

Data Hog Plus-U

User's Manual

Version : 2.16

Date : 02/16/2022

Data Hog Plus-U: High speed and low power battery or USB powered programmable serial port controller. Convert between USB serial, RS232, +3.3V logic level serial, SPI, and I2C in any combination and direction with optional auto-baud detection. SD card with Data Logger, Protocol Analyzer, custom batch programs, and Upload/Download Files and Firmware. Supplies up to 250mA @ +3.3V power out plus six GPIO pins with interrupts, 12 bit ADC conversion, and 10 segment bar graph. Command line interface, API, timer, and high accuracy Clock/Calendar. Complete programmable control and monitoring of USB, RS232, I2C/SPI, logic level serial and GPIO.

Table of Contents

1 – Hardware	4
1.1 – Battery & Power	4
1.2 – Clock and Clock Battery	5
1.3 – RS232 Serial Port & Null Modem Adapter	5
1.4 – USB Port	5
1.5 – SD Card	6
1.6 – +3.3V Logic Level Serial, SPI, & I2C Ports	6
1.7 – GPIO	7
1.8 – Bug Reports & Change Requests	7
2 – Ten Segment Bar Graph LED Display	8
3 – Serial Communications Protocol	10
3.1 – Sign-on (or Log-on) to Command Mode	10
3.2 – List of Terminal Commands	10
3.3 – Error Codes	12
3.4 – Terminal Command Details	13
3.5 – Low Power Modes	24
4 – Programming (Batch File) Protocol	26
4.1 – Batch File Commands	27
4.2 – Batch File Examples	36
5 – Updating Data Hog Plus Firmware	37
5.1 – D-Terminal Program	37
6 – USB, RS232, and Logic Level Serial Ports	38
6.1 – Logic Level Serial Port	39
7 – SPI and I2C Ports	41
7.1 – SPI Port	41
7.2 – I2C Port	44
8 – Converting/Adapting, Logging, & Protocol Analyzing	45
8.1 – Converting/Adapting	45
8.2 – Data Logging/Protocol Analyzing during Connect	47
8.3 – Data Capture (logging)	49
9 – GPIO (General Purpose Input/Output)	52
9.1 – GPIO Inputs	54
9.2 – GPIO Outputs	55
9.3 – EXTERNAL TRIGGER System	56
9.4 – ADC (Temperature & Voltage Monitor) System	62
10 – Complete Example (SPI Radio Module Monitor)	66

Terminal & Batch Command Reference:

Command		Page
<i>Clock</i>	CK, TIME, DATE	13
<i>New Firmware</i>	NF	14
<i>Password & Model</i>	PWD, MODEL	14
<i>Resetting</i>	RESET	15
	RESTART	32
<i>LED Control</i>	LEDS	15
	LEDLEVEL	15
	LEDO	15
<i>Baud Rate</i>	BAUD	16
	BOOTBAUD	16
	AUTOBAUD	16
<i>Power</i>	SLEEP	17
	IDLE	27
	DEEPSLEEP	17
<i>Serial Number</i>	SERIALNUM	17
<i>SD Card</i>	MEM	18
	MEM=INIT!	18
	FORMAT	18
<i>Directories</i>	CD	18
	DIR	19
	TREE	19
	MD	19
	RD	20
	DEFDIR	21
	DLDIR / ULDIR	21
<i>Files</i>	DEL	20
	REN	20
	COPY	21
<i>File Output</i>	TYPE	21
	TYPEH	21
	TABS	21
<i>Store Text File</i>	STORE	22
	STOREA	22
<i>Store Text/Binary</i>	FASTLOG	49
	TRIGLOG	49
	MAXFILESIZE	51
<i>Download</i>	DL / DLY	23
	DLX	23
<i>Upload</i>	UL / ULY	23
	ULX	23
<i>GPIO Pin Control</i>	GPIO	53
	TRIGGER	56
	ADC	62

Command		Page
<i>Executing Batch File</i>	RUN	27
	CALL	28
	EXIT	27
<i>Batch Control</i>	GOTO	31
	IF	31
	LOOP	32
<i>Comment</i>	REM	28
	;	28
	/	28
<i>Timer</i>	TIMER	28
<i>Serial Input / Output</i>	RCV	30
	SEND	29
<i>File Access</i>	OPEN	33
	OPENN	33
	OPENA	33
	CLOSE	33
<i>File Read/Write</i>	INPUT	33
	OUTPUT	28
	SEEK	34
	SEEKSTART	34
	SEEKEND	34
	LINESEEK	34
<i>Set Variable</i>	%x = <val>	35

<i>Adapting/Converting</i>	CONNECT	46
<i>Protocol Analysis</i>	CONNECT=LOG	47
	CONNECT=LOGHTML	47

<i>USB Port:</i>	SENDU, RCVU, DLU, ULU, BAUDU, BOOTBAUDU, etc.	38
<i>RS232 Port:</i>	SEND1, RCV1, DL1, UL1, BAUD1, BOOTBAUD1, etc.	38
<i>+3.3V Serial:</i>	SEND2, RCV2, DL2, UL2, BAUD2, BOOTBAUD2, etc.	38
<i>SPI Port:</i>	SENDS, RCVS, SPICS, SPI	42
<i>I2C Port:</i>	SENDI, RCVI, I2C	44

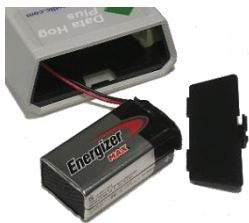
1 – Hardware

The Data Hog Plus (DH+) is a compact battery powered device with the following features:

- 9V Battery Compartment.
- USB Port (“B” Style).
- RS232 serial port connector (DB9 Male).
- Power On/Off switch and 10 segment LED bar graph.
- SD Card Slot.
- Side port for power input/output.
- Side port for SPI/I2C/Logic Level Serial connection.
- Internal temperature compensated high accuracy clock/calendar with battery.



1.1 – Battery & Power



The DH+ will operate from an input voltage of around 4V (5V or higher recommended) to as high as 16V DC. It draws from ~1.5mA to as much as 70mA when all the LED’s are on and RS232 port is in use. To make the unit portable and easy to use, a small 9V battery compartment is located at the top end of the case. Simply open the port and install a 9V battery or plug in a USB cable to power unit from USB port. When both USB and a battery are present, the battery is used first.

Depending on what the unit is being used for, a typical 9V battery will last from 24 hours to as much as 2 weeks of continuous use. Many users use lower modes like **SLEEP** and **DEEPSLEEP** or only turn on the unit when it is needed, so the DH+ lasts much longer under typical circumstances. If providing power by USB, be sure to use a full power port. *See section 3.5 for information on using low power modes.*

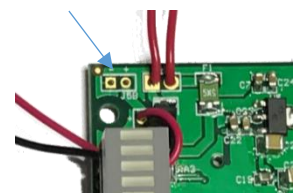
If USB power is not available, there are several ways other than an alkaline 9V battery to power DH+:

- 1) Connect a “9V Battery Adapter” such as the “Pinpoint 9-Volt Battery Universal AC Adapter Kit” (available on Amazon and other retailers). This simple and inexpensive device plugs into a wall socket and outputs 9V to a connector that plugs right into the 9V connector already on the DH+.
- 2) Use a higher capacity 9V battery such as the Energizer L522BP2 9V Lithium battery. This battery gives about 40% longer run time than a standard 9V battery. A second alternative is the 1200mAh Lithium Battery from PKCELL (Li-SOCI2) which will last twice as long as a conventional 9V battery.
- 3) Use the power input/output port that is present on the side of the DH+ shown here:



The first two pins (“GND” and “V+ IO”) serve as either power output (when a battery, USB, or other power source is installed) or as a power input if positive power and ground are applied to these pins. Do NOT apply more than 16V to the “V+ IO” pin or draw more than 250mA of power out of “V+ IO” or “+3.3V” pins!

For longer term use, the DH+ has secondary power input pads on the circuit board inside case. This is located as shown in this image: Simply solder wires to the board (+ and -) and connect the other end to a 4V to 16V source. Make sure you connect the pad labeled “+” to positive voltage and the pad labeled “-” to ground.



The commands **LEDS**, **SLEEP**, **DEEPSLEEP**, and **IDLE** can help lower the power draw for those applications that need the lowest power possible. See section 3 and 3.5 for more information.

1.2 – Clock and Clock Battery

The DH+ has a high accuracy temperature compensated real-time clock which tracks the current time and date. This is set at the factory to the U.S. West Coast/Pacific Time, but can be changed at any time by the user with the **TIME**, **DATE**, and **CK** commands (see section 3).

The clock is run from a 3V coin cell battery installed in the coin battery holder inside the case (as shown in the picture here) which preserves the current time/date even when the switch is in the OFF position. This battery will typically last from 5 to 8 years before it must be replaced.



If the clock battery gets low, the rightmost red LED will blink quickly after power on with no other LED's being lit. In this situation, you must replace the clock battery before the DH+ can be used. The battery used is a 3.2V Panasonic # CR-2032VP (or equivalent).

1.3 – RS232 Serial Port & Null Modem Adapter

To enable communication between two serial devices the transmitter from Device A must be tied to the receiver from Device B, and the receiver on Device A must be tied to the transmitter on Device B. For a typical setup like a computer connected to a modem, the computer has a Male DB9 connector with pins which is then plugged into a Female DB9 connector with sockets on a modem. The computer is called the “Data Terminal Equipment (or DTE)” and the modem is called a “Data Communication Equipment (or DCE)”. All this means is that the wiring inside the DB9 connector, cable, and modem is setup to tie the receive and transmit lines correctly to each other.

The Data Hog Plus has a male DB9 connector and acts just like a computer in the role of a DTE device. The male DB9 connector is designed to be plugged into things just like a computer is. This can cause a problem though if you want to connect the DH+ directly to another DTE device (like your computer) because you are now trying to plug to two DTE devices with male DB9 connectors together.

To get around this problem, the DH+ comes with a “NULL MODEM ADAPTER”. This has two female DB9 connectors on it and reverses the connection so that the DH+ can be connected directly to another DTE device like your computer. 99% of the time this is all that is required.

Unfortunately, some manufacturers don't follow the rule of male pins being DTE devices and female pins being DCE devices. In this case, you may need to customize your serial connection or purchase a “gender changer”. The following chart shows the exact pinout of the Data Hog Plus DB9 connector:

9 Pin Sub-D			
#1 - Carrier Detect	DCD	<-	(Input to Data Hog Plus)
#2 - Receive	RXD	<-	(Input to Data Hog Plus)
#3 - Transmit	TXD	->	(Output from Data Hog Plus)
#4 - Data Term. Rdy	DTR	->	(Output from Data Hog Plus)
#5 - Ground	GND	<->	(Common Ground)
#6 - Data Set Ready	DSR	<-	(Unused by Data Hog Plus)
#7 - Ready To Send	RTS	->	(Output from Data Hog Plus)
#8 - Clear To Send	CTS	<-	(Unused by Data Hog Plus)
#9 - Ring Detect	RNG	<-	(Unused by Data Hog Plus)

1.4 – USB Port

The DH+ contains a “B” style USB connector on the side of the case. Plug this into almost any Windows, Linux, or other device which supports USB. The DH+ uses the FTDI # FTX230XS chip for USB to Serial conversion and most operating systems support this chip directly and immediately. If you need a separate driver to install, please refer to www.ftdichip.com or install the “D-Terminal” application and follow instructions for installing the device driver.

1.5 – SD Card

The Data Hog Plus is designed to be used with a wide variety of SD Cards. Everything from high-speed Class 10 cards to low-speed Class 1 cards can normally be used without issue. However, use caution when purchasing or using very inexpensive SD Cards. Some manufacturers utilize low-quality memory which either fails outright or does not last more than a few read/write cycles. This is not a failure of the Data Hog Plus, but a result of a badly made memory card.

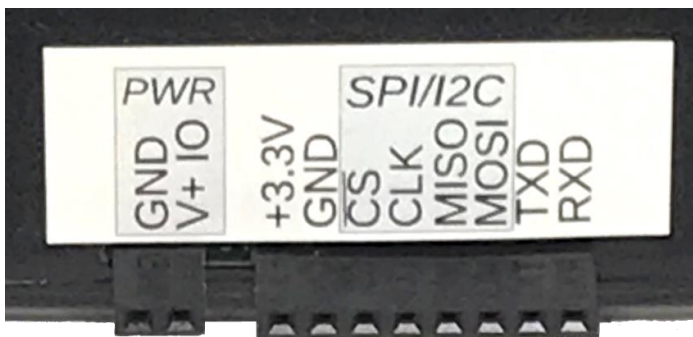
It is recommended to only use name brand SD Cards manufactured by companies like SanDisk, Kingston, Sony, and Lexar. Avoid any off-brand card which is sold extremely cheaply. These cards will not last and can give the false impression that the Data Hog Plus is not working, not to mention may cause the loss of collected data.

In addition, if you plan on using an SD Card for storing large amounts of frequently changing data (such as video or image files), select an “Industrial” or “High Endurance” type of card. These most often contain advanced features to prevent loss of data after many read/write cycles.

It should be noted that even a high-quality SD Card it will not last forever. Frequent write, read, erase cycles will eventually degrade even the best memory and will require replacement.

1.6 – +3.3V Logic Level Serial, SPI, & I2C Ports

On the side of the Data Hog Plus case there are two connectors – a 2 pin and an 8 pin. The two-pin connector is the power in/out port described in section 1.1 above. The eight-pin connector is the logic level serial and SPI/I2C port connection:



The two rightmost pins are the logic level serial (TXD & RXD). These pins along with the GND (Ground) pin allow you to connect the DH+ to almost any device with a logic level serial port. Note that “logic level” means that the port uses 2.8V to 3.3V for “high” and 0V for “Low”. This port is referred to as the second serial port (or “COM2”) for DH+ commands.

See section 6 for complete details.

The middle four pins are used for the SPI/I2C port. These are what is known as “synchronous” serial and are referred to by the system as the “S” port or “COMS” port when utilizing it as an SPI connection or as the “I” port or “COMI” when utilizing it as an I2C. See section 7 for complete details on using these ports.

WARNING: Although the pins are protected to some degree, you can still permanently damage the Data Hog Plus beyond repair by connecting it up to a high voltage, connecting to the wrong pin, or creating a short between power or other pins. ONLY change connections while the power switch is in the OFF position and carefully check all connections before proceeding. If you discover a problem, immediately remove power and make sure the unit is not warm before proceeding.

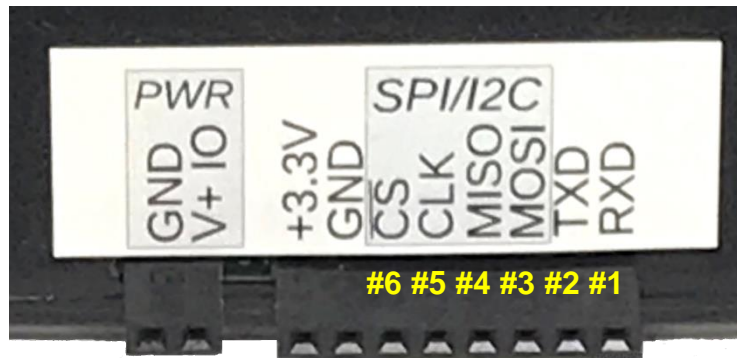
IMPORTANT: There is a resettable fuse built into the circuit board which trips automatically if more than about 0.5Amps of power is drawn from either the 9V battery or the USB input. This is a self-healing type of fuse which stays “open” as long as the drain is present, and the heals when the short is removed to “close” again. If your unit stops working due to a short circuit, disconnect all power and external devices and wait 5-10 minutes for the fuse to recover before powering it up again.

1.7 – GPIO

The Data Hog Plus supports up to six GPIO (General Purpose Input/Output) pins. These pins share the SPI/I2C and Logic Level Serial port pins so cannot be used at the same time if these other pins are enabled. GPIO is labeled #1 to #6 as follows:

Function	Variable Name	SEND/OUTPUT Control	Shares Pin With	ARM CPU Pin
GPIO #1	%G1	\G1x	RXD	PC0
GPIO #2	%G2	\G2x	TXD	PC1
GPIO #3	%G3	\G3x	MOSI	PB15
GPIO #4	%G4	\G4x	MISO	PB14
GPIO #5	%G5	\G5x	CLK	PB13
GPIO #6	%G6	\G6x	CS	PB12

Specifically, the GPIO pins are numbered like this:



Turning on GPIO functionality on any pin will disable the logic level serial (GPIO #1 or GPIO #2), the SPI (GPIO #3-#6), or the I2C (GPIO #3 or GPIO #4). Similarly, enabling the logic level serial, SPI, or I2C functions automatically turns off the GPIO (if enabled).

Each GPIO can be configured as an Input or Output. When configured as an Input, the GPIO can optionally be pulled High (to +3.3V) or Low (to Ground). Pull up or down resistance is ~40K ohms, with a “low” (or “0”) being around 1V or less and a “high” (or “1”) being around 2V or more.

When configured as an Output, the GPIO’s can source or sink +/- 8mA. Refer to the ST Microelectronics # STM32L451 data sheet for more specific information on the electrical characteristics of the GPIO pins.

See section 9 of the manual for more information on using the GPIO. It is recommended users become familiar with the terminal command line and batch files before trying to use GPIO.

EXTERNAL TRIGGER’s:

All of the GPIO’s can also be used in “EXTERNAL TRIGGER” mode which gives some advanced features when detecting state changes and to wake up from **DEEPSLEEP**. See section 9.3 for more information.

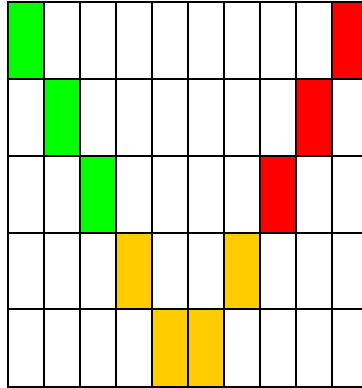
1.8 – Bug Reports & Change Requests

Diamond Edge Technology is dedicated to its products and customers. If you find a bug or have a new feature request, please email us at ryan@detllc.com. If the feature is one that we think others might also use, it is likely we can quickly add it for you and send you a firmware update. Thank you for your support of the Data Hog Plus-U!

2 – Ten Segment Bar Graph LED Display

The Data Hog Plus (DH+) indicates its current status with different displays on the ten segment LED bar graph. Users can custom program the display to show anything desired, or turn the display off to save power. By default, the LED display acts as follows:

1. **Power on/Initialization** - Two LEDs travel from the left and right towards the center, and then back again. This shows that the unit is powering on and initializing.



2. SD Card Status

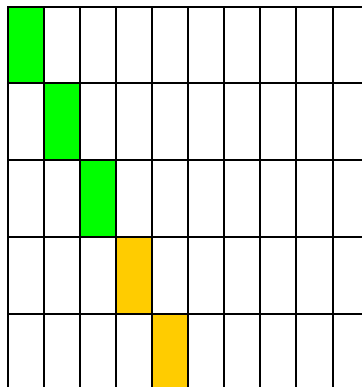
a. SD CARD INSTALLED

LED's will display from left to right (on a 1-10 scale) how much memory is used. For example, if 63% of the memory on the card is used to first 6 LED's will blink:



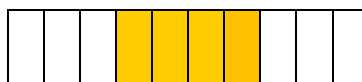
b. SD NOT INSTALLED

A single LED will blink rapidly from left to right again until a SD card is plugged in. Once SD is recognized, LEDs will show memory used and continue to step 3.



(and so on to 10th LED then back again)

3. **Serial Cable Not Connected** - The middle 4 LED's will blink while waiting for a serial cable to be plugged in:



4. Connected, Step 1:

- a. The Data Hog Plus by default will wait to receive a Ctrl+E (0x05) at 19200bps from the connected device. While this process is happening the 4th LED will blink:



- b. Once the Ctrl+E is received, it will switch to command mode indicated by the outside 2 LED's staying ON, with the 4th LED blinking.



NOTE: The Data Hog Plus can be configured to operate in many different ways when a cable is plugged in. It can automatically upload a file, send and receive custom strings and characters, and do almost anything else under your control. Refer to the “Serial Protocol” and “Batch File” sections for more information. By default, it simply waits for commands to be sent to it via the serial port.

5. **Connected, Step 2:** If passwords have been enabled, the 5th LED will start blinking. Send the password (with “PWD=<password>” command) to continue:



6. **Connected, Step 3:** If the DH+ has been configured to start an upload or download, or it receives a command to start an upload or download, the 6th LED will start blinking until the download or upload starts:



Once the download or upload starts, it switches to the 7th LED blinking and LED's 3, 4, & 5 showing a changing graph of each block sent or received:



When the file is done downloading or uploading, it will switch back to the 6th LED blinking and wait for the next command.

7. **Connected, Step 4:** If the DH+ has been configured to start an upload or download immediately on power up, then at the end of that process it will pause with no LED's blinking and show the amount of memory left on the SD card like this:



(This shows ~90% of the memory used – No LED's are blinking!)

