ACDC DAQ Programers Manual

Jonathan Eisch, Miles Lucas, Eric Oberla and Others

Contents

1 Introduction

This document is intended to document the commands sent to the ACC and ACDC cards.

It includes source listings of the source and destinations for each command as they were when this documentation was written. New implementations may have been written since, but that is outside the scope of this document.

The command protocol for each 32-bit instruction set is shown in Figure 1.

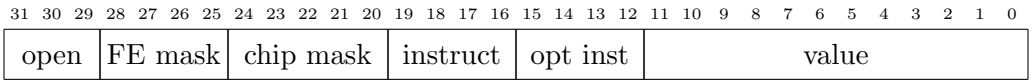

Figure 1: Command protocol

This is parsed on the front-end board like this

```
INSTRUCT_PSEC_MASK      <= xINSTRUCT_WORD(24 downto 20);
INSTRUCTION             <= xINSTRUCT_WORD(19 downto 16);
INSTRUCTION_OPT         <= xINSTRUCT_WORD(15 downto 12);
INSTRUCT_VALUE          <= xINSTRUCT_WORD(11 downto 0);
```

Table 1 lists all the instruction flags passed by software.

The instruction for setting self-trigger instructions has many optional instructions, listed in Table 2.

Table 1: List of Instructions

Bits	Description
0x0	Do nothing
0x1	Set delay-locked loop VDD control voltage
0x2	Calibration pulse switch enable
0x3	Set pedestal
0x4	Reset DLL
0x5	Reset internal trigger
0x6	Set self trigger mask
0x7	Set self trigger instructions
0x8	Set trigger threshold
0x9	Adjust ring oscillation frequency
0xA	Enable/disable on-board LEDs
0xB	(DC) Central card FIFO toggle
0xB	(CC) Central card done
0xC	(CC) Central card read mode
0xD	(CC) Align/setup SERDES
0xE	(CC) USB trigger
0xF	(CC) Sync USB

Table 2: Optional instructs for 0x7 - set self-trigger instructions

Bits	Description
0x0	Enable
0x1	Wait for system trigger
0x2	Measure rate only
0x3	Trigger sign

2 Instructions

2.1 0x1 - Set DLL VDD

This sets the dll vdd

2.1.1 Bit Fields

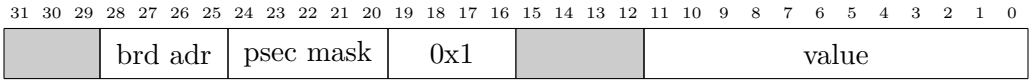

Figure 2: Command 0x1 bit fields

Bits	Name	Description
28-25	board address	The address of the board
24-20	psec mask	The psec chips to apply this to
19-16	**0x1**	Command marker
11-0	value	VDD Value

2.1.2 Source

acdc-daq:src/DAQinstruction.cpp

```
void SuMo::set_dll_vdd(unsigned int VALUE,
        unsigned int boardAdr,
        int device,
        unsigned int psec_mask)
{
        createUSBHandles();
        const unsigned int hi_cmd = 0x00010000;
        unsigned int send_word = hi_cmd | VALUE | boardAdr <<
            ↪ boardAdrOffset
        | psec_mask << psecAdrOffset;

        if(device == 0)                    usb.sendData(send_word);
        if(device == 1 && mode==USB2x) usb2.sendData(send_word);

        closeUSBHandles();
}
```

2.1.3 Destination

project:path/to/filename.vhd

2.2 0x2 -Toggle Cal

2.2.1 Bit Fields

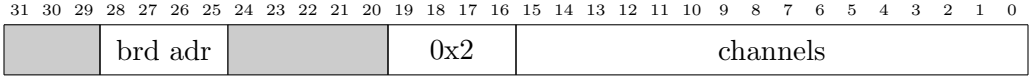

Figure 3: 0x2 - Cal enabled

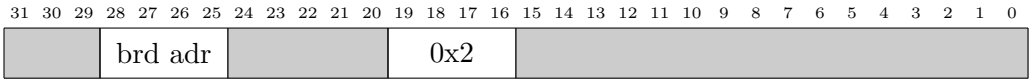

Figure 4: 0x2 - Cal disabled

Bits	Name	Description
28-25	board address	The address of the board (default = 15)
19-16	**0x2**	Command marker
15-0	channels	Channels (default = 0x7FFF)
		Not used if cal is disabled

