Using Custom Chambers with the LI-8250 Multiplexer

Application Note

The LI-8250 Multiplexer can be connected to a user-built custom chamber. Digital communication between the LI-8250 Multiplexer and the custom chamber enables integration of data and affords the multiplexer some basic control over the chamber. Data integration allows custom chambers to take full advantage of the file structure features of the LI-8250 Multiplexer, trace gas analyzers compatibility, and the data processing tools of SoilFluxPro™ Software.

This application note outlines some general design considerations for closed transient chambers and details how to use the LI-8250 user interface. Some example scripts are provided for the Arduino IDE and Python to help facilitate communication. Example hardware configurations for connecting to the multiplexer's digital communication interface are also provided.

Design considerations

When designing a custom chamber for use with the LI-8250 Multiplexer, some key design considerations need to be addressed related to the chamber headspace concentration between measurements, chamber volume, and air mixing inside the chamber.

Chamber headspace

The headspace concentration is important because of the way the system is applied to measure soil gas flux. The LI-8250 Multiplexer is designed to use a closed transient method for flux measurement. Here, air is circulated in a fixed volume between the chamber and gas analyzer. As the sample interacts with the air in this volume, the gas concentration changes proportionally to the flux into or out of the sample and the size of the volume. In a closed system, the flux for a particular gas species is given by:

$$f = \frac{PV}{RTS}\frac{dc'}{dt} \qquad 1$$

Where f is the flux in mol m^{-2} s^{-1}, P is pressure in Pa, V is volume in m^3, R is the Ideal Gas Constant (~8.314 Pa m^3 k^{-1} mol^{-1}), T is temperature in K, and S is sample surface area in m^2. dc'/dt is the rate of change in the dry mixing ratio of the gas species of interest in mol mol^{-1} s^{-1}.

In the LI-8250 Multiplexer system, dc'/dt is calculated for the moment of chamber closure using a non-linear diffusion-based model that accounts for the progressive suppression of the diffusive flux while the chamber is closed. Where the flux is truly diffusive in nature, as the chamber headspace concentration changes during the measurement, the diffusive gradient between the chamber and the sample also changes and, in turn, changes the flux. For CO_2 emitted from soils, for example, the flux becomes progressively reduced as the chamber headspace concentration increases during the measurement. If the chamber is left closed long enough, the flux will eventually be reduced to zero. For this reason, a chamber must include some mechanism to return the chamber headspace to ambient conditions after an observation.

In a LI-COR long-term chamber, such as the 8200-104 Opaque Long-Term Chamber, the headspace is returned to ambient conditions after an observation by lifting the chamber bowl and rotating away from the collar. This ensures free air exchange between the chamber and the atmosphere between measurements.

The LI-8250 Multiplexer can trigger the opening and closing of a custom chamber timed with sampling. It is up to the user to design the actual mechanism the chamber will use to open and close.

Chamber volume

Chamber volume, or particularly the surface-area-to-volume ratio, should be carefully considered in the context of expected flux rates when designing a chamber. As chamber volume increases for a given surface area and a given flux, the observed dc'/dt will decrease. With a very large chamber relative to a very small flux, it is quite possible to drive dc'/dt well below the peak-to-peak noise for the analyzer used for measurements. In this case, long observations times will be required to make a measurement and uncertainty in the final computed flux will be high.

A simple sensitivity analysis using equation 1 above should be done to optimize the chamber surface-area-to-volume ratio. This should be based on the expected minimum and maximum fluxes and the precision of the gas analyzer. The goal is to ensure that for the selected chamber volume and measurement area, dc'/dt for the minimum expected flux is sufficiently large relative to the analyzer's peak-to-peak noise and that the chamber headspace concentration at the end of the observation for the largest expected flux is below the analyzer's maximum detection limit.

Air mixing

Volume and chamber shape are also important for turbulent air mixing within the chamber. A measurement inherently assumes that the gas sample pulled from the chamber and routed through the LI-8250 Multiplexer to the gas analyzer, is representative of the conditions in the chamber at the moment the sample is pulled. If the chamber volume is well mixed, this assumption is valid. The observed gas concentration will change predictably as a function of the sample flux rate.

