Modbus Sparkplug Installation Instructions for Digi DAL Routers

Change Log

1.0.0	Initial Release

Contents

Installation and Auto-Start Instructions

The following instructions are to be executed within the web interface of the Digi DAL Routers

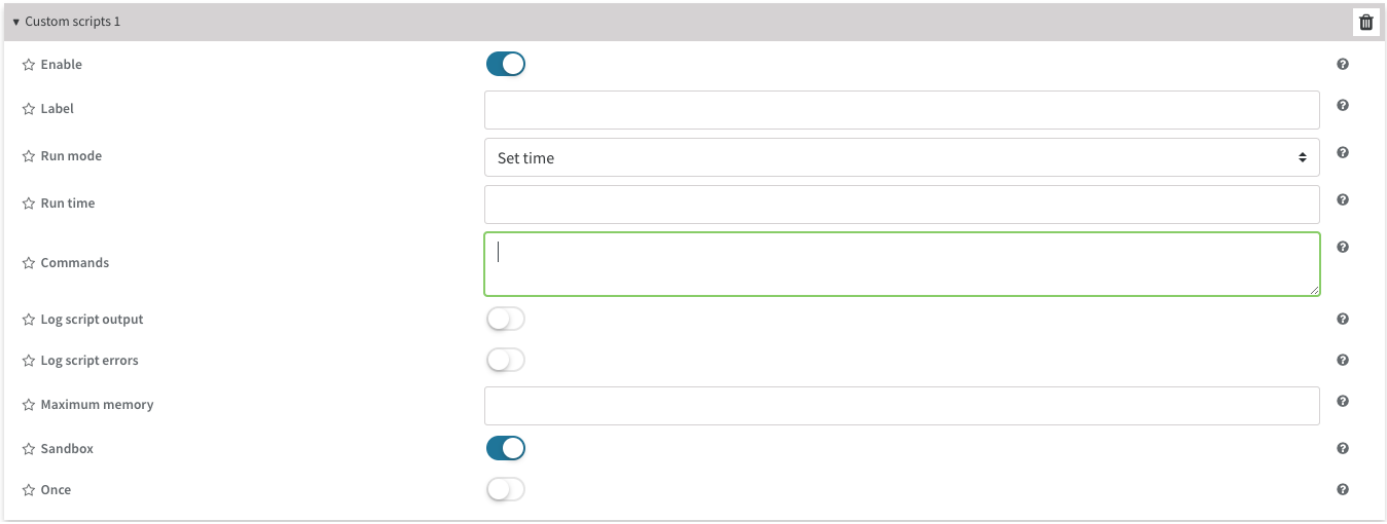
a) Installation

Upload **mb_spb.zip,** and the desired **config.cfg, slaves.cfg and any necessary register map CSV files** onto the /etc/config/scripts (or Digi RM "applications") directory of Digi DAL Routers using SFTP or Digi Remote Manager

Uploading a newer version will require uploading a new **mb_spb.zip.**

b) Setting the Application to auto-start via the web interface

a. Login into the Local Web UI
b. Navigate to **System -> Scheduled tasks -> Custom Scripts**
c. To add optional arguments click the + Add Script button.
d. Enter '**python mb_spb.zip [optional arguments]**' to the custom scripts list (without quotes)



e. Enable the custom script.
f. Add a label that will be used to identify the custom script.
g. Choose the mode that will be used.
h. Choose a runtime which will be a specific time that task will run in this format **HH:MM**
i. You can choose to enable or disable logging the script output using stdout stream to system log.
j. You can choose to enable or disable logging the scripts stdeer stream to system log.
k. Choose the maximum amount of memory for the custom script and its spawned processes that you may want to allocate.
l. You can choose to Enable or Disable to run the script in the Sandbox to prevent the behavior from affecting the system.
m. Choose to enable or disable to run the script once at a specified time refer to (h)

n. Click "**Apply**" on the top right to apply the changes.

c) Add Parameters

optional arguments:

