Ergänzende Information / Additional Information			
Artikel-Nr./Stock No.	X9751375.01	Auftrags-Nr./Order No.	
Produkt/Product	colorSENSOR CFO Option 100	Rahmen-Nr./B-Order No.	
Basis-Betriebsanleitung/Main Manual		colorSENSOR CFO - X9751375	

MICRO-EPSILON

Modbus Documentation for colorSENSOR CFO Option 100

1. Introduction

The modbus protocol is a single-master protocol. Data is exchanged over a serial or via network (TCP/IP) interface. The Controller acts as a Modbus slave: It responds to requests from a master.

The Modbus protocol allows partial access to the most relevant features of the controller. Internally it uses the HTTP-based API of the controller for all operations.

The colorSENSOR Modbus interface supports the following protocol features:

- Transport via TCP (IPv4 and IPv6)
- Transport via RS232 and USB using RTU (default) or ASCII format
- Serial baudrates: 9600, 19200 (default), 115200

The Modbus slave address (relevant only for serial connections) is configurable. By default the colorSENSOR CFO is not bound to a specific address, but responds to every packet.

The full set of supported commands is available as a JSON dump. This structured dataset is supposed to ease the generation of a vendor-specific Modbus mapping for the controller.

2. Quickstart

The following configuration details and hints should ease the first steps with the controllers Modbus protocol implementation:

- Connect to the sensors Modbus protocol via RS232 (Baudrate: 19200), USB or TCP (port 502).
- Use Big-Endian (byte-order and word-order) when interpreting data in Modbus responses.
- Consider the 1-based addressing scheme when accessing registers. For example a documented address of 501 is transmitted over the wire as 500. Most Modbus client implementations will apply this translation implicitly. Only very few implementations use the on-wire address instead. In this case the documented address needs to be decremented for these specific clients.
- Retrieve the Input Registers from 500 up to 508 via a Modbus request, see 5.1.17. These registers contain fixed values in different formats (e.g. float, 32 bit and 64 bit integer). Ensure that your client implementation interpretes these values properly according to their documented value (see the register content documentation). In case of misinterpretations you may need to adjust the endianness or the address offset of your client implementation.

3. Supported colorSENSOR Features

The Modbus interface of the colorsensors provides most features of the following API endpoints:

- /*defaults* (only matcher-releated defaults)
- /*device*
- /*firmware* (only status retrieval; no upgrade)
- /*firmware*/*recovery*
- /*firmware*/*recovery*/*upgrade-from-current*
- /*sensor*/*samples*/*current*
- /*sensor*/*matchers*
- /*sensor*/*detectables*
- /*sensor*/*detection-profiles*
- /*sensor*/*detection-profiles*/*autogain*
- /*sensor*/*detection-profiles*/*white-reference*
- /*sensor*/*capabilities*
- /*system*
- /*system*/*factory-reset*
- /*system*/*reboot*
- /*peripherals*/*outputs*
- /*peripherals*/*rs232*
- /*peripherals*/*usb*
- /*settings*

The following API endpoints are not supported due to the volatile nature of their data or their complexity (hard to express within the modbus protocol):

- */access*
- */action-triggers*
- */actions*
- */firmware/images*
- */firmware/settings*
- */network/interfaces/*
- */peripherals/keypad*
- */peripherals/trigger-sources*
- */system/time*
- */system/time/zones*

4. Data Types and Register Addressing

4.1 Data Types and Modbus Functions

The Modbus protocol specifies different functions for accessing and manipulating values.

The following functions (and their respective function codes) are used for the different types of data:

Function	Code	Function name
Read-only bits	2	Read Discrete Inputs
Writable bits	1	Read Coils
	5	Write singe coils
	15	Write multiple coils
Read-only words	4	Read input registers
Writable words	3	Read Multiple Holding Registers
	6	Write Single Holding Register
	16	Write Multiple Holding Register
	23	Read/Write Multiple Registers

4.2 Register Addresses

The addressing of data via the Modbus protocol is not strictly specified. Different implementations use a variety of name schemes and offsets. The relevant details of this Modbus implementation are:

- All addresses written in this documentation are register offsets relative to the specific Modbus function.
- All addresses are 1-based. This approach is used by most Modbus implementations.