2.2.2 Source

acdc-daq:src/DAQinstruction.cpp

```
void SuMo::toggle_CAL(bool EN,  int device)
{
        createUSBHandles();

        unsigned int send_word = 0x00020000;
        unsigned int channels  = 0x7FFF;
        unsigned int boardAdr  = 15;

        if(EN){
                send_word = send_word | boardAdr <<
                    ↪ boardAdrOffset | channels;
                if(device == 0)                    usb.sendData(
                    ↪ send_word);
                if(device == 1 && mode==USB2x) usb2.sendData(
                    ↪ send_word);
```

```
        } else{
                send_word = send_word | boardAdr <<
                    ↪ boardAdrOffset;
                if(device == 0)                    usb.sendData(
                    ↪ send_word);
                if(device == 1 && mode==USB2x) usb2.sendData(
                    ↪ send_word);
        }
closeUSBHandles();
}
```

2.2.3 Destination

project:path/to/filename.vhd

2.3 0x3 -Set Pedestal Value

2.3.1 Bit Fields

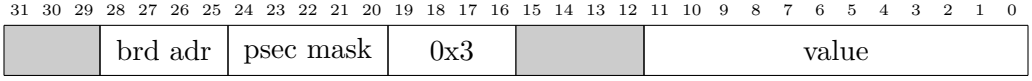

Figure 5: Command 0x3 bit fields

Bits	Name	Description
28-25	board address	The address of the board (default = 15)
24-20	psec mask	Mask for chip addresses
19-16	**0x2**	Command marker
12-0	value	Pedestal value (default = 0x800)

2.3.2 Source

acdc-daq:src/DAQinstruction.cpp

```
void SuMo::set_pedestal_value(  unsigned int PED_VALUE,
unsigned int boardAdr,
int device,
unsigned int psec_mask)
{
        createUSBHandles();
        const unsigned int hi_cmd = 0x00030000;
        unsigned int send_word = hi_cmd | PED_VALUE
        | boardAdr << boardAdrOffset
        | psec_mask << psecAdrOffset;

        if(device == 0)                  usb.sendData(send_word);
        if(device == 1 && mode==USB2x)   usb2.sendData(send_word);

        closeUSBHandles();
}
```

2.3.3 Destination

project:path/to/filename.vhd

2.4 0x4 - Reset DLL

This command resets the DLL

2.4.1 Bit Fields

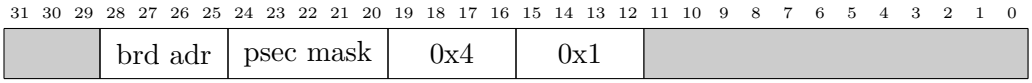

Figure 6: 0x4 Instruct Bit Fields

Bits	Name	Description
28-25	board address	The address of the board.
24-20	psec mask	Which psec chips to apply this to
19-16	**0x4**	Command marker
15-12	**0x1**	Optional command marker

2.4.2 Source

acdc-daq:src/DAQinstruction.cpp

```
void SuMo::reset_dll(bool sync)
{
        createUSBHandles();
        const unsigned int hi_cmd = 0x00041000;
        const unsigned int mask = 31; //chip mask
        unsigned int send_word = hi_cmd |  mask << 20 | 15 <<
            ↪ boardAdrOffset;

        if(sync) prep_sync();
        usb.sendData(send_word); //dll reset pulse
        if(mode == USB2x) usb2.sendData(send_word);
        if(sync) make_sync();

        closeUSBHandles();
}
```

2.4.3 Destination

2.5 0x4 - Reset Self Trigger

This command resets the self trigger

2.5.1 Bit Fields

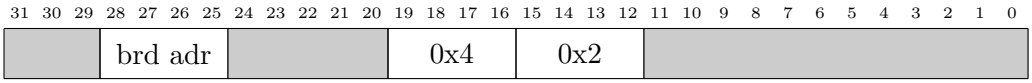

Figure 7: 0x4 Instruct Bit Fields

Bits	Name	Description
28-25	board address	The address of the board.
19-16	**0x4**	Command marker
15-12	**0x2**	Optional command marker

2.5.2 Source

acdc-daq:src/DAQinstruction.cpp

```
void SuMo::reset_self_trigger(unsigned int boardAdr, int device)
{
        createUSBHandles();
        unsigned int send_word = 0x00042000 | boardAdr <<
            ↪ boardAdrOffset;

        if(device == 0) usb.sendData(send_word);
        if(device == 1 && mode == USB2x) usb2.sendData(send_word)
            ↪ ;

        closeUSBHandles();
}
```