If mixing is poor, the observed gas concentration can change independently of the sample flux. These changes may appear as oscillations or sudden steps in the observed gas concentration during the observation. Or they may simply cause erroneous measurements that represent the combined effects of the sample flux and diffusion into or out of pockets of air trapped in the chamber that are at a different concentration than the bulk chamber air.

In general, conditions are favorable for mixing when the chamber air is turbulent and few flow obstructions are present in the chamber. When flow from the LI-8250 Multiplexer drives turbulence in the chamber, smaller chamber volumes are more favorable for mixing. LI-COR chambers are bowl-shaped with a volume of approximately 5 liters. For larger or more irregularly shaped chambers, it may be necessary to include fans inside the chamber to aid in mixing.

Pneumatics

The LI-8250 Multiplexer and the long-term chamber cable assembly (part number: 9982-056) use quick-connect fittings to attach pneumatic lines. On this assembly, the male quick-connect fitting provides airflow into the chamber. See Table 4 on the facing page for recommended quick-connect fittings.

The flow through the chamber provided by the LI-8250 Multiplexer and the long-term chamber cable assembly is ~2.8 SLPM. The actual flow rate through the chamber flow loop is measured inside the multiplexer and is reported in the data file and user interface.

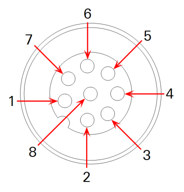
Power

The LI-8250 Multiplexer provides a regulated 24 VDC power supply via the power and communications port. The supply is electronically fused, limiting peak current to a single chamber at ~1.8 A, and is shared between all ports. We strongly recommended that you keep total steady-state power consumption for all connected custom chambers at or below 1 A. If the chamber(s) will require more power than this, you should provide a separate power supply to the chamber(s) directly. In this case, the multiplexer and chamber grounds should remain separate.

For custom chambers built off platforms with limited input voltage ranges, such as Arduino or Raspberry Pi, direct connection to the 24 VDC supply is not possible. In these cases, you will need to step down the voltage to an acceptable level (typically 5.0 or 3.3 VDC) using a DC-DC converter. For the best protection, use a DC-DC converter that provides ground isolation between the input and output sides and leave the input and output grounds unconnected from each other.

For connection between the chamber and the multiplexer, an 8-pin bulkhead to flying lead adapter is available (part number: 310-19090). This adapter provides a direct, watertight connection to the standard 15 m long-term chamber cable assembly. The pin assignment shown in Table 1 below represents the male end of the bulkhead.

Table 1. Pin assignment for the bulkhead flying lead that connects to a custom chamber.

	Connector Pin #	Chamber Signal Name	Bulkhead Wire Color
	1	RS422_TX-	White
	2	RS422_RX-	Brown
	3	RS422_RX+	Green
	4	+24VDC	Yellow
	5	Ground	Gray
	6	+24VDC	Pink
	7	RS422_TX+	Blue
	8	Ground	Red

Serial interface

The LI-8250 Multiplexer uses full-duplex RS-422 operating at 115,200 baud to communicate with the chamber. Few off the shelf platforms typically used in custom chambers offer native hardware support for RS-422. For these platforms, you will need an adapter to convert between RS-422 and a supported hardware communication interface.

There are several such adapters available through third-party suppliers that you can use to convert RS-422 to RS-232, USB, or TTL. We have tested adapters from CommFront (commfront.com) and Zihatec (hwhardsoft.de) and have verified compatibility with the LI-8250 Multiplexer and these adapters. See Table 2 and Table 3 below for the configuration and wiring of the Zihatec adapters.

For those interfacing to platforms such as Arduino or Raspberry Pi, an RS-422 to TTL adapter is recommended. The TTL output from the adapter should be connected to a physical serial UART on these devices. Note that for Arduino specifically, the software emulated serial ports (such as those provided by the AltSoftSerial library) are not fast enough at the LI-8250 Multiplexer's baud rate, leading to message corruption. For making a TTL connection to the platform's serial port, be mindful of the TTL voltage levels supplied by the adapter. Raspberry Pi for example, only supports a 3.3 VDC maximum input on its GPIO pins, whereas the high logic level for many TTL devices will be 5 VDC.

Table 2. DIP switch settings for the Zihatec RS-422/RS-485 Arduino Shield or Raspberry Pi Hat.