This display of memory used is stopped with another command being received (or receiving a Ctrl+E).

3 – Serial Communications Protocol

The Data Hog Plus (DH+) USB port and RS232 port can range in speed from 300 to 2Mbps (115Kbps is the max supported by many other products) with optional auto-baud rate detection. The USB/RS232 ports are normally used to connect the DH+ to a computer or any other device with a compatible port that you want to control, adapt, convert, or store data from.

The DH+ supports a set of terminal commands that any program or communication system can send while it is in “Command Mode”. By default, command mode becomes active when the DH+ receives a single Ctrl+E character (0x05) which is usually sent by a computer running a terminal program like “D-Terminal” while plugged into the USB port or the RS232 port (usually through a NULL-MODEM adapter) or through the logic level serial port.

3.1 – Sign-on (or Log-on) to Command Mode

Send a single Ctrl+E (0x05) character to the DH+ to enter “Command Mode”. While in Command Mode, you can send different commands that control the DH+ like downloading files, formatting the SD card, etc. This is often referred to as a “Terminal Emulator”. The response to the Ctrl+E is a sign on string like this:

DHP#200a 04/04/21

The values after the “**DHP#**” are the firmware version and date of firmware release. This signals the DH+ is now in *command mode* and is ready to process new commands (see below). It also verifies that you are connected and that the baud rate matches.

NOTE: If passwords are enabled, an asterisk (“*”) will appear at the end of the sign-on string. You must send the password with the “PWD=<password>” command before any other commands will be processed.

3.2 – List of Terminal Commands

The following commands are available:

SD Card File & Directory Specific Commands:

- DIR/DIR <path> - Displays current file directory of the SD card or the files in the sub-directory <path>
- CD <dir> - Changes the current working directory
- TYPE <file> - Outputs a text file.
TABS=<size> - Sets the size of tabs (0=Default, 1-100 fixed size with spaces)
- TYPEH <file> - Outputs a file as hexadecimal values.
- DEL <file> - Deletes a file (can also use ERASE).
- MD/RD <path> - Makes a directory or removes a directory
- REN <src><dst> - Renames <src> to <dst>
- COPY <src><dst>- Copies <src> to <dst>
- TREE - Displays a tree view of all existing directories from current directory.

Upload/Download Commands:

- DL / DLY - Starts a YModem download (puts data onto the DH+ SD Card). You can optionally specify a <path> and/or <file> name, though YModem supports this itself.
- DLX <file> - Starts XModem download (stores data on Data Hog Plus) into <file>.
- UL/ULY <file> - Starts YModem upload (data from Data Hog Plus to external device) of <file(s)>.
- ULY <file> - Starts XModem upload (data from Data Hog Plus to external device) or <file(s)>.
- DLDIR <path> - Sets the default download directory for downloading files into DH+.
- ULDIR <path> - Sets the default upload directory for uploading files from DH+.
- STORE <file> - Starts a text file download into file named <file>. If <file> is blank, will create a file named “yyyy-mm-dd hhmmss.txt”.
- FASTLOG <file> - Similar to STORE except it highly prioritizes the capture (see section 8.3)
- TRIGLOG <file> - Similar to FASTLOG except GPIO triggers can exit the mode.

SD Card Specific Commands:

- `FORMAT=YES!` - Reformats the SD Card.
- `MEM` - Display SD Card status (Size, Amount Used, Disk Status)
- `MEM=INIT!` - Re-initializes SD Card link.
- `MAXFILESIZE` - Gets or sets the maximum file size for FASTLOG and TRIGLOG.

System Commands:

- `CK` - Clock Get or Set (time or date).
- `TIME / DATE` - Gets or Sets the Time and Date (`TIME` also allows setting time zone and DL Savings)
- `BAUD` - Changes the current baud rate (will revert to original baud on power up).
- `BOOTBAUD` - Changes the current baud rate and sets the default baud rate on power up.
- `AUTOBAUD` - Configures the auto-baud rate detect functionality.
- `MODEL` - Gets or Sets the type of Data Hog Plus and initializes options for that model.
- `DEFDIR <path>` - Sets the default directory when system powers on.
- `SERIALNUM` - Gets/sets the last 10 characters of the serial number (first 10 are set by DET and should not be changed by user).
- `LEDS` - Turns On or Off the LED's (turn off to save power).
- `LEDO` - Overrides other LED settings to force LED to specific state.
- `LEDLEVEL` - Sets the bar graph to a level from 0 to 10.
- `SLEEP` - Enters low power sleep mode.
- `DEEPSLEEP` - Enters VERY low power mode (see section 3.5)
- `PWD+` - Set Password
- `PWD=` - Send Password
- `NF=YES!` - Start Firmware Upload via YModem.
- `RESET` - Restores all settings to their original value (except Serial # and Manufacture Date)
- `TIMER=<action>` - Starts, Stops, Resets, or displays the high-speed timer.
- `CONNECT` - Enables tying the USB, RS232, and Logic Level serial ports together with or without a data logger and protocol analyzer.
- `GPIO` - Configures the GPIO pin functions including reading back values and setting outputs.
- `TRIGGER` - Configures and enables the GPIO "EXTERNAL TRIGGER" functions.
- `ADC` - Configures and enables Analog to Digital Conversion pins and temperature monitor.

Direct Control (often used with batch file execution):

- `SEND <text>` - Sends out a specific string. This is often used in combination with a Batch file.
- `RCV <text>` - Waits for a specific string to be sent back.
- `IF <text>` - Checks if the last thing received is equal to `<text>` and then jumps to `<label>` if it is.
- `GOTO <label>` - See the "Batch File Execution" section for more information.
- `LOOP <x><label>` - Repeats `<x>` number of times by jumping to `<label>`.
- `GOTO <label>` - Jumps to a point in the batch file.
- `REM <text>` - Marks the rest of the line as a "remark".
- `IDLE <time>` - Idles the system until a specific time or for a set number of seconds.
- `RUN <file>` - Runs a different batch file.
- `CALL <file>` - Runs a different batch file and then returns to current batch file when done.
- `RESTART <time>` - Reboots the Data Hog Plus immediately or after a certain amount of time.
- `EXIT <code>` - Exits out of running batch file.

File Read/Write (most often used with batch file execution):

- `OPENN <file>` - Opens a new file for writing. If the file exists, it is automatically overwritten.
- `OPENA <file>` - Opens a file for writing. If the file exists, new data is appended to the end.
- `OPEN <file>` - Opens an existing file for reading. If the file does not exist, an error is returned.
- `CLOSE` - Closes the file opened by `OPENN`, `OPENA`, or `OPEN`.
- `OUTPUT <data>` - Writes data to an open file.
- `INPUT <data>` - Reads data from an open file.
- `SEEK` - Jumps to a specific location in a file.
- `SEEKEND` - Jumps to end of a file (with optional offset)
- `SEEKSTART` - Jumps to beginning of a file.
- `LINESEEK` - Jumps to a specific line of a file.

3.3 – Error Codes

In the case of a terminal command formatting error, a batch file command error, or if something else goes wrong, the DH+ sends out one of the following error responses instead of either “OK” or the data requested.

ERR#00 (Unknown Command!)
ERR#01 (Password not sent!)
ERR#02 (Incorrect Parameters!)
ERR#03 (SD Card not installed!)
ERR#04 (SD Card not working!)
ERR#05 (SD Card Error!)
ERR#06 (File does not exist!)
ERR#07 (Directory error!)
ERR#08 (Directory does not exist!)
ERR#09 (Directory already exists!)
ERR#10 (Program Flash Upload Error!)
ERR#11 (Rename error!)
ERR#12 (Formatting error!)
ERR#13 (Can't open file!)
ERR#14 (File Delete Error!)
ERR#15 (Download Error!)
ERR#16 (Upload Error!)
ERR#17 (Copy Error)
ERR#18 (SPI Error)
ERR#19 (Can't open Log File Error)

Batch File Errors:

ERR#20 (Goto Label Error)
ERR#21 (Send format error)
ERR#22 (If/Goto format error)
ERR#23 (CALL nesting error)
ERR#24 (Loop Error)
ERR#25 (File OPEN/CLOSE error)
ERR#26 (File Seek error)

3.4 – Terminal Command Details

The following commands are supported while logged on and are received by the USB, RS232, and/or logic level serial ports (see section 6).

ALL of these commands also work from inside of a batch file (see section 4). However, users will frequently use these commands directly from a terminal.

Terminal Command – Clock (CK)

Format: CK
CK=<time>
CK=<date>
CK=<date>,<time>

Returns the current time & date (with just **CK** on the command line) or will set the time or the date to a specific value. For example:

CK=12:23:00 (Sets the time to 12:23:00pm)
CK=08/11/17 (Sets the date to August 11th, 2017)
CK=12:23:00,08/11/17 (Sets the time to 12:23:00pm and date to August 11th, 2017)

Just using **CK** on the command line by itself returns the current time & date like this:

```
CK
Time=07:01:55 Date=09/12/17
```

Terminal Command – Clock Time & Date

Format: TIME or DATE
TIME=<time> or DATE=<date>

These are similar to **CK=<time>** and **CK=<date>** and are included to make it easier to remember the command for setting the system time and date.

In addition, you can add to the command **DLON** (to turn on Daylight Savings Adjust), **DLOFF** (to turn off daylight savings adjust, and **ZONE=<val>** to set the time zone where <val> is from “-11” to “+12” for the UTC zone. You can also use “PST”, “EST”, “MST”, “CST”, “AKST” to specify a specific zone by name. For example:

TIME=DLON, ZONE=PST - Sets Daylight Savings ON and sets the time zone to “-8” (Pacific standard time).
TIME=DLOFF, ZONE=-3 - Sets Daylight Savings OFF and sets the time zone to “-3”.

Terminal Command – Send/Set Password

Format: PWD=<x>
PWD+<x>

Sends or sets the current password. Use **PWD=<x>** to send the current password and DH+ will respond with “OK” if it is correct, or “ERR#02” if it is not. Passwords are NOT case sensitive.

Use **PWD+<x>** to set the password to a new string. This also turns on passwords if they are currently disabled.

Use **PWD+** to disable the password.

Terminal Command – New Flash (NF)**Format: NF=YES!**

This function starts a firmware update on the Data Hog Plus as follows:

- 1) After this command is sent, the program will send “YModem Start ...” and wait for the initial header block to be sent. Nothing has been done to the original code yet - and won't be until a valid header block is received. Power can be removed and reapplied and system will return to normal operation at this point.
- 2) Once the header block is received, the current program in memory will be erased and the system will download and store the uploaded code into the program flash memory.
- 3) At the end, system sends either “ERR!” on failure or “GO!” if succeeded.
- 4) Data Hog Plus does a hard reset and restarts.

A firmware update is a fairly complex process that must be performed in the exact steps indicated, otherwise a catastrophic failure could occur. You WILL BRICK THE UNIT AND VOID YOUR WARRANTY if you load something in other than approved Data Hog Plus code!

Make sure you are plugged into the USB port or your battery is fresh before starting this process.

Terminal Command – Set Model & Parameters (MODEL)**Format: MODEL****MODEL=<param>**

MODEL controls some basic ways the DH+ operates:

<pre>MODEL=U MODEL=U,<login>,<ul/dl> <file></pre>	<p>General purpose Data Hog Plus-U. Users can control the DH+ through the USB, RS232, or logic level serial port directly, use a batch file on the SD card, or use the Upload/Download option on power up.</p> <p>There several options:</p> <ul style="list-style-type: none"><login> - Set to “0” to disable needing the Ctrl+E to login, or “1” to enable.<ul/dl> - Specifies starting an Upload (ul/ulx/uly) or Download (dl/dlx/dly) with <file> as soon as a SD card and cable is detected. <p>NOTE: Start YModem downloads on power up without a file name like this:</p> <pre>MODEL=U, DLY ...or... MODEL=U, DL</pre> <p>The DH+ will store the file based on the file name included from the sender.</p>
---	--

For example, to set the DH+ to not require a Ctrl+E to logon before any other commands will be accepted:

```
model=U,0
Model="U" (Login=0/'No')
```

To set the DH+ to automatically begin a YMODEM download (data to be stored on the SD card), send:

```
model=u, DLY
Model="U" (Login=1/'Yes', DLY on Power Up="(determined by sender)")
```

To set the DH+ to automatically begin a YMODEM upload of file TEST.BIN on power up, send:

```
model=U, ULY TEST.BIN
Model="U" (Login=1/'Yes', ULY on Power Up="TEST.BIN")
```

Terminal Command – Reset to Defaults**Format: RESET**

Resets to the default settings. This does not affect the Serial Number, Baud Rate, or Model. All other settings are returned to their factory set values.

Terminal Command – LED Control**Format: LEDS
LEDS=<x>**

Turns ON or OFF the LED Display as follows:

- LEDS=0 - LED's are turned off which saves ~40mA of power draw from the battery. Can also use command **LEDS=OFF** for same result.
- LEDS=1 - LED's are turned on and operate normally (default). Can also use command **LEDS=ON** for same result.
- LEDS=2 - LED's are turned on only at very beginning, and then just blink every so often until downloading is complete (when they go back on). They will also come on to indicate an error. This is a lower battery usage option.
- LEDS=3 - LED's are a 0-10 scale controlled by **LEDLEVEL** command.

Terminal Command – LED Level**Format: LEDLEVEL=<x>**

Sets the LED level from 0-10 when **LEDS=3**.

Terminal Command – LED Override**Format: LEDO=OFF / ALL
LEDO=<x>,<y>**

The LED Override command allows for direct control over any or all of the 10 segment bar graph LED's. By default, it is set to "OFF". If one or more LED's are directly overridden, they will always display the value specified.

- LEDO=OFF - Turns off the LED override. All LED's revert to their non-overridden state.
- LEDO=ALL - Overrides all LED's and sets them to OFF by default.
- LEDO=<x>,<y> - Overrides LED # <x> (where x is from 1 to 10) to be ON (<y>=1) or OFF (<y>=0). Once overridden, the LED stays at the state specified until you change it with another LEDO=<x>,<y> or send LEDO=OFF to disable the override command.

For example, suppose you wanted LED # 10 to be ON when your batch file is running and OFF otherwise. You could put the following command at the beginning of the batch file:

```
LEDO=10,1
```

And put this command right before your batch file exits:

```
LEDO=10,0
```

This overrides any other operation that might change LED # 10 and will only be on when the batch file is running. To return the system to normal operation, send:

```
LEDO=OFF
```

Or send:

```
RESET
```

Terminal Command – Set Baud Rate**Format: BAUD=<bps>
BOOTBAUD=<bps>**

Sets the current baud rate (**BAUD**) or sets the current baud rate and sets the default rate the DH+ will use when it powers up (**BOOTBAUD**). Values can range from 300 to 2000000, but note that the serial port on your computer may not support rates above 115,200. Always use **BAUD** first to check system still responds before using **BOOTBAUD** to make sure you can cycle power and get the unit back.

Terminal Command – Auto Baud Rate**Format: AUTOBAUD=<on/off>
AUTOBAUD=<mode>**

The auto baud command turns on or off auto baud rate detection. By default this is OFF and the only way the baud rate changes is with the BAUD or BOOTBAUD commands.

To turn auto baud rate detection on, send the command:

```
AUTOBAUD=ON
```

You can also set the auto baud detection mode to one of four values as follows:

```
AUTOBAUD=FALLINGEDGE      (can be shortened to AUTOBAUD=FA)
```

```
AUTOBAUD=STARTBIT         (can be shortened to AUTOBAUD=ST)
```

```
AUTOBAUD=7FFRAME         (can be shortened to AUTOBAUD=7F)
```

```
AUTOBAUD=55FRAME         (can be shortened to AUTOBAUD=55)
```

These are four different ways auto baud rate detects changes in the rate. By default, “STARTBIT” is used and is usually fine for most users. Note that setting the mode automatically turns on auto baud rate detection if it was off. You can also turn it off with:

```
AUTOBAUD=OFF
```

To set both the On/Off and the mode, combine the two like this:

```
AUTOBAUD=ON, STARTBIT
```

NOTE 1: When converting/adapting from one port to another, such as from RS232 to USB, enabling auto baud on one channel will automatically adjust it on the other channel. See section 8 for more information on how this works.

NOTE 2: You cannot do auto baud with the logic level serial port (COM2). However, the rate on this port can be changed when auto-baud is enabled on either the RS232 or USB ports and the two channels are tied together with the “CONNECT” command. See section 8 for more info.

NOTE 3: Because the DH+ first must determine that the rate has changed, then determine the new rate from incoming data, it can take 2-5 characters before the auto-baud system identifies the new rate. Be sure to account for this when using auto baud changes in your system. If using command mode, often it works to just send the Ctrl+E character until you get a response at the new rate. However, sometimes it can work faster to send other characters (like a space). If it doesn't auto-baud with a Ctrl+E, try sending other characters to get a new rate recognized.

Terminal Command – Sleep**Format: SLEEP=<serial>,<time>**

Puts the DH+ into a low power sleep mode that will wake only when it receives a character on its serial port (if <serial>="on"), or after <time> number of seconds has passed. <time> can also be set to a specific "hh:mm:ss" value or the word "top" to wakeup at the top of the next hour.

For example:

SLEEP	- Goes to sleep until a character is received ("on" is assumed).
SLEEP 60	- Goes to sleep until 60 seconds has elapsed.
SLEEP on, 60	- Goes to sleep for 60 seconds or a character is received.
SLEEP 00:00:00	- Goes to sleep until midnight (serial is shutoff)
SLEEP top	- Goes to sleep until top of next hour (serial is shutoff). For example, if it currently 12:34 it would exit sleep at 13:00.
SLEEP on, top	- Goes to sleep until top of next hour or a serial character is received.

Sleep power draw and how long a typical 9V battery will last depends on a variety of factors.

- 1) If serial port wakeup is enabled, power draw will be ~22mA.
- 2) If serial port wakeup is disabled, any of the time based wakeups will draw ~10mA if a serial cable is connected, or ~8mA if no cable is connected.
- 3) Estimated battery life with 9V / 550mAh 22mA = 25 Hours 10mA = 55 Hours
Note that the Data Hog Plus can accept any DC power input from 5V to 16V. You can hook up an alternate power source to the 9V battery terminals, or to "J5B" on PCB.

NOTE: See section 3.5 for more information on using SLEEP and DEEPSLEEP!

Terminal Command – Set Serial Number**Format: SERIALNUM
SERIALNUM=<val>**

Gets or sets the last 9 characters of the Data Hog Plus serial number. The first 10 characters are set by the factory (DET) before shipping the DH+ and are not changeable by the user.

The factory serial number is set as follows:

DHPmmyyxxx - Where "mm" is the month of manufacture, "yy" is the year of manufacture, and "xxx" is the sequence number (3 digit hexadecimal value).

Setting your own serial number automatically adds a dash ("-") and then up to 9 characters of your choosing. For example, to set the last part to "MyDatahog", you would send:

```
SERIALNUM=MyDataHog
```

Resulting in a new serial number that looks something:

```
DHP07200BC-MyDataHog
```

Note that you can request the Serial Number without first sending the password (if enabled). However, you cannot change it without sending the password.

Terminal Command – Memory Status (MEM)**Format: MEM**

Returns the current SD Card status information, or an error message if no card has been found. The DH+ responds with something similar to this:

```
Total Memory: 59GB
Memory Used : 0GB (0%)
SD Card Name: 46HSG V8.0
```

Terminal Command – Memory Initialize**Format: MEM=INIT!**

Re-initializes the SD Card interface. Will return “OK” or “ERR#05” after the initialization.

Terminal Command – Format SD Card**Format: FORMAT=YES!**

Re-formats the installed SD Card. Will return “OK” or “ERR#05” after formatting to indicate the status of the SD Card. Note that cards are formatted to exFAT to maximize their broad usage and large files.

Terminal Command – Change Directory**Format: CD**
CD <path>
CD ..

The DH+ starts by default in the root directory of the SD card, although this can be changed with the **DEFDIR** command. Sending **CD** by itself will return a string indicating the current directory:

```
cd
Current Directory = "(root)"
```

To change to a sub-directory send **CD <path>** where **<path>** is the name of the directory. For example:

```
cd Dudley
Current Directory = "Dudly"
```

If the directory does not exist, an error will be displayed. To go back up one level, send:

```
cd ..
```

To go back to the root directory regardless of where you currently are, send:

```
cd /
```

<paths> are relative to your current position, unless you put a slash in front of the path name. For example, suppose you are currently in “/Dudly/Test” and you want to go to “/Boo/Test/2” you would send:

```
cd /boo/test/2
```

Note that **<path>**'s are not case sensitive. The DH+ also treats forward slashes (“/”) and backward slashes (“\”) the same, but will always report directory names using a forward slash.

You can optionally surround path names with quote marks. However, this is rarely necessary since the DH+ assumes everything after **CD** is the directory name. Only with the **REN** and **COPY** commands are quote marks usually needed.

