Control Interface

Ola S. Grøttvik

Version 1 - August 27, 2019

1 pRU Application Layer Protocol

This document describes the protocol needed to communicate with the pRU embedded system, and the pRU bus system. The pRU embedded system is functioning as a TCP server listening to any hosts trying to make a connection. Note that all tables show the protocol in network byte order(NBO).

2 General format

The following table shows the general format for all messages sent either to or from the pRU.

Offsets	0	1	2	N	N+1
	LEN		CMDTYP	PAYLOAD	SEQ_NUM

2.1 LEN

Specified the number of bytes in the payload. The total size is limited by TCP/IP max frame size.

2.2 CMDTYP

CMDTYPE	Description
0xAA	Firmware Module R/W Request
0xFF	ALPIDE chip R/W Request
0xBB	Special command request
0x03	pRU Reply

2.3 SEQ_NUM

Sequence number is an optional value. Replies will have the same sequence number as the request that initiated something on the pRU.

2.4 Payload

The payload length is varies with CMDTYPE and the number of request sent. E.g. one can transmit many read register request in one message. This feature permits a bandwidth reduction if many requests are to be sent in a row (for instance when configuring ALPIDE chips).

2.4.1 Payload Opcodes

The following table shows different opcodes that initiates each part of the payload.

CMDTYPE	Description
0xAA	READ_OPCODE
0xFF	WRITE_OPCODE
TBD	Not implemented, mask pixels
TBD	Not implemented, unmask pixels
TBD	Not implemented, select pixels
TBD	Not implemented, deselect pixels
0x1	Deprecated (only used for PTB), Spawn offload thread
0x2	Deprecated (only used for PTB), Delete offload thread
TBD	Not implemented, add chip to monitor
TBD	Not implemented, remove chip from monitor

2.5 Firmware Module Requests

These requests send simple read or write requests to the modules directly connected to the bus system. These modules include global_regs, trigger_manager, etc. All pRU tasks can be achieved by using these requests only, but require full knowledge of all parts of firmware[1].

2.5.1 Firmware Module Read Request

Format for a single read request:

Offsets	0	1	2	3	4
	READ_OPCODE	REGADDR			

Format for a double read request:

Offsets	0	1	2	3	4	5	6	7	8	9
	READ_OPCODE	REGADDR_0				READ_OPCODE	REGADDR_1			

Note that even more requests can be sent at one time. This is true for all requests.

2.5.2 Firmware Module Write Request

Format for a single write request:

Offsets	0	1	2	3	4	5	6	7	8
	WRITE_OPCODE	REGADDR				REGVAL			

Format for a double write request:

Offsets	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17
	WRITE_OPCODE	REGADDR_0				REGVAL_0				WRITE_OPCODE	REGADDR_1				REGVAL_1			

[1] E.g. do not use these for ALPIDE communication if not critical to do some serious debugging.

2.6 ALPIDE Requests[2]

These requests enables the user to communicate with ALPIDE sensor chips. Some special opcodes (opted from the ALPIDE control interface), is used to determine what can of operation is performed (see figure below). Only 1 byte opcodes are supported.

OPCODE	Hex value	Purpose
GRST	0x00D2	Chip global reset
PRST	0x00E4	Pixel matrix **RESET**
PULSE	0x0078	Pixel matrix **PULSE**
BCRST	0x0036	Bunch Counter reset
RORST	0x0063	Readout (RRU/TRU/DMU) reset
DEBUG	0x00AA	Store snapshot into debug registers
TRIGGER	0x00B1,0055,00C9,002D	Trigger command
WROP	0x009C	Start Unicast or Multicast Write
RDOP	0x004E	Start Unicast Read
CMU CLEAR ERR	0xFF00	Clear CMU error flags
FIFOTEST	0xFF01	Starts regions memory test
LOADOBDEFCFG	0xFF02	Loads default configuration for the OB Module. Initial Token is set in the MASTER chip; Previous Chip ID is set to 0x6 in the MASTER, and to Chip ID minus 1 in the SLAVEs
XOFF	0xFF10	Stops sending data off-chip
XON	0xFF11	Resume data sending
ADCMEASURE	0xFF20	Start ADC measure

2.6.1 Request for updates

This whole scheme would need an update to allow for the following:

1. Transmit even more requests per message (this is now limited by 3-bit NSNGL-field). This could e.g. be fixed by having 1 byte shared by CHIPID and STAVEID, and a whole byte for number of requests. This would allow for 255 requests in one message.
2. Possibility to mask/unmask and select/deselect pixels
3. Possibillity to transmit all opcodes (even 2-byte opcodes like ADCMEASURE)

2.6.2 ALPIDE Read Request

Format for a single read request:

Offsets	0	1	2	3	4
	RDOP	CHIPID	STAVEID(5b) NSNGL(3b)	REGADDR	

Format for a double read request:

Offsets	0	1	2	3	4	5	6
	RDOP	CHIPID	STAVEID(5b) NSNGL(3b)	REGADDR_0		REGADDR_1	

Note that even more requests can be sent at one time. This is true for all requests.