2.5.3 Destination

2.6 0x4 - Reset Time Stamp

This command resets the time stamp

2.6.1 Bit Fields

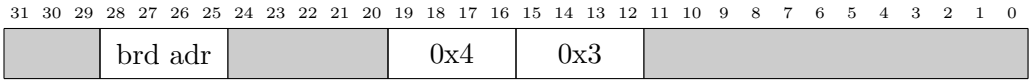

Figure 8: 0x4 Instruct Bit Fields

Bits	Name	Description
28-25	board address	The address of the board.
19-16	**0x4**	Command marker
15-12	**0x3**	Optional command marker

2.6.2 Source

acdc-daq:src/DAQinstruction.cpp

```
void SuMo::reset_time_stamp(bool sync)
{
        createUSBHandles();
        unsigned int send_word = 0x00043000 | 15 <<
            ↪ boardAdrOffset;

        if(sync) prep_sync();
        usb.sendData(send_word);
        if(mode == USB2x) usb2.sendData(send_word);
        if(sync) make_sync();

        closeUSBHandles();
}
```

2.6.3 Destination

2.7 0x4 - Reset ACDC

This command resets the acdc

2.7.1 Bit Fields

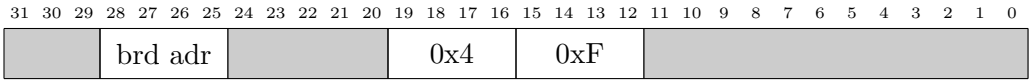

Figure 9: 0x4 Instruct Bit Fields

Bits	Name	Description
28-25	board address	The address of the board.
19-16	**0x4**	Command marker
15-12	**0xF**	Optional command marker

2.7.2 Source

acdc-daq:src/DAQinstruction.cpp

```
void SuMo::reset_acdc()
{
        usb.createHandles();
        unsigned int send_word = 0x0004F000 | 15 <<
            ↪ boardAdrOffset;

        usb.sendData(send_word);
        if(mode == USB2x) usb2.sendData(send_word);

        closeUSBHandles();
}
```

2.7.3 Destination

2.8 0x4 - Hard Reset

This command resets the acdc

2.8.1 Bit Fields

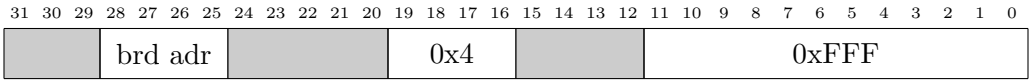

Figure 10: 0x4 Instruct Bit Fields

Bits	Name	Description
28-25	board address	The address of the board.
19-16	**0x4**	Command marker
11-0	**0xFFF**	Hard reset value

2.8.2 Source

acdc-daq:src/DAQinstruction.cpp

```
void SuMo::hard_reset(bool DEVICE)
{
        usb.createHandles();
        unsigned int send_word = 0x00040FFF | 15 <<
            ↪ boardAdrOffset;
        usb.sendData(send_word);
        if(mode == USB2x && DEVICE==true) usb2.sendData(send_word
            ↪ );

        closeUSBHandles();
}
```

2.8.3 Destination

2.9 0x4 - USB Force Wakeup

This command resets the acdc

2.9.1 Bit Fields

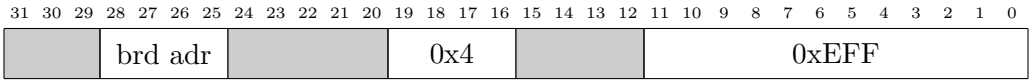

Figure 11: 0x4 Instruct Bit Fields

Bits	Name	Description
28-25	board address	The address of the board.
19-16	**0x4**	Command marker
11-0	**0xEFF**	USB wakeup value

2.9.2 Source

acdc-daq:src/DAQinstruction.cpp

```
void SuMo::usb_force_wakeup()
{
        usb.createHandles();
        unsigned int send_word = 0x00040EFF;
        usb.sendData(send_word);
        if(mode == USB2x) usb2.sendData(send_word);

        closeUSBHandles();
}
```

