

UC20-WL2000-AC

Application Note for data storage and logging to SD card

Abstract:

This document describes a way how a user can have write access to a SD card inside u-control web. Possible applications could be a data logging in case of no connectivity to the internet or another network member. The implementation is done in Node-RED.

Application Note for data storage and logging to SD card

Hardware reference

No.	Component name	Article No.	Hardware / Firmware version
1	UC20-WL2000-AC	1334950000	-

Software reference

No.	Software name	Article No.	Software version
1	u-create web	-	1.0.0 (or higher)
2	Node-RED	-	1.0.0 (or higher)

File reference

No.	Name	Description	Version
1	AN0010-UC20-Data storage and logging to SD card.zip	The archive contains .json files related to this application note	-

Contact

Weidmüller Interface GmbH & Co. KG
Klingenbergstraße 26
32758 Detmold, Germany
www.weidmueller.com

For any further support please contact your
local sales representative:
<https://www.weidmueller.com/countries>

Content

1	Warning and Disclaimer.....	4
2	General Information	5
3	Example: Node-RED Logging.....	6
4	Example: Remanent Data Storage	11
4.1	Example 1: Save remanently to Data Storage (DS)	11
4.2	Example 2: Save remanently to SD card	16

1 Warning and Disclaimer

Warning

Controls may fail in unsafe operating conditions, causing uncontrolled operation of the controlled devices. Such hazardous events can result in death and / or serious injury and / or property damage. Therefore, there must be safety equipment provided / electrical safety design or other redundant safety features that are independent from the automation system.

Disclaimer

This Application Note / Quick Start Guide / Example Program does not relieve you of the obligation to handle it safely during use, installation, operation and maintenance. Each user is responsible for the correct operation of his control system. By using this Application Note / Quick Start Guide / Example Program prepared by Weidmüller, you accept that Weidmüller cannot be held liable for any damage to property and / or personal injury that may occur because of the use.

Note

The given descriptions and examples do not represent any customer-specific solutions, they are simply intended to help for typical tasks. The user is responsible for the proper operation of the described products. Application notes / Quick Start Guides / Example Programs are not binding and do not claim to be complete in terms of configuration as well as any contingencies. By using this Application Note / Quick Start Guide / Example Program, you acknowledge that we cannot be held liable for any damages beyond the described liability regime. We reserve the right to make changes to this application note / quick start guide / example at any time without notice. In case of discrepancies between the proposals Application Notes / Quick Start Guides / Program Examples and other Weidmüller publications, like manuals, such contents have always more priority to the examples. We assume no liability for the information contained in this document. Our liability, for whatever legal reason, for damages caused using the examples, instructions, programs, project planning and performance data, etc. described in this Application Note / Quick Start Guide / Example is excluded.

Security notes

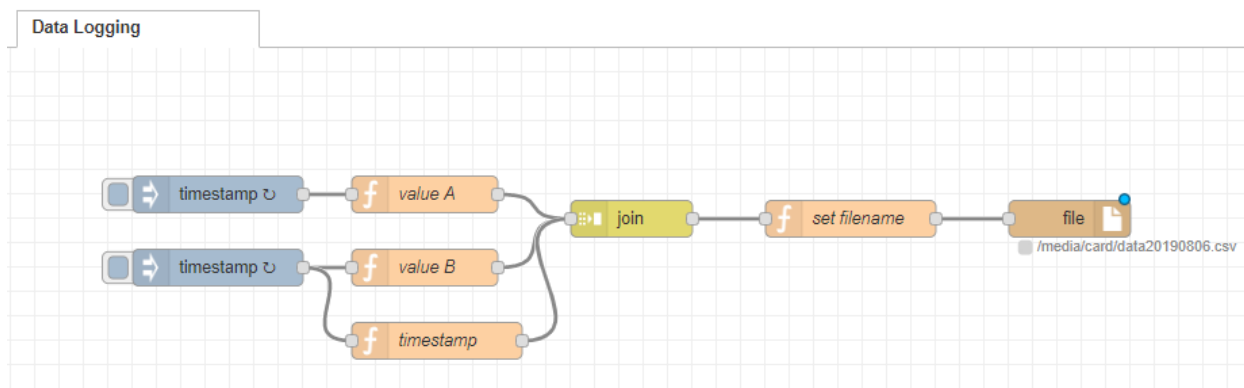
In order to protect equipment, systems, machines and networks against cyber threats, it is necessary to implement (and maintain) a complete state-of-the-art industrial security concept. The customer is responsible for preventing unauthorized access to his equipment, systems, machines and networks. Systems, machines and components should only be connected to the corporate network or the Internet if necessary and appropriate safeguards (such as firewalls and network segmentation) have been taken.


