IS3710: I2C DMX512 Receiver

Full Universe, 512 Channels

Main Advantages

- Reduces engineering time and costs
- Reduces product time-to-market
- Makes the DMX protocol transparent to both the microcontroller and the engineer
- Provides a low-cost solution
- Fewer ISRs, lower microcontroller CPU load
- Reduces microcontroller memory usage
- Saves microcontroller dedicated pins with I2C
- Minimizes impact on microcontroller peripherals (no need for dedicated timers, UARTs, etc.)
- Compact, easy-to-solder SO8N package
- I2C speeds: 100kHz, 400kHz, and 1MHz.

Applications

- Custom Lighting
- LED PAR Lights
- Stage Lighting
- Museum Lighting
- Smoke and Fog Machines
- Custom DMX Devices
- Artistic Installations and Devices
- Light Festivals Devices
- Interactive Devices
- Dimmers

DMX Characteristics

- DMX512A protocol
- Receives from 1 up to all 512 DMX Channels
- Integrated filter to reject non-DMX data
- Compatible with delta transmitters (transmit only on data change)

General Description

The IS3710 is a DMX Receiver chip. It stores all 512 DMX Channels in its memory map, and the data is accessible by a microcontroller via I2C.

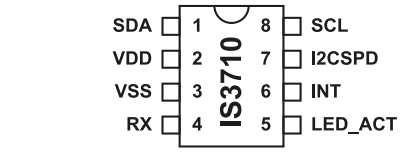
The chip features an interrupt pin to flag new DMX data available, and an activity LED pin as a visual indicator of DMX signal presence.

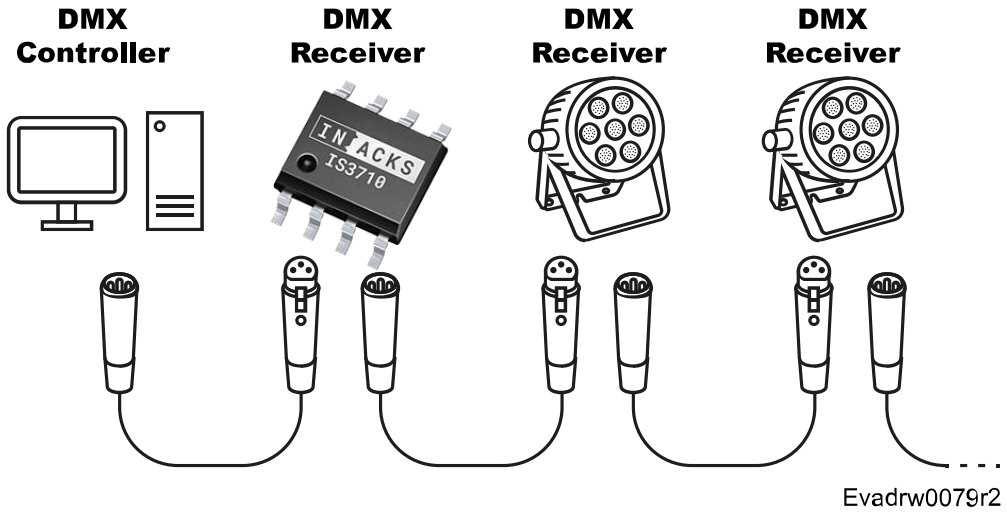
It operates as an I2C slave and supports 100 kHz, 400 kHz, and 1 MHz communication speeds.

The IS3710 frees your microcontroller from the timing constraints, memory requirements, and ISR load of receiving DMX data, reducing the need for timers, flash, and RAM. It also eliminates the need for a dedicated pins by operating over a shared bus (I2C).

The device operates at 3.3 V, with 5 V-tolerant I2C and RX pins, enabling compatibility with 3.3 V or 5 V transceivers and microcontrollers. It is offered in Industrial (–40 °C to +85 °C) and Extended (–40 °C to +125 °C) temperature ranges.

Part Number	Package	Op. Temperature
IS3710-S8-I	SO8N	-40ºC to +85ºC
IS3710-S8-E	SO8N	-40ºC to +125ºC





Evadrw0079r2

1. Electrical Specifications

Absolute Maximum Ratings

Parameter			Min	Max	Unit
Input Voltage	VDD Pin		-0.3	4	V
	SCL Pin		-0.3	5.5	
	SDA Pin		-0.3	5.5	
	RX Pin		-0.3	5.5	
	I2CSPD Pin		-0.3	4	
Current Sourced/Sunk by any I/O or Control Pin				±20	mA
Temperature	Operating Temperature	IS3710-S8-I	-40	+85	°C
		IS3710-S8-I	-40	+125	
	Storage Temperature		-65	+150	
Electrostatic Discharge (T_A = 25°C)	Human-body model (HBM), Class 1C		-2000	+1500	V
	Charged-device model (CDM), Class C2a		-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 Operation Conditions

Parameter	Symbol	Min	Nom	Max	Unit
Supply Voltage	V_{DD}	2.0	3.3	3.6	V
Input Voltage at SCL, SDA and RX Pins	$V_{I/O-IN}$	-0.3	3.3	5.5	
Source/Sink Current at SCL, SDA and RX Pins	$I_{I/O-SS}$	-	-	±6	mA

