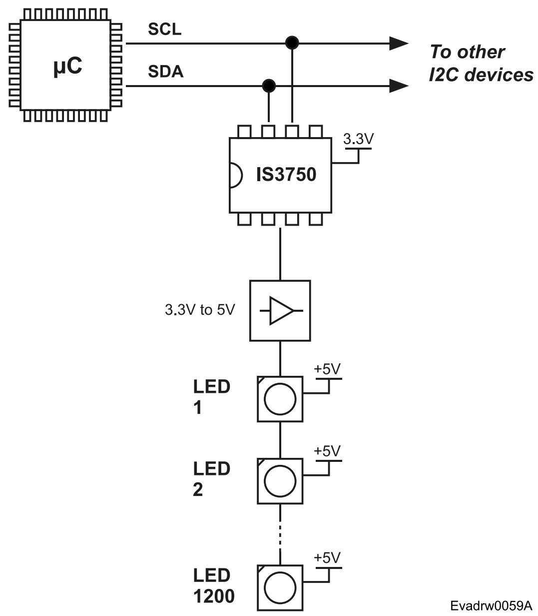
IS3750: I2C Addressable LED Controller

Control up to 1200 RGB LEDs

General Description

The IS3750 handles Addressable LED signal generation autonomously, requiring no CPU intervention. It is capable of controlling up to 1200 LEDs, with a dedicated RAM of 3600 registers—one for each color of each LED.

Your microcontroller writes the LED data to the IS3750's internal memory map via I2C, and the IS3750 will render and stream the data.



Note: This schematic is intended for illustration purposes only.

Since the IS3750 handles all the time-critical tasks of data rendering and streaming, your microcontroller can send data without being constrained by the timing requirements of the addressable LED protocol. This allows for more flexible data transmission over I2C, enabling the IS3750 to keep the LEDs perfectly synchronized.

The usage of the IS3750 also reduces CPU interruptions, peripheral usage (such as timers), and both flash and RAM consumption. Additionally, it eliminates the need for dedicated pins, as it operates over a shared I2C bus.

The IS3750 saves engineering time and associated costs by handling the communication protocol, providing a reliable solution that reduces the time-to-market (TTM) for your product.

IS3750 Main Advantages

- LED Agnostic: compatible with any color sequence and color quantity: GRB, RGB, GRBW, etc.
- Offloads your microcontroller's processing load by independently handling the Addressable LED protocol.
- Reduces firmware development time—no need to implement the protocol yourself.
- Minimizes Flash and RAM footprint in your microcontroller.
- The use of I2C eliminates the need for dedicated pins.
- 3,600 bytes of RAM: drive 1200 3-color LEDs or 900 4-color LEDs.

Applications

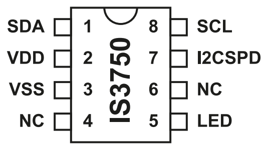
- Custom lighting
- LED signaling
- Color-coded process indicators
- Architectural lighting
- Status bars or meters
- Interactive buttons or sliders with visual feedback

Compatible LEDs

- WS2811
- WS2812 / WS2812B / WS2812C
- WS2813
- WS2815
- NeoPixel
- SK6812
- GS8208
- Works with any LED using the compatible LED protocol

Pinout

Part Number	Package	Op. Temperature
IS3750-S8-I	SO8N	-40ºC to +85ºC
IS3750-S8-E	SO8N	-40ºC to +125ºC



Product Selection Guide

	Part Number		Form Factor	Stack	Description
Only Stack	IS3750-S8		SO8N	Addressable LED WS2811, WS2812, WS2812B, WS2812C, WS2813, WS2815, and compatible protocol LEDs.	Addressable LED Controller Stack Chip. [Vist Product Page]
Evaluation Boards	Kappa3750Ard		Arduino Compatible		Evaluation board for the IS3750 with Arduino form factor. It features the IS3750 mounted on a PCB compatible with Arduino and other commercial microcontroller boards, such as the STMicroelectronics Nucleo. The board includes a series of LEDs, allowing you to easily test the IS3750 without any need for soldering. [Visit Product Page]
Evaluation Boards	Kappa3750Rasp	-	Raspberry Pi Compatible		Evaluation board for the IS3750 with Raspberry Pi form factor. It features the IS3750 mounted on a PCB compatible with Raspberry Pi and other commercial embedded computer boards. The board includes a series of LEDs, allowing you to easily test the IS3750 without any need for soldering. [Visit Product Page]

