal register.

This errno means device driver can't control the device.

If this error occurrs often, it may be hardware error is happend.

Please execute cold-reboot and try again.

## close

**【Function】**

　　　The API closes　a device file.

**【Format】**

int close(int fd);

**【Parameter】**

int fd…

Specify a file descriptor.

**【Return】**

0 is returned. If an error occurs, -1 is returned in which case, an errno is set.

**【Errors】**

This function can fail with the following errors.

EBUSY (driver version 2.0.3 or later)

When other functions refer to the same device file, this function sets EBUSY to errno.

If your application closes a device file without calling munmap system call, please call munmap and release memory for DMA.

## mmap

**【Function】**

　　　The API for mapping the host data buffer into the user space.

**【Format】**

void \*mmap(void \*addr, size\_t length, int prot, int flags, int fd, off\_t offset);

**【Parameter】**

void \*addr…

Specify NULL.

size\_t length…

When you receive data, specify 4194304 Byte x 35 Blocks x Number of Groups (= 146800640 Byte x Number of Groups). The number of groups must be equal to the value specified by ioctl(DRXPD\_SET\_RCVGROUPS).

When you transmit data, specify 146800640 Byte regardless of ioctl (DRXPD\_SET\_SNDSIZ).

int prot…

When you receive data, specify PROT\_READ.

When　you transmit data, specify PROT\_READ | PROT\_WRITE.

int flags…

Specify　 MAP\_SHARED.

int fd…

Specify the file descriptor.

off\_t offset…

Specify 0.

**【Return】**

Mapped address pointer is returned. If an error　occurs, MAP\_FAILED is returned in which case, errno is set. MAP\_FAILED is defined in the sys/mman.h.

Mapped address pointer is a virtual address. Mapped area is consecutive in virtual.

**【Errors】**

This function can fail with the following errors.

EAGAIN

The DMA buffer of this device file is already mmaped to user memery space.

EINVAL

When argument (offset) or argument(length) is invalid.

Please confirm the description of Parameter field.

ENOMEM (driver version 2.0.3 or later)

When this function fails to allocate memory, it sets ENOMEM to errno.

Please check the size of free memroy on your system.

**【Remarks】**

Physical　addresses are consecutive in block(one block size is 4MiB).　If your application interface uses DMA, it might need to issue a copy instruction on a block-by-block basis and the physical address.

## munmap

**【Function】**

　　　The API for unmapping the data buffer.

**【Format】**

int munmap(void \*addr, size\_t length);

**【Parameter】**

void \*addr…

Specify the pointer mapped by mmap.

size\_t length…

Specify the same length as the value specified by mmap.

**【Return】**

0 is returned. If an error occurs, -1 is returned.

**【Remarks】**

An error does not occur even if no page is mapped within the specified range.

## ioctl

Ioctl has a variety of functions by the commands.

The second argument changes by commands.

**【Common Format】**

int ioctl( int fd, unsigned long command,(Various types) argument);

**【Synchronization/Asynchronization】**

The following subsection has “Synchronization” field. The DRXP APIs operate data in the kernel space and registers in the hardware. This field represents policy how DRXP APIs operate hardware. There are two policies, synchronized and asynchronized.

In the case of the synchronous policy API, when it tries to change a hardware status or register value, it confirms the result after operation. If it isn't completed operation successfully in a certain time, it sets ETIME/ECANCELD to an errno. This timeout length or latency is written in the description of each ioctl command and control items. This value is not taken into account the physical layer timeout (the completion timeout value of PCIe packet). A Physical layer timeout value is determined by implementation of motherboard. Default value is between 50us and 50ms. System can change this value. Maximum value is from 17s to 64s. When physical layer timeout is occurred, DRXP API sets EIO to an errno. In such case, actually timeout value is the sum of physical layer timeout and ioctl command timeout (or latency). User can check physical layer timeout value by lspci command. (Please refer following description, “Confirmation method of completion timeout”.) This EIO means device driver can’t control device via physical layer. In this case, it may need cold-reset of user’s system.

In the case of the asynchronous policy API, when it tries to change a hardware status or register value, it doesn't wait for the completion. It returns immediately after its operation.

■Confirmation method of completion timeout

If you want to check a completion timeout value of your system, please execute following commands.

Step1: Detemin DRXP board

$ lspci

**5e:00.0** Memory controller: Xilinx Corporation Device 903frep Xilinx

lspci command represents the device by the combination of bus number, device number and function number. Notation is “Bus:Device.Function”. Above output means DRXP board’s bus is 5e, device number is 00, and function number is 0.

Step2: Confirm the connection of PCI bus

Please execute following command and find out DRXP board

$lspci –t

(snip)

+-[0000:**5d**]-+-**00.0**-[**5e**]----**00.0**

(snip)

Above output means “5e:00.0(DRXP board)” is connected to “5d:00.0”.

Step3: Please check the DevCtl2 register value of the upstream device.

# lspci –vvv –s **5d:00.0** (<- upstream device )

**5d:00.0** PCI bridge: Intel Corporation Sky Lake-E PCI Express Root Port A (rev 04) …

(snip)

DevCtl2: Completion Timeout: 260ms to 900ms, TimeoutDis-, LTR-, OBFF Disabled ARIFwd-

(snip)

Above output means your systems completion timeout is 260ms to 900ms.

### Command :DRXPD INIT

**【Function】**

This command initializes the FPGA logic for receiving/transmitting ALMA frame.

Your application should call this command per device file. On a production DRXP board, if the DRXPD\_INIT command is executed for one IF does not affect another IF.

**【Format】**

int ioctl(int fd, DRXPD\_INIT, int \*p\_result);

**【Parameter】**

int fd…

Specify a file descriptor.

int \*p\_result…

Specify a pointer to store the result below.

<Device driver version 2.0.3 or later>

Driver always sets DRXPD\_OK(0) to this argument.

This reporting system is not available. Please use an errno to specify the reason.

<Device driver version 2.0.2 or older>

DRXPD\_OK(0): Initialization success.

DRXPD\_NG(-1): Initialization failure.

DRXPD\_OVH(-2): Initialization failure due to overheating.

**【Return】**

0 is returned. If an error occurs, -1 is returned in which case errno is set.

**【Errors】**

This command can fail with the following errors.

EIO

When this command fails to read or write internal register.

This errno means device driver can't control the device.

If this error occurrs often, it may be hardware error is happend.

Please execute cold-reboot and try again.

ECANCELED

This errno means device is in overheat state.

We recommend that you stop using a device for cooling.

And also please check the syslog.

EINVAL

A device file is opened with invalid flags.

EFALUT

When this command fails to copy the execution result to p\_result, it sets EFAULT to errno.

In this case, please ignore the value of \*p\_result.

User application may give an invalid address to this pointer,

please check its address.

**【Errors】**(Continued from the previous page.)

EBUSY

When the resource busy.

When the device has already started transmitting/receiving frame, it sets EBUSY to errno.

EAGAIN

When this command detects an illegal sequence.

Please confirm the operation sequence. (**“4.4 Example of reception, 4.5 Example of transmission”**)

ETIME

When timeout occurs, it sets ETIME to errno.

Please refer to the description of “Synchronization” field.

**【Synchronization】**

Synchronous

This function works as asynchronous with the driver version 2.0.2 or older.

Timeout

Maximum timeout value is 10s.

When timeout occurs, this function returns -1 and sets ETIME to errno.

This function can fail with timeout in following case.

.

\* DMA transfer is not completed

(DRXPD\_INIT sends ALMA frame payload data

or the list of address for DMA transfer).

If timeout occurs, your system might need more time to transfer data by DMA. You can change the timeout value with device driver option. Please reload device driver with the option as below. You can extend timeout with this option. Unit is 100[ms]. The following example adds 200[ms] to the timeout value.

# insmod drxpd.ko gi\_init\_timeout=2

### Command :DRXPD\_SET\_RCVEVT

**【Function】**

　　This command sets an eventfd of the reception completion notification.

**【Format】**

int ioctl(int fd, DRXPD\_SET\_RCVEVT, int eventfd);

**【Parameter】**

int fd…

Specify the file descriptor.

int eventfd…

Specify the event file descriptor gained by calling “int eventfd(unsigned int initval, int flags)”. eventfd should not use EFD\_SEMAPHORE. The resource related to eventfd in the device driver is released when user process is finished.

**【Return】**

0 is returned. If an error occurs, -1 is returned in which case errno is set.

**【Errors】**

This command can fail with the following errors.

EAGAIN

When this command detects an illegal sequence.

Mmap system call is not called before calling this API.

Please confirm the operation sequence. (**“4.4 Example of reception”**)

EBUSY

When the resource busy.

The device has already started receiving data process.

EBADF

The eventfd your application gives to driver is invalid.

Please confirm whether the eventfd file descriptor is valid or not.

EINVAL

The eventfd your application gives to driver is not an eventfd.

Please confirm whether the eventfd file descriptor is valid or not.

Or, when you don’t open the device file with O\_RDWR flag.

**【Synchronization】**

Synchronous

There is no FPGA access.

**【Description】**

When your application needs the reception completion notification, use this command.

Reception completion notification is generated on a group basis in the Host Receive Ring Buffer. In other word, it is generated in 48 milliseconds.

When it encounters overheat, 100 is read from the eventfd on the production DRXP board with the driver version 2.0.2 or older. Normally 1 is read from the eventfd.

On the production DRXP board with version 2.0.3 or later, when it encounters the overheating, 100 or 101 is read from the eventfd. When the FPGA junction temperature goes over the use-defined threshold, it returns 100. When the FPGA junction temperature goes over the hardware threshold, it returns 101. Normally eventfd return 1.

The resource acquired by driver in this function is released when user application calls 5.2 close or user application is terminated. Even if user application is terminated by error (e.g. SEGMENTATION FAULT), this resource is released.

**【Remarks】**

When you use this command, you must stop the reception. If you use it during the operation, ioctl returns an error(-1). Also ioctl returns an error(-1) if you open the device file with in O\_RDWR flag.

### Command :DRXPD RCVEVTSTART

**【Function】**

This command starts the event.

**【Format】**

int ioctl(int fd, DRXPD\_RCVEVTSTART, int \*p\_result);

**【Parameter】**

int fd…

Specify the file descriptor.

int \*p\_result…

Specify the pointer to store the result below.

<Device driver version 2.0.3 or later>

Driver always sets DRXPD\_OK(0) to this argument.

This reporting system is not available. Please use an errno to specify the reason.

<Device driver version 2.0.2 or older>

DRXPD\_OK(0): It started to event.

DRXPD\_NG\_MODE(-2): You. Don’t open the device file with O\_RDWR flag.

DRXPD\_NG\_BUSY(-3): The event has already started.

DRXPD\_NG(-1): Other failures.

**【Return】**

0 is returned. If an error occurs, -1 is returned in which case errno is set**.**

**【Errors】**

This command can fail with the following errors.

EINVAL

When you don’t open the device file with O\_RDWR flag.

If you application opened device file with other than O\_RDWR flag,

please open device file with the O\_RDWR flag.

EIO

When this command fails to read or write internal register.

This errno means device driver can't control the device.

If this error occurs often, it may be hardware error is happened.

Please execute cold-reboot and try again.

EBUSY

When the resource busy.

User application has already called ioctl(DRXPD\_RCV\_EVTSTART) with this device file.

EAGAIN

When this command detects an illegal sequence.

Mmap, ioctl(DRXPD\_INIT) or ioctl(DRXPD\_SET\_RCVEVT) is not called

before calling this API.

Please confirm the operation sequence. (**“4.4 Example of reception”**)

**【Errors】**(Continued from the previous page.)

EFAULT

When this command failed to copy the execution result to p\_result,

it sets EFAULT to errno. In this case, please ignore the value of \*p\_result.

User application may give an invalid address to this pointer,

please check its address.

ETIME(driver version 2.0.3 or later)

When timeout occurs, it sets an errno to ETIME.

Please confirm the description of Synchronization field.

**【Synchronization】**

Synchronous

It works as asynchronous with the driver version 2.0.2 or older.

Timeout

Timeout value is 1ms.

This function checks the register after writing data. If it fails to change register, it returns -1 and sets ETIME to errno. In such a case, there may be a hardware error. Please call RESET\_DFR command with master\_reset bit or reload DRXP driver.

### Command :DRXPD RCVEVTSTOP

**【Function】**

This command stops the event.

**【Format】**

int ioctl(int fd, DRXPD\_RCVEVTSTOP, int \*p\_result);

**【Parameter】**

int fd…

Specify the file descriptor.

int \*p\_result…

Specify the pointer to store the result below.

<Device driver version 2.0.3 or later>

Driver always sets DRXPD\_OK(0) to this argument.

This reporting system is not available. Please use an errno to specify the error reason.

<Device driver version 2.0.2 or older>

DRXPD\_OK(0): Event stops.

DRXPD\_NG\_MODE(-2):You don’t open the device file with \_RDWR flag.

DRXPD\_NG\_BUSY(-3): The event has already stopped .

DRXPD\_NG(-1): Other failures.

**【Return】**

0 is returned. If an error occurs, -1 is returned in which case errno is set.

**【Errors】**

This command can fail with the following errors.

EINVAL

When you don’t open the device file with O\_RDWR flag.

If you application opened device file with other than O\_RDWR flag,

please open device file with the O\_RDWR flag.

EIO

When this command fails to read or write internal register.

This errno means device driver can't control the device.

If this error occurrs often, it may be hardware error is happend.

Please execute cold-reboot and try again.

EBUSY

When the resource busy.

The device driver has already stopped time event notification via eventfd.

Or it has not started time event notification yet.

EAGAIN (version 2.0.3 or later)

When this command detects an illegal sequence.

ioctl(DRXPD\_RCVSTOP) is not called before calling this API.

Please confirm the operation sequence. (**“4.4 Example of reception”**)

EFAULT

When this command fails to copy the execution result to p\_result,

it sets EFAULT to errno. In this case, please ignore the value of \*p\_result.

User application may give an invalid address to this pointer,

please check its address.

ETIME(driver version 2.0.3 or later)

When timeout occurs.

Please confirm the description of Synchronization field.

**【Synchronization】**

Synchronous

It works as asynchronous with the driver version 2.0.2 or older.

Timeout

Timeout value is 1ms.

This function checks the status of register after writing data. If it fails to change register status, it returns -1 and sets ETIME to errno. In such a case, there may be a hardware error. Please call RESET\_DFR command with master\_reset bit or reload DRXP drvier.

### Command :DRXPD RCVSTART

**【Function】**

This command start the reception.

**【Format】**

int ioctl(int fd, DRXPD\_RCVSTART, int \*p\_result);

**【Parameter】**

int fd…

Specify the file descriptor.

int \*p\_result…

Specify the pointer to store the result below.

<Device driver version 2.0.3 or later>

Driver always sets DRXPD\_OK(0) to this argument.

This reporting system is not available. Please use an errno to specify the error reason.

<Device driver version 2.0.2 or older>

DRXPD\_OK(0): It started to receive.

DRXPD\_NG\_MODE(-2): You don’t open the device file with O\_RDWR flag.

DRXPD\_NG\_BUSY(-3): The receiving process has already started.

DRXPD\_NG\_INIT(-4): It has not been initialized. The initialization is required.

DRXPD\_NG(-1): Other failures.

**【Return】**

0 is returned. If an error occurs, -1 is returned in which case errno is set.

**【Errors】**

This command can fail with the following errors.

EINVAL

When you don’t open the device file with O\_RDWR flag.

If you application opened device file with other than O\_RDWR flag,

please open device file with the O\_RDWR flag.

EAGAIN

When this command detects an illegal sequence.

Mmap, DRXPD\_INIT and DRXPD\_RCVEVTSTART is not called before calling this API.

Please confirm the operation sequence. (**“4.4 Example of reception”**).

In the device driver version 2.0.2 or older,

it doesn’t care whether ioctl(DRXPD\_RCVEVTSTART) is called before this command.

EIO

When this command fails to read or write internal register.

This errno means device driver can't control the device.

If this error occurrs often, it may be hardware error is happend.

Please execute cold-reboot and try again.

EBUSY

When the resource busy.

The device has already started receiving data process.

EFAULT

When this command fails to copy the execution result to p\_result,

it sets EFAULT to errno. In this case, please ignore the value of \*p\_result.

User application may give an invalid address to this pointer,

please check its address.

ETIME(driver version 2.0.3 or later)

When timeout occurs.

Please confirm the description of Synchronization field.

**【Synchronization】**

Synchronous

It works as asynchronous with the driver version 2.0.2 or older.

Timeout

Timeout value is 1ms.

This function checks the status of register after writing data. If it fails to change register status, it returns -1 and sets ETIME to errno. In such a case, there may be a hardware error. Please call RESET\_DFR command with master\_reset bit or reload DRXP drvier.

### Command :DRXPD RCVSTOP

**【Function】**

This command stops the reception.

**【Format】**

int ioctl(int fd, DRXPD\_RCVSTOP, int \*p\_result);

**【Parameter】**

int fd…

Specify the file descriptor.

int \*p\_result…

Specify the pointer to store the result below.

<Device driver version 2.0.3 or later>

Driver always sets DRXPD\_OK(0) to this argument.

This reporting system is not available. Please use an errno to specify the error reason.

<Device driver version 2.0.2 or older>

DRXPD\_OK(0): The receiving process is successfully terminated.

DRXPD\_NG\_MODE(-2): You don’t open the device file with O\_RDWR flag.

DRXPD\_NG\_BUSY(-3): The receiving process has already been terminated.

DRXPD\_NG(-1): Other failures.

**【Return】**

0 is returned. If an error occurs, -1 is returned in which case errno is set.

**【Errors】**

This command can fail with the following errors.

EINVAL

When you don’t open the device file with O\_RDWR flag.

If you application opened device file with other than O\_RDWR flag,

please open device file with the O\_RDWR flag.

EIO

When this command fails to read or write internal register.

This errno means device driver can't control the device.

If this error occurrs often, it may be hardware error is happend.

Please execute cold-reboot and try again.

EBUSY

When the resource busy.

The device has already stopped receiving data process.

EFAULT

When this command fails to copy the execution result to p\_result,

it sets EFAULT to errno. In this case, please ignore the value of \*p\_result.

Your application may give an invalid address to this pointer,

please check its address.

ETIME(driver version 2.0.3 or later)

When timeout occurs.

Please confirm the description of Synchronization field.

**【Synchronization】**

Synchronous

It works as asynchronous with the driver version 2.0.2 or older.

Timeout

Maximum timeout value is 200ms.

When timeout is occurred, this function returns -1 and sets ETIME to errno.

This function can fail with timeout in following case.

\* DMA transfer does not stop in a certain time.

If timeout occurs, there may be a problem in FPGA logic. Please call RESET\_DFR command with master\_reset bit or reload DRXP driver.

### Command :DRXPD\_GET\_RCVSTS

**【Function】**

This command sets the reception state.

**【Format】**

int ioctl(int fd, DRXPD\_GET\_RCVSTS, ST\_DRXPD\_RCVSTS \*p\_rcv\_staus );

**【Parameter】**

int fd…

Specify the file descriptor.

ST\_DRXPD\_RCVSTS \*p\_rcv\_staus…

Specify the pointer to store the information of ST\_DRXPD\_RCVSTS (see below).

typedef struct drxpd\_rcv\_sts {

int State; //DRXPD\_STOP(0) :Stop

//DRXPD\_RUN(1) :In action

int GroupNum; // Number of Group(16～64[default:16])

int Group[64]; //the host receive ring buffer state(group units).

//DRXPD\_IDLE(0) :Reception completion waiting

//DRXPD\_IDLE\_WR(256) :Under receiving.

//DRXPD\_COMPLETE(1) :Reception completion

// (acquisition waiting for applications )

//DRXPD\_OVERRUN(2) :Overrun

// (Waiting for the completion of the

// processing of the application.)

//DRXPD\_OVERRUN\_WR(258):Overrun ,Under writing process, the

// application shall wait the completion.

}　ST\_DRXPD\_RCVSTS;

**【Return】**

0 is returned. If an error　occurs, -1 is returned in which case errno is set.

**【Errors】**

This command can fail with the following errors.

EINVAL

When you don’t open the device file with O\_RDWR flag.

If you application opened device file with other than O\_RDWR flag,

please open device file with the O\_RDWR flag.

EFAULT

When this command fails to copy the status to p\_rcv\_status, it sets EFAULT to errno.

In this case, please ignore the value of \*p\_rcv\_status.

Your application may give an invalid address to this pointer,

please check its address.

**【Synchronization】**

Synchronous

There is no FPGA access.

**【Remarks】**

ioctl returns an error(-1) if you open the device file with O\_RDWR flag.

This command cannot be used when the current state is in the transmission mode.

### Command :DRXPD\_CLR\_RCV

**【Function】**

This command sets the state of the host receive ring buffer to DRXPD\_IDLE.

**【Format】**

int ioctl(int fd, DRXPD\_CLR\_RCV, int clr\_group);

**【Parameter】**

int fd…

Specify the file descriptor.

int clr\_group…

Specify　0～63.

It specifies a group of the host receive ring buffer. The specified number of the host ring buffer is set to the IDLE state.

**【Return】**

0 is returned. If an error　occurs, -1 is returned in which case errno is set.

**【Errors】**

This command can fail with the following errors.

EINVAL

When you don’t open the device file with O\_RDWR flag.

If you application opened device file with other than O\_RDWR flag,

please open device file with the O\_RDWR flag.

Otherwise, argument(clr\_group) is out of range (0 - 63).

**【Synchronization】**

Synchronous

There is no FPGA access.

**【Description】**

When your application processes a data in a group complete, please change group status

by this command.

### Command :DRXPD SNDSTART

**【Function】**

This command starts the transmission.

**【Format】**

int ioctl(int fd, DRXPD\_SNDSTART, int \*p\_result);

**【Parameter】**

int fd…

Specify the file descriptor.

int \*p\_result…

Specify the pointer to store the result below.

<Device driver version 2.0.3 or later>

Driver always sets DRXPD\_OK(0) to this argument.

This reporting system is not available. Please use an errno to specify the reason.

<Device driver version 2.0.2 or older>

DRXPD\_OK(0): Transmission starts.

DRXPD\_NG\_MODE(-2): You don’t open the device file with O\_RDWR flag.

DRXPD\_NG\_BUSY(-3): Transmission has already started.

DRXPD\_NG\_INIT(-4): It has not been initialized. The initialization is required.

DRXPD\_NG(-1): Other failures.

**【Return】**

0 is returned. If an error occurs, -1 is returned in which case errno is set.

**【Errors】**

This command can fail with the following errors.

EINVAL

When you don’t open the device file with O\_RDWR flag.

If you application opened device file with other than O\_RDONLY flag,

please open device file with the O\_RDONLY flag.

EAGAIN

When this command detects an illegal sequence.

Mmap or ioctl(DRXPD\_INIT) is not called before calling this API.

Please confirm the operation sequence. (**“4.5 Example of transmission”**)

EIO

When this coammnd fails to read or write internal register.

This errno means device driver can't control the device.

If this error occurrs often, it may be hardware error is happend.

Please execute cold-reboot and try again.

EBUSY

When the resource busy.

The device has already started transmitting frames.

**【Errors】**(Continued from the previous page.)

EFAULT

When this command fails to copy the execution result to p\_result, it sets EFAULT to errno.

In this case, please ignore the value of \*p\_result.

Your application may give an invalid address to this pointer,

please check its address.

ETIME (driver version 2.0.3 or later)

When timeout occurs.

Please confirm the description of Synchronization field.

**【Synchronization】**

Synchronous

It works as asynchronous with the driver version 2.0.2 or older.

Timeout

Timeout value is 1ms.

This function checks the status of register after writing data. If it fails to change register status, it returns -1 and sets ETIME to errno. In such a case, there may be a hardware error. Please call RESET\_DFR command with master\_reset bit or reload DRXP driver.

### Command :DRXPD SNDSTOP

**【Function】**

This command stops the transmission.

