IS4420: I2C/UART Current Loop Receiver

Digital Reading of Current Loop Signals

Applications

- Reading 2-wire, 3-wire and 4-wire 4–20 mA sensors: temperature, pressure, flow, level
- 4–20 mA inputs for PLCs, data loggers and controllers
- Test and calibration tools for 4–20 mA transmitters

Main Advantages

- No analog design: just connect the loop
- Factory calibrated: ready to use out of the box
- Easy to read over I2C or UART, with no library
- Reduces engineering time and costs
- Provides a cost-effective solution

Characteristics

- Measuring Range: 0 to 21.5 mA
- Resolution: 0.01 mA
- Input Resistance: 150 Ω
- Interfaces: I2C and UART
- I2C Addresses: 26 to 30 (up to five modules)
- Supply: 3.3 V, with 5 V-tolerant I2C and UART

General Description

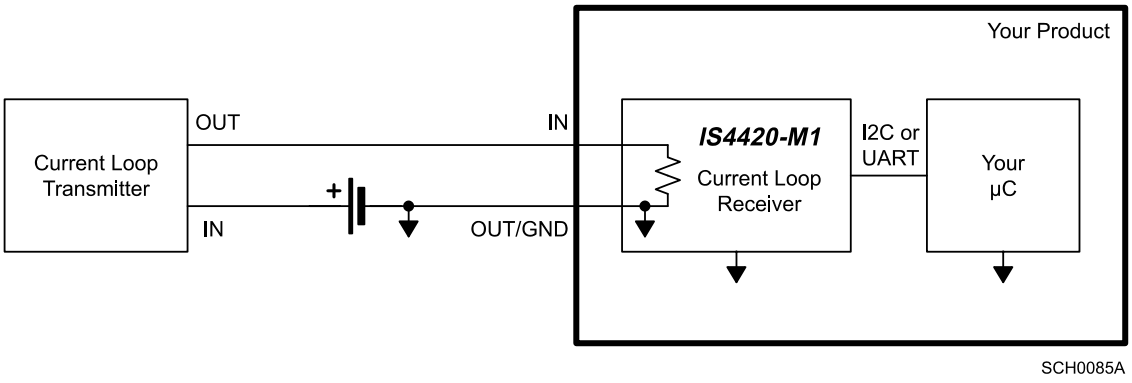
The IS4420-M1 reads a 4–20 mA current loop for you. Connect the loop to the module, and your microcontroller gets the current as a number, in milliamps or as a percentage, over I2C or UART.

There is no analog circuit to design and nothing to calibrate: the module measures, filters and converts the signal, and every unit is calibrated at the factory.

It is tiny (12.7 × 10.16 mm). Solder it onto your PCB like any other component, or use it on a breadboard to try it first. Examples for Arduino, Raspberry Pi and PC help you get started quickly.







The IS4420-M1 sits between the current loop and your microcontroller.

Revision History

Document Revision

The revision of the present document.

Date	Revision Code	Description
November 2026	ISDOC146**A**	- Initial Release

Module Revision

This is the revision of the module hardware. It can be found marked on the module PCB.

Date	Revision Code	Description
November 2026	20**D**	- Initial Release

Firmware Revision

This is the revision of the module firmware. It can be found in the `FIRMWARE_REV` register of the memory map and on the streamed UART data.

Date	Revision Code	Description
November 2026	**0**	- Initial Release

Ordering Information

Orderable Device	Status	Package		Description
IS4420-M1I www.inacks.com/is4420	ACTIVE		Module (12.7 × 10.16 mm)	4–20 mA current loop receiver module with UART and I2C interface.
Kappa4420USB www.inacks.com/kappa4420usb	ACTIVE		Module	4–20 mA current loop receiver module with isolated USB interface. This module is an IS4420-M1I mounted on an isolated USB-to-UART adapter, which lets you receive the current loop readings safely on your PC or Mac and read the data with our software or develop your own software.

Status Description:

ACTIVE: Product recommended for new designs.

SPECIAL: Available only for high-volume pre-orders. Contact sales.

NRND: Device is in production to support existing customers, but it is not recommended for use in new designs.

OBSOLETE: Production of the device has been discontinued.

1 Electrical Specification

Unless otherwise noted, all values in this chapter are given at V_{DD} = 3.3 V and ambient temperature T_A = 25 °C.

Absolute Maximum Ratings

Parameter			Min	Max	Unit
Input Voltage	3V3 Pad		-0.3	4	V
	SCL, SDA, and TX Pads	Module Powered	-0.3	5.5	
		Module Unpowered (3)	-0.3	4	
	SPD and ADR Pads		-0.3	4	
	CIN Pad		0	4.5	
Current into the CIN pad, continuous (2)			0	30	mA
Temperature	Storage Temperature (5)		-40	85	°C
Electrostatic Discharge	Human-body model (HBM) (5)		-2000	1500	V
	Charged-device model (CDM) (5)		-500	500	

Exceeding the specifications outlined in the Absolute Maximum Ratings could potentially lead to irreversible harm to the device. It's important to note that these ratings solely indicate stress limits and don't guarantee the device's functionality under such conditions, or any others not specified in the Recommended Operating Conditions. Prolonged exposure to conditions at or beyond the absolute maximum ratings might compromise the reliability of the device.

Recommended Operating Conditions

Parameter			Symbol	Min	Nom	Max	Unit
Supply Voltage			V_{DD}	3.2	3.3	3.5	V
Input Voltage	SCL, SDA Pads		—	0	0, 3.3	5	
	SPD Pad		—	0	0, 1.65, 3.3	3.5	
	ADR Pad		—	0	0, 0.83, 1.65, 2.48, 3.3	3.5	
	TX Pad (1)		—	0	0, 3.3	5	
	CIN Pad (2)		V_{CIN}	0	—	3.23	
Sink Current	SCL, SDA, TX Pads		—	—	—	8	mA
	CIN Pad (2)		I_{CIN}	0	—	21.5	
Temperature	Operating Temperature (5)	IS4420-M1I	—	-40	25	85	°C

Electrical Characteristics

Parameter			Symbol	Min	Nom	Max	Unit
Current Consumption			I_{OP}	—	4.3	—	mA
Logical Levels for SCL, SDA Pads	Logical High-Level		V_{IH}	0.7×V_{DD}	—	—	V
	Logical Low-Level		V_{IL}	—	—	0.3×V_{DD}	
Module Temperature Reading (4)	Resolution		—	—	1	—	°C
	Error (5)	Over -40 to 85 °C	—	—	±4	±16	
Internal Non-Volatile Memory (5)	Endurance	T_A = -40 to 85 °C	—	10	—	—	kcycles
	Data Retention	1 kcycle at T_A = 85 °C	—	30	—	—	Years
		10 kcycles at T_A = 85 °C	—	15	—	—	

Current Loop Characteristics

Parameter		Symbol	Min	Nom	Max	Unit
CIN Input Resistance (150 Ω ±0.1 %, ±25 ppm/°C)		—	149.85	150	150.15	Ω
Voltage Drop at CIN	I_{CIN} = 20 mA	V_{CIN}	—	3.0	—	V
CIN Input Capacitance		—	—	100	—	nF
Measuring Range		I_{CIN}	0	—	21.5	mA
Measurement Error (5)	T_A = 25 °C	—	—	±0.015	±0.055	
	Over -40 to 85 °C	—	—	±0.035	±0.19	
Output Resolution	Milliamperes Reading	—	—	0.01	—	
	Percentage Reading	—	—	0.1	—	%
Measurement Window (5 periods of 50 Hz, 6 of 60 Hz)		—	—	100	—	ms
Data Update Frequency		—	—	9.7	—	Hz
Isolated Current Loop Input Pad (CIN)		No (CIN returns to GND)				

(1): If the UART is not used, leave TX floating and never connect it to GND.

(2): The CIN pad has a 150 Ω resistor to GND. Drive CIN only from a current-limited loop, with the current flowing into CIN. A reversed loop or a direct voltage source to CIN may permanently damage the module.

(3): When using a 5 V microcontroller, don’t keep these pads pulled up to 5 V while the module is off: unpowered, their maximum is 4 V.

(4): This is the temperature of the module, which may differ from the ambient due to self-heating.

(5): Calculated from the design and the ratings of the module’s components; not tested on the module.

2 Detailed Description

2.1 Description

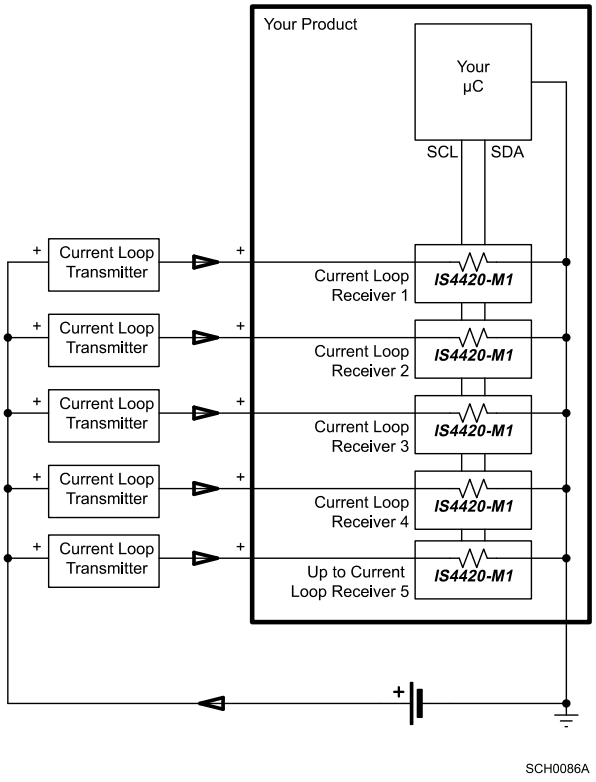
The IS4420-M1 is a small and cost-effective module that simplifies the analog circuitry required for a microcontroller to read the milliamp value of a 4–20 mA current loop via I2C or UART. The module can read the most popular current loop configurations: 2-wire, 3-wire and 4-wire.

It is useful for designing custom 4–20 mA receivers, for developing test and calibration equipment, and for reading the most common current loop sensors: temperature, pressure, flow meters, etc.

The module replaces the analog front end of your design, allowing you to prototype faster, validate ideas sooner and shorten the product time to market.

The loop current flows through an on-board shunt resistor, and the module measures it. The current, its percentage of the 4–20 mA span and the module's temperature can be read via I2C, and the module also sends them continuously on the UART, which makes it easy to integrate into any digital system.

The module can be configured with the I2C device addresses 26, 27, 28, 29 or 30, so up to five current loop receivers can share the same I2C-serial interface. It can work at 100 kHz, 400 kHz, and 1 MHz speeds.

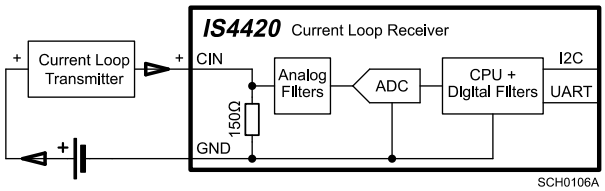


The module is powered at 3.3 V, and its UART and I2C interfaces are 5 V tolerant.

The module size is 12.7 × 10.16 mm, and it has flat-castellated pads, so it can be soldered directly to a PCB or mounted with pin headers. Its 2.54 mm pitch also makes it compatible with breadboards and protoboards.

2.2 Wire Configuration

The loop current enters the IS4420-M1 through the CIN pad, flows through the internal shunt resistor and leaves through the GND pad. The GND pad is also the reference for the 3V3 input voltage pad and for all the module's internal circuitry.

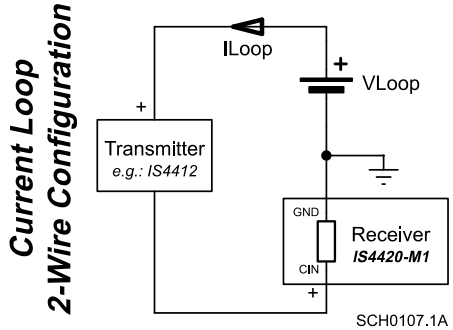


The IS4420-M1 can be directly integrated into 2-wire, 3-wire and 4-wire current loop systems.