Terminal Command – Directory Listing**Format: DIR
DIR <path>**

To list all the files and directories in directory send the **DIR** command. Sending it without a specified <path> will list files in the current directory, sending it with a <path> shows files from that directory. For example:

```
dir
-----
Directory Listing of "dudly":
-----
2020-07-19  23:15:12  <DIR>          test 1
2020-07-19  23:15:14  <DIR>          test 2
2020-07-19  23:15:22  <DIR>          Test 3

                0 Files                0 bytes
                3 Dirs   15931015168 bytes free
```

DIR works almost exactly the same as the Command Prompt in Windows. The DH+ starts by default in the root directory of the SD card, although this can be changed with the **DEFDIR** command.

DIR also supports wildcard characters such as:

DIR *.txt - Displays only files ending in ".txt" from current directory.

Terminal Command – Directory Tree View**Format: TREE
TREE <path>**

Displays a tree structure view of all the directories on the SD card. With nothing specified for <path> it starts at the current directory, otherwise you can use:

```
TREE \ - Always display entire directory tree from the root directory.
TREE "\Foo" - Displays tree view of the Foo directory off the root.
TREE ".." - Displays a tree view starting in the directory up one level from this one.
TREE "Foo" - Displays a tree view starting in sub-directory Foo from current directory.
```

Terminal Command – Make Directory**Format: MD <path>**

Makes a new directory. For example, to make a new directory from the current directory send:

```
md Foo Bar
```

Or you can send:

```
md "Foo Bar"
```

Another way to always make the directory from the root:

```
md "\Foo Bar"
```

Terminal Command – Remove Directory**Format: RD <path>**

Removes a directory and all files and sub-directories in that directory. For example, to remove the director “Foo Bar” from the root you can send:

```
rd "\Foo Bar"
```

This can be a dangerous command because it can quickly remove a lot of files. Use it carefully!

Terminal Command – Delete File**Format: DEL <file>**

Deletes a file from the SD card. For example, to remove the file “Dud.txt” from the root you can send:

```
del "\Dud.txt"
```

To delete a file in the current directory, send:

```
del "dud.txt"
```

You can also use wild cards in the file name. To erase all files ending in “.txt”. send:

```
del *.txt
```

NOTE: For compatibility purposes, you can also use **ERASE** exactly the same as **DEL**.

Terminal Command – Rename File or Directory**Format: REN <src> <dst>**

Renames <src> to <dst>. This command works on files or directories, and can also move a file from one directory to another.

This command benefits from surrounding the <src> and <dst> with quote marks (helps to identify them), and is required if the <src> has any spaces in the name.

To rename file “Foo Bar.txt” to “Foo.bin”, send the command:

```
ren "Foo Bar.txt" "Foo.bin"
```

Terminal Command – Copy File or Directory**Format: COPY <src> <dst>**

Copies <src> to <dst>. This command works on files or directories, and wildcards can be used for the <src> but not for the <dst>.

This command benefits from surrounding the <src> and <dst> with quote marks (helps to identify them), and is required if the <src> has any spaces in the name.

To copy the file “Foo Bar.txt” to “Foo.bin”, send the command:

```
copy "Foo Bar.txt" "Foo.bin"
```

Terminal Command – Type out a File**Format:** **TYPE** <file>
TYPEH <file>
TABS <size>

Outputs the contents of <file> to the serial port. Output can be aborted by sending a Ctrl+X (0x18) or Ctrl+Z (0x1A). Renames <src> to <dst>.

TYPE outputs the file as straight text. **TYPEH** outputs it as a hexadecimal conversion.

By default **TYPE** outputs tabs as a straight character (0x09). However, you can set it with the **TABS** command as follows:

- TABS=0 - Output tab characters as a straight 0x09.
- TABS=<size> - Replace tabs with spaces and set <size> to 1-100 in width.

Terminal Command – Default Directory**Format:** **DEFDIR**
DEFDIR <path>
DLDIR
DLDIR <path>
ULDIR
ULDIR <path>**Download Directory****Upload Directory**

These commands display or set the default file directory to use on power up (**DEFDIR**), the directory to store files in when doing a download (**DLDIR**), and the default directory to use when uploading files from the DH+ (**ULDIR**).

Note that when downloading files using the YMODEM protocol, a file directory can be part of the name. In this case, the file is always stored as though the **DLDIR** is the root directory.

In a similar way, when uploading files with YMODEM the **ULDIR** is treated as the root directory and is not included in any path name for the file.

The **DEFDIR** value will automatically be created on any SD Card when it is installed in the DH+.

Creates a new file named <file> and beginning storing everything received from the serial port into the file. File is closed automatically on receipt of Ctrl+X (0x18) or Ctrl+Z (0x1A) or if the cable is unplugged. If no file name is given, then a file named “yyyy-mm-dd hhmmss.txt” is automatically created.

STOREA differs from **STORE** in that it will automatically append new data to the end of <file> if it exists (otherwise it creates a new file). **STORE** will always overwrite any existing file with the new data.

Note that characters are not echoed back during storage. The DH+ assumes this is being done under software control and does not need to see the actual data being stored. To view the text after it has been stored, use the **TYPE** command.

The file directory used to store the file uses the following logic:

- 1) If the file name starts with a slash (“/”) and then a file name, then it will always go in the root directory.
- 2) If the file name is just a single slash (**STORE /**), then a file named in the format “yyyy-mm-dd hhmmss.txt” will go in the root directory.
- 3) If the default download directory is blank, or you start the file name with “./”, then the file will be stored in the current directory plus whatever directory has been specified in <file>.
- 4) All other situations will store the file in the default download directory plus whatever directory has been specified in file.

Some examples:

```
STORE
```

Creates a “yyyy-mm-dd hhmmss.txt” file in the default download directory.

```
STORE /
```

Creates a “yyyy-mm-dd hhmmss.txt” file in the root directory.

```
STORE ./
```

Creates a “yyyy-mm-dd hhmmss.txt” file in the current directory.

```
STORE Dud.txt
```

Creates file “Dud.txt” in the default download directory.

```
STORE /Dud.txt
```

Creates file “Dud.txt” in the root directory.

```
STORE /MyDir/Dud.txt
```

Creates file “Dud.txt” in the “/MyDir” directory.

```
STORE ./Dud.txt
```

Creates file “Dud.txt” in the current directory.

```
STORE ./MyDir/Dud.txt
```

Creates file “Dud.txt” in the sub-directory “MyDir” from the current directory.

```
STOREA Dud.txt
```

Appends new text data to file “Dud.txt” in the default download directory.

NOTE: It is recommended to use **FASTLOG** or **TRIGLOG** instead of **STORE** if your capture rate is 230400 baud or higher. This will insure you won't miss any data (see section 8.3).

Terminal Command – Download File**Format: DL / DL <file>
DLY / DLY <file>
DLX <file>**

Download transfers a file from a remote system (like a computer) to the Data Hog Plus SD card. The parameters for **<file>** are the same as the **STORE** command (see above).

DL and **DLY** are identical and start a YMODEM CRC protocol download (either 1K or 128 byte), including allowing the sending device to specify the **<file>** (use **DL** or **DLY** without a file name to allow this). The Data Hog Plus implements standard YMODEM with 1K block sizes and a CRC check.

DLX does an XMODEM download instead. You must specify the file name for XMODEM.

Refer to the following for specific details on how XMODEM and YMODEM works:

<http://textfiles.com/programming/ymodem.txt>

<http://web.mit.edu/6.115/www/amulet/xmodem.htm>

Once the download is complete the DH+ will send “OK (Download Complete!)”.

NOTE: **DLY** supports multiple file transfer. The sending device can send as many files as it wants and the DH+ will store them as received.

Terminal Command – Upload File**Format: UL <file>
ULY <file>
ULX <file>**

Upload sends a file from the SD card on the DH+ to a remote device over the serial port. The options for **<file>** are the same as the **STORE** command (see above).

UL and **ULY** are identical and start a YMODEM 1K with CRC protocol upload with the file named. **ULX** starts a XMODEM 1K transfer with CRC upload.

Refer to the following for specific details on how XMODEM and YMODEM works:

<http://textfiles.com/programming/ymodem.txt>

<http://web.mit.edu/6.115/www/amulet/xmodem.htm>

Once the upload is complete the DH+ will send “OK (Upload Complete!)”.

NOTE: **UL** and **ULY** supports multiple file transfer. Simply specify a wildcard in the file name to select more than one file to send. For example:

UL *.* - Sends all files in the directory.
UL *.txt - Sends all files ending in “.txt”.

3.5 – Low Power Modes

The DH+ has low power modes that will allow the 9V battery to last up to two weeks by disabling some functions. The following commands are used to control power:

SLEEP / IDLE	<p>All functions except the LED's remain on and active and the unit enters a lower power mode waiting for either (a) specific amount of time to lapse (such as 60 seconds), (b) the clock reaches a specific time (such as 8am), (c) an "EXTERNAL TRIGGER" activates (see section 9.3), or (d) you enable exiting SLEEP/IDLE with the receipt of a character on any of the serial channels.</p> <p>SLEEP and IDLE are identical except that if you exit from low power mode using a character received on a serial channel, SLEEP reads that character out and IDLE does not. With IDLE, the character used to wake the device is available to the next RCV command or in the general buffer, If SLEEP/IDLE is set to wake from a character received on a serial channel, the unit power draw is around 9mA if nothing is connected to the RS232 port (or serial wakeup disabled) or 21mA if a serial cable is plugged into the RS232 port (and serial wakeup enabled). This will result in a typical 9V battery life of 50 hours (no RS232 cable) or 24 hours (RS232 cable plugged in). Using the logic level serial, SPI/I2C, or USB port does not increase Data Hog Plus power draw, so you can expect around 50 hours when using those channels.</p>
DEEPSLEEP	<p>This mode disables the LED's, RS232, USB, and logic level serial ports and all other serial communication channels. It can only exit deep sleep after (a) a number of seconds elapses, (b) the clock reaches a specific date and time, or (c) if the "EXTERNAL TRIGGER" system is activated (see section 9.3).</p> <p>DEEPSLEEP is much lower power than SLEEP or IDLE and draws around 1.5mA of current from the battery. This equals about 350 hours of battery life from a typical 9V battery, or around 2 weeks. DEEPSLEEP is ideal when you want to record activations from any of the EXTERNAL TRIGGERS or if you want to do something periodically (such as every day, hour, or minute).</p>

Command – Sleep

Format: SLEEP=<serial>,<time>

Puts the DH+ into a low power sleep mode that will wake only when it receives a character on its serial port (if <serial>="on"), or after <time> number of seconds has passed. <time> can also be set to a specific "hh:mm:ss" value or the word "top" to wakeup at the top of the next hour.

For example:

- | | |
|----------------|--|
| SLEEP | - Goes to sleep until a character is received ("on" is assumed). |
| SLEEP 60 | - Goes to sleep until 60 seconds has elapsed. |
| SLEEP on, 60 | - Goes to sleep for 60 seconds or a character is received. |
| SLEEP 00:00:00 | - Goes to sleep until midnight (serial is shutoff) |
| SLEEP 16:00 | - Goes to sleep until 4pm (serial is shutoff, seconds assumed to be ":00") |
| SLEEP top | - Goes to sleep until top of next hour (serial is shutoff). For example, if it currently 12:34 it would exit sleep at 13:00. |
| SLEEP on, top | - Goes to sleep until top of next hour or a serial character is received. |

Command – Idle

Format: IDLE=<serial>,<time>

The **IDLE** command is identical to the **SLEEP** command except that if it is set to wake from a serial character received, that character is not read out and is available to the next command.

For example, suppose you enter:

IDLE

And the DH+ receives the character 'R'. That 'R' character will become the first character read out by the next **RCV** command instead of being discarded.

Puts DH+ into a very low power mode that wakes only when (a) specific time/date is reached or specific number of seconds have elapsed, or (b) an “EXTERNAL TRIGGER” event occurs (see section 9.3).

<time> can be the number of seconds you want to be in **DEEPSLEEP** or it can be “top” to **DEEPSLEEP** until the top of the next hour. It can also be a specific time and/or date. Note that an “EXTERNAL TRIGGER” will wake the unit up from **DEEPSLEEP** regardless of the specified time (see section 9.3).

For example:

DEEPSLEEP 60	- Goes into deep sleep for 60 seconds.
DEEPSLEEP 00:00:00	- Goes into deep sleep until midnight.
DEEPSLEEP 00:00	- Same as the above (if no seconds specified, assumes “00”).
DEEPSLEEP top	- Goes to deep sleep until top of next hour (serial is shutoff). For example, if it currently 12:34 it would exit sleep at 13:00.
DEEPSLEEP 16:00	- Goes into deep sleep until 4pm.
DEEPSLEEP 10/22	- Deep sleep until 00:00 on October 22 nd of the current year (if it is not yet Oct 22 nd) or October 22 nd of the next year (if it is already past Oct 22 nd).
DEEPSLEEP 10/22/22	- Deep sleep until October 22 nd of 2022.
DEEPSLEEP 01:00,10/22	- Goes into deep sleep until 1am on October 22 nd .

While in **DEEPSLEEP** the DH+ draws less than 1.5mA of current which makes most 9V batteries last about 2 weeks. This time will be shortened the more time it spends out of **DEEPSLEEP**.

As an example, suppose you wanted to send the current time out the RS232 port at the top of each hour. You can create any file and then **RUN** it, or use a batch file named `AUTORUN.BAT` located in the root directory which would run automatically on power up. The batch file could contain these lines:

```
BAUD1=19200
:MainLoop
SEND1 "Current Time=%H:%N:%S\r\n"
DEEPSLEEP TOP
GOTO MainLoop
```

This starts out by setting the COM1 (RS232 port) baud rate to 19200bps and then sending the current time. The DH+ then enters **DEEPSLEEP** until the top of the next hour. **DEEPSLEEP** is exited at the top of the next hour and the program jumps back to “MainLoop” and repeats!

The only way to exit the above program is to pull the SD card out. If you were using a file named `AUTORUN.BAT`, you would need to rename this file to anything else before inserting it into the DH+ or it would automatically start again.

As an alternative, you can also name the file `AUTOSTART.BAT`. This works exactly the same as `AUTORUN.BAT` except that it only executes on initial power up or after a **RESTART** command. This is handy because you can pop the SD card out and plug it back in while the unit is powered on and it won't automatically restart until you cycle power or use the **RESTART** command. This lets you use other commands without cycling power on the unit.

4 – Programming (Batch File) Protocol

The Data Hog Plus (DH+) supports a complete batch file system which allows control over the unit and what it is connected to. Simply by placing a file in the root directory of the inserted SD Card, you can instruct the Data Hog Plus to send out characters or strings, logged received data from the main USB, RS232 port or the logic level serial and SPI/I2C ports, wait for certain responses, and upload/download different files depending on the responses received.

This feature also allows users to insert different SD cards for different purposes, and makes the Data Hog Plus almost infinitely configurable as a portable serial controller, data retriever, and programmer.

All batch files are formatted in standard text file format with one command per line ending in a CR/LF. Leading spaces or tabs are ignored, and comments can be inserted by starting the line with a semicolon (“;”), a forward slash (“/”), or the command “REM”. For example:

```
REM This is my batch file to change to the \DATA directory
REM and display the file "HELLO.TXT".
CD \DATA
TYPE "HELLO.TXT"
```

An identical version of this batch file would be:

```
; This is my batch file to change to the \DATA directory
; and type out the file "HELLO.TXT".
CD \DATA
TYPE "HELLO.TXT"
```

Comments should be used liberally throughout the batch file to make clarification easier later on if you need to modify the file in any way.

Note that unless a batch file has a **GOTO** command, it will automatically end after the last command is executed. You can terminate a batch file before the last command using the **EXIT** command:

```
; This is my batch file to change to the \DATA directory
; and type out the file "HELLO.TXT".
CD \DATA
EXIT
TYPE "HELLO.TXT"
```

In this case the Data Hog Plus will change to the \DATA directory, but it won't type out "HELLO.TXT" because the **EXIT** command terminates it immediately.

Batch files can be any length and can do all sorts of advanced things like:

- Send out characters and wait for responses.
- Start uploads or downloads.
- Run other batch files.
- Run any regular terminal command (not just batch file commands).
- Use **IF/GOTO** and **LOOP** statements to branch around the batch file and execute different commands based on certain things happening.

<p>NOTE: On power up (or when a new SD card is inserted) the DHP looks to see if "AUTOEXEC.BAT" or "AUTORUN.BAT" exists in the root directory of the SD card. If found, it automatically runs it. Otherwise, batch files are only executed in response to the RUN command. "AUTOSTART.BAT" will also automatically run, but only on power up or after RESTART command. This is helpful to stop the program by unplugging and then re-inserting the SD card.</p>
--

Batch files are a combination of a very simple programming language with a list of file, directory, and data transmission commands that operate on the SD card and the connected device. See 5A – BATCH FILE COMMANDS for more information.

4.1 – Batch File Commands

The following commands are used with batch files. Note that most of them can be used just like any Terminal Command described in section 3, but normally are most useful with batch file execution. A few commands (like **GOTO**) only relate to batch files.

Batch File Command – Run Batch File

Format: RUN <file>

Executes a batch file. By custom, batch files should end in “.BAT” but this is not required by the system. Note that running a batch file from another batch file is allowed, but when the new batch file ends it does NOT return to the old file (use the **CALL** command instead if you want to return).

The standard file name conventions apply like this:

- RUN TEST.BAT - Runs the batch file named “TEST.BAT” in the current directory.
- RUN \TEST.BAT - Runs the batch file named “TEST.BAT” in the root directory.
- RUN FOO\TEST.BAT - Runs the batch file named “TEST.BAT” in the FOO sub-directory from the current directory.

NOTE: If the file does not exist exactly as entered, the DH+ will automatically search for a file ending in “.BAT” with the same name and run that instead. For example, if you sent:

```
RUN TEST
```

If a file named “TEST” does not exist but a file named “TEST.BAT” does, then DH+ will run “TEST.BAT”

Batch File Command – Exit Batch File

**Format: EXIT
EXIT <code>**

Stops execution of batch file. Optionally, you can exit with a code which can be used in other batch files to do different things (see the “**CALL**” command). For example:

```
CD \DATA  
EXIT  
CD \
```

In the above batch file example, the current working directory will stay “\DATA” because it exits out of running the batch before getting to the last line.

EXIT is especially helpful with the **CALL**, **IF/GOTO**, and **RUN** batch file commands.

Batch File Command – Idle

Format: IDLE=<serial>,<time>

The **IDLE** command is identical to the **SLEEP** command except that if it is set to wake from a serial character received, that character is not read out and is available to the next command.

For example, suppose you enter:

```
IDLE
```

And the DH+ receives the character ‘R’. That ‘R’ character will become the first character read out by the next **RCV** command instead of being discarded.

Batch File Command – Call Batch File**Format: CALL <file>**

Executes a different batch file, similar to RUN, but when the new batch file exits it returns to the current batch file at the same position. In addition, the %0 variable will be set to the **EXIT** code.

Consider the following two batch files:

File Main.BAT:

```
:Top
SEND "Type in a string and press Enter: "
RCV "%s"
IF %1="" GOTO Done
CALL ShowAnswer.BAT
GOTO Top
:Done
SEND "\r\nDONE!\r\n"
```

File ShowAnswer.BAT:

```
SEND "\r\nANSWER=\"%1%\r\n"
```

Running Main.BAT will cause the DH+ to ask for a string. When the user types one in and presses Enter, it then calls "ShowAnswer.BAT" to display the results. If the user types an empty string, it quits.

Note that all variables (%1 through %9) are common to all functions. The %0 variable will always contain the **EXIT** code of the last batch file that exited, with a default value of "0" if no other value specified.

You can **CALL** other batch files from inside of batch files that are themselves **CALL**ed. However, you cannot go more than ten layers deep.

Batch File Command – Remark (Command)**Format: REM ...**

```
; ...
/ ...
```

Simply marks the rest of the line as a comment and is ignored by the Data Hog Plus. For example:

```
REM This is my batch file to change
; This is my batch file
/ This is my batch file
// This is my batch file
```

All of these result in the rest of the line being ignored.

Batch File Command – High Speed Timer**Format: TIMER
TIMER=START/STOP
TIMER=RESET**

The timer is a high-speed timer that can be reset, started, and stopped at any time. To access the current value, simply send **TIMER** or use the "%t", "%T", "%e", or "%E" variables inside of **SEND**.

The timer is useful for various functions and helps in storing timestamps in files.