1. Specifications

Absolute Maximum Ratings

Parameter		Symbol	Min	Nom	Max	Unit
Supply Voltage		V_{DD}	-0.3	-	4	V
Input Voltage at any I/O pin		$V_{I/O-IN}$	-0.3	-	5.5	
Current Consumption	T_A = 25ºC	I_{DD}	-	3.05	3.60	mA
	T_A = 85ºC		-	3.15	3.80	
	T_A = 125ºC		-	3.35	4.16	
Temperature	Operating Temperature	T_{OP}	-40	-	+85	ºC
	Storage Temperature	T_{STO}	-65	-	+150	
Electrostatic Discharge (T_A = 25ºC)	Human-body model (HBM)	$V_{ESD-HBM}$	-2000	-	+1500	V
	Charged-device model (CDM)	$V_{ESD-CDM}$	-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 any I/O pin	$V_{I/O-IN}$	-0.3	-	5.5	
Source/Sink Current at any I/O Pin	$I_{I/O-SS}$	-	-	20	mA

Electrical Characteristics

Parameter		Symbol	Min	Nom	Max	Unit
Standby Current Consumption (T_A = 25ºC)		I_{SB}	-	-	2.8	mA
Operating Current Consumption (T_A = 25ºC)		I_{OP}	-	-	3.15	mA
Input Voltage	Logical High-Level	V_{IH}	0.7xV_{DD}	-	-	V
	Logical Low-Level	V_{IL}	-	-	0.3xV_{DD}	
Input Hysteresis		V_{HYST}	-	0.2	-	

LED Pin Timings

Parameter		Symbol	Min	Nom	Max	Unit
Reset (Low-Level Time)		T_R	338	-	-	ns
T0H		T_{0H}	360	-	380	
T1H		T_{1H}	800	-	820	
T0L		T_{0L}	860	-	880	
T1L		T_{1L}	430	-	450	
T0H+T0L		T_0	1.22	1.25	1.26	µs
T1H+T1L		T_1	1.23	1.25	1.27	
Total Frame Time	((1200 LEDs × 24 bits × 1.25 µs) + 0.338 µs	T_{TF}	36	-	-	ms
Frame Refresh Rate	((1200 LEDs × 24 bits × 1.25 µs) + 0.338 µs)$^{-1}$			27.77		Hz

The above timings represent the measurements made on the “LED Pin” of the IS3750.

2. Detailed Description

2.1. Description

The IS3750 is an Addressable LED Controller accessed via I2C. It can control from a single LED up to a strip of 1,200 RGB LEDs. It is designed to offload the LED data timing and transmission tasks from microcontrollers and FPGAs. The chip is available in two temperature ranges: Industrial (-40ºC to +85ºC) and Extended (-40ºC to +125ºC).

The IS3750 consists of three modules: the I2C-Serial Interface, the Memory Map, and the Render.

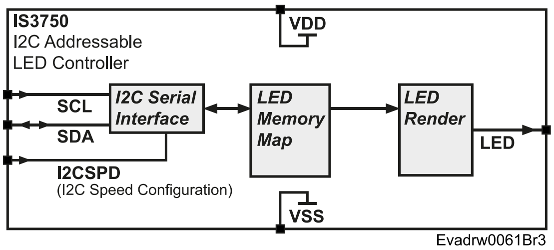


Figure 1: IS3750 Internal Block Diagram

I2C-Serial Interface Module

Data is sent via I2C from the user application (microcontroller, FPGA, single-board computer, or any I2C master) to the IS3750's internal memory map.

The IS3750 acts as an I2C slave, eliminating the need for a dedicated pin on the microcontroller, since it uses a shared bus.

The I2C interface supports Standard Mode (100kHz), Fast Mode (400kHz), and Fast Mode Plus (1MHz). A dedicated pin (I2CSPD) configures the appropriate internal filters for the selected speed.

The chip operates at a 3.3 V. Its I2C pins are 5 V tolerant, making it compatible with both 3.3 V and 5 V microcontrollers.

LED Memory Map Module

Each `LEDx` memory register represents the brightness of one color of a LED. The chip includes 3,600 registers, allowing control of up to 1,200 RGB addressable LEDs:

3,600 registers ÷ 3 colors = 1,200 LEDs.

LEDs with more than three colors or with non-standard color orders can also be controlled.

LED Render Module

Writing a `1` to a special register called `SHOW`, activates the LED Render module. This triggers a read of all `LEDx` registers and generates the corresponding output signal on the LED pin.

The LED pin operates at 3.3 V. When interfacing with 5 V LEDs, a buffer, Schmitt trigger, or level shifter is required to adapt it’s 3.3 V to 5 V. Refer to chapter “Hardware Example” for more details.

2.2. How it works

The IS3750's internal memory map consists of a single page containing 3,601 registers, with addresses ranging from `0` to `3600`. These registers support both individual and block read/write operations. Each register is 8 bits wide and implemented as volatile RAM.

You can think of the IS3750 as an I2C memory device—this analogy helps in understanding its memory behavior.

There are two types of memory registers: the `SHOW` register (address `0`), and the `LEDx` registers (address `1` to `3600`).