For example the register for the float test value is documented as a read-only word at address 501. This address could also be written as 30501 (based on a traditional Modbus addressing scheme mapping the functions to specific address ranges). The content of this register can be retrieved with the Read Input Registers function (function identifier „4“). The internal address of this value (as used for the on-wire format of Modbus) is 500 (due to the 1-based register addressing). This internal address is only used by very few Modbus client implementations. Most implementations use the 1-based address, instead.

Clients without support for address offsets may need to decrement every address (as documented here) when assembling the Modbus data frame.

4.3 Simple Data Types

The Modbus specification describes simple data types (bits and 16-bit-words). Additionally the following data types are used by the Modbus implementation of the colorSENSOR:

- Float values: Two registers (32 bit), IEEE-754, big-endian word-order and byte-order
- Integer values with 32 bit (two registers) or 64 bit (four registers): Big-endian word-order and byte-order
- Strings: The first byte contains the length; all following bytes contain the ASCII characters. Thus the first „word“ register contains the length byte and the first character. Each following „word“ register contains two characters. Reading past the end of the string length is allowed and returns null bytes. Thus usually a trailing null byte is present at at the end of the string. But you may not rely on this, as the trailing null byte is missing, if the string uses exactly the maximum number of allowed characters for this string.
- Bitmask: 16 bit words are used to represent or manipulate boolean fields. Each bit represents a single boolean value. The description of each bitmask data field maps bit positions to the boolean state described by this bit. A value of zero is considered to be false (not active). A value of one is true. The bit positions start with zero with the least significant bit.

5. Session State, Concurrency and Multiple Interfaces

Multiple interfaces of the sensor can communicate via the modbus protocol. Each hardware interface (e.g. RS232, USB) manages its own state. This is relevant for stateful operations (e.g. access to a collection), that require a sequence of read or write requests. The Ethernet interface accepts TCP connections. Each connection tracks its own state for the duration of the connection.

5.1 Functions

5.1.1 Autogain Procedure API Endpoint: /sensor/detection-profiles/current/autogain

Execute the autogain procedure in order to determine suitable sampling properties for the current optical environment. The resulting sampling setup is applied automatically. These new settings are in effect as soon as the response is sent. The success or failure of an autogain procedure can be verified as soon as the autogain_is_running flag is cleared.

Address	Type	Operation	Description		FC
00020	Bit	write	Start an autogain procedure		5, 15
00302	Bitmask	read	Status of the most recently started autogain procedure		4
			Position	**Description**	
			0	Is still running	
			1	Finished successfully	
			2	Failed: Target is too dark	
00410	Float	read / write	Minimum wanted sample rate		3, 4, 6, 16
00412	Float	read / write	Target analog input level		3, 4, 6, 16
00414	Uint16	read / write	Number of samples used for averaging		3, 4, 6, 16
00415	Bitmask	read / write	Boolean flags for autogain procedure Default value: 65535		3, 4, 6, 16
			Position	**Description**	
			0	Enable internal emitter	
			1	Enable ambient tight compensation	
00416	Bitmask	read / write	Override default autogain settings with custom values		3, 4, 6, 16
			Position	**Description**	
			0	Overwrite minimum wanted sample rate	
			1	Overwrite target analog input level	
			2	Overwrite number of samples used for averaging	

5.1.2 White reference API Endpoint: /sensor/detection-profiles/current/white-reference

The white reference is used for calculating accurate color positions in the colorspaces. The factory default white reference is suitable for a special set of sensor and optics. A custom white reference can be sampled. A reference white target is recommended for this.