2 General Information

- The SD card must be less or equal than 32GB.
- The absolute path to the SD card on the file system inside the u-control is:
 - Firmware Version $\leq 1.2 \rightarrow$ **/media/card/**.
 - Firmware Version $\geq 1.3 \rightarrow$ **/run/media/mmcblk0p1**.
- Beside **File** nodes in Node-RED there is no other way to access the data on the SD card.

3 Example: Node-RED Logging


The example flow does not write serious data to the log file. It sets value A to 0.005 and value B to 5000, adds a timestamp and joins everything to one msg-object which gets a filename-attribute and is finally written to the SD card.

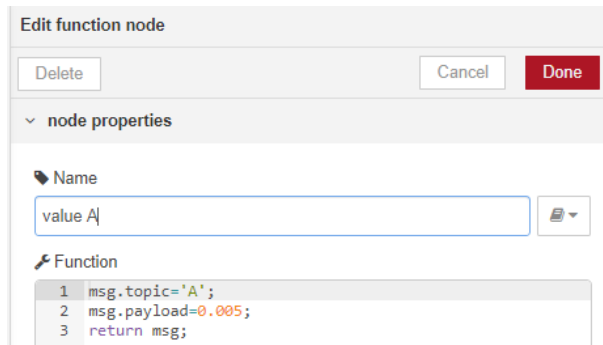


Let's have a look inside the inject node . It sends a timestamp every second. Both inject-nodes behave like this.

The "Edit inject node" window is shown. It has a title bar "Edit inject node" and three buttons: "Delete", "Cancel", and "Done". Below the buttons is a section "node properties" with a dropdown arrow. Under "node properties", there are several fields: "Payload" with a dropdown menu showing "timestamp", "Topic" with an empty text field, "Inject once after" with a checkbox, a text field "0.1", and the text "seconds, then", "Repeat" with a dropdown menu showing "interval", and "every" with a text field "1", a spinner, and a dropdown menu showing "seconds".

Application Note for data storage and logging to SD card

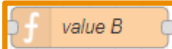
Now we have a look at the function node **value A** . Here the attribute topic is assigned to **A** and payload is assigned to **0.005**. In the end the **msg** is returned.

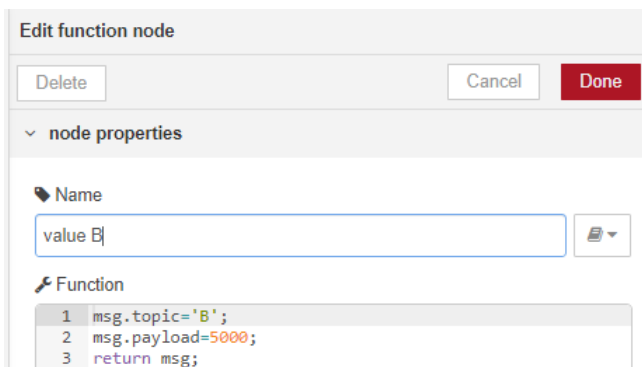


The screenshot shows the 'Edit function node' dialog box. At the top, there are 'Delete', 'Cancel', and 'Done' buttons. Under the 'node properties' section, the 'Name' field contains 'value A'. The 'Function' section contains a code editor with the following code:

```
1 msg.topic='A';  
2 msg.payload=0.005;  
3 return msg;
```

```
msg.topic='A';  
msg.payload=0.005;  
return msg;
```


The function node **value B**  behaves similar to **value A**. The value for topic and payload is here **B** and **5000**.

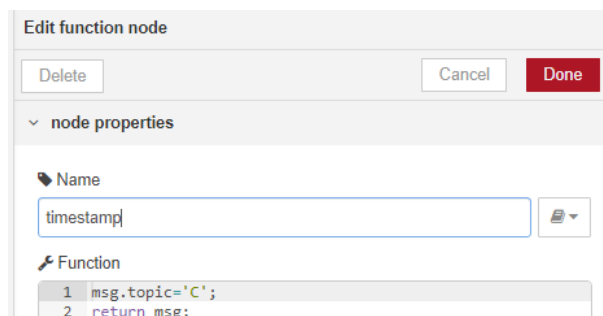


The screenshot shows the 'Edit function node' dialog box. At the top, there are 'Delete', 'Cancel', and 'Done' buttons. Under the 'node properties' section, the 'Name' field contains 'value B'. The 'Function' section contains a code editor with the following code:

```
1 msg.topic='B';  
2 msg.payload=5000;  
3 return msg;
```

```
msg.topic='B';  
msg.payload=5000;  
return msg;
```

The function node **timestamp**  just redirects the msg object, which includes a timestamp generated by the inject node.