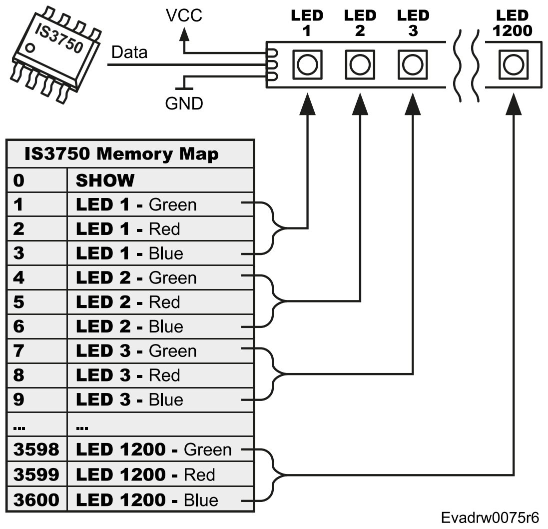


Figure 2: Mapping Between Memory Registers and GRB LEDs

The `SHOW` register triggers the action of rendering and generating all the data on the LED pin. Writing a `1` to this register indicates the IS3750 the beginning of this operation. This byte is automatically cleared to `0` once the operation has started.

The `LEDx` registers contain the data that will be sent to the LEDs; therefore, they represent the brightness of each LED color. A value of 0 means the LED color is off, while a value of 255 sets the LED to full brightness.

`LEDx` register data is only applied to the LEDs when the `SHOW` register is triggered (by writing a `1`).

All the `LEDx` registers are sent consecutively from address `1` to address `3600` via the LED pin, encoded in NZR Addressable LED protocol.

2.3. LED Agnostic

The IS3750 is agnostic to the LED model—it does not interpret or enforce a specific color sequence (e.g., RGB, GRB) or the number of colors per LED. The controller simply transmits all data stored in the LEDx registers in sequence, without interpretation, only generating the NZR encoding.

If you are using 3-color LEDs (e.g., RGB or GRB), each LED consumes 3 bytes of data. In this case, the IS3750 can control up to 1,200 LEDs (3,600 registers ÷ 3 bytes per LED).

For 4-color LEDs (e.g., RGBW), each LED requires 4 bytes, allowing control of up to 900 LEDs (3,600 ÷ 4).

Note: The most common LED configuration on the market is a 3-channel GRB sequence.

Color Sequence Agnostic Example

Suppose you write the value 255 to address 1 and then trigger the SHOW register.

If you are using a GRB LED strip, the first LED will display green, since the first byte corresponds to the green channel.

If you are using an RGB LED strip, the same byte will result in red, since it maps to the red channel.

Note: GRB is the most common color sequence in popular addressable LEDs such as the WS2812B family.

Color Count Agnostic Example

Suppose you write the value 255 to address 4 and then trigger the SHOW register.

With a GRB LED strip, the second LED will light up green (since its first byte is at address 4).

With a GRBW LED strip, the first LED will show white, as address 4 corresponds to its fourth channel.

Note: Most addressable LEDs in the market are 3-channel (RGB or GRB).

The LED-agnostic design of the IS3750 makes it versatile and easy to use. Furthermore, it allows you to use different types of addressable LED with minor firmware review on your firmware.

While LED-agnostic, the IS3750 is specifically designed to control single-wire addressable LEDs using NZR coding.

Two-wire (SPI-like) LEDs—such as APA102, SK9822, LPD8806, P9813—are not supported by the IS3750. These LEDs can instead be controlled using the IS3755.

In environments with electrical noise that may cause LED glitches, it is recommended to periodically refresh the LEDs to clear any potential display artifacts caused by noise on the LED power or data pins.

2.4. Advantages

The use of INACKS integrated silicon stack chips offers the advantage of making communication protocols transparent to the engineer: no dedicated libraries are required, and no knowledge of protocol implementation is needed.

This results in significant time savings during the engineering stage, as engineers do not need to spend time understanding, implementing and testing the communication protocol.

These key time savings help lower engineering costs and accelerate the development of a minimum viable product or prototype, ultimately leading to a faster time-to-market.

The use of addressable LEDs requires strict adherence to microsecond and nanosecond-level timing, making software-based implementations challenging and consuming microcontroller resources such as timers, SPI, or PWM, while also keeping the CPU busy handling interrupts. Using an external dedicated chip offloads these tasks, freeing up system resources and significantly reducing CPU load—allowing the microcontroller to send new LED data in a much more relaxed and efficient manner.

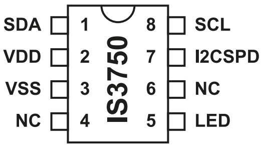
Besides the internal resources required by a microcontroller for the addressable LED protocol implementation, an external dedicated pin is also needed. However, with the IS3750 controller chip, this constraint is eliminated, as the chip communicates via I2C. Since I2C is a multi-device communication protocol, no microcontroller pins are exclusively sacrificed for LED control. The IS3750 supports I2C speeds of 100kHz, 400kHz, and 1MHz, enabling good LED refresh rates even over I2C.

