PIGEON
PLEX

Pigeon+

User Manual

Revision Control

Version	Description	Name	Date
1.0	Pre-release	Randolph Bock	7 June 2023
1.1	Fixed some commands and added new ones, edited dlms and Modbus explanations added rtc set reboot time explanation, explained password commands, added antenna explanations and images.	Michael Kaplan	27 March 2025
1.2	Added 2 more modbus commands and descriptions	Michael Kaplan	21 May 2025
1.3	Added kmp dlms history and rtc sync to console commands	Michael Kaplan	19 September 2025

Index

Overview

Pigeon+ is an IOT (Internet of Things) gateway able to communicate over user configurable connections such as 4G/LTE, Wi-Fi, Bluetooth and 10/100 Ethernet.

The device comes standard with two RS485 ports with the ability to communicate over a range of different communication protocols. These protocols include but are not limited to the following:

- Modbus Client
- Modbus Server
- Modbus Passive Listener
- DLMS (Device Language Message Specification)
- Pigeon+ also comes standard with the following Input and outputs;
    - One Universal Input [User selectable for 0-10V, 10k resistor, 10k thermistor and digital input]
    - One Sensor Input [User selectable 10k resistor, 10k thermistor and digital input]
- One built in ambient temperature sensor
- Two digital outputs [User selectable Digital output, open drain]
- The small footprint of the Pigeon+ and din rail mounting capabilities makes it ideal for deployment into cabinets close to points of measurements and control, reducing wiring and complexity. All user configured parameters are stored in flash memory with nonvolatile registers stored in EERAM.

General Applications

Some use cases for the Pigeon+ include the following:

- **Remote Metering**: The IoT device can be used in utility metering applications to collect data from gas, water, or electricity meters, enabling remote meter reading and billing.
- **Environmental Control in Buildings:** By connecting to sensors for temperature, humidity, and air quality, the IoT device can control HVAC systems or ventilation to maintain optimal indoor conditions.
- **Remote Asset Monitoring:** The device can be deployed in remote locations to monitor the condition and performance of critical assets like pipelines, remote machinery, or unmanned stations. This can help with predictive maintenance and reducing downtime.
- **Smart Home Automation:** With the digital outputs, the IoT device can control home appliances or lighting systems remotely. Combined with sensors like motion detectors or door/window sensors, it can enable smart home automation and security features.
- **Agriculture and Irrigation:** Farmers can utilize the IoT device to monitor soil moisture levels, temperature, and other environmental factors critical for agriculture. The device can control irrigation systems through digital outputs, ensuring efficient water usage.
- **Energy Management:** By connecting the IoT device to energy meters or smart plugs, it can help monitor energy consumption in homes, offices, or factories. The data can be analyzed to identify energy-saving opportunities and optimize energy usage.
- **Industrial Monitoring and Control:** The IoT device can be deployed in industrial settings to monitor various parameters like temperature, pressure, humidity, or flow rates using the analog inputs. It can also control actuators or machines through digital outputs, enabling remote automation and management.
- **Environmental Monitoring:** The device can be used to collect environmental data from sensors such as air quality monitors, weather stations, or soil moisture sensors. This data can be transmitted through Wi-Fi or LTE for real-time monitoring and analysis.

These are just a few examples, and the versatility of the Pigeon+ opens numerous possibilities for various industrial and commercial applications.

Digital Twin Eco-system

Part Numbering

Powering Device

The power supply is 12 to 24 volts at 2 amps DC. The device uses a 2 Wire connection for the power supply. When a DC supply is used the Ground connection is also connected to the COM Rail.

Hardware Setup

Network hardware for the Pigeon Plus has the option for Mobile or Wi-Fi network, therefore there are 2 antenna mounts on the front of the pigeon plus device, marked Mobile and Wi-fi there are different antennas used in each case for mobile an LTE Blade antenna is used as seen below.

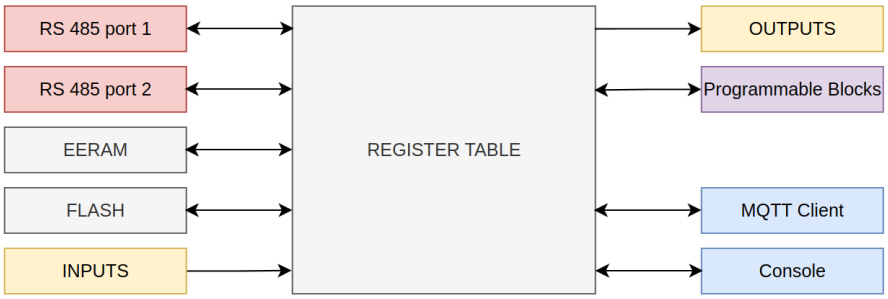
And for Wi-Fi we use a 2.4MHz terminal antenna as seen below. It is really important to use the correct antenna on the correct mount, to ensure functionality.

IO

No specific requirements for wiring. Follow standard practice for screening of inputs. Note the I/O ports are not isolated, so the ground rail is susceptible to induced voltages if field wiring is not installed as per standard practice. Care to be given to preventing induced voltages by using screen cable, terminating screen at one end only and not running cables next to higher voltages. The RS485 ports are not optically isolated, a dedicated ground connection is recommended for each bus connection.

Technical Overview

Register Table



A common register table is used to provide a flexible modular product that is easily adapted to each use case. The diagram below shows the register table positioned as the central repository for data – all operations read from this table and write the result to the table. The table definition details each register and is divided into user and system areas. These are further divided into volatile, flash values, flash for values that are loaded at boot and will not change – for example, a setpoint where the default value always starts at a known value. Eeram is used for values that change and must be retained during power fail – examples are runtime, pulse count or a setpoint that must remember its last value.

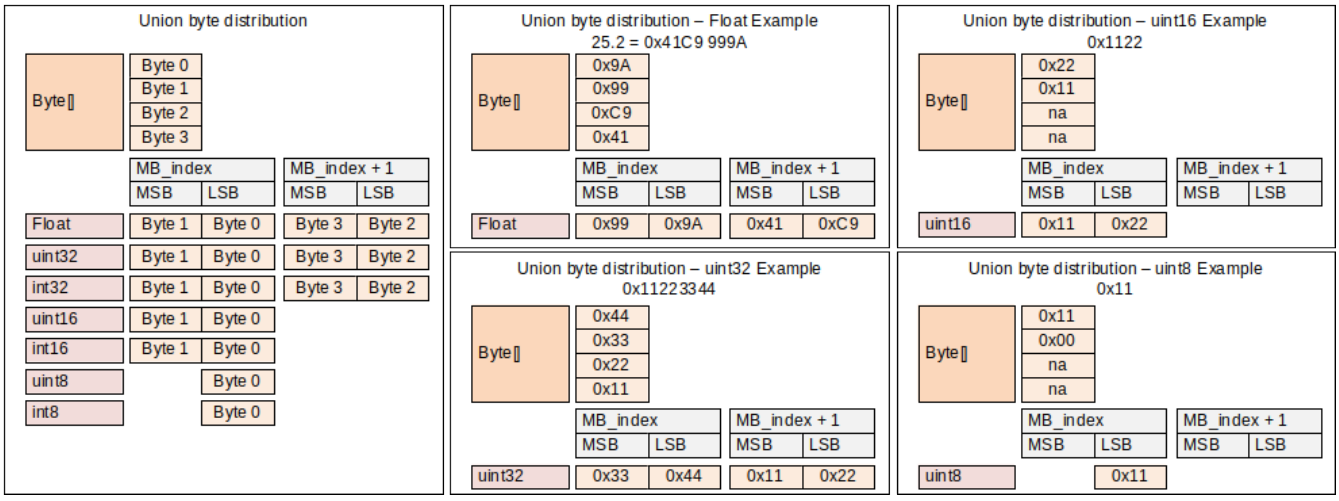
Data Types

The register table is made up of 16-bit registers. The following data types are supported by using 16-bit registers as a base.

Data Type	Data Type Enum	Number of Registers	Note
uint8	1	1	0 to 255
uint16	2	1	0 to 65 535
uint32	3	2	0 to 4 294 967 295
int8	4	1	-127 to +127
int16	5	1	-32 767 to +32 767
int32	6	2	-2 147 483 647 to + 2 147 483 647
float	7	2	-3.4E+38 to +3.4E+38. Little Endian byte swap

Remember to allocate the correct number of registers when creating the Index. If a data type uses 2 registers, then allocate register x for the data type and register x+2 for the next.

The format is Little Endian Byte Swap. As an example, the float value of 25.2 (0x41C9 999A) is written as 0x999A 0x41C9. The words are from lowest Modbus index to Highest Modbus index and the word sequence is MSB LSB.



When writing to registers where the value exceeds the max value of a data type, the results will be as follows.

Example write value 300 to an uint8. The result will be 44. This is explained by looking at the bytes

300 in decimal = 0x12C in hex. If we look at the size of an uint8 it is 1 byte. take the 0x2C and convert to decimal. The result was 44. This means that type casting shows the bytes available in the registers at the correct byte position and does not do range checking.

Register Table - Logical Areas

The register table is divided into user and system logical areas. Each area is further divided by memory type into volatile and nonvolatile with non-volatile divided into flash and EERAM.

Each register is 16 bits and can be assigned a data type to identify how it is represented. Each register therefore has 2 tables, the register table, and the register type table.

Register Section	Note	Address from	Address to
Volatile user	Use for variables display variables such as IO	0	149
Non-Volatile user Flash	Use for variables display variables such as setpoints. These values are read from Flash on restart and NOT written to flash on change. This allows for default values. If the value is to be saved on write, then use EERAM.	150	199
Non-Volatile user EERAM	Use for changing variables such as counters and runtime. These values are not saved in flash so have no initialization. Can be initialized via file and will retain last value between power cycles.	200	249
Volatile system	Predefined use. Refer to User Selected System Values.	250	299
Non-Volatile system Flash	Predefined use. Refer to User Selected System Values. Flash	300	349
Non-Volatile system EERAM	Predefined use. Refer to User Selected System Values. EERAM	350	397
Special Registers	Reboot – Write 0x55AA. Not written to NV	398	
	Factory Reset – Write 0x1928. Not written to NV	399	
Non-Volatile String value registers	Used to store string values	400	420

Note – Control blocks can write to any Non-Volatile register. Any write from a control block to Flash or EERAM space does not initiate a write to either Flash or EERAM – this allows registers in this space to be used as scratch registers. Ensure that registers are not double allocated.

SPIFFS FILING SYSTEM

A filing system to store non-volatile data – SPIFFS is an acronym for Serial Peripheral Interface Flash File System. The filing system contains ASCII text files to save the various user defined options. SPIFFS is not a memory type – it is a file system.

The files are used for configuration, default values, personalities, and control loops. Each file has a defined format and syntax and is described in the relevant sections.

The register table has a user and a system section. The user values hold registers that need to maintain the values between power cycles can be seen as the fallback values. The system section holds the configuration of the display such as baud rates, address etc. Refer to the detailed **Register Table.**

Non-Volatile section of the register table will be written on change according to the following rules.

When a change is detected a 30 second timer is set.

The timer will be reset each time there is a change.

Continual requests shorter than 30 seconds will prevent NVR (Non-Volatile Registers) write until there is more than a 30 second gap between consecutive writes.

After the 30 second timer, the obj_mb_nvr.txt SPIFFS file will be written.