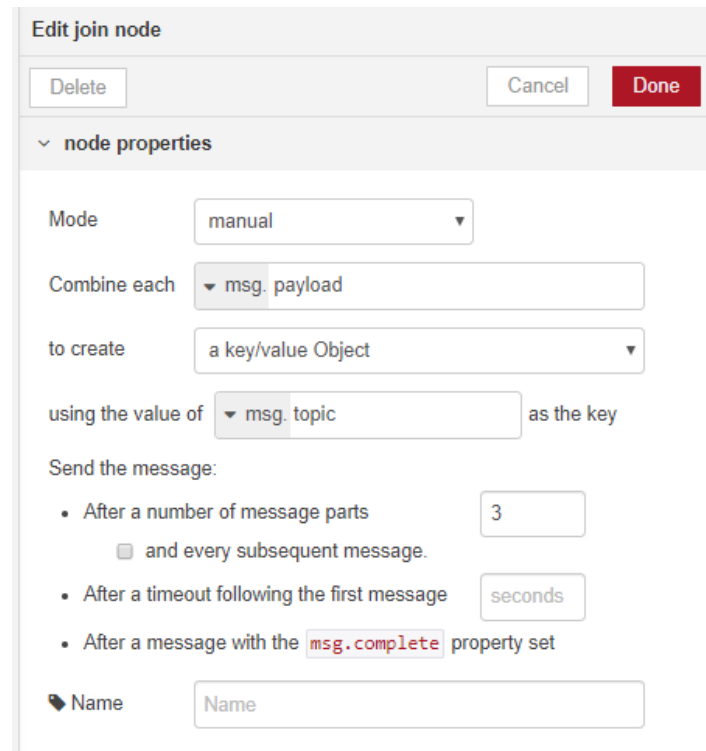
The screenshot shows the 'Edit function node' dialog box. At the top, there are 'Delete', 'Cancel', and 'Done' buttons. Under the 'node properties' section, the 'Name' field contains 'timestamp'. The 'Function' section contains a code editor with the following code:

```
1 msg.topic='C';  
2 return msg;
```

```
msg.topic='C';  
return msg;
```

Application Note for data storage and logging to SD card

With the **join** node  it is possible to restructure the msg object and combine all topics to one single object. The entries of this object can be accessed by keys.




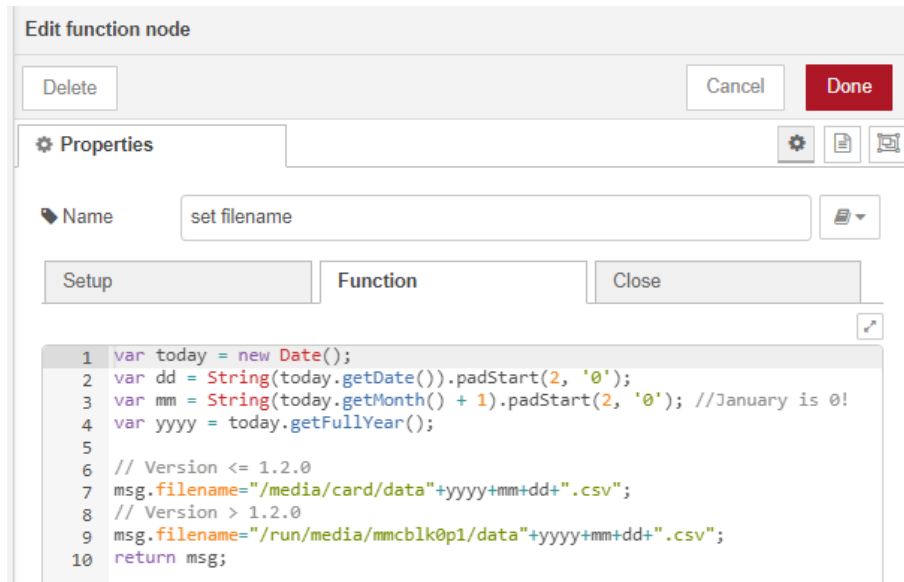
Combine each `msg.payload` object to create a **key/value Object** using the value of `msg.topic`. After 3 message input parts the object should be created. This is (partially) related to the amount of transitions we have connected to this node.

Output:

```
▼payload: object
  A: 0.005
  B: 5000
  C: 1565113787426
```


Application Note for data storage and logging to SD card

Next is the function node **set filename** . Here the filename is set. But also, an implicit rule is defined which regulates how often a new log file is created.



```
var today = new Date();
var dd = String(today.getDate()).padStart(2, '0');
var mm = String(today.getMonth()+1).padStart(2, '0'); //January is 0!
var yyyy = today.getFullYear();

msg.filename = "/run/media/mmcblk0p1/data"+yyyy+mm+dd+".csv";
return msg;
```

First the date is stored to a variable. Out of this date the information about year, month, day, hour, minute, second and millisecond can be extracted. This information is used to generate the filename.