Another benefit of offloading addressable LED control to the IS3750 is the significant memory savings. Each addressable LED requires 24 bits (8 bits per color), meaning that controlling 1,000 LEDs would require 3,000 bytes of memory. Since the IS3750 features an internal memory of 3,600 bytes, it can drive up to 1200 LEDs without requiring the microcontroller to allocate that memory.

This series of resource savings (physical pin, internal peripherals such as timers, SPI or PWM, interruptions, and memory) allows the selection of a microcontroller with fewer features.

Additionally, the IS3750 features an easy-to-solder SO8N package with a 1.27 mm pin pitch, making it ideal for both oven and hand soldering while reducing the risk of pin short circuits during the soldering process.

3. Pin Description



Pin	Name	Type	Description
1	SDA	Open Drain 5V Tolerant	I2C-compatible Data pin. Open drain, it requires pull-up.
2	VDD	Supply	3.3 V power supply pin. Bypass this pin to GND with a 100 nF ceramic capacitor.
3	VSS	Ground	Ground reference pin.
4, 6	NC	No Connection	These pins are unused and can be left floating. However, it is recommended to connect them to GND for improved noise immunity.
5	LED	Digital Output Push-Pull	3.3 V addressable LED data pin. The logic level on this pin must be shifted to 5 V to properly drive the LED's data input.
7	I2CSPD	Analog Input 0 to 3.3V	I2C-Serial Interface Speed Selection pin. • For 100 kHz pull to GND. • For 400 kHz make a voltage divider of VDD/2 (1.65 V). • For 1 MHz pull to VDD (3.3 V). Attention: Voltage above 4 V will damage the device.
8	SCL	Open Drain 5V Tolerant	I2C-compatible Clock pin. Open drain, it requires pull-up.

LED

3.3 V Addressable LED Data Pin.

The Addressable LED Data Pin operates at 3.3 V. However, addressable LEDs typically require a 5 V logic level, so a non-inverting buffer is needed to shift the signal from 3.3 V to 5 V. There are many suitable buffers available; in the implementation guide chapter, the 74LVC1G17 is used.

VDD

3.3 V Power Supply Pin.

A decoupling capacitor should be placed on the power pin. It is recommended to use a 100nF, 10-25V low-ESR ceramic capacitor.

SCL and SDA Pins

I2C-Compatible Bus Interface Pins.

SCL (Serial Clock Line): This pin is used to synchronize data transfer between the IS3750 device and the Microcontroller or other CPU.

SDA (Serial Data Line): This bidirectional pin is used for both sending and receiving data between the IS3750 and the Microcontroller or other CPU.

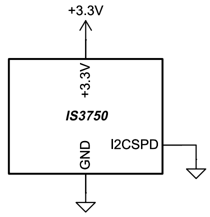
Both pins are open-drain and must be pulled up to 3.3V or 5V. The pull-up resistor value should be chosen based on the bus speed and capacitance. Typical values are 4.7kΩ for Standard Mode (100kbps) and 2.2kΩ for Fast Mode (400kbps) at both 3.3V and 5V.

I2CSPD Pin

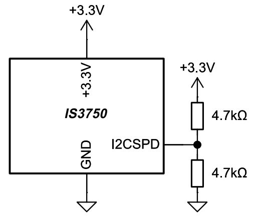
I2C-Serial Interface Speed Selection Pin.

This pin configures the IS3750 internal I2C-Serial Interface timings and filters to properly work with the selected bus speed.

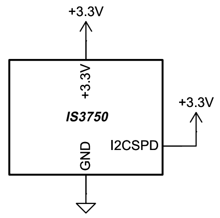
- For a **100kHz** setting, set the I2CSPD pin to VSS.



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



- For a **1000MHz** setting, set the I2CSPD pin to VDD.



Important Remark:
A mismatch between the configured I2C speed and the actual operating I2C speed (e.g., configuring the bus for 100kHz but operating at 1MHz) 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

I2C Register Address	Register Name	Register Description
0 (0x0000)	**SHOW**	Send data to the LEDs
1 (0x0001)	**LED1G**	LED 1, color Green brightness
2 (0x0002)	**LED1R**	LED 1, color Red brightness
3 (0x0003)	**LED1B**	LED 1, color Blue brightness
4 (0x0004)	**LED2G**	LED 2, color Green brightness
5 (0x0005)	**LED2R**	LED 2, color Red brightness
6 (0x0006)	**LED2B**	LED 2, color Blue brightness
7 (0x0007)	**LED3G**	LED 3, color Green brightness
8 (0x0008)	**LED3R**	LED 3, color Red brightness
9 (0x0009)	**LED3B**	LED 3, color Blue brightness
…	(LED4 to LED1197)	…
3592 (0x0E08)	**LED1198G**	LED 1198, color Green brightness
3593 (0x0E09)	**LED1198R**	LED 1198, color Red brightness
3594 (0x0E0A)	**LED1198B**	LED 1198, color Blue brightness
3595 (0x0E0B)	**LED1199G**	LED 1199, color Green brightness
3596 (0x0E0C)	**LED1199R**	LED 1199, color Red brightness
3597 (0x0E0D)	**LED1199B**	LED 1199, color Blue brightness
3598 (0x0E0E)	**LED1200G**	LED 1200, color Green brightness
3599 (0x0E0F)	**LED1200R**	LED 1200, color Red brightness
3600 (0x0E10)	**LED1200B**	LED 1200, color Blue brightness