DO NOT continually write to Non-Volatile addresses. Only use this area to hold setpoints or values that are expected to be retained between reboots. If a setpoint is written from a controller where the value is not expected to be retained between power cycles, then use the Volatile address space. 30 second write cycle equates to a minimum flash life of 30 years.

The files are loaded using edgeUP-App or a 3rd party serial port utility – edgeUP-App is recommended for reliable trouble-free filing.

EERAM

Memory that retains data values after power cycle or power fail. EERAM user section holds user non-volatile values where a pre-determined fallback value is not required. EERAM system holds runtime and pulse counts.

EERAM Rules

User values are saved when changed (either comms or console).

System values are saved on change every second. If there is no change, no registers are saved.

FLASH

Memory that retains data values after power cycle or power fail. These registers are read from a text file on SPIFFS. Flash registers are read after power cycle or reboot.

When a write is received in the user or system section of flash, it must be written to SPIFFS if the value is to become nonvolatile. This is done by using the write configuration register.

LED indicator Definitions

The Pigeon+ is equipped with 4 application controlled LED indicators:

1. **Power**
   - Solid LED indicator - The device is powered up.
2. **Status**
   - 1 second pulse - The device firmware application is running.
3. **Network**
   - 1 second pulse – The device is connected to a network & has a valid internet connection.
   - 2 second pulse – The device is connected to a network but with no internet connection.
4. **Bus**
   - 1 second pulse – The device has received valid packets (MODBUS or DLMS) from a server device.

Console

A text console for direct basic commands is available on the USB port. The command echo is on meaning the character typed is echoed back. If a command is not completed, the console will time out after 5 seconds. To check if the console is connected, press enter – reply will be error if USB connected and device communicating. The console commands are used to communicate with the Engineers Tool.

By default, the console is locked and requires the "password" command to unlock it. The device comes with a default password, which can be changed; however, if the password is changed, there is no "forgot password" functionality, and the device will be locked out if the new password is forgotten. In the locked state, the console displays basic logs but no configuration changes can be made. Once unlocked, configurations can be set, and additional logs will be shown. For safety, the console automatically locks after 300 seconds, requiring it to be unlocked again for further configuration. Passwords are always case sensitive.

Command	Description
psw:<Password>	**Unlock console - default password is "pigeon" + last 4 characters of the MAC case sensitive Example: If MAC = "xx xx xx xx B3 F5" password is "pigeonB3F5"**
psw-set:<New password>	**Set new password - Warning if new password is forgotten there is no forgot password functionality**
?	Help – prints out the list of commands
dir	Lists the SPIFFS directory files
log	Set the USB log verbosity
printn	Print file contents with no line number – then the file index as listed by the SPIFFS directory
printl	Print file contents with line number – then the file index as listed by the SPIFFS directory
reboot	Reboot the device
writet	Write a text file - replies ok. Send the file name followed by each text line, all lines are Ack'd (0x06).
info	Product Family model, firmware version, serial number,
format	Format SPIFFS
rit	Register index type. Used to set the data type of the register. rit index type Example, set the type of register index 20 to float: rit 20 7
riv	Register index value. Used to set the data value of a register. riv index value Example, set the value of register index 20 to 12,5: riv 20 12,7
rir	Register index read. Used to read the type and value of a register index. rir index
ris	Register index string. Used to write string values to special registers, ris index, (String value) Example, set the Wi-Fi SSID to "Test Wi-Fi": ris 401,Test Wi-Fi The command is followed by a space, then the special register address, then delimited by a comma ',' and finally the string value to store. String will be stored up to the termination character
pd	This will return a print-out of the directory of the file system.
pf:<file name>↵	This will print the contents of the file with the name
rf:<file name>↵	Remove a file with the name
nf:<file name>|<file contents>↵	Create/Replace a file with the name containing
mb-print-points:Portx	Prints all the read points and associated data to console.
dlms-print-points:Portx	Prints all the read points and associated data to console.
modbus-write: port, address, register address, register type,value	Used to write to a single Modbus register of a connected device (only works if pigeon is set up as a client)
modbus-write-multi: port, address, register address, num registers, register type,value	Used to write to multiple registers in order of a connected device (only works if pigeon is set up as a client)
dlms-rtc-sync	This command is used to initiate a time synchronization between Pigeon system time and all configured meters. dlms-rtc-sync
kmp-set-pp:<Serial number>,<kwhs>	This command is used to set the number of Kilowatt hours in the meters prepaid register. E.g. kmp-set-pp:32555400,0 . This will set the kilowatt hours in the prepaid register to 0. The meter will cut out. E.g. kmp-set-pp:32555400,5 . This will set the kilowatt hours in the prepaid register to 5.

kmp-add-pp:<serial number>,<kwhs>	This command is used to add the number of Kilowatt hours to the meters prepaid register E.g. kmp-add-pp:32555400,5 This will add 5 kilowatt hours to the current value in prepaid register.
dlms-history:<request>	This command is used to pull historical data from the meter. Example command to read data from 1.1.98.1.0.255 (This one has a LOT of data in one record) for 1 record For meter Serial No. 32552812 dlms-history:1105450828031,7,32552812,1,1,1 Example command to read data from 1.1.99.1.0.255 from Nov 17 2021 12:56:40 GMT+0000 to Wed Nov 17 2021 14:02:28 GMT+0000 with no header info dlms-history:1105450828031,7,32552812,2,1637153800,1637157748,0

User Selected System Values

File obj_mb_nvr.txt

User defined system values and registers that must retain their value after power cycle (e.g., Port one board rate) are defined in the obj_mb_nvr.txt file. Registers 300 to 349 are fixed and cannot be used for user defined values. Registers 300 to 349 are saved in SPIFFS. Registers 150 to 199 are for user defined values. The structure of obj_mb_nvr.txt must not be changed. The line sequence must be as per the template.

After updating registers in the non-volatile flash memory space, a timer is started. The register is only stored 30 seconds after it has been updated

Register	Type	Enum	Notes
300/301	uint32	3	Port 1 Baud Rate. Default: 9600. Valid Baud rate supported: 4800, 9600, 19200, 38400, 57600, 76800, 115200
302	uint8	1	Port 1 Parity, Default: None. Valid Parity supported: 0 (None), 1 (Odd), 2 (Even)
303	uint8	1	Port 1 Stop Bits, Default: 1, Valid Stop bits supported: 1 (1 stop bit), 2 (2 stop bits)
304	uint8	1	Port 1 Data Bits, Default 8, Valid data bits: 5, 6, 7, 8
305	uint8	1	Port 1 Address. Default 1. Options between 1 and 255. Values outside this range will not be accepted. Not used in modbus Client mode.
306	uint8	1	Port 1 Type, Default 0 (Disabled). Supported types: 0: Disabled, 1: Modbus Client, 2: Modbus Server,3: Modbus Listener, 4: DLMS Client
307/308	uint32	3	Port 2 Baud Rate. Default: 9600. Valid Baud rate supported: 4800, 9600, 19200, 38400, 57600, 76800, 115200
309	uint8	1	Port 2 Parity, Default: None. Valid Parity supported: 0 (None), 1 (Odd), 2 (Even)
310	uint8	1	Port 2 Stop Bits, Default: 1, Valid Stop bits supported: 1 (1 stop bit), 2 (2 stop bits)
311	uint8	1	Port 2 Data Bits, Default 8, Valid data bits: 5, 6, 7, 8
312	uint8	1	Port 2 Address. Default 1. Options between 1 and 255. Values outside this range will not be accepted. Not used in modbus Client mode.
313	uint8	1	Port 2 Type, Default 0 (Disabled). Supported types: 0: Disabled, 1: Modbus Client, 2: Modbus Server,3: Modbus Listener, 4: DLMS Client
314			SIM info update interval in minutes
315			Spare
316	uint16	2	Log verbosity. Refer to log verbosity
317	uint16	2	System network mode. Refer to system network mode
318	Uint8	1	UI Mode. Refer to Universal input section
319	Uint8	1	SI Mode. Refer to Universal input section
320	Uint8	1	UI Debounce. Refer to Universal input section
321	Uint8	1	SI Debounce. Refer to Universal input section
322	Uint8	1	UI flag register. Refer to Universal input section
323	Uint8	1	SI flag register. Refer to Universal input section
324	Uint16	2	UI offset register. Refer to Universal input section
325	Uint16	2	SI offset register. Refer to Universal input section
326	Uint16	2	UI moving average time constant. Refer to Universal input section
327	Uint16	2	SI moving average time constant. Refer to Universal input section
328	Uint16		spare
329	Uint16		spare
330	uint16	2	UO port 1 configuration bits/flags - see UO flag bitmask definitions
331	uint16	2	UO port 2 configuration bits/flags - see UO flag bitmask definitions
332	uint16	2	MQTT (Message Queuing Telemetry Transport) – ping interval
333	uint16	2	MQTT – Device settings update interval
334	uint16	2	MQTT – Device location update interval
335	uint16	2	MQTT – Input data update interval
336	uint16	2	MQTT – Input settings update interval
337	uint16	2	MQTT – Output data update interval
338	uint16	2	MQTT – Output settings update interval
339	uint16	2	MQTT – Modbus data update interval
340	uint16	2	MQTT – Modbus settings update interval
341	uint16	2	MQTT – DLMS point update interval
342	uint16	2	MQTT – DLMS object update interval (Not used for now)
343	uint16	2	MQTT – DLMS settings update interval (Not used for now)
344/345	uint32	3	MQTT – log date update interval
346	uint16	2	System enable mask
347	uint16	2	Daily reboot interval

User Selected Text Values

User defined text values and registers that must retain their value after power cycle (e.g., Wi-Fi SSID). Each value is then stored in its own discrete text file:

The register ranges from 400 onwards and are defined in the table below:

Register	Description
401	Wi-Fi SSID – stored in SSID.txt
402	Wi-Fi Password – stored in wifi_pass.txt
403	OTA (Over the Air) server URL address store – stored in OTA_url.txt
404	PPPOS APN store
405	MQTT URL – stored in mqtt_url.txt
406	MQTT Username – stored in mqtt_username.txt
407	MQTT password – stored in mqtt_password.txt
408	MQTT client – stored in mqtt_client.txt
499	RTC manually set time (Only available when device is not connected to internet access)

To write to one of the text registers use the register index string “ris” command.

Example, set the Wi-Fi SSID to “Test Wi-Fi”: ris 401,Test Wi-Fi

The command is followed by a space, then the special register address, then delimited by a comma ‘,” and finally the string value to store. String will be stored up to the termination character

To read a specific value, firstly use the ‘dir’ command to get the file directory and each file’s index. Afterwards use the ‘printn’ or ‘printl’ then the file index as listed by the SPIFFS directory of the associated file to read the file context.

Input

Input operation is determined by the associated configuration registers. There are two inputs – Universal Input UI) & Sensor Input (SI). Each device has a unique Inputs table detailed in the respective device register Input table. The explanations that follow detail the operation of each Input type.