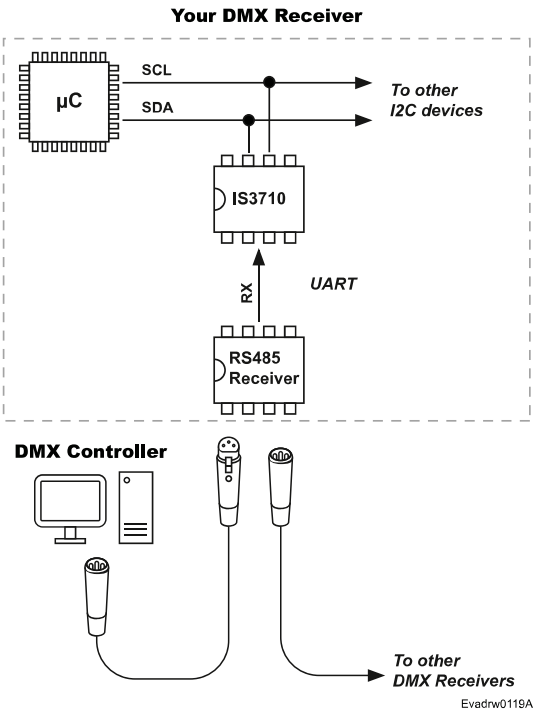
Electrical Characteristics

Parameter		Symbol	Min	Nom	Max	Unit
Current Consumption (T_A = 25°C)		I_{OP}	-	3.40	3.90	mA
Input Voltage	Logical High-Level	V_{IH}	0.7xV_{DD}	-	-	V
	Logical Low-Level	V_{IL}	-	-	0.3xV_{DD}	

2. Detailed Description

2.1. Description

The IS3710 is a protocol-specialized IC that receives DMX512A lighting data and stores it in its internal memory map, making it accessible via I2C to microcontrollers and other I2C master devices.



It can handle from a single channel up to all 512—that is, a full DMX universe. Each DMX channel is mapped to its corresponding I2C memory address; for example, DMX Channel `1` is stored at I2C Address `1`, and so on up to address `512`.

Since the IS3710 receives all DMX channels (from channel `1` to channel `512`), it's up to your firmware to read only the channels your application needs—or read all of them and use only the relevant ones.

The chip incorporates a DMX data filter; all non-DMX packets—those with a starting code different from `0`, such as RDM packets—are automatically rejected.

Upon receiving a valid DMX frame, the IS3710 generates an interrupt signal on the INT pin, which goes high (logic '1') for 500 µs and then automatically returns to low (logic '0').

This optional feature can be used by your I2C master device to read the IS3710 only when new DMX data is available.

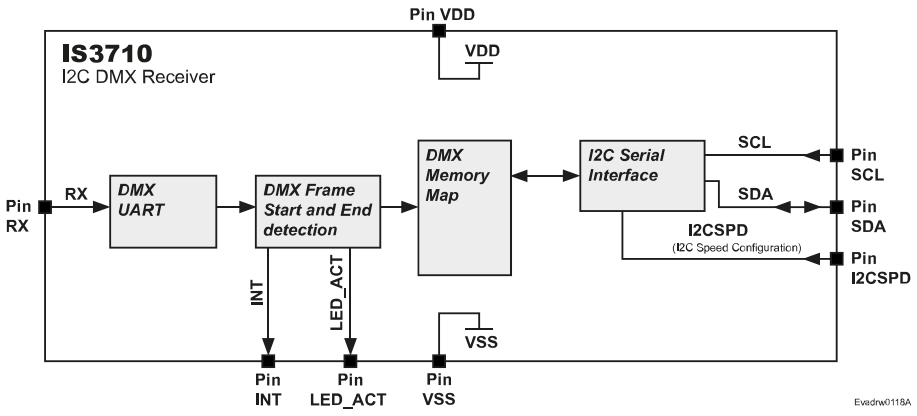
Another optional feature is the LED Activity Pin (`LED_ACT`), which can directly drive a simple LED (up to 6 mA) to indicate that valid DMX data is being received. Placing this DMX activity LED near the DMX connectors provides useful feedback for the operator during DMX system setup. Be sure to use a low-brightness LED to avoid visual distractions during the show.

The IS3710 operates at 3.3V and its RX, SCL and SDA pins are 5V tolerant.

An RS485 receiver or transceiver is required at the RX Pin to adapt the DMX (RS485) voltage levels to TTL-compatible voltage levels for the IS3710. Both 3.3V and 5V transceivers can be used, as RX pin is 5V tolerant.

Connecting DMX (RS485) signals directly to the RX pin without a receiver or transceiver will permanently damage the device.

Refer to chapter *Hardware Examples* for more information.



2.2. Usage

Ensure you have a proper hardware design by:

1. Supplying 3.3 V power to the IS3710.
2. Having an RS485 receiver or transceiver, such as the THVD1400DR, in receiving mode and routed to the IS3710 RX pin.
3. Having your microcontroller connected via I2C on the SCL and SDA pins, with pull-up resistors to 3.3 V or 5 V.
4. Having the I2CSPD pin to GND for 100 kHz, or any other configuration.

Once these basic hardware requirements are met, the firmware task can be executed.

Refer to section *Hardware Examples* section for more details about the hardware design.

The IS3710 functionality is extremely simple: once powered, it automatically receives and stores all the 512 DMX channels in its internal memory map.