Figure 3: IS3750 Memory Map

4.1. SHOW Register

Writing a `1` to this register triggers the rendering of all memory map LEDx data and sends it through the LED pin. The register is automatically reset to 0. Writing a 0 makes no effect.

At power-up, the default value for this register is 0, meaning all LEDs remain off.

Name: SHOW
Description: Send data to the LEDs.
Address: `0` (`0x0000`)
Memory Type: Volatile RAM
Allowed values: `0` or `1` (`0x00` to `0x01`)
Reset value: `0` (`0x00`)

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

4.2. LEDx Register

LEDx registers store the color brightness values that will be sent to the LEDs.

The value stored in each register is what will be directly sent to the LED. A value of 0 means the corresponding color is off, while a value of 255 means the color is at maximum brightness.

These LED registers are volatile RAM: you can modify them individually or in blocks. When the IS3750 loses power, the contents of the registers are erased. At power-up, the default value for all registers is 0.

The registers can be both written to and read from. If needed, they can also be used as an extension of your microcontroller's RAM, helping to reduce memory usage in your project regarding to LED managing.

You can write or read these registers individually, or perform block reads/writes for up to 3,601 registers in a single I2C operation.

Writing to `LEDx` registers does not have any effect on the LEDs until the `SHOW` register is triggered.

Name: LEDx
Description: Brightness register of LEDx color [red, green or blue].
Address Range: `0` to `3599` (`0x0000` to `0x0E0F`)
Memory Type: Volatile RAM
Allowed values: `0` to `255` (`0x00` to `0xFF`)
Reset value: `0` (`0x00`)

Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
LEDx[R, G, B] [7 to 0]							

5. I2C-compatible Bus Description

The IS3750 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 single board computer, initiates and manages all read and write operations to the Slave.

The IS3750 is represented on the bus by the I2C device address: `18 (0x12)`.

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 IS3750's I2C 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 operability of the Read and Write commands of the IS3750 is very similar to an EEPROM memory. Thinking of the IS3750 as an EEPROM memory is a good analogy to quickly understand how to communicate with the device.

5.1. Highlights

- **I2C Device Address**: `18 (0x12)`

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

- **Supported Operations**:
  - Single-Byte Write
  - Multiple-Byte Write (up to 3,601 registers)
  - Single-Byte Read
  - Multiple-Byte Read (up to 3,601 registers)

- **Other**:
  - If a write operation starts at a valid memory address (`0` to `3599`) and continues past the last valid address, it will roll over to address `0`.
  - Starting a write operation to an invalid memory address (greater than `3599`) will result in a NACK and data will be discarded.
  - If a read operation starts at a valid memory address (`0` to `3599`) 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 3599) will return a value of 0xFF.

5.2. Write Operations

5.2.1. Single Byte Write

Writing a single byte is an action performed by the microcontroller (I2C-Master) to write data to any register within the IS3750 memory (I2C-Slave), regardless of the last read or written position. To perform this action, the microcontroller must load the register address intended to be written into the IS3750's Pointer Register. Once the address is set, the Microcontroller can send the data to be written.

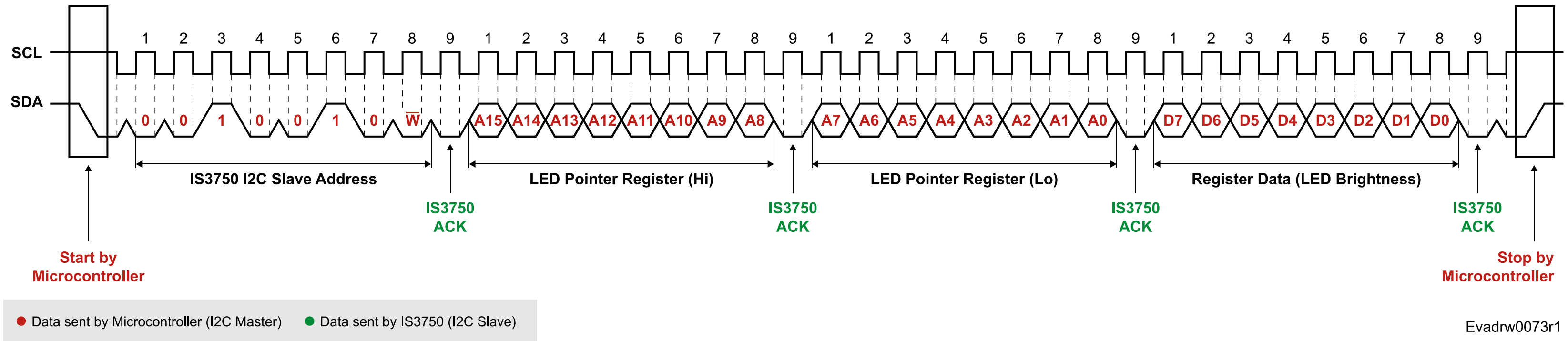
To initiate the Single Byte Write operation, the following steps must be performed from the beginning: The microcontroller begins by pulling down the SDA line while the SCL line is high, creating a Start Condition. It then sends the IS3750 I2C device address (`0x12`) with the R/W bit set to '0' (indicating a write operation). Note that the IS3750's I2C address is fixed and does not change, allowing it to be uniquely identified among other devices on the I2C serial interface.