SW1		SW2		SW3	
Channel	Setting	Channel	Setting	Channel	Setting
1	On	1	On	1	On
2	Off	2	On	2	Off
3	On	3	Off	3	Off
4	Off	4	Off	4	Off

Table 3. Bulkhead to flying lead (part number: 310-19090) to Zihatec RS-422/RS-485 Arduino Shield or Raspberry Pi Hat connections.

Input Terminal Strip Pin	Flying Lead Wire Color
B	Brown
A	Green
Z	White
Y	Blue
Shield	Red and gray if not used to power the chamber, empty otherwise

Custom chamber control kit

Part number: 8200-402

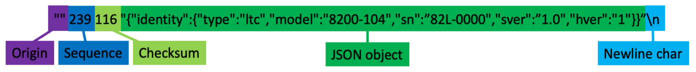
LI-COR offers a custom chamber control kit that can simplify integrating your custom chamber with the LI-8250 Multiplexer.

Table 4. Parts included in the custom chamber control kit.

Part number	Name	Description	# per chamber
310-19090	Bulkhead flying lead for custom chambers	An 8-pin Turk bulkhead with flying leads used to connect a custom chamber to the multiplexer for power and communication	1
9982-056	Long-term chamber cable assembly	Includes a 15 m RS-422 to RS-422 cable and two 15 m lengths of Bev-A-Line® tubing with quick-connect fittings. Provides power, communication, and gas flow between the multiplexer and the long-term chamber.	1
300-07126	Quick-connect bulkhead (female)	Connects a custom chamber to the male end of the Bev-A-Line® tubing on the long-term chamber cable assembly.	1
300-07127	Quick-connect bulkhead (male)	Connects a custom chamber to the female end of the Bev-A-Line® tubing on the long-term chamber cable assembly.	1
167-07256	Seal washer	Used to create a seal between the quick-connect bulkheads and the panel they are inserted through.	2

Messaging

Messages sent between the LI-8250 Multiplexer and a chamber have a five-part structure. Each message begins with an origin identifier, followed by a sequence number, checksum, and JSON object. All messages are terminated with a newline character (ASCII 10).



```
"" 239 116 "{"identity":{"type":"ltc","model":"8200-104","sn":"82L-0000","sver":"1.0","hver":"1"}}"\n
```

Origin

The origin defines the source of the message. For messages originating from a chamber, the origin will always be null (""). For messages arising from the LI-8250 Multiplexer, general communication on the port will have a null origin. Port specific commands will include the port number in the origin. Where a non-null origin is sent, the custom chamber does not need to do anything with it. Though this information does tell the chamber what port it is connected to on the multiplexer.

Sequence

The sequence is an integer (-1 or 1 to 32767) used to track messages. For each message received with a sequence number greater than zero, an acknowledgment should be returned by the receiving device. For the example above sent by an 8200-104 Opaque Long-Term Chamber, the multiplexer would respond with:

```
"" 239 -1 "{"ack":""}"
```

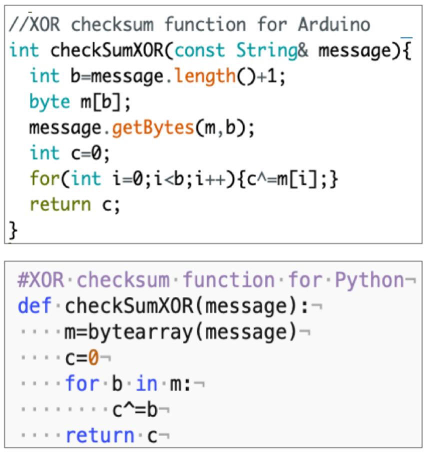
This acknowledges receipt of message 239 from the chamber. For messages using a sequence of -1, no acknowledgment is expected.