The process of reading its DMX data via I2C is the same as it would be used for reading classical EEPROM memory.

Whenever your microcontroller needs DMX data, perform an I2C Multiple Byte Read Operation to the I2C Slave Address `16 (0x10)`, specifying the starting DMX channel (using two bytes) and the number of channels to read.

It is important to note that you need to specify the starting DMX channel—also called the DMX Pointer Register—using two bytes, with the most significant byte first.

This I2C Memory Read Operation is implemented in almost every microcontroller IDE, so provably you don't need to implement it yourself.

You can find implementation examples in STM32, Raspberry Pi and Arduino at the *Firmware Examples* section.

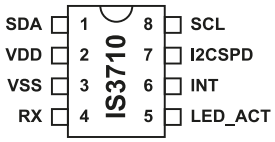
If you're designing a DMX light with 512 colors (a full universe), you'll need to start the I2C Read Operation at register 1 and read up to 512 registers.

If you're designing a DMX light with only 3 colors, the I2C read operation should start at the DMX Address your light is configured to respond to, and you'll only need to read 3 registers—one for each DMX channel.

You can use more DMX channels for additional control features, such as master dimmer, strobe, and others.

When first attempting to communicate with the IS3710, you can verify that I2C communication is working by reading the `CHIP_ID` register, located at register address `513`. This register contains a constant value of `16`.

3. Pin Description



Pin	Name	Type	Description
1	SDA	Open Drain 5V Tolerant	I2C Data pin. Open drain, it requires pull-up.
2	VDD	Supply	3.3V power supply pin. Bypass this pin to GND with a 100nF ceramic capacitor.
3	VSS	Ground	Ground reference pin.
4	RX	Digital Input 5V Tolerant	DMX UART Receive Pin in TTL voltage levels. Attention: Only digital 3.3V or 5V can be applied to this pin. Use an RS485 receiver or transceiver to adapt the DMX voltage levels to TTL voltage levels.
5	LED_ACT	Digital Output Push-Pull	LED Activity Pin. Connect the anode of an LED to this pin through an appropriate resistor.
6	INT	Digital Output Push-Pull	Interruption Pin
7	I2CSPD	Analog Input 0 to 3.3V	I2C-Serial Interface Speed Selection pin. • For 100kHz pull to GND. • For 400kHz make a voltage divider of VDD/2 (1.65V). • For 1MHz pull to VDD (3.3V). Attention: Voltage above 4V will damage the device.
8	SCL	Open Drain 5V Tolerant	I2C Clock pin. Open drain, it requires pull-up.

RX Pin

DMX UART Receive Pin.

This pin receives DMX512 data from the RS485 transceiver. It operates at 3.3V TTL levels and is 5V tolerant.

Use an RS485 transceiver, such as the THVD1400DR, to convert DMX (RS485) differential signals to TTL voltage levels. Refer to the Hardware Examples section for more details.

Important: Applying DMX (RS485) voltage levels directly to this pin will permanently damage the device.

LED_ACT Pin

Activity LED Pin.

This pin generates a toggling signal while DMX data is being detected on the RX pin. This visual indicator is useful for quickly validating that the DMX installation is working properly.

Connect the anode of an LED to this pin through an appropriate resistor. The maximum current this pin can source is 6 mA.

INT Pin

Interruption Pin.

This pin goes high for 500µs to indicate a new DMX data packet has been received.

You can either continuously poll the I2C interface for new data or use this INT pin to poll only when new DMX data has arrived.

The use of this pin is optional.

SCL and SDA Pins

I2C-Compatible Bus Interface Pins.

SCL (Serial Clock Line): This pin is used to synchronize data transfer between the IS3710 device and the microcontroller or other CPU.

SDA (Serial Data Line): This bidirectional pin is used for both sending and receiving data between the IS3710 and the microcontroller or other CPU.

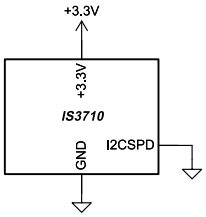
Both pins are open-drain and must be pulled up to 3.3V or 5V. The pull-up resistor value should be chosen based on the bus speed and capacitance. Typical values are 4.7kΩ for Standard Mode (100kbps) and 2kΩ for Fast Mode (400kbps) at both 3.3V and 5V.

I2CSPD Pin

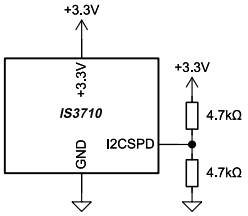
I2C-Serial Interface Speed Selection Pin.

This pin configures the IS3710 internal I2C-Serial Interface timings and filters to properly work with the selected bus speed.

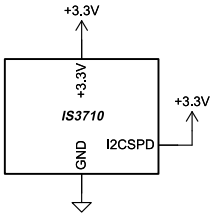
For a **100 kHz** setting, set the I2CSPD pin to VSS.



For a **400 kHz** setting, set the I2CSPD to 1.65 V (VDD/2) using a balanced voltage divider. This can be achieved by placing two 4.7 kΩ resistors from the I2CSPD pin: one to VDD and the other to VSS.



For a **1000 MHz** setting, set the I2CSPD pin to 3.3 V.



Important Remarks:
Voltages above 4 V on this can permanently damage the device.