Upon receiving the device address, the IS3750 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 IS3750.

The microcontroller then sends the byte to be written, which the IS3750 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 IS3750 goes from addresses `0` to `3599 (0x0E0F)`. If a Write Operation is performed with a Pointer Register higher than `3599 (0x0E0F)`, the IS3750 will answer with a NACK.



5.2.2. Multiple Byte Write

The Multiple Byte Write operation functions similarly to the Single Byte Write, but allows writing a block of up to 3,601 registers in a single operation—that is, the entire memory in one go.

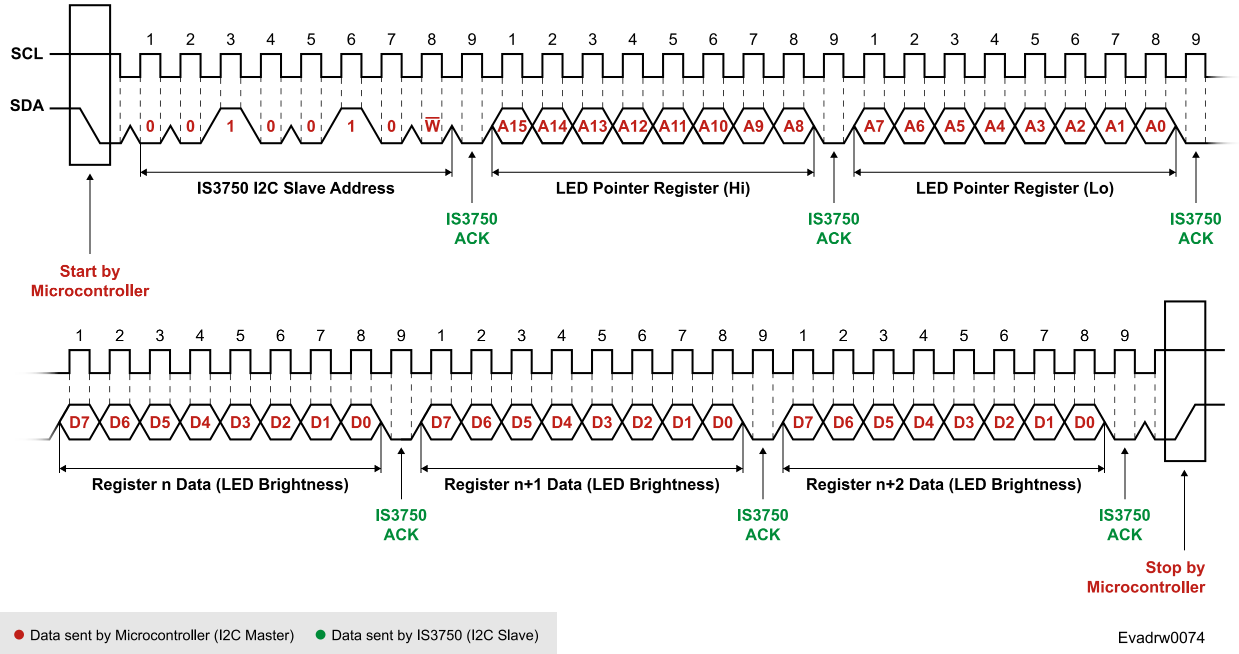
To perform a Multiple Byte Write operation, follow the same procedure as for a Single Byte Write until the first data byte is written. After writing the first byte, instead of generating a Stop Condition, the microcontroller should continue writing data bytes. To conclude the write operation, after sending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

Invalid Memory Addressing

The valid memory range of the IS3750 goes from addresses `0` to `3599 (0x0E0F)`.

If a Multiple Byte Write Operation is performed with a Pointer Register within the valid memory range (`0` to `3599`) but exceeds the last memory register (`3599`), a rollover to register `0` will occur.

If a Multiple Byte Write Operation is performed with a Pointer Register outside from the valid memory range (greater than `3599`), the IS3750 will respond with a NACK upon receiving the first data byte.