Set the register value for each of the following configuration options for the Universal and sensor inputs

Register	**Type**	**Enum**	**Notes**
318	uint8	1	UI mode. See Input Modes
319	uint8	1	SI mode. See **Input Modes**
320	uint8	1	UI, debounce counter - Time in seconds that a digital mode input must remain in a state for the state to take effect.
321	uint8	1	SI, debounce counter - Time in seconds that a digital mode input must remain in a state for the state to take effect.
322	uint8	1	UI, configuration flag. See flags table
323	uint8	1	SI, configuration flag. See flags table
324	int16	5	UI offset. 0-10V mode - in millivolts 0-10% mode - in percentage Resistance mode – in ohms 10K Thermistor mode – in degrees Celsius Analog direct mode – in millivolts
325	int16	5	SI offset. Resistance mode – in ohms 10K Thermistor mode – in degrees Celsius
326	Uint16	2	UI moving average time constant - Time constantin seconds. Used in (Exponentially weighted moving average) EWMA filter applied to analogue inputs. For a step change in input, it will take approx. 10 x tau for the average to settle to the new value.
327	Uint16	2	UI moving average time constant - Time constantin seconds. Used in (Exponentially weighted moving average) EWMA filter applied to analogue inputs. For a step change in input, it will take approx. 10 x tau for the average to settle to the new value.

Input Modes

The UI mode can be set to the following types by writing the value to the register.

Value	Type		UI	SI	UI Biasing
0	Disabled		Yes	Yes	No Bias
1	Digital Input		Yes (pull-up)	Yes	Pulled up (default high level)
2	0 to 10 Volt		Yes	no	Pulled down
3	0 to 10 Volt in percentage		Yes	no	Pulled down
4	Analog, Resistive Ohms		Yes	Yes	Pulled up
5	Thermistor 10K type II		Yes	Yes	Pulled up
6	Analog Voltage direct 0-3.3V		Yes	no	No Bias

The Sensor input is always biased with an active high level, i.e. pulled up to 3.3V regardless of the input type set.

Inputs Flags Register

The flags register is set as a bitmask, with each bit representing an enable (1) or disable (0) state

The flag register is described as a uint8 with the following bit definitions:

Bit 8	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1
Reserved	Reserved	Reserved	Reserved	Reserved	Change of value notify enable	Pulse count on rising edge enable	Invert Logic enable

Input Register Values

The UI's value outputs are stored in volatile register spaces described below:

Register	Type	Enum	Notes
250/251	float	7	Internal ambient temperature sensor output
252/253	int32	6	UI Value store
254/255	int32	6	SI Value Store
256/257	int32	6	UI moving average value
258/259	int32	6	SI moving average value

Digital Outputs (DO)

Universal output operation is determined by the associated configuration registers. The explanations that follow detail the operation of each UO type.

The Pigeon+ has two digital output ports. Both output ports are set up as open collectors.

Set the following non-volatile registers value for each of the following configuration options for the two DO ports

Register	Type	Enum	Notes
330	uint16	2	DO port 1 configuration bits/flags - see **DO flag bitmask definitions**
331	uint16	2	DO port 2 configuration bits/flags - see **DO flag bitmask definitions**

DO Flag Register

The flags register is set as a bitmask, with each bit representing an enable (1) or disable (0) state

The flag register is described as a uint16 with the following bit definitions:

Bit 8	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1
Reserved	Reserved	Reserved	Invert enable	Reserved	Reserved	Reserved	Digital Mode enable

DO Example Output Registers

The DO's value outputs are stored in volatile register spaces described below.

Register	Type	Enum	Notes
260	uint8	1	DO port 1 Set value
261	uint8	1	DO port2 Set value
262/263	uint32	6	DO port 1 runtime counter – seconds that the output of the port was driven high
264/265	uint32	6	DO port 2 runtime counter – seconds that the output of the port was driven high
266/267	uint32	6	DO port 1 change of state count – number of state changes for the port
268/269	uint32	6	DO port 2 change of state count – number of state changes for the port

Example

To drive output port on high set the mode to digital and write a value of one to the DO port 1 set value:

riv 330 1

riv 260 1

System network mode

System network modes are set by writing to the non-volatile register 317. The register determines the network modes by using the following bitmask in the register, with each bit representing an enable (1) or disable (0) state.

The bitmask for the system network mode is described as the following

Bit 16	Bit 15	Bit 14	Bit 13	Bit 12	Bit 11	Bit 10	Bit 9
Reserved	Reserved	Reserved	Reserved	Reserved	Reserved	Reserved	Bluetooth server enable

Bit 8	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1
Reserved	Reserved	Reserved	Reserved	Reserved	Ethernet TCP/IP enable	Mobile Data enable	Wi-Fi enable

Wi-Fi

To connect to a Wi-Fi network, attach a Wi-Fi Antenna (2.4GHz) to the designated UFL receptacle labled Wi-Fi, then set the System mode register (311) to enable Wi-Fi (Set bit 1 high). riv 317 1

The device's Wi-Fi will then try to connect to the SSID stored in the text register 401 with the passkey stored in text register 402.

Register	Description
401	Wi-Fi SSID – stored in SSID.txt
402	Wi-Fi Password – stored in wifi_pass.txt

Example

In this example we will set the network mode to Wi-Fi only. Then set the SSID to myWIFI and the passkey to myPasskey:

riv 317 1

ris 401,myWIFI

ris 402,myPasskey

Mobile

To connect to a mobile LTE network, first insert a SIM card into the internal sim tray on the LTE module card and connect an appropriate LTE antenna to the UFL receptacle labelled Mobile. Afterwards set the System mode register (317) to enable Mobile (Set bit 2 high). riv 317 2

The device will then start up the LTE modem and connect to an LTE network on the next boot after the config has been set for mobile.

If the SIM inserted requires a specific APN it can be specified in register 404.

Register	Description
404	PPPOS APN store

Ethernet

To connect the device to an ethernet network for the intention of using ethernet as a data connection, firstly insert the ethernet cable into the RJ45 connector, followed by setting the System mode register (317) to enable Ethernet (Set bit 3 high). riv 317 4

Ethernet will then Connect to a DHCP enabled network. Currently only DHCP enabled networks are supported, additional Ethernet features in development.

System specific modes

System daily reboot

To enable daily reboot on the Pigeon+, set bit 0 of the bitmask in register “System enable mask” (346) to 1.

Afterwards the system will reboot after the daily reboot interval has elapsed. The daily reboot interval is stored in register “Daily reboot interval” (347). The interval is stored in minutes and can be changed to the user’s desire. The default interval is 1440 minutes, or 24 hours.

To enable specific time daily reboot on the Pigeon+, set bit 0 of the bitmask in register “System enable mask” (346) to 2.

Afterwards the system will reboot after the daily reboot interval time starting from 12am as 0 . The daily reboot interval is stored in register “Daily reboot interval” (347). The interval is stored in minutes and can be changed to the user’s desire. The default interval is 1440 minutes, or 12:00 at night.

BLE Server

Creates a BLE server that a device can connect to, with bi-directional comms.

The GAP device name is set to: PP_xx_xx_ xx_ xx_ xx_ xx. XX represents the device’s mac address in Hex.

Example:

To enable the BLE server write to register 317 with the value 256: riv 317 256

To enable both the BLE Server and Wi-Fi write a value of 257 to register 311: riv 317 257

To enable both the BLE Server and Mobile network write a value of 258 to register 317: riv 317 258

MQTT

MQTT is an OASIS standard messaging protocol for the Internet of Things (IoT). It is designed as an extremely lightweight publish/subscribe messaging transport that is ideal for connecting remote devices with a small code footprint and minimal network bandwidth.

To set up the MQTT functionality of the Pigeon+, the device should first be connected to a network source. The Pigeon+ can connect to Mobile networks through an LTE data connection or to IP (Internet Protocol) network through Wi-Fi or ethernet.

The birds transmit JSON strings to a user defined MQTT broker client through a data connection, either WIFI, LTE/4G or ethernet. The Pigeon+ publishes information to the “data” topic and subscribes to the “commands” topic. The birds are differentiated by their ICCIDs on the MQTT broker, i.e. *ICCID/data* (*A4E57C764FA0/data*) or *ICCID/commands* (*A4E57C764FA0/commands*).

Only after the device’s network interface is configured will the device try and connect to a MQTT broker. The MQTT broker’s config can be set with the following text registers:

Register	**Description**
405	MQTT URL – stored in mqtt_url.txt
406	MQTT Username – stored in mqtt_username.txt
407	MQTT password – stored in mqtt_password.txt
408	MQTT client – stored in mqtt_client.txt

The timeouts (In number of minutes are stored in the registers described below. For example, if we want the device to send a device settings log message every hour, we will set register 333 to 60. The console command to set register 333 to 60 will be: riv 333 60

Register	Type	Enum	Description
333	uint16	2	MQTT – Device settings update interval
334	uint16	2	MQTT – Device location update interval
335	uint16	2	MQTT – Input data update interval
336	uint16	2	MQTT – Input settings update interval
337	uint16	2	MQTT – Output data update interval
338	uint16	2	MQTT – Output settings update interval
339	uint16	2	MQTT – Modbus data update interval

340	uint16	2	MQTT – Modbus settings update interval
341	uint16	2	MQTT – DLMS point update interval
342	uint16	2	MQTT – DLMS object update interval (Not used for now)
343	uint16	2	MQTT – DLMS settings update interval (Not used for now)
344	uint32	3	MQTT – log date update interval

Input and output in the table above refers to the two integrated inputs and two integrated digital outputs of the device.

Real Time Clock

The Pigeon+ has a real time clock installed onto the device with a backup battery.

The time is synced to the RTC (Real Time Clock) with a SNTP (Simple Network Time Protocol) service when connected to a network with internet access. This means that the time is automatically synced to the SNTP time every minute, but only if the device is connected to have internet access.

If the device does not have internet access it will keep time using the RTC. The RTC can also be set manually when the device is not connected to the internet by doing a String write to index 499 with the following format: ris 499,hh:mm:ss:DD:MM:YY

Once the device regains internet access, it will then sync the RTC with the SNTP service again.

Log Verbosity

Use the console log command to change the log output level to the console.

Value	Description
0	None
1	Error
2	Warning
3	Info (default prior to reading mb_nvr settings)
4	Debug
5	Verbose

Console Errors

dir / format

Error Value	Description
0	No Error
1	SPIFFS get file list error or format error

Log

Error Value	Description
0	No error
1	Length exceeded
2	Undefined mb type
3	No string null terminator
4	Not a number

printn / printl

Error Value	Description
0	No error
1	File at index given not found
2	
3	

4	NSPIFFS index read error
5	No file at SPIFFS index
6	Unable to read file for print
7	No CR or max line length exceeded

writet

Error Value	Description
0	No error
>0 <0xFC	File append error in line number shown Or SPIFFS file name not found if parsing file
0xFC	File name error
0xFD	File open error
0xFE	Could not mount SPIFFS
0xFF	SPIFFS locked

Modbus

The Pigeon+ has two dedicated RS485 Ports that can be configured for various communication protocols. One such protocol that is supported is Modbus. Each port can be independently configured as a modbus Client, Server or Passive Listener.