In 2-wire and 3-wire systems, the current loop shares its return path with the ground reference of the power supply.

4-wire devices can have galvanic isolation between the power supply and the current loop. Therefore, using them with the IS4420-M1 requires special attention to the grounding and to the transmitter and receiver voltage reference connections, to avoid ground loops or breaking the isolation.

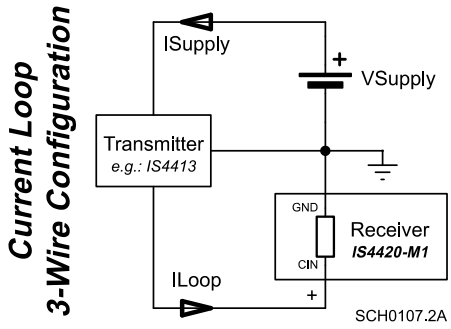
2-Wire Configuration



The 2-wire configuration is very popular in devices that need less than 4 mA to operate, because they can be powered directly from the loop current. This makes them simpler and cheaper to install.

In this configuration, the current output of the transmitter connects to the current input pad (CIN) of the IS4420-M1. And the negative terminal of the loop supply (VLoop) connects to the GND pad of the IS4420-M1, the return path of the loop.

3-Wire Configuration

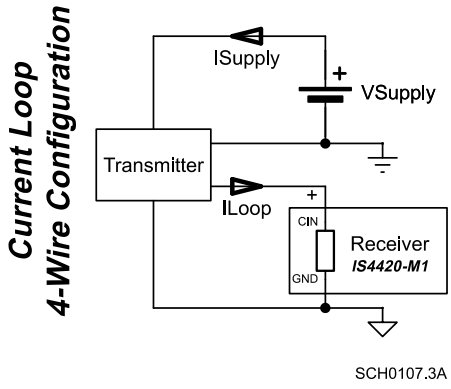


The 3-wire configuration has the benefit that the loop signal is independent of the supply current the transmitter needs to operate. This allows transmitters to use more than 4 mA, so they can be more complex and include, for example, a configuration screen.

While 2-wire transmitters work in the 4–20 mA range, 3-wire transmitters can work in the 0–20 mA range (note that one starts at 4 mA and the other starts at 0 mA), because they do not take their supply from the loop. In 4 –20 mA devices, the range below 4 mA is often used to signal fault states.

As in the 2-wire configuration, the power supply and the loop return share the same reference: the GND pad. And the current output of the transmitter connects to the CIN pad.

4-Wire Configuration



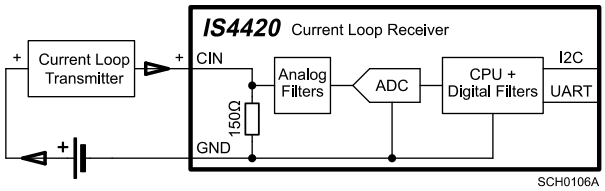
In the 4-wire current loop configuration, the transmitter can have galvanic isolation between the power supply (ISupply) and the current loop (ILoop). Therefore, using a 4-wire transmitter with the IS4420-M1 requires special attention to the grounding and to the transmitter and receiver voltage reference connections.

2.3 Measurement Principle

The loop current flows through a 150 Ω shunt resistor inside the module, which turns it into a voltage: 0.6 V at 4 mA and 3.0 V at 20 mA. Analog filters reduce the high-frequency noise and interference on this voltage before the module measures it.

A digital filter then averages all the samples the module takes during a 100 ms window into one measurement. The window is exactly 5 periods of 50 Hz and 6 periods of 60 Hz, so mains ripple picked up by the loop averages out. A new measurement is ready about every 103 ms.

The module corrects every measurement for the supply voltage, so the reading does not depend on the exact voltage at the 3V3 pad.



Then the module corrects the result with its two-point calibration (see Two-Point Calibration section) and rounds it to the nearest 0.01 mA, so the reported value is within ±0.005 mA of the calculated one. Two registers report it: `MA_INT` holds the whole milliamps and `MA_FRAC` the hundredths. For example, 12.34 mA reads `12` in `MA_INT` and `34` in `MA_FRAC`.

The percentage of the 4–20 mA span comes from the same measurement: 4 mA is 0 % and 20 mA is 100 %, that is, percentage = (current - 4 mA) / 16 mA × 100. It is rounded to the nearest 0.1 % and reported in `PERCENT_INT` (whole percent, sign in bit 7) and `PERCENT_FRAC` (tenths).

Below 4 mA the percentage is negative, down to -25.0 % at 0 mA: bit 7 of `PERCENT_INT` is set, and the UART line shows a minus sign. Above 20 mA the percentage goes over 100 %, up to 109.4 % at 21.50 mA (see Memory Map Table).

Above a reading of 21.50 mA the measurement is over-range: `MA_INT` and `PERCENT_INT` read `0xFF`, `MA_FRAC` and `PERCENT_FRAC` read `0x00`, and the UART line shows `EOVER` instead of the current and the percentage. With ripple on the loop, over-range can appear slightly below 21.50 mA, when the ripple peaks reach the top of the input range. The temperature is still reported.

Samples that fail are discarded. If a whole window has no valid sample, the module reports no value rather than a wrong one: `MA_INT`, `PERCENT_INT` and `MODULE_TEMP` read `0xFE`, `MA_FRAC` and `PERCENT_FRAC` read `0x00`, and the UART line shows `ERROR` (`ERR` for the temperature). The I2C registers read the same way after power-up, until the first measurement is ready (about 0.6 s).

In the same window the module also measures its internal temperature, and reports it in whole °C in `MODULE_TEMP` and on the UART line. Because of the

module's own heating, it reads a little above the ambient temperature (see MODULE_TEMP Register).

2.4 Two-Point Calibration

Every IS4420-M1 is calibrated at the factory at 4 mA and 20 mA. The two calibration points are stored in the module's non-volatile memory, and every measurement is corrected with them before it is reported, on I2C and on the UART line. A calibrated module reads `0x01` in `CAL_STATUS`.

The correction is a straight line through the two points: at each reference current the module reads the reference value, and the readings in between and beyond follow the same line. It removes the module's own offset and gain errors, which come from the tolerances of the shunt resistor and the other internal components. A calibration cannot remove the small non-linearity of the measurement or the error of the reference current used. Like any measuring device, the module's reading still changes slightly after a calibration: with temperature and, very slowly, as its components age. Recalibrating from time to time removes the aging part.

You can calibrate a module again with the procedure below, for example against your own reference, or at the supply voltage and temperature of your application for the best accuracy. A new calibration replaces the factory one: the module keeps only one calibration, and no copy of the factory one.

As the reference, use a loop calibrator, or a stable current source measured with a precision ammeter in series with the loop. Any error of the reference goes straight into the calibration.

Each command is one I2C write starting at address `13`: the command byte goes to `CAL_CMD` and the key `0xA5` to `CAL_KEY`. For example, the command `'L'` is the write `0x4C 0xA5` at address `13`. A command without the right key is ignored, and `CAL_ERROR` then reads `0x01`. `CAL_STATUS` and `CAL_ERROR` are at addresses `7` and `8`, so one 2-byte read from address `7` returns both.

1. Set the loop to the low reference current, 4 mA recommended, and let it settle.
2. Write the reference current in µA to `CAL_REF_LO`, at addresses `9` and `10`, most significant byte first: for example, 4012 µA is `0x0F 0xAC`. Then send `'L'` (`0x4C`). `CAL_REF_LO` holds 4000 after power-up, so with a reference of exactly 4.000 mA you only need to send `'L'`. One 6-byte write from address `9` can also set both references, the command and the key at once.
3. Read `CAL_STATUS` and `CAL_ERROR` until BUSY (bit 3) is `0`. The capture takes about 1.2 s, because the module averages ten measurements. The low point is captured if `CAL_ERROR` is `0x00` and LOW (bit 1) is `1`. If the current moved during the capture, `CAL_ERROR` reads `0x07`: let the loop settle and send `'L'` again.
4. Set the loop to the high reference current, 20 mA recommended, and let it settle. Write its value to `CAL_REF_HI`, at addresses `11` and `12` (20000 after power-up), and send `'H'` (`0x48`). Check as in step 3; this time HIGH (bit 2) must be 1.
5. Send `'S'` (`0x53`) to save. The module checks the two points, stores them and uses them at once. Wait 60 ms after the Stop condition before the next I2C transfer: storing takes up to about 46 ms, and the module holds SCL low if it is addressed meanwhile. Then read `CAL_STATUS` and `CAL_ERROR`: `0x01` and `0x00` mean the calibration is stored.

If `CAL_ERROR` is not `0x00` after the save, the new points were not stored (see `CAL_ERROR Register` section). The save is refused, for example, when the references are less than 4 mA apart, or when the correction would change the gain by more than 5 % or the offset by more than 200 µA. A working module never needs a correction that large, so a refused save almost always means a mistake during the calibration: for example, a reference value typed wrong, or the loop set to the wrong current when a point was captured. Check the setup and calibrate again.

Captured points are kept only until the save: each save needs both points captured again, and if the module is switched off before `'S'`, the captured points are lost and the stored calibration does not change.

To erase the calibration, send `'E'` (`0x45`) and wait 60 ms. This erases the factory calibration too: bit 0 of `CAL_STATUS` reads 0, and the module reports uncalibrated values until it is calibrated again.

The calibration memory is rated for at least 10,000 saves or erases.

2.5 Usage

Using the module is very simple: your microcontroller only needs to read the milliamp value over the I2C-serial interface or from the UART, no calibration and no specific libraries are needed.

Reading via I2C

Reading the module over I2C is as easy as reading an EEPROM memory: you tell the module which register you want to start from, and then you read.

The loop current is split into two registers. `MA_INT` holds the whole milliamps, and `MA_FRAC` holds the hundredths of a milliamp. They are the first two registers of the memory map, so one read of two bytes from the beginning gives you both.

To get the current, put the two parts back together: divide `MA_FRAC` by 100 and add it to `MA_INT`. For example, if `MA_INT` reads 12 and `MA_FRAC` reads 34, the loop current is 12 + 34 / 100 = 12.34 mA.

Always read both registers in the same read operation. That way both parts come from the same measurement, and you never mix the whole milliamps of one measurement with the hundredths of the next one.

The module refreshes these registers about ten times per second; reading more often just gives you the same value again. Two values are special: if `MA_INT` reads `255`, the current is above the measuring range, and if it reads `254`, the module has no measurement to give, for example just after power-up.

The same memory map also holds the percentage of the 4–20 mA span and the module's temperature. The I2C Memory Map Description chapter explains them.

Reading via UART

Reading over the UART is even easier, because you don't have to ask for anything: the module sends the measurement by itself, as a line of text, about ten times per second. Your microcontroller only needs one pin to listen.

Each line looks like this:

```
12.34mA_+052.1%_+025deg_IS4420M1_ver0
```

The first five characters are the loop current in milliamps, here 12.34 mA. In your program, read a whole line, take its first five characters and convert them into a number. If you start listening in the middle of a line, just skip that first piece and use the next full line.

The rest of the line holds the percentage of the 4–20 mA span, the module's temperature, the model and the firmware revision.

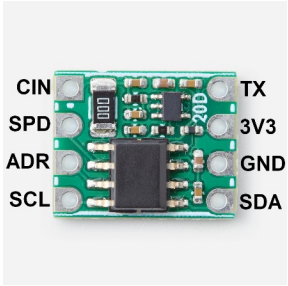
If the module cannot give a value, the first characters show `EOVER` (current above the measuring range) or `ERROR` (no measurement) instead of a number. So check that you really have a number before you use it.

Because the line is plain text, you can also watch it on a computer with a USB-to-UART adapter. You don't need any special software: any serial terminal program will do. The Hardware Examples chapter shows how to connect it, and the UART Description chapter explains the whole line.

Hardware Design

Refer to the Hardware Examples chapter for more information about hardware design.