2.9.3 Destination

2.10 0x6 - Set Self-trigger Mask

This command sends certain trigger mask

2.10.1 Bit Fields

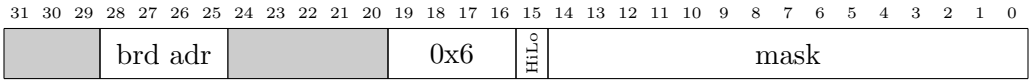

Figure 12: 0x6

Bits	Name	Description
28-25	board address	The address of the board.
19-16	**0x6**	Command marker
15	HiLo	Which half of the mask is being sent[1]
14-0	mask	AC/DC channel mask

2.10.2 Source

acdc-daq:src/DAQinstruction.cpp

```
void SuMo::set_self_trigger_mask(int mask, bool HiLo, unsigned
    ↪ int boardAdr, int device)
{
        unsigned int hi_cmd;
        if(HiLo) hi_cmd = 0x00068000;
        else     hi_cmd = 0x00060000;

        unsigned int send_word = hi_cmd | mask | boardAdr <<
            ↪ boardAdrOffset;

        createUSBHandles();
        if(device == 0)                    usb.sendData((unsigned
            ↪ int)send_word);
        if(device == 1 && mode == USB2x) usb2.sendData((unsigned
            ↪ int)send_word);
```

[1] When set to 0, the values for the 15 least significant bits are sent, which correspond to channels 1-15. When set to 1, the values for the 15 most significant bits are sent, which correspond to channels 16-30.

```
        closeUSBHandles();
}
```

2.10.3 Destination

2.11 0x7 - Set Self-trigger Lo

This command sends certain trigger commands

2.11.1 Bit Fields

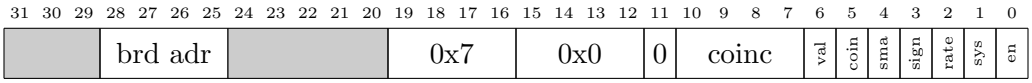

Figure 13: Command 0x7 bit fields

Bits	Name	Description
28-25	board address	The address of the board. Default is 0xF
19-16	**0x7**	Command marker
15-12	**0x0**	Optional command marker
10-7	coincidence window	The window of unknown units for the coincidence (<15)
6	use trig valid	Use trig valid as a reset on AC/DC
5	use coincidence	Use channel coincidence
4	use board sma trig	Use SMA input on AC/DC board for trigger
3	trig sign	1 for rising edge, 0 for falling edge
2	rate only	
1	sys trig option	
0	trig enable	Enables self-trigger

2.11.2 Source

acdc-daq:src/DAQinstruction.cpp

```
void SuMo::set_self_trigger_lo(      bool ENABLE_TRIG,
                                     bool SYS_TRIG_OPTION,
                                     bool RATE_ONLY,
                                     bool TRIG_SIGN,
                                     bool USE_BOARD_SMA_TRIG,
                                     bool USE_COINCIDENCE,
                                     bool USE_TRIG_VALID_AS_RESET,
                                     unsigned int coinc_window,
                                     unsigned int boardAdr,
                                     int device)
{
```

```
    const unsigned int hi_cmd = 0x00070000;
    unsigned int send_word = hi_cmd | 0 << 11
      | USE_TRIG_VALID_AS_RESET << 6
      | USE_COINCIDENCE << 5
      | USE_BOARD_SMA_TRIG << 4
      | TRIG_SIGN << 3 | RATE_ONLY << 2
      | SYS_TRIG_OPTION << 1 | ENABLE_TRIG
      | coinc_window << 7
      | boardAdr << boardAdrOffset;
    //printf("%i\n", send_word);

    createUSBHandles();

    if(device == 0)                          usb.sendData((unsigned int)
      ↪ send_word);
    if(device == 1 && mode == USB2x) usb2.sendData((unsigned int)
      ↪ send_word);

    closeUSBHandles();
}
```

2.11.3 Destination

2.12 0x7 - Set Self-trigger hi

This command sends certain trigger commands

2.12.1 Bit Fields

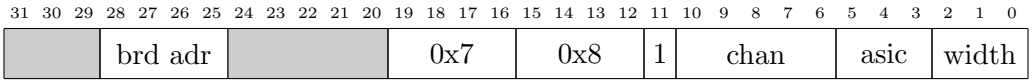

Figure 14: Command 0x7 bit fields

Bits	Name	Description
28-25	board address	The address of the board.
19-16	**0x7**	Command marker
15-12	**0x8**	Optional command marker
10-6	channel coincidence min	Number of coincident channels to enable trigger (<30)
5-3	asic coincidence min	Number of coincident asic chips to enable trigger (<5)
2-0	coincidence pulse width	Width of coincidence pulse in unknown units (<7)