Checksum

The checksum is a bitwise XOR of the JSON object. For acknowledged messages, it should always be included. The checksum is calculated by the receiving device (LI-8250 Multiplexer or chamber) and compared to the checksum in the original message. This ensures the message has not been corrupted. For acknowledged messages where the checksums do not match, a non-acknowledgement would be sent:

```
"" 239 -1 "{"nak":""}"
```

Example functions for computing the checksum using Python and the Arduino IDE are given in Figure 1 below. These functions accept the JSON object as a string and return the checksum as an integer value.



```
//XOR checksum function for Arduino
int checkSumXOR(const String& message){
  int b=message.length()+1;
  byte m[b];
  message.getBytes(m,b);
  int c=0;
  for(int i=0;i<b;i++){c^=m[i];}
  return c;
}
```

```
#XOR checksum function for Python
def checkSumXOR(message):
    m=bytearray(message)
    c=0
    for b in m:
        c^=b
    return c
```

Figure 1. Example functions for computing the checksum using Python and the Arduino IDE.

JSON object

The JSON object contains the data items passed between the chamber and LI-8250 Multiplexer. Each data item is composed of a key:value pair. For a custom chamber, there are four general message types expected from the chamber with specific JSON objects associated with each.

- Acknowledgments as described above
- Identity messages containing information about the chamber
- Status messages containing the chamber state
- Data messages containing measurements made by the chamber

Identity messages are sent in response to an identity request from the LI-8250 Multiplexer. The identity message consists of an identity object, containing type, model, serial number (`sn`), and software version (`sver`). The type for a custom chamber will always be the string `dcc`. Model, serial number, and software version are all free-form strings. Other data items can be included in the identity message if desired but are not used by the multiplexer. The request for a chamber identity by the multiplexer is an unacknowledged message:

```
"" -1 -1 "{"identify":""}"
```

Once received, the chamber should respond with its identity as in the example below:

```
"" 78 53 "{"identity":{"model":"User_
Chamber","type":"dcc","sn":"UC-
01","sver":"0.1"}}"
```

Status messages should be issued by the chamber with each identity message, when the chamber initiates a move operation, and upon reaching a new position. Status messages contain three objects: the model and serial number (`sn`), the chamber state (`chamber_status`), and a diagnostic value (`diag_code`).

- Chamber state is a string where valid values are open, opening, closed, closing, and unknown.
- Model and serial number are the same as they appear in the identity.
- Diagnostic value is a bit field containing error flags.

A diagnostic value of 0 indicates normal operation. Other values indicate an error. The bits in Table 5 on the facing page are those used by the 8200-104/C Long-Term Chambers and are interpreted in the same way by the LI-8250 Multiplexer for a user-built custom chamber. Additional bits can be set in the diagnostic code and are defined by the user.

Table 5. Diagnostic code (`diag_code`) bit fields used by the 8200-104/C Long-Term Chambers. These can be used to indicate errors from a user-built custom chamber, but they will be interpreted by the multiplexer following the error description here.

Bit	Error description
1	Message error
2	Motor error, motor does not move or stalled during a move
4	Issue with EEPROM
8	Issue with SDI-12 sensor or its configuration
16	Issue with light sensor or its configuration
32	Problem with chamber temperature sensor
64	Problem with temperature sensor on chamber control board
128	Input voltage issue
256	Fatal error occurred

Here is an example status message indicating that the chamber is currently open:

```
"" 77 53 "{"type":"dcc","sn":"UC-01","chamber_status":"open","diag_code":0}"
```

The LI-8250 Multiplexer sets the chamber status at the start and end of an observation and when using the manual controls in the LI-8250 user interface. The multiplexer sends a message containing a chamber object with the new desired chamber state as a string. Here is an example message requesting the user chamber to open:

```
"" 1002 90 "{"chamber":"open"}"
```

or to close:

```
"" 1003 56 "{"chamber":"close"}"
```

The chamber sends data messages continuously during an observation and when you set the port to active in the LI-8250 user interface. The multiplexer will send a message requesting the chamber to start and stop sending data. These messages are port-specific and, as such, include a non-null origin value. To request the start of data messages, the multiplexer sends:

```
"1" 1004 54 "{"measurement":"start"}"
```

or to request the end of data messages:

```
"1" 1005 78 "{"measurement":"stop"}"
```

The data message itself contains three objects: data, source, and a diagnostic code (`diag_code`). Diagnostic code is analogous to that used in the status message.