**【Format】**

int ioctl(int fd, DRXPD\_SNDSTOP, int \*p\_result);

**【Parameter】**

int fd…

Specify the file descriptor.

int \*p\_result…

Specify the pointer to store the result below.

<Device driver version 2.0.3 or later>

Driver always sets DRXPD\_OK(0) to this argument.

This reporting system is not available. Please use an errno to specify the error reason.

<Device driver version 2.0.2 or older>

DRXPD\_OK(0): Transmission has stopped.

DRXPD\_NG\_MODE(-2): You don’t open the device file with O\_RDWR flag.

DRXPD\_NG\_BUSY(-3): Transmission has already stopped.

DRXPD\_NG(-1): Other failures.

**【Return】**

0 is returned. If an error occurs, -1 is returned in which case errno is set.

**【Errors】**

This command can fail with the following errors.

EINVAL

When you don’t open the device file with O\_RDONLY flag.

If you application opened device file with other than O\_RDONLY flag,

please open device file with the O\_RDONLY flag.

EIO

When this command fails to read or write internal register.

This errno means device driver can't control the device.

If this error occurrs often, it may be hardware error is happend.

Please execute cold-reboot and try again.

EBUSY

When the resource busy.

The device has already stopped transmitting frames.

EFAULT

When this command fails to copy the execution result to p\_result, it sets EFAULT to errno.

In this case, please ignore the value of \*p\_result.

Your application may give an invalid address to this pointer,

please check its address.

**【Errors】**(Continued from the previous page.)

ETIME(driver version 2.0.3 or later)

When timeout occurs.

Please confirm the description of Synchronization field.

**【Synchronization】**

Synchronous

It works as asynchronous with the driver version 2.0.2 or older.

Timeout

Timeout value is 1ms.

This function checks the status of register after writing data. If it fails to change register status, it returns -1 and sets ETIME to errno. In such a case, there may be a hardware error. Please call RESET\_DFR command with master\_reset bit or reload DRXP driver.

### Command :DRXPD\_GET\_SNDSTS

**【Function】**

This command gets the transmission state.

**【Format】**

int ioctl(int fd, DRXPD\_GET\_SNDSTS, ST\_DRXPD\_SNDSTS \*p\_snd\_staus );

**【Parameter】**

int fd…

Specify the file descriptor.

ST\_DRXPD\_SNDSTS \*p\_snd\_staus…

Specify the pointer to store the information of ST\_DRXPD\_SNDSTS (see below).

typedef struct drxpd\_snd\_sts {

int State; //DRXPD\_STOP(0) :Stop

//DRXPD\_RUN(1) :In action

}　ST\_DRXPD\_SNDSTS;

**【Return】**

0 is returned. If an error occurs, -1 is returned in which case errno is set.

**【Errors】**

This command can fail with the following errors.

EINVAL

When you don’t open the device file with O\_RDONLY flag.

If you application opened device file with other than O\_RDONLY flag,

please open device file with the O\_RDONLY flag.

EFAULT

When this command fails to copy the status to p\_snd\_status,

it sets EFAULT to errno. In this case, please ignore the value of \*p\_snd\_status.

Your application may give an invalid address to this pointer,

please check the address of it.

**【Synchronization】**

Synchronous

There is no FPGA access.

**【Remarks】**

ioctl returns an error(-1) if you open the device file with O\_RDONLY flag.

### Command :DRXPD\_SET\_SNDSIZ

**【Function】**

This command sets the transmit size.

**【Format】**

int ioctl(int fd, DRXPD\_SET\_SNDSIZ, int size);

**【Parameter】**

int fd…

Specify the file descriptor.

int size…

Specify DRXPD\_SND\_MIN(48)～DRXPD\_SND\_MAX(144,000,000 = 48ms).

Specify in units 48(= 1frame). If the size is not given in units of 48, it is rounded up

a multiple of 48. If it is less than 48, it is set to 48.

The maximum size is 144,000,000. if you set the larger value, the size is set to 144,000,000.

**【Return】**

0 is returned. If an error　occurs, -1 is returned in which case errno is set.

**【Errors】**

This command can fail with the following errors.

EINVAL (driver version 2.0.3 or later)

When you don’t open the device file with O\_RDWR flag.

If you application opened device file with other than O\_RDWR flag,

please open device file with the O\_RDWR flag.

EAGAIN (driver version 2.0.3 or later)

When this command detects an illegal sequence.

Mmap is not called before calling this API.

EBUSY (driver version 2.0.3 or later)

When the resource busy.

The device has already started sending data process.

**【Synchronization】**

Synchronous

There is no FPGA access.

**【Remarks】**

ioctl returns an error(-1) if you open the device file with O\_RDONLY flag.

Groups shall be mapped with mmap command (146,800,640 Bytes including information 48 Bytes and padding bits in total) regardless of the transmit size.

### Command :DRXPD\_GET\_MONSTS

**【Function】**

This command gets the monitoring state.

For more information about monitoring items, See “**6.1 Monitoring Items”**.

**【Format】**

int ioctl(int fd, DRXPD\_GET\_MONSTS, ST\_DRXPD\_MONDAT \*p\_mon\_dat);

**【Parameter】**

int fd…

Specify the file descriptor.

ST\_DRXPD\_MONDAT \*p\_mon\_dat…

Specify the pointer to store the information of ST\_DRXPD\_MONDAT (see below).

typedef struct drxpd\_mon\_dat {

int ItemID; // Monitoring item , set this field in your application .

　　unsigned char RdValue[8]; 　//Read value(8byte).　The driver side writes.

}　ST\_DRXPD\_MONDAT;

**【Return】**

0 is returned. If an error occurs, -1 is returned in which case errno is set.

**【Errors】**

This command can fail with the following errors.

EFAULT

When this command fails to copy the user specified address to kernel space,

it sets EFAULT to errno. Also when it failed to copy the read data to user specified address, it sets EFAULT.

User application may give an invalid address to this pointer,

please check its address.

EINVAL

When invalid ItemID is given.

Others

Each monitoring item fails with its own error code.

For the detail, Please see “**6.1 Monitoring Items”.**

**【Synchronization】**

See “**6.1 Monitoring Items”.**

### Command :DRXPD\_SET\_CONTROL

**【Function】**

This command controls DRXP board functions.

For more information about control items, See “**6.2 Control Items”.**

**【Format】**

int ioctl(int fd, DRXPD\_SET\_CONTROL, ST\_DRXPD\_CONTDAT \*p\_cont\_dat);

**【Parameter】**

int fd…

Specify the file descriptor.

ST\_DRXPD\_CONTDAT \*p\_cont\_dat…

Specify the pointer to store the information of ST\_DRXPD\_CONTDAT (see below).

typedef struct drxpd\_cont\_dat {

int ItemID; // Control item , set this field in your application .

unsigned char WrValue[8]; 　//Write value(8byte),set this field in your application .

}　ST\_DRXPD\_CONTDAT;

**【Return】**

0 is returned. If an error occurs, -1 is returned in which case errno is set.

**【Errors】**

This command can fail with the following errors.

EFAULT

When this control fails to copy the user specified data to kernel space, it sets EFAULT to errno. User application may give an invalid address to this pointer,

please check its address.

EINVAL

When invalid ItemID is given.

Please check the ItemID that application gave.

Others

Each monitoring item fails with its own error code.

For the detail, Please see “**6.2 Control Items”.**

**【Synchronization】**

See “**6.2 Control Items”.**

### Command :DRXPD\_GET\_I2C\_DATA (for Debug)

**【Function】**

This command gets data via I2C bus of SFP+ module.

**【Format】**

int ioctl(int fd, DRXPD\_GET\_I2C\_DATA, ST\_DRXPD\_I2CRD \*p\_rd\_dat);

**【Parameter】**

int fd…

Specify the file descriptor.

ST\_DRXPD\_I2CRD \*p\_rd\_dat…

Specify the pointer to store the information of ST\_DRXPD\_I2CRD” (see below).

typedef struct drxpd\_i2crd\_dat {

char result;

// 0:Success. -1:Failed.　The driver side writes.

　 unsigned char sfp\_no;

//Set the 0～2. set this field in your application.

unsigned char i2c\_address;

//Set the I2C address(0xA0 or 0xA2). set this field in your application.

unsigned char data\_address;

//Set the address. set this field in your application.

unsigned char length;

//Set the length of read(Byte). set this field in your application.

unsigned char RdValue[96];

//Read value(Max:96byte).The driver side writes.

}　ST\_DRXPD\_I2CRD;

**【Return】**

0 is returned. If an error　occurs, -1 is returned in which case errno is set.

**【Errors】**

This command can fail with the following errors.

INVAL

When number of SFP modules is invalid, I2C address is invalid

or read data size is invalid.

Please confirm the description of Parameter field.

EIO

When this command fails to read or write internal register.

This errno means device driver can't control the device.

If this error occurrs often, it may be hardware error is happend.

Please execute cold-reboot and try again.

EFAULT

When this command fails to copy the data to p\_rd\_dat, it sets EFAULT to errno.

In this case, please ignore the value of \*p\_rd\_dat.

Your application may give an invalid address to this pointer,

please check its address.

EREMOTEIO(driver version 2.0.3 or later)

I2c operation is completed. But, i2c timeout or NACK is occurred.

ETIME(driver version 2.0.3 or later)

When timeout occurs.

Please confirm the description of Synchronization field.

**【Synchronization】**

Synchronous

Timeout

Maximum Timeout value is 2000ms.

I2c operation is not completed in this time.

Please confirm a state of SFP+ module.

Also, please confirm the SFP+ module’s I2c specification that is inserted to a DRXP

board.

**【Remarks】**

I2C detail conforms to the SFF-8472 specification.

Note that this API is blocked until the I2C process is being done.

### Command :DRXPD\_SET\_I2C\_CMD (for Debug)

**【Function】**

This command sends command via I2C bus of SFP+ module.

**【Format】**

int ioctl(int fd, DRXPD\_SET\_I2C\_CMD, ST\_DRXPD\_I2CCMD \*p\_cmd);

**【Parameter】**

int fd…

Specify the file descriptor.

ST\_DRXPD\_I2CCMD \*p\_cmd…

Specify the pointer to store the information of ST\_DRXPD\_I2CCMD” (see below).

typedef struct drxpd\_i2c\_cmd {

char result;

// 0:Success. -1:Failed.　The driver side writes.

　 unsigned char sfp\_no;

//Set the 0～2. set this field in your application.

unsigned char i2c\_address;

//Set the I2C address(0xA0 or 0xA2). set this field in your application.

unsigned char data\_address;

//Set the address. set this field in your application.

unsigned char length;

//Set the length of write(Byte). set this field in your application.

unsigned char WrValue[96];

//Write value(Max:96byte). set this field in your application.

}　ST\_DRXPD\_I2CCMD;

**【Return】**

0 is returned. If an error occurs, -1 is returned in which case errno is set.

**【Errors】**

This command can fail with the following errors.

EINVAL

When number of SFP modules is invalid ,

I2C address is invalid or write data size is invalid.

Please confirm the description of Parameter field.

EIO

When this command fails to read or write internal register.

This errno means device driver can't control the device.

If this error occurrs often, it may be hardware error is happend.

Please execute cold-reboot and try again.

EFAULT

When this command fails to copy the command information (\*p\_cmd pointed) from a userspace, it sets EFAULT to errno.

Your application may give an invalid address to this pointer, please check its address.

**【Errors】**(Continued from the previous page.)

EREMOTEIO(driver version 2.0.3 or later)

I2c operation is completed. But, i2c timeout or NACK is occurred.

ETIME(driver version 2.0.3 or later)

When timeout occurs.

Please confirm the description of Synchronization field.

**【Synchronization】**

Synchronous

Timeout

Maximum Timeout value is 2000ms.

I2c operation is not completed in this time.

Please confirm a state of SFP+ module.

Also, please confirm the SFP+ module’s I2c specification that is inserted to a DRXP

board.

**【Remarks】**

I2C detail conforms to the SFF-8472 specification.

Note that this API is blocked until the I2C process is being done.

(Caution!)

The SFP+ modules selected for Prototype/Production DRXPs (Sumitomo Electric SPP5200ER-GL) do not allow the users to write any values via I2C. This command may be useful when the SFP+ modules are replaced by different models. However, such SFP+ modules are not supported and this command may not work as intended.

### Command :DRXPD\_SET\_RCVGROUPS

**【Function】**

This command changes the number of groups in Host Data buffer (Receive Ring Buffer).The nominal value is 16.

This command was defined as a command “DRXPD\_RCVGROUPS” with the driver version 2.0.2 or older. The device driver version 2.0.3 or later maintain the old command identifier. Your application can also call this command with old identifier.

**【Format】**

int ioctl(int fd, DRXPD\_SET\_RCVGROUPS, int group\_num);

**【Parameter】**

int fd…

Specify the file descriptor.

int group\_num…

Specify　16, 32 or 64.

**【Return】**

0 is returned. If an error occurs, -1 is returned in which case errno is set.

**【Errors】**

This command can fail with the following errors.

EINVAL

When you don’t open the device file with O\_RDONLY flag.

If you application opened device file with other than O\_RDONLY flag,

please open device file with the O\_RDONLY flag.

Otherwise, argument(group\_num) is invalid.

EAGAIN

When this command detects an illegal sequence.

Please call this API before calling mmap systemcall and ioctl(DRXPD\_INIT).

Please confirm the operation sequence. (**“4.4 Example of reception”**)

**【Synchronization】**

Synchronous

There is no FPGA access.

**【Description】**

　 　 Your application can use this command to increase Group of Host Data buffer. (Receive Ring Buffer).

**【Remarks】**

“ioctl” returns an error(-1) if you open the device file with O\_RDWR flag.

### Command :DRXPD\_SET\_SEUEVT

**【Function】**

　　This command registers evnetfd on production/prototype DRXP board. When SEU function module encounters error state, driver notifies the event through registered eventfd. For distinguishing the event of 48msec TE/data received please use different eventfd from registered by ioctl(DRXPD\_SET\_RCVEVT).

**【Format】**

int ioctl(int fd, DRXPD\_SET\_SEUEVT, int eventfd);

**【Parameter】**

int fd…

Specify the file descriptor.

int eventfd…

Specify the event file descriptor gained by calling “int eventfd(unsigned int initval, int flags)”. eventfd should not use EFD\_SEMAPHORE. The resource related to eventfd in the device driver is released when user process is finished.

**【Return】**

0 is returned.　if an error　occurs, -1 is returned in which case errno is set.

**【Errors】**

This command can fail with the following errors.

EAGAIN

When this command detects an illegal sequence.

(An eventfd for monitoring SEU module is already set up.).

EBADF

The eventfd your application gives to driver is invalid file descriptor.

Please confirm whether the eventfd file descriptor is valid or not.

EINVAL

The eventfd your application gives to driver is not an eventfd.

Please confirm whether the eventfd file descriptor is valid or not.

**【Synchronization】**

Synchronous

There is no FPGA access.

**【Description】**

If your application needs the notification of SEU function module’s error, please use this command. If SEU function module detects module’s error state, driver generates notification.

SEU (Single Event Upset) means a soft-error caused by charged particle. For example, cosmic ray rarely causes the bit-flip in a memory. The SEU function module implemented to the DRXP board monitors bit-flip event in the own FPGA’s configuration data. For the detail of SEU function module, please refer **“7.3 SEU Mitigation”**.

EVENT LIST

* SEU function module transition to Error state.
* SEU function module detects following error occured.
* heartbeat error
* watchdog timer error

For detail, please refer to **“6.1.14 GET\_SEU\_STATUS”**.

Note: This function only notifies the event when detect changing point.

* Correctable error count is incremented
* Uncorrectable error count is incremented

The resource acquired by driver in this function is released when user application calls 5.2 close, 5.5.19 Command :DRXPD\_CLR\_SEUEVT, or user application is terminated by error. Even if user application is terminated by error (e.g. SEGMENTATION FAULT), this resource is released.

**【Remarks】**

Correctable error and Uncorrectable error can’t be detected with the driver version before 2.0.2.

### Command :DRXPD\_CLR\_SEUEVT

**【Function】**

This command unregisters evnetfd on production/prototype DRXP board.

**【Format】**

int ioctl(int fd, DRXPD\_CLR\_SEUEVT, int dummy);

**【Parameter】**

int fd…

Specify the file descriptor.

int dummy…

This function doesn’t use this variable.

**【Return】**

0 is returned. If an error occurs, -1 is returned in which case errno is set.

**【Errors】**

This command can fail with the following errors.

EAGAIN

This device driver doesn’t have eventfd resource.

**【Synchronization】**

Synchronous

There is no FPGA access.

**【Description】**

If your evnetfd for the SEU event is no longer needed, please unregister eventfd on production/prototype DRXP board. About the detail of SEU function module, please refer **“7.3 SEU Mitigation”**.

### Command :DRXPD\_FLASH\_OVHLOG

**【Function】**

This command reads the overheat protection log from on board EEPROM and prints them to syslog.

This command prints all log entries in the EEPROM. And the log is not changed/deleted by calling this command. Therefore, if you call this command twice, the same log message is printed to syslog.

**【Format】**

int ioctl(int fd, DRXPD\_FLASH\_OVHLOG, int dummy);

**【Parameter】**

int fd…

Specify the file descriptor.

int dummy…

This function doesn’t use this variable.

**【Return】**

0 is returned. If an error occurs, -1 is returned in which case errno is set.

**【Errors】**

This command can fail with the following errors.

ENOSYS(version 2.0.3 or later)

When this command is called on prototype DRXP board,

it fails and sets ENOSYS to errno.

In the case of driver version 2.0.2 or older, sets EINVAL to errno.

EIO(version 2.0.3 or later)

When this command fails to read or write internal register.

This errno means device driver can't control the device.

If this error occurs often, it may be hardware error is happend.

Please execute cold-reboot and try again.

EREMOTEIO(driver version 2.0.3 or later)

I2c operation is completed. But, i2c timeout or NACK is occurred.

ETIME(driver version 2.0.3 or later)

When timeout occurs.

Please confirm the description of Synchronization field.

**【Synchronization】**

Synchronous

Timeout

Maximum Timeout value is 30s.

I2c operation is not completed in this time.

Please call RESET\_DFR command with master\_reset but or reload DRXP driver.

After that, please try again. If this error occurs often, there is a possibility that an error happens in onboard EEPROM.

**【Description】**

Please refer to the Appendix **“7.4 The overheating protection”** for more details.

### Command :DRXPD\_SEL\_EQ

**【Function】**

User application can select the equalizer type of FPGA transceiver by this command.

The type of FPGA transceiver equalizer is related to the Bit Error Rate of received data. There are two available equalizer types, LPM and DFE. The LPM equalizer equalizes received data by using linear equalizer. The DFE equalizer equalizes received data by using linear equalizer and filter. In the case of DRXP board, we recommend you should use LPM.

**【Format】**

int ioctl(int fd, DRXPD\_SEL\_EQ, int i\_eq\_type);

**【Parameter】**

int fd…

Specify the file descriptor.

int i\_eq\_type…

Specify the equalizer type.

Available values are as below. And these values are defined in the drxpd.h.

(When user calls open system call, the type of equalizer is reset to LPM equalizer.)

- DRXPD\_SEL\_EQ\_LPM

Select the LPM equalizer.

- DRXPD\_SEL\_EQ\_DFE

Select the DFE equalizer.

**【Return】**

0 is returned. If an error occurs, -1 is returned in which case errno is set.

**【Errors】**

This command can fail with the following errors.

EINVAL

When this command is called with invalid argument,

it fails and sets EINVAL to errno.

EIO

When this command fails to read or write internal register.

This errno means device driver can't control the device.

If this error occurrs often, it may be hardware error is happend.

Please execute cold-reboot and try again.

ETIME(driver version 2.0.3 or later)

When this command fails to change equalizer status.

Please confirm the description of Synchronization field.

**【Synchronization】**

Synchronous

It works as asynchronous with the driver version 2.0.2 or older.

Timeout

Timeout value is 10ms.

This function checks the status of register after writing data. If it fails to change register status, it returns -1 and sets ETIME to errno. In such a case, there may be a hardware error. Please call RESET\_DFR command with master\_reset bit or reload DRXP driver.

**【Remarks】**

■Remarks for the prototype DRXP board

This command is available with driver version 1.0.6 or later.

And it needs FPGA version 7524B073 or later.

■Remarks for the production DRXP board

This command is available with driver version 2.0.4 or later.

And it needs FPGA version 52291486 or later.

### Command :DRXPD\_GET\_EQ

**【Function】**

User application can get the current equalizer type of FPGA transceiver by this command.

**【Format】**

int ioctl(int fd, DRXPD\_GET \_EQ, int \*i\_eq\_type);

**【Parameter】**

int fd…

Specify the file descriptor.

int \*i\_eq\_type…

When this command is performed successfully, it stores a current equalizer type.

Following values are set by driver. And these values are defined in the drxpd.h.

- DRXPD\_SEL\_EQ\_LPM

Current equalizer type is LPM equalizer.

- DRXPD\_SEL\_EQ\_DFE

Current equalizer type is DFE equalizer.

**【Return】**

0 is returned. If an error occurs, -1 is returned in which case errno is set.

**【Errors】**

This command can fail with the following errors.

EFAULT(driver version 2.0.3 or later)

When this command fails to copy the type of equalizer to i\_eq\_type,

it sets EFAULT to errno. In this case, please ignore the value of \*i\_eq\_type.

Your application may give an invalid address to this pointer,

please check its address.

EAGAIN

The register value for controlling equalizer is invalid value.

Please try to set equalizer type by ioctl(DRXPD\_SEL\_EQ).

EIO

When this command fails to read or write internal register.

This errno means device driver can't control the device.

If this error occurrs often, it may be hardware error is happend.

Please execute cold-reboot and try again.

**【Synchronization】**

Synchronous

Timeout

This command just reads a FPGA’s internal register. If there is an error in the physical layer, it sets EIO to errno. For the timeout value of physical layer, please refer a section **“5.5 ioctl”.**

**【Remarks】**

■Remarks for the prototype DRXP board

This command is available with driver version 1.0.6 or later.

And it needs FPGA version 7524B073 or later.

■Remarks for the production DRXP board

This command is available with driver version 2.0.4 or later.

And it needs FPGA version 52291486 or later.

### Command :DRXPD\_GET\_OVRFLW

**【Function】**

User application can get the overflow flag. This flag represents the overflow status of the Device Ring Buffer for receiving data. This value indicates whether the overflow event has occurred between the last time you called this function (or the time you opened device file) and the time current call.

**【Format】**

int ioctl(int fd, DRXPD\_GET\_OVRFLW, int \*i\_flag);

**【Parameter】**

int fd…

Specify the file descriptor.

int \*i\_flag…

When this command is performed successfully, it stores a current value.

Following values are set by driver. And these values are defined in the drxpd.h.

- DRXPD\_DRB\_OVRFLW

The Device Ring Buffer is overflow.

- DRXPD\_DRB\_NORMAL

The Device Ring Buffer is not in overflow state.

**【Return】**

0 is returned. If an error occurs, -1 is returned in which case errno is set.

**【Errors】**

This command can fail with the following errors.

EFAULT

When this command fails to copy the status of Device Ring Buffer,

it sets EFAULT to errno. In this case, please ignore the value of \*i\_flag.

Your application may give an invalid address to this pointer,

please check its address.

EIO

When this command fails to read or write internal register.

This errno means device driver can't control the device.

If this error occurrs often, it may be hardware error is happend.

Please execute cold-reboot and try again.

**【Synchronization】**

Synchronous

Timeout

This command just reads a FPGA’s internal register. If there is an error in the physical layer, it sets EIO to errno. For the timeout value of physical layer, please refer a section **“5.5 ioctl”.**

**【Remarks】**

This function is available with device driver version 2.0.8 or later. And it needs a newer FPGA version than “52291486”. (In the case of prototype DRXP, it needs a newer FPGA than 7524B073).

