LabSmith Application Note

Logging Data in uProcess™

By Michael Duperly and Annika Armstrong
LabSmith, Inc., Livermore, CA 94550

LabSmith's uProcess™ microfluidic automation software lets you track statuses and measurements from connected uDevices using data logs. Data logs are useful for analyzing the readings of individual uDevices, tracking their changes over time, comparing data from several different uDevices simultaneously, and logging experimental parameters. The latest version of uProcess introduces custom data logs for greater flexibility.

Introduction

Data logs made with uProcess are saved as .csv spreadsheets on your computer. All logs contain columns for the time of day and the amount of time passed since logging started. Then, depending on the connected uDevices, additional columns display all other traceable measurements. The data collected by different uDevices is shown in Table 1.

Table 1. uDevice Data Collection

uDevice		Data Collected
4VM valve manifold	**AV201 AV202 AV303 AV801 Valves**	• Valve state
4AM sensor manifold	**uPS pressure sensor**	• Pressure • Compensation temperature
	uTS temperature sensor	• Probe temperature • Compensation temperature
4PM power manifold	**uTE**	• Status • Voltage • Current
uEP01 electrophoresis power module		• Voltage • Current • Flags
SPS01 syringe pump		• Volume
Third-party devices		• Varies depending on device

In data logs, recorded entries have the same default units as uProcess. However, 4VM entries display the state of each channel using numbers rather than descriptors (like "Position A," "Position B," and "Closed"). The corresponding values are shown in Table 2.

Table 2. Log Entries for Valve Positions

Valve Type	Position / Movement	Corresponding Log Value
AV201	At Position A (left)	1
	At Closed (middle)	2
AV202	At Position B (right)	3
	Moving from Position A	5
AV303	Moving from Closed	6
	Moving from Position B	7
AV801	At Position 1	1
	At Position 2	2
	...	...
	At Position 8	8
	Did not detect Position 1	9+
Any	Disconnected / indeterminate position	15
	Stuck / stalled	-1

There are two main ways to create a data log. First, data collection can be manually started and stopped. This is best for collecting data as you change the states of uDevices with the Interfaces tab in uProcess. Second, data collection can be automatically started and stopped via script commands. This is best for collecting data while a script is running. In both cases, data is collected from all connected uDevices.

Creating a Data Log Manually

Logging device statuses and measurements from uDevices while manually changing their settings does not require a script. First, connect any uDevices for intended data collection. Then, go to the **View** tab in uProcess and click **Status/Meas Logging Options**. This opens the **Status Tracing Settings** window, with options for how often data is collected ("log entry interval") and whether uProcess should automatically name and save log files. If log files are not automatically named, uProcess will request a name each time a new log file is created (see Fig. 1).