The source object includes a type and serial number (`sn`), as in the identity message. The data object, at a minimum, must contain the temperature measured in the user chamber (`temperature`) since this is required for flux calculations. The data object can include other measurements as desired. All values reported for measurements in the data object must be interpretable as floating point numbers. Mapping to associate user-specific keys with appropriate units and variable names is done in the LI-8250 Multiplexer configuration described later. Here is an example data message:

```
"" 79 96 "{"data":{"temperature":24.1},"source":{"type":"dcc","sn":"UC-01"},"diag_code":0}"
```

Tools exist in Python and the Arduino IDE for constructing and parsing JSON objects. For Python, the standard json library includes manipulation of JSON. For the Arduino IDE, the third-party library ArduinoJson (arduinojson.org) provides an efficient implementation. Both environments also provide built-in functions for serial communications.

The examples in Listing 1-1 below and Listing 1-2 on the next page show how to serialize and deserialize JSON in both environments and how to publish a response to an identity request.

Note: The code for these scripts may be copied from below or downloaded from licor.com/documents/9t2om4h11d6rg4xuus3la3d293yfkayr.

Listing 1-1. An example of the Python script.

```python
#Example Python script for publishing identity responses to the LI-8250
import json
import serial

ser=serial.Serial('/dev/serial0', 115200)
sequence=0

def checkSumXOR(message):
    m=bytearray(message)
    c=0
    for b in m:
        c^=b
    return c

def publishResponse(message):
    message= '"'+message+'"'
    global sequence
    sequence+=1
    if sequence>32767:
        sequence=1
    ser.write(bytes('"" '+str(sequence)+' '+str(checkSumXOR(message))+' '+message+'\n'))
    return True

while True:
    if ser.inWaiting()>0:
        request=ser.readline()
        j=json.loads(request[request.find('{'):-2])
        if 'identify' in j:
            publishResponse(json.dumps({'identity':{'type':'dcc','model':'User_Chamber','sn':'UC-01','sver':'0.1'}}))
```

Note that in the Arduino IDE example, the choice to declare the buffers for working with the JSON objects inside a function separate from the main loop is deliberate and stems from how these buffers are handled by ArduinoJson. Please refer to the library's documentation for more details.

Listing 1-2. An example of the Arduino IDE script.

```
//Example Arduino script for publishing identity responses to the LI-8250
#include "ArduinoJson-v5.13.3.h"
int sequence=0;

int checkSumXOR(const String& message){
  int b=message.length()+1;
  byte m[b];
  message.getBytes(m,b);
  int c=0;
  for(int i=0;i<b;i++){c^=m[i];}
  return c;
}

bool publishResponse(const String& message){
  sequence++;
  if(sequence<0){sequence=1;}
  Serial.println("\"\" "+String(sequence)+" "+String(checkSumXOR(message))+" \""+message+"\"");
  return true;
}

void requestHandler(const String& message){
  StaticJsonBuffer<200> recieveBuffer;
  JsonObject& mux_json = recieveBuffer.parseObject(message.substring(message.indexOf("{"),message.lastIndexOf("}")+1));
  if(mux_json.success()){
    if(mux_json.containsKey("identify")){
      StaticJsonBuffer<100> sendBuffer;
      JsonObject& root = sendBuffer.createObject();
      JsonObject& id = root.createNestedObject("identity");
      root["identity"]["type"].set("dcc");
      root["identity"]["model"].set("User_Chamber");
      root["identity"]["sn"].set("UC-01");
      root["identity"]["sver"].set("0.1");
      String jstr;
      root.printTo(jstr);
      publishResponse(jstr);
      sendBuffer.clear();
    }
  }
  recieveBuffer.clear();
}

void setup() {
  Serial.begin(115200);
}

void loop() {
  if(Serial.available()){requestHandler(Serial.readStringUntil('\n'));}
}
```

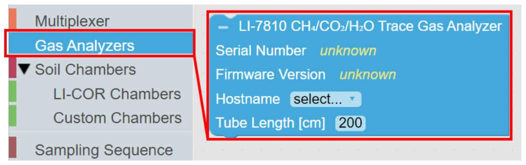
Configuring a custom chamber

The LI-8250 Multiplexer is capable of communicating with custom chambers using the Configuration page of the user interface. For the most part, configuring a custom chamber is identical to configuring a LI-COR long-term chamber. Those steps are described in your LI-8250 Multiplexer manual. Where custom chambers differ is that they use the **Digital Custom Chamber** block and require different inputs to this block.