5.3. Read Operations

5.3.1. Single Byte Read

Reading a single byte is an action performed by your microcontroller (I2C-Master) to access any register within the IS3750 memory (I2C-Slave), regardless of the last read or written position. To perform this action, your microcontroller must load the register address intended to be read into the IS3750's Pointer Register. Once the address is set, the microcontroller can retrieve the 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 IS3750 I2C device address (`0x12`) with the R/W bit set to '0' (indicating a write operation). Note that the IS3750's I2C address is fixed and does not change, allowing it to be uniquely identified among other devices on the I2C serial interface.

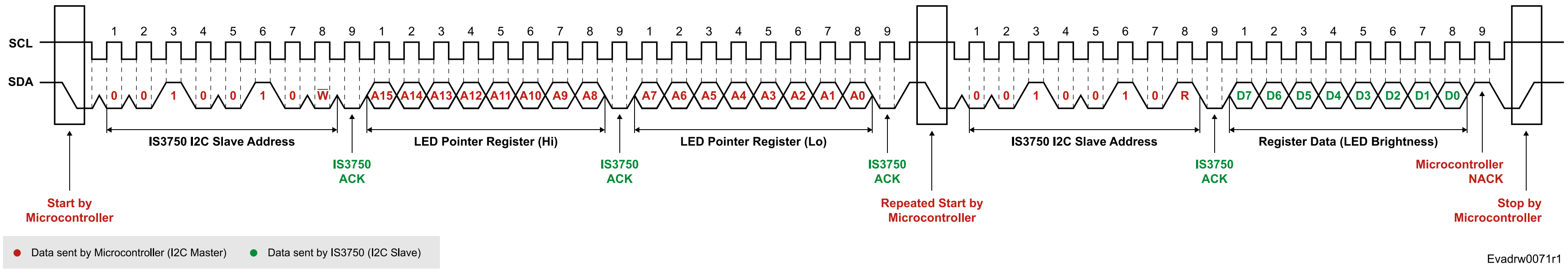
Upon receiving the device address, the IS3750 acknowledges it. Subsequently, the microcontroller sends the two bytes of the register address it intends to read: the most significant byte first, followed by the less significant byte, each acknowledged by the IS3750.

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

The microcontroller generates a Repeated Start Condition, followed by the IS3750 I2C device address (`0x12`) with the R/W bit set to '1' (indicating a read operation), instructing the IS3750 to retrieve data. The IS3750 acknowledges and responds with the register color, 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 IS3750 goes from addresses `0` to `3599` (`0x0E0F`). If a Read Operation is performed with a Pointer Register higher than `3599`, the read result will be `0xFF`.



5.3.2. Multiple Byte Read

The Multiple Byte Read operation functions similarly to the Single Byte Read, but allows reading a block of up to 3,601 registers in a single operation—that is, the entire memory in one go.

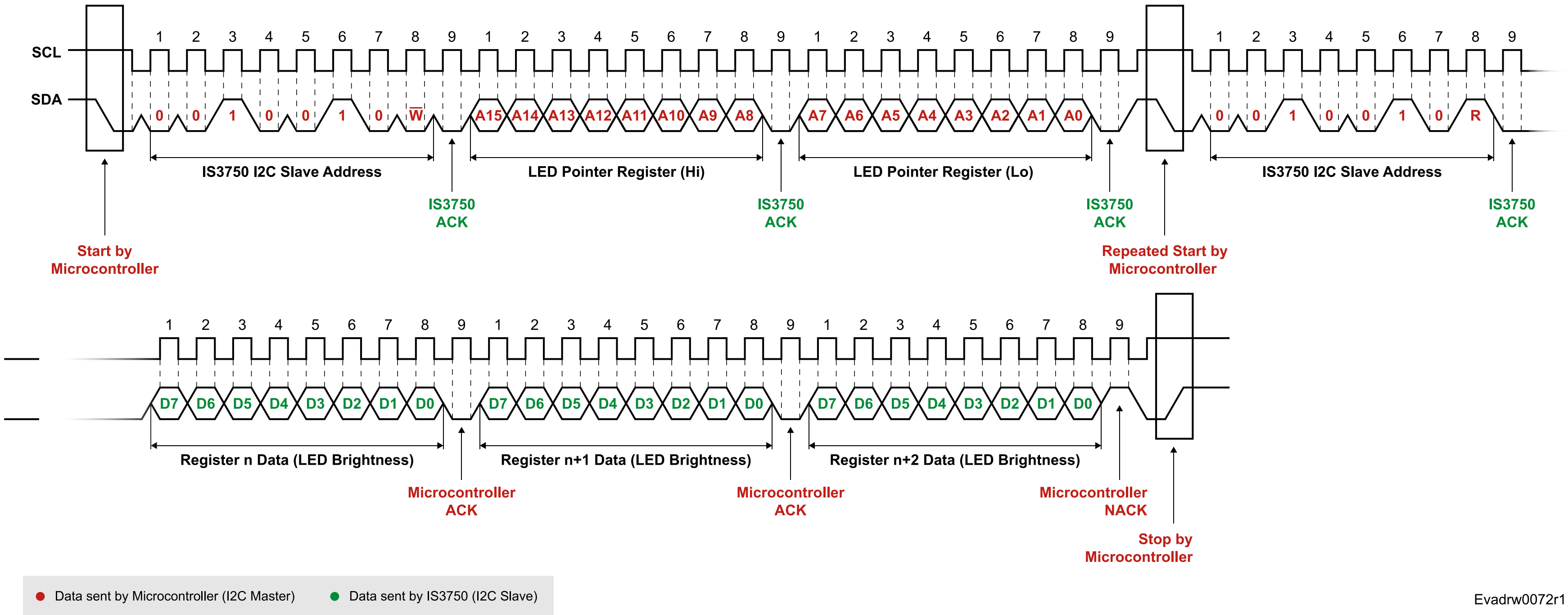
To perform a Multiple Byte Read operation, follow the same procedure as for a Single Byte Read until the first byte is received. After receiving the first byte, instead of generating a Not Acknowledge (NACK), the microcontroller should continue acknowledging (ACK) each received data byte from the IS3750 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.