Configuration for modbus client / server can be done by writing to the following configuration registers in the registry table:

Port 1

Register	Type	Enum	Notes
300/301	uint32	3	Port 1 Baud Rate. Default: 9600. Valid Baud rate supported: 4800, 9600, 19200, 38400, 57600, 76800, 115200
302	uint8	1	Port 1 Parity, Default: None. Valid Parity supported: 0 (None), 1 (Odd), 2 (Even)
303	uint8	1	Port 1 Stop Bits, Default: 1, Valid Stop bits supported: 1 (1 stop bit), 2 (2 stop bits)
304	uint8	1	Port 1 Data Bits, Default 8, Valid data bits: 5, 6, 7, 8
305	uint8	1	Port 1 Address. Default 1. Options between 1 and 255. Values outside this range will not be accepted. Not used in modbus Client mode.
306	uint8	1	Port 1 Type, Default 0 (Disabled). Supported types: 0: Disabled, 1: Modbus Client, 2: Modbus Server,3: Modbus Listener, 4: DLMS Client

Port 2

Register	Type	Enum	Notes
307/308	uint32	3	Port 2 Baud Rate. Default: 9600. Valid Baud rate supported: 4800, 9600, 19200, 38400, 57600, 76800, 115200
309	uint8	1	Port 2 Parity, Default: None. Valid Parity supported: 0 (None), 1 (Odd), 2 (Even)
310	uint8	1	Port 2 Stop Bits, Default: 1, Valid Stop bits supported: 1 (1 stop bit), 2 (2 stop bits)
311	uint8	1	Port 2 Data Bits, Default 8, Valid data bits: 5, 6, 7, 8
312	uint8	1	Port 2 Address. Default 1. Options between 1 and 255. Values outside this range will not be accepted. Not used in modbus Client mode.
313	uint8	1	Port 2 Type, Default 0 (Disabled). Supported types: 0: Disabled, 1: Modbus Client, 2: Modbus Server,3: Modbus Listener, 4: DLMS Client

Modbus Server

The two RS485 ports on the Pigeon+ can be configured to independently operate in a modbus server mode. Using the above Registry table settings for Port 1 & Port 2 respectively.

Example

To setup Port 1 as a modbus server with address 25 (0x19), at baud 115200, no parity, 1 stop bit, 8 data bits. Issue the following commands on the console:

riv 300 115200

riv 302 0

riv 303 1

riv 304 8

riv 305 25

riv 306 2

Modbus Client

The two RS485 ports on the Pigeon+ can be configured to independently operate in a modbus client mode. Using the above Registry table settings for Port 1 & Port 2 respectively.

Example

To setup Port 2 as a modbus client at baud 9600, no parity, 1 stop bit, 8 data bits issue the following commands on the console:

riv 307 9600

riv 309 0

riv 310 1

riv 312 8

riv 313 1

Multiple baud rates are supported where the baud rate is set according to the control block settings. The default baud rate is set according to the register table.

To set up endpoints for a modbus client to poll or write to registers, control blocks need to be configured. Please refer to the control blocks section of this manual.

Control Block Examples for Modbus Client

Modbus Read Coil

Control Block: 1,[1,1,115200,20,1,0,70,0,0,1,0,0,,,,]

[Port:1, DevAdd:1, Baud:115200, RegAdd:20, RegType:Coil, SwapF:0, DestAdd:70, InputDataType:unused, OutputDataType:unused, ScanRateMs:1s Scale:unused, Shift:unused, LowThresh:unused, HighThresh:unused, DeltaThresh:unused, TimeDelay:unused]

Modbus Read Discrete Inputs

Control Block: 1,[1,1,115200,40,2,0,72,0,0,1,0,0,,,,]

[Port:1, DevAdd:1, Baud:115200, RegAdd:40, RegType:Discrete, SwapF:0, DestAdd:72, InputDataType:unused, OutputDataType:unused, ScanRateMs:1s Scale:unused, Shift:unused, LowThresh:unused, HighThresh:unused, DeltaThresh:unused, TimeDelay:unused]

Modbus Read Holding Register

Control Block: 1,[1,1,115200,200,3,2047,0,3,0,1,1,0,,,,]

[Port:1, DevAdd:1, Baud:115200, RegAdd:200, RegType:Holding, SwapF:2047, DestAdd:0, InputDataType:uint32, OutputDataType: unused, ScanRateMs:1s Scale:1, Shift:0, LowThresh:unused, HighThresh:unused, DeltaThresh:unused, TimeDelay:unused,]

Modbus Read Input Register

Control Block: 1,[1,1,115200,300,4, 2047,35,3,0,1,1,0,,,,]

[Port:1, DevAdd:1, Baud:115200, RegAdd:300, RegType:Input, SwapF:2047, DestAdd:35, InputDataType:uint32, OutputDataType: unused, ScanRateMs:1s Scale:1, Shift:0, LowThresh:unused, HighThresh:unused, DeltaThresh:unused, TimeDelay:unused,]

Modbus Write Coils

Control Block: 2,[1,1,115200,30,1,0,72,0,1]

[Port:1, DevAdd:1, Baud:115200, RegAdd:30, RegType:Coil, SwapF:0, SourceAdd:72, DataType:1(uint8), ScanRateMs:1ms]

Modbus Write Holding Register

Control Block: 2,[1,1,115200,210,3,0,35,0,1]

[Port:1, DevAdd:1, Baud:115200, RegAdd:210, RegType:Holding, SwapF:0, SourceAdd:35, DataType:0(follows source address type), ScanRateMs:1s

The above control block examples Can be combined into one control block file as follows:

obj_cntl.txt

1,[1,1,115200,20,1,0,70,0,0,1,0,0,,,,]

1,[1,1,115200,40,2,0,72,0,0,1,0,0,,,,]

1,[1,1,115200,200,3,2047,0,3,0,1,1,0,,,,]

1,[1,1,115200,300,4, 2047,35,3,0,1,1,0,,,,]

2,[1,1,115200,30,1,0,72,0,1]

2,[1,1,115200,210,3,0,35,0,1]

The above set of control blocks can be sent to the Pigeon+ using the MQTT console nf command.

nf:obj_cntl.txt|obj_cntl.txt

1,[1,1,115200,20,1,0,70,0,0,1,0,0,,,,]

1,[1,1,115200,40,2,0,72,0,0,1,0,0,,,,]

1,[1,1,115200,200,3,2047,0,3,0,1,1,0,,,,]

1,[1,1,115200,300,4, 2047,35,3,0,1,3,0,,,,]

2,[1,1,115200,30,1,0,72,0,1]

2,[1,1,115200,210,3,0,35,0,1]

Alternatively, you can use the app to create control blocks and upload them to the Pigeon+.

Modbus register commands

You can also use the console commands modbus-write or modbus-write_multi to manually write to device registers connected to the Pigeon+. These commands will only work if the device is configured as a client and the Baud rate should be set to the correct rate of the receiving device for these commands to work.

Examples:

Command
modbus-write: 1, 1, 200,3,50
Explanation
modbus-write: Port number, Device address to write to, Register address you want to write to, the register type you are writing too see (Pigeon+UserManual.docx) ,the value you want to write into the register.

Comand
modbus-write-multi:2,1,0,5,3,220,221, 222, 223, 2500
Explanation
modbus-write-multi: Port number, Device address, Starting register, Number of registers, Register type (all register needs to be same type see (Pigeon+UserManual.docx) ,Starting register value, Next register value, Next register value, Continue for values up until number of registers.

Modbus Passive Listener

The two RS485 ports on the Pigeon+ can be configured to independently operate in a Modbus passive listener mode. Using the above Registry table settings for Port 1 & Port 2 respectively.

Example

To setup Port 2 as a modbus client at baud 115200, no parity, 1 stop bit, 8 data bit. Issue the following commands on the console:

riv 307 115200

riv 309 0

riv 310 1

riv 312 8

riv 313 3

To set up endpoints for a modbus client to poll or write to registers, control blocks need to be configured. Please refer to the control blocks section of this manual.

Control register Examples for Modbus Passive Listener

Modbus Read Holding Register

Control Block: 1,[1,1,115200,200,3,2047,0,3,0,0,1,0,,,,]

[Port:1, DevAdd:1, Baud:115200, RegAdd:200, RegType:Holding, SwapF:2047, DestAdd:0, InputDataType:uint32, OutputDataType: unused, ScanRateMs:0ms (unused), Scale:1, Shift:0, LowThresh:unused, HighThresh:unused, DeltaThresh:unused, TimeDelay:unused,]

DLMS

The Pigeon+ has two dedicated RS485 Ports that can be configured for various communication protocols. One such protocol that is supported is DLMS. Each port can be independently configured as a DLMS Client.

Configuration for DLMS client can be done by writing to the following configuration registers in the registry table:

Port 1

Register	Type	Enum	Notes
300/301	uint32	3	Port 1 Baud Rate. Default: 9600. Valid Baud rate supported: 4800, 9600, 19200, 38400, 57600, 76800, 115200
302	uint8	1	Port 1 Parity, Default: None. Valid Parity supported: 0 (None), 1 (Odd), 2 (Even)

303	uint8	1	Port 1 Stop Bits, Default: 1, Valid Stop bits supported: 1 (1 stop bit), 2 (2 stop bits)
304	uint8	1	Port 1 Data Bits, Default 8, Valid data bits: 5, 6, 7, 8
305	uint8	1	Port 1 Address. Default 1. Options between 1 and 255. Values outside this range will not be accepted. Not used in modbus Client mode.
306	uint8	1	Port 1 Type, Default 0 (Disabled). Supported types: 0: Disabled, 1: Modbus Client, 2: Modbus Server,3: Modbus Listener, 4: DLMS Client

Port 2

Register	Type	Enum	Notes
307/308	uint32	3	Port 2 Baud Rate. Default: 9600. Valid Baud rate supported: 4800, 9600, 19200, 38400, 57600, 76800, 115200
309	uint8	1	Port 2 Parity, Default: None. Valid Parity supported: 0 (None), 1 (Odd), 2 (Even)
310	uint8	1	Port 2 Stop Bits, Default: 1, Valid Stop bits supported: 1 (1 stop bit), 2 (2 stop bits)
311	uint8	1	Port 2 Data Bits, Default 8, Valid data bits: 5, 6, 7, 8
312	uint8	1	Port 2 Address. Default 1. Options between 1 and 255. Values outside this range will not be accepted. Not used in modbus Client mode.
313	uint8	1	Port 2 Type, Default 0 (Disabled). Supported types: 0: Disabled, 1: Modbus Client, 2: Modbus Server,3: Modbus Listener, 4: DLMS Client

DLMS-Client

The two RS485 ports on the Pigeon+ can be configured to independently operate in a dlms client mode. Using the above Registry table settings for Port 1 & Port 2 respectively.

Example

To setup Port 2 as a dlms client at baud 9600, no parity, 1 stop bit, 8 data bits issue the following commands on the console:

riv 307 9600

riv 309 0

riv 310 1

riv 311 8

riv 313 4

Multiple baud rates are supported where the baud rate is set according to the control block settings. The default baud rate is set according to the register table.

To set up endpoints for a dlms client to poll registers, control blocks need to be configured. Please refer to the control blocks section of this manual.

Control Block Examples for DLMS Client

The two examples below are for a Kamstrup Omnipower single phase meter.