When you first open the **Configuration** page, you will see two empty root blocks: the **LI-8250 Multiplexer** block and the **Sampling Sequence** block. This section will cover the steps needed to configure the elements under the LI-8250 Multiplexer block. For details on how to configure the Sampling Sequence block, see your LI-8250 Multiplexer manual.

Add a gas analyzer

If you are using a LI-COR gas analyzer, the first step is to add the gas analyzer(s) your system is using. Select **Gas Analyzers** from the Toolbox to open the drawer. Then find one of the gas analyzers you will be using and click to add it. In this case, we added an LI-7810 $CH_4/CO_2/H_2O$ Trace Gas Analyzer. Repeat this for each analyzer on the system.



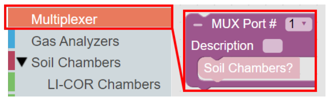
Serial Number unknown
Firmware Version unknown
Hostname select...
Tube Length [cm] 200

All gas analyzer blocks are nested under the LI-8250 Multiplexer block. If you are using a LI-COR cable assembly for the Trace Gas Analyzers (part number: 9982-011) or the LI-870 CO_2/H_2O Analyzer (part number: 9982-010), the tubing length is already entered. If you have multiple gas analyzers connected using the T-split tubing (part number: 9982-073), you will need to add 15 cm to the tubing length of each analyzer.

Note: If you are using an LI-870 CO_2/H_2O Analyzer or have added the gas analyzer to the **Device Network** previously, the analyzer will automatically be added to the LI-8250 Multiplexer block when you open the Configuration page.

Add a port

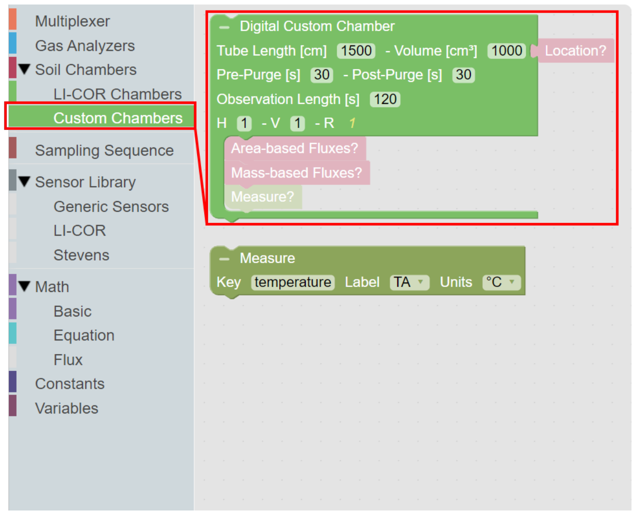
Before you can add a chamber, you will need to add the port that the chamber is connected to. To do this, select **Multiplexer** from the Toolbox, then click and drag **MUX Port #** to place the block under the LI-8250 Multiplexer block.



You can assign a port number or the user interface will automatically assign a port number as these blocks are added. You can also provide a description of the devices on that port that will appear in the data file.

Add a custom chamber

After you have added a port, you can add a custom chamber to that port. To add a custom chamber, expand the **Soil Chambers** drop-down in the Toolbox and select **Custom Chambers**. Then click and drag the **Digital Custom Chamber** block under the MUX Port # block. Repeat this for each chamber on each port.



Digital Custom Chamber
Tube Length [cm] 1500 - Volume [cm³] 1000 Location?
Pre-Purge [s] 30 - Post-Purge [s] 30
Observation Length [s] 120
H 1 - V 1 - R 1
Area-based Fluxes?
Mass-based Fluxes?
Measure?