2.12.2 Source

acdc-daq:src/DAQinstruction.cpp

```
void SuMo::set_self_trigger_hi(unsigned int coinc_pulse_width,
        unsigned int asic_coincidence_min,
        unsigned int channel_coincidence_min,
        unsigned int boardAdr,
        int device)
        {
        const unsigned int hi_cmd = 0x00078000;
        unsigned int send_word = hi_cmd | 1 << 11
        | channel_coincidence_min << 6
        | asic_coincidence_min << 3
        | coinc_pulse_width
        | boardAdr << boardAdrOffset;
        //printf("%x\n", send_word);

        createUSBHandles();
```

```
        if(device == 0) usb.sendData((unsigned int)send_word);
        if(device == 1 && mode == USB2x) usb2.sendData((unsigned
            ↪ int)send_word);

        closeUSBHandles();
}
```

2.12.3 Destination

2.13 0x8 - Set Self-trigger Threshold

This command sets trigger threshold

For some reason this seems to create a word with the psec_mask value assigned twice. It looks like he bit shifts it 20, but does the same thing again and bit shifts it by a defined value. Sorry if that doesn't make sense just look at the 4th line of the method.

2.13.1 Bit Fields

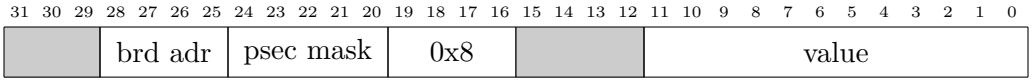

Figure 15: Command 0x8 bit fields

Bits	Name	Description
28-25	board address	The address of the board.
24-20	psec mask	Which psec chips to apply this to
19-16	**0x8**	Command marker
11-0	value	Trigger threshold value

2.13.2 Source

acdc-daq:src/DAQinstruction.cpp

```
void SuMo::set_trig_threshold(unsigned int TRIG_VALUE,
        unsigned int boardAdr,
        int device,
        unsigned int psec_mask)
{
        createUSBHandles();
        const unsigned int hi_cmd = 0x00080000;
        const unsigned int mask = 31;  //chip mask
        unsigned int send_word = hi_cmd | TRIG_VALUE | psec_mask
            ↪ << 20 | boardAdr << boardAdrOffset
```

```
                | psec_mask << psecAdrOffset;

        if(device == 0)                    usb.sendData(send_word);
        if(device == 1 && mode==USB2x)  usb2.sendData(send_word);

        closeUSBHandles();
}
```

2.13.3 Destination

2.14 0x9 - Set RO Target Count

This command sets the RO Target count. What is the RO target count?

2.14.1 Bit Fields

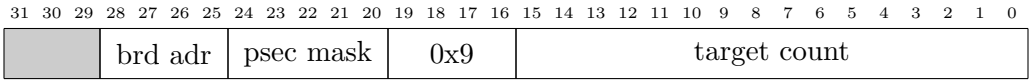

Figure 16: 0x9 Instruct Bit Fields

Bits	Name	Description
28-25	board address	The address of the board.
24-20	psec mask	Which psec chips to apply this to
19-16	**0x9**	Command marker
15-0	target count	The value of the ro target count

2.14.2 Source

acdc-daq:src/DAQinstruction.cpp

```
void SuMo::set_ro_target_count(unsigned int TARGET_RO_COUNT,
        unsigned int boardAdr,
        int device,
        unsigned int psec_mask)
{
        createUSBHandles();
        const unsigned int hi_cmd = 0x00090000;
        unsigned int send_word = hi_cmd | TARGET_RO_COUNT |
            ↪ boardAdr << boardAdrOffset
                | psec_mask << psecAdrOffset;

        usb.sendData(send_word);
        if(mode==USB2x) usb2.sendData(send_word);

        closeUSBHandles();
}
```

2.14.3 Destination

2.15 0xA - Toggle LED

This command toggles the LED on all connected boards.

