Acclima
Closed Loop Irrigation Systems

# DataSnap Developer Reference

## SDI-12 Data Recorder
## Part Number: ACC-AGR-D01


Acclima, Inc.
1763 W. Marcon Ln., Ste. 175
Meridian, ID 83642 USA
www.acclima.com

# Contents

# Introduction

This guide is designed to provide developers or advanced users with knowledge of how the DataSnap works and how to use the DataSnap without the Acclima SnapView software. It is intended for advanced users who have a strong knowledge of Microsoft Windows and computer hardware, and wish to interface with the DataSnap using a terminal program or custom software. For normal operating use or for general features and guidelines, please refer to the ***DataSnap User Manual***.

**Note:** The command interface outlined here is not intended for interchangeable use with the Acclima SnapView software. DataSnaps and sensors that have been configured using the DataSnap command interface outlined here may not work as intended when connected to the SnapView software.

## Basic Operation

When the DataSnap is properly installed on a computer USB port, it will implement a generic COM port on that computer. This COM port should be visible in Device Manager as 'Acclima Virtual COM port'. The communications settings for this COM port are unimportant, and can be ignored, since the output of the DataSnap is fixed as the SDI-12 protocol. This means that the DataSnap will operate regardless of the BAUD rate, stop bits, parity, or flow control settings. Interfacing with the DataSnap requires software that can open and send/receive data over a COM port.

## Definitions and Conventions used in this document

### Character Code Definitions

Some characters are not easily printed. These characters are shown instead by special sequences as defined here:

| Character Sequence | Name | Decimal Value | Description |
|---|---|---|---|
| <CR> | Carriage Return | 13 | The character usually generated by pressing the Enter key. In C and C# programming languages, this is the "\r" character. |
| <LF> | Line Feed | 10 | Sometimes known as a new line character. This character is represented as "\n" in the C and C# programming languages. |

### Special Font Use

Throughout this document, the use of `Courier Font` will be used to indicate information that is sent to or from the DataSnap over the USB communications channel. *`Courier Italics`* are used to indicate portions of a command string that should be replaced by one or more optional key words or character sequences. For example, consider the following text:

```
SENSOR action item=value<CR>
```

In this example, we will replace *`action`* and *`item=value`* with appropriate sub strings to perform a desired action. The data that will actually be transmitted to the DataSnap might look like this:

```
SENSOR READ ADDR=1<CR>
```

This command will cause the DataSnap to read the sensor found at SDI-12 address 1. Note the sequence <CR> is actually only 1 character that is not printable in document format.

## Command Interfaces

The DataSnap provides a USB communications interface for two different sets of commands: SDI-12 commands and DataSnap commands. Both of these command interfaces are active at the same time. The DataSnap determines which command was intended and processes that command appropriately through the use of command triggers. Until a command trigger is received, the DataSnap will collect and queue received characters, but will not evaluate or act on them in any way.

## Command Triggers

| Trigger Character | Description | Action (when detected) |
|---|---|---|
| ! | SDI-12 end of command character | All buffered characters up to this point are sent out the SDI-12 port with the timing (breaks, marks, etc) as required by the SDI-12 protocol version 1.3. |
| <CR> | Carriage Return character:<br>End of logger command | Previous characters in the buffer are examined for a command match. If a match is found, the internal command is executed. Finally, the buffer contents to this point are deleted even if a match is not found. |
| >127 | All non-ASCII or extended ASCII characters greater than 127 decimal | Extended ASCII characters are not allowed. These characters will trigger the same response as the internal version command: VER<CR>. All received data previous to the illegal byte value will be deleted from the receive buffer. |

## SDI-12 Command overview

The DataSnap acts as a transparent USB-to-SDI12 bridge for SDI-12 commands. The DataSnap transfers SDI-12 commands from the USB port to the SDI-12 port without modification to the command string. However, any required break signals, address mark, or other timing specific requirements are handled by the DataSnap. This removes the burden of SDI-12 timing from the computer, and simplifies the programming requirements for computer software, while providing a modern USB interface. Due to this architecture, all SDI-12 commands and capabilities are present when using the DataSnap, provided that the software using it is not limited. The SnapView software provided with the DataSnap can communicate with all types of SDI-12 sensors and is not limited to Acclima sensors.