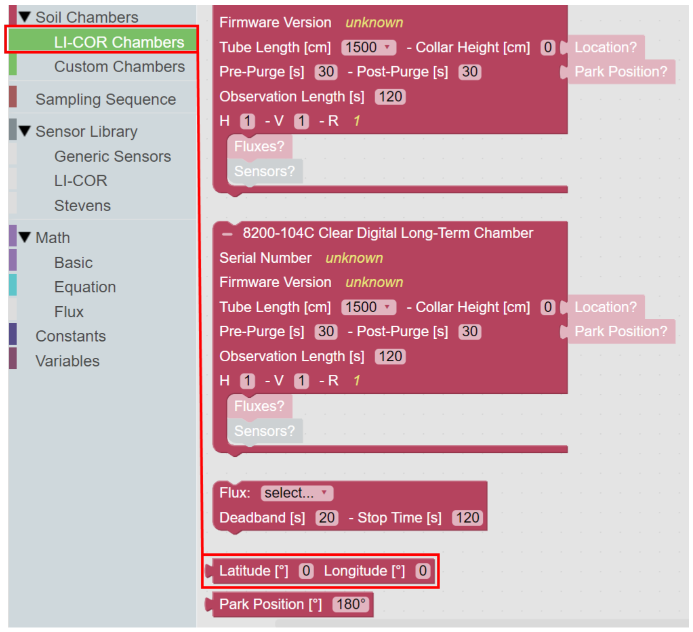
Measure
Key temperature Label TA Units °C

Each custom chamber block allows you to change several parameters. If you are using the long-term chamber cable assembly from LI-COR (part number: 9982-056), the **Tube Length** of 1500 cm is correct. You can also adjust the **Volume**, **Pre-Purge**, **Post-Purge**, and **Observation Length** according to your needs. Volume is the volume of the chamber and is an important parameter used in flux calculations.

The custom chamber block also allows you to enter the location of the chamber under the **Location** block.

Note: If a location is not defined, the multiplexer will use the GPS location of the LI-8250 Multiplexer as the chamber location.

To add the Location block, select **LI-COR Chambers** in the Toolbox and find the **Latitude/Longitude** block. Then click and drag the Latitude/Longitude block to the chamber block.



Firmware Version unknown
Tube Length [cm] 1500 - Collar Height [cm] 0 Location?
Pre-Purge [s] 30 - Post-Purge [s] 30 Park Position?
Observation Length [s] 120
H 1 - V 1 - R 1
Fluxes?
Sensors?

8200-104C Clear Digital Long-Term Chamber
Serial Number unknown
Firmware Version unknown
Tube Length [cm] 1500 - Collar Height [cm] 0 Location?
Pre-Purge [s] 30 - Post-Purge [s] 30 Park Position?
Observation Length [s] 120
H 1 - V 1 - R 1
Fluxes?
Sensors?

Flux: select...
Deadband [s] 20 - Stop Time [s] 120

Latitude [°] 0 Longitude [°] 0

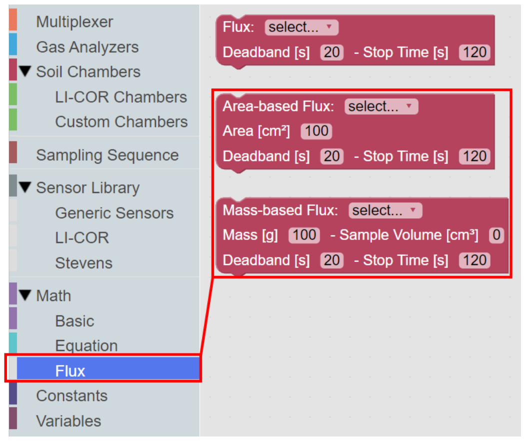
Park Position [°] 180°

Add a flux

Two different flux calculations are available to be added to a custom chamber configuration: Area-based Flux and Mass-based Flux. The Area-based Flux is used to specify a flux calculated on a per area basis and the Mass-based Flux is used to specify a flux calculation on a per mass basis.

To add a flux block to your custom chamber block, expand the **Math** drop-down and select **Flux** from the Toolbox. From the Flux drawer click and drag either the **Area-based Flux** block or the **Mass-based Flux** block under the chamber you would like to add the flux to. You can have multiple Flux blocks under each chamber depending on the analyzer(s) you are using and the gases you would like recorded.

Note: Only one type of flux block (area or mass) may be placed inside a Digital Custom Chamber block. Multiple flux blocks may be placed under the Digital Custom Chamber block to accommodate different gas species, but they must be of the same type.