NSNGL This is the number of requests in the message.

[2]It is HIGHLY advised not to use these requests for any kind of synchronization! Instead use trigger_manager features for GRST, PRST, BCRST, etc.

2.6.3 ALPIDE Write Request

Format for a single write request:

Offsets	0	1	2	3	4	5	6
	WROP	CHIPID	STAVE(5b) NSNGL(3b)	REGADDR		REGVAL	

Format for a double write request:

Offsets	0	1	2	3	4	5	6	7	8	9	0
	WROP	CHIPID	STAVE(5b) NSNGL(3b)	REGADDR_0		REGVAL_0		REGADDR_1		REGVAL_1	

2.6.4 ALPIDE Broadcast Opcode Request

Transmits the indicated broadcast opcode to all staves. However, transmits it one after another. Do NOT use for synchronization.

Offsets	0
	OPCODE

2.7 Replies

As commands are received, they are executed in order by the control interface. The reply payload is formed when iterating over the received packet. The response depends on action taken.

Action Taken	Code	Response
Module Register Read	0x06	Code followed by register value
ALPIDE Register Read	0x07	Code followed by register value
Module Register Write	0x08	Code
ALPIDE Register Write	0x09	Code
Special Command	0x0A	Code
ALPIDE Broadcast Opcode	0x0B	Code
Error CMD (Invalid CMDTYP received)	0x01	Error Code
Error LEN	0x02	Error Code
Error CRC (Deprecated)	0x03	Error Code
Error delimeter (Deprecated)	0x04	Error Code
Error write/read (could not be performed)	0x0C	Error Code
Error OPCODE (invalid OPCODE)	0x0E	Error Code
Error special (Invalid special command)	0x0F	Error Code
Pixel mask/select	TBD	Not implemented

2.8 Examples

2.8.1 Firmware Module R/W

Example of a single read. This command shows the reading of module address 0x2000_0004. Which, by chance, is the global_regs register hash_code. Note that the length field value is 5, which is the number of payload bytes.

Offsets	0	1	2	3	4	5	6	7	8
	LEN		CMDTYP	READ_OPCODE	REGADDR				SEQ_NUM
	0x00	0x05	0xAA	0xAA	0x20	0x00	0x00	0x04	0x00

Reply from pRU based on the previous example of read. Note that the value of the register read was 0xE321_8a56.

Offsets	0	1	2	3	4	5	6	7	8
	LEN		CMDTYP	REPLY_OPCODE	REGVAL				SEQ_NUM
	0x00	0x05	0x03	0x06	0xe3	0x21	0x8a	0x56	0x00

Example of a double read. Note that the length field value is 10, which is the number of payload bytes. SEQ_NUM was randomly chosen.

Offsets	0	1	2	3	4	5	6	7	8	9	10	11	12	13
	LEN		CMDTYP	READ_OPCODE	REGADDR_0				READ_OPCODE	REG_ADDR_1				SEQ_NUM
	0x00	0x0A	0xAA	0xAA	0x20	0x00	0x00	0x00	0xAA	0x20	0x00	0x00	0x04	0x12

Reply from pRU based on the previous example of read.

Offsets	0	1	2	3	4	5	6	7	8	9	10	11	12	13
	LEN		CMDTYP	REPLY_OPCODE	REGVAL_0				REPLY_OPCODE	REGVAL_1				SEQ_NUM
	0x00	0x0A	0x03	0x06	0x19	0x08	0x20	0x21	0x06	0xe3	0x21	0x8a	0x56	0x12

Example of a double write.

Offsets	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21
	LEN		CMDTYP	WRITE_OPCODE	REGADDR_0				REGVAL_0				READ_OPCODE	REG_ADDR_1				REGVAL_1				SEQ_NUM
	0x00	0x0A	0xAA	0xFF	0x20	0x00	0x00	0x08	0x00	0x00	0x00	0x00	0xAA	0x20	0x00	0x00	0x18	0xFF	0xFF	0xFF	0xFF	0x24

Reply from pRU based on the previous example of write.

Offsets	0	1	2	3	4	5
	LEN		CMDTYP	REPLY_OPCODE	REPLY_OPCODE	SEQ_NUM
	0x00	0x0A	0x03	0x08	0x08	0x24

2.8.2 ALPIDE R/W

Example of a single read of the BUSY min width register. 0x09 indicates stave ID = 1, and 1 single request.

Offsets	0	1	2	3	4	5	6	7	8
	LEN		CMDTYP	READ_OPCODE	CHIPID	STAVEID(5b) NSNGL(3b)	REGADDR		SEQ_NUM
	0x00	0x05	0xFF	0x4E	0x01	0x09	0x00	0x1B	0xCE

Reply from the pRU based on the previous example of read.

Offsets	0	1	2	3	4	5	6
	LEN		CMDTYP	REPLY_OPCODE	REGVAL		SEQ_NUM
	0x00	0x03	0x03	0x07	0x00	0x08	0xCE