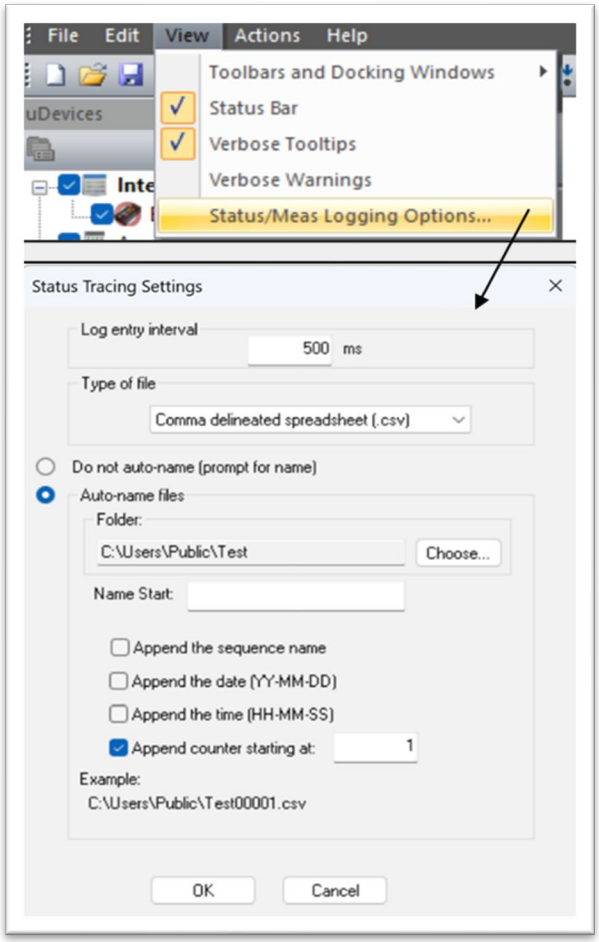

Figure 1. Status Tracing Settings

Once you have configured these settings, click **OK**. Next, go to the **File** tab and click **Log**. Status/Measurements (see Figure 2).

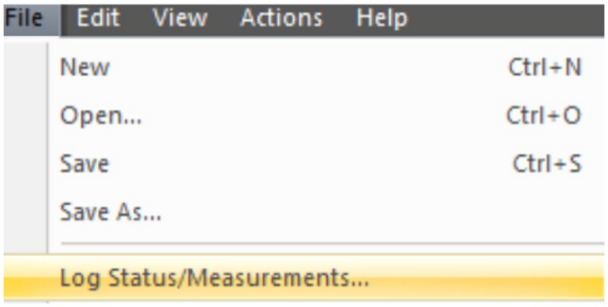

Figure 2. Manually enabling logging

If you did not opt to auto-name log files earlier, you will now be asked to choose a file name and file save location. Otherwise, the file is automatically created, and logging begins. Now, if you change device states or have connected sensors, state changes and measurements are logged.

Data will continue to be collected until you uncheck **Log Status/Measurements**, click **Rescan Devices**, or close uProcess.

Creating a Data Log Automatically

Device statuses and measurements can be automatically logged using the scripting feature in uProcess. First, write a script to complete your desired task per usual. Then, simply add a `Log(on)` command at the beginning to start collecting data, and a `Log(off)` command at the end to stop collecting data.

```
*Syringe = SPS 80 ul

Log("C:\Temp\example name.csv")

       Syringe: MoveTo(80.0 ul)
       WaitDone()

Log(off)
```

If you did not opt to auto-name log files earlier, you will be asked to choose a file name and file save location upon running the script. Otherwise, the file is automatically created, and logging begins. Data collection occurs only during the duration between `Log(on)` and `Log(off)` in the script. The resulting log file will contain columns for all connected uDevices, including those not mentioned in the script.

Additionally, a `Log("optional filename")` command can be used in place of a `Log(on)` command to start logging and to pre-name log files. The file name should end in .csv and the entire file name/path must be in quotes. If you include a directory, make sure it exists and that you have access.

```
*LabSmith_SPS01 = SPS 80 ul

Log("C:\Temp\example_name.csv")

Fill_Syringe:
   LabSmith_SPS01: MoveTo( 80.000 ul)
   WaitDone()

Log(off)
```

When using the `Log("optional filename")` command without including a directory, log files are saved to the same location specified in the **Status Tracing Settings** window.

NOTE: if you do not use a unique file name each time you run the script, new data will be entered into the preexisting log file of the same name (after the last occupied row in the spreadsheet).

Creating a Custom Data Log

uProcess version 2.00.64 and later includes a custom-defined data log feature. These data logs allow standard or user-defined variables to be logged at specified times within a script via commands.

All commands used to create custom logs must referece a file for entering text into. Treat the file as a variable; define it at the start of the script and simply reference the variable in each command instead of the whole file path. Include a file name, type, and directory (make sure it exists and that you have access) when defining the file.

```
file = "C:\Temp\example name.csv"
```

NOTE: the file type should be .csv and the entire file path must be enclosed by quotes. Also, if you do not use a unique file name each time you run the script, new data will be entered into the preexisting log file of the same name (after the last occupied row in the spreadsheet).

Within each command, parameters (true, 1, false, or 0) indicate whether uProcess should write column headings or enter rows of data into the log. Using "true" (or 1) tells uProcess to write the name of the object or uDevice rather than entering a measurement or value. This is used for generating headings in the log file. "False" (or 0, or the absence of the parameter) tells uProcess to enter the measurement/value. This is used for generating rows of data below the column headings.

