DASH 8 How to use the UDP interface.

Have a look at the majestic forum and search for UDP . a post from ana668 Rienhard should pop up . I got the starter information there. If you scroll down there is a video of my overhead panel and the DC control panel working via UDP over ethernet

First in the sim pc open ports 49250 outbound and 49260 inbound in the fire wall.

Open firewall ports in Windows 10

1. Navigate to Control Panel, System and Security and **Windows** Firewall.
2. Select Advanced settings and highlight Inbound Rules in the left pane.
3. Right click Inbound Rules and select New Rule.
4. Add the **port** you need to **open** and click Next.
5. Add the protocol (TCP or UDP) and the **port** number into the next **window** and click Next.

Next I downloaded packetsender from https://packetsender.com/ and instal it on a remote PC on the same network as the sim PC. This can be used to test the connection and sent UDP packets to the sim.

The UDP data packet is in four parts the header 2152, filter ffff, the size Byte 0100 then the data 01.

There is an error in the Majestic documentation , the correct header for the Dash8 is 21 52

21 52 FF FF ***03 10*** **01 00** 01 is the data to switch on the master battery

21 52 FF FF ***03 10*** **01 00** 00 will switch off the master battery

Note the information from Rienhard about reversing the bytes . The varlist index lists 1003 as master battery, however, its two bytes and you need to send the 03 first and then the 10.

Header	Filter	Size	Data
21 52	ffff	01 00	01
This is in every command sent	This will be used to filter types of inputs and outputs. Unfortunately until we get the training edition we are limited to connector 3	Byte the size of the data	01 switch on 00 switch off Some inputs have a 02 and 03 for multi position switches

In Packet sender enter the **IP address** of the flightsim PC and **port number 49260**

then type 21 52 FF FF ***03 10*** **01 00** 01 into the hex box in packet sender and press send. The master battery switch should move to On . if you change the last 1 to 0 and

press send again it should switch off. This proves the sim PC is receiving UDP data into the Dash8.

Hardware

I am using Ardunio Mega boards with an ethernet shield, just under £20 with about 50 inputs.

https://www.ebay.co.uk/itm/Arduino-Mega-2560-Compatible-Board-W5100-Ethernet-Shield-Lan-ATMega328-/263101689976?hash=item3d42153078

Download Ardunio IDE from https://arduino.cc/

After much messing about I managed to put together this code (This is the first code I had ever done . I only just understand what it is doing. I am sure that someone who knows what they are doing may have a much better way of scripting it)

This is the code for the DC control panel . It looks a lot, but is just a lot of copy and past with the datagram and name changed for each switch.

Remember you need to change the IP address to the sim pc and set the ardunio ip address. The button numbers are the pin numbers on the Ardunio that I have the switches connected to .