Application Note for data storage and logging to SD card

The **file** node  is simple to configure.

Here the setup means that the data is appended to the end of the file and the entries are done line wise.



If the following error message appears. Double check if the SD card is inside your u-control.

```
8.8.2019, 16:06:19 node: 6662870c.f943d8
msg : string[95]
"failed to append to file: Error:
EACCES: permission denied, open
'/media/card/data20190806.csv'"
```

4 Example: Remanent Data Storage

There are multiple ways to store data remanently if necessary.

1) Data storage (DS) in the web PLC editor



Attention! Writing to variables that store data in the data storage degrades the memory. Do not write to these variables too often.

2) SD card

If the first way is chosen, please be aware that older hardware versions of u-control web store the data currently into the internal flash memory. Every write-action reduces the lifetime of the flash memory. Writing too often/frequently to this memory can cause a malfunction of the u-control web. This relates to all u-control web which have no additional remanent memory block on board.

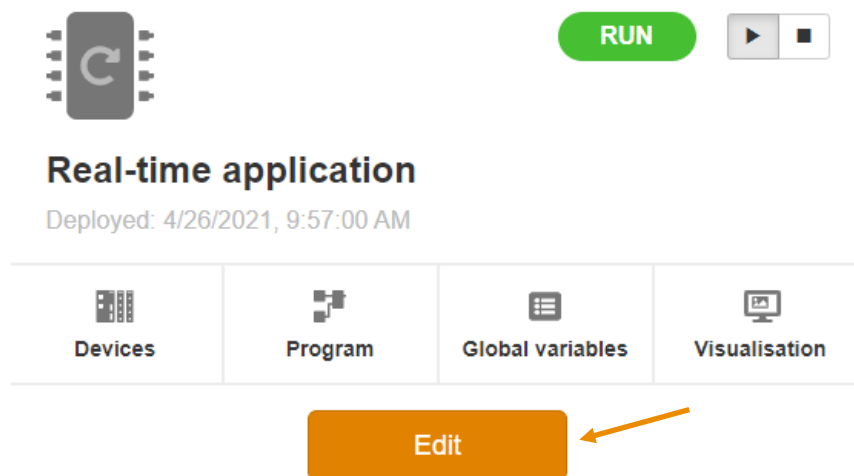


In case of any doubts please ask your corresponding sales contact before choosing this option.

Writing data to the SD card is another possibility to save data remanently.

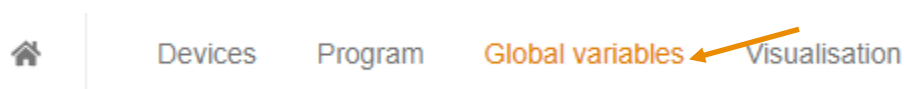
4.1 Example 1: Save remanently to Data Storage (DS)

- Enter the homepage of u-create web. Click **Edit**.



- Navigate to the **Global variables** tab, create a global variable for triggering a write action to the DS variable. Here G_xWriteDS of type BOOL.

Application Note for data storage and logging to SD card



- Click **+** to add a global variable.



The variable may be connected (mapped) to a digital input which is considered to trigger the data storage write operation.

Application example: if an uninterruptable power supply unit gives a feedback that the system shuts down in a few seconds.

1 variable(s) selected				Filter	+	📁	🗑️
<input checked="" type="checkbox"/> Name ↑	Data type	Initial value	Mapping				
<input checked="" type="checkbox"/> G_xWriteDS	BOOL	0					

- Navigate to the web PLC editor by clicking **Program** tab.



- Click **Data storage (DS)** and create a variable within the next dialog. Here e.g. DS_intRemanentVar of type INT.

Explorer

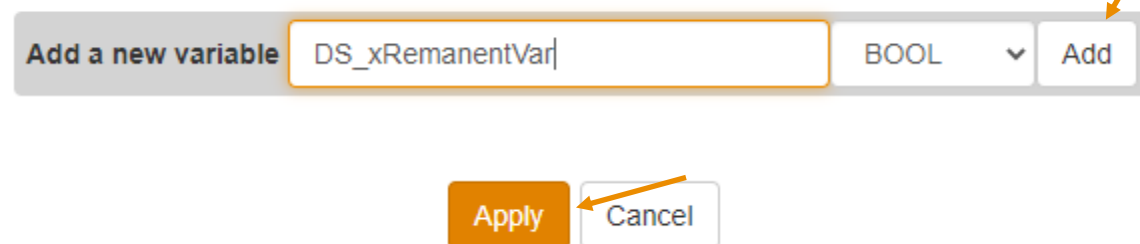
PLC_PROG

Data storage (DS)

Application Note for data storage and logging to SD card

- Click **Add** and **Apply**, then **Deploy** the project.