LogEntry

This command tracks the time of day and the amount of time passed since logging started.

`LogEntry(file, true)` `LogEntry(file, 1)`	Writes "Time of day" and "Time (s)"

`LogEntry(file, false)`
`LogEntry(file, 0)`
`LogEntry(file)`
	Logs the values of the time of day and time passed

LogMeasurements

This command tracks the same measurements from connected uDevices as a standard data log (see Table 1 for collected data).

`LogMeasurements(file, true)` `LogMeasurements(file, 1)`	Writes the names of any connected uDevices

`LogMeasurements(file, false)`
`LogMeasurements(file, 0)`
`LogMeasurments(file)`
	Logs the statuses of any connected uDevices

LogVariables

This command tracks all standard and user-defined variables in the script.

`LogVariables(file, true)` `LogVariables(file, 1)`	Writes the names of any variables

`LogVariables(file, false)`
`LogVariables(file, 0)`
`LogVariables(file)`
	Logs the values of any variables

NOTE: the list of standard variables appears in the Sequencer window when the script is run. Available variables correspond to devices referenced in the script. For example, Figure 3 shows the variables available when a script uses one syringe pump and one valve.

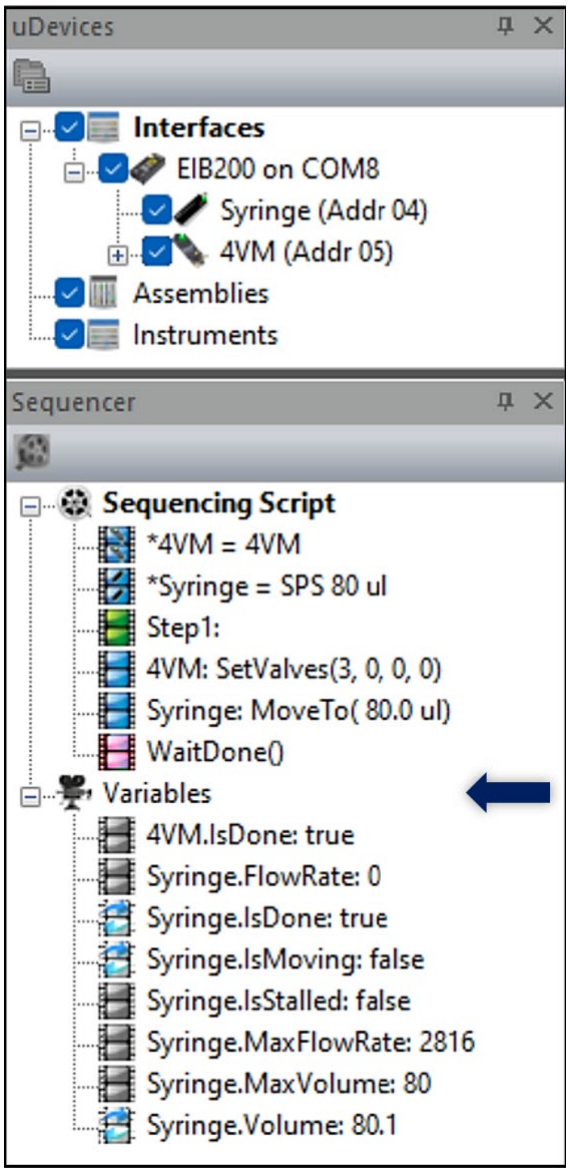

Figure 3. Variable list

LogItem

This command tracks a singular uDevice, variable, expression, or enters text into the log.

`LogItem(file, x, true)` `LogItem(file, x, 1)`	Writes the name of the measurement, variable, expression, or "text" (in double quotes)

`LogItem(file, x, false)`
`LogItem(file, x, 0)`
`LogItem(file, x)`
	Logs the value of the measurement, variable, expression, or the "text"

NOTE: the "x" is a placeholder for the desired measurement, variable, expression, or "text" in the log.