A mismatch between the configured I2C speed and the actual operating I2C speed (e.g., setting I2CSPD to GND for 100 kHz but operating at 1 MHz) can lead to an inconsistent state where some I2C messages are processed while others are not.

Ensure a proper match between the actual operating speed and the configured speed at the I2CSPD pin: If your bus works at 100kHz, ensure the I2CSPD pin is tied to VSS. If it works at 400kHz ensure the pin is at 1.65V. If it works at 1000MHz, ensure the pin is at 3.3V.

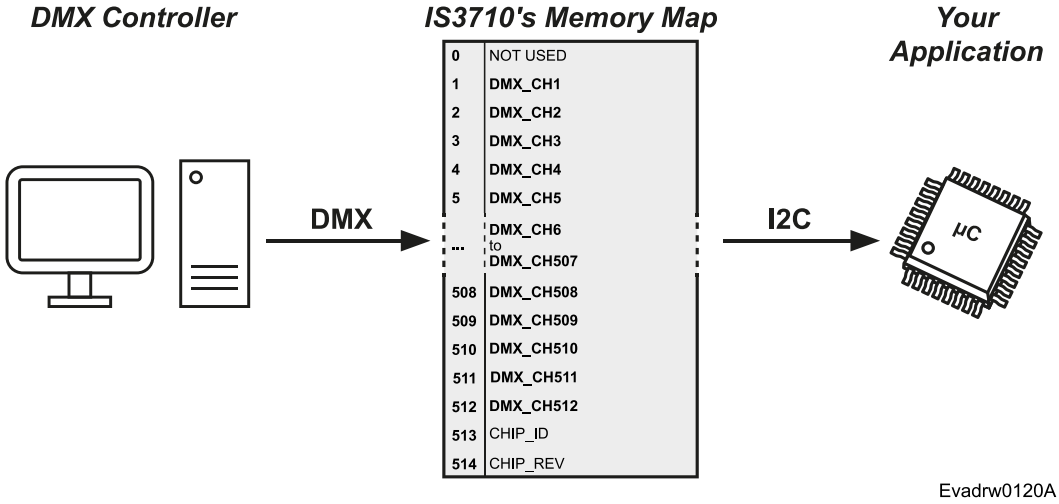
4. Memory Map Description

The IS3710 is organized internally as a single page containing 515 registers of 8 bits each, with addresses ranging from `0` to `514`. Accessing any register requires a 2-byte address, even for registers below address 256.

The memory is RAM type, which means you can access any register and read any number of them; there is no page block limitation.

There are two types of registers: the DMX channel registers (`DMX_CHx`) and the chip details registers. The DMX channel registers are mapped in the memory map so that each register number matches its corresponding DMX channel number, making it easy for engineers to understand and address the memory. For example, DMX Channel 15 data is stored into the memory register 15.

The two chip information registers contain the Chip ID, which is fixed throughout the product's lifetime, and the Chip Revision, which changes if the chip undergoes any revision. The Chip ID can be used for chip detection in the I2C-Serial Interface during firmware development.



Evadrw0120A

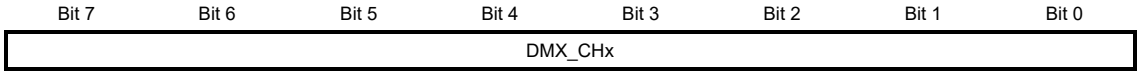
4.1. DMX_CHx Registers

The `DMX_CHx` registers contain the values of their corresponding DMX channels received from the DMX controller. Each register number matches the DMX channel number. For example, DMX Channel 12 is stored in the `DMX_CH12` register.

There are a total of 512 `DMX_CHx` registers, which are volatile RAM: if the chip loses power, the registers will reset to `0` on power-up until the next DMX data is received. You can access and read any register individually or read a block of them.

These registers are read-only.

Name: DMX_CHx
Description: DMX Channel Values
Address Range: 1 to 512 (0x001 to 0x200)
Memory Type: Volatile RAM
Allowed values: 0 to 255 (0x00 to 0xFF)
Reset value: 0 (0x00)



4.2. CHIP_ID Register

The `CHIP_ID` register contains the chip identifier, which is a fixed value of `16`. This value is used for production tracking. It is stored in ROM and will not change throughout the product's lifecycle.

Since this register value is constant, reading it during firmware development can help verify that I2C communications are working and that the chip's memory can be properly read.

This register is read-only.

.

Name: CHIP_ID
Description: Chip Identification Number.
Address: 513 (0x201)
Memory Type: ROM
Value: 16 (0x10)

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

4.3. CHIP_REV Register

The `CHIP_REV` register indicates the chip revision.

This value is intended for production and product tracking. It is stored in ROM and may change throughout the product's lifecycle.

This register is read-only.

.

Name: CHIP_REV
Description: Chip Revision Register
Address: 514 (0x202)
Memory Type: ROM
Value: (Depends on the revision)

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

5. I2C Description

The IS3710 operates as a Slave in the I2C-Serial Interface. It supports Standard Mode (100kHz), Fast Mode (400kHz), and Fast Mode Plus (1MHz). The I2C-Master device, typically a microcontroller or a microprocessor, initiates and manages all read operations to the Slave.

The IS3710 is represented on the bus by the I2C device address: `16 (0x10)`.