3 Pads Description



Pad	Name	Type	Description
1	CIN	Analog Input	Current Loop Input Pad
2	SPD	Analog Input	I2C-Serial Interface Speed Selection Pad • For 100 kHz pull to GND. • For 400 kHz make a voltage divider of ½ 3V3 (1.65 V). • For 1 MHz pull to 3V3.
3	ADR	Analog Input	I2C-Serial Interface Address Selection Pad • For 26 pull to GND. • For 27 make a voltage divider of ¼ 3V3 (0.83 V). • For 28 make a voltage divider of ½ 3V3 (1.65 V). • For 29 make a voltage divider of ¾ 3V3 (2.48 V). • For 30 pull to 3V3.
4	SCL	Open Drain 5 V Tolerant	I2C-Serial Interface Pads These pads are open-drain and need pull-up resistors to 3.3 V or 5 V. The IS4420-M1 requires your I2C-master device to support I2C-clock stretching.
5	SDA		
6	GND	Power	Reference Pad Module ground and current loop return.
7	3V3		3.3 V Power Supply Pad A 100 nF ceramic capacitor to GND, close to this pad, is good practice.
8	TX	Open Drain 5 V Tolerant	UART Transmitter Pad UART output, one line of text about every 103 ms; needs an external pull-up to 3.3 V or 5 V. If unused, leave floating. Do not connect to GND, or the module will not boot.

3.1 CIN Pad

Current Loop Input Pad.

The loop current enters the module through this pad, flows through a 150 Ω shunt resistor and returns to the loop through the GND pad. The input is not isolated: the loop return is the module ground.

CIN has no protection on the module: a reversed loop, or a voltage source connected straight to CIN such as the loop supply, may permanently damage the module. A diode in series with CIN protects it against a reversed loop, but it adds about 0.7 V to the loop's voltage drop.

3.2 TX Pad

UART Transmitter Pad.

The module sends the loop current and its own temperature as one line of ASCII text per measurement, about every 103 ms (see UART Description). The host only listens: the module has no receive pad and there are no commands.

This is an open-drain pad: fit a pull-up resistor to 3.3 V or 5 V, from 1 kΩ to 4.7 kΩ. 2.2 kΩ is recommended.

Connect TX only to the host's RX input: an output on the same wire would fight the module's output.

Do not hold the line low while the module powers up: the module does not start until the line is released.

The line can go low for a moment at power-up, so the host may receive one stray byte before the first line.

When using a 5 V pull-up on the TX pad, the module must remain powered whenever the 5 V pull-up supply is present. When unpowered, the maximum voltage allowed on the TX pad is 4 V (see Absolute Maximum Ratings).

IMPORTANT
If the UART is not used, leave the TX pad floating. If you wrongly connect the TX pad to GND, the module will not boot.

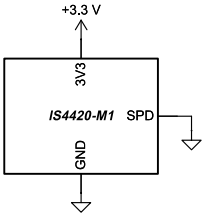
3.3 SPD Pad

I2C Speed Selection Pad.

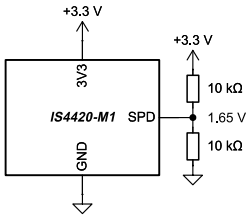
This pad configures the IS4420-M1 internal I2C-serial interface timings and filters to properly work with the selected bus speed. This pad is only checked at power-up. Set it before power-up, and do not change it while the module is powered.

Do not leave this pad floating. If you don't use the I2C-serial interface, connect SCL, SDA, ADR and SPD straight to GND.

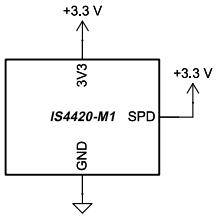
For a **100 kHz** setting, connect the SPD pad to GND.



For a **400 kHz** setting, connect the SPD pad to 1.65 V (½ 3V3) using a balanced voltage divider. This can be achieved by placing two 10 kΩ resistors from the SPD pad: one to 3V3 and the other to GND.



For a **1 MHz** setting, connect the SPD pad to 3.3 V.



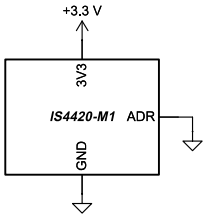
3.4 ADR Pad

I2C Device Address Selection Pad.

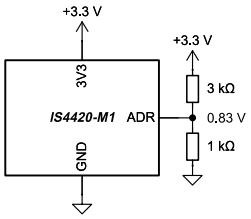
This pad configures the IS4420-M1 internal I2C device address. This pad configuration is only checked at power-up. Set it before power-up, and do not change it while the module is powered.

Do not leave this pad floating. If you don't use the I2C-serial interface, connect SCL, SDA, ADR and SPD straight to GND.

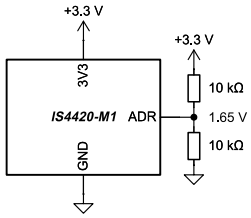
For **I2C device address 26**, connect the ADR pad to GND.



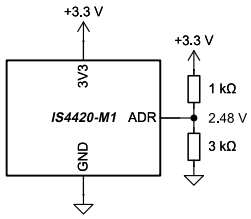
For **I2C device address 27**, connect the ADR pad to 0.83 V (¼ 3V3) with a voltage divider: 3 kΩ from the ADR pad to 3V3 and 1 kΩ to GND.



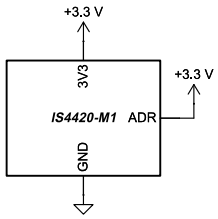
For **I2C device address 28**, set the ADR pad to 1.65 V (½ 3V3) using a balanced voltage divider. This can be achieved by placing two 10 kΩ resistors from the ADR pad: one to 3V3 and the other to GND.



For **I2C device address 29**, connect the ADR pad to 2.48 V (¾ 3V3) with a voltage divider: 1 kΩ from the ADR pad to 3V3 and 3 kΩ to GND.



For **I2C device address 30**, connect the ADR pad to 3.3 V.



3.5 SCL and SDA Pads

I2C-Serial Interface Pads.

SCL (Serial Clock Line): This pad is used to synchronize data transfer between the IS4420-M1 device and the microcontroller or other CPU.

SDA (Serial Data Line): This bidirectional pad is used for both sending and receiving data between the IS4420-M1 and the microcontroller or other CPU.

When using 5 V pull-ups on SDA and SCL, the module must remain powered whenever the 5 V pull-up supply is present. When unpowered, the maximum voltage allowed on the SDA and SCL pads is 4 V (see Absolute Maximum Ratings).

Both pads are open-drain and must be pulled up to 3.3 V or 5 V. The pull-up resistor value should be chosen based on the bus speed and capacitance. Typical values are 4.7 kΩ at 100 kHz, 2 kΩ at 400 kHz and 1 kΩ at 1 MHz.

If you don’t use the I2C-serial interface, connect SCL, SDA, ADR and SPD straight to GND.

3.6 3V3 and GND Pads

Module Power Pads.

These two pads power the module. GND is also the return of the current loop.

Supply the module with 3.3 V. The module filters and decouples the supply on board, but a 100 nF ceramic capacitor between 3V3 and GND, close to the module, is still good practice. The top of the measuring range follows the supply: at 3.2 V, readings above about 21.0 mA show `EOVER`. There is no reverse-polarity protection: a reversed supply may permanently damage the module.

4 UART Description

The module sends the loop current and its own temperature as one line of ASCII text per measurement, about every 103 ms. The host only listens: the UART is transmit-only, with no receive pad and no commands.

The serial configuration is:

- 115200 bps
- 8 data bits
- No parity
- 1 stop bit
- No flow control

The lines that the IS4420-M1 sends on the TX pad look like these examples:

```
12.34mA_+052.1%_+025deg_IS4420M1_ver0
03.00mA_-006.3%_+025deg_IS4420M1_ver0
EOVERmA_+EOVER%_+025deg_IS4420M1_ver0
ERRORmA_+ERROR%_+ERRdeg_IS4420M1_ver0
```

- Every line is 39 bytes: 37 printable characters, then CR LF (`0x0D 0x0A`), which appear only at the end.
- The first line comes about 0.6 s after power-up; each line takes about 3.4 ms on the wire.
- Each field is always at the same place, and an underscore separates the fields (5 parts).
- Reading rule: keep a line only if exactly 37 characters come before its CR LF. This drops a line caught halfway and a stray byte from power-up.

Frame Fields

The fields of each line are:

`xx.xx``mA_`**`yzzz.z`**`%_`**`stttdeg_IS4420M1_ver`**`v`**

- **`xx.xx`** (characters 1 to 5): the loop current in mA, with 2 decimals.
- **`y`** (character 9): `-` only below 0 %, that is, below 4 mA; otherwise `+`. Zero is `+000.0`.
- **`zzz.z`** (characters 10 to 14): the percentage of the 4 –20 mA span: 4 mA is 0 %, 20 mA is 100 %.
- **`s`** (character 17): `-` only below 0 °C; otherwise `+`. Zero is `+000`.
- **`ttt`** (characters 18 to 20): the temperature inside the module in whole °C, as in the `MODULE_TEMP` register.
- **`v`** (character 37): the firmware revision, as in the `FIRMWARE_REV` register. It may change during the product life cycle.

The following table shows the `xx.xx` and `yzzz.z` fields for some measured currents.

Measured Current	Field `xx.xx`	Field `yzzz.z`
0.00 mA	`00.00`	`-025.0`
3.00 mA	`03.00`	`-006.3`
4.00 mA	`04.00`	`+000.0`
12.00 mA	`12.00`	`+050.0`
20.00 mA	`20.00`	`+100.0`
20.50 mA	`20.50`	`+103.1`
Over-range	`EOVER`	`+EOVER`
No measurement	`ERROR`	`+ERROR`

Error Reporting

- Over-range: above a reading of 21.50 mA (about 21.0 mA at VDD = 3.2 V), the `xx.xx` and `zzz.z` fields show `EOVER`; the temperature is still shown. Do not exceed the absolute maximum current of the CIN pad.
- No measurement: if the module cannot complete a measurement, the `xx.xx` and `zzz.z` fields show `ERROR` and the `ttt` field `ERR`, so the host never processes a wrong value. `ERROR` takes priority over `EOVER`.

5 I2C Memory Map Description

5.1 Memory Map Table

The memory map registers are 8 bits long, and register addresses (Pointer Register) are 16 bits long.

Address	Register Name	Access	Description
0x0000	**MA_INT**	R	Loop current, integer mA; 0xFF over-range, 0xFE no measurement
0x0001	**MA_FRAC**	R	Loop current, hundredths of mA (0 to 99); 0x00 with either mark
0x0002	**PERCENT_INT**	R	% of the 4–20 mA span: bit 7 = sign (1 below 4 mA), bits 6 to 0 = integer part; 0xFF over-range, 0xFE no measurement
0x0003	**PERCENT_FRAC**	R	% of the span, tenths (0 to 9); 0x00 with either mark
0x0004	**MODULE_ID**	R	Product identifier, always 149 (0x95)
0x0005	**FIRMWARE_REV**	R	Firmware revision; the UART line shows it after "ver"
0x0006	**MODULE_TEMP**	R	Temperature inside the module in °C: bit 7 = sign (1 below 0 °C), bits 6 to 0 = whole degrees; 0xFE no measurement
0x0007	**CAL_STATUS**	R	Calibration status: bit 0 CAL, bit 1 LOW, bit 2 HIGH, bit 3 BUSY (see Two-Point Calibration)
0x0008	**CAL_ERROR**	R	Result of the last calibration command; 0x00 = no error
0x0009 0x000A	**CAL_REF_LO**	R/W	Low reference current in µA, most significant byte first; 4000 after power-up
0x000B 0x000C	**CAL_REF_HI**	R/W	High reference current in µA, most significant byte first; 20000 after power-up
0x000D	**CAL_CMD**	W	Calibration command: 'L' (0x4C) low point, 'H' (0x48) high point, 'S' (0x53) save, 'E' (0x45) erase
0x000E	**CAL_KEY**	W	Calibration key: 0xA5, in the same write as CAL_CMD

Table 1: IS4420-M1's Memory Map

5.2 MA_INT and MA_FRAC Registers

The `MA_INT` and `MA_FRAC` registers contain the loop current in milliamperes: `MA_INT` holds the integer part and `MA_FRAC` the hundredths. For example, `12` and `34` is `12.34` mA.

The module updates them with every reading, about every 103 ms. Read both in one I2C read operation: all the bytes of one read come from the same measurement, so you never mix values from two different measurements.