# Monitoring and Control Items

ItemIDs are defined in drxpd.h.

## Monitoring Items

This section describes the monitoring items that are used in the ioctl(DRXPD\_GET\_MONSTS).

All of data in each item are cleared when device is probed (DRXP driver module is loaded to kernel).

Some monitoring items keep monitoring status even if your application doesn’t open device file. And also, if these monitoring item’s registers are latched value (\*1), your application can know the status while the device file is closed. When your application calls those monitoring items, the returned register value means that the data that latched event is detected or not after your application clears registers (or device driver is loaded). Users can distinguish whether the data type is latched data or real-time value by seeing the "Latch" field in each monitoring Item’s subsection. For the detail behavior of each monitoring item, please refer to the following subsections.

Each monitoring item is categorized into the following 5 levels. Users can confirm monitoring item’s category by seeing the "Category" field in each monitoring item’s subsection.

|  |  |
| --- | --- |
| Level | Description |
| Critical | The failure may cause a damage of hardware.  The hardware and/or the driver stop some functions or shutdown machine without an adequate notification to the software. |
| Fatal | The failure may cause a damage of hardware.  The hardware and/or the driver may stop some function or shutdown machine. |
| Error | The failure should not cause any damage of hardware.  The hardware and/or the device driver do not work correctly. The data in the host ring buffer may not be correct. |
| Warning | The failure should not cause any damage of hardware.  The hardware and/or the device driver does not work correctly. The data in the host ring buffer should be correct. |
| Info | Others than those above categories. |

Table 6‑1: The monitoring item categories

(\*1) Latched value keeps the last status/value until a user application clears its status/value by calling reset.

### GET\_ALARM\_STATUS\_REG

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| ItemID | | GET\_ALM\_STS\_REG\_ID\_D | 0x0000 3100 | CH1-BitD |
| ItemID | | GET\_ALM\_STS\_REG\_ID\_C | 0x0000 3200 | CH2-BitC |
| ItemID | | GET\_ALM\_STS\_REG\_ID\_B | 0x0000 3300 | CH3-BitB |
| Description | | Read the Alarm Status Register from SFP+ module. Each bit is 1 for alarm and 0 for normal. RESET\_DFR resets to 0. | | |
| Update Rate | | 192ms or less. The update is not synchronized with 48ms TE. | | |
| TE Related | | No | | |
| Synchronization | | Synchronous (\*1) | | |
| Data | Size | 8Bytes | | |
| Reset | RESET\_DFR | | |
| Latch | All bits in Byte0 are latched value. Bit0 in Byte1 is set to 1 (alarm) when the access of the I2C interface to SFP+ module is failed. And Bit0 in Byte1 is set to 0 (normal) when it is succeeded. | | |
| Value | Byte0:  bit 7: TempWARN(Temperature warning active)  bit 6: TxBIASWARN(Output bias current warning active)  bit 5: TxPOWWARN(Output power warning active)  bit 4: RxPOWWARN (Rx input power warning active)  bit 3: TempALM(Temperature alarm active)  bit 2: TxBIASALM(Output bias current alarm active)  bit 1: TxPOWALM(Output power alarm active)  bit 0: RxPOWALM (Rx input power alarm active)  Byte1:  bit 7-1: Reserved (undefined)  bit 0: I2C Access Error  Byte2-7: Reserved (undefined) | | |
| Category | ■Warning  TempWARN, TxBIASWARN, TxPOWWARN, RxPOWWARN  In general, these monitoring items are hardware warning signal of SFP+ module. Please confirm SFP+ module status. Device driver just reads status registers in the SFP+ module. For the detail, please see SFP+ module datasheet or ask SFP+ module vendor.  ■Fatal  TempALM, TxBIASALM, TxPOWALM, RxPOWALM  In general, these monitoring items may cause a hardware damage of SFP+ module. Please confirm SFP+ module status. Device driver just reads status registers in the SFP+ module. For the detail, please see SFP+ module datasheet or ask SFP+ module vendor.  ■Fatal/Info  I2C Access Error  Fatal:  Device driver (DRXP) fails to access internal registers in the SFP+ module.  Even though SFP+ module is installed, DRXP fails to access. It may mean hardware error of SFP+ module or DRXP. Or, DRXP just can’t access currently installed SFP+ module. DRXP is tested with Sumitomo electric SPP5200ER-GL. If you use other vendor’s module, it may not be able to access to SFP+ module register correctly.  Info:  The SFP+ module isn’t installed. | | |
| Remarks | | The DRXP driver just tries to read "Alarm and Warning Flags" area specified by SFF-8472 from SFP+ module. The SFP+ module vendor presets thresholds of these flags. For the detail meaning of these flags, please confirm SFP+ module's datasheets currently used and SFF-8472. The information of the absolute maximum raitings and the recommended operating condition in the datasheets helps to interpret these bits. In the case of the SFP+, Sumitomo electric SPP5200ER-GL, please refer to **“7.1 SFP module warning and alarm threshold”** about each　alarm　threshold.  When I2C Access Error occurs, all bits of Byte0 are set to ‘0’.  (\*1) The driver starts monitoring these alarms when it is loaded. It keeps monitoring them periodically even when the device file is not opened. Once an alarm condition is met, the corresponding alarm bit remains active until it is explicitly reset by calling RESET\_DFR with bit0 = 1. The DRXP driver keeps this latched register value include latest monitoring data. This command just copies this data to user memory. Therefore, there is no specific timeout value.  In driver 1.0.1 or older, these statuses are monitored only when device is opened.  In driver 2.0.2 or older, these status is masked following rules.  - In the case that device file is opened with O\_RDONLY,  bit 6, 5, 2 and 1 is always 0.  - In the case that device file is opened with O\_RDWR flag,  bit 4 and 0 is always 0. | | |
| Errors | | EAGAIN  When I2C Access Error occurred. | | |

### GET\_POWER\_ALARM\_REG

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| ItemID | | GET\_POW\_ALM\_REG\_ID\_D | 0x0000 3101 | CH1-BitD |
| ItemID | | GET\_POW\_ALM\_REG\_ID\_C | 0x0000 3201 | CH2-BitC |
| ItemID | | GET\_POW\_ALM\_REG\_ID\_B | 0x0000 3301 | CH3-BitB |
| Description | | Read the Power Supply Alarm Status Register from SFP+ module. Each bit is 1 for alarm and 0 for normal. RESET\_DFR resets to 0. | | |
| Update Rate | | 192ms or less. The update is not synchronized with 48ms TE. | | |
| TE Related | | No | | |
| Synchronization | | Synchronous (\*1) | | |
| Data | Size | 8Bytes | | |
| Reset | RESET\_DFR | | |
| Latch | Bit1 and bit0 in Byte0 are latched values. Bit0 in Byte1 is set to 1 (alarm) when the access of the I2C interface to SFP+ module is failed. And Bit0 in Byte1 is set to 0 (normal) when it is succeeded. | | |
| Value | Byte0:  bit 7-2: Reserved (undefined)  bit 1: 3.3V\_WARN  bit 0: 3.3V\_ALM  Byte1:  bit 7-1: Reserved (undefined)  bit 0: I2C Access Error  Byte2-7: Reserved (undefined) | | |
| Category | ■Warning  3.3V\_WARN  In general, this monitoring item means power warning of SFP+ module. Device driver just reads status registers in the SFP+ module. For the detail, please see SFP+ module datasheet or ask SFP+ module’s vendor.  ■Fatal  3.3V\_ALM  Device driver just reads status registers in the SFP+ module. For the detail, please see SFP+ module datasheet or ask SFP+ module’s vendor.  This monitoring item may mean hardware error of SFP+ module’s power. The 3.3V power is supplied by on-board power circuit. Therefore, please also confirm onboard power circuit status. For the detail, please see a “6.1.5 GET\_DFR\_VOLTAGE\_STATUS”.  ■Fatal/Info  I2C Access Error  Fatal:  Device driver (DRXP) fails to access internal registers in the SFP+ module  Even though SFP+ module is installed, DRXP fails to access. It may mean hardware error of SFP+ module or DRXP. Or, DRXP just can’t access currently installed SFP+ module. DRXP is tested with Sumitomo electric SPP5200ER-GL. If you use other vendor’s module, it may not be able to access to SFP+ module register.  Info:  The SFP+ module isn’t installed. | | |
| Remarks | | Refer to **“7.1 SFP module warning and alarm threshold”** about each alarm　threshold.  When I2C Access Error occurs, all bit of Byte0 is set to‘0’.  (\*1) The driver starts monitoring these alarms when it is loaded. It keeps monitoring them periodically even when the device file is not opened. Once an alarm condition is met, the corresponding alarm bit remains active until it is explicitly reset by calling RESET\_DFR with bit0 = 1. The DRXP driver keeps this latched register value include latest monitoring data. This command just copies this data to user memory. Therefore, there is no specific timeout value.  In driver 1.0.1 or older, these statuses are monitored only when device is opened  In driver 2.0.2 or older, this status should be cleared after power-up by using RESET\_DFR with register reset flag. | | |
| Errors | | EAGAIN  When I2C Access Error occurred. | | |

### GET\_SIGNAL\_AVG\_nW

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| ItemID | | GET\_SIG\_AVG\_ID\_D | 0x0000 3102 | CH1-BitD |
| ItemID | | GET\_SIG\_AVG\_ID\_C | 0x0000 3202 | CH2-BitC |
| ItemID | | GET\_SIG\_AVG\_ID\_B | 0x0000 3302 | CH3-BitB |
| Description | | Read the Receiver Signal Monitor Average Optical Power from SFP+ module. This returns the average received optical power in unit of nW. | | |
| Update Rate | | 192ms or less. The update is not synchronized with 48ms TE. | | |
| TE Related | | No | | |
| Synchronization | | Synchronous (\*1) | | |
| Data | Size | 8Bytes | | |
| Reset | RESET\_DFR | | |
| Latch | Byte0-2 are set when the DRXP board accesses SFP+ module by I2C interface. Bit0 in Byte3 is set to 1 (alarm) when the access of the I2C interface to SFP+ module is failed. And Bit0 in Byte3 is set to 0 (normal) when it is succeeded. | | |
| Value | Byte0-2:  24-bit integer, MSByte first（0-6553500nW）  Byte3:  bit 7-1: Reserved (undefined)  bit 0: I2C Access Error  Byte4-7: Reserved (undefined) | | |
| Category | ■Info  Optical Power.  Device driver just reads registers in the SFP+ module. For the detail, please see SFP+ module datasheet or ask SFP+ vendor.  ■Fatal/Info  I2C Access Error  Fatal:  Device driver (DRXP) fails to access internal registers in the SFP+ module  Even though SFP+ module is installed, DRXP fails to access. It may mean hardware error of SFP+ module or DRXP. Or, DRXP just can’t access currently installed SFP+ module. DRXP is tested with Sumitomo electric SPP5200ER-GL. If you use other vendor’s module, it may not be able to access to SFP+ module register.  Info:  The SFP+ module isn’t installed. | | |
| Remarks | | When I2C Access Error occurs, all bit of Byte0 -2 is set to ‘0’.  (\*1) After driver detecting production/prototype DRXP boards, it keeps monitoring these statuses periodically. If device isn’t opened, driver keeps monitoring them. The DRXP driver keeps this latched register value include latest monitoring data. This command just copies this data to user memory. Therefore, there is no specific timeout value.  In driver 1.0.1 or older, these statuses are monitored only when device is opened | | |
| Errors | | EAGAIN  When I2C Access Error occurred. | | |

### GET\_DFR\_STATUS

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| ItemID | | GET\_DFR\_STS\_ID | 0x0000 3801 |  |
| Description | | Read the status. Each bit is 1 for alarm and 0 for normal. The alarm status is latched until it is cleared by RESET\_DFR (with RST\_DFR\_ID\_ALL only). | | |
| Update Rate | | Real time  (FPGA updates these items in real time. Every time this command is called, it reads values from FPGA.) | | |
| TE Related | | No | | |
| Synchronization | | Synchronous (\*1) | | |
| Data | Size | 8Bytes | | |
| Reset | RESET\_DFR | | |
| Latch | All bits are latched | | |
| Value | Byte0:  bit 7-3: Not used (undefined)  bit 2: DDR clock (200 MHz) PLL 1 alarm (\*2)  bit 1: DDR clock (200 MHz) PLL 2 alarm (\*2)  bit 0: Reference clock (125 MHz) PLL alarm  Byte1:  bit 7-3: Not used (undefined)  bit 2: Internal data input clock PLL alarm (CH3-BitB)  bit 1: Internal data input clock PLL alarm (CH2-BitC)  bit 0: Internal data input clock PLL alarm (CH1-BitD)  Byte2:  bit 7-3: Not used (undefined)  bit 2: Sync error detected(CH3-BitB)  bit 1: Sync error detected(CH2-BitC)  bit 0: Sync error detected(CH1-BitD)  Byte3-7: Reserved (undefined) | | |
| Category | ■Fatal  DDR clock (200 MHz) PLL 1/2 alarm  Reference clock (125 MHz) PLL alarm  Internal data input clock PLL alarm (CH3/2/1-BitB/C/D)  These statuses may cause/mean hardware error.  If these alarms occur repeatedly, it may be a malfunction.  ■Error  Sync error detected(CH3/2/1-BitB/C/D)  These statuses are ALMA de-formatter’s sync status. Please confirm signal integrity and data source. | | |
| Remarks | | "Internal data input clock alarm" will be set to 0 (normal) when the reference clock on the drxp board is properly supplied. This information is not related to the input of data.  The PLL statuses are always monitored and updated, even if the driver is not loaded or the device is not opened. The sync error statuses are monitored and updated only when driver is opened.  (\*1) This command just reads a FPGA’s internal register. If there is an error in the physical layer, it sets EIO to errno. For the timeout value of physical layer, please refer a section **“5.5 ioctl”.**  (\*2)  When using with production DRXP, these values sometimes are meaningless. When the user application calls this function from IF1, DDR clock (200 MHz) PLL 2 alarm is pointless. Also, it calls this function from IF2. DDR clock (200 MHz) PLL 1 alarm is meaningless. The device driver 2.0.7 or older sometimes sets alarm status when alarm status does not occur. Please ignore the meaningless value. In the device driver 2.0.8 or later, it sets 0 to the meaningless value. | | |
| Errors | | EIO  When this command fails to read or write internal register.  This errno means device driver can't control the device.  If this error occurs often, it may be hardware error is happened.  Please execute cold-reboot and try again. | | |

### GET\_DFR\_VOLTAGE\_STATUS

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| ItemID | | GET\_DFR\_VOL\_STS\_ID | 0x0000 3802 |  |
| Description | | Read the voltage status.  Each bit is 1 for alarm and 0 for normal. The alarm status is latched until it is cleared by RESET\_DFR with register reset flag. All bits are set to 0 after RESET\_DFR if there are no errors. Otherwise, the operator should be notified. | | |
| Update Rate | | Real time  (FPGA updates these items in real time. Every time this command is called, it reads values from FPGA.) | | |
| TE Related | | No | | |
| Synchronization | | Synchronous (\*1) | | |
| Data | Size | 8Bytes | | |
| Reset | RESET\_DFR | | |
| Latch | All bits are latched | | |
| Value | Byte0:  bit 7: Not used (undefined)  bit 6: +0.95V(for FPGA core)(prototype)/+0.85V(production)  bit 5: +1.8V (for FPGA)  bit 4: +1.0VMGTAVCC (for FPGA)/+0.9VMGTAVCC(production)  bit 3: +1.2VMGTAVTT(for FPGA)  bit 2: +1.8VMGTVCCAUX(for FPGA)  bit 1: +3.3V(for FPGA-IO)  bit 0: +1.35VDDR(for DDR)  Byte1-7: Reserved (undefined) | | |
| Category | ■Fatal  +0.95V(for FPGA core)(prototype)/+0.85V(production)  +1.8V (for FPGA)  +1.0VMGTAVCC (for FPGA)/+0.9VMGTAVCC(production)  +1.2VMGTAVTT(for FPGA)  +1.8VMGTVCCAUX(for FPGA)  +3.3V(for FPGA-IO)  +1.35VDDR(for DDR)  These statuses may cause/mean hardware error.  If these alarms occur repeatedly, it may be a malfunction of on-board power circuit, server’s power supply, or power detection circuit.  It may cause hardware damage, we recommend stop using hardware. | | |
| Remarks | | The statuses are always monitored and updated, even if the driver is not loaded or the device is not opened.  The criteria of voltage alarms is ±3% for VCCINT, ±5% for others.  These statuses should be cleard after power-up by using RESET\_DFR with register reset flag with the driver version 2.0.2 or older.  (\*1) This command just reads a FPGA’s internal register. If there is an error in the physical layer, it sets EIO to errno. For the timeout value of physical layer, please refer a section **“5.5 ioctl”.** | | |
| Errors | | EIO  When this command fails to read or write internal register.  This errno means device driver can't control the device.  If this error occurs often, it may be hardware error is happened.  Please execute cold-reboot and try again. | | |

### GET\_DFR\_CHKSUM\_COUNTER

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| ItemID | | GET\_DFR\_CHK\_CNT\_ID\_D | | 0x0000 1805 | CH1-BitD |
| ItemID | | GET\_DFR\_CHK\_CNT\_ID\_C | | 0x0000 2805 | CH2-BitC |
| ItemID | | GET\_DFR\_CHK\_CNT\_ID\_B | | 0x0000 3805 | CH3-BitB |
| Description | | Read the internal checksum counter. It counts the number of checksum errors of ALMA frame after the reset.  RESET\_DFR or RESET\_DFR\_CHKSUM resets to 0. | | | |
| Update Rate | | Real time  (FPGA updates these items in real time. Every time this command is called, it reads values from FPGA.) | | | |
| TE Related | | No | | | |
| Synchronization | | | Synchronous (\*1) | | |
| Data | Size | | 8Bytes | | |
| Reset | | RESET\_DFR or RESET\_DFR\_CHKSUM | | |
| Latch | | All bits are real time value | | |
| Value | | Byte 0-7: Checksum counter value (Unsigned, MSByte first) | | |
| Category | | ■Error  　Checksum counter value  If this value is increasing, it means some received ALMA frames have an error. Please confirm signal integrity and data source. | | |
| Remarks | | Once FPGA stage 2 configuration is completed, the FPGA starts receiving frames and continues checking the correctness of the checksum of every frame all the time. The counter in the FPGA is incremented every time a frame has an incorrect checksum regardless of whether the device driver is loaded.  The driver version 1.0.1 or older clears these values, when user opens device.  (\*1) This command just reads a FPGA’s internal register. If there is an error in the physical layer, it sets EIO to errno. For the timeout value of physical layer, please refer a section **“5.5 ioctl”.** | | | |
| Errors | | EIO  When this command fails to read or write internal register.  This errno means device driver can't control the device.  If this error occurs often, it may be hardware error is happened.  Please execute cold-reboot and try again. | | | |

### GET\_DFR\_SYNC\_ERR\_CNT

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| ItemID | | GET\_DFR\_SYNC\_ERR\_CNT\_ID\_D | 0x0000 1D01 | CH1-BitD |
| ItemID | | GET\_DFR\_SYNC\_ERR\_CNT\_ID\_C | 0x0000 2D01 | CH2-BitC |
| ItemID | | GET\_DFR\_SYNC\_ERR\_CNT\_ID\_B | 0x0000 3D01 | CH3-BitB |
| Description | | Read the internal sync error counter (SYNC\_ERR\_CNT).  If sync is not found within 160 frames captured from incoming data, SYNC\_ERR\_CNT counts up. This indicates that sync has not been found anywhere within 160-bit frame.  RESET\_DFR with register reset flag or RESET\_DFR\_SYNC resets to 0. | | |
| Update Rate | | Real time  (FPGA updates these items in real time. Every time this command is called, it reads values from FPGA.) | | |
| TE Related | | No | | |
| Synchronization | | Synchronous (\*1) | | |
| Data | Size | 8Bytes | | |
| Reset | RESET\_DFR or RESET\_DFR\_SYNC | | |
| Latch | All bits are real time value | | |
| Value | Byte0-3: Unsigned, MSByte first.  Byte4-7: Reserved (undefined) | | |
| Category | ■Error  Unsigned, MSByte first.  If this value is increasing, it means DRXP fails to find sync-pattern from received ALMA frames. If DRXP can’t find sync-pattern correctly, it can’t transfer correct data to host ring buffer. Please confirm signal integrity and data source. | | |
| Remarks | | Once FPGA stage 2 configuration is completed, the FPGA starts receiving frames and continues checking frame synchronization. The counter in the FPGA is incremented every time FPGA fails to find the sync pattern within 160 frames regardless of whether the device driver is loaded.  In driver version 1.0.1 or older, driver clears these values, when user opens device.  (\*1) This command just reads a FPGA’s internal register. If there is an error in the physical layer, it sets EIO to errno. For the timeout value of physical layer, please refer a section **“5.5 ioctl”.** | | |
| Errors | | EIO  When this command fails to read or write internal register.  This errno means device driver can't control the device.  If this error occurs often, it may be hardware error is happened.  Please execute cold-reboot and try again. | | |

### GET\_DFR\_SYNC\_STATUS

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| ItemID | | GET\_DFR\_SYNC\_STS\_ID\_D | 0x0000 1D03 | CH1-BitD |
| ItemID | | GET\_DFR\_SYNC\_STS\_ID\_C | 0x0000 2D03 | CH2-BitC |
| ItemID | | GET\_DFR\_SYNC\_STS\_ID\_B | 0x0000 3D03 | CH3-BitB |
| Description | | Read the internal sync status.  Reports the sync lock status and the number of searched frames before sync is established (or re-established) as the offset. If a suitable signal is being received on the fiber optical receiver, this shouldn’t take longer than 160 frames. However, if a suitable signal is not present, the search continues and this offset value continues being incremented. This offset value can be used in conjunction with RESET\_DFR\_SYNC for verifying the performance of the search algorithm. When device file is opened, these are resets to 0.  The frame sync state machine of ALMA frame de-formatter has three stages, search stage, check stage and monitor stage. The offset value is incremented in the search stage. The sync lock status is asserted in the monitor stage. Once the sync lock status is asserted, it is not cleared until RESET\_DFR\_SYNC is called. The behavior of ALMA frame de-formatter is described in the user’s manual. For the detail, please refer it. | | |
| Update Rate | | Real time  (FPGA updates these items in real time. Every time this command is called, it reads values from FPGA.) | | |
| TE Related | | No | | |
| Synchronization | | Synchronous (\*1) | | |
| Data | Size | 8Bytes | | |
| Reset | RESET\_DFR, RESET\_DFR\_SYNC or device file open | | |
| Latch | Sync lock status (bit7 in Byte0) and Offset (bit0 in Byte0, bit7-0 in Byte1) are latched value. | | |
| Value | Byte0:  bit 7: Sync lock status(1=sync found 0=sync not found)  bit 6-1: Not used (undefined)  bit 0: Offset (MSBit of offset value)  Byte1:  bit7-0: Offset  Byte2-7: Reserved (undefined) | | |
| Category | ■Error  Sync lock status, Offset  If DRXP does not set “sync lock status” to 1(=sync found), it means DRXP can’t find sync-pattern from received data. This also means DRXP can’t transfer correct data to the host ring buffer. Please confirm signal integrity and data source. | | |
| Remarks | | (\*1) This command just reads a FPGA’s internal register. If there is an error in the physical layer, it sets EIO to errno. For the timeout value of physical layer, please refer a section **“5.5 ioctl”.** | | |
| Errors | | EIO  When this command fails to read or write internal register.  This errno means device driver can't control the device.  If this error occurs often, it may be hardware error is happened.  Please execute cold-reboot and try again. | | |