Flux: select...
Deadband [s] 20 - Stop Time [s] 120

Area-based Flux: select...
Area [cm²] 100
Deadband [s] 20 - Stop Time [s] 120

Mass-based Flux: select...
Mass [g] 100 - Sample Volume [cm³] 0
Deadband [s] 20 - Stop Time [s] 120

Each flux block allows you to customize additional parameters required for the flux calculation.

Area-based flux

The drop-down menu allows you to select the type of gas and the source for flux calculations. Under **Area** you will enter the sample surface area in cm^2. You can also adjust the **Deadband** and **Stop Time** as needed. The Deadband and Stop Time are parts of the total **Observation Length**. Stop Time can be any time desired so long as it is beyond the Deadband and does not exceed the Observation Length.

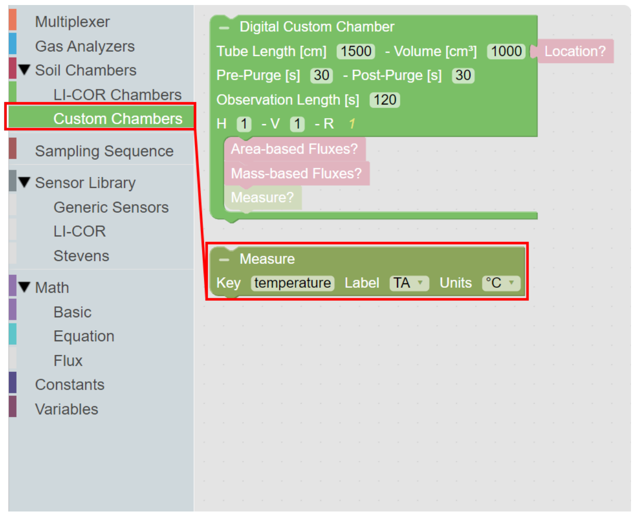
Mass-based flux

The drop-down menu allows you to select the type of gas and the source for flux calculations. Under **Mass** you will enter the sample mass in grams. Under **Sample Volume** you will enter the calculated volume of the sample in cm^3. This is used to correct the total volume for sample displacement. You can also adjust the **Deadband** and **Stop Time** as needed. The Deadband and Stop Time are parts of the total **Observation Length**. Stop Time can be any time desired so long as it is beyond the Deadband and does not exceed the Observation Length.

Note: You will need one Flux block for each gas flux you would like to have computed. Without a Flux block, no flux will be calculated for that gas. If a Flux block was not added for a gas, a flux can still be calculated later using SoilFluxPro Software.

Add a measured variable

The LI-8250 Multiplexer can include measured variables from a custom chamber. This block defines an expected measurement in each data message sent by the chamber. To add a measured variable, expand the **Soil Chambers** drop-down in the Toolbox and select **Custom Chambers**. Then click and drag the **Measure** block under the custom chamber you would like to add the variable to.



You can add a variable of your own using **Create variable**. You must include an air temperature measurement for the configuration to work. If temperature is not included, flux calculations will not be performed by the LI-8250 Multiplexer. Multiple measured variable blocks can be added under one custom chamber block. Within this block:

- **Key** identifies the key used for the measured variable in the data message's JSON object.
- **Label** specifies the data label used for this measured variable in the .82z file.
- **Unit** provides a selection of supported units used for this measured variable.

Resources

The resources below can provide you with more details about using your digital custom chamber with the LI-8250 Multiplexer.

- LI-8250 Multiplexer manual: licor.com/8250manual
- LI-8250 Multiplexer support: licor.com/8250support
- Support: licor.com/env/support

LI-COR

LI-COR Biosciences
4647 Superior Street
Lincoln, Nebraska 68504
Phone: +1-402-467-3576
Toll free: 800-447-3576 (U.S. and Canada)
envsales@licor.com

LI-COR Distributor Network:
www.licor.com/env/distributors

Regional Offices

LI-COR Biosciences GmbH
Siemensstraße 25A
61352 Bad Homburg
Germany
Phone: +49 (0) 6172 17 17 771
envsales-gmbh@licor.com

LI-COR Biosciences UK Ltd.
St. John's Innovation Centre
Cowley Road
Cambridge
CB4 0WS
United Kingdom
Phone: +44 (0) 1223 422102
envsales-UK@licor.com

Copyright © 2021 LI-COR, Inc. All rights reserved.
979-19673 05/2021