2.15.1 Bit Fields

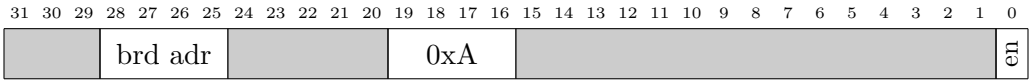

Figure 17: Command 0xA bit fields

Bits	Name	Description
28-25	board address	The address of the board. Default is 0xF
19-16	**0xA**	Command marker
0	enable	Enable the leds

2.15.2 Source

acdc-daq:src/DAQinstruction.cpp

```
void SuMo::toggle_LED(bool EN)
{
      unsigned int boardAdr_all = 15;

      createUSBHandles();
      unsigned int send_word = 0x000A0000;
      send_word = send_word | boardAdr_all << boardAdrOffset;

      if(EN != false){
        usb.sendData(send_word | 0x1);
        if(mode == USB2x) usb2.sendData(send_word | 0x1);
      }
      else{
        usb.sendData(send_word);
        if(mode == USB2x) usb2.sendData(send_word);
      }
      closeUSBHandles();
}
```

2.15.3 Destination

2.16 0xA - Read ACDC Ram

THIS COMMAND DOES NOT FOLLOW ERIC'S BASIC PROTOCOL OUTLINE (section 1, Figure 1)

2.16.1 Bit Fields

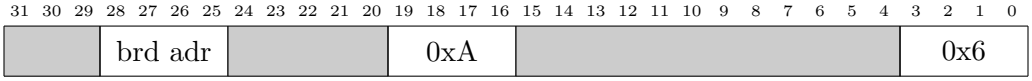

Figure 18: Command 0xA bit fields

Bits	Name	Description
28-25	board address	The address of the board (default = 15)
19-16	**0xA**	Command marker
3-0	**0x6**	Optional command marker

2.16.2 Source

acdc-daq:src/DAQinstruction.cpp

```
void SuMo::readACDC_RAM(int device, unsigned int boardAdr)
{
        unsigned int boardAdr_override  = 15;

        createUSBHandles();
        unsigned int send_word = 0x000A0006;
        send_word = send_word | boardAdr << boardAdrOffset;

        if(device == 0)                    usb.sendData(send_word);
        if(device == 1 && mode==USB2x)     usb2.sendData(send_word);

        closeUSBHandles();
}
```

2.16.3 Destination

project:path/to/filename.vhd

2.17 0xB - Manage CC FIFO

This command manages the CC FIFO

2.17.1 Bit Fields

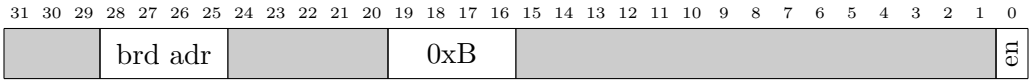

Figure 19: 0xB Instruct Bit Fields

Bits	Name	Description
28-25	board address	The address of the board.
19-16	**0xB**	Command marker
0	enable	Enable CC FIFO

2.17.2 Source

acdc-daq:src/DAQinstruction.cpp

```
void SuMo::manage_cc_fifo(bool VALUE)
{
        usb.createHandles();
        const unsigned int hi_cmd = 0x000B0000;
        unsigned int send_word = hi_cmd | VALUE | 15 <<
            ↪ boardAdrOffset;
        usb.sendData(send_word);
        usb.freeHandles();
}
```

2.17.3 Destination

2.18 0xB - Prep Sync

This command preps sync

2.18.1 Bit Fields

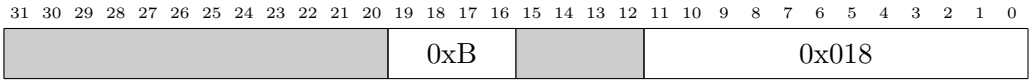

Figure 20: 0xB Instruct Bit Fields

Bits	Name	Description
19-16	**0xB**	Command marker
11-0	**0x018**	Prep sync value

2.18.2 Source

acdc-daq:src/DAQinstruction.cpp

```
void SuMo::prep_sync()
{
        usb.createHandles();
        const unsigned int hi_cmd = 0x000B0018;
        unsigned int send_word = hi_cmd;
        usb.sendData(send_word);
        usb.freeHandles();
}
```

2.18.3 Destination

2.19 0xB - Make Sync

This command preps sync

2.19.1 Bit Fields

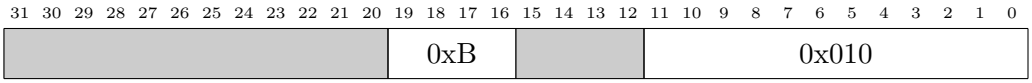

Figure 21: 0xB Instruct Bit Fields

Bits	Name	Description
19-16	**0xB**	Command marker
11-0	**0x010**	Make sync value

2.19.2 Source

acdc-daq:src/DAQinstruction.cpp

```
void SuMo::make_sync()
{
        usb.createHandles();
        const unsigned int hi_cmd = 0x000B0010;
        unsigned int send_word = hi_cmd;
        usb.sendData(send_word);
        usb.freeHandles();
}
```

2.19.3 Destination

2.20 0xB - System Card Trig Valid

This command validates system card triggers.