### GET\_DFR\_TEST\_DATA

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| ItemID | | GET\_DFR\_TST\_DAT\_ID\_D | 0x0000 1D04 | CH1-BitD |
| ItemID | | GET\_DFR\_TST\_DAT\_ID\_C | 0x0000 2D04 | CH2-BitC |
| ItemID | | GET\_DFR\_TST\_DAT\_ID\_B | 0x0000 3D04 | CH3-BitB |
| Description | | Read the Test Data configuration. When the device file is opened, these items are reset to 0. | | |
| Update Rate | | Real time  (FPGA updates these items in real time. Every time this command is called, it reads values from FPGA.) | | |
| TE Related | | No | | |
| Synchronization | | Synchronous (\*1) | | |
| Data | Size | 8Bytes | | |
| Reset | RESET\_DFR or device file open | | |
| Latch | All bits are real time value | | |
| Value | Byte0:  Definition of Byte0 is same as SET\_DFR\_TEST\_DATA.  Please refer to**”6.2.4SET\_DFR\_TEST\_DATA”**.  Byte1-7: Reserved (undefined.) | | |
| Category | ■Info  　This function just returns hardware/device driver information. | | |
| Remarks | | (\*1) This command just copies the test data configuration in the device driver to the user memory. Therefore, there is no specific timeout value. | | |
| Errors | | This command always sets 0 to errno. | | |

### GET\_DFR\_METAFRAME\_DELAY

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| ItemID | | GET\_DFR\_MTFRM\_DLY\_ID\_D | 0x0000 1D06 | CH1-BitD |
| ItemID | | GET\_DFR\_MTFRM\_DLY\_ID\_C | 0x0000 2D06 | CH2-BitC |
| ItemID | | GET\_DFR\_MTFRM\_DLY\_ID\_B | 0x0000 3D06 | CH3-BitB |
| Description | | When the device file is open, DRXP always measures the time between the rising edge timing of the 48ms TE and the timing when a metaframe bit is detected in the ALMA frames. The unit of measured values is a count of 200MHz clock. | | |
| Update Rate | | 48ms  (FPGA updates these items every 48ms. Every time this command is called, it reads values from FPGA.) | | |
| TE Related | | YES | | |
| Synchronization | | Synchronous (\*1) | | |
| Data | Size | 8Bytes | | |
| Reset | RESET\_DFR or device file open | | |
| Latch | All bits are real time value | | |
| Value | Byte0-3:  Number of 200 MHz clocks counted 32-bit , MSByte first.  Byte4-7: Reserved (undefined) | | |
| Category | ■Info  　This function just returns hardware/device driver information. | | |
| Remarks | | ・The delay count starts when DRXP receives 48ms TE.  ・The delay count returns to 0 when DRXP receives 48ms TE.  ・The value of count is copied to somewhere as the latest metaframe delay  when DRXP detects the metaframe bit in the ALMA frame. The delay count  does not stop.  ・The delay count stops when the delay count reaches the maximum value  of 32 bit integer (i.e.,0xFFFFFFFF).There is no copy of the delay count to the metaframe delay.  ・As a result, GET\_DFR\_METAFRAME\_DELAY always returns to the  latest metaframe delay.  (\*1) This command just reads a FPGA’s internal register. If there is an error in the physical layer, it sets EIO to errno. For the timeout value of physical layer, please refer a section **“5.5 ioctl”.** | | |
| Errors | | EIO  When this command fails to read or write internal register.  This errno means device driver can't control the device.  If this error occurs often, it may be hardware error is happened.  Please execute cold-reboot and try again. | | |

### GET\_DFR\_SN

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| ItemID | | GET\_DFR\_SN\_ID | 0x0000 7FFF |  |
| Description | | Read the serial number of the DRXP-boards. | | |
| Update Rate | | Static  (Every time this command is called, it reads values from FPGA.) | | |
| TE Related | | No | | |
| Synchronization | | Synchronous (\*1) | | |
| Data | Size | 8Bytes | | |
| Reset | - | | |
| Latch | All bits are real time value | | |
| Value | Byte0-2:  Serial number, MSByte first.  Byte3:  bit 7-1: Not used (undefined)  bit 0: 0=Prototype,　1=Product  Byte4: (driver version 2.0.3 or later)  bit 7-1: Not used (undefined)  bit 0: 0=IF1,　1=IF 2  (Bit 0 in byte4 is valid only for production DRXP board.)  Byte5-7: Reserved (undefined) | | |
| Category | ■Info  　This function just returns hardware/device driver information. | | |
| Remarks | | ■Remarks for the production DRXP boards  This register is common between both production DRXP board IFs except bit0 in Byte4. This function gives a common value except the bit when your application calls from either one.  (\*1) This command just reads a FPGA’s internal register. If there is an error in the physical layer, it sets EIO to errno. For the timeout value of physical layer, please refer a section **“5.5 ioctl”.** | | |
| Errors | | EIO  When this command fails to read or write internal register.  This errno means device driver can't control the device.  If this error occurs often, it may be hardware error is happened.  Please execute cold-reboot and try again. | | |

### GET\_DFR\_VER

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| ItemID | | GET\_DFR\_VER\_ID | 0x0000 7FF0 |  |
| Description | | Read the version information. | | |
| Update Rate | | Static  (Every time this command is called, it reads values from FPGA.) | | |
| TE Related | | No | | |
| Synchronization | | Synchronous (\*1) | | |
| Data | Size | 8Bytes | | |
| Reset | - | | |
| Latch | All bits are real time value | | |
| Value | Byte0:  Driver major version (0x0~0xFF)  Byte1:  Driver minor version (0x0~0xFF)  Byte2:  Driver micro version (0x0~0xFF)  Byte3:  Reserved (undefined)  Byte4-7:  FPGA version (0x0~0xFFFFFFFF) | | |
| Category | ■Info  　This function just returns hardware/device driver information. | | |
| Remarks | | ■Remarks for the production DRXP boards  This register is common between both production DRXP board IFs. This function gives a common value when your application calls from either one.  (\*1) This command just reads a FPGA’s internal register. If there is an error in the physical layer, it sets EIO to errno. For the timeout value of physical layer, please refer a section **“5.5 ioctl”.** | | |
| Errors | | EIO  When this command fails to read or write internal register.  This errno means device driver can't control the device.  If this error occurs often, it may be hardware error is happened.  Please execute cold-reboot and try again. | | |

### GET\_FPGA\_TEMP

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| ItemID | | GET\_FPGA\_TEMP\_ID | 0x0000 3808 |  |
| Description | | Read the FPGA junction temperature value.  The unit of the value is Celsius. | | |
| Update Rate | | Real time  (FPGA updates these items in real time. Every time this command is called, it reads values from FPGA.) | | |
| TE Related | | No | | |
| Synchronization | | Synchronous (\*1) | | |
| Data | Size | 8Bytes | | |
| Reset | RESET\_DFR or device file open | | |
| Latch | All bits are real time value | | |
| Value | Byte0-3:  4Byte Float ( the FPGA junction temperature )  LSByte first  Byte4-7:  Not defined  ■sample code  ST\_DRXPD\_MONDAT st\_mon;  float \*pf\_fpgatemp = (float\*)st\_mon.RdValue;  st\_mon.ItemID = GET\_FPGA\_TEMP\_ID;  memset( st\_mon.RdValue, 0x00, 8 );  ioctl( devfd, DRXPD\_GET\_MONSTS, &st\_mon );  printf( “temp=%f\n”, \*pf\_fpgatemp ); | | |
| Category | ■Info  　This function just returns hardware/device driver information. | | |
| Remarks | | This monitoring command is available with driver version 1.0.2 or later.  ■Remarks for the production DRXP boards  This register is common between both production DRXP board IFs. This command gives a common value when your application calls from either one.  (\*1) This command just reads a FPGA’s internal register. If there is an error in the physical layer, it sets EIO to errno. For the timeout value of physical layer, please refer a section **“5.5 ioctl”.** | | |
| Errors | | EIO  When this command fails to read or write internal register.  This errno means device driver can't control the device.  If this error occurs often, it may be hardware error is happened.  Please execute cold-reboot and try again. | | |

### GET\_SEU\_STATUS

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| ItemID | | GET\_SEU\_STATUS\_ID | 0x0000 0E00 |  |
| Description | | Read SEU function module status.  For the detail of SEU function module, please refer **“7.3 SEU Mitigation”**. | | |
| Update Rate | | Real time  (FPGA updates these items in real time. Every time this command is called, it reads values from FPGA.) | | |
| TE Related | | No | | |
| Synchronization | | Synchronous (\*4) | | |
| Data | Size | 8Bytes | | |
| Reset | RESET\_DFR | | |
| Latch | All bits are real time value | | |
| Value | This command returns below data structure. It is defined in “drxpd.h”.  ST\_SEU\_STATUS   |  |  | | --- | --- | | Type | Member | | unsigned int | uiStatus | | unsigned int | uiError |   <uiStatus>  The member “uiStatus” means SEU function module status.  Returns values are defined in “drxpd.h”.   |  |  | | --- | --- | | Value (defined in drxpd.h) | Description | | SEU\_STATUS\_INITIALIZATION | SEU module is in initialization state. | | SEU\_STATUS\_OBSERVATION | SEU module is in observation state. | | SEU\_STATUS\_CORRECTION | SEU module is in correction state. | | SEU\_STATUS\_INJECTION | SEU module is in injection state. | | SEU\_STATUS\_DETECTONLY | SEU module is in detect only state. | | SEU\_STATUS\_DIAGNOSTICSCAN | SEU module is in diagnostic scan state. | | SEU\_STATUS\_IDLE | SEU module is in idle state. | | SEU\_STATUS\_ERROR | SEU module is in error state. | | Other value | Undefined |   <uiError>  The member “uiError” shows SEU function module detail error status.  uiError represents error factor by bit position. Bits are defined in “drxpd.h”   |  |  | | --- | --- | | Bit position | Description | | SEU\_BITERR\_HEARTBEAT | Detect heartbeat error. (\*2) | | SEU\_BITERR\_HEARTBEAT\_S | Detect heartbeat error. (sticky)(\*1) | | SEU\_BITERR\_WD\_CLA | Watchdog timer error1. (\*3) | | SEU\_BITERR\_WD\_CLA\_S | Watchdog timer error1. (sticky)(\*1) | | SEU\_BITERR\_WD\_COR | Watchdog timer error2. (\*3) |   (Continued from previous page)   |  |  | | --- | --- | | Bit position | Description | | SEU\_BITERR\_WD\_COR\_S | Watchdog timer error2. (sticky)(\*1) | | Other value | Undefined |   (\*1) Once this bit is set, it keeps its state until user’s clear function is called.  (\*2) heartbeat error …　SEU function module monitors keep-alive signal in observation/detect only/diagnostic scan state. If it detects the timeout of this signal, it reports heartbeat error. It means SEU function module is in fatal error.  (\*3) watchdog error …　SEU function module monitors how long this module stays in correction state. Normally, this module transits to own state from correction state within at least 2 second. When it detects the too long stay in correction state, it reports watch dog error. Watch dog error1/2 represents where the timeout occurs. This detail info is for manufacture’s analysis.  ■sample code  ST\_DRXPD\_MONDAT st\_mon;  ST\_SEU\_STATUS \*pst\_sts = (ST\_SEU\_STATUS\*)st\_mon.RdValue;  st\_mon.ItemID = GET\_SEU\_STATUS\_ID;  memset( st\_mon.RdValue, 0x00, 8 );  ioctl( devfd, DRXPD\_GET\_MONSTS, &st\_mon);  printf("status=%08x, error=%08x\n", pst\_sts->uiStatus, pst\_sts->uiError); | | |
| Category | ■Fatal  ST\_SEU\_STATUS  This data structure informs user software of the status of SEU module. Normally, the SEU module status is in SEU\_STATUS\_OBSERVATION. Even though user application doesn’t change the SEU module status, it is not in this state. If it in other status, it may have an error in the SEU module or FPGA configuration. In the case that the FPGA configuration has an error, it may not work correctly. Please confirm whether an error is detected in the FPGA configuration or not. Or, please reboot your system and re-configure the FPGA. For the detail of the SEU module, please refer to “**7.3 SEU Mitigation**”. | | |
| Remarks | | This monitoring command is available with driver version 1.0.4 or later.  ■Remarks for the production DRXP boards  The SEU function is common between both production DRXP board IFs. It means both IFs control/monitor share the registers. If one IF changes the register value, it affects another IF.  (\*4) This command just reads a FPGA’s internal register. If there is an error in the physical layer, it sets EIO to errno. For the timeout value of physical layer, please refer a section **“5.5 ioctl”.** | | |
| Errors | | EIO  When this command fails to read or write internal register.  This errno means device driver can't control the device.  If this error occurs often, it may be hardware error is happened.  Please execute cold-reboot and try again. | | |

### GET\_SEU\_ERRCNT

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| ItemID | | GET\_SEU\_ERRCNT\_ID | 0x0000 0E01 |  |
| Description | | Read the SEU event counter.  For the detail of SEU function module, please refer **“7.3 SEU Mitigation”**. | | |
| Update Rate | | Real time  (FPGA updates these items in real time. Every time this command is called, it reads values from FPGA.) | | |
| TE Related | | No | | |
| Synchronization | | Synchronous (\*1) | | |
| Data | Size | 8Bytes | | |
| Reset | - | | |
| Latch | All bits are real time value | | |
| Value | This command returns below data structure. It is defined in “drxpd.h”.  ST\_SEU\_ERRCNT   |  |  | | --- | --- | | Type | Member | | unsigned int | uiCorErrCnt | | unsigned int | uiUnCorErrCnt |   <uiCorErrCnt>  The member “uiCorErrCnt” is counter of correctable error occurred.  Counter value range is 0 – 65535.  <uiUnCorErrCnt>  The member “uiUnCorErrCnt” is counter of uncorrectable error occurred.  Counter value range is 0 – 65535.  ■sample code  ST\_DRXPD\_MONDAT st\_mon;  ST\_SEU\_ERRCNT \*pst\_cnt = (ST\_SEU\_ERRCNT\*)st\_mon.RdValue;  st\_mon.ItemID = GET\_SEU\_ERRCNT\_ID;  memset( st\_mon.RdValue, 0x00, 8 );  ioctl( devfd, DRXPD\_GET\_MONSTS, &st\_mon);  printf("correctable=%u, uncorrectable=%u\n",  pst\_cnt->uiCorErrCnt, pst\_cnt->uiUnCorErrCnt ); | | |
| Category | ■Fatal  ST\_SEU\_ERRCNT  This data structure informs user software of the number of soft-error. If “uiUnCorErrCnt” is not zero, the FPGA configuration has an error (errors). In the case that the FPGA configuration has an error, it may not work correctly. Please confirm whether an error is detected in the FPGA configuration or not. Or, please reboot your system and re-configure the FPGA. For the detail of the SEU module, please refer to “**7.3 SEU Mitigation**”. | | |
| Remarks | | This monitoring command is available with driver version 1.0.4 or later.  ■Remarks for the production DRXP boards  The SEU function is common between both production DRXP board IFs. It means both IFs control/monitor share the registers. If one IF changes the register value, it affects another IF.  (\*1) This command just reads a FPGA’s internal register. If there is an error in the physical layer, it sets EIO to errno. For the timeout value of physical layer, please refer a section **“5.5 ioctl”.** | | |
| Errors | | EIO  When this command fails to read or write internal register.  This errno means device driver can't control the device.  If this error occurs often, it may be hardware error is happened.  Please execute cold-reboot and try again. | | |

### GET\_SEU\_ELAPSED

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| ItemID | | GET\_SEU\_ELAPSED\_ID | 0x0000 0E02 |  |
| Description | | Read the SEU function module’s elapsed time counter. Counter unit is second.  For the detail of SEU function module, please refer **“7.3 SEU Mitigation”**. | | |
| Update Rate | | Real time  (FPGA updates these items in real time. Every time this command is called, it reads values from FPGA.) | | |
| TE Related | | No | | |
| Synchronization | | Synchronous (\*1) | | |
| Data | Size | 8Bytes | | |
| Reset | - | | |
| Latch | All bits are real time value | | |
| Value | This command returns below data structure. It is defined in “drxpd.h”.  ST\_SEU\_ELPSD   |  |  | | --- | --- | | Type | Member | | unsigned int | uiObs | | unsigned int | uiTot |   <uiObs>  The member “uiObs”is incremented during SEU function module in Observation state. Counter unit is second. Value range is 0-1073741823.  <uiTot>  The member “uiTot”is incremented during SEU function module in state exclude Initialization. Counter unit is second. Value range is 0-1073741823.  ■sample code  ST\_DRXPD\_MONDAT st\_mon;  ST\_SEU\_ELPSD \*pst\_elps = (ST\_SEU\_ELPSD\*)st\_mon.RdValue;  st\_mon.ItemID = GET\_SEU\_ELAPSED\_ID;  memset( st\_mon.RdValue, 0x00, 8 );  ioctl( devfd, DRXPD\_GET\_MONSTS, &st\_mon);  printf("elapsed time(observation)=%u\n”, pst\_elps->uiObs );  printf("elapsed time=%u\n”, pst\_elps->uiTot ); | | |
| Category | ■Info  　This function just returns hardware/device driver information. | | |
| Remarks | | This monitoring command is available with driver version 1.0.4 or later.  (\*1) This command just reads a FPGA’s internal register. If there is an error in the physical layer, it sets EIO to errno. For the timeout value of physical layer, please refer a section **“5.5 ioctl”.**  ■Remarks for the production DRXP boards  The SEU function is common between both production DRXP board IFs. It means both IFs control/monitor share the registers. If one IF changes the register value, it affects another IF. | | |
| Errors | | EIO  When this command fails to read or write internal register.  This errno means device driver can't control the device.  If this error occurs often, it may be hardware error is happened.  Please execute cold-reboot and try again. | | |

### GET\_SEU\_DIAERRCNT

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| ItemID | | GET\_SEU\_DIAERRCNT\_ID | 0x0000 0E03 |  |
| Description | | Read the SEU function module’s error counter.  This command is available with driver version 2.0.3 or later.  Read the detected error count during diagnostic scan. Please call this function after confirming that diagnostic scan is completed.  You can change the SEU module state by calling control item "SET\_SEU\_STATE". And when the diagnostic scan is completed, the SEU module state transits to IDLE automatically. You can check the SEU module state by calling monitoring item "GET\_SEU\_STAUS". For the detail of diagnostic scan, please refer **“7.3 SEU Mitigation”.** | | |
| Update Rate | | Real time  (FPGA updates these items in real time. Every time this command is called, it reads values from FPGA.) | | |
| TE Related | | No | | |
| Synchronization | | Synchronous (\*1) | | |
| Data | Size | 8Bytes | | |
| Reset | - | | |
| Latch | All bits are real time value | | |
| Value | Byte0-3:  4 Byte integer  LSByte first  Byte4-7: Reserved (undefined) | | |
| Category | ■Fatal  4Byte integer  If this value is not zero, the FPGA configuration has an error (errors). In the case that the FPGA configuration has an error, it may not work correctly. Please confirm whether an error is detected in the FPGA configuration or not. Or, please reboot your system and re-configure the FPGA. For the detail of the SEU module, please refer to “**7.3 SEU Mitigation**”. | | |
| Remarks | | (\*1) This command just reads a FPGA’s internal register. If there is an error in the physical layer, it sets EIO to errno. For the timeout value of physical layer, please refer a section **“5.5 ioctl”.**  ■Remarks for the production DRXP boards  The SEU function is common between both production DRXP board IFs. It means both IFs control/monitor share the registers. If one IF changes the register value, it affects another IF. | | |
| Errors | | EIO  When this command fails to read or write internal register.  This errno means device driver can't control the device.  If this error occurs often, it may be hardware error is happened.  Please execute cold-reboot and try again. | | |

### GET\_CONSUMPTION

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| ItemID | | GET\_CONSUMPTION\_ID | 0x0000 3900 |  |
| Description | | Read the DRXP board power consumption value.  Return value’s unit is Watt. | | |
| Update Rate | | Real time  (FPGA updates these items in real time. Every time this command is called, it reads values from FPGA.) | | |
| TE Related | | No | | |
| Synchronization | | Synchronous (\*1) | | |
| Data | Size | 8Bytes | | |
| Reset | RESET\_DFR or device file open | | |
| Latch | All bits are real time value | | |
| Value | Byte0-3:  4Byte Float (DRXP board power consumption)  LSByte first  Byte4-7:  Not defined  ■sample code (Also see 6.1.13 GET\_FPGA\_TEMP)  ST\_DRXPD\_MONDAT st\_mon;  float \*pf\_consumption= (float\*)st\_mon.RdValue;  st\_mon.ItemID = GET\_CONSUMPTION\_ID;  memset( st\_mon.RdValue, 0x00, 8 );  ioctl( devfd, DRXPD\_GET\_MONSTS, &st\_mon );  printf( “result=%f\n”, \*pf\_consumption ); | | |
| Category | ■Info  　This function just returns hardware/device driver information. | | |
| Remarks | | This monitoring command is available with driver version 2.0.0 or later. This monitoring item is implemented only for production DRXP boards.  (\*1) This command just reads a FPGA’s internal register. If there is an error in the physical layer, it sets EIO to errno. For the timeout value of physical layer, please refer a section **“5.5 ioctl”.**  ■Remarks for the prototype DRXP boards  < version from 2.0.0 to 2.0.2>  When this command is called from prototype DRXP boards, it tries to access nonexistent register. It causes unexpected behavior. Please don’t use this command with prototype DRXP boards in these driver versions.  < version 2.0.3 or later >  When this command is called from prototype DRXP boards, ioctl returns -1 and sets ENOSYS to errno.  ■Remarks for the production DRXP boards  This register is common between both production DRXP board IFs. This command gives a common value when your application calls from either one. | | |
| Errors | | EIO  When this command fails to read or write internal register.  This errno means device driver can't control the device.  If this error occurs often, it may be hardware error is happened.  Please execute cold-reboot and try again.  ENOSYS(version 2.0.3 or later)  When this command is used with prototype DRXP boards. | | |

### GET\_OVH\_THRESHOLD

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| ItemID | | GET\_OVH\_THRESHOLD\_ID | 0x0000 0F00 |  |
| Description | | Get the current threshold value of overheat protection. | | |
| Update Rate | | Real time | | |
| TE Related | | No | | |
| Synchronization | | Synchronous (\*1) | | |
| Data | Size | 8Bytes | | |
| Reset | Device file open | | |
| Latch | All bits are real time value | | |
| Value | Byte0-3:  4 Byte integer (DRXP board overheat protection.)  LSByte first  Byte4-7: Reserved (undefined) | | |
| Category | ■Info  　This function just returns hardware/device driver information. | | |
| Remarks | | This monitoring command is available with driver version 2.0.0 or later.  This monitoring item is implemented only for production DRXP boards.  This value is reset to 90 degree Celsius when DRXP driver is loaded.  When the FPGA junction temperature exceeds this threshold, DRXP board initiates forcible reset.  This register is common between both production DRXP board IFs. This command gives a common value when your application calls this from either one.  (\*1) This command just copies device driver’s internal data. Therefore, there is no specific timeout value.  ■Remarks for prototype DRXP boards  < version from 2.0.0 to 2.0.1>  When this command is called from prototype DRXP boards, it gets the overheat flag value. But that value is meaningless.  < version 2.0.3 to 2.0.7>  When this command is called from prototype DRXP boards, ioctl returns -1 and sets ENOSYS to errno. | | |
| Errors | | EIO  When this command fails to read or write internal register.  This errno means device driver can't control the device.  If this error occurs often, it may be hardware error is happened.  Please execute cold-reboot and try again.  ENOSYS(version 2.0.3 to 2.0.7)  When this command is used with prototype DRXP boards. | | |

