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.

2 Commands

2.1 Template

2.1.1 Bit Fields

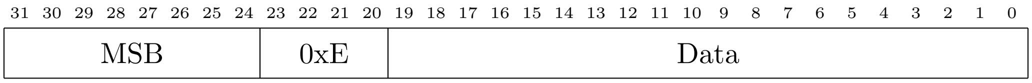

Figure 1: Command 0xE bit fields

Bit Range	Name	Description
31-24	MSB	This is really the most significant byte
23-20	**0xE**	Command marker
19-0	Data	All of the data.

2.1.2 Source

acdc-daq:src/DAQinstruction.cpp

```
void SuMo::software_trigger_slaveDevice(unsigned int
   ↪ SOFT_TRIG_MASK, bool set_bin , unsigned int bin)
{
    usb2.createHandles();
    const unsigned hi_cmd = 0x000E0000;
    unsigned int send_word = hi_cmd | SOFT_TRIG_MASK | set_bin <<
       ↪  4 | bin << 5;
    usb2.sendData(send_word);
    usb2.freeHandles();
}
```

2.1.3 Destination

ACDC-Firmware:SRC/dff_async_rst.vhd

```
library IEEE;
use IEEE.STD_LOGIC_1164.all;
use IEEE.numeric_std.all;
```

```
entity dff_async_rst is
  port( d, clk, reset: in std_logic;
      q: out std_logic);
end dff_async_rst;
architecture behav of dff_async_rst is
  begin
  process(clk,reset)
  begin
  if (reset=’1’) then
    q <= ’0’;
  elsif rising_edge(clk) then
    q <= d;
  end if;
  end process;
end behav;
```

2.2 Sync Usb

This command does something regarding to syncing the usb

2.2.1 Bit Fields

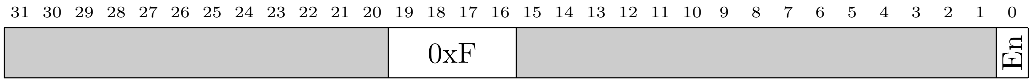

Figure 2: Command 0xF bit fields

Bit Range	Name	Description
19-16	**0xF**	Command marker
0	En	Boolean to enable or disable usb sync

2.2.2 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();
}
```

2.2.3 Destination

2.3 Align LVDS

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

2.3.1 Bit Fields

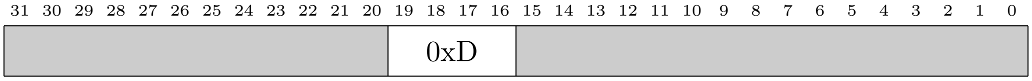

Figure 3: Command 0xD bit fields

Bit Range	Name	Description
19-16	**0xD**	Command marker

2.3.2 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();
}
```

2.3.3 Destination

2.4 Toggle LED

This command toggles the LED on all connected boards.

2.4.1 Bit Fields

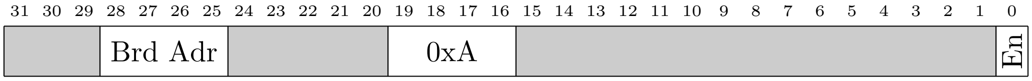

Figure 4: Command 0xA bit fields

Bit Range	Name	Description
28-25	Board Address	The address of the board. Default is 0xF
19-16	**0xA**	Command marker
0	Enable	Boolean to enable or disable the leds

2.4.2 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();
}
```

2.4.3 Destination