Some of the detail on how the DataSnap handles SDI-12 commands can be adjusted through configuration settings. Please refer to the CONFIG command outlined in this document, with focus on the ECHO and RETRY items.

For more information about the SDI-12 protocol and associated commands, please visit the SDI-12 website at www.SDI-12.org
For more information on the Acclima SDI-12 Sensor, please visit www.acclima.com

## DataSnap Command overview

The primary purpose of this document is to describe the DataSnap command set. These commands are used to configure and alter the behavior of the DataSnap data logger. Refer to the following sections for more information about the DataSnap command set.

# DataSnap Commands

The DataSnap provides a command interface for setting the modes of operation and other tasks. This interface works over the same communication port and shares the connection with the SDI-12 communications. The DataSnap will determine if the command is intended as an internal function or as an SDI-12 command, and process it accordingly.

The DataSnap monitors and queues all incoming data until a command trigger (or end of command character) has been received. Once a command trigger has been received, all buffered characters up to and including the command trigger are acted on (the command is executed) and then the characters are deleted from the buffer.

**Note:** The command interface outlined here is not intended for interchangeable use with the Acclima SnapView software. DataSnaps and sensors that have been configured using the DataSnap command interface outlined here may not work as intended when connected to the SnapView software.

## DataSnap Command Table

| Root Commands | Name | Description |
|---|---|---|
| HELP<CR> | Help | List the commands that are available |
| VER<CR> | Version | Report the version of the DataSnap |
| CONFIG *item=value*<CR> | Configuration | Set the indicated configuration item to a new value. This is used to change global configuration items within the DataSnap, such as Time, Date, Logging Interval, etc. |
| STATUS<CR> | Logger Status | Show the current status of the Logger, including: date, time, operating voltage, log memory status, etc. |
| LOG *action*<CR> | Logs | Read or erase sensor data that has been recorded in the logger's internal memory. |
| Factory_Reset<CR> | Factory Reset | Erase all configuration and log data within the logger. Some data cannot be erased and will remain intact (such as LASTLOGID). |
| SENSOR *action item=value*<CR> | Sensors | Read, configure, or automatically locate sensors on the SDI-12 bus. |
| ---<CR> | Unrecognized | If the data preceding the carriage return <CR> is not recognized as a command, then no action is taken other than to purge the data up to and including the <CR> from the buffer. |

**Note:** The command interface is case-sensitive. For example, sending Ver<CR> instead or VER<CR> will not be recognized and will only purge the communications buffer.

Parameters are passed in and out of the DataSnap through the use of *item=value* fields. Multiple fields may be included on a single command line, and multiple fields are often reported by commands. These fields will be separated by a space and the entire command response will be terminated by a <CR><LF>. Some commands report multiple sets of data (such as SENSOR GET which will report all active sensor configurations). In this case, each record will terminate with a <LF><CR> and the entire command sequence after all records have been reported will terminate with the usual <CR><LF>. See individual command descriptions for more information.

## *HELP Command*

Command Sequence:

`HELP<CR>` or `HELP` *`command`*`<CR>`

The `HELP` command can be used to gather usage information from the DataSnap command interface using a terminal program. Typing `HELP<CR>` by itself will list the available commands. Including another command after the word `HELP` will print more detailed information specific to that command.

For example, the response to `HELP<CR>` may look something like this:

```
******** HELP Menu *********
   HELP          = Show this menu
   HELP <cmd>    = Show specific help about other commands in this menu
   VER           = Display product and version information
   STATUS        = Show the current status of the Data Snap logger
   CONFIG        = Read or Set global configuration data
   LOG           = Read/Delete Logs stored within the Data Snap
   SENSOR        = ADD/MODIFY/DELETE/READ Sensors
   Factory_Reset = Erase all logs and configuration data
Note:  All commands are case-sensitive
```

As another example, the response to `HELP VER<CR>` may look like this:

```
****  VER (Version) Command Help  ****
This command reports information about the logger
   Command format:  VER
      No arguments or parameters expected
   Output format:   Label=Data pairs
      MFG=Manufacturer  (Variable length string, no spaces no quotes)
      MODEL=Model_Name  (Variable length string, no spaces no quotes)
      VERSION=M.N.B     (M=Major rev, N=Minor Rev, B=Build)
      ID=Unique_ID      (64-bit number in Hexadecimal format)
```

## *VER Command (Version)*

Command Sequence:

`VER<CR>`

This command returns a string that shows the product information and version of the DataSnap. No sensor information is returned.