### GET\_OVH\_FLG

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| ItemID | | GET\_OVH\_FLG\_ID | 0x0000 0F01 |  |
| Description | | Get the overheat flag. This flag represents the production DRXP in the overheating state.  For the detail of overheating protection, please refer to **“7.4 The overheating protection”**. | | |
| Update Rate | | Real time | | |
| TE Related | | No | | |
| Synchronization | | Synchronous (\*1) | | |
| Data | Size | 8Bytes | | |
| Reset | - | | |
| Latch | All bits are real time value | | |
| Value | Byte0:  bit0:　1…DRXP is in the overheating state.  With the driver version from 2.0.0 to 2.0.2, this bit is set when the FPGA junction temperature exceeds the user-defined threshold or hardware threshold.  With the driver version 2.0.3 or later, this bit is set when the FPGA junction temperature exceeds the user-defined threshold.  0…Normal state  bit1: This bit is available with driver version 2.0.3 or later.  1…DRXP is in the overheat state.  (The FPGA junction temperature exceeds the hardware threshold.)  0…Normal state  bit2-7: Reserved(undefined)  Byte1-7: Reserved (undefined) | | |
| Category | ■Fatal  　Bit0 and bit1 in the Byte0  If these bits are asserted, it means DRXP is in the overheat state. The overheating may cause hardware damage. Please confirm the server machine environment or turn off the server for preventing hardware damage. | | |
| Remarks | | This monitoring command is available with driver version 2.0.0 or later.  This monitoring item is implemented only for production DRXP boards.  This register is common between both production DRXP board IFs. This command gives a common value when your application calls this from either one.  (\*1) This command just reads a FPGA’s internal register. If there is an error in the physical layer, it sets EIO to errno. For the timeout value of physical layer, please refer a section **“5.5 ioctl”.**  ■Remarks for the prototype DRXP boards  < version from 2.0.0 to 2.0.2>  When this command is called from prototype DRXP boards, it gets the overheat flag value. But that value is meaningless.  < version 2.0.3 to 2.0.8>  When this command is called from prototype DRXP boards, ioctl returns -1 and sets ENOSYS to errno. | | |
| Errors | | EIO  When this command fails to read or write internal register.  This errno means device driver can't control the device.  If this error occurs often, it may be hardware error is happened.  Please execute cold-reboot and try again.  ENOSYS(version 2.0.3 to 2.0.7)  When this command is used with prototype DRXP boards. | | |

### GET\_DFR\_FORCETX

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| ItemID | | GET\_DFR\_FORCETX\_ID | 0x0000 1F00 |  |
| Description | | This command gets the status of debugging function for transmitting PRBS31/PRBS63 pattern forcibly. | | |
| Update Rate | | Real time | | |
| TE Related | | No | | |
| Synchronization | | Synchronous (\*1) | | |
| Data | Size | 8Bytes | | |
| Reset | RESET\_DFR or device file open | | |
| Latch | All bits are real time value | | |
| Value | Byte0:  Enable/Disable this capability.  - DFRFORCETX\_ENABLE(0x01) … enabled.  - DFRFORCETX\_DISABLE(0x02) … disabled.  - DFRFORCETX\_UNKNOWN(0x00) …unexpected value is read from register.  Above value is defined in the drxpd.h.  Byte1-3: Reserved (undefined)  Byte4:  Current data type.  - DFRFORCETX\_PRBS31R(0x02) (\*1)  - DFRFORCETX\_PRBS31(0x03) (\*2)  - DFRFORCETX\_PRBS63(0x04) (\*2)  - DFRFORCETX\_PRBS31SR(0x05) (\*2)  - DFRFORCETX\_PRBS31S(0x06) (\*2)  - DFRFORCETX\_PRBS63S(0x07) (\*2)  Above value is defined in the drxpd.h.  If the other value is returned to this Byte, the DRXP board is transmitting undefined data.  For the detail fo data format aboves, please refer to the **“6.2.12.1　DFR\_FORCETX data format”.**  (\*1) In the device driver version 2.0.8 or older, it defined as DFRFORCETX\_PRBS. User application can also use this definition with version 2.0.9 or later.  (\*2) These definitions can be available with device driver version 2.0.9 or later.  Byte5-7: Reserved (undefined) | | |
| Category | ■Info  　This function just returns hardware/device driver information. | | |
| Remarks | | This monitoring command is available with driver version 2.0.2 or later.  This monitoring item is implemented only for production DRXP boards.  (\*1) This command just reads a FPGA’s internal register. If there is an error in the physical layer, it sets EIO to errno. For the timeout value of physical layer, please refer a section **“5.5 ioctl”.** | | |
| Errors | | ENOSYS  If this command is called for prototype DRXP board, it sets ENOSYS to an errno.  EIO  When this command fails to read or write internal register.  This errno means device driver can't control the device.  If this error occurs often, it may be hardware error is happened.  Please execute cold-reboot and try again. | | |

### GET\_PARITY\_RANGE

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| ItemID | | GET\_PARITY\_RANGE\_ID | 0x0000 1F01 |  |
| Description | | This command gets how many frames the FGPA uses to generate the data parity.  The FPGA generates the 16bit parity by using N ALMA frames. Your application can confirm the data integrity of internal FPGA logic. The valid range of N is from 1 to 3,000,000. This FPGA logic starts to generate parity when it detects a meta frame bit. | | |
| Update Rate | | Real time | | |
| TE Related | | No | | |
| Synchronization | | Synchronous (\*1) | | |
| Data | Size | 8Bytes | | |
| Reset | RESET\_DFR or device file open | | |
| Latch | All bits are real time value | | |
| Value | Byte0-3:  4Byte Integer  The number of ALMA frame that FPGA logic uses to calculate the data parity. The default value is 1.  LSByte first.  Byte4-7: Reserved (undefined) | | |
| Category | ■Info  　This function just returns hardware/device driver information. | | |
| Remarks | | This monitoring command is available with driver version 2.0.3 or later.  This monitoring item is implemented only for production DRXP boards.  (\*1) This command just reads a FPGA’s internal register. If there is an error in the physical layer, it sets EIO to errno. For the timeout value of physical layer, please refer a section **“5.5 ioctl”.** | | |
| Errors | | ENOSYS (version 2.0.3 to 2.0.7)  If this command is called for prototype DRXP board, it sets ENOSYS to an errno.  EIO  When this command fails to read or write internal register.  This errno means device driver can't control the device.  If this error occurs often, it may be hardware error is happened.  Please execute cold-reboot and try again. | | |

## Control Items

This section describes control items used in the ioctl(DRXPD\_SET\_CONTROL).

### RESET\_DFR

|  |  |  |  |
| --- | --- | --- | --- |
| ItemID | RST\_DFR\_ID\_D | 0x0000 9000 | CH1-BitD |
| ItemID | RST\_DFR\_ID\_C | 0x0000 A000 | CH2-BitC |
| ItemID | RST\_DFR\_ID\_B | 0x0000 B000 | CH3-BitB |
| ItemID | RST\_DFR\_ID\_ALL | 0x0000 C000 | All Bits |
| Description | Reset the ALMA de-formatter. Initiate a software reset of the FPGA. | | |
| TE Related | No | | |
| Synchronization | Synchronous (driver version 2.0.3 or later)  In the case of driver version 2.0.2 or older, works as asynchronous  Timeout  Maximum timeout value is 800ms (\*). If timeout occurs, There may be a hardware error in FPGA.  About the physical layer timeout value, please refer a section **“**Command :DRXPD INIT**5.5 ioctl”**.  (\*) The timeout value of the device driver version 2.0.7 or older is 100ms | | |
| Data | 8Bytes | | |
| Byte0:  bit7: 1=Master reset (\*2)(\*3)  0=No action.  (This bit is valid only for RST\_DFR\_ID\_ALL.)  bit6: 1=Common logic reset(\*3)  0=No action  (This bit is valid only for RST\_DFR\_ID\_ALL.)  bit5-1: Reserved  bit0: 1=Clear status register and voltage status register (\*1)  0=No action  Byte1-7: Reserved (always 0)  If RESET\_DFR is called with bit0 in byte0, bit6 and bit7 in byte0 are equal to zero, it does nothing. It doesn’t reset FPGA and status/voltage status registers.  (\*1) Target status register /voltage status register list   * ALARM STATUS register * POWER ALARM register * DFR STATUS register (\*1-1) * DFR VOLTAGE STATUS register * DFR CHECKSUM COUNTER register * DFR SYNC ERR CNT register   Note: DFR\_STATUS and DFR\_VOLTAGE\_STATUS register are reset when RESET\_DFR is called with RST\_DFR\_ID\_ALL .  (\*1-1)  When using with production DRXP and device driver 2.0.7 or older, the DFR\_STATUS register has IF1’s DDR PLL status and IF2’s one. When user application calls this function, it sometimes causes bug behavior. For example, when the user application calls RESET\_DFR with a clear register flag using IF1’s device file, it resets register information related to the IF1. But, it doesn’t reset DDR PLL status of IF2. Therefore, the device driver sometimes decides that the register reset of RESET\_DFR has failed. In this case it sets ECANCELED to errno. This behavior is fixed in device driver 2.0.8 or later.  (\*2) If you call RESET\_DFR with master reset flag, please restart reception/transmission operation from ioctl(DRXPD\_INIT). For the detail, please refer to “4.4 Example of reception” and “4.5 Example of transmission”.  (\*3) The DRXP logic is roughly composed of following logics.  - PCIe logic (for IF1)  - DDR logic (for IF1)  (- ALMA logic (for IF2))  (- DDR logic (for IF2))  - ALMA logic  - SEU logic  - FLASH logic  - System Monitoring logic  The master reset bit and common logic bit reset FPGA logic.  Normally, it is not necessary to call RESET\_DFR command with mater\_reset bit or common\_logic\_reset bit. If user application encounters an unexpected error, there is the possibility which RESET\_DFR with master reset bit or common logic reset bit recoveries this error.  In the case of production DRXP board, it has DDR and ALMA logic for each IF.  The RESET\_DFR command with master\_reset bit resets following logics. In the case of production DRXP board, This command only resets the IF logic related to the device file. If you initiate master reset of IF1, it doesn’t reset IF2 logic.   |  |  | | --- | --- | | Prototype DRXP board | Produciton DRXP board | | - DDR logic  - ALMA logic  - FLASH logic  - System Monitoring logic  - SEU logic | - DDR logic for IF1 or IF2  - ALMA logic for IF1 or IF2 |   (\* RESET\_DFR command only resets the SEU core logic, it doesn’t reset status register (e.g. soft-error error counter) )  The RESET\_DFR command with common\_logic\_reset bit resets  the logics which are not related to PCIe, DDR and ALMA logic.  This bit is only available on the production DRXP board.  FLASH logic, System Monitoring logic and SEU logic are shared logic between IFs. Resetting the common logic by one IF affects another IF. | | |
| Remarks | Bit7 and bit0 can be set simultaneously.  Master reset does not clear the status register and voltage status register.  ■Remarks for the driver version 2.0.2 or older  <the behavior of mater reset>  The master reset clears the status register and voltage status register with the driver version 2.0.2 or older.  <ItemID>  This command was defined as ItemIDs, “DFR\_RST\_ID\_｛B|C|D|ALL｝” with the driver version 2.0.2 or older. The device driver version 2.0.3 or later maintain the old ItemID definitions. Your application can also call this command with old definitions.  ■Remarks for the production DRXP boards  The following registers are common between the production DRXP board IFs. Therefore, resetting them by one IF affects the value another IF reads.   * DFR VOLTAGE STATUS register   The following registers are independent for each IF. If one IF resetting them by one IF doesn’t affect the value another IF reads.   * DFR STATUS register * ALARM STATUS register * POWER ALARM register * DFR CHECKSUM COUNTER register * DFR SYNC ERR CNT register | | |
| Errors | EAGAIN  When this command fails to reset FPGA. | | |



### RESET\_DFR\_CHKSUM

|  |  |  |  |
| --- | --- | --- | --- |
| ItemID | RST\_DFR\_CHK\_ID\_D | 0x0000 9502 | CH1-BitD |
| ItemID | RST\_DFR\_CHK\_ID\_C | 0x0000 A502 | CH2-BitC |
| ItemID | RST\_DFR\_CHK\_ID\_B | 0x0000 B502 | CH3-BitB |
| ItemID | RST\_DFR\_CHK\_ID\_ALL | 0x0000 C502 | All Bits |
| Description | Reset the checksum counters to zero.  RESET\_DFR causes this reset. | | |
| TE Related | No | | |
| Synchronization | Synchronous (version 2.0.3 or later)  In the case of driver version 2.0.2 or older, works as asynchronous  Timeout  This command checks the register after resetting. If there is the possibility that it fails reset counter, it sets ECANCELED to errno. Normally, this command returns less than 1ms.  About the physical layer timeout value, please refer a section **“**Command :DRXPD INIT**5.5 ioctl”**. | | |
| Data | 8Bytes | | |
| Byte0-7: Reserved (always 0) | | |
| Errors | EIO  When this command fails to read or write internal register.  ECANCELED  When this command fails to reset counter. | | |

### RESET\_DFR\_SYNC

|  |  |  |  |
| --- | --- | --- | --- |
| ItemID | RST\_DFR\_SYNC\_ID\_D | 0x0000 9507 | CH1-BitD |
| ItemID | RST\_DFR\_SYNC\_ID\_C | 0x0000 A507 | CH2-BitC |
| ItemID | RST\_DFR\_SYNC\_ID\_B | 0x0000 B507 | CH3-BitB |
| ItemID | RST\_DFR\_SYNC\_ID\_ALL | 0x0000 C507 | All Bits |
| Description | Reset the internal sync status.  This restarts the sync engine and all sync counters are reset to zero.  RESET\_DFR resets them to 0. | | |
| TE Related | No | | |
| Synchronization | Synchronous (version 2.0.3 or later)  It works as asynchronous with the driver version 2.0.2 or older.  Timeout  This command checks the sync engine status after resetting. If there is the possibility that it fails to reset, it sets ECANCELED to errno. Normally, this command returns less than 1ms.  About the physical layer timeout value, please refer a section **“**Command :DRXPD INIT**5.5 ioctl”**. | | |
| Data | 8Bytes | | |
| Byte0-7: Reserved (always 0) | | |
| Errors | EIO  When this command fails to read or write internal register.  ECANCEND  When this command fails to reset sync engine and all sync counters. | | |

### SET\_DFR\_TEST\_DATA

|  |  |  |  |
| --- | --- | --- | --- |
| ItemID | SET\_DFR\_TST\_DAT\_ID\_D | 0x0000 9509 | CH1-BitD |
| ItemID | SET\_DFR\_TST\_DAT\_ID\_C | 0x0000 A509 | CH2-BitC |
| ItemID | SET\_DFR\_TST\_DAT\_ID\_B | 0x0000 B509 | CH3-BitB |
| ItemID | SET\_DFR\_TST\_DAT\_ID\_ALL | 0x0000 C509 | All Bits |
| Description | Change the output of DRXP-board to system-memory.  RESET\_DFR resets to 0. When the device file is opened, these items are resets to 0.  You can confirm completion of this command by calling the monitoring item (GET\_DFR\_TEST\_DATA). | | |
| TE Related | No | | |
| Synchronization | Asynchronous | | |
| Data | 8Bytes | | |
| Byte0:  bit7-6: Not used (always 0).  bit5-4: 00=Test data output disabled (normal data output)  01=Random number generator  　　PRBS-31(free-run). LSB side is the old.  10=Counting data  　　It is following, It is free-run.  　　 0x00000003\_00000002\_00000001\_00000000　(first)  　　 0x00000007\_00000006\_00000005\_00000004　(next)  ：  　　　　　　　　　　　<Continued.>  11=Random number generator  　　PRBS-31(initialization every 48ms). LSB side is the old.  bit3-0: These bits override the above selection.  0x0=Not used (according to bit 5-4, if set)  0x1=0x3C3C3C3C\_3C3C3C3C\_3C3C3C3C\_3C3C3C3C(all ‘0x3C’)  0x2=0xC3C3C3C3\_C3C3C3C3\_C3C3C3C3\_C3C3C3C3(all ‘0xC3’)  0x3=0x33333333\_33333333\_33333333\_33333333(all ‘0x3’)  0x4=0xCCCCCCCC\_CCCCCCCC\_CCCCCCCC\_CCCCCCCC(all ‘0xC’)  0x5=0x55555555\_55555555\_55555555\_55555555(all ‘0x5’)  0x6=0x5555AAAA\_5555AAAA\_5555AAAA\_5555AAAA  0x7=Not used (according to bits 5-4, if set)  0x8=0x00000000\_00000000\_00000000\_00000000(all '0x0')  0x9=0x11111111\_11111111\_11111111\_11111111(all '0x1')  0xA=0xAAAAAAAA\_AAAAAAAA\_AAAAAAAA\_AAAAAAAA(all ‘0xA’)  0xB=0xAAAA5555\_AAAA5555\_AAAA5555\_AAAA5555  0xC= Not used (according to bits 5-4, if set)  0xD=0xFEDCBA98\_01234567\_BA987654\_456789AB  0xE=0x01234567\_89ABCDEF\_FEDCBA98\_76543210  0xF=Not used (according to bit 5-4, if set)  Byte1-7: Reserved (always 0) | | |
| Errors | EIO  When this command fails to read or write internal register. | | |
| Remarks | This command was defined with ItemIDs, “DFR\_TST\_DAT\_ID\_{B/C/D/ALL}”. With the driver version 2.0.2 or older. The device driver version 2.0.3 or later maintain the old ItemID definitions. Your application can also call this command with old definitions. | | |

### INSERT\_SYNCPTN\_ERR

|  |  |  |  |
| --- | --- | --- | --- |
| ItemID | INS\_SYNC\_ERR\_ID\_D | 0x0000 950A | CH1-BitD |
| ItemID | INS\_SYNC\_ERR\_ID\_C | 0x0000 A50A | CH2-BitC |
| ItemID | INS\_SYNC\_ERR\_ID\_B | 0x0000 B50A | CH3-BitB |
| ItemID | INS\_SYNC\_ERR\_ID\_ALL | 0x0000 C50A | All Bits |
| Description | During transmitting the data into the optical fiber, you can insert a sync pattern error of ALMA frame by using this item.  Every time you perform the item, only one frame raises the sync pattern error.  When this function inserts sync pattern error, it writes all 0 to the sync pattern field of ALMA frame.  You can confirm completion of this command by calling the monitoring the monitoring item (GET\_DFR\_SYNC\_ERR\_CNT) in the loopback test environment. The value of counter is incremented. | | |
| TE Related | No | | |
| Synchronization | Asynchronous | | |
| Data | 8Bytes | | |
| Byte0-7: Reserved (always 0) | | |
| Errors | EIO  When this command fails to read or write internal register. | | |

### SET\_CHKSUM\_ERR

|  |  |  |  |
| --- | --- | --- | --- |
| ItemID | SET\_CHK\_ERR\_ID\_D | 0x0000 950B | CH1-BitD |
| ItemID | SET\_CHK\_ERR\_ID\_C | 0x0000 A50B | CH2-BitC |
| ItemID | SET\_CHK\_ERR\_ID\_B | 0x0000 B50B | CH3-BitB |
| ItemID | SET\_CHK\_ERR\_ID\_ALL | 0x0000 C50B | All Bits |
| Description | During transmitting the data into the optical fiber, you can set a sum error of ALMA frame by using this item.  You can confirm completion of this command by calling the monitoring item (GET\_DFR\_CHKSUM\_COUNTER) in the loopback test environment. The value of counter is incremented. | | |
| TE Related | No | | |
| Synchronization | Asynchronous | | |
| Data | 8Bytes | | |
| Byte0:  bit7-1: Not used (always 0).  bit0: 0 = Adding the correct sum.  1 = Adding the incorrect sum( Invert the correct sum).  Byte1-7: Reserved (always 0) | | |
| Errors | EIO  When this command fails to read or write internal register. | | |

### SET\_SQCNT\_ERR

|  |  |  |  |
| --- | --- | --- | --- |
| ItemID | SET\_SQ\_ERR\_ID\_D | 0x0000 950C | CH1-BitD |
| ItemID | SET\_SQ\_ERR\_ID\_C | 0x0000 A50C | CH2-BitC |
| ItemID | SET\_SQ\_ERR\_ID\_B | 0x0000 B50C | CH3-BitB |
| ItemID | SET\_SQ\_ERR\_ID\_ALL | 0x0000 C50C | All Bits |
| Description | During transmitting the data into the optical fiber, you can set a sequence count error of ALMA frame by using this item.  You can confirm completion of this command by calling the monitoring item (GET\_DFR\_CHKSUM\_COUNTER) in the loopback test environment. The value of counter is incremented. | | |
| TE Related | No | | |
| Synchronization | Asynchronous | | |
| Data | 8Bytes | | |
| Byte0:  bit7-1: Not used (always 0).  bit0: 0 = Adding the correct sequence counter　(increment by 1).  1 = Adding the incorrect sequence counter　(increment by 2).  Byte1-7: Reserved (always 0) | | |
| Errors | EIO  When this command fails to read or write internal register. | | |

### SET\_SEU\_STATUS

|  |  |  |  |
| --- | --- | --- | --- |
| ItemID | SET\_SEU\_STATUS\_ID | 0x0000 0E04 | - |
| Description | Transition the SEU function module state.  For the detail of SEU function module, please refer **“7.3 SEU Mitigation”**. | | |
| TE Related | No | | |
| Synchronization | Synchronous  Timeout  This command waits for completion of SEU module state transition. is changed or not after writing register. If it fails to shift its state in 500ms, it sets ECANCELED to errno. | | |
| Data | 8Bytes | | |
| Please specify the transition destination state by following data structure.  ST\_SEU\_STATUS   |  |  | | --- | --- | | Type | Member | | unsigned int | uiStatus | | unsigned int | uiError |   <uiStatus>  You can specify the transition destination state by using this member variable.  Available value is in the following table.   |  |  | | --- | --- | | Value (defined in drxpd.h) | Description | | SEU\_STATUS\_OBSERVATION | observation state. | | SEU\_STATUS\_DETECTONLY | detect only state. | | SEU\_STATUS\_DIAGNOSTICSCAN | diagnostic scan state. | | SEU\_STATUS\_IDLE | idle state. |   The possible transition destination state is defined at state transition diagram in **“7.3.1.1 State”**.  <uiError>  This member is not used.  ■sample code  ST\_DRXPD\_CONTDAT st\_con;  ST\_SEU\_STATUS \*pst\_sts = (ST\_SEU\_STATUS\*)st\_con.WrValue;  // change state : Observation -> Idle  st\_con.ItemID = SET\_SEU\_STATUS\_ID;  pst\_sts->uiStatus = SEU\_STATUS\_IDLE;  ioctl( devfd, DRXPD\_SET\_CONTROL, &st\_con); | | |
| Remarks | This command is available with driver version 1.0.4 or later.  ■Remarks for prototype DRXP boards  The diagnostic scan state is not available with device driver 2.0.7 or older.  In the verision 2.0.7 or older, your application can shift SEU function module state to “Diagnostic Scan” state but can’t acquire detected error count. Please don’t use this state.  ■Remarks for the production DRXP boards  \* The diagnostic scan state is not available with device driver 2.0.2 or older.  \* The SEU function is common function between both production DRXP board IFs. It means both IFs control/monitor share the registers. Changing the register value by on IF affects another IF. | | |
| Errors | EINVAL  When this command detects invalid argument. (Unmatch uiStatus)  EIO  When this command fails to read or write internal register.  ECANCELED  SEU module state is not changed. | | |