DLMS Device Block (Block 3)

Control Block: 3,[1,9600,0,8,1,32555400,18,16,32,1,1,512,512,1,1,12345,L,0,30]

[Port:1, Baud:9600, Parity:None,Databits:8,Stopbits:1,SerialNum: 32555400,ClientAddr:18,ServelLogicalAddr:16,ServerPhysAddr:32,AddrSizeLogical:1,AddrSizePhys:1,MaxTxPayloadSize:512, MaxTxPayloadSize:512,WinFrameSizeTx:1, WinFrameSizeRx:1,Password:12345,LogicalNameRef:Long,TimeAlignHour:0,TimeAlignMinute:30]

DLMS Endpoint Block (Block4)

Control Block: 4,[32555400,1103823896831,3,2,A14,33,0,9,0.01,0.00,,,,]

[SernialNum: 32555400,RegAddr:1103823896831,ClassID:3,Attr:2,Name: A14,DestAddr:33,InDataType:0,OutDataType:float,Scale:0.01,Shift:0.0,,,,]

DLMS Control block file names

These control blocks named above must be stored in a file on the device.

DLMS control block files must follow the naming convention: dlms_cntl<device_number - 1>.txt. For each connected device, create a file with the corresponding number. For example, with 5 devices:

- dlms_cntl0.txt for device 1
- dlms_cntl1.txt for device 2
- dlms_cntl2.txt for device 3
- dlms_cntl3.txt for device 4
- dlms_cntl4.txt for device 5

This ensures Pigeon can connect to multiple DLMS devices. If the files are named incorrectly, they will overwrite each other, and only the last device's control block will function.

DLMS-Historic Data request

Use command dlms-history explained in the console section. This command allows us to read historical data off the meter.

DLMS RTC-Sync

The Pigeon will now attempt to set the time on any connected & configured meter on command,

Meter time syncs are only possible every 30 minutes. Should you try to force a time sync more frequently, time sync will not be successful. dlms-rtc-sync This command is used to initiate a time synchronization between Pigeon system time and all configured meters. dlms-rtc-sync/

DLMS Prepaid API

The pigeon now has the ability to either add KWhs to the meters prepaid register or set the KWhs value in the prepaid register.

*To allow for Top-up & prepaid value setting, the KMP address must be explicitly set and match the DLMS physical address. The address values are typically 31,32 or 33. Setting the address is done using the Omnipower meter tool. * These commands will only work correctly if the meters configuration file has been configured correctly and the "sn" key-value pair contains the correct Serial number for the meter in question. I.e. All configuration files should have unique serial number values in the "sn" field.

kmp-set-pp:SerialNumber,KWHs

This command is used to set the number of Kilowatt hours in the meters prepaid register.

SerialNumber: This value must match the serial number in the config files

KWHs: This is the number of Kilowatt hours to be set in the meters prepaid register.

E.g. kmp-set-pp:32555400,0 . This will set the kilowatt hours in the prepaid register to 0. The meter will cut out.

E.g. kmp-set-pp:32555400,5 . This will set the kilowatt hours in the prepaid register to 5.

kmp-add-pp:SerialNumber,KWHs

This command is used to add the number of Kilowatt hours to the meters prepaid register.

SerialNumber: This value must match the serial number in the config file

KWHs: This is the number of Kilowatt hours to be added to the meters prepaid register.

E.g. kmp-add-pp:32555400,5 . This will add 5 kilowatt hours to the current value in prepaid register

Control Blocks

A standard set of control blocks is available to perform control logic and additional communications functions: The control block definitions are in obj_cntl.txt file. The maximum number of lines in the file is 250 – this includes the file name in line 1. This means the maximum number of control blocks allowed is 249 dependent on available memory. Each line is a maximum of 100 characters long including the function block name. This excludes the CR or CRLF line terminator. The lines contain a JSON string that details the control block. Refer to the index of control blocks for details of each block.

The control block file is read on boot (if the file exists) and each line is read and checked. If the line syntax is ok, the control block is created. The entire file is read first to determine the size of memory required for the control block index. If there is enough memory, each line is read and the control block memory allocated and populated with the values in the JSON string. This is done line by line until the end of the file or there is no more memory to allocate at which point an error will be displayed on the console. If a register table is indexed, the register type is set according to the data type definition for the control block.

The syntax of a control block line is:

index,[param1,param2,param....,param last]

Control Block data types are important – in the tables for each control block there are 2 data type columns – Data Type and Register Type.

Data Type defines what the number range and type is allowed in the JSON string defining the control block. Register Type defines the register type that will be assigned on initialization of the control block. If 2 control blocks use the same register where 2 different types are assigned, the last block in line sequence of obj_cntl.txt will take precedence.

If registers of different types are to be linked then use a DataType block to convert.

Some registers are optional – these are defined by allowing a -1 to indicate the register is not used. The -1 must be included in the JSON string because the string is interpreted by index – this means all JSON strings must follow exactly the format of the block.

To protect the configuration from erroneous writes, the outputs are limited to write only in the register table address in the user or system volatile space. The user space includes volatile, EERAM and FLASH.

Control blocks set up the register type on creation of the block. To prevent system registers from having data types changed, no control blocks except the System Transfer can read from the system space. The System Transfer block does not set the register data types.

Note – The concepts of register table, data types, setting up text files and downloading using eZi-App are required to use control blocks. Refer to the eZi-COM-IO user manual for more information.

Index	Control Block Name	Scan	Description
0	No Block		
Communication Modbus Client			
1	Read Register	N/A	Modbus Client Read single / multiple registers.
2	Write Register	N/A	Modbus Client Write single / multiple registers.
Communication DLMS Client			
3	Device Config	N/A	DLMS Client general device configuration
4	Read Endpoint	N/A	DLMS endpoint read config
Communication Modbus TCP			
5	Device Config	N/A	Set IP address
6	Read points	N/A	Modbus TCP readpoints config
Logic			
10	Logic	100mSec	Logically AND/OR/XOR up to 5 inputs either as a 0/non zero register or bitwise
11	INVERT	100mSec	Invert the input either value or bitwise
12	SET - RESET	100mSec	Latch an input edge until reset line goes high.

Switch and timer			
20	Analog Selector	100mSec	Select from up to 4 inputs which value is set to the output
21	Delay	100mSec	Select from the delay mode to either delay On, Off or both.
22	Runtime	100mSec	Count runtime of input and divide by value to give output runtime.
23	Totalizer	100mSec	Count input pulses and multiply by unit to give totalized output value.
24	Minimum On / Off	100mSec	Set the Minimum On and Minimum Off times to prevent short cycling.
Convert			
40	AV-MSV	100mSec	Convert an analog value to multistate value or multistate value to analog
41	MSV-AV	100mSec	Convert a multistate value to analog value.
42	MSV-BV	100mSec	Convert MSV value from 1 to 5 to binary bits 0 to 4.
43	Transfer	100mSec	Transfer register from input to output. Register types are not set. Value is converted from input type to output type.
Control			
60	On / Off	100mSec	Direct or reverse acting on/off control
61	PID	100mSec	PID control
62	3Point or Floating	100mSec	3 Point control with runtime count and synchronization. Uses 2 x digital output
63	FAN	100mSec	Up to 4 fan speed or variable speed drive0 to 100 % with enable and rundown timer
64	Latch	100mSec	Check input against a high and low limit and set a high and low multistate output. Clear latched output will set the output to either clear or acknowledged.
Math			
80	Math	100mSec	2 input min/max/average/add/subtract/multiply/divide. Mode determines the math function.
81	Limit	100mSec	Limit
82	Linear	100mSec	Use 2 points on the X axis and 2 points on the Y axis to determine the output of a linear equation.
Console			
90	commands	100mSec	Use commands as a string followed by the time between command runs.

Communication - Modbus Client/Listener Block

Read Register - Index 1

Read 1 or more registers with data format from the server address at the baud rate. Convert to little endian and save at the address in the register table.

Name	Param Index	Param Data Type	Required	Description
Port	0	uint8_t	Yes	RS485 Physical Port number (1 or 2) that the device is attached.
Device Address	1	uint8_t	Yes	Server Device Address. 1-255
Device Baud Rate	2	int32_t	No	Server Device Baud rate. 4800, 9600, 19200, 38400, 57600, 76800, 115200 If using in passive listener mode. This baud rate setting is ignored, and listener will operate at baud specified by register table value
Device Register Address	3	uint16_t	Yes	Register (Start) Address to read from.
Device Register Type	4	uint8_t	Yes	Register Type. See Device **Register Type**
Swap Flags	5	uint16_t	No	Swap Flag Bit Mask: Default 2047. See Swap Flags
Destination Address	6	uint16_t	Yes	Address in Register table in which to store the data.
Input Data Type	7	uint8_t	Yes	Device Point type - Width and how the data is to be interpreted. See Data Types
Output Data Type	8	uint8_t	No	Destination address Data type - Width and how the data is to be interpreted. 0 means out data type follows Input Data Type. See Data Types
ScanRate_S	9	uint32_t	Yes	Interval in seconds between which the server register address is polled. No Effect on Passive Listener
Scale	10	float	No	Value by which to scale the raw data
Shift	11	float	No	Value by which to shift the raw data
Alarm Low Threshold	12	float	No	Minimum value for alarm. Must go below this for alarm to trigger. Must have changed by this amount for a period of Alarm Time Delay (seconds). Leave empty if not used.
Alarm High Threshold	13	float	No	Maximum value for alarm. Must go above this for alarm to trigger. Must have changed by this amount for a period of Alarm Time Delay (seconds). Leave empty if not used.
Alarm Delta Threshold	14	float	No	Change of Value (COV) alarm. Must have changed by this amount for a period of Alarm Time Delay (seconds). Leave empty if not used.
Alarm Time Delay	15	uint16_t	No	Value must be outside the alarm condition for this many seconds before alarm is triggered. Leave empty if not used.

Write Register – Index 2 (Not applicable to Passive Listener)

Write 1 or more registers with data format to the server address at the baud rate. Get write value at the address in the register table.

Name	Param Index	Param Data Type	Required	Description
Port	0	uint8_t	Yes	RS485 Physical Port number (1 or 2) that the device is attached.
Device Address	1	uint8_t	Yes	Server Device Address. 1-255
Device Baud Rate	2	int32_t	No	Server Device Baud rate. 4800, 9600, 19200, 38400, 57600, 76800, 115200
Device Register Address	3	uint16_t	Yes	Register (Start) Address to write to.
Device Register Type	4	uint8_t	Yes	Register Type
Swap Flags	5	uint16_t	No	Swap Flag Bit Mask: Default 2047
Source Address	6	uint16_t	Yes	Address in Register table in which to retrieve the data.
Output Data Type	7	uint8_t	No	Server Destination register address Data type - Width and how the data is to be interpreted.
ScanRate_S	8	uint32_t	Yes	Interval in seconds between which the server register address is written to.

Parameter Details

Device Register Type

Modbus devices make their data available in the form of different register types.

Register Type	Value	Description
Coil	1	Usually a digital output. (Read/Write)
Discrete Input	2	Usually a digital input. (Read only)
Holding Register	3	Usually, an internal value or analogue output (Read/Write at discretion of implementation on the device)
Input Register	4	Usually an analogue input (Read only)

Swap Flags

Different Modbus devices may encode data differently due to architectural differences (endianness), or implementation differences. Byte & word swap flags allow us to configure the driver to swap different parts of the raw binary data we receive from the device to account for these differences.