Batch File Command – Send Data Out**Format: SEND <format>**

SEND can be a simple or complex command. In the simplest form, you simply list the characters you want to output like this:

SEND "hello"

And the DH+ would send out the current port: hello

Note that quote marks are not included unless you put a backslash in front of them like this:

SEND "\"hello\""" And the DH+ would output: "hello"

It is not required to start and end what you are going to send with quotes, but it is generally a good idea to avoid confusion. Here are some of the other special values that can be included:

Code	Explanation
\r	Carriage Return (0x0D)
\n	New Line (0x0A)
\t	Tab Character (0x09)
\"	Double quote
\%	Percent (cannot just include these because they are used for parameters, see below)
\\	Backslash character
\0 or \1	When sending out SPI/I2C in Master/Manual, \0 sets the CS to "0" and \1 sets it to "1".
\Gyx	Sets a GPIO output ("y" from 1-6) to value "x" (0 or 1). See section 9.
\D or \d	Enables DTR on RS232 (\D) or disables it (\d).
\xYY	A single character represented by the hex value of YY. For example, to output an 0xFE byte, you would use: SEND "\"\xFE" You can string multiple characters together too: SEND "\"\xFE\x7F\xFF\x00"
\pSSS	Does not send a character but pauses for SSS number of seconds. For example, to send the character R, pause for 10 seconds, and then a T, use: SEND "R\p010T"
\mNNN	Repeats the previous character NNN times. For example: SEND "-\m077\r\n*\m077\r\n" This would output a dash 78 total times, a new line (CR/LF), then an asterisk 78 times, then a final new line (CR/LF).

The percent character (%) is special purpose. It allows you to insert previously saved values that are usually loaded by the **RCV** command. Here are the % codes:

Code	Explanation
%1 - %9	Saved variables from RCV command. For example, the following receives line from the serial port ended by a Carriage Return (0x0D) and then returns it back followed by a CR/LF: RCV "%s" SEND "Received: %1\r\n" When this is run, the DH+ waits to receive a line of text and when it is received (indicated by getting a Carriage Return), it sends back what it got saying that is stored in %1: Received: TEXT Refer to the batch file examples and the RCV command for more information.
%0	The last EXIT code received ("0" is the default).
%h %n %s %H %N %S	The current Hour (%h), Minute (%n), and Second (%s) Same as the above except always as two digits.
%y %m %d %Y %M %D	The current Year (%y), Month (%m), and Day (%d) Same as the above except always 4 digit and 2 digit.
%r %a / %b / %u %i	Set to a "1" if any data is ready on serial port to be received, or a "0" if not. %a is the same except it is always on COM1 (RS232), %b is COM2, and %u is USB. %i is the same except it is for if a character has been received as an SPI/I2C slave.
%t or %T %e or %E	High Speed Timer as "days hh:mm:ss.msec" (%t) or "days.fractime" (%T). High Speed Timer as "seconds.msec" (%e) or "hh:mm:ss.msec" (%E).
%c	Returns current SPI Chip Select value or 1 or 0 (valid normally in SLAVE SPI/I2C mode).
%G1 to %G6	Returns the current GPIO value (see section 9).
%X1 to %X6 %x1 to %x6 %K1 to %K6	Returns the last external trigger time and date (see section 9.3). Returns the last external trigger time and date as Unix Epoch Time (see section 9.3). Returns "1" when external trigger activated (automatically cleared to "0" after read).
%f	Returns 1 if RS232 Carrier Detect (DCD) is active, or 0 if it is not active.
%v1/2 & %V1/2 %vt/T & %Vt/T	Returns the current ADC value from ADC #1, ADC #2. Capital 'V' is integer, 'v' is float. Use "%vt" for Fahrenheit and "%vT" for Celsius. Capital 'V' is integer, 'v' is float.

RCV inputs data from the port. If the USB is connected or a RS232 cable is detected when the command starts, losing that connection during it will automatically abort the **RCV**.

There are two types of things that can be received – fixed characters and variables. Fixed characters are specific characters that must be read before moving on. Variables are different kinds of values that can be specified and then are stored in the %1 through %9 variables.

For example, to simply wait until the characters “GO” are received, use the command: `RCV "GO"`

The DH+ will wait until “GO” is received before finishing. Note that it has to be ‘G’ then ‘o’ in that order, any other combination will not match. To wait for “GO” and then one more character of any type, use:

```
RCV "GO%c"
```

The “%c” in this case says to receive a single character. It is automatically saved in the %1 variable because it is the first thing being received. To do the same thing with two characters:

```
RCV "GO%c%c"
```

Now %1 will equal the first character received after “GO”, and %2 will equal the second character.

Alternately, you can use:

```
RCV "GO%2c"
```

Both characters that are then stored in %1 (one after the other). See the **IF/GOTO** command to perform checks on the variables and change the batch file flow based on what is received.

Regular characters have special codes for certain characters (like the **SEND** command):

<i>Code</i>	<i>Explanation</i>
<code>\r</code>	Carriage Return (0x0D)
<code>\n</code>	New Line (0x0A)
<code>\t</code>	Tab Character (0x09)
<code>\"</code>	Double quote
<code>\%</code>	Percent (cannot just include these because they are used for parameters, see above)
<code>\\</code>	Backslash character
<code>\xYY</code>	A single character represented by the hex value of YY. For example, to wait to receive a 0xFE, you would say: <code>RCV "\xFE"</code>
<code>\mNNN</code>	Repeats previous character NNN times.
<code>\0 or \1</code>	When receiving from SPI/I2C in Master/Manual, \0 sets the CS to “0” and \1 sets it to “1”.
<code>\Gyx</code>	Sets a GPIO output (“y” from 1-6) to value “x” (0 or 1). See section 9.
<code>\D or \d</code>	Enables DTR on RS232 (\D) or disables it (\d).

Format of the “%” commands:

<i>Code</i>	<i>Explanation</i>
<code>%c</code>	Single character (Tab, punctuation, or upper/lower case letter or number).
<code>%<num>c</code>	Multiple characters specified by <num>. For example, “%5c” loads five characters.
<code>%b</code>	Single byte of any value.
<code>%<num>b</code>	Multiple bytes specified by <num>. For example, “%5b” loads five bytes.
<code>%B</code>	Same as “%b” above except it is stored as a decimal number in variable.
<code>%2B</code>	Stores a word in memory (High-Low) as a decimal number in variable.
<code>%Y</code>	Same as “%b” above except it is stored as a hex number in variable.
<code>%2Y</code>	Stores a word in memory (High-Low) as a hex number in variable.
<code>%s</code>	String terminated by a CR (0x0D), LF (0x0A), Break (0x03), or Abort (0x18). Note that the character that terminates the string is not included in the result.
<code>%S</code>	Same as “%s” except response must be longer than zero length.
<code>%i</code>	Integer terminated by a comma, CR (0x0D), LF (0x0A), Break (0x03), or Abort (0x18).
<code>%<num>i</code>	Receives a specific number of integer characters.
<code>%f</code>	Floating point terminated by comma, CR (0x0D), LF (0x0A), Break (0x03), Abort (0x18).
<code>%<num>f</code>	Receives a specific number of floating point characters.
<code>%x</code>	Hex number terminated by comma, CR (0x0D), LF (0x0A), Break (0x03), or Abort (0x18).
<code>%<num>x</code>	Receives a specific number of hex characters.

Batch File Command – Goto Label**Format: GOTO <label>**

The **GOTO** command jumps to a new line in the batch file. This line can be either before or after the **GOTO** command, and is identified by a unique <label> identifier that starts with a colon (":"). For example:

```
CD \  
GOTO NEWLABEL  
CD \NEWDATA  
EXIT  
:NEWLABEL  
CD \OLDDATA
```

This batch file jumps around the “CD \NEWDATA” to “NEWLABEL” and then runs “CD \OLDDATA”. **GOTO** is especially useful when combined with **IF**, **CALL**, and **RUN** commands.

Batch File Command – If Then Goto**Format: IF <condition> GOTO <label>**

The **IF** command tests the results of previously stored values to see if they match specified conditions. If they do, then the batch file will branch to the label specified after the **GOTO**.

This command can be hard for those new to programming to understand, but it is fairly simple. Consider the following batch file:

```
:Top  
SEND "Type 0 or 1: "  
RCV "%c"  
IF %1 = "0" GOTO GotZero  
IF %1 = "1" GOTO GotOne  
SEND "\r\nERROR: Did not type 0 or 1!\r\n"  
GOTO Top  
:GotZero  
SEND "\r\nGOT A ZERO!\r\n"  
EXIT  
:GotOne  
SEND "\r\nGOT A ONE!\r\n"
```

This code starts out by sending “Type 0 or 1: “. It then waits for a character to be received back. Once that is received, it checks to see if it is a “0” and jumps to the line after the “GotZero” label if it is. If it is a “1”, it jumps to the line after the “GotOne” label. If it is neither of these, it sends an error message and then retries!

You can see how the “RCV %c” is used to receive the character and it is automatically stored in the %1 variable. This %1 value is what the IF command uses to check against to see what was actually received.

IF supports the following comparisons:

= (Equals)	< (Less Than)	>= (Greater Than or Equals)
!= (Not Equals)	> (Greater Than)	<= (Less Than or Equals)
.x (Bit “x” is of variable is equal to 0 or 1)		

For example:

```
IF %4 >= "12" GOTO MyLabel
```

If %4 is greater than or equal to 12, it jumps to MyLabel. You can also put numbers after the comparison (like IF %1 = 1 GOTO MyLabel) or hex numbers (like IF %3 > 0x32 GOTO MyLabel).

Batch File Command – Loop Goto**Format: LOOP <count> GOTO <label>**

The **LOOP/GOTO** command allows the batch file to repeat a loop a specified number of times. Consider the following batch file:

```
:Top
SEND "Here!\r\n"
LOOP 10 GOTO Top
```

This would send out the string "Here!" ten times. You can also use a % variable for the count:

```
SEND "How many times? "
RCV "%i"
SEND "\r\n"
:Top
SEND "Here!\r\n"
LOOP %1 GOTO Top
```

The **RCV** command will store an integer in %1, which will control how many times it loops and sends the "Here!" line.

NOTE: A loop count of one or zero results in no action. The DH+ assumes the first time through you have done one loop. In addition, don't use nested **LOOPS** inside of **CALLs**. The DH+ only supports one **LOOP** at a time, and a loop inside a **CALL** may result in the main batch file **LOOP** being repeated infinitely.

Batch File Command – Restart Data Hog Plus**Format: RESTART
RESTART=<sec>
RESTART=<time>
RESTART=TOP**

Restarts the Data Hog Plus. This command is identical to switching the power switch off and then on again.

Optionally include the number of seconds you want to wait or a specific time you want to restart at. During this period the Data Hog Plus ignores everything and won't respond again until this time has passed.

For example:

```
RESTART          - Immediately restarts
RESTART 60       - Goes to sleep and restarts after 60 seconds.
RESTART 00:00:00 - Goes to sleep and restarts at midnight.
RESTART 12:00    - Goes to sleep and restarts at noon (seconds are assumed to be "00").
RESTART top      - Goes to sleep and restarts at top of next hour. For example, if it currently 12:34
                  it would exit and restart at 13:00.
```

Batch File Command – File Open**Format: OPEN <file>
OPENN <file>
OPENA <file>**

Opens an existing file or creates a new file named <file> and begins reading everything from it when the **INPUT** command is used or storing everything sent by the **OUTPUT** command to it.

OPENN creates a new file, and **OPENA** creates a new file if none exists or appends new data to the end of any existing file. **OPEN** is used to read data out of existing files.

The file directory used has the following logic:

- 1) If the file name starts with a slash ("/") and then a file name, then it will always go in the root directory.
- 2) If the file name is just a single slash, then a file "yyyy-mm-dd hhmss.txt" will go in the root directory.
- 3) If the default download directory is blank, or you start the file name with ".", then the file will be stored in the current directory plus whatever directory has been specified in <file>.
- 4) All other situations will store the file in the default download directory plus whatever directory has been specified in file.

Some examples:

```
OPENN
```

Creates a "yyyy-mm-dd hhmss.txt" file in the default download directory.

```
OPENN /
```

Creates a "yyyy-mm-dd hhmss.txt" file in the root directory.

```
OPENN ./
```

Creates a "yyyy-mm-dd hhmss.txt" file in the current directory.

```
OPENA Dud.txt
```

Creates file "Dud.txt" in the default download directory if it doesn't exist, otherwise adds to this file.

```
OPENA /Dud.txt
```

Creates file "Dud.txt" in the root directory if it doesn't exist, otherwise adds to this file.

```
OPEN /MyDir/Dud.txt
```

Opens the existing file "Dud.txt" in the "/MyDir" directory for the **INPUT** command.

Batch File Command – File Close**Format: CLOSE**

Closes any file that has been opened with the **OPEN**, **OPENN**, or **OPENA** command. This command is optional – the DH+ automatically closes the open file if another file is opened or the batch file exits.

Batch File Command – Input from File**Format: INPUT <text>**

The **INPUT** command is just like the **RCV** command, except it reads data from an existing file that has been opened with the **OPEN** command. The format of <text> is identical to the **RCV** command except the characters are read out from the file instead of from the serial port.

Note that any **INPUT** command automatically ends if the end of the file is reached and an error message is shown.

Batch File Command – Output to File**Format: OUTPUT <text>**

The **OUTPUT** command sends the data specified by <text> to the file previously opened with the **OPENN** or **OPENA** command. The format of <text> is identical to that described by the **RCV** command (see above). For example:

```
OPENN "MyFile.txt"
OUTPUT "This is a line of text!\r\n"
CLOSE
```

At the end of this sequence, the file "MyFile.txt" will contain "This is a line of text!" followed by a CR/LF. You can include % variables in the output as well as any backslash code.

If you wanted to append to an existing file, you would use:

```
OPENA "MyFile.txt"
OUTPUT "This is a line of text!\r\n"
CLOSE
```

In this example, every time the sequence is run another "This is a line of text!" is added to the file.

Batch File Command – Seek File Location**Format: SEEKSTART
SEEKEND
SEEK <position>
SEEKEND <position>
LINESEEK <line>**

When a file has been opened the seek command lets you move the file position. When the file position has been moved, the next **INPUT** or **OUTPUT** from or to that file will happen from the new position. For byte specific moves, use the following commands:

- SEEKSTART** - Jumps to the beginning of the file.
- SEEKEND** - Jumps to the end of the file.
- SEEKEND x** - Jumps to the end of the file minus the number of characters specified by "x".
- SEEK x** - Jumps to the byte location in file specified by "x".

If the file is a standard text style of file, with ending CR characters on each line, you can use:

- LINESEEK <linenum>** - Jumps to a specific line number with one (**LINESEEK 1**) being the very beginning of the file.

As previously described, the Data Hog Plus has nine main variables for use during batch file operation numbered %1 to %9. In addition, %0 represents the value of the last **EXIT** code which is used in combination with the **CALL** function.

Most of the time % variables are set with the **RCV** and **INPUT** commands. These read data out of the one of the communication ports or from a file, respectively. For example:

```
RCV "%s"
```

This will read a string of characters out of the communication port terminated with a CR and store the result in the %1 variable. If you used instead something like:

```
RCV "%c%s"
```

Then the first character would be stored in the %1 variable and then a string terminated by a CR would be stored in the %2 variable.

Another option is to assign directly to a variable a different string or variable. For example:

```
SEND "Type in string and press Enter: "  
RCV "%s"  
%2 = "Got \"%1\"!\r\n"  
SEND "\r\n%2"
```

This would get a string and store it in %1. It then assigns to the %2 variable the word "Got ", a quote, whatever was received by the **RCV** command, and then a trailing quote, exclamation, and CR\LF. It then sends that new %2 value back out.

Assigning values to variables is especially useful if you want to preserve time stamps or other values that might change between accessing them.

4.2 – Batch File Examples

Refer to the “SAMPLE BATCH FILES” sub-directory on the SD card that came with the Data Hog Plus. This directory contains several sub-directories with different kinds of example batch files.

Other examples may be present, but the following is a brief description:

Send Time Once Per Hour

Will send the time when it first runs, and then send it again at the top of each hour.

Serial Data Logger (Text)

When DH+ powers up, it immediately opens a new text file with a name based on the current date and time. All text received is stored in this file. The file is closed if the DH+ receives a Ctrl+X, and then another new text file is opened with a new date and time.

This allows you to use the DH+ to simply record all text data that it receives.

Serial Data Logger (YModem)

When DH+ powers up, it immediately starts a YModem download. At the end of the download, it will start another YModem download.

This allows you to use the DH+ to receive files via YModem continuously from whatever it is connected to. Since YModem has the file name in the header, the device it is connected to can set the file name to be saved into.

Serial Data Logger (Text or YModem)

Similar to the previous two examples except it first sends out a message saying “Send 0 for Text or 1 for YModem”. Once it receives one of those two characters, it starts either a Text or YModem download. When finished, it re-asks “Send 0 for Text or 1 for YModem”.

Stop Watch

Sends the message “Stop Watch (SPACE to Start/Stop):”. When it receives a SPACE or a CR, it starts the stop watch and sends out the hours, minutes, and seconds each second. Send another SPACE or CR to stop the timer.

Send Ctrl+X to abort and exit the Stop Watch example.

Stop Watch Logger

Same as Stop Watch except it also opens a text file and stores the stop watch times to a file named “STOPWATCH.TXT”. New data is appended to existing data.

Send Ctrl+X to abort and exit the Stop Watch example.

Data Time Stamp

Opens a new file based on the current date and time. Any character that is received is immediately stored in the file along with an exact time stamp of that character.

If a Ctrl+X is received, the file is closed and a new file is opened with the current date time. If a Ctrl+Z is received the batch file terminates.

Refer to the “SAMPLE BATCH FILES” sub-directory for more information and other examples. The DH+ is very customizable for your particular system needs!

5 – Updating Data Hog Plus Firmware

The Data Hog Plus firmware can be updated fairly easily. Diamond Edge Technology provides updates periodically (refer to www.dettllc.com) for firmware labeled like this: DHP_Vxxxxa.HEX

The “xxx” will be the version number and the “a” refers to any customization for specific customers. You can determine what version you are currently running by:

- 1) Open a terminal window on your computer set to the baud rate of the DHP and the COM port you have connected your USB cable or RS232 cable with NULL MODEM adapter to.
- 2) Send Ctrl+E to get the sign-on string like this:

```
DHP#200a 03/03/21
```

- 3) This shows the currently connected DHP is V2.00a.
- 4) It is a good idea to increase speed to 921.6kbps for firmware updates. Send the command:

```
BAUD=912600
```

And then change the terminal baud before starting the update. NOTE: You will have to switch back to 19200 baud after the update because the Data Hog Plus will revert to its original setting after restarting.

To update the firmware on the Data Hog Plus, send the following command: NF=YES!

The DHP will respond with a YMODEM Upload message. Start your YMODEM Upload of the new version of firmware. Once the upload is complete the DHP will automatically restart.

WARNING: You will “brick” your Data Hog Plus and void the warranty if you upload anything to it other than approved firmware from Diamond Edge. It is better You should also be sure to plug in your USB cable or have a fresh 9V battery installed before starting the process, and there is no chance that your computer will lose power during the transfer.

5.1 – D-Terminal Program

Diamond Edge provides a free program named “D-Terminal” to communicate to and control the Data Hog Plus. This program is included on the SD card that comes with the DHP and is also available as free download from www.dettllc.com from the Downloads page.

D-Terminal is a complete terminal emulator plus has special features for the Data Hog Plus. Simply install and run D-Terminal, connect the DHP to your computer, select the “Port” the Data Hog Plus is connected to, and click the “Data Hog Plus” tab at the bottom of the screen. All files and directories are shown and you can edit, view, copy, cut, and paste files between your computer and the DHP easily.

In addition, an “Update” button at the top of the DHP link page allows you to quickly update new firmware to the Data Hog. It is highly recommended to keep your DHP updated to the most current release. Note that this button only appears if a new version of firmware is available.

To access the “Update” function, do the following:

- 1) Install D-Terminal (if you haven’t already).
- 2) Copy any firmware update files to the directory:
C:\Program Files\DiamondEdge\DTerminal\Hexfiles
- 3) Run D-Terminal
- 4) Click “Port” and select the serial port the Data Hog is connected to (usually the USB or RS232 port, USB connection is recommended for updates).
- 5) Press <Enter> or Ctrl+E to make sure you get the sign-on string (“DHP#xxx ...”). This makes sure you are talking to the unit. NOTE: You may want to remove the SD card if you are running batch files because the unit won’t respond to Ctrl+E while it is running a program.
- 6) At the bottom of the screen is a tab named “Data Hog Plus” – click it.
- 7) The program will link to the unit. After a few seconds the “Update” button will appear if a new version is available in the directory specified above. Click the button and follow instructions!

That’s it. The update is automatic and will let you know when finished.

6 – USB, RS232, and Logic Level Serial Ports

These three communication channels (USB, RS232, and logic level serial) support the complete command line interface described in the previous sections. You can send almost any command from one of these ports and the DH+ will respond back out that same port.

For example, if you send the **DIR** command from the USB port, the DH+ will send the current directory back out the USB connection. This same **DIR** command from RS232 or logic level serial ports will similarly respond back to the same channel.