Address	Type	Operation	Description	FC
00021	Bit	write	Reset the custom white reference	5
00022	Bit	write	Sample a custom white reference	5

5.1.3 Add Color to Color Table API Endpoint: /sensor/matchers

Address	Type	Operation	Description	FC
00024	Bit	write	Create a new matcher and assign the current color position to it (as a detectable).	5, 15
00451	Uint16	read	Retrieve the identifier of the most recently created matcher.	4

5.1.4 Manage Color Positions of a Color Group API Endpoint: /sensor/matchers

Each color group (matcher) may refer to one or more color positions (detectables).

Address	Type	Operation	Description	FC
00025	Bit	write	Add a new detectable to an existing matcher (color group).	5
00026	Bit	write	Delete all detectables of an existing matcher (color group).	5
00027	Bit	read	Indicate whether the currently selected matcher exists.	1
00311	Uint16	read	Current number of detectables (color positions) assigned to the matcher.	4
00450	Uint16	read / write	Specify the matcher (color group) when adding or removing detectables (color positions).	3, 6

5.1.5 Read Sensor Capabilities API Endpoint: /sensor/capabilities

Inspect the available features of the controller.

<table>
<tr><th>Address</th><th>Type</th><th>Operation</th><th colspan="2">Description</th><th>FC</th></tr>
<tr><td>00300</td><td>Uint16</td><td>read</td><td colspan="2">Number of available switching outputs</td><td>4</td></tr>
<tr><td rowspan="7">00301</td><td rowspan="7">Bitmask</td><td rowspan="7">read</td><td colspan="2">Colorspaces supported by the sensor</td><td rowspan="7">4</td></tr>
<tr><td><b>Position</b></td><td><b>Description</b></td></tr>
<tr><td>0</td><td>XYZ</td></tr>
<tr><td>1</td><td>L*a*b*</td></tr>
<tr><td>2</td><td>xyY</td></tr>
<tr><td>3</td><td>L*u*v*</td></tr>
<tr><td>4</td><td>L*u’v’</td></tr>
<tr><td rowspan="6">00303</td><td rowspan="6">Bitmask</td><td rowspan="6">read</td><td colspan="2">Available tolerance shapes</td><td rowspan="6">4</td></tr>
<tr><td><b>Position</b></td><td><b>Description</b></td></tr>
<tr><td>0</td><td>Infinite (classification)</td></tr>
<tr><td>1</td><td>Sphere</td></tr>
<tr><td>2</td><td>Cylinder</td></tr>
<tr><td>3</td><td>Box</td></tr>
<tr><td rowspan="6">00304</td><td rowspan="6">Bitmask</td><td rowspan="6">read</td><td colspan="2">Available switching output drivers</td><td rowspan="6">4</td></tr>
<tr><td><b>Position</b></td><td><b>Description</b></td></tr>
<tr><td>0</td><td>Disabled</td></tr>
<tr><td>1</td><td>NPN</td></tr>
<tr><td>2</td><td>PNP</td></tr>
<tr><td>3</td><td>Push-Pull</td></tr>
<tr><td>00305</td><td>Float</td><td>read</td><td colspan="2">Maximum sample rate</td><td>4</td></tr>
<tr><td>00307</td><td>Uint16</td><td>read</td><td colspan="2">Maximum number of detectables</td><td>4</td></tr>
<tr><td>00308</td><td>Uint16</td><td>read</td><td colspan="2">Maximum number of matchers</td><td>4</td></tr>
</table>

5.1.6 Get Current Sample API Endpoint: /sensor/samples/current

Retrieve the latest color detection sample. A single read operation covering the complete memory range of the sample is guaranteed to be consistent. Multiple read operations in series will probably result in a combination of values from the different samples gathered during the time between the first and the last request.