Variables in the data storage



Add a new variable

DS_xRemanentVar

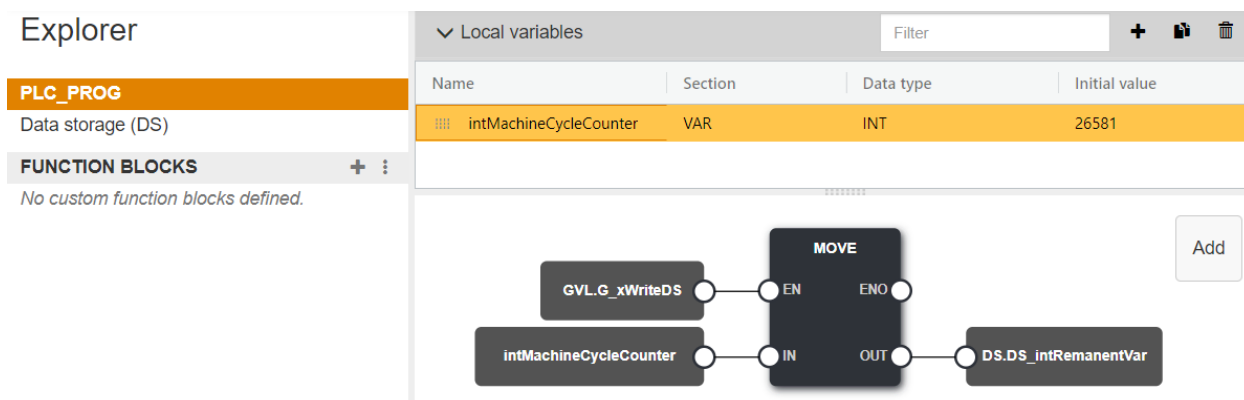
BOOL

Add

Apply

Cancel

- Create a simple PLC project consisting of a **MOVE** block which assigns a constant value to the DS variable once the trigger G_xWriteDS happens and deploy it.



⌚ Deploying application ...



⌚ The application has been deployed.



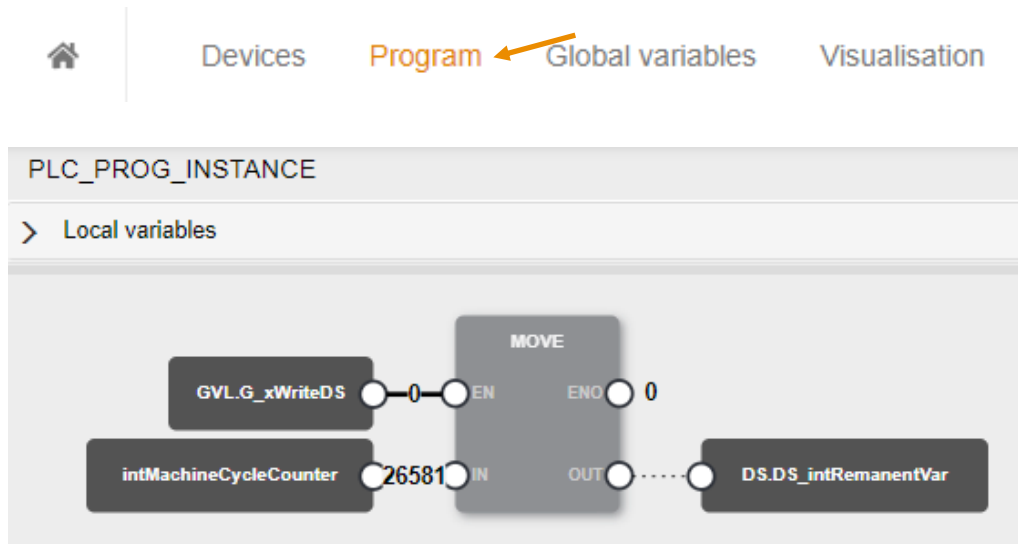
Start

Close

Application Note for data storage and logging to SD card

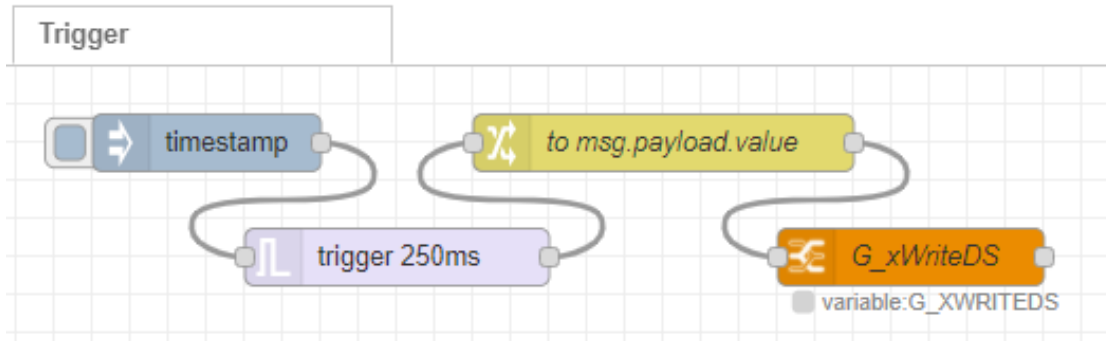
After clicking **Start** the browser opens the monitoring view in a new tab. Here we check whether the write operation works or not.