The Data reported by this command includes:

| Item name | Description | Format |
|---|---|---|
| `MFG` | Reports the Manufacturer (Acclima, Inc.) | String, terminated by space |
| `MODEL` | Reports the device model name (DataSnap) | String, terminated by space |
| `VERSION` | Reports the firmware version | Three integers separated by periods:<br>M.N.B=Major.Minor.Build<br>Each number is an independent field; the decimal is used as a separator only. (.21 is bigger than .9!) |
| `ID` | ID number that identifies this particular DataSnap | 64-bit number in Hexadecimal format |

For example, the response to `VER<CR>` may look like this:

```
MFG=Acclima MODEL=DataSnap VERSION=0.1.21 ID=495782488D93087D<CR><LF>
```

## *STATUS Command*

Command Sequence:

`STATUS<CR>`

This command reports the current operating status of the DataSnap

The Data reported includes:

| Item name | Description | Format |
|---|---|---|
| DATE | The current date<br>(To set the date, see CONFIG command) | dd/mm/yyyy |
| TIME | The current time<br>(to set the time, see CONFIG command) | hh:mm:ss (24-hour format) |
| SUPPLY_VOLTAGE | The operating voltage (Volts) inside of the DataSnap | Floating point number xx.xx (Volts) |
| OUTPUT_CURRENT | Shows the output current in Amps.<br>Current limit will occur at approximately 0.23 | Floating point number: x.xx (Amps) |
| SDI12_POWER | Shows if the SDI-12 power is currently on or off | ON = SDI-12 sensors are powered<br>OFF = SDI-12 sensors are not powered |
| SHORT_CIRCUIT | Reports the status of the over-current protection circuit. While over-current protection is active, the output voltage to the SDI-12 sensors may drop and cause unreliable readings. | NO = No over-current conditions exist (OK!)<br>YES = Current limit active. The voltage source is protected, but sensors may be experiencing low voltage. |
| FIRSTLOGID | The unique ID of the first data record in log memory (oldest). | 64-bit unsigned integer |
| LASTLOGID | The unique ID of the last data record written to log memory (most recent). This number will increment with each data point (log) that is written. | 64-bit unsigned integer |
| MEM_USED | Shows the amount of log memory that currently holds data. The memory space is circular, so instead of filling completely, older data is overwritten. If this number is greater than 95, then it is very likely that old data has been overwritten by new data. The total log space can hold 59392 records. | Integer percent (0 to 100), no percent sign is printed |

For example, the response to `STATUS<CR>` may look like this:

```
DATE=06/02/2012 TIME=12:05:00 SUPPLY_VOLTAGE=11.92 OUTPUT_CURRENT=0.00
SDI12_POWER=ON SHORT_CIRCUT=NO FIRSTLOGID=0 LASTLOGID=15262 MEM_USED=25<CR><LF>
```

## *CONFIG Command (Configuration)*

Command Sequence:

`CONFIG<CR>` or `CONFIG` *`item=value item=value …`*`<CR>`

This command is used to read and to set items within the global configuration for the DataSnap. These configuration items will affect the operation of the DataSnap and all attached sensors (if any). For configuring specific sensors, see the `SENSOR` command.

The output from the `CONFIG` command will always include all available items accessible through the `CONFIG` command. To read the configuration, no arguments are required. To write configuration items, the items must be specified after the `CONFIG` command in item=value format with spaces between items. Note that there should not be a space on either side of the =. Any number of items may be specified on the command line. If an item is specified twice, only the last item will be recognized. The total string length should not exceed 1500 bytes.