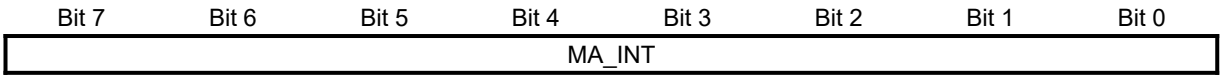
Over-Range
If the module reads more than 21.50 mA, the current is over-range and `MA_INT` reads `255` (`0xFF`).

No Measurement
If the module has no valid reading, `MA_INT` reads `254` (`0xFE`). This is normal for about 0.6 s after power-up, until the first measurement is ready.

In both cases `MA_FRAC` reads `0`.

These registers are read-only.

Name:	MA_INT
Description:	Loop Current, Integer mA
Address:	`0` (`0x0000`)
Memory Type:	Volatile RAM
Allowed Values:	`0` to `21` (`0x00` to `0x15`), `254` (`0xFE`) and `255` (`0xFF`)
Reset Value:	`254` (`0xFE`)
Access Type:	I2C Read



Name:	MA_FRAC
Description:	Loop Current, Hundredths of mA
Address:	`1` (`0x0001`)
Memory Type:	Volatile RAM
Allowed Values:	`0` to `99` (`0x00` to `0x63`)
Reset Value:	`0` (`0x00`)
Access Type:	I2C Read



5.3 PERCENT_INT and PERCENT_FRAC Registers

The PERCENT_INT and PERCENT_FRAC registers contain the loop current as a percentage of the 4–20 mA span: 4 mA is 0 % and 20 mA is 100 %. PERCENT_INT holds the whole percent and PERCENT_FRAC the tenths. Bit 7 of PERCENT_INT is the sign: it is 1 below 4 mA, where the percentage is negative. For example, 52.1 % reads 0x34 and 1, and -6.3 % (3.00 mA) reads 0x86 and 3. Zero is never negative.

With over-range PERCENT_INT reads 255 (0xFF), and without a measurement 254 (0xFE); in both cases PERCENT_FRAC reads 0.

These registers are read-only.

Name: PERCENT_INT
Description: Loop Current, Integer %
Address: 2 (0x0002)
Memory Type: Volatile RAM
Allowed Values: 0x00 to 0x6D, 0x80 to 0x99, 0xFE and 0xFF
Reset Value: 254 (0xFE)
Access Type: I2C Read

Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
Sign	Percentage, Integer Part						

Name: PERCENT_FRAC
Description: Loop Current, Tenths of %
Address: 3 (0x0003)
Memory Type: Volatile RAM
Allowed Values: 0 to 9 (0x00 to 0x09)
Reset Value: 0 (0x00)
Access Type: I2C Read

Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
PERCENT_FRAC							

5.4 MODULE_ID Register

The `MODULE_ID` register contains the module identifier, a fixed value of `149` (`0x95`). This value is used for production tracking and will not change throughout the product's life cycle.

Since this register value is constant, reading it during firmware development can help verify that I2C communications are working.

This register is read-only.

Name: MODULE_ID
Description: Module Identification Number
Address: `4` (`0x0004`)
Memory Type: ROM
Value: `149` (`0x95`)
Access Type: I2C Read

Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
1	0	0	1	0	1	0	1

5.5 FIRMWARE_REV Register

The `FIRMWARE_REV` register indicates the firmware revision. This value is intended for production and product tracking, and may change throughout the product's life cycle.

The UART line shows the same value, after `ver`.

This register is read-only.

Name: FIRMWARE_REV
Description: Module Firmware Revision Number
Address: `5` (`0x0005`)
Memory Type: ROM
Value: Depends on the revision
Access Type: I2C Read

Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
FIRMWARE_REV							

5.6 MODULE_TEMP Register

The `MODULE_TEMP` register contains the temperature inside the module, in degrees Celsius. Bit 7 is the sign (1 below 0 °C) and bits 6 to 0 are the value: for example, `0x19` is 25 °C and `0xA8` is -40 °C.

The module updates it with every reading, about every 103 ms. Without a valid reading, also before the first one after power-up, it reads `254` (`0xFE`).

The module's own heating puts this temperature above the ambient temperature: use it to monitor the module, not as a thermometer.

This register is read-only.

Name: MODULE_TEMP
Description: Module Temperature
Address: `6` (`0x0006`)
Memory Type: Volatile RAM
Allowed Values: `0x00` to `0x7D` (0 to 125 °C), `0x81` to `0xFD` (-1 to -125 °C) and `0xFE`
Reset Value: `254` (`0xFE`)
Access Type: I2C Read

Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
Sign	Temperature (°C)						

5.7 CAL_STATUS Register

The `CAL_STATUS` register shows the state of the two-point calibration (see Two-Point Calibration section):

- `CAL` (bit 0): a calibration is stored and corrects every reading.
- `LOW` (bit 1): the low point is captured, waiting for a SAVE.
- `HIGH` (bit 2): the high point is captured, waiting for a SAVE.
- `BUSY` (bit 3): a capture is running, about 1.2 s.

Bits 7 to 4 read 0.

A factory-calibrated module reads `0x01`.

This register is read-only.

Name: CAL_STATUS
Description: Calibration Status
Address: `7` (`0x0007`)
Memory Type: Volatile RAM
Allowed Values: `0x00` to `0x0F`
Reset Value: `0x01` (calibrated module)
Access Type: I2C Read

Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
0	0	0	0	BUSY	HIGH	LOW	CAL

5.8 CAL_ERROR Register

The `CAL_ERROR` register contains the result of the last calibration command; `0x00` means no error:

- `0x01`: wrong key in `CAL_KEY`, command ignored.
- `0x02`: unknown command.
- `0x03`: command refused, a capture is running.
- `0x04`: reference out of range (1000 to 21000 µA).
- `0x05`: current at the top of the range during the capture.
- `0x06`: no valid reading during the capture.
- `0x07`: the current moved more than 20 µA during the capture.
- `0x08`: SAVE without both points captured.
- `0x09`: references less than 4 mA apart, or the low one above the high one.
- `0x0A`: gain outside 0.95 to 1.05.
- `0x0B`: offset outside ±200 µA.
- `0x0C`: the calibration memory could not be written.
- `0x0D`: at power-up the stored calibration was not valid; the module reads uncalibrated.

This register is read-only.

Name:	CAL_ERROR
Description:	Calibration Error Code
Address:	`8` (`0x0008`)
Memory Type:	Volatile RAM
Allowed Values:	`0x00` to `0x0D`
Reset Value:	`0x00` (`0x0D` if the stored calibration is not valid)
Access Type:	I2C Read

Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
CAL_ERROR							

5.9 CAL_REF_LO and CAL_REF_HI Registers

The `CAL_REF_LO` and `CAL_REF_HI` registers contain the reference currents of the two calibration points, in microamperes: the real loop current when the `'L'` or `'H'` command captures the point (see Two-Point Calibration).

Each one is a 16-bit register made of two 8-bit registers, most significant byte first: `CAL_REF_LO` is at addresses `9` (high byte) and `10` (low byte), and `CAL_REF_HI` at `11` and `12`.

Allowed values are `1000` to `21000` µA. On each power-up they are set to `4000` and `20000` µA. They only set the next capture: the stored calibration does not change until a SAVE.

These registers can be read and written via I2C.

Name:	CAL_REF_LO
Description:	Low Calibration Reference (µA)
Address Range:	`9` to `10` (`0x0009` to `0x000A`)
Memory Type:	Volatile RAM
Allowed Values:	`1000` to `21000` (`0x03E8` to `0x5208`)
Reset Value:	`4000` (`0x0FA0`)
Access Type:	I2C Read and Write

Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
CAL_REF_LO, high byte first							

Name:	CAL_REF_HI
Description:	High Calibration Reference (µA)
Address Range:	`11` to `12` (`0x000B` to `0x000C`)
Memory Type:	Volatile RAM
Allowed Values:	`1000` to `21000` (`0x03E8` to `0x5208`)
Reset Value:	`20000` (`0x4E20`)
Access Type:	I2C Read and Write

Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
CAL_REF_HI, high byte first							

5.10 CAL_CMD and CAL_KEY Registers

The `CAL_CMD` register runs the calibration commands and the `CAL_KEY` register protects them: write the command to `CAL_CMD` and `0xA5` to `CAL_KEY` in the same I2C write, starting at address `13`. A command without the right key is ignored.

- `'L'` (`0x4C`): capture the low point.
- `'H'` (`0x48`): capture the high point.
- `'S'` (`0x53`): save the two points.
- `'E'` (`0x45`): erase the calibration.

After a SAVE or an ERASE, wait 60 ms after the Stop condition before the next I2C transfer.

These registers are write-only: they always read `0`.

Name: CAL_CMD
Description: Calibration Command
Address: `13` (`0x000D`)
Memory Type: Volatile RAM
Allowed Values: `0x4C`, `0x48`, `0x53` and `0x45`
Reset Value: `0` (`0x00`)
Access Type: I2C Write

Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
CAL_CMD							

Name: CAL_KEY
Description: Calibration Key
Address: `14` (`0x000E`)
Memory Type: Volatile RAM
Allowed Values: `165` (`0xA5`)
Reset Value: `0` (`0x00`)
Access Type: I2C Write

Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
1	0	1	0	0	1	0	1

6 I2C-Serial Interface Description

The IS4420-M1 operates as a slave on the I2C-serial interface. It supports Standard Mode (100 kHz), Fast Mode (400 kHz), and Fast Mode Plus (1 MHz). The speed is selected with the SPD pad. The I2C-master device, typically a microcontroller or a microprocessor, initiates and manages all read and write operations to the slave.

The IS4420-M1 is represented on the bus by one I2C device address, selected with the ADR pad: 26, 27, 28, 29 or 30.

Pull-up resistors are required on the SCL and SDA lines for proper operation. The resistor values depend on the bus capacitance and operating speed. Typical values are 4.7 kΩ at 100 kHz, 2 kΩ at 400 kHz and 1 kΩ at 1 MHz.

The IS4420-M1's I2C pads high state can be either 3.3 V or 5 V. A logical '0' is transmitted by pulling the line low, while a logical '1' is transmitted by releasing the line, allowing it to be pulled high by the pull-up resistor. The master drives the Serial Clock (SCL) line, which clocks the data on the Serial Data (SDA) line. The IS4420-M1 can hold SCL low to stretch the clock (see below).

A Start or Stop condition occurs when the SDA line changes during the high period of the SCL line. Data on the SDA line must be 8 bits long and is transmitted most significant bit first and most significant byte first. After the 8 data bits, the receiver must respond with either an acknowledge (ACK) or a no-acknowledge (NACK) bit during the ninth clock cycle, which is generated by the master. To keep the bus in an idle state, both the SCL and SDA lines must be released to the high state.

The memory map consists of 15 registers (`0` to `14`), each 8 bits wide. Addressing a register requires a 2-byte register address, most significant byte first. Every read must start by writing the register address, followed by a Repeated Start and the read; reads without the register address first are not supported.

The IS4420-M1 holds SCL low (clock stretching) while it prepares its answer: about 2 ms normally, up to about 5.5 ms while a UART line is being sent, and up to about 46 ms right after a calibration SAVE or ERASE. Your I2C-master device must support clock stretching. After a SAVE or an ERASE, wait 60 ms after the Stop condition before the next access.

Reading and writing the IS4420-M1 works much like an EEPROM, which is a good way to quickly understand how to communicate with the device. One difference: every read must start with the register address.

6.1 Highlights

I2C Device Address: 26, 27, 28, 29 or 30, set with the ADR pad

I2C Memory Map Addressing Size: 16-bit (2 bytes, MSB first)

I2C Memory Map Register Size: 8-bit

I2C Clock Stretching: yes, the I2C-master device must support it

Compatible I2C Speeds:

- Standard Mode (100 kHz), typical SCL and SDA pull-up value: 4.7 kΩ
- Fast Mode (400 kHz), typical SCL and SDA pull-up value: 2 kΩ
- Fast Mode Plus (1 MHz), typical SCL and SDA pull-up value: 1 kΩ

Supported Operations:

- Single Byte Write
- Multiple Byte Write (up to 15 registers)
- Single Byte Read
- Multiple Byte Read (up to 15 registers)

6.2 Single Byte Write

Writing a single byte is an action performed by the microcontroller (I2C-master) to write data to one register of the IS4420-M1 memory (I2C-slave). To perform this action, the microcontroller must load the register address intended to be written into the IS4420-M1's Pointer Register. Once the address is set, the microcontroller can send the data to be written. Only registers `9` to `14` can be written: registers `0` to `8` are read-only, and the IS4420-M1 acknowledges a write to them and ignores the data.