2.20.1 Bit Fields

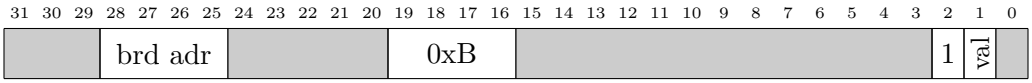

Figure 22: 0xB Instruct Bit Fields

Bits	Name	Description
28-25	board address	The address of the board.
19-16	**0xB**	Command marker
2	**0x1**	Optional command marker
1	valid	Validate system trigger

2.20.2 Source

acdc-daq:src/DAQinstruction.cpp

```
void SuMo::system_card_trig_valid(bool valid)
{
        usb.createHandles();
        const unsigned int hi_cmd = 0x000B0004;
        unsigned int send_word = hi_cmd | valid << 1 | 15 <<
            ↪ boardAdrOffset;;
        usb.sendData(send_word);
        usb.freeHandles();
}
```

2.20.3 Destination

2.21 0xC - Set USB Read Mode

This command sets the USB read mode. In the source code a few different modes are called frequently: 0, 5, and 16. 0 Corresponds to passing the USB instruction straight to the ACDC. 5 corresponds to reading out the CC buffer. 16 corresponds to setting the trig delay to 1.

2.21.1 Bit Fields

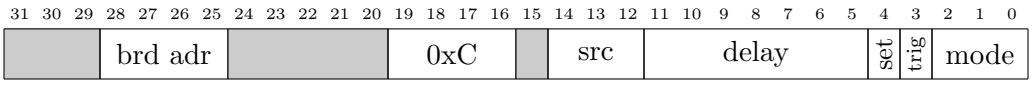

Figure 23: 0x6

Bits	Name	Description
28-25	board address	The address of the board.
19-16	**0xC**	Command marker
14-12	trig source	If 4 is enabled, defines the trig source
11-5	trig delay	
4	set trig source	If enabled, sets the trig source from bits 14-12
3	trig mode	
2-0	read mode	The value of the read mode. Shown further in subsubsection 2.21.2

2.21.2 Read Modes

Bits	Mode	Description
000	ACDC instruction	
111	ACDC trig validd CC only	
101	Read CC buffer	The value of the read mode
110		

2.21.3 Source

acdc-daq:src/DAQinstruction.cpp

```
void SuMo::set_usb_read_mode(unsigned int READ_MODE)
{
        usb.createHandles();
        const unsigned int hi_cmd = 0x000C0000;
        unsigned int send_word = hi_cmd | READ_MODE | 15 <<
           ↪ boardAdrOffset;
        usb.sendData(send_word);
        usb.freeHandles();
}
```

2.21.4 Destination

ACC-2xSPF Firmware:src/usbWrapperACC.vhd

```
when x"C" =>

CC_READ_MODE <= USB_INSTRUCTION(2 downto 0);
if USB_INSTRUCTION(4) = '1' then
        TRIG_MODE <= USB_INSTRUCTION(3);
        TRIG_DELAY  (6 downto 0) <= USB_INSTRUCTION(11 downto
           ↪ 5);
        SET_TRIG_SOURCE (2 downto 0) <= USB_INSTRUCTION(14
           ↪ downto 12);
end if;

if delay > 10 then
        delay := 0;
        state <= st1_WAIT;
--this is a hack:
-- basically, only want to send along to AC/DC if certain
   ↪ conditions apply
-- also want to only read CC info buffer if read mode = 0b101
else
        delay := delay + 1;
        if delay > 1 then
                case CC_READ_MODE is
                when "101" =>
                        read_cc_buffer <= '1';
                        CC_INSTRUCTION <= (others=>'0');
```

```
                        CC_INSTRUCT_RDY<= '0';
                when "110" =>
                        read_cc_buffer <= '0';
                        CC_INSTRUCTION <= (others=>'0');
                        CC_INSTRUCT_RDY<= '0';
                ---only send-along data to AC/DC cards when
                   ↪ 111 or 000
                when "111" =>
                        --cc_only_instruct_rdy <= '1';
                        trig_valid_CC_only <= '1';
                        read_cc_buffer <= '0';
                        CC_INSTRUCTION <= (others=>'0');
                        CC_INSTRUCT_RDY<= '0';
                when "000" =>
                        trig_valid_CC_only <= '0';
                        read_cc_buffer <= '0';
                        CC_INSTRUCTION <= USB_INSTRUCTION;
                        CC_INSTRUCT_RDY<= '1';
                ------
                when others =>
                        read_cc_buffer <= '0';
                        CC_INSTRUCTION <= (others=>'0');
                        CC_INSTRUCT_RDY<= '0';
                end case;
        end if;
end if;
```