The items set by the CONFIG command are committed to internal non-volatile memory. These settings will be saved even if power to the DataSnap has been lost. After sending this command, there must be a one second delay before the DataSnap can be powered off.

### Items recognized and reported by the CONFIG command:

| Item name | Description | Format |
|---|---|---|
| DATE | Set the current date | dd/mm/yyyy |
| TIME | Set the current time | hh:mm:ss (24-hour format) |
| NAME | The customizable name of this DataSnap | Delimited string format, 40 chars max<br>Delimiter = vertical bar \|<br>Illegal characters in the string: Delimiter, Command triggers<br>Example: NAME=\|My new DataSnap\| |
| INTERVAL | Number of minutes between automatic sensor readings. All configured sensors will read on the same interval. | Integer value = minutes<br>Read Range = 1 to 43200 (minutes)<br>Other values (like 0) = Disable automatic sensor readings |
| STARTUP | Number of milliseconds to wait for sensors to be ready after they have been powered on. This value is only used if IDLE_POWER=OFF. | Integer value = milliseconds<br>Range = 0 to 5000<br><br>Example: STARTUP=200 |
| IDLE_POWER | This setting controls the SDI-12 PWR wire when the DataSnap is sleeping between sensor readings. For lowest power consumption, use the OFF setting. When the USB connection is valid, the DataSnap will not sleep, and the SDI-12 PWR will always be ON regardless of this setting. | ON = Always leave the SDI-12 sensors powered<br>OFF = Turn off power to the SDI-12 sensors when the DataSnap is idle (and sleeping).<br><br>Example: IDLE_POWER=OFF |
| ECHO | Turn on or off the echo feature inside the DataSnap. When ECHO=ON, any characters received over the USB connection will be transmitted back over the same link. This can be useful when using a terminal program so that typed characters will appear on the terminal. | ON = Transmit received characters and responses<br>OFF = only transmit responses<br><br>Example: ECHO=ON |
| RETRY | Turn on or off automatic retries of SDI-12 pass-through commands. This setting only applies to SDI-12 commands issued through the USB connection. Internal logging functions will always retry. | ON = Retry failed SDI-12 commands up to 9 times.<br>OFF = Automatic retry disabled.<br>*See "Retry Mode" below for more information* |

## *Retry Mode*

The DataSnap is capable of automatically performing retries as outlined in the SDI-12 Protocol Specification version 1.3. This capability has some impact on timing and response format as received by the host program on the other side of the USB connection. Retry mode only impacts SDI-12 pass-though commands from the host program. Internal logging functions will always retry where appropriate. Retry mode behaves as follows:

### RETRY=OFF

When retry mode is disabled, any data that appears on the SDI-12 bus will immediately be sent to the USB COM port without further examination or delay. In this mode, the DataSnap acts as a transparent bridge device, and if any retries are desired, then the host software is responsible for detecting the "no response" condition and performing the appropriate retries. This mode will return SDI-12 character data without the delays caused by the buffering and analysis that occurs in the Retry On mode. Break timing and address mark timing are still applied by the DataSnap as required by the SDI-12 specification.

RETRY=ON

The DataSnap will monitor and buffer the response sent by the SDI-12 sensor. If an SDI-12 response terminator is found (<CR><LF>), then the entire message is relayed to the computer. If a response terminator is not found, or if there are parity errors, then the DataSnap will transmit again (up to 9 total attempts) as outlined in the SDI-12 Protocol Specification version 1.3.

If there was no response after 9 attempts:

The DataSnap sends only <CR><LF> to the computer. The listening computer or terminal software will know that the command is complete when the <CR><LF> has been received. If there is no data preceding the <CR><LF>, then there was no response from the SDI-12 sensor.

If there were Parity errors:

The DataSnap sends the received SDI-12 message, including the terminating <CR><LF> to the host computer. Wherever a parity error occurred, the character will be replaced by É (decimal value 144). The host software can search the received string for this character to detect parity errors.