Address	Type	Operation	Description	FC
00150	Uint64	read	Timestamp of the current sample	4
00154	Float	read	Signal level of the current sample	4
00156	Float	read	Representation of the color in the XYZ colorspace (X)	4
00158	Float	read	Representation of the color in the XYZ colorspace (Y)	4
00160	Float	read	Representation of the color in the XYZ colorspace (Z)	4
00162	Float	read	Representation of the color in the currently active colorspace L	4
00164	Float	read	Representation of the color in the currently active colorspace a	4
00166	Float	read	Representation of the color in the currently active colorspace b	4
00168	Float	read	Representation of the color as RGB values red (between 0.0 and 1.0)	4
00170	Float	read	Representation of the color as RGB values green (between 0.0 and 1.0)	4
00172	Float	read	Representation of the color as RGB values blue (between 0.0 and 1.0)	4
00174	Uint16	read	Inputs with a high level event during the last sample period (bit 0 -> IN0)	4
00175	Uint16	read	Inputs with a low level event during the last sample period (bit 0 -> IN0)	4
00176	Uint16	read	Inputs with a rising edge event during the last sample period (bit 0 -> IN0)	4
00177	Uint16	read	Inputs with a falling edge event during the last sample period (bit 0 -> IN0)	4
00178	Uint16	read	ID of the closest matcher in range of the last sample‘s color position. The value 65535 is returned if the sampled color position was not in range of any of the available matchers.	4
00179	Uint16	read	Currently active state of the Switching Outputs (bit 0 -> OUT0)	4
00180	Float	read	Distance (based on the axes of the currently configured colorspace) between the last sampled color position and the closest suitable matcher (if any). A negative value (-1) indicates that no matcher is in range. Distance 1	4

Address	Type	Operation	Description	FC
00182	Float	read	Distance (based on the axes of the currently configured colorspace) between the last sampled color position and the closest suitable matcher (if any). A negative value (-1) indicates that no matcher is in range. Distance 2	4
00184	Float	read	Distance (based on the axes of the currently configured colorspace) between the last sampled color position and the closest suitable matcher (if any). A negative value (-1) indicates that no matcher is in range. Distance 3	4

5.1.7 Status of the Color Table API Endpoint: /sensor/dtection-profiles/current

Retrieve the current usage of the color table

Address	Type	Operation	Description	FC
00309	Uint16	read	Current number of matchers (color groups) stored in the color table	4
00310	Uint16	read	Current number of detectables (color positions) stored in the color table	4

5.1.8 Clear Color Table API Endpoint: /sensor/matchers

Delete all colors that are stored in the color table.

Address	Type	Operation	Description	FC
00023	Bit	read	Remove all stored colors	5, 15

5.1.9 Switching Outputs Driver API Endpoint: /peripherals/outputs

Electrical output lines can drive external actors in different electrical modes. The currently active mode can be retrieved and changed.

<table>
<tr><th>Address</th><th>Type</th><th>Operation</th><th colspan="2">Description</th><th>FC</th></tr>
<tr><td rowspan="6">00400</td><td rowspan="6">Uint16</td><td rowspan="6">read / write</td><td colspan="2">Retrieve and change the current switching output driver. Possible values:</td><td rowspan="6">3, 4, 6, 16</td></tr>
<tr><td><b>Description</b></td><td><b>Values</b></td></tr>
<tr><td>off</td><td>0</td></tr>
<tr><td>npn</td><td>1</td></tr>
<tr><td>pnp</td><td>2</td></tr>
<tr><td>push-pull</td><td>3</td></tr>
</table>

5.1.10 Firmware Version API Endpoint: /firmware

Read information about the firmware.

Address	Type	Operation	Description	FC
00100	Uint16	read	Firmware Version (Major: X.0.0)	4
00101	Uint16	read	Firmware Version (Major: 0.X.0)	4
00102	Uint16	read	Firmware Version (Major: 0.0.X)	4

5.1.11 Settings Reset API Endpoint: / settings

Reset the controller settings to their factory defaults.

Address	Type	Operation	Description	FC
00006	Bit	write	Reset all settings	5

5.1.12 Factory Reset API Endpoint: /system/factory-reset

Reset the controllers firmware to its factory default and initiate a reboot.