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 4.7kΩ for Standard Mode (100kHz) and 2kΩ for Fast Mode and Fast Mode Plus (400kHz and 1MHz).

The IS3710's I2C pins high state can be either 3.3V or 5V. 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 controls the Serial Clock (SCL) line, which generates the synchronous clock used by the Serial Data (SDA) line to transmit data.

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 515 registers, each 8 bits wide. Addressing a register requires a 2-byte pointer (DMX Pointer Register).

The operability of the Read commands of the IS3710 is very similar to an EEPROM memory. Thinking of the IS3710 as an EEPROM memory is a good analogy to quickly understand how to communicate with the device.

5.1. Highlights

- **I2C Device Address**: `16 (0x10)`

- **I2C Memory Map Registers Size**: 8-bit

- **Compatible I2C Speeds**:
  - Standard Mode (100kHz), recommended SCL and SDA pull-up value: 4.7kΩ
  - Fast Mode (400kHz), recommended SCL and SDA pull-up value: 2kΩ
  - Fast Mode Plus (1MHz), recommended SCL and SDA pull-up value: 2kΩ

- **Supported Operations**:
  - Single-Byte Read
  - Multiple-Byte Read (up to 515 registers)

- **Overreading the memory**:
  - If a read operation starts at a valid memory address (`0` to `514`) and continues past the last valid address, it will roll over to address `0`.
  - Starting a read operation at an invalid memory address (greater than `514`) will return a value of 0xFF.

5.2. Read Operations

5.2.1. Single Byte Read

Reading a single byte is an action performed by the microcontroller (I2C-Master) to access any register within the IS3710 memory (I2C-Slave), regardless of the last read position. To perform this action, the microcontroller must load the DMX channel into the IS3710's DMX Pointer Register. Once the address is set, the microcontroller can retrieve the DMX data from the specified register.

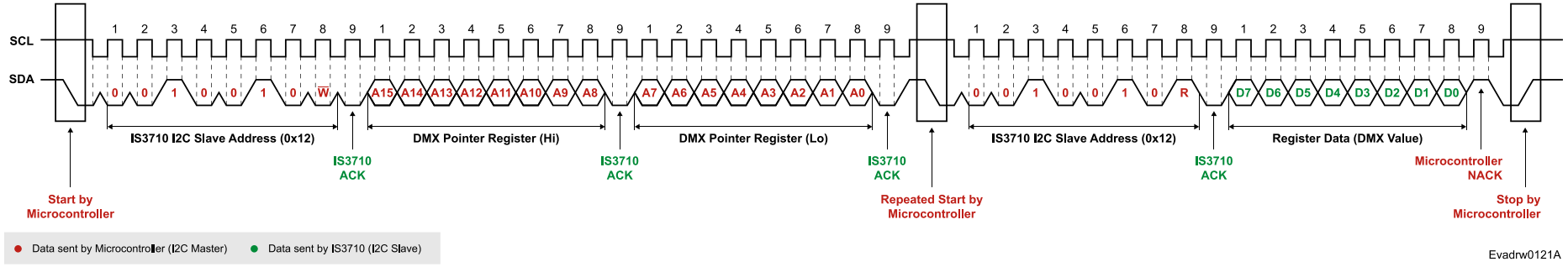
To initiate the Single Byte Read operation, the following steps must be performed from the beginning: The microcontroller starts by pulling SDA low while SCL is high to generate a Start Condition. It then sends the IS3710 I2C device slave address `16 (0x10)` with the R/W bit set to '`0`' (write). Upon receiving the device address, the IS3710 acknowledges it. Subsequently, the microcontroller sends the two bytes of the DMX Pointer Register address: the most significant byte first, followed by the less significant byte, each acknowledged by the IS3710. This sets the address of the next DMX channel to be read in the DMX Pointer Register.

Next, the content of the DMX Pointer Register needs to be read.

The microcontroller generates a Repeated Start Condition, followed by the IS3710 I2C device address `16 (0x10)` with the R/W bit set to '`1`' (read), instructing the IS3710 to retrieve data. The IS3710 acknowledges and responds with the DMX data, which the microcontroller does not acknowledge (NACK). Finally, the microcontroller issues a Stop Condition by raising the SDA line while the SCL is high.

Invalid Memory Addressing

The valid memory range of the IS3710 goes from addresses 0 to 514. If a Read Operation is performed with a Pointer Register higher than 514, the read result will be 0xFF.



5.2.2. Multiple Byte Read

Multiple Byte Read operation functions similarly to Single Byte Read but can read a block of up to 515 registers in a single operation, corresponding to the full memory map.