## *Factory_Reset Command*

Command Sequence:

```
Factory_Reset<CR>
```

This command will erase all sensor data and configuration data from the DataSnap.
Progress will be output and re-output periodically as follows:

```
Erasing... 3%<CR>
```

Once the DataSnap has been erased and reset, the output will be:

```
Factory Reset Complete.  LASTLOGID=12345<CR><LF>
```

Where LASTLOGID shows the current value of the auto-incrementing, unique data record ID. (See LOG command)

## *SENSOR Command*

Command Sequence:

```
SENSOR action item=value<CR>
```

This command performs several operations or actions regarding sensors or their configuration. The *action* is a required argument, while *item=value* arguments are optional in many cases.

SENSOR actions:

| Item name | Description | Additional Requirements |
|---|---|---|
| GET | Read the configuration for the indicated sensors. Automatic logging will only occur for configured sensors. | Show all sensors:<br>None<br>Show one sensor:<br>ADDR=*value* or ID=*value* |
| SET | Add a sensor or modify its configuration. Automatic logging will only occur for configured sensors. When adding a sensor to the configuration, NAME and IDENTITY will default to empty strings if no value is assigned. ID will default to an automatically generated Unique ID if not specified. When modifying a sensor configuration, unspecified values will remain unchanged. | Add:<br>ADDR=*value*<br>Modify:<br>ADDR=*value* or ID=*value* |
| DEL | Delete a sensor from the configuration. All remaining sensor configurations will be output as a result of this command (similar to GET<CR>) | ADDR=*value* or ID=*value* |
| READ | Read the indicated sensor. Return a log entry in the same format that is used in the LOG command. | ADDR=*value* or ID=*value* |
| ACCFIND | Scan for Acclima SDI-12 sensors. This feature requires special extended commands and special communications hardware, and works only with Acclima sensors. | None |

## *SENSOR GET, SET, DEL*

These commands are used to modify the configured sensors in the DataSnap sensor configuration. Once a sensor is configured, the DataSnap will automatically record up to 5 readings from that sensor every INTERVAL minutes (see the CONFIG command).

The SET action returns the entire configuration record for the sensor. The GET action can return a single sensor configuration record if ADDR or ID is specified, but will otherwise return all sensor configuration records. The DEL action returns all remaining configuration records after the DEL action is applied.

Items recognized and reported by the SENSOR GET/SET/DEL command:

| Item name | Description | Format |
|---|---|---|
| ADDR | This is the SDI-12 address of the configured sensor. This must be unique among the sensors that are attached to a particular DataSnap. This can be used to identify which sensor to perform an operation on (READ, DEL, etc). | Use SDI-12 address guidelines.<br>Recommended addresses: 1-9, A-Z, a-z<br>Not Recommended: 0<br>While address 0 is allowed, it is not recommended since sensors may be programmed to that value by default. Avoiding address 0 may help prevent address conflicts. |
| ID | This number should uniquely identify a particular SDI-12 sensor. This number and the address item are used to identify sensors in the sensor logs. For Acclima sensors, this is the sensor serial number. | 32-bit unsigned integer |
| NAME | The customizable name of the sensor. | Delimited string format, 40 chars max<br>Delimiter = vertical bar \|<br>Illegal characters in the string: Delimiter, Command triggers<br>Example: NAME=\|Sensor 1: 3" deep\| |
| IDENTITY | This value is used by the SnapView software, but is not required for the DataSnap to operate independently. For independent developers, this value can be customized. | Delimited string format, 34 chars max<br>Delimiter = vertical bar \|<br>Illegal characters in the string: Delimiter, Command triggers<br>Example:<br>IDENTITY=\|B13Acclima 0030317.06002553\| |

For a single configuration record, the output may look like this:

```
ADDR=9 ID=567890 NAME=|Sensor4| IDENTITY=|913Acclima 0030317.0567890|<LF><CR><CR><LF>
```

For multiple sensors, the output format will look like this:

```
ADDR=B ID=6002553 NAME=|Sensor1| IDENTITY=|B13Acclima 0030317.06002553|<LF><CR>
ADDR=1 ID=6003561 NAME=|Sensor2| IDENTITY=|113Acclima 0030317.06003561|<LF><CR>
ADDR=5 ID=12345 NAME=|Sensor3| IDENTITY=|513Acclima 0030317.012345|<LF><CR>
ADDR=9 ID=567890 NAME=|Sensor4| IDENTITY=|913Acclima 0030317.0567890|<LF><CR><CR><LF>
```