You must have Button.h installed in the adrunio library

Start of Ardunio code sketch

```
#include <Button.h>

#include <SPI.h>

#include <Ethernet.h>

#include <EthernetUdp.h>



EthernetUDP Udp;

byte mac[] = {0x90, 0xA2, 0xDA, 0x0D, 0x5C, 0x18};    // mac of ethernet shield

IPAddress ip(192, 168, 0, 177);                       // ip of Ardunio Mega
```

```
unsigned int localport = 8888;


IPAddress remoteIP(192, 168, 0, 178);              // ip of target computer to send datagram too, Sim PC running P3D with Q400
unsigned int remotePort = 49260;                   // listening port of tatget computer


byte BattMasterOn[10] = {0x21, 0x52, 0xff, 0xff, 0x03, 0x10, 0x01, 0x00, 0x01};      // Datagram to send when switch is moved to on
byte BattMasterOff[10] = {0x21, 0x52, 0xff, 0xff, 0x03, 0x10, 0x01, 0x00, 0x00};     //Datagram to send when switch is moved to off


byte BattMainOn[10] = {0x21, 0x52, 0xff, 0xff, 0x02, 0x10, 0x01, 0x00, 0x01};      // Datagram to send when switch is moved to on
byte BattMainOff[10] = {0x21, 0x52, 0xff, 0xff, 0x02, 0x10, 0x01, 0x00, 0x00};     //Datagram to send when switch is moved to off


byte BattAuxOn[10] = {0x21, 0x52, 0xff, 0xff, 0x01, 0x10, 0x01, 0x00, 0x01};      // Datagram to send when switch is moved to on
byte BattAuxOff[10] = {0x21, 0x52, 0xff, 0xff, 0x01, 0x10, 0x01, 0x00, 0x00};     //Datagram to send when switch is moved to off


byte BattStbyOn[10] = {0x21, 0x52, 0xff, 0xff, 0x00, 0x10, 0x01, 0x00, 0x01};      // Datagram to send when switch is moved to on
byte BattStbyOff[10] = {0x21, 0x52, 0xff, 0xff, 0x00, 0x10, 0x01, 0x00, 0x00};     //Datagram to send when switch is moved to off


byte EXT_DC_PWR_On[10] = {0x21, 0x52, 0xff, 0xff, 0x08, 0x10, 0x01, 0x00, 0x01};      // Datagram to send when switch is moved to on
byte EXT_DC_PWR_Off[10] = {0x21, 0x52, 0xff, 0xff, 0x08, 0x10, 0x01, 0x00, 0x00};     //Datagram to send when switch is moved to off


byte Bus_Fault_Reset[10] = {0x21, 0x52, 0xff, 0xff, 0x07, 0x10, 0x01, 0x00, 0x01};      // Datagram to send when switch is moved to on
byte Bus_Fault_clear[10] = {0x21, 0x52, 0xff, 0xff, 0x07, 0x10, 0x01, 0x00, 0x00};      // Datagram to send when switch is moved to on
```

```
byte MainBustieON[10] = {0x21, 0x52, 0xff, 0xff, 0x06, 0x10, 0x01, 0x00, 0x01};       // Datagram to send when switch is moved to on

byte MainBustieOff[10] = {0x21, 0x52, 0xff, 0xff, 0x06, 0x10, 0x01, 0x00, 0x00};      //Datagram to send when switch is moved to off


byte DC_GEN_1_ON[10] = {0x21, 0x52, 0xff, 0xff, 0x04, 0x10, 0x01, 0x00, 0x01};       // Datagram to send when switch is moved to on

byte DC_GEN_1_OFF[10] = {0x21, 0x52, 0xff, 0xff, 0x04, 0x10, 0x01, 0x00, 0x00};      //Datagram to send when switch is moved to off


byte DC_GEN_2_ON[10] = {0x21, 0x52, 0xff, 0xff, 0x05, 0x10, 0x01, 0x00, 0x01};       // Datagram to send when switch is moved to on

byte DC_GEN_2_OFF[10] = {0x21, 0x52, 0xff, 0xff, 0x05, 0x10, 0x01, 0x00, 0x00};      //Datagram to send when switch is moved to off


byte AC_EXT_PWR_ON[10] = {0x21, 0x52, 0xff, 0xff, 0x09, 0x10, 0x01, 0x00, 0x01};       // Datagram to send when switch is moved to on

byte AC_EXT_PWR_OFF[10] = {0x21, 0x52, 0xff, 0xff, 0x09, 0x10, 0x01, 0x00, 0x00};      //Datagram to send when switch is moved to off



Button button12(12); // Connect your button (toggle switch)  between pin and GND

Button button11(11);

Button button10(10);

Button button9(9);

Button button8(8);

Button button7(7);

Button button6(6);

Button button5(5);

//Button button4(4);

Button button3(3);


void setup() {
```

```
 button12.begin();
 button11.begin();
 button10.begin();
 button9.begin();
 button8.begin();
 button7.begin();
 button6.begin();
 button5.begin();
// button4.begin();
 button3.begin();

 Serial.begin(9600);
 Ethernet.begin(mac, ip);
 Serial.print("IP : ");
 Serial.println(Ethernet.localIP());
 Udp.begin(localport);
}

void loop() {

 if (button12.toggled())
 {
   if (button12.read() == Button::PRESSED) {
     Serial.println ("Master Battery On");      // Just to test it
     Udp.beginPacket(remoteIP, remotePort);