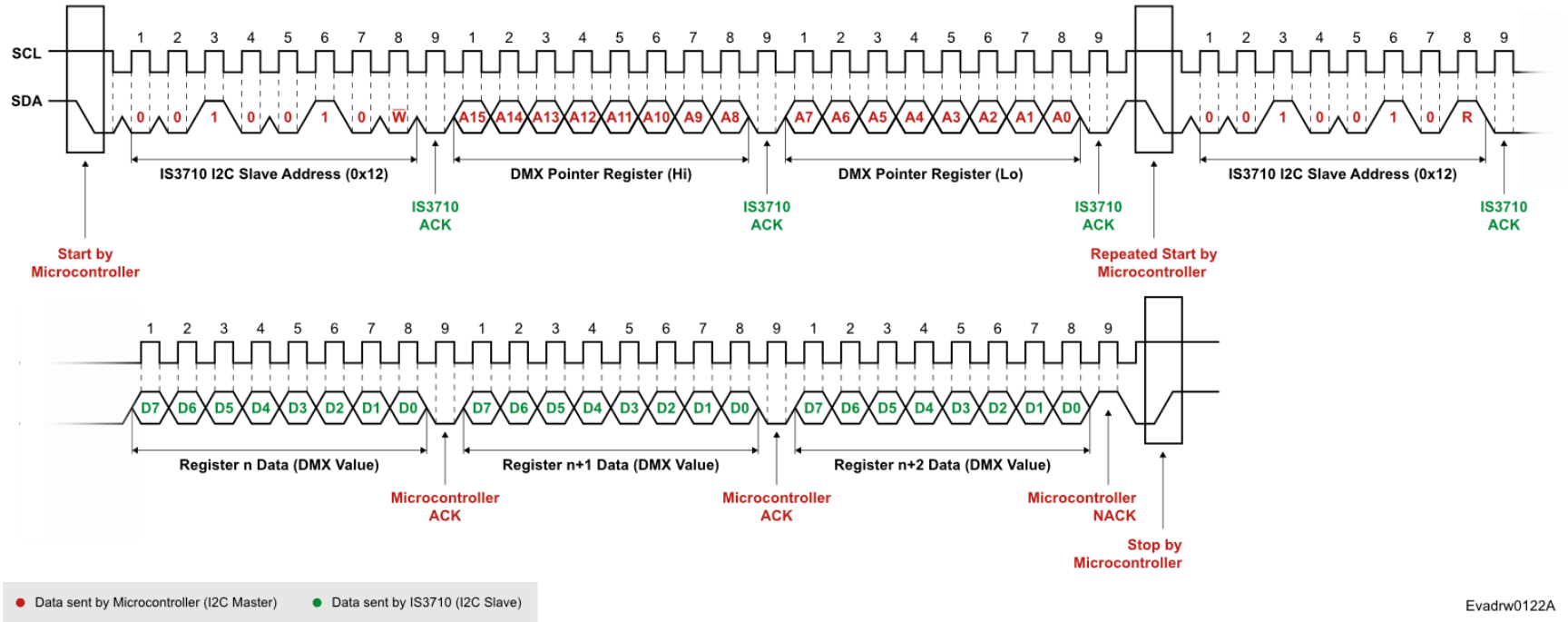
To perform a Multiple Byte Read operation, follow the same procedure as for a Single Byte Read until the first byte is received. After receiving the first byte, instead of generating a Not Acknowledge (NACK), the microcontroller should continue acknowledging (ACK) each received data byte from the IS3710 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 Not Acknowledge (NACK) and a Stop Condition.

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

Invalid Memory Addressing
The valid memory range of the IS3710 goes from addresses `0` to `514`.

If the Read Operation is performed with a Pointer Register within the valid memory range (`0` to `514`), but the data retrieval extends beyond register `514`, a rollover to position `0` will occur. For example, the value of register `516` will correspond to the content of register `1` (`DMX_CH1`).

If a Read Operation is performed with a Pointer Register value higher than `514`, the read result will be `0xFF`.



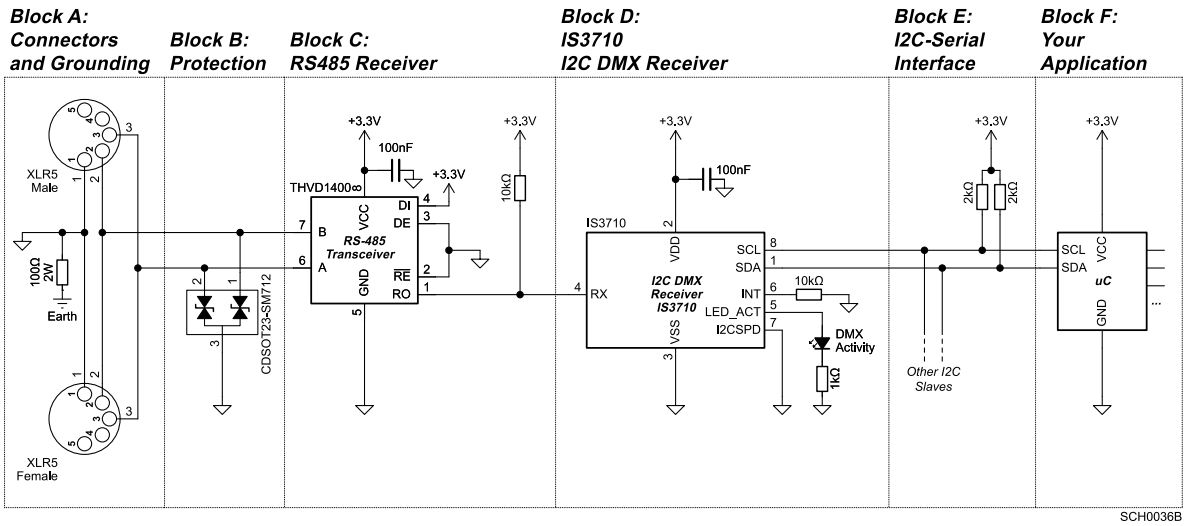
6. Hardware Examples

The following chapter represents an application design example for explanation proposals and is not part of the product standard. The customer must design his own solution, choose its most appropriate components and validate the final product according to the legislation and the Modbus specifications.