- Switch to **Program** again.



Application Note for data storage and logging to SD card

We are triggering with a simple flow implemented in Node-RED. This flow consists of an **inject**, one **trigger**, a **change** and an **io-data-out** node. After manually triggered, the **trigger** node sends immediately a message with **true** to the G_xWriteDS variable. After 250ms it sends again automatically a **false** to reset the data storage write operation.

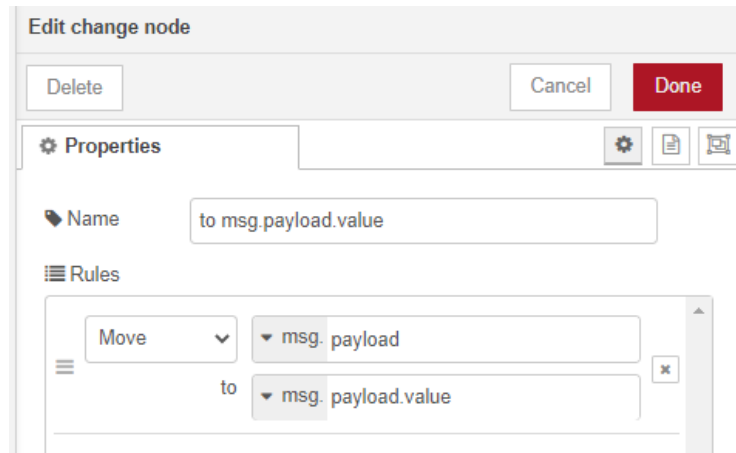


This is how the **trigger** node is configured:

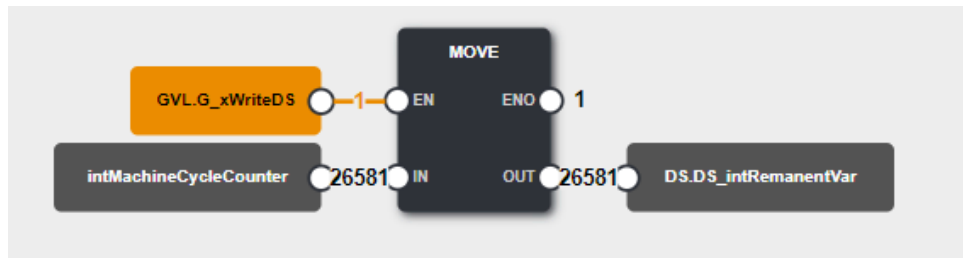
The 'Edit trigger node' window shows the configuration for the trigger node. It includes a 'Delete' button, 'Cancel' and 'Done' buttons, and a 'Properties' tab. The configuration is as follows:

- Send:** true
- then:** wait for
- Delay:** 250 Milliseconds
- extend delay if new message arrives:** ☐
- then send:** false
- send second message to separate output:** ☐
- Reset the trigger if:**
 - msg.reset is set
 - msg.payload equals optional
- Handling:** all messages
- Name:** Name

This is how the **change** node is configured:

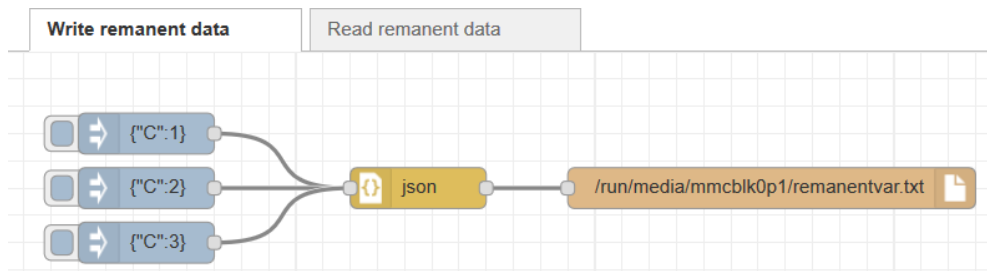


Triggering the **inject** node, causes a write operation, you will possibly not see the highlighting when using a time of 250ms inside the **trigger** node. Experimentally you can expand this timespan to 2s, but do not forget to change this value back to e.g. 50ms:



















4.2 Example 2: Save remanently to SD card

This example shows writing and reading a remanent variable to and from the SD card. The example **write remanent data flow** shows a simple possibility to write three different values to the SD card.