Valid Range: 0 to 2047

Default: 2047

Create the Swap Flag value by adding the desired swap values from below:

Swap Flag	Value
INT16 BYTE	1
INT32 BYTE	2
INT32 WORD	4
INT64 BYTE	8
INT64 WORD	16
INT64 DOUBLE WORD	32
FLOAT / REAL BYTE	64
FLOAT / REAL WORD	128
DOUBLE BYTE	256
DOUBLE WORD	516
DOUBLE DOUBLE WORD	1024

Data Types

Modbus registers can represent their data in many formats. When data is received, our Modbus

driver can interpret the raw binary information in one of the following ways:

Data Type (Input & Output)	Value
Unsigned 8-bit integer (0 - 65,535)	1
Unsigned 16-bit integer (0 - 65,535)	2
Unsigned 32-bit integer (0 to 4,294,967,295)	3
Unsigned 64-bit integer (0 to 4,294,967,295)	4
Signed 8-bit integer (-32,767 - +32,767)	5
Signed 16-bit integer (-32,767 - +32,767)	6
Signed 32-bit integer (−2,147,483,647 to +2,147,483,647)	7
Signed 64-bit integer (−9,223,372,036,854,775,807 to +9,223,372,036,854,775,807)	8
Single Precision Floating Point Number (IEEE 754 - 32-bit)	9
Double Precision Floating Point Number (IEEE 754 - 64-bit)	10

Communication - DLMS Client Block

Device Configuration – Index 3

This control block index will allow for the setup of a single DLMS server device.

Note: If this block is not defined, *control block index 4 (DLMS endpoints)* block will be invalid, as they require a parent device to link back to.

Name	Param Index	Param Data Type	Required	Description
Port	0	uint8_t	Yes	RS485 Physical Port number (1 or 2) that the device is attached.
Device Baud Rate	1	int32_t	No	Device Baud rate. 4800, 9600, 19200, 38400, 57600, 76800, 115200
Parity	2	uint8_t	No	Device Parity
Data Bits	3	uint8_t	No	Device Data Bits
Stop Bits	4	uint16_t	No	Device Stop Bits
Serial Number	5	uint32_t	Yes	DLMS Device Serial number
Client Address	6	int32_t	Yes	
Server Address Logical	7	int32_t	Yes	
Server Address Physical	8	int32_t	Yes	
Server Address Logical Size	9	int8_t	Yes	
Server Address Physical Size	10	int8_t	Yes	
Max Info Field Size Tx	11	int16_t	No	Default: 512
Max Info Field Size Rx	12	int16_t	No	Default: 512
Max Window Size Tx	13	int32_t	No	Default: 1
Max Window Size Rx	14	int32_t	No	Default: 1
Password	15	string	No	Default: No Authentication
Logical Name Referencing	16	char	Yes	Addressing mechanism (L = Long, S = short)
Date Time Alignment (Hour)	17	uint8_t	No	Default: 0
Date Time Alignment (Minute)	18	uint8_t	No	Default: 0

DLMS Read Endpoint Configuration – Index 4

This control block defines a single DLMS device endpoint.

Note: This block is only valid and used if a parent device block is correctly defined (control block 3).

Name	Param Index	Param Data Type	Required	Description
Serial Number	0	uint32_t	Yes	DLMS Device Serial number – To link back to
Register Address / Logical Name	1	uint64_t	Yes	Note 48-bit value (MAX = 0xFFFFFFFFFFFF = 281,474,976,710,655)
Context / ClassID	2	uint8_t	no	
Attribute	3	uint8_t	no	
CustomTag	4	String	no	Default logical name as string.
Destination Address	5	uint16_t	yes	Address in Register table in which to store the data.
Input Data Type	6	uint8_t	no	Device Point type - Width and how the data is to be interpreted. Default: it will be handles by internal DLMS driver. Leave blank for now
Output Data Type	7	uint8_t	yes	Destination address Data type - Width and how the data is to be interpreted. See **Data Types**
Scale	8	float	No	Default:1. Value by which to scale the raw data.
Shift	9	float	No	Default:0. Value by which to shift the raw data

Alarm Low Threshold	10	float	No	Minimum value for alarm. Must go below this for alarm to trigger. Must have changed by this amount for a period of Alarm Time Delay (seconds). Leave empty if not used.
Alarm High Threshold	11	float	no	Maximum value for alarm. Must go above this for alarm to trigger. Must have changed by this amount for a period of Alarm Time Delay (seconds). Leave empty if not used.
Alarm Delta Threshold	12	float	No	Change of Value (COV) alarm. Must have changed by this amount for a period of Alarm Time Delay (seconds). Leave empty if not used.
Alarm Time Delay	13	uint16_t	no	Value must be outside the alarm condition for this many seconds before alarm is triggered. Leave empty if not used.

Communication – Modbus TCP

Modbus TCP Device Configuration – Index 5

Control block for configuring the IP address.

Name	Param Index	Data Type	Internal Register Type	Description
IP Address	0	String	constant	The IP address used to connect.

Modbus TCP Read Points Configuration – Index 6

Control block for configuring the read points.

Name	Param Index	Param Data Type	Required	Description
cid	0	uint16_t	yes	Characteristic CID
Key name	1	string	yes	Key name of the point
Physical units	2	string	yes	Measurement units
Slave adress	3	uint8_t	no	Slave address of device in segment
parameter Register Type	4	uint8_t	yes	parameter Type. See Device **Register Type**
Register address	5	uint16_t	yes	Modbus register 0 based value
Size of parameter	6	uint16_t	yes	The size of the TCP Parameter
Parameter offset	7	uint16_t	no	The offset of the modbus parameter
Parameter type	8	uint8_t	yes	Parameter type according too **Data Types**
Parameter size	9	uint8_t	yes	The number of bytes in the parameter.
Destination Address	10	uint16_t	yes	Address in Register table in which to store the data.
Output Data Type	11	uint8_t	no	Destination address Data type - Width and how the data is to be interpreted. 0 means out data type follows Input Data Type. See Data Types
ScanRate_S	12	uint32_t	yes	Interval in seconds between which the server register address is polled. No Effect on Passive Listener

Logic

AND / OR / XOR – Index 10

Take up to 5 inputs and perform the logic type operation. Save to the output register. Input 5 is a constant.

Name	Param Index	Data Type	Required	Description

Logic Operation	1	uint8	yes	Logic operation to perform. Refer AND / OR / XOR Logic Operation
Input 1	2	int16	yes	Any register table address.
Input 2	3	int16	yes	Any register table address.
Input 3	4	int16	yes	-1 for unused. Any register table address.
Input 4	5	int16	yes	-1 for unused. Any register table address.
Input 5	6	int32	yes	-1 for unused.
Output	7	uint16	yes	Any register table address in the user space.

AND / OR / XOR Logic Operation

Value	Logic Operation
0	Register AND. Register>0 then register is a logic1.
1	Bitwise AND. Bitwise AND all inputs.
2	Register OR. Register>0 then register is a logic1.
3	Bitwise OR. Bitwise OR all inputs.
4	Register XOR. Register>0 then register is a logic1.
5	Bitwise XOR. Bitwise XOR all inputs.

Initialization error codes
1 Type failed
2 no index found
3 Index Error
4 Parse Error
5 Index out of table range
6 Invalid Logic type
7 Only 1 valid input

INVERT - Index 11

Take the input register and invert as per Invert Logic type.

Name	Param Index	Data Type	Required	Description
Invert Type	1	uint8	yes	Logic operation to perform. Refer INVERT Type
Input	2	uint16	yes	Any register table address.
Output	3	uint16	yes	Any register table address in the user space.

INVERT Type

Value	Logic Operation
0	Register INVERT. If Register > 0 then register is a logic 1.
1	Bitwise INVERT. Invert the register bits

Initialization error codes
1 Type failed
2 no index found
3 Index Error
4 Parse error
5 Index out of table range

SET – RESET – Index 12

Take the input register and invert as per Invert Logic type.

Name	Param Index	Data Type	Required	Description
Input Edge	1	uint8	yes	Input Edge to set on. Refer SET Input
Input	2	uint16	yes	Any register table address.
Reset	3	uint16	yes	Any register table address.
Output	4	uint16	yes	Any register table address in the user space.

SET Input Edge

Value	Logic Operation
0	Set on Input Leading Edge 0 to 1
1	Set on input Trailing Edge 1 to 0

Initialization error codes
1 Type failed
2 no index found
3 Index Error
4 Parse error
5 Index out of table range
6 Edge out of table range

Select, Switch and Timer

Analog Selector – Index 20

Uses a multistate value to select between up to 4 inputs. The output value follows the selected input value. Inputs and output are floats. If an unused input is selected then the output value will be 0.

Name	Param Index	Data Type	Required	Description
Selector	1	uint16	yes	Any register table address.
Input 0	2	uint16	yes	Any register table address.
Input 1	3	uint16	yes	Any register table address.
Input 2	4	int16	yes	-1 for unused. Any register table address.
Input 3	5	int16	yes	-1 for unused. Any register table address.
Output	6	uint16	yes	Any register table address in the user space.

Initialization error codes
1 Type failed
2 no index found
3 Index Error
4 Parse error
5 Index out of table range
6 Invalid Data Type

Delay – Index 21

Checks the input for change and applies a delay as per the mode. Options are delay from off to on. Delay from on to off and delay on both off to on and on to off.

Name	Param Index	Data Type	Required	Description
Mode	1	uint8	yes	Delay Mode. Refer to Delay Mode
Delay Time	2	uint16	yes	Any register table address. Time in Seconds.

Input	3	uint16	yes	Any register table address.
Output	4	uint16	yes	Any register table address in the user space.

Delay Mode

Value	Delay Operation
0	0 Delay on leading edge (0 to > 0).
1	1 Delay on trailing edge. (>0 to 0)
2	2 Delay on both edges

Initialization error codes
1 Type failed
2 no index found
3 Index Error
4 Parse error
5 Index out of table range
6 Invalid Delay type

Runtime – Index 22

Count the runtime in 100mSec increments and divide to give an output runtime in the units required.

Name	Param Index	Data Type	Required	Description
Mode	1	uint8	yes	Runtime mode. Refer to Runtime Mode
Divider	2	uint16	yes	Any register table address. 100mSecond ticks. Example 10 will give a 1 second count. Value of 0 and 1 represent no divider.
Input	3	uint16	yes	Any register table address.
Runtime	4	uint16	yes	Any register table address in the user space. For non-volatile applications use EERAM.

Runtime Mode

Value	Delay Operation
0	Accumulate when input <= 0
1	Accumulate when input > 0

Initialization error codes
1 Type failed
2 no index found
3 Index Error
4 Parse error
5 Index out of table range
6 Invalid Input Edge type

Totalizer – Index 23

Totalize the input pulses.

Name	Param Index	Data Type	Required	Description
Input Edge	1	uint8	yes	Totalize mode. Refer to Totalizer Mode
Units	2	uint16	yes	Any register table address. Number of units that represent each edge transition (pulse). Value of 0 will not totalize any units.

Input	3	uint16	yes	Any register table address.
Totalizer	4	uint16	yes	Any register table address in the user space. For non-volatile applications use EERAM.

Totalizer Mode

Value	Delay Operation
0	Totalize on input leading edge. Input <=0 to >0
1	Totalize on input trailing edge. Input > to <=0

Initialization error codes
1 Type failed
2 no index found
3 Index Error
4 Parse error
5 Index out of table range
6 Invalid Input Edge type

Minimum On / Off – Index 24

Index 24

Minimum On and Minimum Off time control. Do not allow the output to be on or off until the minimum time has elapsed.