However, many commands allow you to specify where they will respond by appending a port code to the end of the command as follows:

1	- RS232 Port	I	- I2C Port
2	- Logic Level Serial	S	- SPI Port
U	- USB Port		

For example, from any port if you send the **DIRU** command (instead of just **DIR**) the directory will be sent out the USB port instead of back to the channel it received the command from. This is especially useful inside of batch files to specify specific channels for specific commands.

Commands that support RS232 (**1**), Logic Level Serial (**2**), USB (**U**), I2C (**I**), and SPI (**S**) channel specifiers in place of the “x” below:

- SENDx <text> - Sends out a specific string. This is normally used in combination with a Batch file.
- RCVx <text> - Waits for a specific character or string to be sent back.

Commands that support RS232 (**1**), Logic Level Serial (**2**), and USB (**U**) channel specifiers in place of the “x” below:

File & Directory Specific Commands:

- TYPEx <file> - Outputs a text file.
- TYPEHx <file> - Outputs a file as hexadecimal values.
- DIRx/DIRx <path> - Displays current file directory or the files in the sub-directory <path>
- TREEx - Displays a tree view of all existing directories from current directory.

Upload/Download Commands:

- DLx / DLYx - Starts a YModem download (stores data on Data Hog Plus). You can optionally specify a <path> and/or <file> name, even though YModem supports this itself.
- DLXx <file> - Starts XModem download (stores data on Data Hog Plus) into <file>.
- ULx/ULYx <file> - Starts YModem upload (data from Data Hog Plus to external device) of <file(s)>.
- ULYx <file> - Starts XModem upload (data from Data Hog Plus to external device) or <file(s)>.
- STOREx <file> - Starts a text file download into file named <file>. If <file> is blank, will create a file named “yyyy-mm-dd hhmmss.txt”
- STOREAx <file>
- FASTLOGx <file> - Same as STORE with a higher priority (see section 8.3).
- TRIGLOGx <file> - Same as FASTLOG but can exit with a GPIO TRIGGER (see section 8.3).

System Commands:

- BAUDx - Changes the current baud rate (will revert to original baud on power up)
- BOOTBAUDx - Changes the current baud rate and sets the default baud rate on power up.
- COMx - Enables or disables commands to be sent/received from COMx

Commands that support RS232 (**1**) or USB (**U**) channel specifiers in place of the “x” below:

- AUTOBAUDx - Enables or disables auto baud rate setting from COM1 (RS232) or USB (COMU)

For example:

- DIR1 - Sends current directory to RS232 Port
- DIR2 \DUD - Sends the “\DUD” directory to the logic level serial port.

This command enables or disables commands being sent/received from a specific COM port. For example:

COM2=DISABLE	- Turns off COM2 (logic level serial) as a source of commands.
COM2=0	- Same as above.
COM1=ENABLE	- Turns on COM1 (RS232) as a source of commands.
COM1=1	- Same as above.
COMU=ENABLE	- Turns on COMU (USB Port) as a source of commands.
COMU=1	- Same as above.

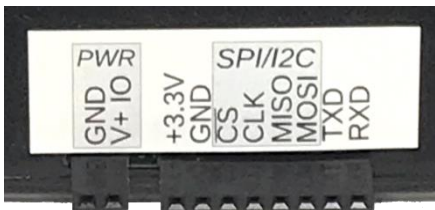
By default both COM ports and USB port work for commands and enquires. However, if using the Data Hog to control other devices on COM1, COM2, or USB port, it can sometimes be helpful to disable commands from the port.

The logic level serial (COM2) has two additional commands:

COM2=INVERT	- Inverts the logic on the RXD/TXD pins.
COM2=STANDARD	- Keeps the RXD/TXD pins at the standard logic level.

6.1 – Logic Level Serial Port

The logic level serial port has a Transmit (TXD) and Receive (RXD) line available on the DH+ side port. These two lines, along with a Ground (GND) connection allow you to connect the DH+ to any logic level serial device:



The rightmost pin labeled “RXD” is the receive input and the second from right labeled “TXD” is the transmit output. These operate at +3.3V and support speeds up to 2Mbps.

The logic level port is named “Serial 2” for all commands.

To use the port, connect the “GND” (Ground) pin to the second systems ground pin, connect the “TXD” (Transmit) pin to the second systems RXD (Receive) pin, and connect the “RXD” pin to the second systems TXD (Transmit) pin.

System Command – Pass Through**Format: PASSTHROUGH
PASSTHROUGH=<x>**

NOTE: The “PASSTHROUGH” command is a holdover from a previous firmware release. It is recommended to instead use the newer “CONNECT” command (see section 8) for linking ports.

Use pass through to enable a transparent passthrough of data between COM1 (RS232) and COM2 (logic level serial). When this mode is active, anything received on COM1 will be sent to COM2 and vice-a-versa. By default, the mode is exited whenever a 0x18 character is received from either channel.

It is important to set the baud rates of each port before starting passthrough. This is done with the **BAUD1=<baud>** and **BAUD2=<baud>** commands. Note that each port has a 10K buffer which cannot be exceeded, so if the baud rates are very different care must be taken to insure the buffers are not overrun.

PASSTHROUGH=<x> is the same except you can specify the character to be used to exit the passthrough mode. For example:

```
PASSTHROUGH=3
```

Would use a Ctrl+C (0x03) to exit the passthrough mode. Send:

```
PASSTHROUGH=-1
```

To disable exiting the passthrough mode (must cycle power to exit).

System Command – Pass Through with Log**Format: PASSTHROUGHLOG
PASSTHROUGHLOG=<x>**

NOTE: The “PASSTHROUGHLOG” command is a holdover from a previous firmware release. It is recommended to instead use the newer “CONNECT” command (see section 8) for linking ports.

Use pass through to enable a transparent passthrough of data between COM1 (RS232) and COM2 (Logic Level Serial). When this mode is active, anything received on COM1 will be sent to COM2 and vice-a-versa. By default, the mode is exited whenever a 0x18 character is received from either channel.

7 – SPI and I2C Ports

The SPI and I2C ports allow advanced users to control specialized serial channels. Under most circumstances, the USB, RS232, logic level serial, or a batch file is used to control the SPI or I2C port.

7.1 – SPI Port

Four of the pins on the Data Hog Plus side connector serve as a direct 3 or 4 wire “SPI” port. SPI is a universal synchronous style interface that can connect to a variety of different devices. This document does not describe SPI in detail (refer to other sources for this information), but the way SPI is used with the Data Hog Plus and a connected device is described below.

The Data Hog Plus SPI port uses four possible lines:

- MISO** - Master Input/Slave Output
- MOSI** - Master Output/Slave Input
- CLK** - Clock
- CS** - Chip Select

The SPI port can be configured as either a “slave” or “master” which affects how the above lines function. Under most circumstances the Data Hog Plus operates as the “Master” device and sends data to a connected SPI “Slave”.

SPI MASTER:

In “Master” mode the Data Hog Plus is controlling the communication to a SPI “Slave”. This is the most common configuration and sets the pins as follows:

Full Duplex:

<i>Pin</i>	<i>Input or Output</i>	<i>Function</i>
MISO	Input to DH+	Master Input / Slave Output
MOSI	Output from DH+	Master Output / Slave Input
CLK	Output from DH+	Clock Output
CS	Output from DH+	Chip Select (normally Active Low)

Half Duplex:

<i>Pin</i>	<i>Input or Output</i>	<i>Function</i>
MOSI	Master Transmit Output from DH+ Master Receive Input to DH+	Output when Transmitting, Input when Receiving.
CLK	Output from DH+	Clock Output
CS	Output from DH+	Chip Select (normally Active Low)

While in Master mode, you can send out the SPI port using the **SENDS** command and receive data in using the **RCVS** command. If the chip select line is set to manual, use the **SPICS** command to set it HIGH (1) or LOW (0) as needed. You can also use the `\0` and `\1` special codes embedded in the strings specified for sending and receiving.

SPI SLAVE:

In “Slave” mode the Data Hog Plus is controlled by an external master. Because it cannot control when data is sent or received, it must pre-set what it is going to return in response to a read request. Controlling the communication to a SPI “Slave”. This is the most common configuration and sets the pins as follows:

Full Duplex:

<i>Pin</i>	<i>Input or Output</i>	<i>Function</i>
MISO	Output from DH+	Master Input / Slave Output
MOSI	Input to DH+	Master Output / Slave Input
CLK	Input to DH+	Clock Input
CS	Input to DH+	Chip Select (normally Active Low)

Half Duplex:

<i>Pin</i>	<i>Input or Output</i>	<i>Function</i>
MOSI	Slave Transmit Output from DH+ Slave Receive Input to DH+	Output when Transmitting, Input when Receiving.
CLK	Input to DH+	Clock Input
CS	Input to DH+	Chip Select (normally Active Low)

Use the **SENDS** command to load characters to be sent to the master when they are next requested. The system buffers the characters until they are clocked out by the master.

Use **RCVS** to retrieve characters sent to the Data Hog Plus from the master. It will wait until characters are received before returning.

SPI Specific Commands

- SPI - Enables or disables the SPI port and sets its type and speed.
- SPICS - Turns on or off the SPI Chip Select line when in Master mode.
- SENDS <text> - Sends out a specific string.
- RCVS <text> - Reads in a specific string or set of values from SPI port.

System Command – SPI**Format: SPI****SPI=CLEAR / TXCLEAR / RXCLEAR****SPI=<param>**

This command enables or disables the SPI port and sets its speed and mode. Note that enabling the SPI port automatically disables the I2C port (and vice-a-versa). Send:

- SPI - By itself to see the current settings
- SPI=CLEAR - Clears any pending receives and transmits.
- SPI=RXCLEAR - Clears any pending receives.
- SPI=TXCLEAR - Clears any pending transmits.

Using **SPI=<param>** has the following values:

<i>Parameter</i>	<i>Values</i>	<i>Explanation</i>
<enable>	Enable/1, Disable/0	Turns on or off the SPI port.
<mode>	Master or Slave	Sets the SPI port to Master or Slave mode.
<clock>	0 to 7	Sets the SPI clock speed (only applies to Master mode) to one of the following values: 0 – 24Mhz 1 – 12Mhz 2 – 6Mhz 3 – 3Mhz 4 – 1.5Mhz 5 – 750Khz 6 – 375Khz 7 – 187.5Khz
<cpol>	1 or 0	Clock Polarity (1 = High, 0 = Low)
<cpha>	1 or 0	Clock Phase (1 = 2 edge, 0 = 1 edge)
<bitorder>	MSB or LSB	Bit order of either Most Significant Bit first (MSB) or Least Significant Bit First (LSB)
<cs mode>	NSS or MAN	Sets the Chip Select Mode in Master mode: NSS - Automatic (toggles with each byte) MAN - Changes only on “SPICS” command or through the special \0 and \1 parameters in SENDS. Sets the Chip Select Mode in Slave mode: NSS - Automatic (toggles with each byte) MAN - Changes only on “SPICS” command or through the special \0 and \1 parameters in SENDS.
<duplex>	Full or Half	Selects either Full Duplex (4 Line SPI) or Half Duplex (3 Line SPI)

System Command – SPICS**Format: SPICS=1 or 0**

Sets the SPI CS (Chip Select) line High or Low. This is used when in Master mode and the CS is set to “MAN” (Manual).

In “Slave” mode, sending just **SPICS** will return the current Chip Select line state.

You can also check and change the CS line during **RCV** or **SEND** using the %c, \0, and \1 command codes embedded in strings.

7.2 – I2C Port

The DH+ can also implement an I2C port. This uses some of the same pins as the SPI port as described below:

- SDA** (labeled **MISO**) - Serial Data (bi-directional)
- SCK** (labeled **CLK**) - Serial Clock (input when Slave, output when Master)

The pins labeled **MOSI** and **CS** are not used in I2C mode.

The I2C port can be configured as either a “slave” or “master” which affects how the above lines function. Under most circumstances the Data Hog Plus operates as the “Master” device and controls the clock to a I2C “Slave”.

Use the **SENDI** command to load characters to be sent to the slave or master and use **RCVI** to retrieve characters sent to the Data Hog Plus from a slave or master.

I2C Specific Commands

- I2C - Enables or disables the I2C port and sets its type and speed.
- SENDI <text> - Sends out a specific string.
- RCVI <text> - Reads in a specific string or set of values from I2C port.

System Command – I2C

Format: I2C

I2C=CLEAR / TXCLEAR / RXCLEAR

I2C=<param>

This command enables or disables the I2C port and sets its speed and mode. Note that enabling the I2C port automatically disables the SPI port (and vice-a-versa). Send:

- I2C - By itself to see the current settings
- I2C=CLEAR - Clears any pending receives and transmits.
- I2C=RXCLEAR - Clears any pending receives.
- I2C=TXCLEAR - Clears any pending transmits.

Using **I2C=<param>** has the following values:

<i>Parameter</i>	<i>Values</i>	<i>Explanation</i>
<enable>	Enable/1, Disable/0	Turns on or off the I2C port.
<mode>	Master or Slave	Sets the I2C port to Master or Slave mode.
<clock>	0, 1, 2	Sets the SPI clock speed to: 0 – Standard (100Khz) 1 – Fast (400Khz) 2 – Fast Plus (1Mhz)
<address>	Slave Address	Sets the address of the slave to communicate with or the Data Hog Plus's Slave Address (when in “Slave” mode)
<add length>	7 or 10	Sets the slave address length to 7bit or 10bit
<analog filter>	1 or 0	Turns on or off the Analog Filter.

8 – Converting/Adapting, Logging, & Protocol Analyzing

One of the most common uses of the Data Hog Plus is to convert from one type of serial channel to another. For example, convert from USB to RS232 or convert from USB to logic level serial. The following sections give complete details on how to enable this functionality.

Note that configuring the Data Hog Plus to act as a converter is done by either using the SD Card and a batch file (see section 4) or by sending it commands through the USB, RS232, or logic level serial channel (see section 3 and section 6). Before attempting to do this, it is recommended all users learn to do the following first:

- 1) Connect the Data Hog Plus to your computer using the USB or RS232 port.
- 2) Power it on.
- 3) Install and run the “D-Terminal” program (or use any other terminal application).
- 4) Select the port the DH+ is enumerated as (if using USB) or the serial port it is connected to (if using the RS232).
- 5) Set the baud rate to 19200bps.
- 6) Send a CTRL+E to the Data Hog Plus to get a response that starts with “DHP#...”

You are now in Command Mode and can configure the DH+. It is highly recommended you familiarize yourself with the process of using the terminal command interface before setting the DH+ up as a converter, adapter, data logger, and/or protocol analyzer.

8.1 – Converting/Adapting

The DHP+ supports several methods of converting one type of serial data to another. For example, converting the USB data to RS232 or logic level serial. The simplest method is to simply use the CONNECT command to tie two or more channels together like this:

```
CONNECT=COMU, COM1
```

From this point on, anything received on the USB channel will be sent to the RS232 port and vice-a-versa. This connection will stay active even if power is cycled, SD card is removed, or the cables are unplugged and reconnected. The DH+ will not respond to any further commands on these channels until the connection is broken by:

- 1) The “escape sequence” is received from any channel. This is normally three special characters sent within one second with a one second break before and after. The DH+ will stay in command mode after this until it receives another CONNECT command, a Ctrl+A is received, or the connection is disabled.
- 2) Power is cycled with a SD card inserted that has a AUTORUN.BAT file in the root directory containing a command which turns off or changes the connection like this:

```
CONNECT=NONE
```

- 3) CONNECT=NONE is sent from a channel not part of the connection. For example, if COM1 and the USB are connected, sending CONNECT=NONE from the Logic Level serial will break the connection for all channels.

The CONNECT command also supports multiple connections like this:

```
CONNECT=COMU, COM1, COM2
```

All three main channels are now connected together. Anything received from any channel is sent out to both of the other channels.

System Command – Connect**Format: CONNECT=NONE / OFF / <chan>
CONNECT=LOG / LOGHTML / PLAINLOG
CONNECT=LOGx / LOGHTMLx
CONNECT=NOLOG
CONNECT='c' / 0xZZ**

The **CONNECT** command ties two or three channels together. For example:

CONNECT=COM1 , COMU - Ties the USB and RS232 ports together.
CONNECT=COM1 , COM2 - Ties the RS232 and TTL Logic Level Serial together.
CONNECT=COM1 , COM2 , COMU - Ties all three channels together.

When a channel is connected, anything received on it is automatically sent to the other channels. In addition, if **AUTOBAUD** is enabled, a baud rate change on one channel will automatically change it on the other channels.

A protocol analyzing log may also be enabled through the following commands:

CONNECT=LOG - Turns on the LOG file. This will be a file named:
"yyyy-mm-dd hhmmsss.txt"
And placed in the default DATA sub-directory.

CONNECT=LOGHTML - Same as above except the format is in HTML with different colors for each channel.

CONNECT=NOLOG - Turns off LOG file creation.

Once **CONNECT** is active no further commands are accepted from either channel until either (a) an SD card is inserted with a batch file that disables the connection or (b) the DH+ receives three escape characters within two seconds with at least a second on either side. By default, the escape character is as follows (differs for each port):

USB (COMU) - Plus ("+")
RS232 (COM1) - Minus ("-")
Logic Level Serial (COM2) - Ampersand ("&")

A different character is used for each port to prevent echoing of characters from tripping the escape sequence from the wrong port.

CONNECT=x, 'c' - Set "c" to the character you want to use as the escape character. "x" is set to "U", "1", or "2" for the port's break character you want to change.

CONNECT=x, 0xZZ - Set "ZZ" to the two digit hexadecimal value you want to use as the escape character. For example, CONNECT=U,0x42 would set the USB break character to a 'B'.

When the escape sequence is received, the DH+ will send:

```
(escape ... Ctrl+A to resume)
```

Send a Ctrl+E to log onto Data Hog Plus. Send a Ctrl+A to exit the escape and return to the CONNECT with this message:

```
(connect resumed)
```

BAR GRAPH DURING CONNECT: While the connect mode is active, the bar graph will display activity on each channel using the last four LED's. LED 7 will always be off, LED 8 blinks when COM1 receives data, LED 9 blinks when COM2 receives data, and LED 10 blinks when COMU (USB) receives data.

8.2 – Data Logging/Protocol Analyzing during Connect

The Data Hog Plus has a complete data logging and protocol analyzer system built into the converter. When you use the CONNECT command to tie two or more channels together, you can also enable the log file in one of two formats:

Text Format (“LOG”) :

This is the default format. The DH+ creates a file based on the current data and time and stores a record of everything received from any port.

HTML Format (“LOGHTML”) :

This is a more advanced version of the log and is easier to read by most humans. However, the HTML file is harder to look at in a standard text editor and some users may not want that.

For example, if you wanted to connect the USB port to the RS232 port and make an HTML log of all communication, send the command:

```
CONNECT=COMU, COM1, LOGHTML
```

From this point on everything received on the USB is sent to the RS232 and everything received on the RS232 is sent to the USB port. In addition, an HTML log file is created containing a complete record of everything that happens. The HTML log file looks something like this:

```
19:08:46.234 USB : [0D]
19:08:46.234 COM1: [0D][0A]01000100F0C1800000000000000000000000000000Test
19:08:46.262 COM1: 020400AF0A010C1814151B0C18141702[0D][0A]
19:08:47.859 USB : B
19:08:47.863 COM1: B
19:08:47.883 USB : 0
19:08:47.883 COM1: 0
19:08:47.902 USB : 1
19:08:47.902 COM1: 1
19:08:47.922 USB : [0D]
19:08:47.930 COM1: [0D][0A]00AF0A[0D][0A]
19:08:48.223 USB : [15]
19:08:48.344 COM1: [02][00][01][FF][FE][01]Test[00] [00] [00]
19:08:48.359 COM1: ) [98][15][1B])[98][17][02]D[04][04][1F][02][01]
19:08:48.375 COM1: [01][00][00][00][00][00][00][00][00][00][00][00][00][00]
19:08:48.383 COM1: [00][00][00][00] [00][00][02][10]DEFAULTX.SPD[0D]DEFAU
19:08:48.398 COM1: LTC.AXL[0D]DEFAULTX.LEN[08]DEFAULTX.GAP[08]DEFAULTX.HED[10]
19:08:48.422 COM1: [00] [F8][01][00][E8][01][B7][04][00][00] [00]
19:08:48.438 COM1: [F8][01][00][E8][01][B7][04][00][00]0[00]
19:08:48.449 COM1: [F8][01][00][E8][01][B7][04][00][00]@[00]
19:08:48.465 COM1: [F8][01][00][E8][01][B7][04][00][00][F1][01][00]
19:08:48.477 COM1: [91][00][00][00][00][00][00][00][00][00][00][00][00][00]
```

The text version is similar but lacks the colors. The text log will look something like this:

```
19:00:25.473 USB : [0D]
19:00:25.473 COM1: [0D][0A]01000100F0C1800000000000000000000000000000Test
19:00:25.500 COM1: 020400AF0A010C1814151B0C18141702[0D][0A]
19:00:27.016 USB : B
19:00:27.016 COM1: B
19:00:27.035 USB : 0
19:00:27.039 COM1: 0
19:00:27.059 USB : 1
19:00:27.059 COM1: 1
19:00:27.078 USB : [0D]
19:00:27.078 COM1: [0D][0A]00AF0A[0D][0A]
19:00:27.375 USB : [15]
19:00:27.500 COM1: [02][00][01][FF][FE][01]Test[00] [00] [00]
19:00:27.516 COM1: ) [98][15][1B])[98][17][02]D[04][04][1F][02][01]
19:00:27.531 COM1: [01][00][00][00][00][00][00][00][00][00][00][00][00][00]
19:00:27.539 COM1: [00][00][00][00] [00][00][02][10]DEFAULTX.SPD[0D]DEFAU
19:00:27.555 COM1: LTC.AXL[0D]DEFAULTX.LEN[08]DEFAULTX.GAP[08]DEFAULTX.HED[10]
19:00:27.578 COM1: [00] [F8][01][00][E8][01][B7][04][00][00] [00]
19:00:27.590 COM1: [F8][01][00][E8][01][B7][04][00][00]0[00]
```

Note that the log file is buffered to some degree. This means that if you pull the SD card out to inspect the log file very quickly not everything in the log may be written to it. To combat this, the log file on the SD card is forced to sync up every 5 seconds. Therefore, make sure you wait 5 seconds after capturing your desired monitor data before pulling the card to ensure that it has been updated with the data you want.