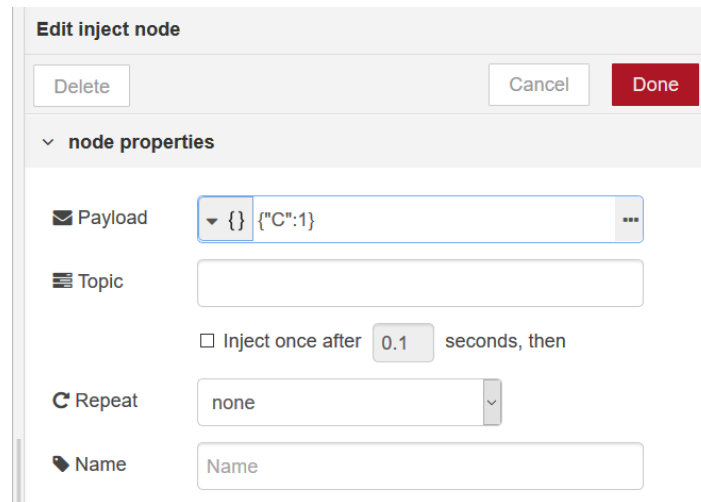


The three inject nodes set the value of the variable C to 1,2 or 3.


Let us have a look inside the inject nodes. When pressed                

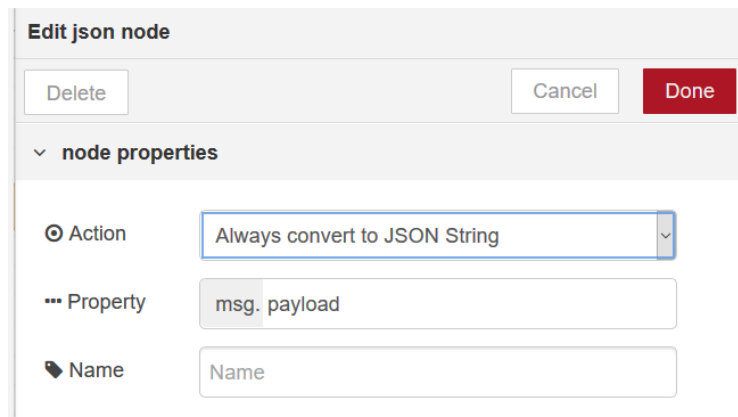
Application Note for data storage and logging to SD card

All three inject nodes have the same parameterization. Only the payload value is different.



The screenshot shows the 'Edit inject node' dialog box. At the top, there are three buttons: 'Delete', 'Cancel', and 'Done'. Below these is a section titled 'node properties' with a dropdown arrow. The 'Payload' field is set to '{ } {"C":1}' with a dropdown arrow and a three-dot menu. The 'Topic' field is empty. There is a checkbox for 'Inject once after 0.1 seconds, then' which is currently unchecked. The 'Repeat' field is set to 'none' with a dropdown arrow. The 'Name' field is set to 'Name'.

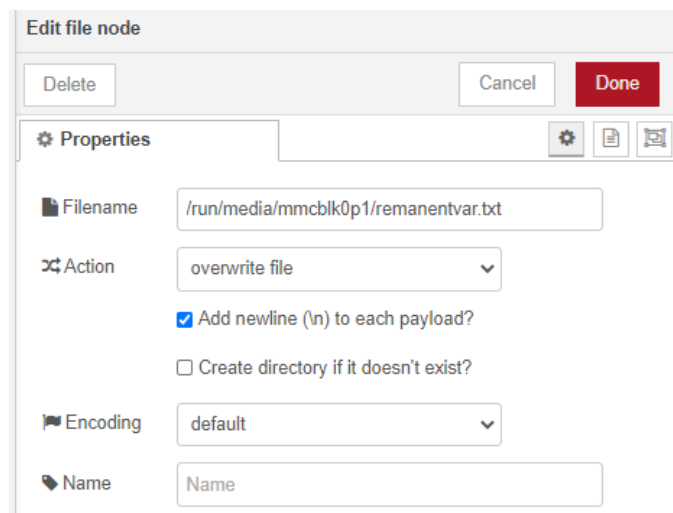
The messages are formatted by the **JSON** node . The incoming object becomes a character string. The following picture shows the node parameterization.