Name	Param Index	Data Type	Required	Description
Input	1	uint16	yes	Any register table address.
MinOff	2	int16	yes	Any valid register table address. -1 for unused (ignore min off). Seconds.
MinOn	3	int16	yes	Any valid register table address. -1 for unused (ignore min off). Seconds.
Output	4	uint16	yes	Any register table address in the user space.

Initialization error codes
1 Type failed
2 no index found
3 Index Error
4 Parse error
5 Index out of table range
6 Invalid Data Type

Convert

AV-MSV – Index 40

Convert analog value to 6 step multistate value.

Name	Param Index	Data Type	Required	Description
Analog Value	1	uint16	yes	Any register table address.
Value1	2	uint16	yes	Any register table address.
Value2	3	uint16	yes	Any register table address. -1 if unused.
Value3	4	uint16	yes	Any register table address. -1 if unused.
Value4	5	uint16	yes	Any register table address. -1 if unused.
Value5	6	uint16	yes	Any register table address. -1 if unused.
Value6	7	uint16	yes	Any register table address. -1 if unused.
MSV	8	uint16	yes	Any register table address in the user space.

Analog to MSV. IP to add diagrams.

- MSV off if value < value 1 - MSV = 0
- Step1 on if value >= value 1 - MSV = 1
- Step2 on if value >= value 2 - MSV = 2
- Step2 on if value >= value 3 - MSV = 3
- Step3 on if value >= value 4 - MSV = 4
- Step4 on if value >= value 5 - MSV = 5
- Step5 on if value >= value 6 - MSV = 6

Initialization error codes
1 Type failed
2 no index found
3 Index Error
4 Parse error
5 Index out of table range
6 Invalid Delay type

MSV-AV – Index 41

Convert up to 6 step multistate value to an analog value.

Name	Param Index	Data Type	Required	Description
MSV	1	uint16	yes	Any register table address.
Step0	2	uint16	yes	Any register table address.
Step1	3	uint16	yes	Any register table address. -1 if unused then value selected by MSV result will be 0.
Step2	4	uint16	yes	Any register table address. -1 if unused then value selected by MSV result will be 0.
Step3	5	uint16	yes	Any register table address. -1 if unused then value selected by MSV result will be 0.
Step4	6	uint16	yes	Any register table address. -1 if unused then value selected by MSV result will be 0.
Step5	7	uint16	yes	Any register table address. -1 if unused then value selected by MSV result will be 0.
Analog Value	8	uint16	yes	Any register table address in the user space.

MSV to Analog

- if MSV <= 0 output is Step0 value
- if MSV = 1 output is Step1 value. Value is 0 if step unused
- if MSV = 2 output is Step2 value. Value is 0 if step unused
- if MSV = 3 output is Step3 value. Value is 0 if step unused
- if MSV = 4 output is Step4 value. Value is 0 if step unused
- if MSV >= 5 output is Step5 value. Value is 0 if step unused

Initialization error codes
1 Type failed
2 no index found
3 Index Error
4 Parse error
5 Index out of table range
6 Invalid Delay type

MSV-BIT – Index 42

Convert up to 6 step multistate value to a bit value. This allows for Multistate values to trigger digital actions.

Name	Param Index	Data Type	Required	Description
MSV	1	uint16	yes	Any register table address.
Bit Value	2	uint16	yes	Any register table address in the user space.

MSV to Bit Value

0	0000 0000
1	0000 0001
2	0000 0010
3	0000 0100
4	0000 1000
5	0001 0000
6	0010 0000

Initialization error codes
1 Type failed
2 no index found
3 Index Error
4 Parse error
5 Index out of table range
6 Invalid Delay type

Transfer - Index 43

Transfer from register to register. Convert data types. This allows transfers from System register table space. Use carefully when working in the user space.

Name	Param Index	Data Type	Required	Description
Input	1	uint16	yes	Any register table address.
Output	2	uint16	yes	Any register table address.

Registers with undefined data types will not be transferred.

Initialization error codes
1 Type failed
2 no index found
3 Index Error
4 Parse error

Control

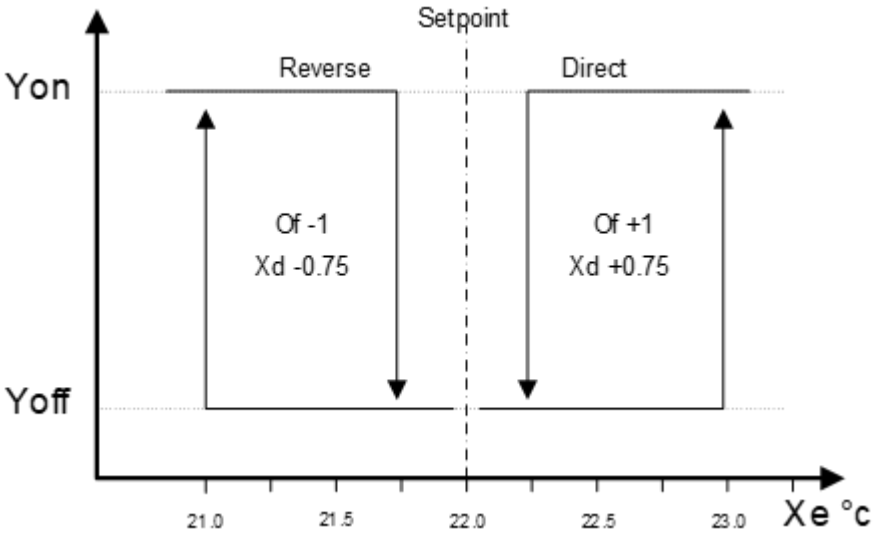
On / Off - Index 60

Direct and reverse acting on/off block

Name	Param Index	Data Type	Required	Description
Xe	1	uint16	yes	Any register table address. Measured value or control input.
w	2	uint16	yes	Any register table address. Setpoint.
Of	3	unit16	yes	Any register table address. Offset from Setpoint. Allows for a common setpoint to be used by more than 1 On/Off block.
Xd	4	uint16	yes	Any register table address. Switching differential. Negative for reverse acting.
y	5	uint16	yes	Any register table address in the user space. Output value.

Initialization error codes
1 Type failed
2 no index found
3 Index Error
4 Parse error
5 Index out of table range

Diagram shows 2 separate blocks.
1 for direct and 1 for reverse acting.



Settings for direct acting (output on with increase input) are:

- Setpoint (w) 22.0
- Offset (Of) +1.0
- Switching differential or Hysteresis +.75

Calculation:

- Switch on at w + Of = 22 + 1 = 23
- Switch off at w + Of – Xd = 23 – 0.75 = 22.25

Settings for reverse acting (output on with decrease input) are:

- Setpoint (w) 22.0
- Offset (Of) -1.0
- Switching differential or Hysteresis -0.75

Calculation:

- Switch on at w + Of = 22 + (-1) = 21
- Switch off at w + Of – Xd = 21 – (-0.75) = 21.75

PID – Index 61

PID control. Dual output.

Name	Param Index	Data Type	Required	Description
Xe	1	uint16	yes	Any register table address. Measured value or control input.
w	2	uint16	yes	Any register table address. Setpoint.
Pgain	3	uint16	yes	Any register table address. Proportional gain. Set to 100 for proportional only control. Reducing this value reduces the proportional component of the PID error.

TI	4	int16	yes	Any register table address. Integral time. Error is divided by TI and will take Scan x TI to go from 0% to 100%. -1 = not used.
Dgain	5	int16	yes	Any register table address. Derivative gain. -1 = not used.
Scan	6	uint8	yes	Loop time in 100mSec ticks. (0 or 1 will scan every 100mSeconds).
Out Enable	7	int16	yes	Any register table address. Output Enable - Yplus/Yminus will control. 0 Yplus/Yminus off. -1 = not used
yPlus	8	uint16	yes	Output value for increasing Xe (Heat). Any register table address in the user space.
yMinus	9	uint16	yes	Output value for decreasing Xe (Cool). Any register table address in the user space.

Initialization error codes

1 Type failed

2 no index found

3 Index Error

4 Parse error

5 Index out of table range

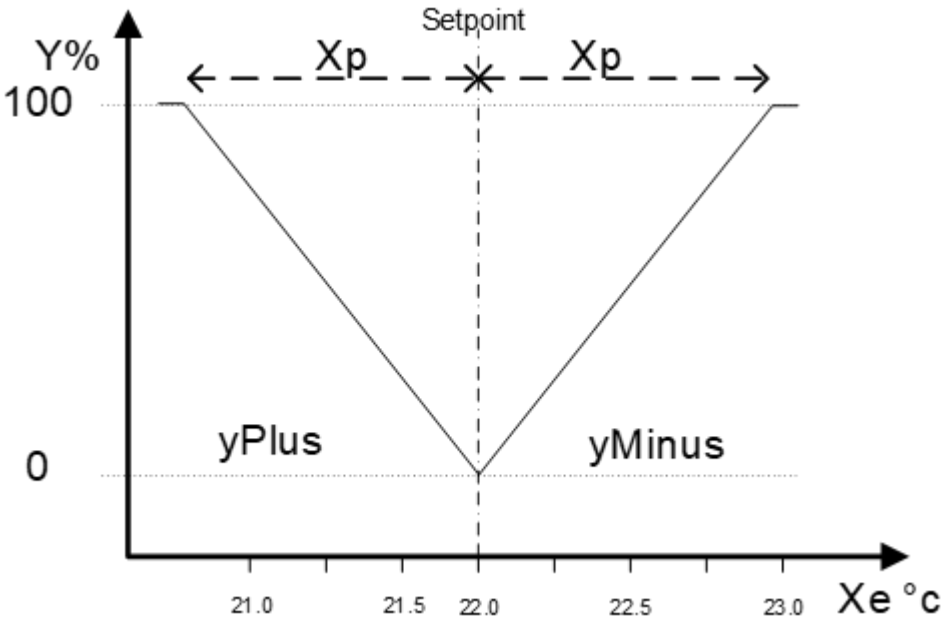


Diagram shows proportional control only. To achieve accurate control use PI or PID.

Refer to the training manual for correct application.

3Point or Floating – Index 62

3Point or floating control – 0 to 100% input is matched to the runtime to the input %. Designed to be used with the PID control block for accurate control of FCU and VAV 3 point actuators.

Name	Param Index	Data Type	Required	Description
Xe	1	uint16	yes	Any register table address. Measured value or control input.
Enable	2	int16	yes	Any register table address. -1 = not used. 0= disable >0 = enable. Does not lock out Synch.
Synch	3	int16	yes	Any register table address. Synch on leading edge of register from 0 to > 0. Drive the motor 1.2 times runtime to synchronize the open or close position to prevent drift. -1 not used
Synch Pos	4	int8	yes	0 for drive to yMinus. 1 for Drive to yPlus. -1 not used
Runtime	5	uint16	yes	Any register table address. Runtime register in seconds.
yPlus	6	uint16	yes	Any register table address in the user space. Output value for increasing runtime.
yMinus	7	uint16	yes	Any register table address in the user space. Output value for decreasing runtime.

Initialization error codes
1 Type failed
2 no index found
3 Index Error
4 Parse error
5 Index out of table range

FAN - Index 63

Index 63

Control block for Fan Coil Unit. Provides a universal interface for controlling either a step (up to 4) or VSD fan matched to a thermostat (wall unit) if required. Thermostats come in many variants which are catered for by the various modes. SpeedX means a multispeed multistate value. Supports up to 4 fan speed with rundown timer and output interlock.