Address	Type	Operation	Description	FC
00002	Bit	write	Trigger a factory reset of the firmware and the settings	5

5.1.13 Reboot the Device API Endpoint: /system/reboot

Trigger a reboot of all controller components

Address	Type	Operation	Description	FC
00001	Bit	write	Trigger a reboot	5

5.1.14 Upgrade Recovery Firmware API Endpoint: /system/factory-reset

Replace the stored recovery image with the current system firmware. This is helpful if you want to update the recovery image to a more recent firmware version.

Address	Type	Operation	Description	FC
00003	Bit	write	Upgrade the recovery firmware to the currently running firmware version	5

5.1.15 RS232 Interface Configuration API Endpoint: /peripherals/rs232

Inspect or change the settings address of the controller for the RS232 interface. Some settings refer to the Modbus slave protocol. The Modbus slave ID is used for serial communication if more than one Modbus device is connected to the same bus. The frame format may be changed according to the needs of the Modbus master.

Address	Type	Operation	Description		FC
00430	Uint16	read / write	Baud rate of RS232 interface		3, 4, 6, 16
			Values	**Description**	
			9600	0	
			19200	1	
			115200	2	
00431	Uint16	read / write	Protocol to be used for the RS232 interface		3, 4, 6, 16
			Values	**Description**	
			eliza	0	
			modbus	1	
00432	Uint16	read / write	Slave ID to be used for the Modbus protocol (1..247)		3, 4, 6, 16
00433	Uint16	read / write	Frame format to be used for the Modbus protocol possible values		3, 4, 6, 16
			Values	**Description**	
			rtu	0	
			ascii	1	

5.1.16 USB Interface Configuration API Endpoint: /peripherals/usb

Inspect or change the settings address of the controller for the USB interface. Some settings refer to the Modbus slave protocol. The Modbus slave ID is used for serial communication if more than one Modbus device is connected to the same bus. The frame format may be changed according to the needs of the Modbus master.

<table>
<tr><th>Address</th><th>Type</th><th>Operation</th><th colspan="2">Description</th><th>FC</th></tr>
<tr><td rowspan="4">00440</td><td rowspan="4">Uint16</td><td rowspan="4">read / write</td><td colspan="2">Protocol to be used for the USB interface possible values</td><td rowspan="4">3, 4, 6, 16</td></tr>
<tr><td><b>Values</b></td><td><b>Description</b></td></tr>
<tr><td>eliza</td><td>0</td></tr>
<tr><td>modbus</td><td>1</td></tr>
<tr><td>00441</td><td>Uint16</td><td>read / write</td><td colspan="2">Slave ID to be used for the Modbus protocol (1..247)</td><td>3, 4, 6, 16</td></tr>
<tr><td rowspan="4">00442</td><td rowspan="4">Uint16</td><td rowspan="4">read / write</td><td colspan="2">Frame format to be used for the Modbus Protocol possible values</td><td rowspan="4">3, 4, 6, 16</td></tr>
<tr><td><b>Values</b></td><td><b>Description</b></td></tr>
<tr><td>rtu</td><td>0</td></tr>
<tr><td>ascii</td><td>1</td></tr>
</table>

5.1.17 Data Format Test API Endpoint: None

Some registers respond with specified fixed values in order to allow clients to verify the correctness of the configured data format easily.

Address	Type	Operation	Description	FC
00500	Uint16	read	A 16 bit integer value containing the number 1234.	4
00501	Float	read	A float value containing the number -1.0.	4
00503	Uint32	read	A 32 bit integer value containing the number 12345678.	4
00505	Uint64	read	A 64 bit integer value containing the number 123456789012.	4

MICRO-EPSILON

MICRO-EPSILON Eltrotec GmbH
Manfred-Wörner-Straße 101 · 73037 Göppingen / Germany
Tel. +49 (0) 7161 / 98872-300 · Fax +49 (0) 7161 / 98872-303
info@micro-epsilon.com · www.micro-epsilon.com
Your local contact: www.micro-epsilon.com/contact/worldwide/