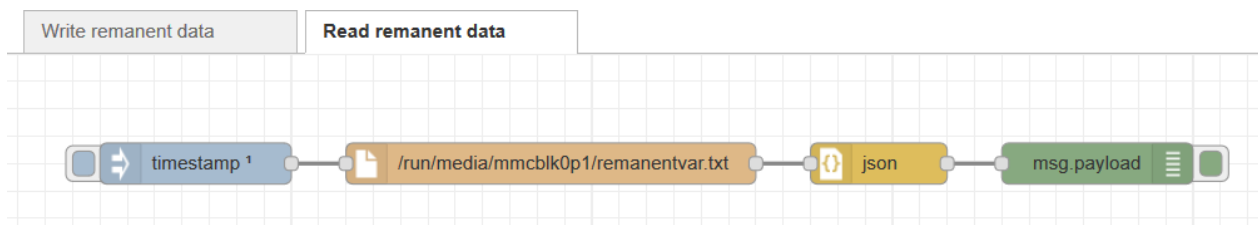
The screenshot shows the 'Edit json node' dialog box. At the top, there are three buttons: 'Delete', 'Cancel', and 'Done'. Below these is a section titled 'node properties' with a dropdown arrow. The 'Action' field is set to 'Always convert to JSON String' with a dropdown arrow. The 'Property' field is set to 'msg. payload'. The 'Name' field is set to 'Name'.


Application Note for data storage and logging to SD card

The **file** node  `/run/media/mmcblk0p1/remanentvar.txt` writes the formatted messages to the SD card.

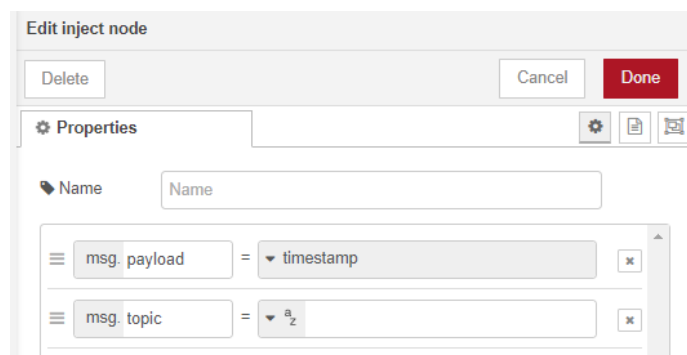


After writing to SD card, the variable is available, also after a system restart.
The following figure shows the flow required to read the remanent variable from the SD card.




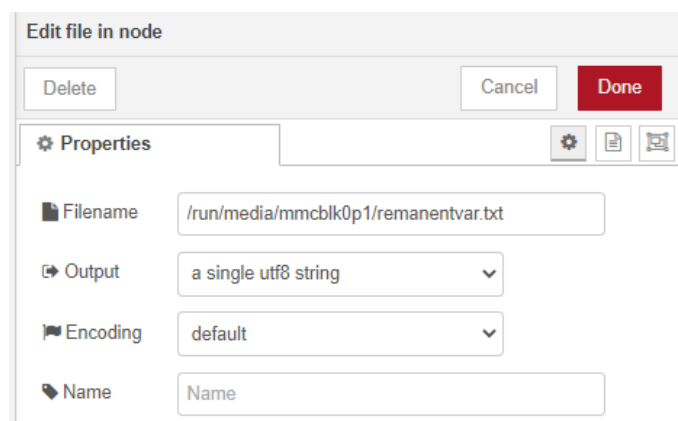
The flow is triggered once at system boot. The inject node sends a timestamp  `timestamp 1`.


The node configuration is shown below.

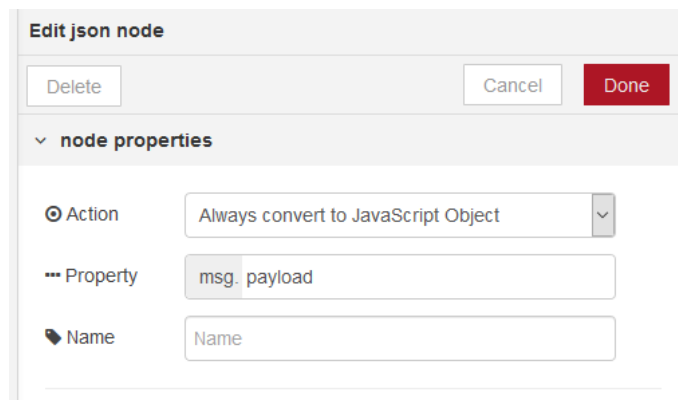



Application Note for data storage and logging to SD card

The message enters the **file in** node  `/run/media/mmcbk0p1/remanentvar.txt` and triggers the reading of the file.



On the output of the **file in** node the message contains the permanent variable with the value. This message is a string format and is formatted by the next node  `json`.



The **debug** node  `msg.payload` then displays the contents of the message in the debug console.

```
6.9.2019, 09:45:03 node: 48b71bed.bef384
msg.payload : Object
  ► { C: 3 }
```