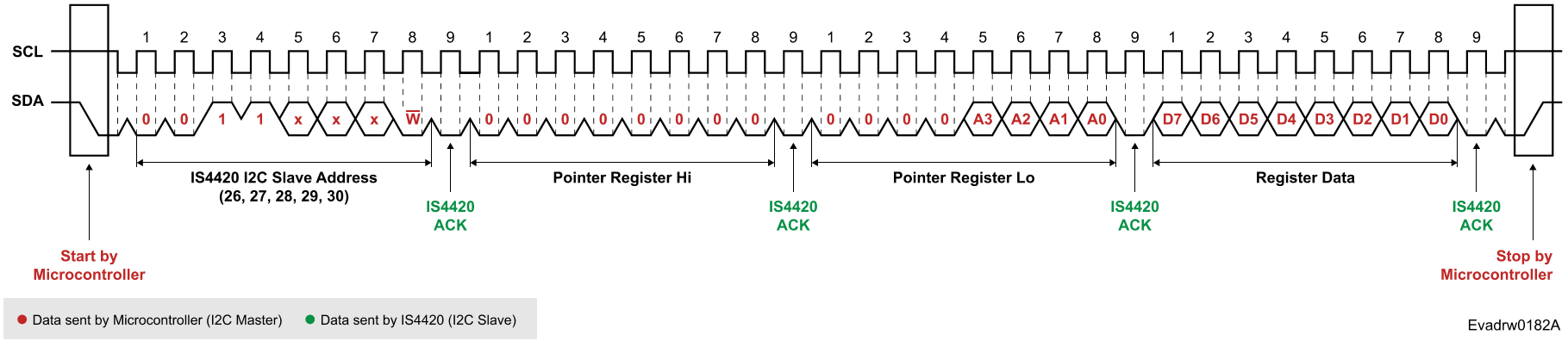
To initiate the Single Byte Write operation, the following steps must be performed from the beginning: The microcontroller begins by pulling down the SDA line while the SCL line is high, creating a Start condition. It then sends the IS4420-M1 I2C device address (one of 26, 27, 28, 29 or 30) with the R/W bit set to `0` (indicating a write operation). The address is the one selected with the ADR pad, which the module reads at power-up (see ADR Pad). With five addresses to choose from, up to five IS4420-M1 modules can share the same I2C-serial interface.

Upon receiving the device address, the IS4420-M1 acknowledges it. Subsequently, the microcontroller sends the two bytes of the register address it intends to write: the most significant byte first, followed by the least significant byte, each acknowledged by the IS4420-M1.

The microcontroller then sends the byte to be written, which the IS4420-M1 acknowledges. Finally, the microcontroller issues a Stop condition by raising the SDA line while the SCL line is high.

Invalid Memory Addressing

The valid memory range of the IS4420-M1 for a write operation goes from addresses `0` to `14`. If a write operation is performed with a Pointer Register higher than `14`, the IS4420-M1 acknowledges the register address but answers the data byte with a no-acknowledge (NACK), and the data is discarded.



6.3 Multiple Byte Write

The Multiple Byte Write operation functions similarly to the Single Byte Write, but allows writing a block of up to 15 registers in a single operation. Registers `0` to `8` are read-only: writes to them are acknowledged and ignored.

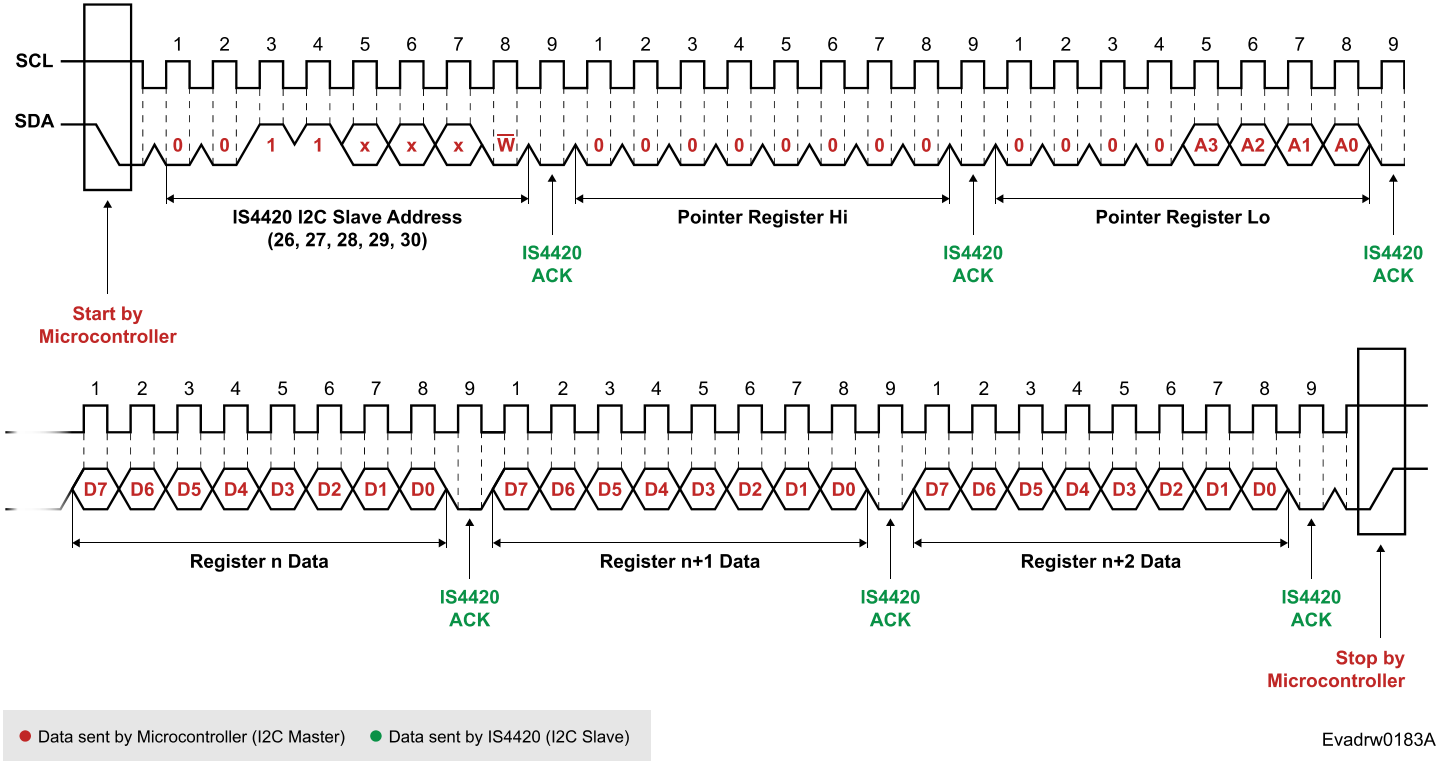
To perform a Multiple Byte Write operation, follow the same procedure as for a Single Byte Write until the first data byte is written. After writing the first byte, instead of generating a Stop condition, the microcontroller should continue writing data bytes. To conclude the write operation, after sending the last data byte, the microcontroller should generate a Stop condition.

Invalid Memory Addressing

The valid memory range of the IS4420-M1 for a write operation goes from addresses `0` to `14`.

If a Multiple Byte Write operation is performed with a Pointer Register within the valid memory range (`0` to `14`) but exceeds the last memory register (`14`), a rollover to register `0` will occur.

If a Multiple Byte Write operation is performed with a Pointer Register outside the valid memory range (greater than `14`), the IS4420-M1 will respond with a no-acknowledge (NACK) upon receiving the first data byte, and the data will be discarded.



6.4 Single Byte Read

Reading a single byte is an action performed by the microcontroller (I2C-master) to read any register within the IS4420-M1 memory (I2C-slave). To perform this action, the microcontroller first loads the address of the register into the IS4420-M1's Pointer Register, and then reads the content of that register. Every read must start this way, with a Repeated Start between the register address and the read: the IS4420-M1 does not support reads without the register address first.

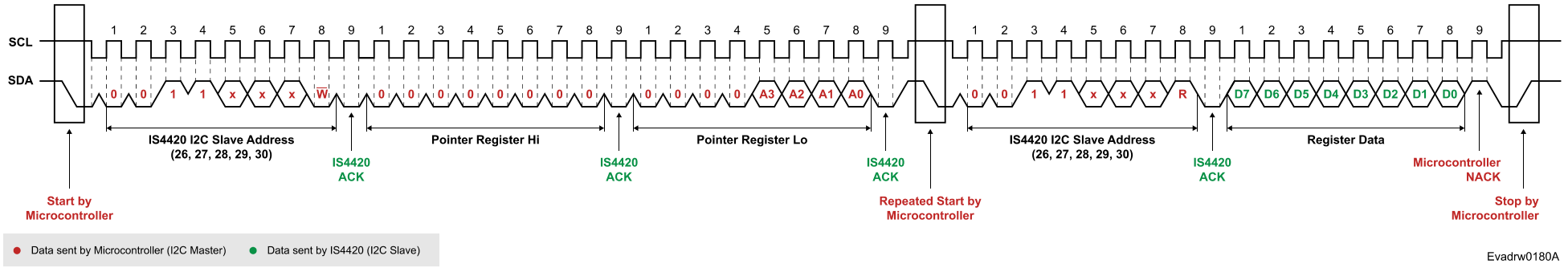
To initiate the Single Byte Read operation, the following steps must be performed from the beginning: The microcontroller starts by pulling SDA low while SCL is high to generate a Start condition. It then sends the IS4420-M1 I2C device address (one of 26, 27, 28, 29 or 30) with the R/W bit set to `0` (write). Upon receiving the device address, the IS4420-M1 acknowledges it. Subsequently, the microcontroller sends the two bytes of the register address: the most significant byte first, followed by the least significant byte, each acknowledged by the IS4420-M1. This loads the address of the register to be read into the Pointer Register.

Next, the microcontroller reads the register selected by the Pointer Register.

The microcontroller generates a Repeated Start condition, followed by the IS4420-M1 I2C device address (one of 26, 27, 28, 29 or 30) with the R/W bit set to `1` (read), instructing the IS4420-M1 to send data. The IS4420-M1 acknowledges and responds with the content of the register, which the microcontroller answers with a no-acknowledge (NACK). Finally, the microcontroller issues a Stop condition by raising the SDA line while the SCL line is high.

Invalid Memory Addressing

The valid memory range of the IS4420-M1 goes from addresses `0` to `14`. If a read operation is performed with a Pointer Register higher than `14`, the read result will be `0xFF`.



6.5 Multiple Byte Read

The Multiple Byte Read operation functions similarly to the Single Byte Read, but can read a block of up to 15 registers in a single operation, corresponding to the full memory map.

To perform a Multiple Byte Read operation, follow the same procedure as for a Single Byte Read until the first byte is received. After receiving the first byte, instead of generating a no-acknowledge (NACK), the microcontroller should continue acknowledging (ACK) each received data byte from the IS4420-M1 for as many bytes as it intends to read. To conclude the read operation, after reading the last data byte, the microcontroller should generate a no-acknowledge (NACK) and a Stop condition.