### SET\_SEU\_INJERR

|  |  |  |  |
| --- | --- | --- | --- |
| ItemID | SET\_SEU\_INJERR\_ID | 0x0000 0E05 | - |
| Description | Injecting error to FPGA configuration memory.  This function is available in SEU function module in IDLE state.  For the detail of SEU function module, please refer **“7.3 SEU Mitigation”**. | | |
| TE Related | No | | |
| Synchronization | Synchronous  Timeout  Maximum timeout value is 500ms. This command waits for the completeion of error injection (SEU module state returns to “IDLE” state.). If it doesn’t return to “IDLE” state in 500ms. It sets ECANCELED to errno. | | |
| Data | 8Bytes | | |
| You can specify the bit in configuration memory space by using three addresses as below.  ■Byte 3-0  <Prototype DRXP boards>   |  |  |  |  | | --- | --- | --- | --- | | Byte 3  (MSB<->LSB) | Byte 2  (MSB<->LSB) | Byte1  (MSB<->LSB) | Byte0 (MSB<->LSB) | | 000lllll | llllllll | llllwwww | ｗwwbbbbb |   l [17bit] This means linear frame address. Value range is 0-26179(decimal)  w [7bit] This means word address. Value range is 0-122(decimal)  b [5bit] This means bit address. Value range is 0-31 (decimal)  Please use below addresses to inject an error.  They are ECC area of configuration data. Therefore, it doesn’t inject error to logic area. You can test error detection function safely by using them. (Xilinx does not recommend error injection test to logic area.)   |  |  |  |  | | --- | --- | --- | --- | | Byte 3  (MSB<->LSB) | Byte 2  (MSB<->LSB) | Byte1  (MSB<->LSB) | Byte0 (MSB<->LSB) | | 0x02 | 0x75 | 0x67 | 0xA5 |   <Production DRXP boards>   |  |  |  |  | | --- | --- | --- | --- | | Byte 3  (MSB<->LSB) | Byte 2  (MSB<->LSB) | Byte1  (MSB<->LSB) | Byte0 (MSB<->LSB) | | 00llllll | Llllllll | llllwwww | ｗwwbbbbb |   l [18bit] This means linear frame address. Value range is 0-75281 (decimal)  w [7bit] This means word address. Value range is 0-92(decimal)  b [5bit] This means bit address. Value range is 0-31(decimal)  Please use below addresses to inject an error.  They are ECC area of configuration data. Therefore, it doesn’t inject error to logic area. You can test error detection function safely by using them. (Xilinx does not recommend error injection test to logic area.)   |  |  |  |  | | --- | --- | --- | --- | | Byte 3  (MSB<->LSB) | Byte 2  (MSB<->LSB) | Byte1  (MSB<->LSB) | Byte0 (MSB<->LSB) | | 0x02 | 0x75 | 0x67 | 0xA5 |   ■Byte 7-4  Undefined  ■sample code  ST\_DRXPD\_CONTDAT st\_con;  st\_con.ItemID = SET\_SEU\_INJERR\_ID;  st\_con.WrValue[0] = 0xD5;  st\_con.WrValue[1] = 0x67;  st\_con.WrValue[2] = 0xA5;  st\_con.WrValue[3] = 0x02;  ioctl( devfd, DRXPD\_SET\_CONTROL, &st\_con); | | |
| Remarks | This command is available with driver version 1.0.4 or later.  ■Remarks for the production DRXP boards  The SEU function is common function between both production DRXP board IFs. It means both IFs control/monitor the same registers. Changing the register value by one IF affects another IF. | | |
| Errors | EIO  When this command fails to read or write internal register.  EAGAIN  This command can be called when SEU module is in “IDLE”state.  When SEU module is not in “IDLE” state, it sets EAGAIN to errno.  ECANCELED  Error injection operation is not completed. | | |

### SET\_SEU\_RSTERR

|  |  |  |  |
| --- | --- | --- | --- |
| ItemID | SET\_SEU\_RSTERR\_ID | 0x0000 0E08 | - |
| Description | This function clears sticky error bits in SEU function module.  For the detail of error bit, please refer the description of ST\_SEU\_STATUS data structure in the **“6.1.14 GET\_SEU\_STATUS”**. To confirm completion of this command, please call monitoring item(GET\_SEU\_STATUS).  ST\_SEU\_STATUS   |  |  | | --- | --- | | Type | Member | | unsigned int | uiStatus | | unsigned int | uiError |   <uiError>  The bits in the following table are cleared by this function.   |  |  | | --- | --- | | Bit position | Description | | 0x00004000 | Detect heartbeat error. | | 0x00000040 | Watchdog timer error1. | | 0x00000010 | Watchdog timer error2. |   For the detail of SEU function module, please refer **“7.3 SEU Mitigation”**. | | |
| TE Related | No | | |
| Synchronization | Asynchronous | | |
| Data | Nothing | | |
| Remarks | This command is available with driver version 1.0.4 or later.  ■Remarks for the production DRXP boards  The SEU function is common function between both production DRXP board IFs. It means both IFs control/monitor the same registers. Changing the register value by one IF affects another IF. | | |
| Errors | EIO  When this command fails to read or write internal register. | | |

### SET\_OVH\_THRESHOLD

|  |  |  |  |
| --- | --- | --- | --- |
| ItemID | SET\_OVH\_THRESHOLD\_ID | 0x0000 3B00 |  |
| Description | Set the threshold value of overheat protection. To confirm completion of this command, please call monitoring item(GET\_OVH\_THRESHOLD). | | |
| Update Rate | Real time | | |
| TE Related | No | | |
| Synchronization | Asynchronous | | |
| Data | 8Bytes | | |
| Byte0-3:  4 Byte integer ( DRXP board overheat protection.)  Valid value range is between 50 and 99 inclusive. Unit is Celsius.  LSByte first  Byte4-7: Reserved (undefined) | | |
| Remarks | This monitoring command is available with driver version 2.0.0 or later.  This monitor point is implemented only for production DRXP boards.  When driver is loaded, driver set this threshold to 90 degrees Celsius.  When the FPGA junction temperature is over this threshold, driver initiates forcibly reset.  This register is common between both production DRXP board IFs. Changing the register by on IF affects another IF.  ■Remarks for prototype DRXP board  < version 2.0.0 and 2.0.1>  When this command is called from prototype DRXP boards, it sets the threshold value of overheat protection. But it is ignored.  < version 2.0.3 to 2.0.7>  When this comamnd is called from prototype DRXP boards, ioctl returns -1 and sets ENOSYS to errno. | | |
| Errors | EINVAL  Specifiy the invalid value for threshold of overheat protection.  ENOSYS(version 2.0.3 to 2.0.7)  When this command is used with prototype DRXP boards. | | |

### SET\_DFR\_FORCETX

|  |  |  |  |
| --- | --- | --- | --- |
| ItemID | SET\_DFR\_FORCETX\_ID | 0x0000 2F00 |  |
| Description | This command controls transmitting PRBS31/PRBS63 pattern forcibly.  When DRXP is in reception mode, it can’t send data from Tx port normally. This function enables the debugging function for transmitting the PRBS31/PRBS63 pattern when it is in reception mode.  The PRBS31/PRBS63 generator is reset every 48 milliseconds. Thus, The same data pattern is repeated in 48milliseonds cycle.  If you use this function, please call this before calling ioctl(DRXPD\_RCVSTTART).  To confirm completion of this command, please call monitoring item(GET\_DFR\_FORCETX). | | |
| Update Rate | Real time | | |
| TE Related | No | | |
| Synchronization | Asynchronous | | |
| Data | 8Bytes | | |
| Byte0:  Enable/Disable this capability.  - DFRFORCETX\_ENABLE(0x01)  - DFRFORCETX\_DISABLE(0x02)  Above values are defined in the drxpd.h  If you set the value other than them, this function disables this capability.  Byte1-3:Reserved(undefined)  Byte4:  Select data source.  - DFRFORCETX\_PRBS31R(0x02) (\*1)  - DFRFORCETX\_PRBS31(0x03) (\*2)  - DFRFORCETX\_PRBS63(0x04) (\*2)  - DFRFORCETX\_PRBS31SR(0x05) (\*2)  - DFRFORCETX\_PRBS31S(0x06) (\*2)  - DFRFORCETX\_PRBS63S(0x07) (\*2)  Above value is defined in the drxpd.h.  For the detail fo data format aboves, please refer to the **“6.2.12.1　DFR\_FORCETX data format”**.  (\*1) In the device driver version 2.0.8 or older, it defined as DFRFORCETX\_PRBS. User application can also use this definition with version 2.0.9 or later.  (\*2) These definitions can be available with device driver version 2.0.9 or later.  Byte5-7: Reserved (undefined) | | |
| Remarks | This monitoring command is available with driver version 2.0.2 or later.  In the version 2.0.2 to 2.0.7, this monitoring item is only available with production DRXP boards. In the version 2.0.8 or later, it is available with production DRXP and prototype DRXP. | | |
| Errors | ENOSYS (version 2.0.2 to 2.0.7)  If this command is called with prototype DRXP board, it sets ENOSYS to an errno.  EBUSY  When the production DRXP is in transmission mode, this command sets EBUSY to an errno. And when it has already started receiving the data, it also sets EBUSY to an errno.  EIO  When this command fails to read or write internal register.  EINVAL  When you set an invalid value to the data byte4. | | |

#### DFR\_FORCETX data format

In the following, it explains the data format when DFR\_FORCETX capability is enabled. This capability uses PRBS binary sequence as a data source. A PRBS (pseudorandom binary sequence) is a periodic binary sequence. The period of PRBS31 is 2^31 -1. And the period of PRBS63 is 2^63 -1.

■DFRFORCETX\_PRBS31R(0x02)

When a user sets DFRFORCETX\_PRBS31R as a data source, the DRXP sends a PRBS31 pattern from each bit. And it resets the PRBS31 generator every 48ms. It means the DRXP sends the same data every 48ms.

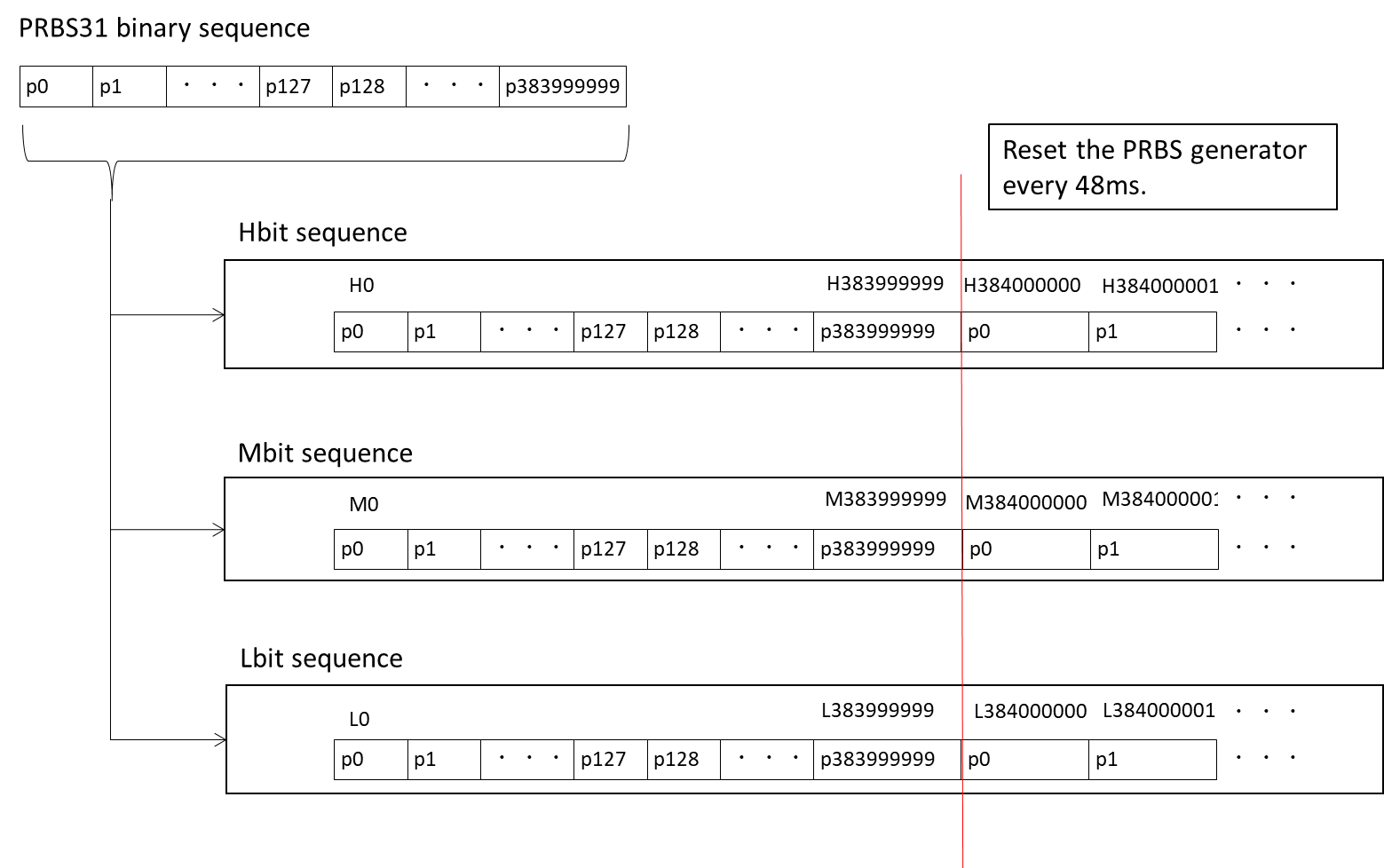


Figure 6‑1: Data format (DFRFORCETX\_PRBS31R) 1

p<n> represents one bit of PRBS31 binary sequence.

L/M/H<m> represents one bit of ALMA frame data.

<m> represents time order.

The following diagram represents how to generate this PRBS31 pattern. The data p0, p1, … is outputted from “OUT” in the below figure. When it resets the generator, it sets 1 to the all PRBS generator’s register.

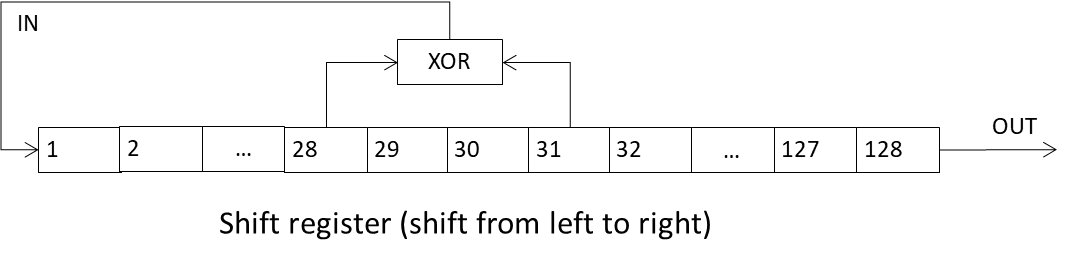


Figure 6‑2: PRBS31generator (128bits)

■DFRFORCETX\_PRBS31(0x03)

When a user sets DFRFORCETX\_PRBS31 as a data source, the DRXP sends a PRBS31 pattern from each bit. And it does not reset the PRBS31 generator. It sends the same PRBS31 binary sequence in a 2^31 -1bit cycle. The generator used is represented in **“Figure 6‑2: PRBS31generator (128bits)”**.

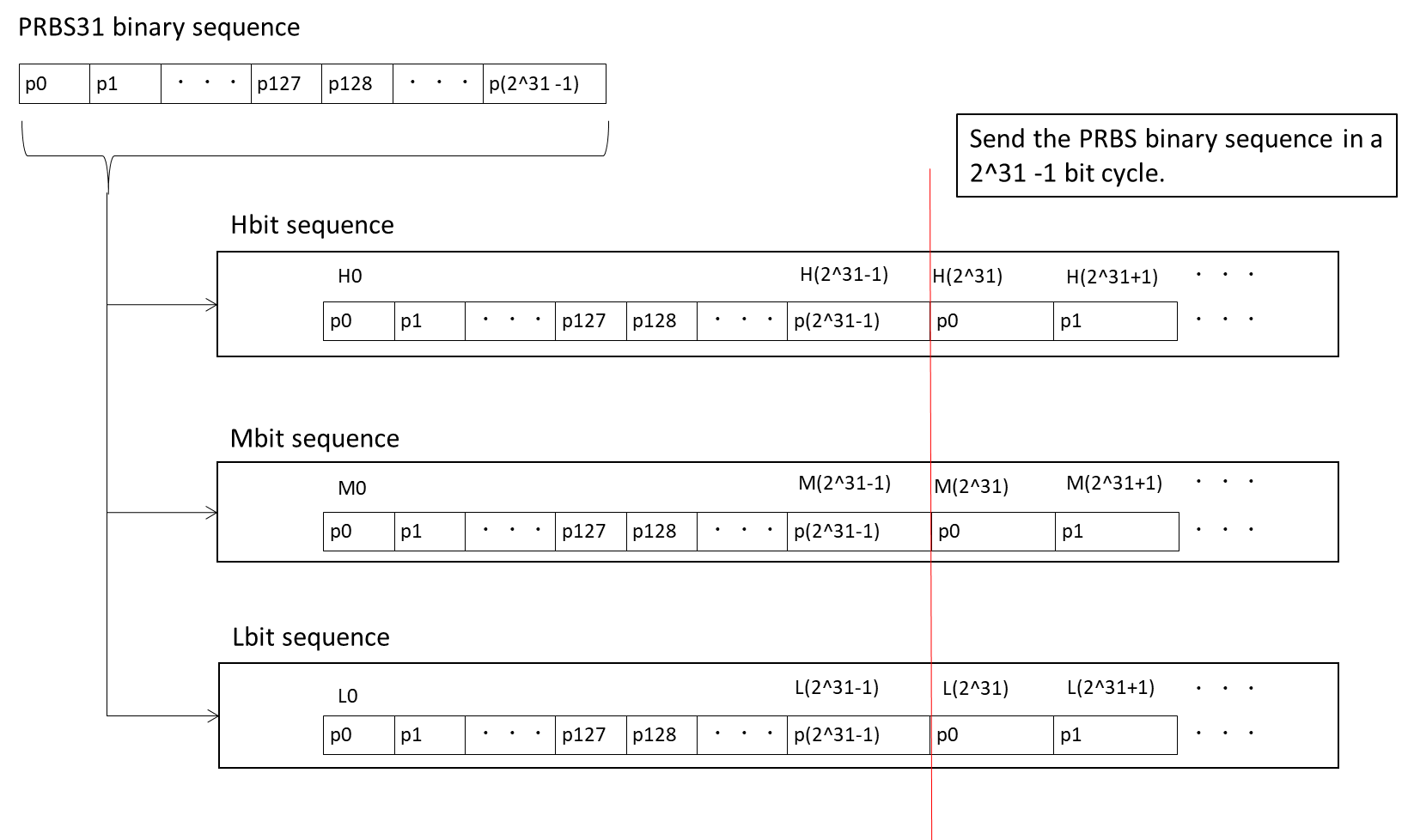


Figure 6‑3: Data format (DFRFORCETX\_PRBS31)

p<n> represents one bit of PRBS31 binary sequence.

L/M/H<m> represents one bit of ALMA frame data.

<m> represents time order.

■DFRFORCETX\_PRBS63(0x04)

In this case, the data format is similar to the DFRFORCETX\_PRBS31. The difference is it uses PRBS63 as a data generator. The following diagram represents how to generate this PRBS63 pattern. The data p0, p1, … is outputted from “OUT” in the below figure.

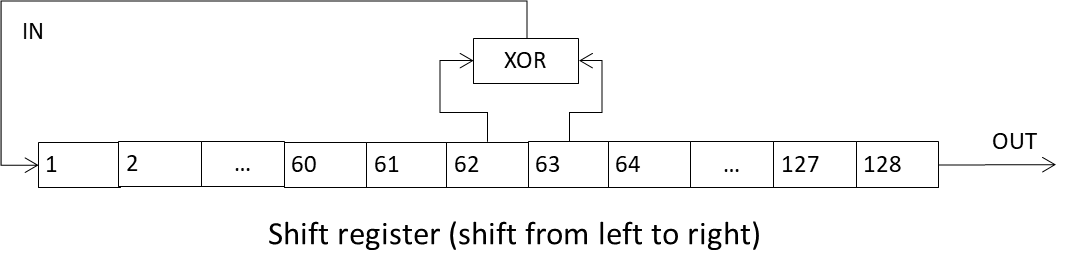


Figure 6‑4: PRBS63 generator (128bits)

■DFRFORCETX\_PRBS31SR(0x05)

When a user sets DFRFORCETX\_PRBS31SR as a data source, the DRXP sends a PRBS31 pattern. And it resets the PRBS31 generator every 48ms. It means the DRXP sends the same pattern every 48ms. The PRBS31 binary sequence is packed as following diagram.

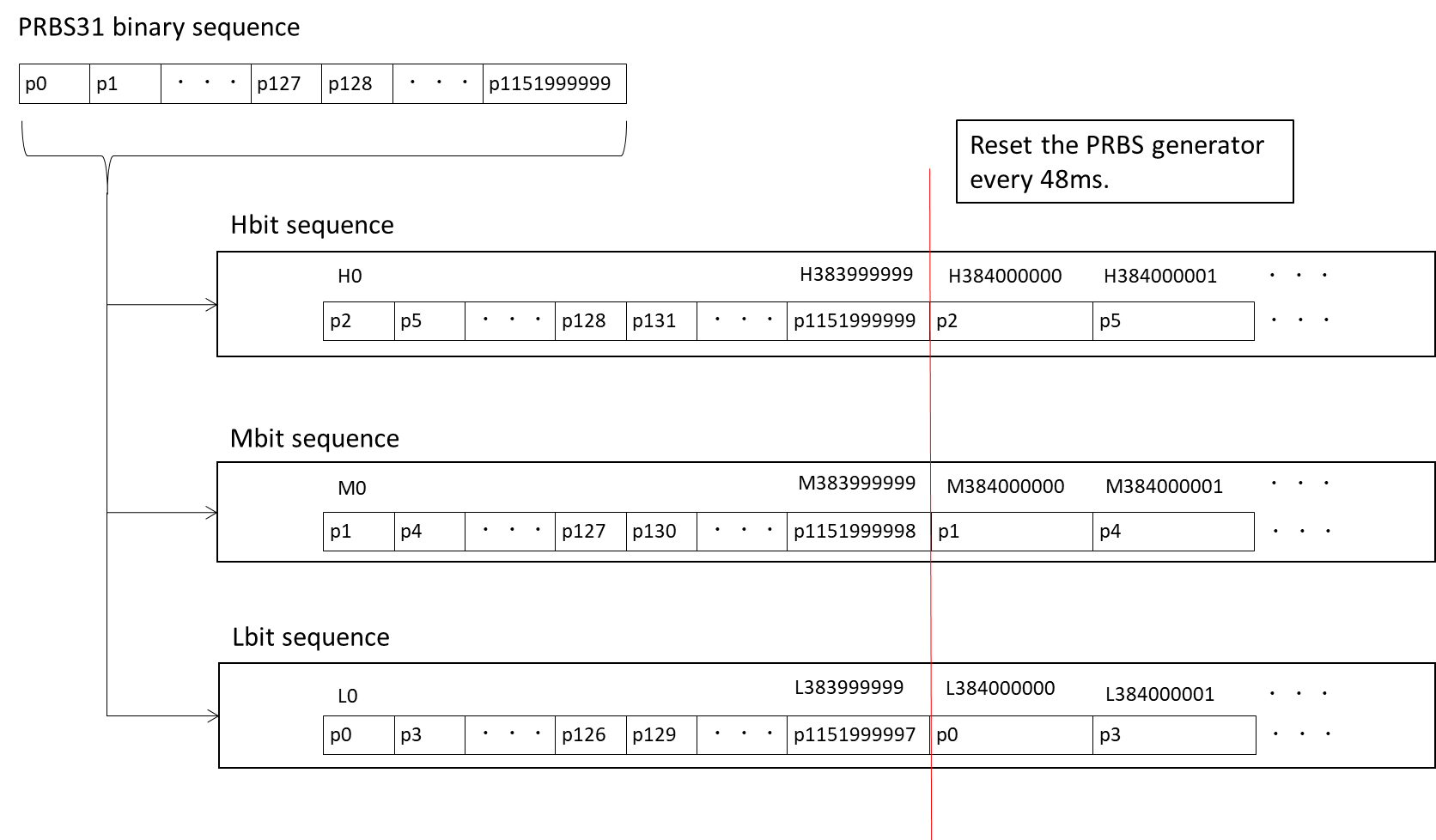


Figure 6‑5: Data format (DFRFORCETX\_PRBS31SR)

p<n> represents one bit of PRBS31 binary sequence.