2.22 0xD - Align LVDS

This command aligns the LVDS system between the central card and any acdc boards. The LVDS system is the RJ-45 connection.

2.22.1 Bit Fields

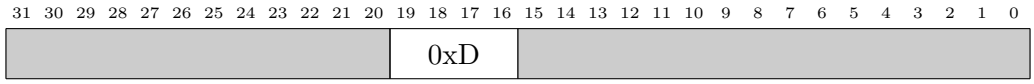

Figure 24: Command 0xD bit fields

Bits	Name	Description
19-16	**0xD**	Command marker

2.22.2 Source

acdc-daq:src/DAQinstruction.cpp

```
void SuMo::align_lvds()
{
	createUSBHandles();
	usb.sendData((unsigned int)0x000D0000);   //toggle align
	    ↪ process
	if(mode == USB2x) usb2.sendData((unsigned int)0x000D0000);
	closeUSBHandles();
}
```

2.22.3 Destination

2.23 0xE - Software Trigger

This command sends a software trigger (maybe).

2.23.1 Bit Fields

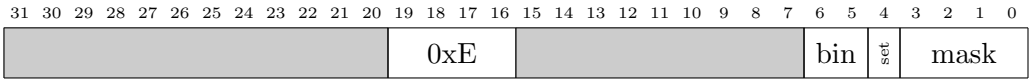

Figure 25: Command 0x7 bit fields

Bits	Name	Description
19-16	**0xE**	Command marker
6-5	bin	bin
4	set bin	Enable bin
3-0	mask	Soft trig mask

2.23.2 Source

acdc-daq:src/DAQinstruction.cpp

```
void SuMo::software_trigger(unsigned int SOFT_TRIG_MASK, bool
    ↪ set_bin, unsigned int bin)
{
        usb.createHandles();
        const unsigned int hi_cmd = 0x000E0000;
        unsigned int send_word = hi_cmd | SOFT_TRIG_MASK |
            ↪ set_bin << 4 | bin << 5;
        usb.sendData(send_word);
        usb.freeHandles();
}
```

2.23.3 Destination

ACC-2xSPF:src/usbWrapperACC.vhd

```
when x"E" =>     --SOFT_TRIG
```

```
cc_only_instruct_rdy <= '1';
SYNC_TRIG <= '1';
SOFT_TRIG <= '1';
SOFT_TRIG_MASK(3 downto 0) <= USB_INSTRUCTION(3 downto 0);
--SOFT_TRIG_MASK <= (others=>'1');  --trigger every AC/DC
SOFT_TRIG_BIN   <= USB_INSTRUCTION(6 downto 4);
if delay > 8 then
        delay := 0;
        state <= st1_WAIT;
else
        delay := delay + 1;
end if;
```

2.24 0xF - Sync Usb

This command does something regarding to syncing the usb

2.24.1 Bit Fields

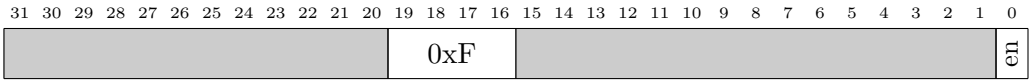

Figure 26: Command 0xF bit fields

Bits	Name	Description
19-16	**0xF**	Command marker
0	enable	Enable USB sync

2.24.2 Source

acdc-daq:src/DAQinstruction.cpp

```
void SuMo::sync_usb(bool SYNC) {
	createUSBHandles();
	if(SYNC != false) { //enable USB_SYNC
		usb.sendData((unsigned int)0x000F0001);
		if(mode == USB2x) usb2.sendData((unsigned int)0x000F0001
		    ↪ );
	} else { //disable USB_SYNC
		usb.sendData((unsigned int)0x000F0000);
		if(mode == USB2x) usb2.sendData((unsigned int)0x000F0000
		    ↪ );

	}
	closeUSBHandles();
}
```

2.24.3 Destination