It should also be noted that the DH+ stays in the **CONNECT** mode even if power is cycled or the SD card is removed. You can even re-insert the SD card after removing it without needing to cycle power to start a new log file.

If you want to stop the connection and logging data, you could do the following:

- 1) Open a terminal window connected to the USB COM port and at the baud rate desired.
- 2) Wait at least one second before sending any data.
- 3) Send three plus characters quickly (+++).
- 4) Wait two seconds.
- 5) The Data Hog Plus will respond with “(escape ... Ctrl+A to resume)”.
- 6) Send the command: `CONNECT=NONE`
- 7) The connection is now broken and the Data Hog Plus will revert to its normal non-adapting mode.

8.3 – Data Capture (logging)

The Data Hog Plus now has two additional easy-to-use data capture system called “Fast Logging” and “Plain Logging”. In these modes, you can easily capture data from a port and store it in a file without any additional commands or other batch file work needed.

Plain Logging

To enable plain logging, simply specify one port on the **CONNECT** command and add the parameter “PLAINLOG”. For example, to log everything received on COM1 (RS232 port) to a file send the command:

```
CONNECT=COM1,PLAINLOG
```

From this point on everything received on the RS232 port will be saved in a file named:

```
yyyy-mm-dd hhmmsss.log
```

To disable the logging, simply send the command:

```
CONNECT=NONE
```

Note that depending on the baud rate and the amount of data being received, there can be up to a 5 second delay between receiving the data and it being saved on the SD card. It is recommended you wait at least 5 seconds between capturing the data and powering the Data Hog off or pulling the SD card to be sure it is saved.

Re-inserting an SD card after removal will automatically start a new log file. Users are free to pull the SD card and re-insert it as often as desired.

Fast & Trigger Logging

The **FASTLOG** command is similar to the **STORE** command with some important changes. When it is run, the Data Hog immediately stores everything received from a channel in file. It stays in this mode until it is shut off or the SD Card is removed. **FASTLOG** is intended for those applications where the user only wants the Data Hog to store data from a single channel and nothing else.

TRIGLOG is the same as **FASTLOG**, except it will automatically stop if ANY trigger activates. See the **TRIGGER** section for more information on how to setup GPIO triggers. Note that the manual gives most examples using **FASTLOG** so keep in mind that **TRIGLOG** would work in all those situations as well.

Enabling **FASTLOG** turns off any **CONNECT** functions and the system no longer responds to **TRIGGER**'s or most other functions. If the SD card is removed, the system stops logging and returns to the normal command mode. For example, if you had a AUTORUN.BAT file with these commands:

```
BOOTBAUD1=230400  
FASTLOG1
```

A log file would be created based on the current date/time which contains everything received by the Data Hog on the RS232 port at 230400. You can also specify the file name like this:

```
FASTLOG1 MyFile.bin
```

FASTLOG always appends data onto the end of an existing file, so if you specify the same file name it will add new log data. You can use the **DEL** command to erase the file first if that is desired. For example, suppose you wanted to make a new file each time:

```
BOOTBAUD1=230400  
DEL MyFile.bin  
FASTLOG1 MyFile.bin
```

FASTLOG should be used with caution since most other functions are stopped during operation. However, it is ideal in situations where high speed non-stop data must be captured. It flushes all data to SD card at least once per second – so be sure to wait at least a second before pulling the card after data is sent.

While **FASTLOG** is active, the second LED will blink On/Off to indicate data is being written to the SD Card. This is a handy way to verify data storage is happening.

It is recommended if you plan to pull the SD card out while logging is happening that you disable commands to the port being monitored with the **COMx=DISABLE** command. This is to prevent data coming in from being processed as commands when the SD card is out.

For example, suppose you want to log data from the RS232 port at 230400 baud whenever the unit is turned on. You want that logging to also work if the SD card is removed and re-inserted. In this case, put the following AUTORUN.BAT file on each SD card you are using for monitoring:

```
COM1=DISABLE
BOOTBAUD1=230400
FASTLOG1
```

This turns off commands for COM1 (the RS232 port), sets its baud to 230400, creates a file based on the current date and time, and then starts the **FASTLOG** on the COM1 port.

NOTE: If this is the first time the batch file runs and the baud rate on the channel on power up is different than the one being set with BOOTBAUD, you can get garbage at the beginning of the log file because the buffer has captured some data before the baud rate changed. This will only happen the first time the batch file is used on this particular Data Hog.

TRIGLOG is especially handy to start a new file whenever a GPIO pin changes. Consider this example:

```
COM1=DISABLE
BOOTBAUD1=230400
TRIGGER=6, PULLUP, FALLING
:LOOP
TRIGLOG1
GOTO LOOP
```

In this example, the log file is created for COM1 at 230400 and data is stored there. However, anytime GPIO #6 is shorted to ground the log stops and then immediately restarts again (the GOTO LOOP causes this). This allows you to restart the log anytime you wish!

ULTRA HIGH-SPEED LOGGING:

There is a special case for baud rates above 1Mbps. Because so much data is being received and written to the SD Card the Data Hog Plus cannot process all activities it normally does. In this case, both **FASTLOG** and **TRIGLOG** do the following if they are started from a baud rate above 1Mbps:

- 1) Turn off any LED updates (the LED's will freeze in place).
- 2) Enter a very high-speed mode that simply stores data received onto the SD card.

Pulling the SD Card out or, in the case of TRIGLOG, activating a GPIO TRIGGER will exit the mode and the LED's will begin operating again.

MAXIMUM FILE SIZE:

To help prevent enormous file sizes and to support the FAT32 file system, both **FASTLOG** and **TRIGLOG** support the setting in **MAXFILESIZE**. This command sets the maximum size of a log file before it is automatically closed and new file reopened.

By default, this is set to 4GB (4,000,000,000). You can set it to any value you wish, but be careful in making it extremely large because some file systems have trouble with files bigger than a certain point.

For example:

```
MAXFILESIZE=1000000
```

Would set the maximum single **FASTLOG** or **TRIGLOG** file size to 1MB.

9 – GPIO (General Purpose Input/Output)

The Data Hog Plus supports up to six GPIO (General Purpose Input/Output) pins. These pins share the SPI/I2C and Logic Level Serial port pins so cannot be used at the same time if these other pins are enabled. GPIO is labeled 1 to 6 as follows:

Function	Variable Name	SEND/OUTPUT Control	Shares Pin With	ARM CPU Pin
GPIO #1	%G1	\G1x	RXD	PC0
GPIO #2	%G2	\G2x	TXD	PC1
GPIO #3	%G3	\G3x	MOSI	PB15
GPIO #4	%G4	\G4x	MISO	PB14
GPIO #5	%G5	\G5x	CLK	PB13
GPIO #6	%G6	\G6x	CS	PB12

Turning on GPIO functionality on any pin will disable the logic level serial (GPIO #1 or GPIO #2), the SPI (GPIO #3-#6), or the I2C (GPIO #3-#5). Similarly, enabling the logic level serial, SPI, or I2C functions automatically turns off the GPIO (if enabled).

To re-enable the logic level serial after using one or both pins for GPIO, send the command “BAUD1=<baud>”, “GPIO=NONE”, “GPIO=G1OFF,G2OFF”, or “RESET”. Any of these will re-enable the logic level serial.

Using the GPIO is done in a variety of ways. The primary way is to use the **GPIO** command (described next) to configure and set the GPIO pin values. You do not usually have to preset a pin to be an output or an input, simply indicate you want to either read the value of the pin or set it to a specific value like this:

```
GPIO=G4OUT1 - Sets GPIO #4 to an output and sets its value to 1 (high, or +3.3V).
```

In addition, you can read the current GPIO value or change the output value with the “%Gx” variable or the “\Gxy” code. In this case, you must preconfigure the pins as inputs or outputs or the value will be meaningless. For example:

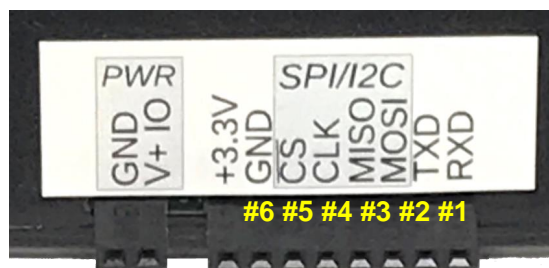
```
GPIO=G1IN,PULLUP
SENDU "GPIO #1 = %G1!\r\n"
```

This sends to the USB port the current GPIO #1 input which is being pulled high. To change the value of an output pin, simply send it like this:

```
GPIO=G1OUT0
SENDU "\G11___BUGS___\G10"
```

This starts GPIO #1 output at a 0. Then it sets it to a 1, sends “___BUGS___” out the USB port, and sets it to 0 again. If you were to measure the duration GPIO #1 was high, this would tell you how long the Data Hog Plus took to load the transmit buffer with “___BUGS___”.

The GPIO pins are numbered like this:



Refer to the ST Microelectronics # STM32L451 data sheet for more specific information on the electrical characteristics of the GPIO pins.

System Command – GPIO**Format: GPIO=NONE / OFF****GPIO=Gx****GPIO=GxOFF****GPIO=GxIN / GxIN,PULLUP/PULLDOWN****GPIO=GxOUTy / GxOy**

The **GPIO** command configures the GPIO pins. To turn the GPIO pins off and revert to normal mode:

GPIO=OFF - Turns off all GPIO configurations.
GPIO=G1OFF - Turns off GPIO #1 configuration (can range from 1 to 6).
GPIO=G4 - Returns the current status of GPIO #4.

To set a GPIO pin as an output, send the following command:

GPIO=G1OUT0 - Sets GPIO #1 as an output and sets it to a 0 (Ground).
GPIO=G100 - Same as above.
GPIO=G601 - Sets GPIO #6 as an output and sets it to a 1 (+3.3V).

To set a GPIO pin as an input, send the following command:

GPIO=G1IN - Sets GPIO #1 to an input and returns its current value (1 or 0). No pull resistor is enabled.
GPIO=G3IN, PULLDOWN - Sets GPIO #3 to a pulled down input and returns its current value (1 or 0).
GPIO=G5IN, PULLUP - Sets GPIO #5 to a pulled up input and returns its current value (1 or 0).

You can also use GPIO pin values in the **SEND**, **RCV**, **OUTPUT**, and **IF** commands using the variable “%Gx” where “x” is from 1 to 6 and the “\Gxy” where “x” is from 1 to 6 and “y” is a 0 or 1. For example:

```
IF %G1 = 0 GOTO MyLabel
```

This would jump to “MyLabel” if GPIO #1 had a current value of zero. To change GPIO output values during a send, you can use:

```
SEND1 "Hello\G11Goodbyte\G10"
```

This sends “Hello” out the RS232 port, sets GPIO #1 to a “1”, sends “Goodbye” out the RS232 port, and then sets GPIO #1 to a “0”.

9.1 – GPIO Inputs

Each GPIO can be configured as an Input or Output. When configured as an Input, the GPIO has the following characteristics:

- Can be left floating or can optionally be pulled High (to +3.3V) or Low (to Ground).
- Pull up or down resistance is about 40K ohms.
- A “low” (or “0”) is around 1V or less and a “high” (or “1”) is around 2V or more.

For example, suppose you wanted a batch file that output the time whenever GPIO #6 was pulled low. A batch file like this may work (will exit if it receives a character on serial port):

```
GPIO=G6IN,PULLHIGH
; Loop until GPIO #6 goes low
:LOOP1
IF %r=1 GOTO DONE
IF %G6=1 GOTO LOOP1
; GPIO #6 went low!
SEND "Low at %H:%N:%S\r\n"
; Now loop until GPIO #6 goes high again
:LOOP2
IF %G6=0 GOTO LOOP2
GOTO LOOP1
:DONE
```

You can also use the “%Gx” variables as part of your SEND output. For example, suppose every second you wanted to output the value of GPIO #1 configured as pulled low input:

```
GPIO=G1IN,PULLLOW
:LOOP1
; Output current value
SEND "GPIO #1 = %G1\r\n"
; Wait a second
IDLE on,1
IF %r=0 GOTO LOOP1
```

This exits if it receives a character on the serial port, otherwise it outputs the current GPIO input value once per second.

WARNING! WARNING! WARNING!

DO NOT apply more than 5V into any of the GPIO pins! You will likely permanently destroy the pin and possibly the entire Data Hog Plus if you do so. The pins have some protection, but it is possible to ruin a Data Hog Plus by applying too high a voltage to any pin! Diamond Edge Technology cannot repair or replace units that have failed due to this reason and you assume all risk when using the GPIO input feature.

9.2 – GPIO Outputs

Each GPIO can be configured as an Input or Output. When configured as an Output, the GPIO has the following characteristics:

- Can be set to High (“1” or +3.3V) or Low (“0” or Ground/0V).
- Can source or sink +/- 8mA per pin.
- DO NOT TRY TO DIRECTLY DRIVE HIGH CURRENT DEVICES FROM THE GPIO! You will likely destroy the pin and/or the entire Data Hog Plus doing so.

The GPIO output function can be very useful in a multitude of situations. However, you must be careful in using this feature as it possible to destroy the Data Hog Plus. Diamond Edge Technology cannot repair or replace units that have failed due to incorrect usage of the GPIO Input or Output function. Please contact us at ryan@detllc.com if you are unsure in advance.

One good use of the GPIO output function is to alert other devices that something has happened. For example, suppose you want to pulse GPIO #1 once per second. A batch file like this would work:

```
:LOOP1
; Wait a second
IDLE on,1
; GPIO #1 High
GPIO=G1OUT1
; GPIO #1 Low
GPIO=G1OUT0
IF %r=0 GOTO LOOP1
```

This will make a short high pulse on GPIO #1 (about 50ms). You can also change the output state of any GPIO pin inside of an output string. For example, if you wanted to make GPIO #1 high during the send command you could have a batch file like this:

```
GPIO=G1OUT0
:LOOP1
; Wait a second
IDLE on,1
; Make GPIO #1 High during send
SEND "\G11__BUGS BUNNY__\r\n\G10"
; Repeat until we receive a character
IF %r=0 GOTO LOOP1
```

This pulse is much shorter, about 90 microseconds, because it does not take long to write to the internal que of things to send out the serial port.

9.3 – EXTERNAL TRIGGER System

In addition to the other GPIO functions, you can use any of the six pins in “EXTERNAL TRIGGER” mode. The external trigger is very similar to the GPIO Input ability, except when the line changes state it triggers an interrupt inside the DH+ which (a) can wake it from **SLEEP** or **DEEPSLEEP**, (b) stores the exact date & time of the event in a special register (%X1 to %X6 for “yyyy-mm-dd@hh:mm:ss” or %x1 to %x6 for “unixepochtime.msec”), and (c) sets another special register to “1” when the state activates the interrupt.

The external trigger pins share the SPI/I2C and Logic Level Serial port pins so cannot be used at the same time if these other pins are enabled. Trigger #1 to #6 is as follows:

Function	Variable Name (for current state)	Variable Name (for last activation time/date)	Variable Name (for trigger status)	Shares Pin With
TRIGGER #1	%G1	%X1 and %x1	%K1	RXD
TRIGGER #2	%G2	%X2 and %x2	%K2	TXD
TRIGGER #3	%G3	%X3 and %x3	%K3	MOSI
TRIGGER #4	%G4	%X4 and %x4	%K4	MISO
TRIGGER #5	%G5	%X5 and %x5	%K5	CLK
TRIGGER #6	%G6	%X6 and %x6	%K6	CS

Turning on the EXTERNAL TRIGGER for any pin will disable the logic level serial (TRIGGER #1 or TRIGGER #2), the SPI (TRIGGER #3-#6), or the I2C (TRIGGER #3-#5). Similarly, enabling the logic level serial, SPI, or I2C functions automatically turns off the TRIGGER (if enabled).

To re-enable the logic level serial after using one or both pins for a TRIGGER, send the command “BAUD1=<baud>”, “TRIGGER=NONE”, or “RESET”. Any of these will re-enable the logic level serial.

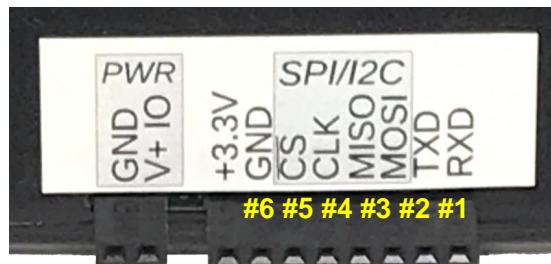
Using the EXTERNAL TRIGGER is done in a variety of ways. The primary way is to use the **TRIGGER** command (described next) to configure and set the TRIGGER pin values like this:

- TRIGGER=1, FALLING, PULLUP - Enables Trigger #1, pulls it high (to 3.3V), and sets the trip to be the falling edge (pin going from High-3.3V to Low-Ground).
- TRIGGER=4, EITHER, NOPULL - Enables Trigger #4, does not pull it high or low, and sets the trip to be either low to high or high to low.

You can read the current input value with the “%Gx” variable. For example:

```
TRIGGER=1, FALLING, PULLUP
SENDU "TRIGGER #1 = %G1!\r\n"
```

This sends out the USB port the current Trigger #1 value (which is being pulled high). The TRIGGER pins are numbered the same as the GPIO pins like this:



Command – TRIGGER **Format: TRIGGER=NONE / OFF**
TRIGGER=<#>,<mode>,<pull>,<tim>,<wake>,<store>
TRIGGER=<num>,OFF
TRIGGER=RESET

The **TRIGGER** command configures the EXTERNAL TRIGGER functions. To turn the TRIGGER pins off and revert to normal mode:

- TRIGGER=OFF - Turns off all TRIGGER configurations.
- TRIGGER=1, OFF - Turns off TRIGGER #1 configuration (can range from 1 to 6).

To show the current settings of an EXTERNAL TRIGGER, send a command like this:

- TRIGGER - Returns the TRIGGER status of all pins.
- TRIGGER=4 - Returns the current status of TRIGGER #4.

To set an EXTERNAL TRIGGER, send a command like this:

- TRIGGER=1, FALLING, PULLUP - Sets TRIGGER #1 to pull-up and trigger when it falls to ground.
- TRIGGER=4, RISING, PULLDOWN - Sets TRIGGER #4 to a pulled down and triggers when it rises to 3.3V.
- TRIGGER=1, BOTH, NOPULL - Sets TRIGGER #1 to trip on either low->high or high->low and sets it to have no pull resistor.
- TRIGGER=1, BOTH, NOPULL, NOWAKE - Sets TRIGGER #1 to trip on either low->high or high->low and sets it to have no pull resistor. It also will NOT wakeup the unit from **SLEEP, IDLE, or DEEPSLEEP**.

To reset the trigger files (if saving to a Text, CSV, or Timestamp file), send the command:

- TRIGGER=RESET - Resets all triggers and opens new files (if any).

Each TRIGGER is followed by a series of parameters as shown above. Only one trigger at a time can be set per command (unlike the GPIO function which allows you to set multiple pins with one command).

Each parameter is described below:

Parameter <#>:

1 to 6	Specifies the EXTERNAL TRIGGER number to enable, disable, or return the status of
--------	---

Parameter <mode>:

FALLING	Sets the TRIGGER to activate when the pin goes from a “High” state (logic level 1 or 3.3V) to “Low” state (logic level 0, zero volts, or Ground).
RISING	Sets the TRIGGER to activate when the pin goes from a “Low” state (logic level 0, zero volts, or Ground) to a “High” state (logic level 1 or 3.3V).
BOTH	TRIGGER will activate when it is either RISING or FALLING (any change of state).