## *SENSOR READ*

This command is used to read a sensor that has been added to the sensor configuration. Only one sensor can be read at a time, and the ADDR or ID must be specified to identify which sensor to read. The output from this command will be in the CSV format (same as the LOG command) as follows:

```
DATE,TIME,Type,ID,ADDR,R1,R2,R3,R4,R5,<CR><LF>
```

In the LOG command, the last field would be populated with the LOGID, but since this reading was not written to the logs, the LOGID field will be empty. For more information regarding this format, see the LOG command.

### *SENSOR ACCFIND*

This command causes the DataSnap to search for Acclima SDI-12 sensors on the SDI-12 bus. This capability to communicate with multiple unaddressed sensors is unique to Acclima sensors. This function will only work if all sensors connected to the DataSnap are made by Acclima.

This command causes the DataSnap to enter a special 'sensor search' mode. While the DataSnap is searching for sensors, it will not be available to process additional commands. Any additional data sent to the DataSnap will be queued and processed later after the scan is complete.

As sensors are located, their identity string is reported to the computer. Each sensor identity is terminated by a <LF><CR> character pair. The identity string itself is delimited with vertical bars (|). When the entire scan sequence is complete, the DataSnap ends the scan mode by sending a <CR><LF> to the computer. Each individual sensor identity will look similar to this:

```
IDENTITY=|113Acclima 0030317.01231234|<LF><CR>
```

Refer to the SDI-12 Protocol Specification found at www.SDI-12.org for more information about the format of the response to the identify command (aI!).

The result of a scan involving 4 sensors might look something like this:

```
IDENTITY=|113Acclima 0030317.01231234|<LF><CR>
IDENTITY=|213Acclima 0030317.01232345|<LF><CR>
IDENTITY=|A13Acclima 0030317.01233456|<LF><CR><CR><LF>
```

## *LOG Command*

Command Sequence:

```
LOG action item=value<CR>
```

This command is used to read sensor data that has been recorded in the DataSnap's memory or to erase that memory. The *action* is a required argument, while *item=value* arguments are optional.

### *About Logs*

The memory that contains historical sensor data within the DataSnap is called the log space. Each record within the log space is called a log. Each individual log can contain either sensor data or error data. If the log contains sensor data, up to 5 data points can be stored for a particular sensor within a single log. Each log contains an identifier that uniquely identifies that particular log entry. This identifier is called the Log ID, and it automatically increments each time a sensor log or error log is written, or when the log space is erased. The Log ID is useful for synchronization purposes where much of the data has already been downloaded and only a portion of the data is yet unknown.

### *LOG actions*

| Item name | Description | Additional Requirements |
|---|---|---|
| GET | Read sensor data that is stored in the DataSnap internal memory. Use LASTLOGID and NUMLOGS items to control the results. | Read all logs:<br>None<br>Read logs starting from:<br>LASTLOGID=*value*<br>Read at most:<br>NUMLOGS=*value* |
| DEL | Delete all sensor data and error data from internal memory. | None |

## *LOG GET*

`LOG GET` is used to read the historical sensor data and error data that are stored in the log space. Since the DataSnap can hold up to 59392 log entries, some additional arguments are accepted to allow finer control over the data download.

### Accepted Arguments:

| Argument | Description | Format |
|---|---|---|
| `LASTLOGID` | This is used to synchronize only the data that has not been downloaded already. `LASTLOGID` should be set to the value of the LOGID that was most recently collected. The DataSnap will output any logs that have a log ID greater than this number. | Positive 64-bit integer (unsigned)<br><br>Example:<br>`LASTLOGID=123456789` |
| `NUMLOGS` | This specifies the maximum number of logs to return. If this number is not specified, then all the logs will be output by the DataSnap. Since this may take several minutes, the use of `NUMLOGS` and `LASTLOGID` in combination allows downloading data in a more controlled fashion. | Positive integer<br>Range = 1 to 59392<br><br>Example:<br>`NUMLOGS=100` |

The output of the `LOG GET` command appears as a table of values in comma separated values (CSV) format. No header row appears, and it is assumed to be:

DATE,TIME,TYPE,ID,ADDR,DATA1,DATA2,DATA3,DATA4,DATA5,LOGID

Each row of data is terminated by `<LF><CR>`
The end of the command (and end of the table) is signaled by `<CR><LF>`
If there is no data present for a particular column, then a number will not appear there but the commas separating the columns will remain in place.