6.1. Non-Isolated Receiver

This example shows the design of a DMX Receiver using a non-isolated receiver.

The standard requires the manufacturers to label this type of receiver as **NON-ISO**.



Block A: Connector and Grounding

Connector

The official DMX connector is the XLR-5. Exceptions include RJ45, miniature connectors, and screw terminal connectors. However, despite its popularity and widespread use, XLR-3 is not part of the DMX standard and should not be used.

XLR-5 connectors are typically used in professional equipment, while XLR-3 connectors are more common in cost-sensitive devices. XLR-3 is generally cheaper than XLR-5.

Using an XLR-3 connector has the drawback of making your product compatible with standard microphone cables, which are specifically designed for low-frequency analog audio—not digital signals. As a result, microphone cables are not suitable for DMX.

DMX products use two connectors in daisy-chain configuration.

- Pin 1: Singal-Common
- Pin 2: Pair A Data -
- Pin 3: Pair A Data +
- Pin 4: Pair B not used
- Pin 5: Pair B not used

Cable

The DMX cable screen must be connected to XLR-5 pin 1 and not to its shell.

Use only twister pair cable to carry the DMX signal.

Grounding-Earthing

Do not connect the pin 1 (Singal-Common) of the XLR-5 connector directly to the earth (the mains earth), as this can create dangerous current loops. Signal-Common should be connected to earth through a 100Ω resistor to limit potential current flow through the DMX cable. This is especially important in large installations.

Block B: Protection

The protection stage is influenced by several factors, including the intrinsic robustness and protection features of the transceiver or receiver chip, the product's budget, and its required reliability, among other considerations. Refer to your transceiver's documentation to determine the appropriate protection requirements.

In the schematic, a bidirectional 400-W transient suppressor diodes (CDSOT23-SM712) are used to protect against surge transients.

Block C: Receiver

DMX operates over the RS485 electrical standard. Therefore, an RS485 transceiver or receiver is required to convert the differential RS485 signals to TTL-compatible voltage levels before entering the IS3710.

5 V transceivers or receivers can be used with the IS3710, as its RX pin is 5 V tolerant.

Since a DMX receiver never transmits data, the DE and RE pins of the transceiver can be tied to GND to keep it permanently in receiver mode.

Block D: IS3710

The IS3710 is very simple to integrate into your design.

A decoupling capacitor should be placed on the power pins (VDD and VSS). It is recommended to use a 100 nF, 10-25 V low-ESR ceramic capacitor.

The I2CSPD pin defines the I2C speed. Connect this pin to GND for a speed of 100 kHz. For 400 kHz, it should be pulled to 1.65 V, which is half of 3.3 V. This can be achieved with a simple resistor voltage divider using 3.3 V and GND. For 1 MHz, the pin must be connected to 3.3 V. This pin is not 5 V tolerant.

Block E: I2C-Serial Interface

For proper operation of the I2C Serial Interface, pull-up resistors to 3.3 V or 5 V are necessary. Typical resistor values are 4.7 kΩ for Standard Mode (100 kHz) and 2 kΩ for both Fast Mode (400 kHz) and Fast Mode Plus (1 MHz).

Block F: Your Application

Here is the rest of your product design. Typically, a microcontroller interfaces with the IS3710, but a microprocessor or a single-board computer, such as a Raspberry Pi, can also be used as long as they are equipped with an I2C Serial Interface.

7. Firmware Examples

7.1. STM32 Example

This example (ISXMPL3710ex2) demonstrates how to use the IS3710 I2C DMX Receiver chip with a STM32 microcontroller using the HAL I2C functions.

For clarity and brevity, all extra HAL definitions have been removed, leaving only the code related with the IS3710.