Parameter <pull>:

NOPULL	Does not apply any pull on the pin. You must connect it to something that is either 3.3V or 0V for the system to work properly.
PULLUP	Pulls the pin to 3.3V through a 40K resistor.
PULLDOWN	Pulls the pin to 0V (ground) through a 40K resistor.

Parameter <tim>:

<val>sec	Set's the maximum trigger rate, or timing. This value specifies a debounce value to keep the trigger from tripping multiple times due to a noisy signal. For example: TRIGGER=1, FALLING, PULLUP, 0.5sec This would set the maximum frequency for TRIGGER 1 to once every 0.5 seconds (2 hertz). The maximum resolution is milliseconds.
(blank)	If this parameter is not specified, all EXTERNAL TRIGGER's default to a timing of 0.1sec (100 milliseconds).

Parameter <wake>:

NOWAKE	Disables waking the DH+ from SLEEP , IDLE , or DEEPSLEEP when the pin activates.
WAKE (or blank)	Enables waking the DH+ from SLEEP , IDLE , or DEEPSLEEP when the pin activates. This is the default value if nothing is specified.

Parameter <store>:

(blank)	If nothing is entered for this parameter, no automatic storage of the EXTERNAL TRIGGER activations occurs. This is the default.
TEXT	Enables automatic storage of any activation in a text file automatically created in the Download Directory (defaults to the root directory). The file will be an easy-to-read description of each activation with a millisecond timestamp and exactly what happened. For example, if you sent TRIGGER=1, FALLING, PULLUP, TEXT then a typical output file might contain values like this: #1: 1->0 on 2021-04-02 @ 21:37:05.613 #1: 1->0 on 2021-04-02 @ 21:37:05.910 The “#” is the trigger, the “1->0” indicates it tripped on the pin going from HIGH to LOW, and then the following values are the date and time of the activation.
CSV	Similar to above except the format is CSV (comma separated values) which is very easy to directly open with Excel or other similar software. For example, if you sent TRIGGER=1, FALLING, PULLUP, CSV then a typical output file might contain values like this: #1,1->0,2021-04-02,21:30:31.539 #1,1->0,2021-04-02,21:30:36.773 The “#” is the trigger, the “1->0” indicates it tripped on the pin going from HIGH to LOW, and then the following values are the date and time of the activation.
TIMESTAMP	Similar to the CSV format except the time is stored as a UNIX timestamp value. This is the number of seconds since Jan 1 st , 1970, with the decimal part equaling the milliseconds value. For example, if you sent TRIGGER=1, RISING, PULLDOWN, TIMESTAMP then a typical output file might contain values like this: #1,0->1,1617399937.922 #1,0->1,1617399939.129 #1,0->1,1617399941.129
COM1, COM2, COMU	Outputs the activation to the specified COM port (always in CSV format). The example for CSV above shows what would be sent out the specified COM port when an EXTERNAL TRIGGER activates.

NOTE: You can use the **TYPE**, **TYPEH**, and any of the download commands to view a currently open TEXT, CSV, or TIMESTAMP file. When the file access is complete, data is again written to the file. This is normally a problem free situation, but if you have very frequent TRIGGER's you are better off using this command first:

```
TRIGGER=RESET
```

To force a new file to open before accessing the currently open trigger output file.

USING %K1 to %K6 VARIABLES:

The %K1 to %K6 values will return a “1” when the TRIGGER activates, otherwise they will return zero. Note that these variables are different than the %G1 to %G6 which return the current state of the pin (either a “1” or “0”). Instead, %K1 to %K6 return a “1” immediately after the TRIGGER activates by either falling or rising (depending on the way the TRIGGER is configured).

For example, suppose you wanted to output the time whenever TRIGGER #6 was pulled low. This could be done by connecting a pushbutton between ground and TRIGGER #6 and using a batch file like this (it will exit the program if it receives a character on serial port):

```
TRIGGER=6,PULLUP,FALLING
; Loop until TRIGGER #6 activates
:LOOP1
IF %r=1 GOTO DONE
IF %K6=0 GOTO LOOP1
; TRIGGER #6 activated!
SEND "Triggered at %H:%N:%S\r\n"
; Loop back and repeat
GOTO LOOP1
:DONE
SEND "DONE!\r\n"
```

IMPORTANT: Reading any %K variable will only return "1" a single time for each activation. Reading it clears it back to "0". This allows you to repeatedly check the variable and know that it only returns "1" when the trigger is activated freshly.

You can also use the "%Gx" variables with TRIGGER's and part of your SEND output. For example, suppose every second you wanted to output the value of TRIGGER #1 configured as pulled low input:

```
TRIGGER=1,RISING,PULLDOWN
:LOOP1
; Output current value
SEND "TRIGGER #1 = %G1\r\n"
; Wait a second
IDLE on,1
IF %r=0 GOTO LOOP1
```

This exits if it receives a character on the serial port, otherwise it outputs the current TRIGGER #1 pin value once per second. This is identical to using the pin as a GPIO input.

USING %X1 to %X6 and %x1 to %x6 VARIABLES:

The %X1 to %X6 and %x1 to %x6 values return the last date and time a trigger was activated. The difference between "X" and "x" is that the "X" returns it in this format:

```
yyyy-mm-dd @ hh:mm:ss.msec
```

And "x" returns it in this format:

```
<unix epoch time>.<msec>
```

For example, suppose you wanted to send out the USB port the time of each TRIGGER #2 activation. You could use the "COMU" parameter to output the value (described above), but for more custom control instead create a batch file with these lines:

```
TRIGGER=2,FALLING,PULLUP
:LOOP1
IF %r=1 GOTO DONE
IF %K2=0 GOTO LOOP1
SENDU "Trigger #2=%X2 (Epoch %x2)\r\n"
GOTO LOOP1
:DONE
SENDU "Done!\r\n"
```

EXTERNAL TRIGGER EXAMPLES:

Example 1 – Recording the winners of a toy car race:

Assume you have sloped track with four lanes. At the end of each lane there is a switch that toy cars close when they reach the end of the track. You can very accurately record the “winner” of each race by doing this:

- 1) Tie one side of each lane’s switch to ground and tie this same ground to the “GND” input pin on the DH+.
- 2) Tie the other side of the lane switch to EXTERNAL TRIGGER #1, #2, #3, and #4.

- 3) Connect to the Data Hog Plus and send these commands:

```
TRIGGER=NONE
TRIGGER=1, FALLING, PULLUP, NOWAKE, TEXT
TRIGGER=2, FALLING, PULLUP, NOWAKE, TEXT
TRIGGER=3, FALLING, PULLUP, NOWAKE, TEXT
TRIGGER=4, FALLING, PULLUP, NOWAKE, TEXT
```

From this point on, any time EXTERNAL TRIGGER #1-#4 gets connected to ground through the switch, it’s exact time will be recorded in a text file on the SD card.

- 4) Make sure an SD card is installed and run your race!
- 5) To see the results, use the TYPE command, pull out the SD card and use a computer to view the file, or use the D-Terminal app to “View” the file on the SD card. A typical race result might look like this:

```
#3: 1->0 on 2021/04/03 @ 23.09.32.426
#1: 1->0 on 2021/04/03 @ 23.09.33.313
#2: 1->0 on 2021/04/03 @ 23.09.33.809
#4: 1->0 on 2021/04/03 @ 23.09.34.398
```

- 6) Note that the DH+ will stay in this mode until you change it. You can power it off and later power it back on ready to work. You could also send the additional command after it is powered on:

```
DEEPSLEEP
```

In this case the LED’s will go off and the DH+ will continue to record the triggers for about 2 weeks from a 9V battery.

Example 2 – Recording the winners of a toy car race (2nd version):

Assume the same as Example #1 except you want to send the trigger output times live over the USB port. This is similar to sending it to a text file, except there is no file created and instead you read the values out in real time using a terminal program.

- 1) Setup the hardware exactly the same.
- 2) Connect to the Data Hog Plus and send these commands:

```
TRIGGER=NONE
TRIGGER=1, FALLING, PULLUP, NOWAKE, USB
TRIGGER=2, FALLING, PULLUP, NOWAKE, USB
TRIGGER=3, FALLING, PULLUP, NOWAKE, USB
TRIGGER=4, FALLING, PULLUP, NOWAKE, USB
```

From this point on, any time EXTERNAL TRIGGER #1-#4 gets connected to ground through the switch, it’s exact time will be sent out the USB port.

- 3) Make sure you have a terminal program opened to view the output and run your races!

Example 3 – Increase the bargraph each time a button is pressed.

In this example you want the bar graph to display increasing amount each time a button is pressed. You also want to send the level out the RS232 port at 115200bps:

- 1) Connect a pushbutton between EXTERNAL TRIGGER #1 and the GND pin.
- 2) Create a file on the SD card named "ButtonPress.bat" and fill it with this code:

```
BAUD1=115200
TRIGGER=1,FALLING,PULLUP
:LOOP0
LEDO=ALL
SEND1 "Level 0!\r\n"
:LOOP1
IF %K1 = 0 GOTO LOOP1
LEDO=1,1
SEND1 "Level 1!\r\n"
:LOOP2
IF %K1 = 0 GOTO LOOP2
LEDO=2,1
SEND1 "Level 2!\r\n"
:LOOP3
IF %K1 = 0 GOTO LOOP3
LEDO=3,1
SEND1 "Level 3!\r\n"
:LOOP4
IF %K1 = 0 GOTO LOOP4
LEDO=4,1
SEND1 "Level 4!\r\n"
:LOOP5
IF %L1 = 0 GOTO LOOP5
LEDO=5,1
SEND1 "Level 5!\r\n"
:LOOP6
IF %K1 = 0 GOTO LOOP6
LEDO=6,1
SEND1 "Level 6!\r\n"
:LOOP7
IF %K1 = 0 GOTO LOOP7
LEDO=7,1
SEND1 "Level 7!\r\n"
:LOOP8
IF %K1 = 0 GOTO LOOP8
LEDO=8,1
SEND1 "Level 8!\r\n"
:LOOP9
IF %K1 = 0 GOTO LOOP9
LEDO=9,1
SEND1 "Level 9!\r\n"
:LOOP10
IF %K1 = 0 GOTO LOOP10
LEDO=10,1
SEND1 "Level 10!\r\n"
:LOOPDONE
IF %K1 = 0 GOTO LOOPDONE
GOTO LOOP0
```

- 3) RUN "ButtonPress.bat". Each time the button is pressed the LED bargraph will increment by one! To exit, unplug the SD card, wait one second, and plug it back in!

9.4 – ADC (Temperature & Voltage Monitor) System

The Analog to Digital Converter (ADC) system is special function of GPIO Pins #1 and #2. You can also enable the current Temperature monitoring.

The ADC pins share the Logic Level Serial port pins so that cannot be used at the same time if ADC Pin #1 or Pin #2 is enabled (does not apply to ADC Temperature). ADC#1, ADC#2, and ADC#T is as follows:

Function	Variable Name (for current state)	Variable Name (for output as an integer)	Variable Name (for output as a float)	Shares Pin With
ADC #1	%G1	%V1	%v1	RXD
ADC #2	%G2	%V2	%v2	TXD
ADC #T	%G3	%VT (Celcius) %Vt (Fareheight)	%vT (Celcius) %vt (Fareheight)	(No pin used)

Turning on the ADC #1 or ADC #2 will disable the logic level serial, TRIGGER #1 or TRIGGER #2, or GPIO #1 or #2. Similarly, enabling the logic level serial, TRIGGER #1 or #2, or GPIO #1 or #2 will disable that pin for ADC. To re-enable the logic level serial after using one or both pins for ADC, send the command “BAUD1=<baud>”, “ADC=NONE”, or “RESET”. Any of these will re-enable the logic level serial.

Command – ADC

Format: ADC=NONE / OFF

ADC=<#>,DISABLE / OFF

ADC=<#>,<zero>,<calibration>,off=<val>

ADC=<#>,<zero>,r1=<ohms>,r2=<ohms>

The **ADC** command configures the ADC functions. To turn the ADC pins off and revert to normal mode:

- ADC=OFF - Turns off all ADC configurations.
- ADC=1, OFF - Turns off ADC #1 configuration (can range from 1 to 2).
- ADC=T, OFF - Turns off ADC #T (Temperature) configuration.

To show the current settings of an ADC channel, send a command like this:

- ADC - Returns the ADC status of all pins.
- ADC=1 - Returns the current status of ADC #1.

To set an ADC Channel, send a command like this:

- ADC=1, 100, 1.0 - Sets ADC #1 to on with a zero point of “100” and a calibration factor of 1.00 (see below)
- ADC=1, ON, 100, 1.0 - Same as above
- ADC=1, ON, 0, 0.000819, off=-0.25 - Sets ADC #1 to calibration of 0.00819 and offset of -0.25.
- ADC=T, ON - Turns on the temperature ADC channel.

The <zero> parameter specifies the number read back from the 12 Bit ADC channel to be considered a zero. The full range of the ADC is 0-4095, but not all inputs get all the way down to “0”. You can easily calibrate the channel to report a different reading as “0” using this parameter.

<calibration> is the adjustment applied to the ADC value when reporting it with the %V and %v variables. For example, suppose you sent the following command:

```
ADC=1, ON, 200, 2.5
```

In this setup, the ADC channel is read and a value of “1255” comes back. The DH+ subtracts 200 (the zero value) from the value read and then multiplies it by 2.5 (the calibration). So “1255” is reported as “2637” (%V1) or “2637.5” (%v1). This ability makes more sense when connected to a battery or other device (see examples below).

ADC EXAMPLE #1: Reading a voltage from 0V to 3.3V

In the simplest form, to read a voltage from zero to 3.3 send the following command:

```
ADC=1,ON,0,0.000819
```

To understand how this works, consider that the Analog to Digital converter returns a number from “0” to “4095” based on the voltage on the pin. Zero represents no voltage and “4095” represents 3.3V. In other words, if the number “4095” was returned you can be certain that 3.3V was being applied to the pin.

In reality it is difficult to get a full “4095” reading back. Resistance and other factors tend to reduce the number slightly, so a reading of around “4030” is more typical for a full 3.3V input. To convert “4030” to 3.3, multiply the reading by “0.000819” to get “3.30”.

The following batch file will record the voltage reading once every 5 seconds saving it in a file named “Voltages.txt” and outputting it to the serial port. It stops if it receives a character on the serial port:

```
ADC=OFF
ADC=1,ON,0,0.000819
OPENA Voltages.txt
SEND "Recording voltages:\r\n"
:LOOP1
%1 = "%Y-%M-%D @ %H:%N - Current Voltage=%v1\r\n"
OUTPUT %1
SEND %1
SLEEP on,60
IF %r=0 GOTO LOOP1
SEND "\r\nDone!\r\n"
```

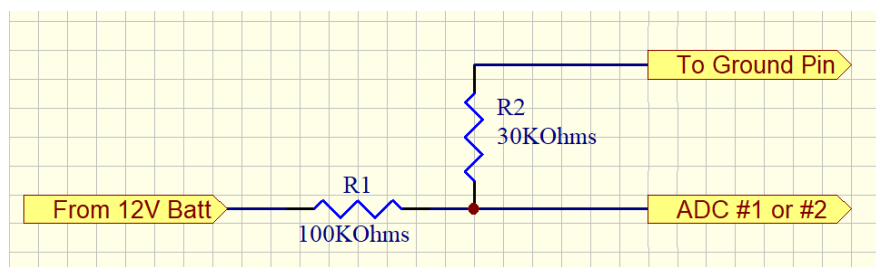
Let this run for a while, varying the voltage on pin #1 from 0V to 3.3V, and then send a character to get it to stop. When it's done, send TYPE “Voltage.txt” and you would see something like this:

```
2021-04-05 @ 19:17 - Current Voltage=2.35
2021-04-05 @ 19:18 - Current Voltage=2.85
2021-04-05 @ 19:18 - Current Voltage=3.35
2021-04-05 @ 19:18 - Current Voltage=0.34
2021-04-05 @ 19:18 - Current Voltage=1.96
2021-04-05 @ 19:18 - Current Voltage=1.65
```

ADC EXAMPLE #2: Reading a voltage from a 12V battery (0V to 14.5V range)

Reading a voltage HIGHER than 3.3V is also possible with the DH+, but it does require some extra electronics. You must NEVER apply a voltage higher than 3.3V to the pin, but you can easily adjust for this using a “resistor divider” network.

A resistor divider is two resistors. You connect one of them to the input voltage (the 12V battery in this case), and the other one to this resistor and ground. The resistor connected to the battery is named “R1” and the resistor from R1 to ground is R2. Here is a schematic of the circuit:



As is shown in the schematic, R1 is labeled as a 100K resistor and R2 is labeled as a 30K resistor. These are two commonly available resistors and by running the 12V battery through this circuit, it converts around 14.5V down to 3.3V so that the voltage input is never exceeded.

To determine the right values, it is handy to use a “Voltage Divider Calculator”. There are many free ones available such as the one at: <https://ohmslawcalculator.com/voltage-divider-calculator>

Simply enter in the R1 value (100K is a good place to start), the maximum input voltage (14.5 Volts in this case), and the desired maximum output voltage (3.3V) and click “Calculate”. This looks something like:

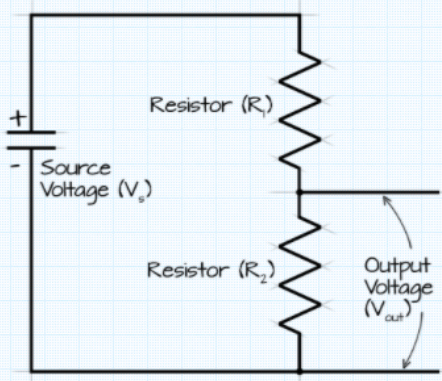
Voltage Divider Calculator

A voltage divider circuit is a very common circuit that takes a higher voltage and converts it to a lower one by using a pair of resistors. The formula for calculating the output voltage is based on Ohms Law and is shown below.

$$V_{out} = \frac{V_s \times R_2}{(R_1 + R_2)}$$

where:

- V_s is the source voltage, measured in volts (V).
- R_1 is the resistance of the 1st resistor, measured in Ohms (Ω).
- R_2 is the resistance of the 2nd resistor, measured in Ohms (Ω).
- V_{out} is the output voltage, measured in volts (V).



Enter any three known values and press 'Calculate' to solve for the other.

Voltage Source (V_s)	<input type="text" value="14.5"/>	Volts (v)
Resistance 1 (R_1)	<input type="text" value="100000"/>	ohms (Ω)
Resistance 2 (R_2)	<input type="text" value="29464.286"/>	ohms (Ω)
Output Voltage (V_{out})	<input type="text" value="3.3"/>	Volts (v)

Click "Calculate" to update the field with orange border.

To connect this to the Data Hog Plus, it looks something like this:



Now that the circuit is in place, hook the negative side of the battery to the DH+ ground pin and the positive side to the resistor R1 that has nothing else connected to it. It is also necessary to set the ADC channel to correctly report the voltage. This is fairly easy by using the built-in calibration calculator like this:

```
ADC=1,ON,R1=100000,R2=30000
```

The DH+ will automatically calculate the calibration (in this case "0.003491"). NOTE: It is often helpful to manually measure the resistors with a multi-meter before using the above commands. A standard "100L" resistor can fluctuate quite a bit, and your accuracy will be improved if you use correct values.

Finally, sometimes errors can creep into the measurement. A final adjustment can be made with the "Offset" parameter. For example:

```
ADC=1,ON,R1=100000,R2=30000,OFF=0.25
```

This would add 0.25 to every measurement. You will need to do some tests to determine if this value is necessary depending on various factors of your setup.

To record the battery voltage as described in ADC Example #1, your new batch file would read:

```
ADC=OFF
ADC=1,ON,R1=100000,R2=30000
OPENA Voltages.txt
SEND "Recording voltages:\r\n"
:LOOP1
%1 = "%Y-%M-%D @ %H:%N - Current Voltage=%v1\r\n"
OUTPUT %1
SEND %1
SLEEP on,60
IF %r=0 GOTO LOOP1
SEND "\r\nDone!\r\n"
```

10 – Complete Example (SPI Radio Module Monitor)

Many companies make development kits and other tools for evaluating their products. Unfortunately, most of these devices are setup be used almost exclusively with a PC and a USB port (sometimes with an added wall wart power supply). Real world testing and evaluation can be difficult because PC's don't have SPI, I2C, or logic level serial ports built in and power is difficult to supply if you don't have a USB available.

The Data Hog Plus was designed to overcome these limitations and give engineers, developers, and companies an easy to use and sophisticated tool for evaluating devices with alternate communication channels. In addition, the DH+ can log details or provide a passthrough function which greatly helps debugging and development.