Organizing the custom log commands into two separate functions is recommended. Designate one for creating column headings and the other for generating rows of data (e.g. `LogHeader` and `LogRow`). Place (or call) `LogHeader` before `LogRow` so that headings appear before rows of data in the log. Ensure that the commands in each function are written in a corresponding order so that the column headings made with `LogHeader` align with the data entries created by `LogRow` (see the following example).

Organizing LogHeader and LogRow Example Script

```
LogHeader:
        LogEntry(file, 1)
        LogVariables(file, 1)
        LogItem(file, "sample text", 1)
        return

LogRow:
        LogEntry(file)
        LogVariables(file)
        LogItem(file, *item*)
        return
```

The above script will create one row of headings and one row of data. The functions in the `LogRow` subroutine must be called every time data is to be logged. This can be achieved by putting the `LogRow` function in a loop, or by calling the function following an event, as described in the next section.

Event-Triggered Logging

The custom-defined logging feature is most commonly used to log variables after an event, such as upon reaching a certain temperature, pressure, volume, time, or cycle count. In the following example, a syringe pump is used to pressurize a reservoir. The event log records the time and the number of cycles required to achieve the desired pressure (see the example script).

Event-Triggered Logging Example Script

```
*4VM            = 4VM
*Syringe        = SPS 80 ul
*P22904         = uPS 800 kPa

file = "C:\Temp\example name.csv"
Count = 1

Call LogHeader

Fill_Syringe:
        4VM:            SetValves(3, 0, 0, 0)
        Syringe:        SetFlowRate(300.0 ul/min)
        Syringe:        MoveTo(80.0 ul)
        WaitDone()

Dispense_Syringe:
        4VM:            SetValves(1, 0, 0, 0)
        Syringe:        MoveTo(1.0 ul)

Loop:
        if (Syringe.Volume <= 5 ul)
        {
                Count = Count+1
                Goto Fill_Syringe
        }

        if (P22904.Reading > 100 kPa)
        {
                Syringe: Stop()
                Call LogRow
                Goto End
        }

        Wait(500 ms)
        Goto Loop

End:
        Quit

LogHeader:
        LogEntry(file, 1)
        LogItem(file, Count, 1)
        LogItem(file, "Pressure", 1)
        return

LogRow:
        LogEntry(file)
        LogItem(file, Count)
        LogItem(file, P22904.Reading)
        return
```

The example script creates headings for "Time of Day," "Time (s)," "Count," and "Pressure." Upon reaching the specified pressure, one line of data is logged. The resulting data log is shown in Figure 4.

	A	B	C	D	E
1	Time of day	Time (s)	Count	Pressure	
2	15:08:04	202.781	7	100.3	
3					

Figure 4. Event log

Troubleshooting

Problem	Solution
Log entries for user-defined variables in a custom data log are not changing.	User-defined variables do not update automatically like the default variables (volumes, pressure readings, etc.). Redefine these variables immediately before the LogRow function to log a row of data.
Log entries are recorded after previous entries in an existing log file.	Assign a unique file name each time you run the script when using the `Log("optional filename")` command or when creating a custom data log.
Script Error: Failed to open file.	The defined file name already exists and the file is open or inacessible. Either close the file (and the new data will be logged in the prexisting log file), or define a unique file name (to create a new log file).
Duplicate/multiple log files exist after running a script to create a custom data log.	When using the custom data log feature, do not use `Log(on)` and `Log(off)` commands, and do not enable **Log Status / Measurements** from the **File** menu. This will create additional logs that save to the location specified in the **Status Tracing Settings** window.
Cannot see standard variables available for use in script.	The list of standard variables is generated (at the bottom of the Sequencer) once the script is run. The available variables will correspond to the devices referenced in the script.

Learn more about logging data in uProcess in the **uProcess User Manual**, or contact LabSmith.

For questions, please call +1 (925) 292-5161 or email sales@labsmith.com. LabSmith is a trademark of LabSmith, Inc. ©2023 LabSmith, Inc. 08/2023. All specifications are subject to change without notice.