Name	Param Index	Data Type	Required	Description
Mode	1	uint8	yes	Refer to Fan mode table below.
Manual Speed	2	int16	yes	Any register table address. Analog or MSV manual fan control. Refer to Fan Mode. -1 not used
Auto Speed	3	int16	yes	Any register table address. Analog or MSV auto fan control. Refer to Fan Mode. -1 not used. For relay mode 0 means off.
Min Lock Speed	4	int16	yes	Any register table address. Variable Speed only. Minimum speed to enable the output control. If either Auto or Manual speed is below this limit the fan will run at this value. Used as a safety to prevent fan from running too slow for heater elements. -1 = not used
On/Off	5	int16	yes	Any register table address. Fan Control. 0= Off >0 = On. Input to control the unit. -1 not used (also used with Modes where fan is controlled by Manual Speed)
Rundown	6	uint16	yes	Any register table address. Rundown Time in Seconds.
Output Enable	7	uint16	yes	Any register table address. 0= disable >0 = enable. Output to lock heat/cool during rundown and off.
Out1	8	uint16	yes	Any register table address in the user space. Output for variable speed fan or fixed speed step 1.

Out2	9	int16	yes	Any register table address in the user space. Output for fixed (relay) speed step 2. -1
Out3	10	int16	yes	Any register table address in the user space. Output for fixed (relay) speed step 3. -1 for variable speed or not used.
Out4	11	int16	yes	Any register table address in the user space. Output for fixed (relay) speed step 4. -1 for variable speed or not used.

FAN Mode

Fan control block has several mode options as detailed in the table below.

The step breaks in relay mode are set to:

1. Step 1: < 30%
2. Step 2: >=30% < 60%
3. Step 3: >=60% < 90%
4. Step 4: >=90%

MODE0 – Step Speed.
MODE3 - Variable speed
Used for thermostat with separate On/Off button and Speed buttons that cycle between speeds – No Auto speed. The speed button does not switch fan off. Use MSV/AV block to match thermostat MSV to the required speed as per step breaks above. Requires On/Off to be used for On/Off Control. If the thermostat has a power button then link this to On/Off input.

INPUT	Note	Output Enable – Logic AND	If output Enable SPEED	Runtime Speed
Manual Speed	Cannot be -1. This is the control value.	>0	Follows MSV	Last Speed
Auto Speed	-1 the value is ignored			
Minimum Speed	if -1 then value is ignored		if enabled minimum fan speed	
On/Off	if -1 then value is ignored. Cannot be -1 if Manual speed does not have 0 value	>0		

MODE1 - Step Speed
MODE4 - Variable speed
Used for thermostat with separate On/Off button and Speed buttons that cycle between speeds with an Auto speed. The speed button does not switch fan off. Use MSV/AV block to match thermostat MSV to the required speed as per step breaks above with Auto Speed = 0. Requires On/Off to be used for On/Off Control. If the thermostat has a power button then link this to On/Off input.

INPUT	Note	Output Enable – Logic AND	SPEED	Runtime Speed
Manual Speed	Cannot be -1. This is the control value.		if MSV > 0 then follow	Last Speed
Auto Speed	Cannot be -1.		if MSV = 0 then follow	Last Speed
Minimum Speed	-1 the value is ignored		if enabled minimum fan speed	
On/Off	Cannot be -1. This enables the unit	>0		

MODE2 - Step Speed
MODE5 – Variable Speed
No Thermostat

INPUT	Note	Output Enable – Logic AND	SPEED	Runtime Speed
Manual Speed	-1.			
Auto Speed	Cannot be -1.		follow	Last Speed

Minimum Speed	-1 the value is ignored		if enabled minimum fan speed	
On/Off	Cannot be -1. This enables the unit	>0		

Initialization error codes
1 Type failed
2 no index found
3 Index Error
4 Parse error
5 Index out of table range
6 Invalid mode

Latch – Index 64

Control block for alarm management. Latch the high output to indicate the input has exceeded the high value. Latch the low output to indicate the input has gone below the low value. Use the Acknowledge input to clear the output. If the input is still outside of the limits, the output will change to an Acknowledge state and automatically clear once the input returns within limits.

Name	Param Index	Data Type	Required	Description
Input	1	uint16	yes	Any register table address.
Acknowledge	2	uint16	yes	Any register table address. Acknowledge Input. Acknowledge is accepted if the value > 0.
Hi Limit	3	uint16	yes	Any register table address. Hi latch value. -1 not used.
Hi Normal	4	uint16	yes	Any register table address. Hi return to normal value. Value must be lower than Hi Limit. -1 not used.
Low Limit	5	uint16	yes	Any register table address. Low latch value. -1 not used.
Low Normal	6	uint16	yes	Any register table address. Low return to normal value. Value must be higher than Low Limit. -1 not used.
Hi output	7	uint16	yes	Any register table address. Hi latch. Multistate value. 0=normal, 1 = unacknowledged, 2 acknowledged. -1 not used.
Low output	8	uint16	yes	Any register table address. Low latch. Multistate value. 0=normal, 1 = unacknowledged, 2 acknowledged. -1 not used.

Note – if Hi Limit is not used (-1) then Hi Normal and Hi Output will be ignored. The same for Lo Limit.

Initialization error codes
1 Type failed
2 no index found
3 Index Error
4 Parse error
5 Index out of table range

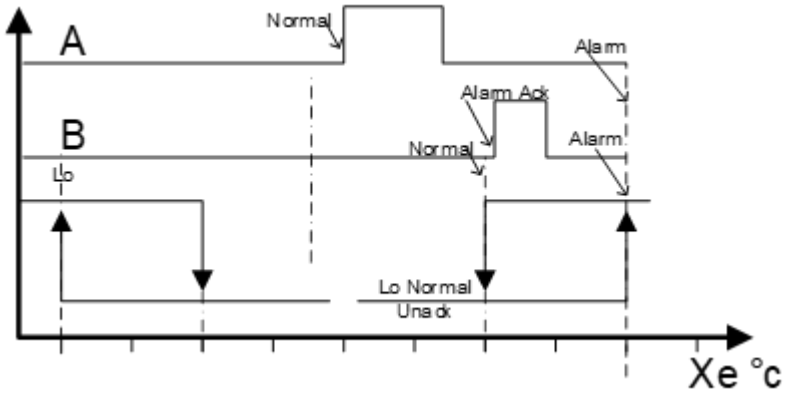


Diagram timeline A shows if input goes above Hi Limit, Hi Output will change to 1 (Alarm). If input goes below Hi Normal, Hi Output will remain 1 (Alarm). When a value > 0 is detected on the Acknowledge input, the output will change to 0 (Normal).

Diagram timeline B shows. if input goes above Hi Limit Hi Output will change to 1 (Alarm). When a value > 0 is detected on the Acknowledge input while in alarm, the output will change to 2 (Acknowledged). If input goes below Hi Normal, Hi Output will change to 0 (Normal).

The comparison uses >= and <= which means the outputs will switch at the set values.

Math

Math 1 – Index 80

Control block for basic maths operations. All registers are typecast to float then returned to their defined types. Remember this when error checking for incorrect results.

Name	Param Index	Data Type	Required	Description
Operation	1	uint8	yes	Maths operation to perform. Refer Math Operation
Operand 1	2	Uint16	yes	Any register table address.
Operand 2	3	Uint16	yes	For register type register math operation. Any register table address. For register type constant math operation.
Result	4	Uint16	yes	Any register table address in the user space.

Math Operation

Value	Operation
0	Minimum. Operand 1 and 2 registers
1	Maximum. Operand 1 and 2 registers
2	Average. Operand 1 and 2 registers
3	Add. Operand 1 and 2 registers
4	Subtract. Operand 1 and 2 registers
5	Multiply. Operand 1 and 2 registers
6	Divide. Operand 1 and 2 registers
7	Minimum. Operand 1 register Operand2 constant value
8	Maximum. Operand 1 register Operand2 constant value
9	Average. Operand 1 register Operand2 constant value
10	Add. Operand 1 register Operand2 constant value
11	Subtract. Operand 1 register Operand2 constant value
12	Multiply. Operand 1 register Operand2 constant value
13	Divide. Operand 1 register Operand2 constant value

Initialization error codes
1 Type failed
2 no index found
3 Index Error
4 Parse error
5 Index out of table range
6 Invalid operation type
7 Invalid data type

Limit – Index 81

Control block for limiting range.

Name	Param Index	Data Type	Required	Description
Input	1	uint16	yes	Any register table address. Input. Any register table in the user or system volatile space.
Hi Limit	2	int16	yes	Any register table address. -1 = not used
Lo Limit	3	int16	yes	Any register table address. -1 = not used
Output	4	uint16	yes	Any register table address in the user space.

Initialization error codes
1 Type failed
2 no index found
3 Index Error
4 Parse error
5 Index out of table range

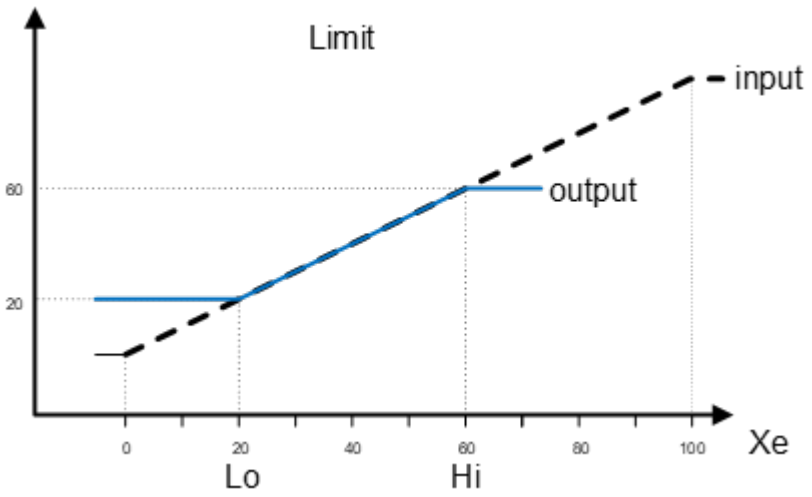


Diagram shows Lo limit at 20 and Hi limit at 60. Result on Y axis will follow the input on X axis but will not move below 20 or above 60.

Examples.

- input 0. Output 20
- Input 10. Output 20
- Input 50. Output 50
- Input 80. Output 60

Linear – Index 82

Control block for applying a linear equation between two points.

Name	Param Index	Data Type	Required	Description
Register allocation	1	uint8	yes	0 means x1, x2, y1 and y2 values are constant. >0 means the values are from the register table.
Input	2	uint16	yes	Any register table address.
x1	3	uint16	yes	Any register table address. Lower value on x axis. Must be lower than x2.
x2	4	uint16	yes	Any register table address.
y1	5	uint16	yes	Any register table address. Value on y axis required for x1.
y2	6	uint16	yes	Any register table address. Value on y axis required for x2.
Output	7	uint16	yes	Any register table address in the user space.

Initialization error codes
1 Type failed
2 no index found
3 Index Error
4 Parse error
5 Index out of table range

Console

Commands – Index 90

Control block for running console commands periodically. Example: 90,[riv 341,30]

Name	Param Index	Data Type	Required	Description
Console command	1	String	yes	Type console command out as string. ex: riv 341.
Time	2	uint16	yes	Type time out.