### Each Data column can be decoded as follows:

| Column Name | Description | Format |
|---|---|---|
| `DATE` | The date that this data was recorded | dd/mm/yyyy |
| `TIME` | The time of day that this data was recorded | hh:mm:ss (24-hour format) |
| `TYPE` | The type of log entry | 0 = Error data<br>1 = Sensor data |
| `ID` | The ID of the sensor that caused this data to be written<br>This will be zero for system errors. | 32-bit unsigned integer |
| `ADDR` | The SDI-12 address of the sensor that caused this data to be written. For system errors, this column is “Don’t Care”. | ASCII Character (not a number)<br>“0”-“9”; “A”-“Z”; “a”-“z” |
| `DATA1` | For errors: This is the error code<br>For sensor data: This is the 1st data point returned by the sensor. | floating point, 6 places after the decimal<br>Example: 19.500000 |
| `DATA2` | For errors: Not used<br>For sensor data: This is the 2nd data point returned by the sensor. | floating point, 6 places after the decimal<br>Example: 19.500000 |
| `DATA3` | For errors: Not used<br>For sensor data: This is the 3rd data point returned by the sensor. | floating point, 6 places after the decimal<br>Example: 19.500000 |
| `DATA4` | For errors: Not used<br>For sensor data: This is the 4th data point returned by the sensor. | floating point, 6 places after the decimal<br>Example: 19.500000 |
| `DATA5` | For errors: Not used<br>For sensor data: This is the 5th data point returned by the sensor. | floating point, 6 places after the decimal<br>Example: 19.500000 |
| `LOGID` | This number uniquely identifies this particular log entry for this DataSnap. This number automatically increments with each log written. | 64-bit unsigned integer |

**Possible error codes in `DATA1` of an error log:**

| Error Code | Error Name | Description |
|---|---|---|
| 0.10 | RTC Failure | The real-time clock has failed and it is not running |
| 0.20 | Short Circuit | The SDI-12 PWR output is overloaded. Voltage output may not be optimal. |
| 1.00 | No Response | No response from the indicated SDI-12 sensor |
| 1.10 | Parity Error | A parity error occurred while communicating with the indicated SDI-12 sensor. Check for address conflicts. |
| 1.20 | Bad Sensor Index | Internal error: Unable to locate the requested sensor configuration |
| 1.30 | Bad Response | Unable to decode the sensor's response to the last SDI-12 command |
| 1.40 | No Data | No data was returned by the indicated sensor after taking a measurement |
| 2.00 | Time Changing | Written before the time and date are changed on the DataSnap. The will happen as a result of the CONFIG command where TIME or DATE are specified. |
| 2.10 | Time Changed | Written after the time has been changed. |
| 2.20 | Power Restored | Power was lost |

The following example illustrates the log download process:

Command Sent:

```
LOG GET LASTLOGID=15000 NUMLOGS=5
```

Data Received:

```
06/02/2012,07:49:03,1,6003561,1,0.000000,19.000000,1.010000,0.000000,,15001<LF><CR>
06/02/2012,07:49:04,0,12345,5,1.000000,,,,,15002<LF><CR>
06/02/2012,07:49:05,0,567890,9,1.000000,,,,,15003<LF><CR>
06/02/2012,07:50:02,1,6002553,B,0.000000,19.400000,1.070000,0.000000,,15004<LF><CR>
06/02/2012,07:50:03,1,6003561,1,0.000000,19.100000,1.010000,0.000000,,15005<LF><CR>
<CR><LF>
```

In this example there are 3 valid sensor readings that contain 4 data points each, and 2 errors that show the sensors located at addresses 5 and 9 are not responding.