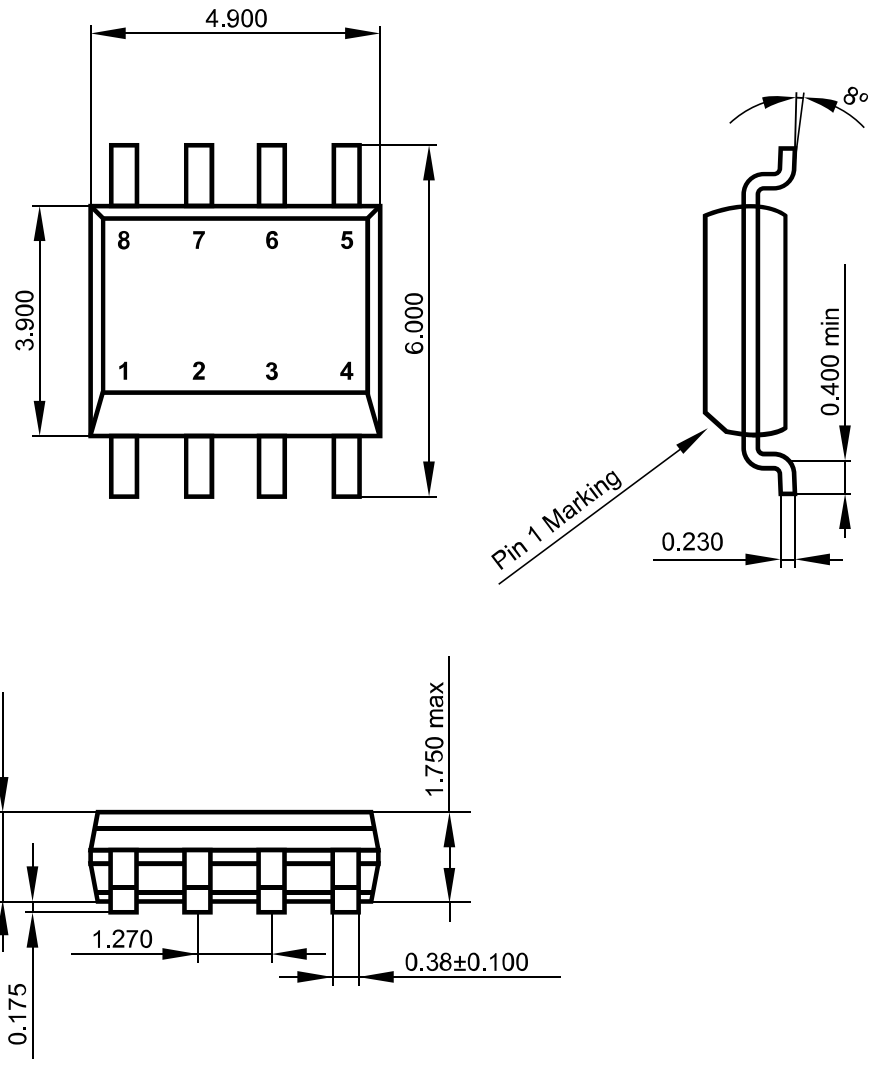
You can find the complete example at: www.inacks.com

You can get the IS3710 evaluation board (Kappa3710Ard) compatible with STM32 Nucleo boards at: http://www.inacks.com

```
/*
* This is the I2C address of the IS3710.
* HAL expects the address in 8-bit format, so we shift it left by 1.
*/
#define IS3710_I2C_ADDR       0x10 << 1

/*
* I2C memory address from which to start reading.
* This corresponds to the DMX channel we want to start reading from.
* The first DMX channel starts at address 1. Therefore, we start at
* address 0 so that Channel 1 aligns with index 1 in the array, and so on.
*/
#define STARTING_ADDRESS      0

/*
* Number of memory registers to read.
* This is the same as how many DMX channels we want to read + 1.
* Since we start reading from address 0 to match array indices
* with DMX channel numbers, we need to read 1 + 512 registers.
*/
#define HOW_MANY_REGISTERS    513

/*
* Timeout for the I2C read operation, in milliseconds.
*/
#define TIMEOUT 1000

uint8_t dmxChannel1, dmxChannel2, dmxChannel3;
uint8_t dmxBuffer[520]; // In this array we'll store the 512 DMX Channels

HAL_StatusTypeDef i2cReadOperationResult;

/*
* xecute the I2C read operation starting at memory address 0,
* reading a total of 513 registers.
* This ensures that DMX Channel 1 is stored at index 1 in the array, and so on.
*/
i2cReadOperationResult = HAL I2C Mem Read(&hi2c1, IS3710 I2C ADDR, STARTING_ADDRESS,
I2C_MEMADD_SIZE_16BIT, dmxBuffer, HOW_MANY_REGISTERS, TIMEOUT);

// If the I2C peripheral encounters an error, reset it
if (i2cReadOperationResult != HAL_OK) {
  HAL_Delay(50);
  HAL_I2C_DeInit(&hi2c1);
  MX_I2C1_Init();
}


// Let's get the DMX Channels 1, 2 and 3 from the buffer
dmxChannel1 = dmxBuffer[1];
dmxChannel2 = dmxBuffer[2];
dmxChannel3 = dmxBuffer[3];
```

8. Mechanical

SO8N Package

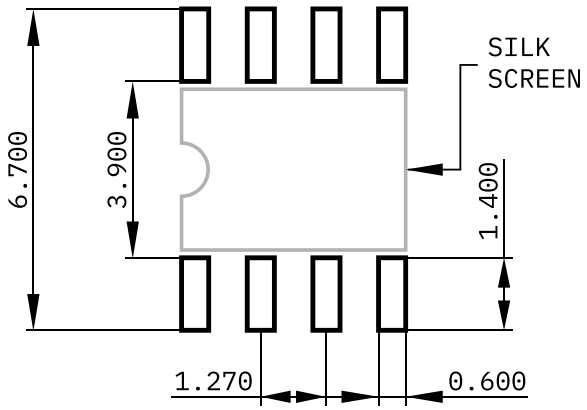


Units: millimeters

Notes:
This drawing is for general information only.
Drawing not to scale.

Evadrw0033A

SO8N Recommended Footprint



Units: millimeters

Notes:
This drawing is for general information only.
Drawing not to scale.

Evadrw0025r2

Content

Appendix

Revision History

Document Revision

Date	Revision Code	Description
July 2025	ISDOC133**A**	- Initial Release.

Chip Revision

Chip Revision can be found in the `CHIP_REV` register of the memory map.

Date	Revision Code	Description
July 2025	0	- Initial Release.

Documentation Feedback

Feedback and error reporting on this document are very much appreciated.

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

AThis 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 chip 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.