     Udp.write(BattMasterOn, 10);              // Datagram for on
     Udp.endPacket();
   }
   else {
     Serial.println("Master Battery Off");     // Just to test it
     Udp.beginPacket(remoteIP, remotePort);
```

```
   Udp.write (BattMasterOff, 10);          // Datagram for off
   Udp.endPacket();
  }
}

if (button10.toggled())
{
  if (button10.read() == Button::PRESSED)  {
   Serial.println ("Main Battery On");      // Just to test it
   Udp.beginPacket(remoteIP, remotePort);
   Udp.write(BattMainOn, 10);              // Datagram for on
   Udp.endPacket();
  }
  else {
   Serial.println("Main Battery Off");     // Just to test it
   Udp.beginPacket(remoteIP, remotePort);
   Udp.write (BattMainOff, 10);          // Datagram for off
   Udp.endPacket();
  }
}

if (button11.toggled())
{
  if (button11.read() == Button::PRESSED)   {
   Serial.println ("Aux Battery On");      // Just to test it
   Udp.beginPacket(remoteIP, remotePort);
   Udp.write(BattAuxOn, 10);              // Datagram for on
   Udp.endPacket();
  }
  else {
   Serial.println("Aux Battery Off");     // Just to test it
```

```
    Udp.beginPacket(remoteIP, remotePort);
    Udp.write (BattAuxOff, 10);          // Datagram for off
    Udp.endPacket();
  }
}

if (button9.toggled())
{
  if (button9.read() == Button::PRESSED)   {
    Serial.println ("Stby Battery On");      // Just to test it
    Udp.beginPacket(remoteIP, remotePort);
    Udp.write(BattStbyOn, 10);             // Datagram for on
    Udp.endPacket();
  }
  else {
    Serial.println("Stby Battery Off");     // Just to test it
    Udp.beginPacket(remoteIP, remotePort);
    Udp.write (BattStbyOff, 10);          // Datagram for off
    Udp.endPacket();
  }
}

if (button8.toggled())
{
  if (button8.read() == Button::PRESSED)   {
    Serial.println ("EXT_DC_PWR_On");      // Just to test it
    Udp.beginPacket(remoteIP, remotePort);
    Udp.write(EXT_DC_PWR_On, 10);             // Datagram for on
    Udp.endPacket();
  }
  else {
```

```
   Serial.println("EXT_DC_PWR_off");     // Just to test it
   Udp.beginPacket(remoteIP, remotePort);
   Udp.write (EXT_DC_PWR_Off, 10);          // Datagram for off
   Udp.endPacket();
  }
}


if (button7.toggled())
{
  if (button7.read() == Button::PRESSED)   {
   Serial.println ("Bus_Fault_Reset");      // Just to test it
   Udp.beginPacket(remoteIP, remotePort);
   Udp.write(Bus_Fault_Reset, 10);             // Datagram for on
   Udp.endPacket();
  }
  else {
   Serial.println("Bus_Fault_clear");     // Just to test it
   Udp.beginPacket(remoteIP, remotePort);
   Udp.write (Bus_Fault_clear, 10);          // Datagram for off
   Udp.endPacket();
  }
}

if (button6.toggled())
{
  if (button6.read() == Button::PRESSED)   {
   Serial.println ("Main BUS tie On");      // Just to test it
   Udp.beginPacket(remoteIP, remotePort);
   Udp.write(MainBustieON, 10);             // Datagram for on
   Udp.endPacket();
```

```
  }
  else {
    Serial.println("Main BUS tie Off");      // Just to test it
    Udp.beginPacket(remoteIP, remotePort);
    Udp.write (MainBustieOff, 10);           // Datagram for off
    Udp.endPacket();
  }
}

if (button5.toggled())
{
  if (button5.read() == Button::PRESSED)   {
    Serial.println ("DC_GEN_2_ON");      // Just to test it
    Udp.beginPacket(remoteIP, remotePort);
    Udp.write(DC_GEN_2_ON, 10);              // Datagram for on
    Udp.endPacket();
  }
  else {
    Serial.println("DC_GEN_2_OFF");      // Just to test it
    Udp.beginPacket(remoteIP, remotePort);
    Udp.write (DC_GEN_2_OFF, 10);           // Datagram for off
    Udp.endPacket();
  }
}

if (button3.toggled())
{
  if (button3.read() == Button::PRESSED)   {
    Serial.println ("DC_GEN_1_ON");      // Just to test it
    Udp.beginPacket(remoteIP, remotePort);
    Udp.write(DC_GEN_1_ON, 10);              // Datagram for on
```

```
    Udp.endPacket();
   }
   else {
    Serial.println("DC_GEN_1_OFF");     // Just to test it
    Udp.beginPacket(remoteIP, remotePort);
    Udp.write (DC_GEN_1_OFF, 10);          // Datagram for off
    Udp.endPacket();
   }
  }
}
```