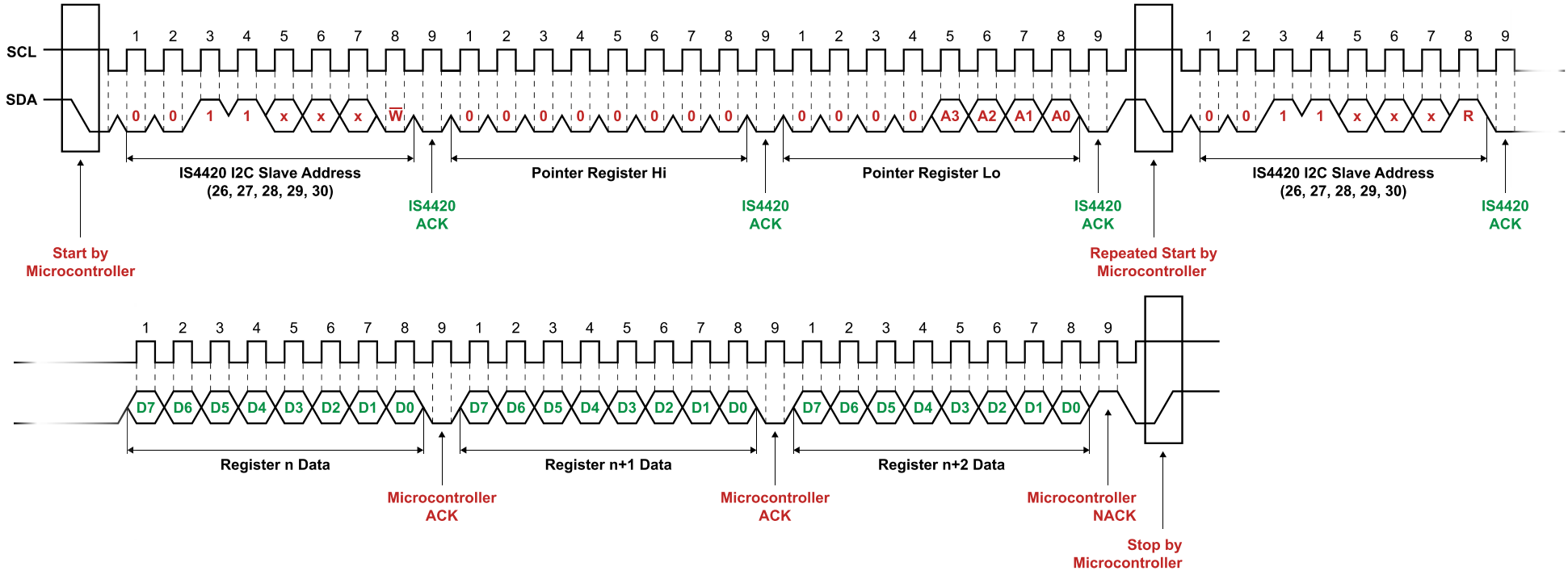
With each byte read, the Pointer Register increments by one.

Invalid Memory Addressing

The valid memory range of the IS4420-M1 goes from addresses `0` to `14`.

If a read operation is performed with a Pointer Register within the valid memory range (`0` to `14`), but the data retrieval extends beyond register `14`, a rollover to register `0` will occur. For example, a 4-byte read that starts at register `13` returns registers `13`, `14`, `0`, and `1`.

If a read operation is performed with a Pointer Register value higher than `14`, the read result will be `0xFF`.



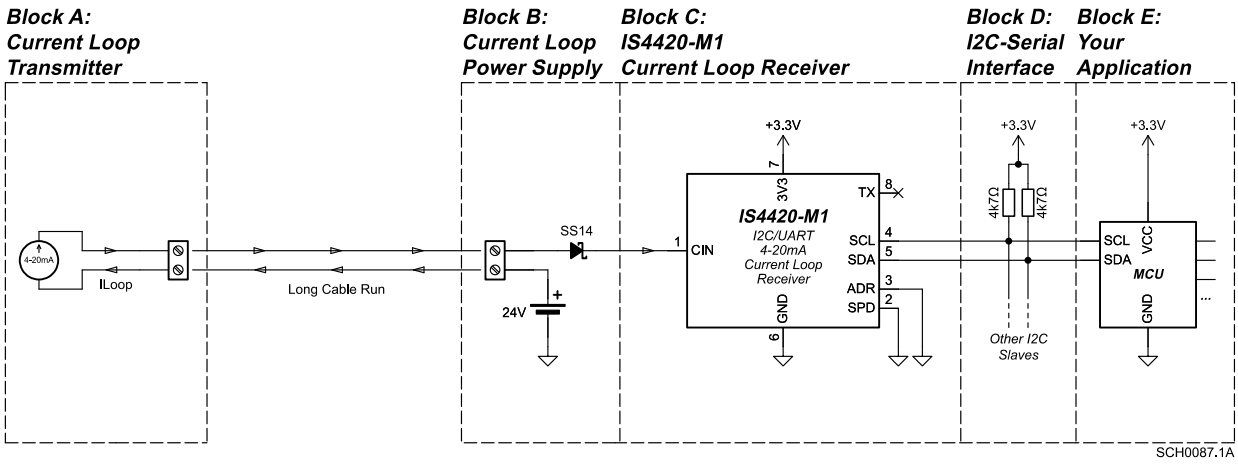
Evadrw0181A

7 Hardware Examples

This chapter shows application examples that explain how the module can be used. They are not part of the product specification. You are responsible for designing your own solution, choosing the components that best fit it, and validating the final product against the applicable legislation and current loop standards.

7.1 Reading Current Loop via I2C

This example shows a design that reads a 4–20 mA transmitter with the IS4420-M1 and passes the measurement to a microcontroller over the I2C-serial interface.



Block A: Current Loop Transmitter

This is the sensor you want to read, for example a pressure, temperature or flow transmitter. It turns what it measures into a current between 4 mA and 20 mA: 4 mA at the bottom of its range and 20 mA at the top.

The example shows a 2-wire transmitter, the most common type. It has no power supply of its own: it takes the little power it needs from the loop current itself, so the same two wires carry both its power and its signal.

Because the information travels as a current and not as a voltage, the cable can be long. The resistance of the wires does not change the current, so the reading at the other end stays the same, as long as the loop power supply has enough voltage (see Block B).

You can also read 3-wire and 4-wire transmitters (see Wire Configuration section). Take special care with 4-wire transmitters: some of them have galvanic isolation between their power supply and their current output. Make sure you understand how their grounds are connected, do not join different ground domains by accident, and watch out for ground loops when you connect them to the IS4420-M1.

Block B: Current Loop Power Supply

This supply powers the whole loop: it pushes the current through the transmitter, the cable and the IS4420-M1. 24 V is the usual choice, and it is what the example uses.

What really matters is that the supply has enough voltage for everything in the loop at 20 mA. Add up the minimum voltage the transmitter needs (see its datasheet), 3.0 V across the IS4420-M1, the drop of the protection diode and the drop along the cable. For example, a transmitter that needs 12 V, plus 3.0 V for the module, about 0.7 V for the diode and 1 V for a long cable, needs at least 16.7 V. A 24 V supply leaves a comfortable margin.

Pay attention to the polarity when you wire the supply: the current must always flow into the IS4420-M1 through the CIN pad. In this example, the negative terminal of the supply is connected to GND, which is also the ground of the module and of your application.

A diode in series with the CIN pad protects the module if the loop is connected the wrong way round: the diode simply blocks the reversed current. The CIN pad has no protection of its own, so without this diode a reversed loop can permanently damage the module. The example uses an SS14 Schottky diode, which drops less voltage than a standard diode.

Block C: IS4420-M1 Current Loop Receiver

This is the IS4420-M1 itself. It was designed so that you have very little to do around it: you power it, connect the loop and read the result.

Power Supply

Connect the 3V3 pad to your 3.3 V supply and the GND pad to ground. The module already has its own decoupling and filtering on board, but a 100 nF ceramic capacitor between 3V3 and GND, close to the module, is still good practice (see 3V3 and GND Pads). If the 3.3 V comes from far away or is noisy, you can add a 2.2 µF capacitor as well.

I2C Connection

Connect the SDA and SCL pads to the I2C pins of your microcontroller. Both lines need pull-up resistors, which are shown in Block D.

Choose the I2C address with the ADR pad and the speed with the SPD pad (see ADR Pad and SPD Pad sections). In the example both pads are connected to GND, which gives address 26 and 100 kHz. The module reads them only at power-up, so set them before you power it.

If you only use I2C, leave the TX pad floating. Never connect TX to GND, or the module will not boot.

Current Loop Input

Connect the loop so that its current flows into the module through the CIN pad and comes back out through the GND pad. This is the current the module measures. In the example it reaches CIN through the protection diode of Block B.

Never connect a power supply straight to the CIN pad: there must always be a transmitter in the loop to limit the current. Without it, nothing keeps the current within the 4 –20 mA range, and the module will be permanently damaged.

Ground Connection

The GND pad is the ground of the module and the return of the current loop. In this example it is also the negative terminal of the loop power supply and the ground of your application: they are all the same point, because the input of the IS4420-M1 is not isolated. Keep this in mind when you connect isolated transmitters or transmitters with more than two wires.

Block D: I2C-Serial Interface

The SDA and SCL lines need one pull-up resistor each, to 3.3 V or 5 V; the module has none of its own. Typical values are 4.7 kΩ at 100 kHz, 2 kΩ at 400 kHz and 1 kΩ at 1 MHz. These values suit a short connection on the same board; longer wires need lower values.

If you pull them up to 5 V, keep the module powered whenever that 5 V is present: when it is unpowered, its pads accept 4 V at most.

Other I2C devices can share the same two lines, as the drawing shows. You can even connect up to five IS4420-M1 modules, each with its own address.

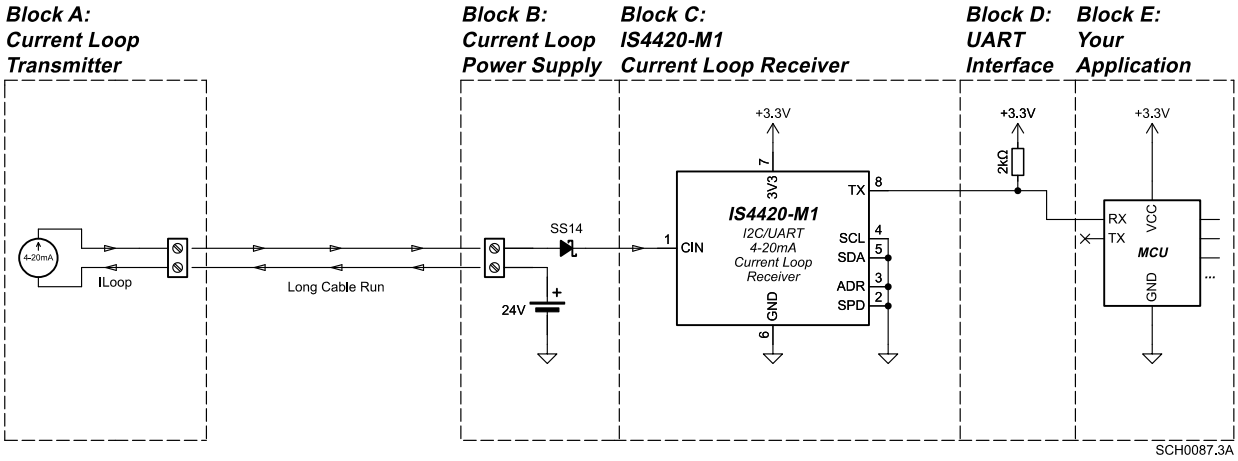
Block E: Your Application

This is the rest of your product. Usually it is a microcontroller, but it can also be a microprocessor or a single-board computer such as a Raspberry Pi: anything with an I2C-serial interface or a UART can read the IS4420-M1.

Reading the current is a simple register read of `MA_INT` and `MA_FRAC`; the Usage section explains how.

The UART output is plain ASCII text, so you can also connect the TX pad to a computer through any USB-to-UART adapter and see the readings in a serial terminal, in a Python script or in any program that reads a serial port. INACKS also provides a Windows app and a Python script for this (see Tools).

7.2 Reading Current Loop via UART



Wiring

Reading the loop through the UART is very similar to reading it through I2C, and the loop side of the circuit (Blocks A, B and C) stays exactly the same. What changes is the connection to your application: instead of the SCL and SDA pads, you use the TX pad.

Connect the TX pad of the module to the RX pin of your microcontroller. The TX pin of your microcontroller stays unconnected, because the module only talks and never listens.

The TX pad is an open-drain output, so it **needs a pull-up resistor** (Block D). You can pull it up to 3.3 V or to 5 V, which makes it work with both 3.3 V and 5 V microcontrollers. A 2.2 kΩ resistor is a good value; do not go above 4.7 kΩ, or the edges become too slow for the UART. The example uses 2 kΩ.