-h, --help show this help message and exit

--debug, -d Show extra output

--file_log, -l Turn on information and debug messages to persistent file logging

slaves.cfg Configuration File

The Slaves.cfg file contains the information of the end Modbus slaves that will be communicated with the application. The following information outlines the parameters of the Slaves.cfg configuration file:

```
[slave1]
mode = rtu
slave_id = 1
port = /dev/serial/port1
register_map = /etc/config/scripts/regs.csv
little_endian = 1
[slave2]
mode = tcp
slave_id = 1
address = 192.168.1.50
port = 502
register_map = /etc/config/scripts/regs.csv
little_endian = 0
```

a) [slave1]/[slave2] - Configuration section name of specific, individual slave. This will be used for the device ID when uploading register values

b) mode – Mode of communication. Options are "tcp" or "rtu".

c) slave_id – Slave ID of slave. This will be used when creating a read query to the device.

d) address – IP address of slave. This is required for "tcp" mode only.

e) port – If "rtu" mode, this is the port name of the Digi Device serial adapter that is attached to the slave. If "tcp" mode, this is the TCP port of the slave (if unknown, use 502).

f) register_map – File path of register map file

g) little_endian – 0 = higher bits are in the lower register ; 1 = lower bits are in the lower register. If little_endian is not defined in the slaves.cfg file, the application will use the default big endian (little_endian = 0).

config.cfg configuration file

The config.cfg file contains the information of how often the application will poll for data and upload that data to the MQTT Broker. This section outlines the parameters of the config.cfg file:

```
[modbus_spb]
poll_frequency = 900
upload_on_change = True
scan_interval = 5
scan_threshold_percent = 0
broker_address = abcd123f4567b33a555e111bb2bfc22.s1.eu.hivemq.cloud
broker_password = mb_sp_test_pw
broker_username = mb_sp_test_un
group_id = test 123
```

a) **[modbus_dpb] –** Configuration section name of general application options.
b) **poll_frequency –** Time, in seconds, currently not used but required to enter poll_frequency integer value
c) **upload_on_change –** True/False. When true, the application will continuously scan the defined slaves/registers and upload the items that change.
d) **scan_interval –** Number of seconds to wait between scans for changes. This is only used when upload_on_change is set to true. Default is 5.
e) **scan_threshold_percent –** Minimum percentage change that a value must meet to be considered new when using **upload_on_change** feature. Setting this to 0 disables this feature. Default is 0 (disabled).
f) **broker_address –** URL for MQTT server connection
g) **broker_password –** Password for MQTT server connection
h) **broker_username –** Username for MQTT server connection
i) **group_id –** for MQTT message topic

Register Map CSV Columns

The CSV file must include the following headers, in order from left to right. Required headers: the register_type, name, data_type and starting_address columns are required. The length is required if the data_type is “Char”. See the following descriptions for additional requirements when using the reset_delay and/or data_factor features. An example CSV file is included with the application.

a) **register_type –** Type of register that holds described data. Options are ‘coil’, ‘discrete_input’, ‘holding_register’, and ‘input_register’. The register_type value is required.
b) **name –** Name of datastream that will receive register data. This name will also be used when writing values. The name value is required.

c) **data_type –** Type of data contained in register. Options (# registers):
   a. **Int (2)**
   b. **Float (2)**
   c. **Char (user defined)**
   d. **long_long (4)**
   e. **unsigned_long (2)**
   f. **unsigned_short (1)**
   g. **unsigned_long_long (4)**
   h. **short (1)**
   i. **double (4)**
   j. **long (2)**
   k. **unsigned_int (2)**

d) **starting_address –** Address of Modbus register that contains data item. The starting address is required.

e) **length –** Count of registers to poll to get entire data item. This field is required if data_type = char, otherwise the application will override this field with the # registers defined in the data_type description above. A value (empty value is acceptable) is required for this column if the reset delay or data_factor is required. See the following descriptions of reset_delay and data_factor.

f) **reset_delay –** After writing a register, reset_delay is the minimum number of seconds that will elapse before the register is set to '0'. Registers with a reset_delay will be set to '0' when the application starts. This column is required (default = 0 = no reset) If a data_factor is required.

g) **data_factor –** Register write: value will be divided by the data_factor before the register write is performed. Register read: value will be multiplied by the data_factor before uploading the value to Remote Manager Data Stream. This column is not required. The default value is "1".

Known issues

Currently does not support write register(s)

Currently does not support MQTT cloud to device messages