L/M/H<m> represents one bit of ALMA frame data.

<m> represents time order.

The following diagram represents how to generate this PRBS31 pattern. When it resets the generator, it sets 1 to the all PRBS generator’s register. The data p0, p1, … is outputted from “OUT” in the below figure.

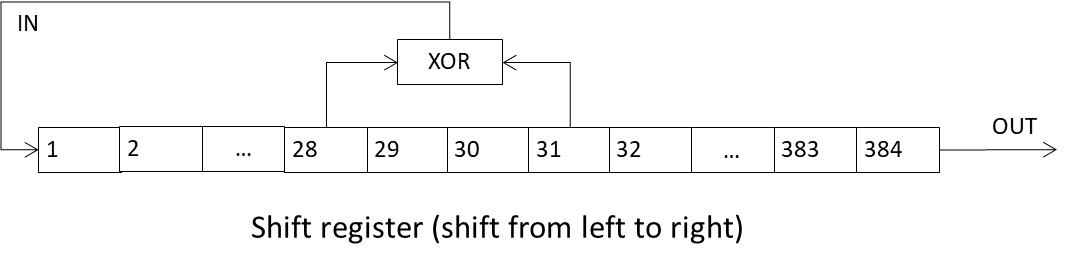


Figure 6‑6: PRBS31 generator (384 bits)

■DFRFORCETX\_PRBS31S(0x05)

When a user sets DFRFORCETX\_PRBS31S as a data source, the DRXP sends a PRBS31 pattern. And it does not reset the PRBS31 generator. It means the DRXP sends the same PRBS31 binary sequence in a 2^31-1 bit cycle. The PRBS31 binary sequence is packed as the following diagram. The generater used is represented in **“Figure 6‑6: PRBS31 generator (384 bits)”.**

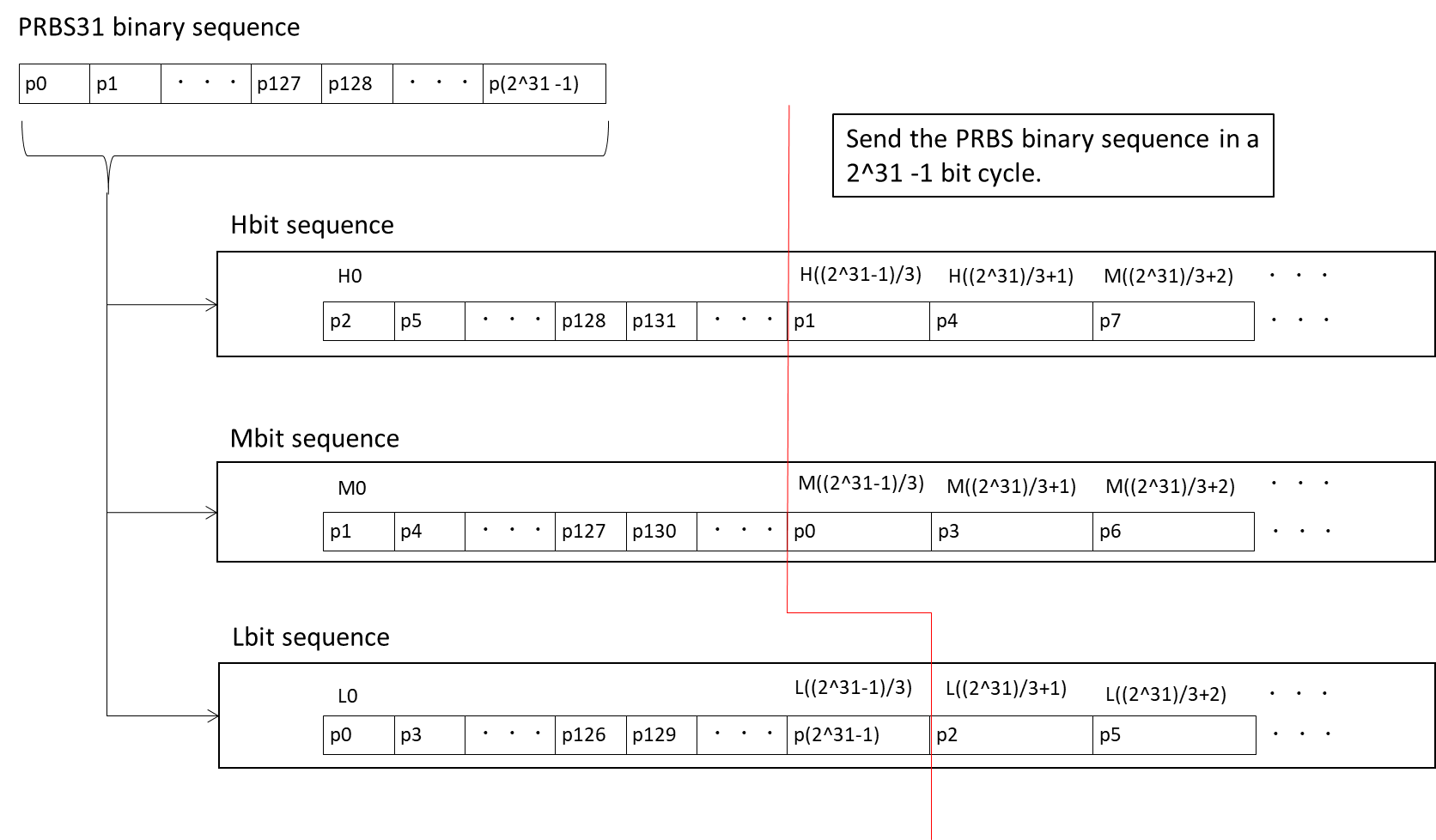


Figure 6‑7: Data format (DFRFORCETX\_PRBS31S)

p<n> represents one bit of PRBS31 binary sequence.

L/M/H<m> represents one bit of ALMA frame data.

<m> represents time order.

■DFRFORCETX\_PRBS63S(0x06)

This is similar to the DFRFORCETX\_PRBS31S. The difference is it uses PRBS63 as a data generator. The PRBS63 binary sequence period is 2^63 -1. The following diagram represents how to generate this PRBS63 pattern.

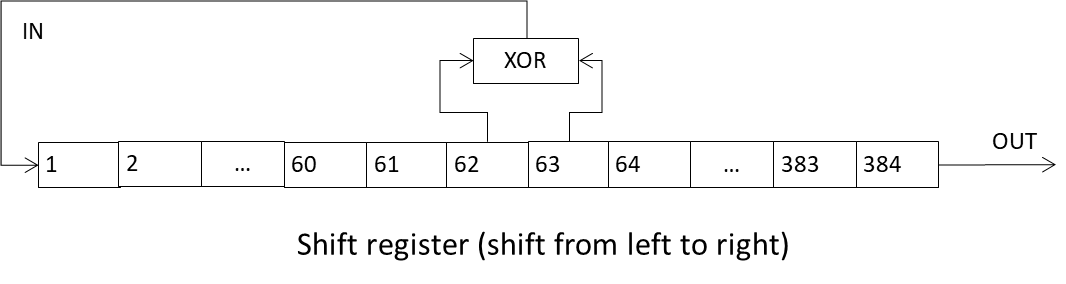


Figure 6‑8: PRBS63 generator (384 bits)

### SET\_PARITY\_RANGE

|  |  |  |  |
| --- | --- | --- | --- |
| ItemID | SET\_PARITY\_RANGE\_ID | 0x0000 2F01 |  |
| Description | This command sets how many frames the FGPA uses to generate the data parity.  The FPGA generates 16bit parity by using N ALMA frames. Your application can confirm the data integrity of internal FPGA logic. The valid range of N is from 1 to 3,000,000. This logic starts to generate parity when it detects a metaframe bit. | | |
| Update Rate | Real time | | |
| TE Related | No | | |
| Synchronization | Synchronous  Timeout  This command confirms the register after writing value. If it failed to change the register value, it sets EIO to errno. Normaly, this command returns less than 1ms. About the physical layer timeout value, please refer a section **“**Command :DRXPD INIT**5.5 ioctl”**. | | |
| Data | 8Bytes | | |
| Byte0-3:  ４Byte Integer  The valid value range is from 1 to 3,000,000.  The default value is 1.  　LSByte first  Byte4-7: Reserved (undefined) | | |
| Remarks | This monitoring command is available with driver version 2.0.3 or later.  In the version 2.0.3 to 2.0.7, this monitoring item is only available with production DRXP boards. In the version 2.0.8 or later, it is available with production DRXP and prototype DRXP. | | |
| Errors | ENOSYS (version 2.0.3 to 2.0.7)  If this command is called with prototype DRXP board, it sets ENOSYS to an errno.  EIO  When this command fails to read or write internal register.  EINVAL  When you set an invalid value to the data byte0-3.  EBUSY  When the resource busy.  The device has already started receiving data process. If you want to change this value, please stop receiving by ioctl(DRXPD\_RCVSTOP). After that, please try again. | | |

# Appendix

## SFP module warning and alarm threshold

Threshold values ​​shown in the following table are the excerpt from the data-sheet of the SFP+ module(Sumitomo electric SPP5200ER-GL).　 The DRXP board hardware and driver software cannot change the thresholds because these values are permanently defined in the ROM on SFP+ module. If another module is used, refer to its data sheet for the definition of thresholds.

|  |  |  |  |
| --- | --- | --- | --- |
| Meaning | The corresponding bit name　in GET\_ALARM\_STATUS\_REG　or GET\_POWER\_ALARM\_REG. | Unit | Threshold value |
| Temperature Warning | TempWARN | deg.C | Greater than or equal to 70 ,  smaller than or equal to 0. |
| Tx Bias Warning | TxBIASWARN | mA | Greater than or equal to 84,  smaller than or equal to 36. |
| Tx Power Warning | TxPOWWARN | dBm | Greater than or equal to 4,  smaller than or equal to -4.7. |
| Rx Power Warning | RxPOWWARN | dBm | Greater than or equal to -1,  smaller than or equal to -15.8. |
| Voltage Warning | 3.3V\_WARN | V | Greater than or equal to 3.465 ,  smaller than or equal to 3.135. |
| Temperature Alarm | TempALM | deg.C | Greater than or equal to 75 ,  smaller than or equal to -5. |
| Tx Bias Alarm | TxBIASALM | mA | Greater than or equal to 90,  smaller than or equal to 30. |
| Tx Power Alarm | TxPOWALM | dBm | Greater than or equal to 7,  smaller than or equal to –8.7. |
| Rx Power Alarm | RxPOWALM | dBm | Greater than or equal to 2,  smaller than or equal to -19.8. |
| Voltage Alarm | 3.3V\_ALM | V | Greater than or equal to 3.63 ,  smaller than or equal to 2.97. |

## Acronym and meanings of abbreviation

ALMA: Atacama Large Millimeter Array

API: Application Programming Interface

DMA: Direct Memory Access

DTS-R: Data Transmission System Receiver

DTS-T: Data Transmission System Transceiver

FPGA: Field Programmable Gate Array

GiB: gibibyte, 2^30 bytes

mmap: The LINUX command inquiring the virtual address of DMA.

MiB: mebibyte, 2^20 bytes

MSByte: the most significant byte

PCIe: A name of a computer bus, the PCI Express

SFP: Small Form-factor Pluggable, optical/electrical data transmission module

TAI: Temps Atomique International in French, International Atomic Time in English

TE: Timing Event

DFR: Deformatter

SEU: Single Event Upset

## SEU Mitigation

From prototype DRXP FPGA after 2018/01/05 11:29:33 SEU Mitigation function is implemented to it. It is also implemented to production DRXP. This function module monitors FPGA’s configuration memory. And it mitigates soft-error. Main features are bellow.

* **Main Features**

・Monitoring configuration memory.

(Block memory, distributed memory and flip-flop are not supported.)

・When this module detects soft-error, if possible, it corrects detected soft-error.

・Your application can receive soft-error event via eventfd.

・Your application can inject soft-error to configuration memory for testing.

Configuring the stage2 bit file, this function module starts automatically. Even if driver isn’t load, SEU Mitigation works. If this function detects soft-error, it tries to correct it. If it is correctable, this function corrects and informs the error event to you. If it is uncorrectable, this function informs error and stops monitoring. In this state (FPGA has uncorrectable error), any FPGA functions isn’t be guaranteed. In this situation, your application has two options. One is rebooting system (reconfiguring FPGA). Two is ignoring uncorrectable error and sending commands to FPGA to restart monitoring.

### Function detail

#### State

This function module has seven states. Your application can acquire SEU Mitigation module status by DRXPD\_GET\_MONSTS command (GET\_SEU\_STATUS\_ID). When stage2 bit file is configured, this module starts working at “Initialization” state. If it operates normally, state transits to “Observation” state. In this state, module keeps scanning configuration memory. If it detects soft-error, its state transits to “Correction” state. If the detected error is correctable, this module corrects the error and returns to “Observation” state. If the detected error is uncorrectable, the state transit to “Idle” state. In “Idle” state, this module doesn’t scan configuration memory data. If FPGA has uncorrectable error, any FPGA functions aren’t guaranteed. This module has another state, “Diagnostic Scan” state, for scanning configuration memory. Diagnostic Scan state scans whole configuration memory one times and counts soft-error. Your application can change its state by DRXPD\_SET\_CONTROL command (SET\_SEU\_STATUS\_ID). About more description of each state, please refer to the following sections (Initialization, Observation, Correction, Idle, Error Injection, Diagnostic Scan, Error).

Bellow figure is state transition diagram.

Initialized

Correct error

Detect error

Command

(SET\_SEU\_STATUS)

Command

(SET\_SEU\_STATUS)

Uncorrectable error

Command

(SET\_SEU\_INJERR)

Command

(SET\_SEU\_STATUS)

Finish Injection

Finish scan

From any state

Figure 7‑1: SEU Mitigation module state transition diagram

Red arrow is transition by user application command.

Black arrow is transition by this module.

#### Initialization

After configuring the stage2 bit file for FPGA, this module automatically initialized. When the initialization is done, its state transits to “Observation” state. If not, this module has a fatal error. Even if this module doesn’t work normally, other functions like receiving frames, transmitting frames, are available.

#### Observation

In this state, this module keeps monitoring configure memory. If it detects soft-error, it transits to “Correction” state. And in this state, it counts elapsed time (the counter uses onboard crystal). If your application wants to know how many times this module encountered soft-error, you can acquire it by DRXPD\_GET\_MONSTS command (GET\_SEU\_ERRCNT\_ID). If you want to stop “Observation” state, you can change its state by DRXPD\_SET\_CONTROL command (SET\_SEU\_STATUS\_ID). In this state, this module can transit to “Idle” state by user command.

#### Correction

In this state, this module tries to correct soft-error. If it is correctable, this module corrects it and moves its states to “Observation” state. If it is uncorrectable, it transits to “Idle” state.

#### Idle

In this state, this function doesn’t scan configuration memory and correct memory. Your application can change the state by DRXPD\_SET\_STATUS command (SET\_SEU\_STATUS\_ID). In this state, this module can transit to “Observation”, “Error Injection”, “Diagnostic Scan” and “Detect Only” state by user command.

#### Error Injection

When module is in “Idle” state, your application can send error injection command. It moves module state to “Error Injection” state. After completing error injection it transits to “Idle” state. Please start injecting error after confirming its state backing to “Idle” state to inject an error again.

#### Diagnostic Scan

In “Idle” state, your application can transit to “Diagnostic Scan” state. In this state, this module scans all configuration memory and counts all detected error. And this state doesn’t correct detected error. After it finishes scanning, its state returns to “Idle” state. Your application can acquire detected error count by using DRXPD\_GET\_MONSTS command (GET\_SEU\_DIAERRCNT\_ID). This state differs from “Observation” state on the following two points. It doesn’t correct an error and doesn’t stop scanning if it finds uncorrectable error.

Note: GET\_SEU\_DIAERRCNT\_ID is not available with the driver version 2.0.2 or older.

Note: In the device driver version 2.0.7 or older, “Diagnostic Scan state” is not available with the prototype DRXP boards.

#### Error

If this module detects internal inconsistency, this module transits to “Error” state. This state is unrecoverable. Please reconfigure FPGA to recover the SEU Mitigation function.

### Usage

#### Monitoring the SEU event

This section describes the standard usage for monitoring the SEU event.

Step1.

First of all, please confirm that the SEU module is in “Observation” state. Your application can read the state of SEU module by using DRXPD\_GET\_MONSTS(GET\_SEU\_STATUS\_ID) command. This module is automatically initialized (“Initialization” state) and starts monitoring configure memory (“Observation” state). If it stays in “Initialization” state, it has fatal error.

Step2.

Please check the SEU module status and soft-error counter periodically. Your application can read the states and error by using DRXPD\_GET\_MONSTS(GET\_SEU\_STATUS\_ID) command. And it can read the error-counter by using DRXPD\_GET\_MONSTS(GET\_SEU\_ERRCNT\_ID) command.

* Ｔhe state transits to ”Idle” state from “Observation” state

This means the SEU module detected uncorrectable error and it shifted own state to “Idle” state. In this situation, FPGA has uncorrectable error bit in configuration memory. Please reset your system and reconfigure FPGA to recover from this situation.

* Ｔhe state is in “Observation” state and the correctable error counter is incremented.

This means the SEU module detected correctable error and repaired it.

* Ｔhe state is in “Observation” state and the error counter isn’t incremented.

This means there is no error during a polling period.

* Error status is detected ( DRXPD\_GET\_MONSTS(GET\_SEU\_STATUS\_ID) )

You can read the SEU module status by DRXPD\_GET\_MONSTS(GET\_SEU\_STATUS\_ID) command. This command returns the status in data structure (ST\_SEU\_STATUS ). A member of this data, “uiError”, shows SEU fuction module’s error. If the error bit is set to “uiError”, The SEU module has fatal error. Please reset your system and reconfigure FPGA to recover this situation.

#### Testing the SEU function module

This section describes the method for testing the SEU function.

Step1.

First of all, please confirm that the SEU module is in “Observation” state. Your application can read the state of SEU module by using DRXPD\_GET\_MONSTS(GET\_SEU\_STATUS\_ID) command. This module is automatically initialized (“Initialization” state) and starts monitoring configure memory (“Observation” state). If it stays in “Initialization” state, it has a fatal error.

Step2.

Please shift the SEU module state to “Idle” state from “Observation” state. Your application can change the SEU module state by using DRXPD\_SET\_CONTROL(SET\_SEU\_STATUS\_ID) command.

Step3.

Please confirm the SEU module error counter. Your application can read the error-counter by using DRXPD\_GET\_MONSTS(GET\_SEU\_ERRCNT\_ID) command.

Step4.

Please inject bit errors to the FPGA. Your application can inject errors to the FPGA by using DRXPD\_SET\_CONTROL(SET\_SEU\_STATUS\_ID) command.

Step5.

Please wait for the SEU function module status returns to “Idle” state.

Step6.

Please shift the SEU module state to “Observation” state from “Idle” state. Your application can change the SEU module state by using DRXPD\_SET\_CONTROL(SET\_SEU\_STATUS\_ID) command.

Step7.

Please confirm the SEU module error counter. The SEU module should detect a correctable error and increment a counter.

#### Dignostic scan

This section describes the usage of “Diagnostic Scan”.

Step1.

First of all, please confirm that the SEU module is in “Observation” state. Your application can read the state of SEU module by using DRXPD\_GET\_MONSTS(GET\_SEU\_STATUS\_ID) command. This module is automatically initialized (“Initialization” state) and starts monitoring configure memory (“Observation” state). If it stays in “Initialization” state, it has a fatal error.

Step2.

Please shift the SEU module status to “Idle” state from “Observation” state. Your application can change the SEU module state by using DRXPD\_SET\_CONTROL(SET\_SEU\_STATUS\_ID) command.

Step3.

Please shift the SEU module status to “Diagnostic Scan” state from “Idle” state. Your application can change the SEU module state by using DRXPD\_SET\_CONTROL(SET\_SEU\_STATUS\_ID) command.

Step4.

Please wait for the SEU function module state returns to “Idle” state.

Step5.

Please confirm the SEU module diagnostic scan error counter. Your application can read diagnostic scan error counter by using DRXPD\_GET\_MONSTS(GET\_SEU\_DIAERRCNT\_ID) command.

Note: In the case of production DRXP, dignostic scan is available in device driver version 2.0.3 or later.

Note: In the case of prototype DRXP, dignostic scan is available in device driver version 2.0.8 or later.

## The overheating protection

The overheating protection is implemented to a production DRXP board. This function has three thresholds for protecting board.

* User defined threshold
* Hardware threshold
* Shutdown threshold

The production DRXP board has the flag represents it is in overheat state. When the FPGA junction temperature exceeds the threshold, this flag is asserted. And it triggers the overheating protection. This flag prevents the state transition to receiving (transmitting) data until it is de-asserted. (This flag prevents the execution of ioctl(DRXPD\_INIT)). This flag is de-asserted when the FPGA junction temperature falls below (user defined threshold – 5) degrees Celsius. You can get the status of this flag by calling monitoring command (GET\_OVH\_FLG). And you also can get/set the user defined threshold by calling GET/SET\_OVH\_THRESHOLD. The hardware threshold and the shutdown threshold is fixed value. User can’t change these values. The protection method is different in each threshold.

* User defined threshold

The default threshold value is 90 degrees Celsius. When the FPGA junction temperature exceeds this threshold, the DRXP driver initiates reset of logic. It means that the function of receiving/transmitting ALMA frame is forcibly stopped. And driver powers off the transceiver and disable Tx ports of SFP+ modules for decreasing power consumption. The event of overheating protection is logged in the EEPROM on the production DRXP board. In reception mode, when overheating flag is asserted, user application can reads “100” from the eventfd. While the flag is asserted, DRXP driver writes the syslog every 10 seconds. When you notice the overheating, please close device file and shuts down your machine.

* Hardware threshold

The threshold value is 95 degrees Celsius. When the FPGA junction temperature exceeds this threshold, FPGA initiates IF reset. It means that the function of receiving/transmitting ALMA frame is forcibly stopped. Even if a DRXP driver isn’t loaded, this function works. If a DRXP driver is loaded, the deriver monitors the overheating register in the FPGA. And when it detects the hardware protection, the event of overheating protection is logged in the EEPROM on the production DRXP board. In reception mode, when overheating flag is asserted, user application can reads “101” from the eventfd (\*1). While the flag is asserted, DRXP driver writes the syslog every 10 seconds. When you notice the overheating, please close device file and shuts down your machine. This function is enabled after configuring FPGA stage2.

(\*1) In the case of driver version from 2.0.0 to 2.0.2, eventfd returns “100”.

* Shutdown threshold

The threshold value is 100 degrees Celsius. When the FPGA junction temperature exceeds this threshold, DRXP driver initiates system shutdown. The event of overheating protection is logged in the EEPROM on the production DRXP board. When the FPGA junction temperature exceeds shutdown threshold, device driver calls kernel\_power\_off in the kernel space. This function shutdowns everything and performs a clean system power\_off. But this fuction doesn’t synchronize file cache data. When device driver detects this overheating event, it turns off machine without call sync system-call (\*1).

(\*1) In the case of driver version 2.0.6

When the FPGA junction temperature exceeds shutdown threshold, device driver just calls shutdown command. This command calls sync system-call before calling kernel\_power\_off internaly. The device driver calls shutdown command with following options. The following shutdown command sends message to all logged-in users and schedules shutdown after 1 minute.

‘shutdown –h +1 “drxpd driver detects overheating of device. ”’

For the detail of this command and option, please refer command manual. (Please execute following command. $ man 8 shutdown.).

\* The FPGA junction temperature exceeds user-defined threshold.