If you pull it up to 5 V, keep the module powered whenever that 5 V is present: when it is unpowered, the TX pad accepts 4 V at most.

The UART settings are **115200 bps, 8 data bits, no parity and 1 stop bit** (8N1).

If you don't use I2C, connect the SCL, SDA, ADR and SPD pads straight to GND, as in the example. No resistors are needed, which keeps the circuit simple, and because none of these pads is left floating, they cannot pick up noise and carry it into the module.

Advantages

The UART needs only one pin on your microcontroller (RX), while I2C needs two (SCL and SDA).

The data is also very easy to process. The module sends plain ASCII text with a fixed length, and every value is always at the same position in the line, so your program just takes the characters it needs. There is no protocol to implement and nothing to ask for: the module sends a new line about ten times per second on its own.

I2C, on the other hand, lets several devices share the same two lines: up to five IS4420-M1 modules, each with its own address, plus any other I2C devices you have.

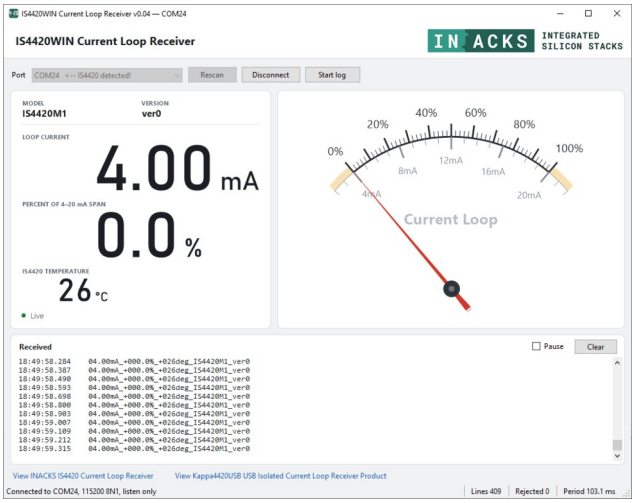
8 Firmware Examples

Firmware examples are provided for demonstration purposes, with examples for Arduino, Raspberry Pi (Python) and PC/Mac (Python). All examples are available as source code on GitHub.

Example Code	Task	Platform	Language
ISXMPL4420ex1 https://github.com/inacks/ISXMPL4420ex1	Read Current Loop via UART	Raspberry Pi	Python
ISXMPL4420ex2 https://github.com/inacks/ISXMPL4420ex2	Read Current Loop via I2C	Raspberry Pi	Python
ISXMPL4420ex3 https://github.com/inacks/ISXMPL4420ex3	Read Current Loop via UART	Arduino UNO	C
ISXMPL4420ex4 https://github.com/inacks/ISXMPL4420ex4	Read Current Loop via I2C	Arduino UNO	C
ISXMPL4420ex5 https://github.com/inacks/ISXMPL4420ex5	Read Current Loop via UART	PC/Mac	Python

9 Tools

9.1 Windows App



Name: IS4420WIN

Download Link: https://github.com/inacks/IS4420WIN

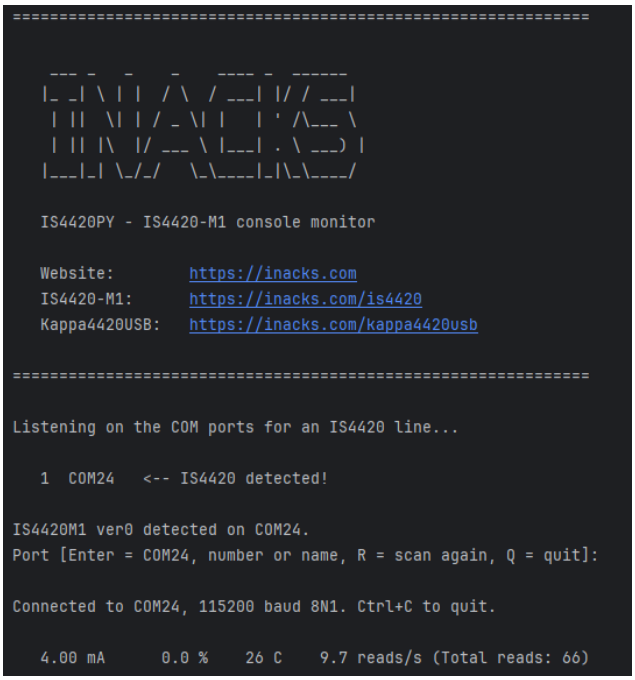
Platform: Windows

Developer: INACKS

License: Free (MIT)

Description: IS4420WIN is a free Windows program for the IS4420-M1. It receives the data from the TX pad of the IS4420-M1 or the Kappa4420USB at your COM port and shows the loop current, in mA and as a percentage, and the module temperature in a very intuitive way. The program can also log the loop current to a CSV file. Its source code is on GitHub, so it is also a good starting point for your own program.

9.2 PC/Mac Python Script



Name: IS4420PY (ISXMPL4420ex5)

Download Link:

https://github.com/inacks/ISXMPL4420ex5

Platform: PC or Mac with Python 3

Developer: INACKS

Description: IS4420PY is a small Python example for PC/Mac. It receives the data from the TX pad of the IS4420-M1 or the Kappa4420USB at your COM port and shows the loop current, in mA and as a percentage, and the module temperature. Its source code is on GitHub, so it is also a good starting point for your own program.

9.3 USB Adapter



Name: Kappa4420USB

Product Page: https://inacks.com/kappa4420usb

Developer: INACKS

Description: The Kappa4420USB is an IS4420-M1 mounted on an isolated USB-to-UART adapter: a complete current loop receiver for your computer. When you plug it in, a COM port appears and continuously receives the loop current. The USB side is isolated from the current loop, so your PC and your loop don’t share a ground.

10 Troubleshooting

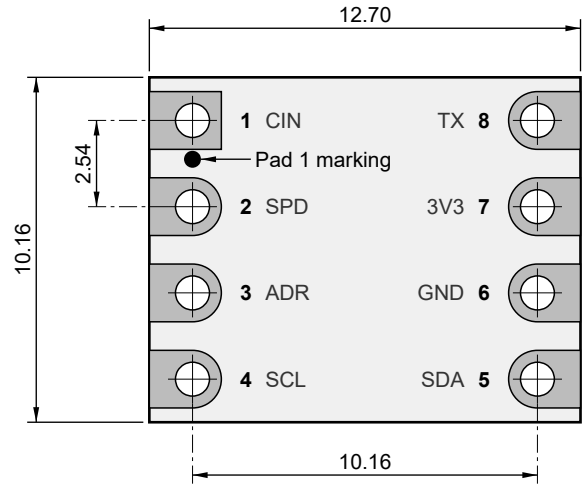
If the module does not behave the way you expect, this chapter helps you find out why. Most problems come from the wiring around the module, not from the module itself, so start with the simple checks. The problems are grouped by what you see.

Problem	Possible Resolution
The Module Does Not Answer on I2C	If the module does not acknowledge its address, check these points, more or less in this order: • **Power.** The 3V3 pad must have 3.3 V, and the GND pad must be connected to the ground of your microcontroller. • **Start-up time.** After power-up, the module needs about half a second before it answers. Wait at least that long before the first access. • **The TX pad.** If you only use I2C, leave TX floating, with nothing connected to it. Never connect TX to GND: if TX is at GND, or is held low while the module powers up, the module does not start. • **The address.** It comes from the ADR pad: 26 with ADR at GND, up to 30 with ADR at 3.3 V (see ADR Pad). Measure the voltage on ADR, and never leave it floating: an open pad gives an unpredictable address. The module reads it only at power-up, so after any change, switch the module off and on again. • **Pull-ups.** SDA and SCL need a pull-up resistor each; the module has none of its own. • **Two devices with the same address.** Every module on the same lines needs its own ADR setting, and no other device can use the same address.
I2C Works, but Not Reliably	If the module answers but some transfers fail, the cause is usually the clock stretching or the pull-ups. • **Clock stretching.** The module holds SCL low for a few milliseconds while it prepares its answer, and for up to about 46 ms right after a calibration SAVE or ERASE. Your master must support clock stretching, and any timeout must be longer than that. • **The register address.** Every read starts by writing the 2-byte register address, followed by a Repeated Start and the read itself. A read that does not start this way is not supported. • **Pull-up values.** Higher speeds and longer wires need lower pull-up values (see SCL and SDA Pads). Keep the wires short and away from noisy signals.
The Module Reports No Measurement (0xFE or ERROR)	This means the module has no measurement to give, so it reports a special value instead of a wrong one: `MA_INT`, `PERCENT_INT` and `MODULE_TEMP` read `0xFE`, and the UART line shows `ERROR`. It is normal for a moment after power-up, until the first measurement is ready, about 0.6 s. If it stays like that, switch the module off and on again. If the problem does not go away, contact INACKS.
The Module Reports Over-Range (0xFF or EOVER)	The module reports over-range when it reads more than 21.50 mA: `MA_INT` and `PERCENT_INT` read `0xFF`, and the UART line shows `EOVER`. Look at the loop: • **The transmitter may be signaling a fault.** Many transmitters drive their output above 21 mA on purpose when something is wrong, for example a broken sensor. Check the transmitter. • **Ripple on the loop.** If the loop current has a lot of ripple, its peaks can reach the top of the input range, and the module reports over-range a little below 21.50 mA. Reduce the ripple at its source. • **The supply of the module.** If the 3V3 pad gets less than 3.3 V, the top of the measuring range comes down (see 3V3 and GND Pads). • **A power supply connected straight to CIN.** Without a transmitter to limit it, the current is far too high and the module is damaged. Disconnect it at once, and see The Module Was Damaged by a Wiring Mistake.
The Module Reads 0 mA, or Less Than Expected	A reading of 0 mA, or very close to it, means that no current flows through the CIN pad. Check that: • The loop is closed: no broken wire and no loose terminal. • The loop power supply is on, and the transmitter is powered. • The current flows into CIN. With a protection diode in series, a loop connected the wrong way round simply carries no current (see Hardware Examples). • The transmitter is working. Many transmitters drive their output below 4 mA, for example to 3.6 mA or less, to signal a fault. If the reading is right at low currents but too low near 20 mA, the loop power supply probably does not have enough voltage. At 20 mA the module alone takes 3.0 V, on top of what the transmitter and the cable need (see Block B in Hardware Examples). If the reading is always a little low, part of the current may be going back through another path, for example through a second ground connection between the transmitter and your system. The module only measures what flows through CIN and GND, so make sure the loop current has only one way back: through the GND pad.
The Reading Is Not Accurate	If the readings are stable but not as accurate as you expect, check the calibration first. • Read `CAL_STATUS`. A calibrated module reads `0x01`. If it reads `0x00`, the calibration has been erased and the module works uncalibrated: calibrate it again (see Two-Point Calibration). • If `CAL_ERROR` reads `0x0D` after power-up, the stored calibration was not valid and the module is working uncalibrated. Calibrate it again. • If someone calibrated the module again, check the reference they used. A new calibration replaces the factory one, and any error in its reference shows up in every reading. • Compare against a good reference. The module reports the current with a resolution of 0.01 mA, so the meter you compare against must be more accurate than that. • The reading changes slightly with temperature and, much less, with the supply voltage. For the best accuracy, calibrate at the temperature and supply voltage of your application.
The Readings Jump or Look Noisy	• **Read MA_INT and MA_FRAC together.** Read `MA_INT` and `MA_FRAC` in the same read operation. If you read them separately, they can come from two different measurements, and 12.99 mA can show up as 12.00 or 13.99 mA. • **Interference on the loop.** The module averages each measurement over 100 ms, which removes mains ripple, but strong interference still moves the reading. Use a shielded or twisted-pair cable, and check how its shield is grounded. • **Ground loops.** If the loop and your system are connected to ground in more than one place, currents can flow where they should not. Connect the grounds at a single point.
Nothing Comes Out of the TX Pad	• **The pull-up.** TX is an open-drain output: without a pull-up resistor the line never goes high, and nothing arrives. Fit a pull-up to 3.3 V or 5 V; 2.2 kΩ is recommended. • **The wiring.** Connect TX to the RX input of your microcontroller or USB-to-UART adapter, and connect their grounds together. • **The timing.** The first line comes about 0.6 s after power-up, and then a new line about every 103 ms.