The following is a detailed example of how the Data Hog Plus can be used to implement a complex system. Data Hog #1 serves as a radio transmitter sending out the current time once per second. Data Hog #2 serves as the radio receiver which scans for the time and, whenever it is received, it shows the strength of the signal on the ten-segment bar graph and records the time and the signal strength in a log file on the SD card. The files for this example (along with other example files) are included on the SD card provided with the DH+ in the "SAMPLE TX-RX TRANSMITTER" sub-directory.

NOTE: This example shows using the Data Hog Plus-P version. This works the same as the -U and -W models except the -P doesn't have a USB port which does not effect this example.

Project Goals:

- Make a battery powered radio transmitter and receiver.
- Data Hog Plus #1 with a Semtech # SX1232 module will be setup as a transmitter.
- Data Hog Plus #2 with a Semtech # SX1232 module will be setup as a receiver.
- When both units are powered on:
 - The transmitter sends the current time over the radio once per second.
 - If the second radio receives it, it displays the strength of the signal on the bar graph.
 - Receiver will also store in a log file the strength of the signal and the time it received. The log file will look like something like this (Lxx = strength from 1-10):

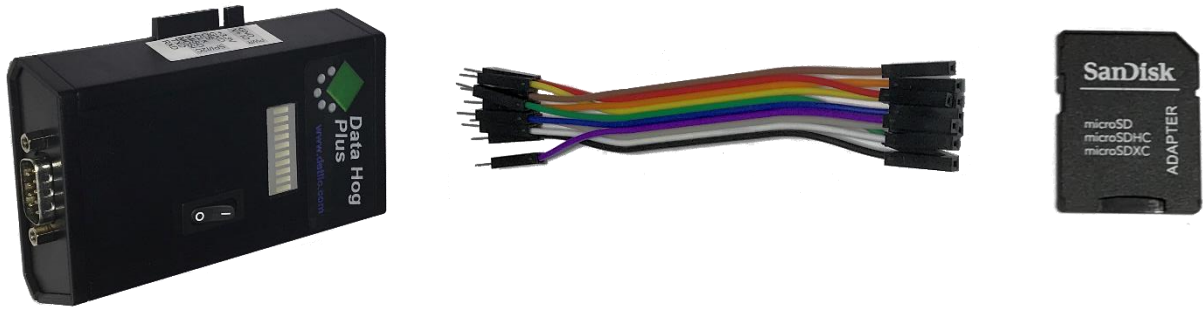
```
-----  
FILE: "2020-11-07 194813.txt"  
-----
```

```
L07-19:48:11  
L07-19:48:12  
L08-19:48:13  
L09-19:48:14  
L10-19:48:15  
L10-19:48:16  
L10-19:48:17
```

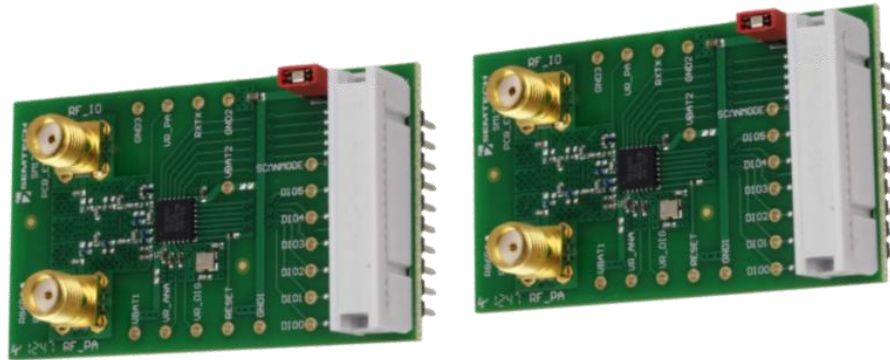
- The configuration of the transmitter and receiver can easily be controlled, changed, and monitored using the Data Hog programming system and batch file control.
- Because they are small battery powered devices, both the receiver and transmitter can be moved in and out of building and separated by distance to gage the affect on the radio signal.

Equipment Needed:

- 1) Two Data Hog Plus-P, Data Hog Plus-U, or Data Hog Plus-W units with included cables and SD card.



- 2) Semtech Corporation Evaluation Kit # SX1232-32SKA915 with two SX1232 module PCB's:

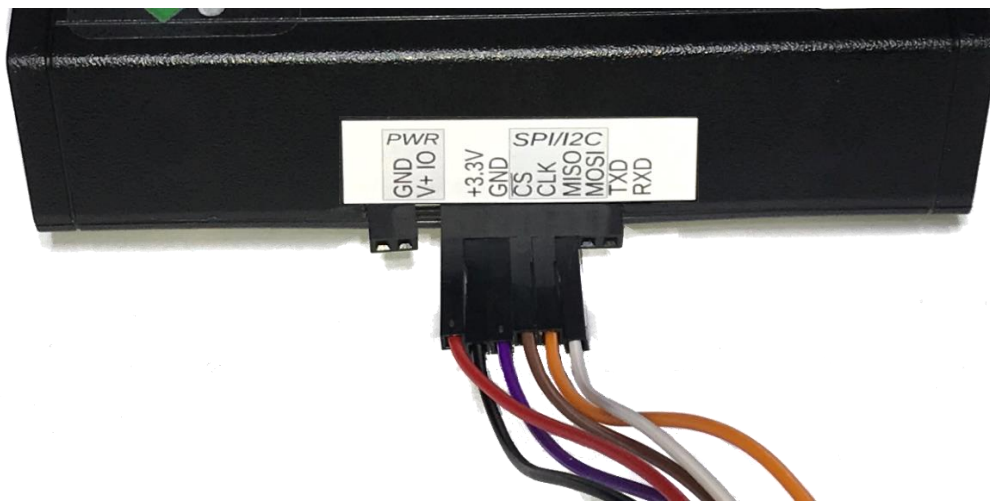


- 3) Two 9V batteries.

Setting up the Transmitter:

The Data Hog Transmitter is created following these steps:

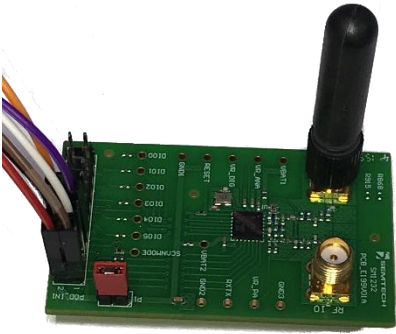
- 1) Plug in wires into the side port on the DH+ as shown here:



Wire colors are as follows:

- | | |
|-------------|---------------|
| +3.3V – Red | CLK – Brown |
| GND – Black | MISO – Orange |
| CS – Purple | MOSI – White |

- 2) Connect the other side of the wires to the SX1232 module as shown here:



- Pin #1 – Brown
- Pin #2 – Red
- Pin #3 – White
- Pin #4 – Black
- Pin #7 – Purple
- Pin #8 – Orange

NOTE: Check the Semtech instructions to verify the pinout, which may change without notice!

- 3) Now plug the SD card for the transmitter into your computer.

- 4) Copy the file:

AUTORUN.BAT

From the directory:

/SAMPLE TX-RX TRANSMITTER/TRANSMITTER

To the root directory of the SD card. This causes the DH+ to automatically run this file when it powers up.

- 5) Plug the SD card into the DH+.
- 6) Install a 9V battery.
- 7) Power on the DH+.
- 8) You will see the four middle LED's flash periodically whenever the system is transmitting.

If you plug in a serial cable + NULL modem adapter and then use D-Terminal or another terminal program (set to default baud rate of 19200bps) you can send a Ctrl+X to the DH+ to break out of the AUTORUN.BAT file. The terminal program will also indicate "Sending..." every second.

Setting up the Receiver:

The Data Hog Receiver is created following these steps:

- 1) Plug in wires into the side port on the DH+ exactly the same as the Transmitter.
- 2) Plug the wires into the SX1232 module exactly the same as with the Transmitter.
- 3) Now plug the SD card for the receiver into your computer.

- 4) Copy the file:

AUTORUN.BAT

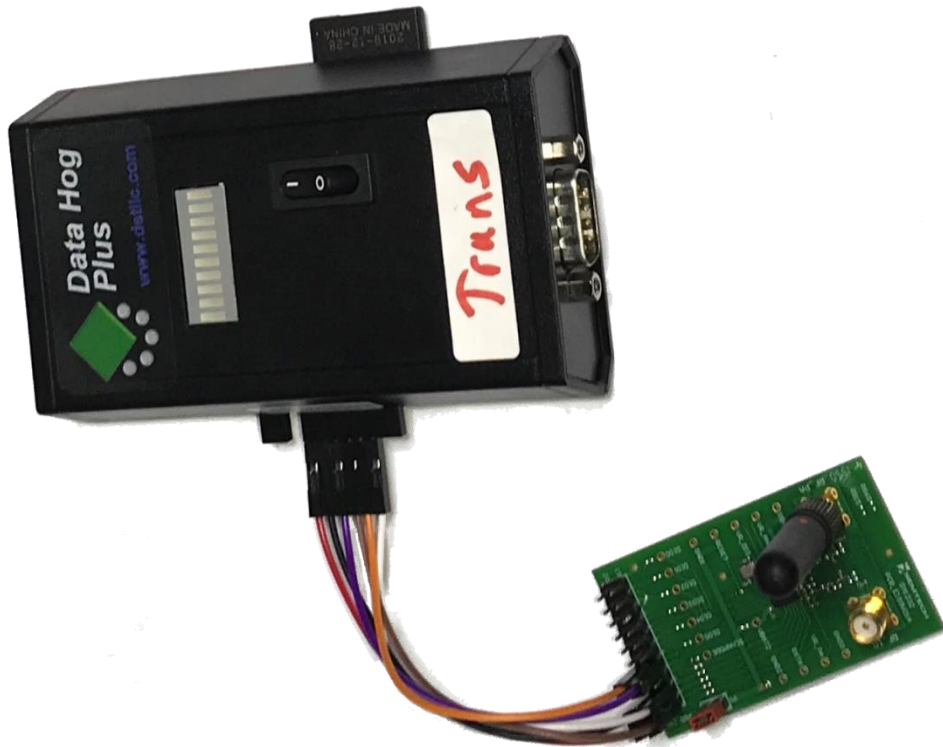
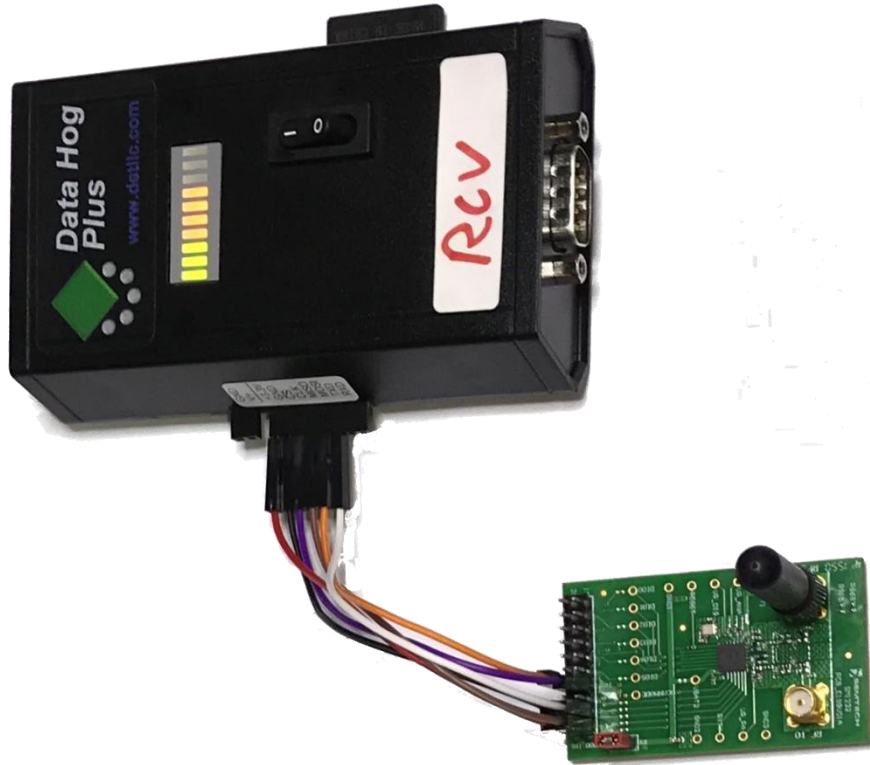
From the directory:

/SAMPLE TX-RX TRANSMITTER/RECEIVER

To the root directory of the SD card. This causes the DH+ to automatically run this file when it powers up.

- 5) Install a 9V battery.
- 6) Power on the DH+.
- 7) If the transmitter is also on, the receiver should start showing signal strength after a few seconds!

Example of completed Project:



Transmitter "AUTORUN.BAT" file:

```
//-----  
// DATA HOG WITH SEMTECH - TRANSMITTER  
//-----  
//Setup LED's (normal mode) and SPI Port Config  
LEDS=1  
SPI=ENABLE,MASTER,3,0,0,MSB,MAN  
  
SEND1 "Setting Mode..."  
  
// Set Transmitter to Standby  
SENDS "\0\x81\x01\1"  
  
// Setup Semtech Transmitter Variables  
//REG      RegBitrateMsb      0x02      0x00  
//REG      RegBitrateLsb      0x03      0xD5  
//REG      RegFdevMsb         0x04      0x09  
//REG      RegFdevLsb         0x05      0x9A  
SENDS "\0\x82\x00\xD5\x09\x9A\1"  
  
//REG      RegPaConfig        0x09      0x8F  
//REG      RegPaRamp          0x0A      0x09  
SENDS "\0\x89\x8F\x09\1"  
  
//REG      RegOcp              0x0B      0x3B  
//REG      RegLna              0x0C      0x23  
//REG      RegRxConfig        0x0D      0x0E  
SENDS "\0\x8B\x3B\x23\x0E\1"  
  
//REG      RegRxBw            0x12      0x10  
//REG      RegAfcBw           0x13      0x11  
SENDS "\0\x92\x10\x11\1"  
  
//REG      RegAfcFei          0x1A      0x01  
SENDS "\0\x9A\x01\1"  
  
//REG      RegPreambleDetect   0x1F      0xAA  
SENDS "\0\x9F\xAA\1"  
  
//REG      RegOsc              0x24      0x07  
SENDS "\0xA4\x07\1"  
  
//REG      RegSyncValue1      0x28      0xAA  
//REG      RegSyncValue2      0x29      0xBB  
//REG      RegSyncValue3      0x2A      0xCC  
//REG      RegSyncValue4      0x2B      0xDD  
SENDS "\0xA8\xAA\xBB\xCC\xDD\1"  
  
//REG      RegPayloadLength    0x32      0x40  
SENDS "\0xB2\x40\1"  
  
//REG      RegFifoThresh       0x35      0x8F  
SENDS "\0xB5\x8F\1"  
  
//REG      RegPaDac            0x5A      0x87  
SENDS "\0xDA\x87\1"  
SEND1 "\r\n"
```

```

// **** SET TO TRANSMIT MODE ****
SEND1 "Setting to transmit..."
//REG          RegOpMode          0x01          0x03 (TX) 0x05 (RX)
SENDS "\\0\x81\x03\1"
SEND1 "\\r\n"

// **** MAIN LOOP ****
// Send the current time. Start with CS=0,
// then the header + length, then the time (8 chars)
// followed by CS=1
:LOOP
SEND1 "Sending..."
SENDS "\\0\x80\x08%H:%N:%S\1"

// Wait for 1 second.
IDLE 1
SEND1 "\\r\n"

// Check if we have received an 0x18 on the RS232 port.
// If so, exit the program. If not, loop back to resend
// the time.
:CHECKRCV
IF %a="0" GOTO LOOP
RCV1 "%c"
IF %1="\x18" GOTO DONE
GOTO CHECKRCV

// Received an 0x18 on RS232 port, exit.
:DONE
SENDS "\\0\x81\x01\1"
SEND1 "\\r\nDONE!\r\n"

```

Receiver "AUTORUN.BAT" file:

```
//-----  
// DATA HOG WITH SEMTECH - RECEIVER  
//-----  
//Setup LED's (off) and SPI Port Config  
LEDS=3  
LEDLEVEL=0  
SPI=ENABLE,MASTER,3,0,0,MSB,MAN  
  
// Open a log file  
OPENA  
  
SEND1 "Setting Mode..."  
  
// Set to Standby  
SENDS "\0\x81\x01\1"  
  
//REG      RegBitrateMsb      0x02      0x00  
//REG      RegBitrateLsb      0x03      0xD5  
//REG      RegFdevMsb         0x04      0x09  
//REG      RegFdevLsb         0x05      0x9A  
SENDS "\0\x82\x00\xD5\x09\x9A\1"  
  
//REG      RegPaConfig        0x09      0x8F  
//REG      RegPaRamp          0x0A      0x09  
SENDS "\0\x89\x8F\x09\1"  
  
//REG      RegOcp              0x0B      0x3B  
//REG      RegLna              0x0C      0x23  
//REG      RegRxConfig        0x0D      0x0E  
SENDS "\0\x8B\x3B\x23\x0E\1"  
  
//REG      RegRxBw             0x12      0x09  
//REG      RegAfcBw           0x13      0x12  
SENDS "\0\x92\x09\x12\1"  
  
//REG      RegAfcFei          0x1A      0x01  
SENDS "\0\x9A\x01\1"  
  
//REG      RegPreambleDetect   0x1F      0xAA  
SENDS "\0\x9F\xAA\1"  
  
//REG      RegOsc              0x24      0x07  
SENDS "\0xA4\x07\1"  
  
//REG      RegSyncValue1       0x28      0xAA  
//REG      RegSyncValue2       0x29      0xBB  
//REG      RegSyncValue3       0x2A      0xCC  
//REG      RegSyncValue4       0x2B      0xDD  
SENDS "\0xA8\xAA\xBB\xCC\xDD\1"  
  
//REG      RegPayloadLength    0x32      0x40  
SENDS "\0xB2\x40\1"  
  
//REG      RegFifoThresh       0x35      0x8F  
SENDS "\0xB5\x8F\1"
```



```

//REG          RegPaDac          0x5A          0x87
SENDS "\0\xDA\x87\1"

SEND "\r\n"
// **** SET TO RECEIVE MODE ****
SEND "Setting to Receive, Variable Length, Zero Payload..."
//REG          RegOpMode          0x01          0x03 (TX) 0x05 (RX)
SENDS "\0\x81\x05\1"
SEND "\r\n"

// *** Wait for Packet ***
:LOOP
LEDLEVEL=0
SEND1 "Receiving..."
:LOOP2
SENDS "\0\x3F"
RCVS "%B\1"
IF %1 .2 1 GOTO GotPacket

:CHECKRCV
IF %a="0" GOTO LOOP2
RCV1 "%c"
IF %1="\x18" GOTO DONE
GOTO CHECKRCV

:DONE
SENDS "\0\x81\x01\1"
SEND1 "\r\nDONE!\r\n"
CLOSE
EXIT

:GotPacket
SENDS "\0\x11"
RCVS "%B\1"
IF %1 < "60" GOTO Level10
IF %1 < "80" GOTO Level9
IF %1 < "100" GOTO Level8
IF %1 < "120" GOTO Level7
IF %1 < "140" GOTO Level6
IF %1 < "160" GOTO Leve5
IF %1 < "180" GOTO Leve4
IF %1 < "200" GOTO Leve3
IF %1 < "220" GOTO Leve2
LEDLEVEL=1
OUTPUT "L01-"
GOTO GetValues
:Level10
LEDLEVEL=10
OUTPUT "L10-"
GOTO GetValues
:Level9
LEDLEVEL=9
OUTPUT "L09-"
GOTO GetValues
:Level8
LEDLEVEL=8
OUTPUT "L08-"
GOTO GetValues

```

```

:Level7
LEDLEVEL=7
OUTPUT "L07-"
GOTO GetValues
:Level6
LEDLEVEL=6
OUTPUT "L06-"
GOTO GetValues
:Level5
LEDLEVEL=5
OUTPUT "L05-"
GOTO GetValues
:Level4
LEDLEVEL=4
OUTPUT "L04-"
GOTO GetValues
:Level3
LEDLEVEL=3
OUTPUT "L03-"
GOTO GetValues
:Level2
LEDLEVEL=2
OUTPUT "L02-"

:GetValues
SEND1 " ["
SENDS "\0\x00"
RCVS "%B%8s\1"
SEND1 "%2"
OUTPUT "%2\r\n"

:GetFIFO
SENDS "\0\x3F"
RCVS "%B\1"
IF %1 .6 1 GOTO Empty
SENDS "\0\x00"
RCVS "%B\1"
SEND1 " 0x%1"
GOTO GetFIFO

:Empty
SEND1 "]\r\n"
GOTO LOOP

```

Receiver output to RS232 port (defaults to 19200 baud):

```

Setting Mode...
Receiving... [19:08:40]
Receiving... [19:08:41]
Receiving... [19:08:42]
Receiving... [19:08:43]
Receiving... [19:08:44]
Receiving... [19:08:45]
Receiving...

```