(The user-defined threshold range is from 50degC to 99degC).

Or

\* The FPGA junction temperature exceeds hardware threshold.

(The hardware threshold is 95degC.)

The FPGA junction temperature falls below user-defined (threshold-5) degC.

(At least it is less than 94degC.)

From any state..

The FPGA junction temperature exceeds 99degC.

The device driver powers off the system for protecting the devices.

Figure 7‑2: Overheat state transition diagram

### Log (version 2.0.6 or older)

When a DRXP driver detects the overheating event, it writes the log to the EEPROM on the DRXP board.

It writes the log following timings.

* The FPGA junction temperature exceeds the user defined threshold.
* The FPGA junction temperature falls below user defined threshold.
* The FPGA junction temperature exceeds the shutdown threshold.
* The FPGA junction temperature exceeds the hardware threshold.

You can read the overheat log from syslog. A DRXP driver dumps logs to syslog when it is loaded or above four conditions are met. You can also force device driver to read overheat log from the DRXP board and dump it to syslog manually by calling ioctl(DRXPD\_FLASH\_OVHLOG). The log examples are as below.

When the overheating protection is triggered, the DRXP driver writes the time when it detects overheat and the FPGA junction temperature. Each red rectangle area is written at the same timing. Timestamp uses UTC +0000. DRXP can log 30 events. If the all event entries are filled, device driver overwrites the oldest entry. If the event entries are not filled, DRXP driver dumps “no log entry”.

Apr 2 02:21:39 localhost kernel: drxpd{#0} : log entry[1] : 2019-04/01 11:16:30 overheat was detected.

Apr 2 02:21:39 localhost kernel: drxpd{#0} : log entry[2] : xxxx-xx/xx xx:xx:xx protection was triggered at 66 degC.

Apr 2 02:21:39 localhost kernel: drxpd{#0} : log entry[3] : 2019-04/01 11:18:31 returned from overheat state.

Apr 2 02:21:39 localhost kernel: drxpd{#0} : log entry[4] : 2019-04/01 11:23:02 overheat was detected.

Apr 2 02:21:39 localhost kernel: drxpd{#0} : log entry[5] : xxxx-xx/xx xx:xx:xx protection was triggered at 66 degC.

Apr 2 02:21:39 localhost kernel: drxpd{#0} : log entry[6] : 2019-04/01 11:24:12 returned from overheat state.

Apr 2 02:21:39 localhost kernel: drxpd{#0} : log entry[7] : 2019-04/01 12:14:03 overheat was detected.

Apr 2 02:21:39 localhost kernel: drxpd{#0} : log entry[8] : xxxx-xx/xx xx:xx:xx protection was triggered at 66 degC.

Apr 2 02:21:39 localhost kernel: drxpd{#0} : log entry[9] : 2019-04/01 12:15:03 returned from overheat state.

Apr 2 02:21:39 localhost kernel: drxpd{#0} : log entry[10] : 2019-04/01 12:17:54 overheat was detected.

Apr 2 02:21:39 localhost kernel: drxpd{#0} : log entry[11] : xxxx-xx/xx xx:xx:xx protection was triggered at 66 degC.

Apr 2 02:21:39 localhost kernel: drxpd{#0} : log entry[12] : 2019-04/01 12:20:04 returned from overheat state.

Apr 2 02:21:39 localhost kernel: drxpd{#0} : log entry[13] : 2019-04/01 12:22:55 overheat was detected.

Apr 2 02:21:39 localhost kernel: drxpd{#0} : log entry[14] : no log entry.

Apr 2 02:21:39 localhost kernel: drxpd{#0} : log entry[15] : no log entry.

### Log (version 2.0.7 or later)

When a DRXP driver detects the overheating event, it writes the log to the EEPROM on the DRXP board. It writes the log following timings.

* The FPGA junction temperature exceeds the user defined threshold.
* The FPGA junction temperature falls below user defined threshold.
* The FPGA junction temperature exceeds the shutdown threshold.
* The FPGA junction temperature exceeds the hardware threshold.

You can read the overheat log from syslog. A DRXP driver dumps logs to syslog when it is loaded or above four conditions are met. You can also force device driver to read overheat log from the DRXP board and dump it to syslog manually by calling ioctl(DRXPD\_FLASH\_OVHLOG). The log examples are as below. Timestamp uses UTC +0000. DRXP can log up to 30 events. If the all event entries are filled, device driver overwrites the oldest entry. If the event entries are not filled, DRXP driver only dumps logged entry.

drxpd{#0} : Begin dumping on-board log.

(snip)

drxpd{#0} : 2020-06/10 07:07:05, Shutdown overheating protection was triggered at the FPGA junction temperature, 65 degC.

drxpd{#0} : 2020-06/10 07:07:20, Shutdown overheating protection was triggered at the FPGA junction temperature, 66 degC.

drxpd{#0} : 2020-06/10 07:36:19, Shutdown overheating protection was triggered at the FPGA junction temperature, 67 degC.

drxpd{#0} : 2020-06/16 09:08:35, Overheating protection was triggered at the FPGA junction temperature, 91 degC.

drxpd{#0} : 2020-06/30 22:06:43, User-define overheating protection was triggered at the FPGA junction temperature, 69 degC.

drxpd{#0} : 2020-06/30 22:27:29, User-define overheating protection was triggered at the FPGA junction temperature, 92 degC.

drxpd{#0} : 2020-06/30 22:28:28, User-define overheating protection was triggered at the FPGA junction temperature, 91 degC.

drxpd{#0} : 2020-06/30 22:29:19, Hardware overheating protection was triggered at the FPGA junction temperature, 96 degC.

drxpd{#0} : 2020-06/30 22:29:29, User-define overheating protection was triggered at the FPGA junction temperature, 95 degC.

drxpd{#0} : 2020-06/30 22:29:42, User-define overheating protection was triggered at the FPGA junction temperature, 95 degC.

drxpd{#0} : 2020-06/30 22:30:03, Hardware overheating protection was triggered at the FPGA junction temperature, 96 degC.

(snip)

drxpd{#0} : End dumping on-board log.

There are 6 type log messages. Followings are some brief description of these log and examples.

1. **The overheating event was detected.**

In the case of driver version from 2.0.0 to 2.0.6, when the FPGA junction temperature exceeded the user-defined threshold or hardware threshold, DRXP driver write this log message to the EEPROM. This message is logged to on-board EEPROM and syslog.

e.g. drxpd{#0} : 2020-06/16 09:08:35, Overheating protection was triggered at the FPGA junction temperature, 91 degC.

1. **The overheating event was detected (user-defined).**

When the device driver detects overheating event caused by the FPGA junction temperature exceeds the user-defined threshold. This log message is available in the device driver version 2.0.7 or later. This message is logged to on-board EEPROM and syslog.

e.g. drxpd{#0} : 2020-07/30 22:06:43, User-define overheating protection was triggered at the FPGA junction temperature,

69 degC.

1. **The overheating event was detected (hardware).**

When the device driver detects overheating event caused by the FPGA junction temperature exceeds the hardware threshold. This log message is available in the device driver version 2.0.7 or later. This message is logged to on-board EEPROM and syslog.

e.g. drxpd{#0} : 2020-07/30 22:29:19, Hardware overheating protection was triggered at the FPGA junction temperature,

96 degC.

1. **The FPGA junction temperature when the device is in overheating state.**

When the device driver detects overheating event, it writes this log message. This message is logged to syslog every 10 seconds.

e.g. drxpd{#0} : 2020-07/30 22:31:33, The FPGA junction temperature, 95 degC, is above the user-defined

threshold temperature, 90 degC. Overheating was detected.

e.g. drxpd{#0} : 2020-07/30 22:31:33, The FPGA junction temperature, 95 degC, is above the hardware

threshold temperature, 90 degC. Overheating was detected.

1. **The overheating event was detected (shutdown).**

When the device driver detects overheating event caused by the FPGA junction temperature exceeds the shutdown threshold. This log message is available in the device driver version 2.0.7 or later. This message is logged to on-board EEPROM and syslog.

e.g. drxpd{#0} : 2020-06/10 07:36:19, Shutdown overheating protection was triggered at the FPGA junction temperature,

101 degC.

1. **The DRXP returned from overheat state.**

When the device driver detects the FPGA junction temperature falls below user-defined threshold -5 degC. This log message is available in the device driver version 2.0.7 or later. This message is logged to on-board EEPROM and syslog.

e.g. drxpd{#0} : 2020-06/10 07:36:19, returned from overheat state.

### Confirm shutdown threshold (version 2.0.7 or later)

If you want to check the function of overheating protection safely, you can user kernel module parameter for this purpose. You can add temperature offset to the FPGA junction temperature. You can confirm shutdown threshold behavior by using this parameter. Please specify the offset value, when you load the device driver.

# insmod drxpd.ko gi\_tmep\_offset=<num>

e.g. # insmod drxpd.ko gi\_temp\_offset=25

The <num> is an integer. Please don’t specify the negative value. If you specify 25 to this option, device driver add 25 to the actually FPGA junction temperature before deciding whether it exceeds threshold or not. After checking function, please reload device driver with no kernel module parameter.

### Oveaheat flag

A device driver monitors whether the FPGA junction temperature exceeds user-defined threshold and whether hardware temperature alarm register is asserted. When the condition of one or both monitoring points are fulfilled, overheat flag is set. Overheat flag is cleared, when the FPGA junction temperature falls below (user defined threshold – 5) degrees Celsius.

## The parity for checking FPGA data integrity

During the development of the Production DRXP, it was found that data in the Host Ring Buffer for IF2 was sometimes corrupted while it was receiving a known data pattern over optical fibers. As a result of debug, the data decoded by the de-formatter seemed to be correct. Therefore, it was thought that the data got corrupted somewhere between the de-formatter and the Host Ring Buffer. The root cause has not been identified yet, but it is likely to be a timing synchronization issue because the probability of the data corruption changes when the FPGA code is re-compiled or modified. Some versions of the FPGA has not experienced this problem. Unfortunately, without parity, such data corruption cannot be detected unless you exactly know the data pattern the Production DRXP receives.

To allow a user application to detect such data corruption in real-time without knowing the exact data pattern, the de-formatter within the FPGA generates a 16-bit horizontal parity for each H(D)/M(C)/L(B) bit every 48ms. The FPGA finally transfers those parities to the information frame in the Host Ring Buffer. The user application should calculate the same parity from the data in the normal frames and compare it with the one calculated by the FPGA to confirm that there is no data corruption.

Because the user application may not be able to calculate the parity of all three million ALMA frames within 48ms due to limited calculation performance of CPU(s), the number of ALMA frames to be used to generate the parity, which is denoted as N below, can be changed by **“6.2.13 SET\_PARITY\_RANGE”**. When the de-formatter detects a metaframe bit, it starts to generate 16-bit horizontal parity for each ALMA frame using the formula shown below.

One frame parity = [143:128] xor [127:112] xor [111:96] xor [95:80]

xor [79:64] xor [63:48] xor [47:32] xor ([31:20] & [15:12])

\* [x:y] is the bit address range within one ALMA frame

The de-formatter internally stores the 16-bit horizontal parity of the first ALMA frame (the frame which has an active metaframe bit) as “accumulated parity” (= acc\_parity). As a new ALMA frame comes, the de-formatter calculates 16-bit horizontal parity, calculates XOR of the 16-bit horizontal parity and the accumulated parity and store the result as a new accumulated parity. It repeats the operation until N frames are processed. Finally, the accumulated parity is transferred to the associated information frame in the Host Ring Buffer.

|  |  |  |
| --- | --- | --- |
| **N** | **One frame parity** | **Parity** |
| 1 | parity(1) | acc\_parity(1) = parity(1) |
| 2 | parity(2) | acc\_parity(2) = parity(2) xor acc\_parity(1) |
| 3 | parity(3) | acc\_parity(3) = parity(3) xor acc\_parity(2) |
| 4 | parity(4) | acc\_parity(4) = parity(4) xor acc\_parity(3) |
| 5 | parity(5) | acc\_parity(5) = parity(5) xor acc\_parity(4) |
| … | | |
| 2,999,998 | parity(2,999,998) | acc\_parity(2,999,998) = parity(2,999,998) xor acc\_parity(2,999,997) |
| 2,999,999 | parity(2,999,999) | acc\_parity(2,999,999) = parity(2,999,998) xor acc\_parity(2,999,998) |
| 3,000,000 | parity(3,000,000) | acc\_parity(3,000 ,000) = parity(3,000 ,000) xor acc\_parity(2,999,999) |

Table 7‑1: N frame parity

A user application should calculate the same 16-bit horizontal parity from the data in the normal frames. It must match with the one generated by the de-formatter. Inconsistent parities indicate that there was data corruption somewhere between the de-formatter to the Host Ring Buffer.

For the convenience of application developers who may not know the detail of the ALMA frames, an example algorithm to calculate the parity is presented below. A user application should first re-arrange the data in DRXP F2 format; take the data of the first frame (the first 48 bytes in the group in the Host Ring Buffer), split it into each H/M/L bit and re-order the bits in time order as shown below. The re-ordered data can be viewed as eight 16-bit integers for each H/M/L bit (e.g. H0 – H15, H16 – H31, …, H112 – H127). Then, the user application should calculate XOR of all those eight integers, which generates the parity of that frame. If N is larger than one, the same calculation should be repeated for the successive frames to generate parities for each frame, and then, calculate XOR of the parities. The final result must be the same as the one generated by the de-formatter in the information frame unless the data is corrupted.

|  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- |
|  | Bit7 | Bit6 | Bit5 | Bit4 | Bit3 | Bit2 | Bit1 | Bit0 |
| Byte0 | M1 | L1 | H16 | M16 | L16 | H0 | M0 | L0 |
| Byte1 | L18 | H2 | M2 | L2 | H17 | M17 | L17 | H1 |
| Byte2 | H19 | M19 | L19 | H3 | M3 | L3 | H18 | M18 |
| Byte3 | M5 | L5 | H20 | M20 | L20 | H4 | M4 | L4 |
| Byte4 | L22 | H6 | M6 | L6 | H21 | M21 | L21 | H5 |
| Byte5 | H23 | M23 | L23 | H7 | M7 | L7 | H22 | M22 |
| Byte6 | M9 | L9 | H24 | M24 | L24 | H8 | M8 | L8 |
| Byte7 | L26 | H10 | M10 | L10 | H25 | M25 | L25 | H9 |
| ・・・・・・・・・・・・・ | | | | | | | | |
| Byte43 | L122 | H106 | M106 | L106 | H121 | M121 | L121 | H105 |
| Byte44 | H123 | M123 | L123 | H107 | M107 | L107 | H122 | M122 |
| Byte45 | M109 | L109 | H124 | M124 | L124 | H108 | M108 | L108 |
| Byte46 | L126 | H110 | M110 | L110 | H125 | M125 | L125 | H109 |
| Byte47 | H127 | M127 | L127 | H111 | M111 | L111 | H126 | M126 |

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| H15 | H14 | H13 | ・・・ | H2 | H1 | H0 |
| H31 | H30 | H29 | ・・・ | H18 | H17 | H16 |
| H47 | H46 | H45 | ・・・ | H34 | H33 | H32 |
| H63 | H62 | H61 |  | H50 | H49 | H48 |
| ・・・・・・・・・・・・・ | | | | | | |
| H111 | H110 | H109 | ・・・ | H98 | H97 | H96 |
| H127 | H126 | H125 | ・・・ | H114 | H113 | H112 |

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| M15 | M14 | M13 | ・・・ | M2 | M1 | M0 |
| M31 | M30 | M29 | ・・・ | M19 | M18 | M17 |
| M47 | M46 | M45 | ・・・ | M34 | M33 | M32 |
| M63 | M62 | M61 | ・・・ | M50 | M49 | M48 |
| ・・・・・・・・・・・・・ | | | | | | |
| M111 | M110 | M109 | ・・・ | M98 | M97 | M96 |
| M127 | M126 | M125 | ・・・ | M114 | M113 | M112 |

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| L15 | L14 | L13 | ・・・ | L2 | L1 | L0 |
| L31 | L30 | L29 | ・・・ | L19 | L18 | L17 |
| L47 | L46 | L45 | ・・・ | L34 | L33 | L32 |
| L63 | L62 | L61 | ・・・ | L50 | L49 | L48 |
| ・・・・・・・・・・・・・ | | | | | | |
| L111 | L110 | L109 | ・・・ | L98 | L97 | L96 |
| L127 | L126 | L125 | ・・・ | L114 | L113 | L112 |

Note that the algorithm presented above is not optimal in terms of calculation time. The above example algorithm is presented only to help the application developers quickly understand the concept.

## Device file attribution

In the device driver version 2.0.5 or later, the attributions information are added to the device file. User application can identify the correspondence between IF and device file by using this attribution. Following attributions are added.

|  |  |
| --- | --- |
| **Attribution name** | **Description (value)** |
| serial | The serial number of the board. |
| if\_nr | 1: IF1 (or prototype DRXP), 2: IF2 |
| bus | The bus number of DRXP board. |
| boardtype | “Prototype” or “Product” |

Table 7‑2: The device file attribution

User can check the attribution information of “/dev/drxpd0”by following command.

# udevadm info -a /sys/class/drxpd0/drxpd0

(snip)

looking at device '/devices/pci0000:5d/0000:5d:00.0/0000:5e:00.0/drxpd0/drxpd0':

KERNEL=="drxpd0"

SUBSYSTEM=="drxpd0"

DRIVER==""

ATTR{if\_nr}=="1"

ATTR{serial}=="134492"

ATTR{bus}=="8d"

ATTR{boardtype}=="Product"

(snip)

### Udev rule

User can create an arbitrary symbolic link by using udev rule and device file attribution. Following is the example of udev rule.

$cat /etc/udev/rules.d/99-example1.rules

KERNEL=="drxpd\*", SUBSYSTEM=="drxpd\*", SYMLINK+="drxp-%s{serial}-%s{boardtype}-if%s{if\_nr}-bus%s{bus} "

In the case of above rule, OS detects the creation of device file “/dev/drxpd\*”, it creates symbolic link to the “/dev/drxpd\*” automatically. %s{serial}, %s{boardtype}, %s{if\_nr} and %s{bus} are replaced with device file attribution of “/dev/drxpd\*”. User application can control/monitor DRXP board via this symbolic link. And if your system has correspondence between bus number and physical slot, you may get physical slot information via “dmidecode” command and add it to symbolic link name. You can call external program by using “PROGRAM” field. Followings are the example of udev rule , sub-script and the output of dmidecode.

$cat /etc/udev/rules.d/99-example2.rules

KERNEL=="drxpd\*", SUBSYSTEM=="drxpd\*", \

PROGRAM="/bin/bash -c '/etc/udev/rules.d/slot\_info.sh %s{bus}'", \

SYMLINK+="drxp-%s{serial}-%s{boardtype}-if%s{if\_nr}-bus%s{bus}-%c{2}"

$ cat /etc/udev/rules.d/slot\_info.sh

#!/bin/sh

/sbin/dmidecode -t slot | grep -B12 ${1} | grep Designation | cut -d':' -f2 | sed 's/^[ \t]\*//'

# dmidecode –t slot

Handle 0x0012, DMI type 9, 17 bytes

System Slot Information

**Designation: CPU2 SLOT8 PCI-E 3.0 X16**

Type: x16 PCI Express 3 x16

Current Usage: In Use

Length: Long

ID: 8

Characteristics:

(snip)

**Bus Address: 0000:d8:00.0**

In the case of above rule, system executes the following program when it detects “/dev/drxpd\*”.

/bin/bash –c '/etc/udev/rules.d/slot\_info.sh **%s{bus}**’

The slot\_info.sh program just cuts out the line of phsical slot position information (Designation) from the output of “dmidecode” command. And udev rule can refer the output of this program by using %c{N}. In this example, program uses bus number for identifying the board. In the case that “demidecode” command returns one-to-one correspondence between DRXP board bus and physical slot number, this example works correctly. For the detail of udev, please refer the manual of udev. You can see it by executing command “$man udev”.

## State transition table

### State table (reception mode)

Following table shows state transition table for reception mode. The table below does not consider overheat state. For the detail of this state, please refer to **“7.4 The overheating protection”**. And some functions and monitoring and control Items are divided into groups for readability. For the detail of these groups, please refer **“Table 7‑3 : function/ItemID group (reception mode)”**.



(\*1) (\*2) (\*3) (\*4) (\*5) For the detail, please refer **“Table 7‑3 : function/ItemID group (reception mode)”**.

|  |  |  |
| --- | --- | --- |
| **Group** | **Function/ItemID** | **Description** |
| DRXPD\_SET\_CONTROL group1 | RESET\_DFR\_CHKSUM  RESET\_DFR\_SYNC  SET\_DFR\_TEST\_DATA  INSERT\_SYNCPTN\_ERR  SET\_CHKSUM\_ERR  SET\_SQCNT\_ERR  SET\_SEU\_RSTERR  SET\_OVH\_THRESHOLD | - |
| DRXPD\_SET\_CONTROL group2 | RESET\_DFR | - |
| DRXPD\_SET\_CONTROL group3 | SET\_DFR\_FORCETX  SET\_PARITY\_RANGE | - |
| DRXPD\_SET\_CONTROL group4 | SET\_SEU\_STATUS  SET\_SEU\_INJERR | These functions depend on SEU module status. For the detail of SEU module, please refer **“7.3 SEU Mitigation”**. |
| DRXPD\_GET\_MONSTS | All monitoring ItemIDs. | - |
| ioctl command group1 | DRXPD\_GET\_RCVSTS  DRXPD\_CLR\_RCV  DRXPD\_GET\_I2C\_DATA  DRXPD\_SET\_I2C\_CMD  DRXPD\_SET\_SEUEVT  DRXPD\_CLR\_SEUEVT  DRXPD\_FLASH\_OVHLOG  DRXPD\_SEL\_EQ  DRXPD\_GET\_EQ  DRXPD\_GET\_OVRFLW | - |

Table 7‑3 : function/ItemID group (reception mode)

### State table (transmission mode)

Following table shows state transition table for transmission mode. The table below does not consider overheat state. For the detail of this state, please refer to **“7.4 The overheating protection”**. And some functions and monitoring and control Items are divided into groups for readability. For the detail of these groups, please refer **“Table 7‑4 : function/ItemID group (transmission mode)”**.



(\*1) (\*2) (\*3) (\*4) (\*5)For the detail, please refer **“Table 7‑4 : function/ItemID group (transmission mode)”**.

|  |  |  |
| --- | --- | --- |
| **Group** | **Function/ItemID** | **Description** |
| DRXPD\_SET\_CONTROL group1 | RESET\_DFR\_CHKSUM  RESET\_DFR\_SYNC  SET\_DFR\_TEST\_DATA  INSERT\_SYNCPTN\_ERR  SET\_CHKSUM\_ERR  SET\_SQCNT\_ERR  SET\_SEU\_RSTERR  SET\_PARITY\_RANGE  SET\_OVH\_THRESHOLD | - |
| DRXPD\_SET\_CONTROL group2 | RESET\_DFR | - |
| DRXPD\_SET\_CONTROL group3 | SET\_DFR\_FORCETX | - |
| DRXPD\_SET\_CONTROL group4 | SET\_SEU\_STATUS  SET\_SEU\_INJERR | These functions depend on SEU module status. For the detail of SEU module, please refer **“7.3 SEU Mitigation”**. |
| DRXPD\_GET\_MONSTS | All monitoring ItemIDs. | - |
| ioctl command group1 | DRXPD\_GET\_SNDSTS  DRXPD\_GET\_I2C\_DATA  DRXPD\_SET\_I2C\_CMD  DRXPD\_SET\_SEUEVT  DRXPD\_CLR\_SEUEVT  DRXPD\_FLASH\_OVHLOG  DRXPD\_SEL\_EQ  DRXPD\_GET\_EQ  DRXPD\_GET\_OVRFLW | - |

Table 7‑4 : function/ItemID group (transmission mode)