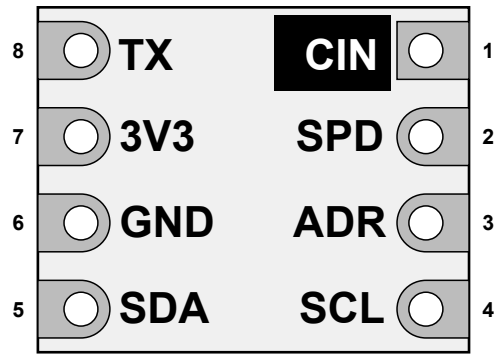
The UART Shows Strange Characters	• **The serial settings.** They must be 115200 bps, 8 data bits, no parity and 1 stop bit (8N1). • **The edges of the signal.** A weak pull-up or a long cable makes the edges slow. Keep the pull-up at 4.7 kΩ or less, and use a lower value for long lines. • **Cut lines.** If you start listening in the middle of a line, the first piece is cut, and the host may also receive one stray byte at power-up. Keep only the lines that have exactly 37 characters before CR LF (see UART Description).
A Calibration Command Does Not Work	After each calibration command, read `CAL_ERROR`: it tells you what went wrong (see CAL_ERROR Register). The most common codes are: • `0x01`: the key was wrong or missing. Write the command to `CAL_CMD` and `0xA5` to `CAL_KEY` in the same write. • `0x03`: a capture was still running. Wait until BUSY (bit 3 of `CAL_STATUS`) is 0, and send the command again. • `0x07`: the current moved more than 20 µA during the capture. Let the loop settle, check it for ripple, and capture the point again. • `0x08` to `0x0B`: the save was refused. Both points must be captured, at least 4 mA apart, and the correction must stay within the limits described in Two-Point Calibration. Check the reference values and the loop current, and calibrate again. Also remember to wait 60 ms after a SAVE or an ERASE before the next I2C access.
The Module Was Damaged by a Wiring Mistake	The CIN pad has no protection. A loop power supply connected straight to CIN, without a transmitter to limit the current, pushes far too much current through the module: it gets hot and is permanently damaged. A reversed loop without a protection diode, or a reversed 3.3 V supply, may also permanently damage it. A damaged module must be replaced. Before you connect a new one, check the wiring, and add the protection diode shown in Hardware Examples.
Still Stuck?	If none of this solves the problem, contact INACKS (see Appendix). Tell us what you see, how the module is connected, and the values of `CAL_STATUS`, `CAL_ERROR` and `FIRMWARE_REV`; this helps us find the cause much faster.

11 Mechanical

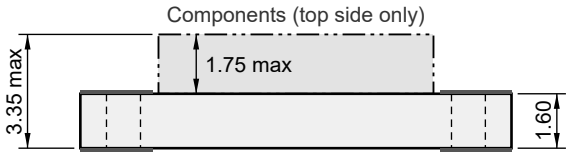
Dimensions



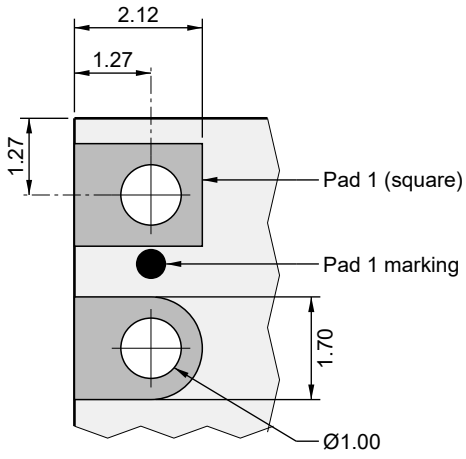
TOP VIEW (component side)



BOTTOM VIEW (labelled side)



SIDE VIEW



PAD DETAIL (top view, enlarged)

NOTES

1. Dimensions in millimetres. Not to scale.
2. Each pad has a plated through-hole and copper on both sides of the board.
3. Pad 1 marking is done by a white circle on the top side of the PCB and/or a unique square pad.
4. Pad names are printed on the bottom side only. The bottom view is the mirror image of the top view: check pad 1 before soldering.
5. Components on the top side only; the bottom side is flat. Heights calculated from component data.
6. Host PCB: for 2.54 mm headers, Ø1.00 mm plated holes at the module pad positions. To solder the module flat, host pads at least as large as the module pads (see PAD DETAIL).

Evadrw0207A

Content

IS4420: I2C/UART Current Loop Receiver 1
Revision History 2
Document Revision 2
Module Revision 2
Firmware Revision 2
Ordering Information 2
1 Electrical Specification 3
Absolute Maximum Ratings 3
Recommended Operating Conditions 3
Electrical Characteristics 3
Current Loop Characteristics 4
2 Detailed Description 5
2.1 Description 5
2.2 Wire Configuration 5
2.3 Measurement Principle 6
2.4 Two-Point Calibration 7
2.5 Usage 8
3 Pads Description 9
3.1 CIN Pad 9
3.2 TX Pad 9
3.3 SPD Pad 10
3.4 ADR Pad 10
3.5 SCL and SDA Pads 11
3.6 3V3 and GND Pads 11
4 UART Description 12
5 I2C Memory Map Description 13
5.1 Memory Map Table 13
5.2 MA_INT and MA_FRAC Registers 14
5.3 PERCENT_INT and PERCENT_FRAC Registers 15
5.4 MODULE_ID Register 16
5.5 FIRMWARE_REV Register 16
5.6 MODULE_TEMP Register 17
5.7 CAL_STATUS Register 17
5.8 CAL_ERROR Register 18
5.9 CAL_REF_LO and CAL_REF_HI Registers 19
5.10 CAL_CMD and CAL_KEY Registers 20
6 I2C-Serial Interface Description 21
6.1 Highlights 21
6.2 Single Byte Write 22
6.3 Multiple Byte Write 23
6.4 Single Byte Read 24
6.5 Multiple Byte Read 25
7 Hardware Examples 26
7.1 Reading Current Loop via I2C 26
7.2 Reading Current Loop via UART 28
8 Firmware Examples 29
9 Tools 30
9.1 Windows App 30
9.2 PC/Mac Python Script 30
9.3 USB Adapter 31
10 Troubleshooting 32
11 Mechanical 34
Content 35
Appendix 36
Documentation Feedback 36
Sales Contact 36
Customization 36
Trademarks 36
Disclaimer 37

Appendix

Documentation Feedback

Feedback and error reporting on this document are very much appreciated. Please indicate the code or title of the document.

feedback@inacks.com

Sales Contact

For special order requirements, large volume orders, or scheduled orders, please contact our sales department at:

sales@inacks.com

Customization

INACKS can develop new products or customize existing ones to meet specific client needs. Please contact our engineering department at:

engineering@inacks.com

Trademarks

This company and its products are developed independently and are not affiliated with, endorsed by, or associated with any official protocol or standardization entity. All trademarks, names, and references to specific protocols remain the property of their respective owners.

Disclaimer

Limited warranty and liability — Information in this document is believed to be accurate and reliable. However, INACKS does not give any representations or warranties, expressed or implied, as to the accuracy or completeness of such information and shall have no liability for the consequences of use of such information. INACKS takes no responsibility for the content in this document if provided by an information source outside of INACKS.

In no event shall INACKS be liable for any indirect, incidental, punitive, special or consequential damages (including - without limitation - lost profits, lost savings, business interruption, costs related to the removal or replacement of any products or rework charges) whether or not such damages are based on tort (including negligence), warranty, breach of contract or any other legal theory.

Notwithstanding any damages that customer might incur for any reason whatsoever, INACKS's aggregate and cumulative liability towards customer for the products described herein shall be limited in accordance with the Terms and conditions of commercial sale of INACKS.

Right to make changes — INACKS reserves the right to make changes to information published in this document, including without limitation specifications and product descriptions, at any time and without notice. This document supersedes and replaces all information supplied prior to the publication hereof.

Suitability for use — INACKS products are not designed, authorized or warranted to be suitable for use in life support, life-critical or safety-critical systems or equipment, nor in applications where failure or malfunction of an INACKS product can reasonably be expected to result in personal injury, death or severe property or environmental damage. INACKS and its suppliers accept no liability for inclusion and/or use of INACKS products in such equipment or applications and therefore such inclusion and/or use is at the customer's own risk.

Quick reference data — The Quick reference data is an extract of the product data given in the Limiting values and Characteristics sections of this document, and as such is not complete, exhaustive or legally binding.

Applications — Applications that are described herein for any of these products are for illustrative purposes only. INACKS makes no representation or warranty that such applications will be suitable for the specified use without further testing or modification.

Customers are responsible for the design and operation of their applications and products using INACKS products, and INACKS accepts no liability for any assistance with applications or customer product design. It is customer's sole responsibility to determine whether the INACKS product is suitable and fit for the customer's applications and products planned, as well as for the planned application and use of customer's third party customer(s). Customers should provide appropriate design and operating safeguards to minimize the risks associated with their applications and products.

INACKS does not accept any liability related to any default, damage, costs or problem which is based on any weakness or default in the customer's applications or products, or the application or use by customer's third party customer(s). Customer is responsible for doing all necessary testing for the customer's applications and products using INACKS products in order to avoid a default of the applications and the products or of the application or use by customer's third party customer(s). INACKS does not accept any liability in this respect.

Limiting values — Stress above one or more limiting values (as defined in the Absolute Maximum Ratings System of IEC 60134) will cause permanent damage to the device. Limiting values are stress ratings only and (proper) operation of the device at these or any other conditions above those given in the Recommended operating conditions section (if present) or the Characteristics sections of this document is not warranted. Constant or repeated exposure to limiting values will permanently and irreversibly affect the quality and reliability of the device.

Terms and conditions of commercial sale — INACKS products are sold subject to the general terms and conditions of commercial sale, as published at http://www.inacks.com/comercialsaleterms, unless otherwise agreed in a valid written individual agreement. In case an individual agreement is concluded only the terms and conditions of the respective agreement shall apply. INACKS hereby expressly objects to applying the customer's general terms and conditions with regard to the purchase of INACKS products by customer.

No offer to sell or license — Nothing in this document may be interpreted or construed as an offer to sell products that is open for acceptance or the grant, conveyance or implication of any license under any copyrights, patents or other industrial or intellectual property rights.

Export control — This document as well as the item(s) described herein may be subject to export control regulations. Export might require a prior authorization from competent authorities.

Non-automotive qualified products — This INACKS product is not suitable for automotive use. It is neither qualified nor tested in accordance with automotive testing or application requirements. INACKS accepts no liability for inclusion and/or use of non-automotive qualified products in automotive equipment or applications.

Protocol Guidance Disclaimer: The information provided herein regarding the protocol is intended for guidance purposes only. While INACKS strive to provide accurate and up-to-date information, this content should not be considered a substitute for official protocol documentation. It is the responsibility of the client to consult and adhere to the official protocol documentation when designing or implementing systems based on this protocol.

INACKS make no representations or warranties, either expressed or implied, as to the accuracy, completeness, or reliability of the information contained in this document. INACKS shall not be held liable for any errors, omissions, or inaccuracies in the information or for any user's reliance on the information.

The client is solely responsible for verifying the suitability and compliance of the provided information with the official protocol standards and for ensuring that their implementation or usage of the protocol meets all required specifications and regulations. Any reliance on the information provided is strictly at the user's own risk.

Certification and Compliance Disclaimer: Please be advised that the product described herein has not been certified by any competent authority or organization responsible for protocol standards. INACKS do not guarantee that the product meets any specific protocol compliance or certification standards.

It is the responsibility of the client to ensure that the final product incorporating this product is tested and certified according to the relevant protocol standards before use or commercialization. The certification process may result in the product passing or failing to meet these standards, and the outcome of such certification tests is beyond our control.

INACKS disclaim any liability for non-compliance with protocol standards and certification failures. The client acknowledges and agrees that they bear sole responsibility for any legal, compliance, or technical issues that arise due to the use of this product in their products, including but not limited to the acquisition of necessary protocol certifications.