Invalid Memory Addressing

The valid memory range of the IS3750 goes from addresses `0` to `3599 (0x0E0F)`.

If the Multiple Byte Read Operation is performed with a Pointer Register within the valid memory range (`0` to `3599`), but the data retrieval extends beyond register `3599`, a rollover to register `0` will occur.

If a Multiple Byte Read Operation is performed with a Pointer Register value higher than `3599`, the read result will be `0xFF`.



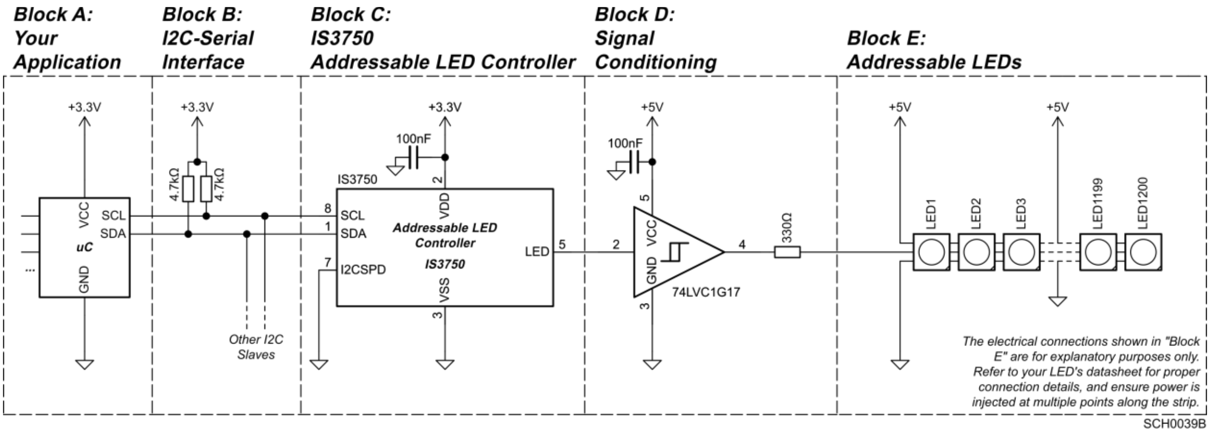
6. Implementation Guide

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. Hardware Example

This example shows how to use a microcontroller with the IS3750 to drive up to 1200 LEDs.

More examples can be found on the website.



Block A: Your Application

This is the core of your project. Typically, it will be a microcontroller, FPGA, or an embedded computer like a Raspberry Pi, among others.

It's the part of the system where you want to offload CPU load, reduce RAM and Flash usage, and eliminate the need for timers — along with the stress of continuously handling timer interrupts. In short, it's the part you want to keep as clean and simplified as possible.

Block B: I2C-Serial Interface

This is the bus where you connect all your I2C slave devices. It can operate at either 3.3V or 5V, as the IS3750 I2C pins are 5V tolerant.

The I2C-Serial Interface requires pull-up resistors on the SCL and SDA lines. Typical values are 4.7kΩ for Standard Mode (100kHz), and 2kΩ for Fast Mode (400kHz) and Fast Mode Plus (1MHz).

The IS3750 uses the I2C device address `18` by default. If you require a different address, please contact our customization department. Refer to section Appendix/Customization for more details.

Block C: IS3750 IC

The IS3750 IC operates at 3.3 V. Its I2C-Serial Interface pins are 5 V tolerant, allowing direct connection to a 5 V microcontroller if needed.

The LED output pin operates at 3.3 V. However, addressable LEDs typically require a 5 V logic level, so a non-inverting buffer is needed to shift the signal from 3.3 V to 5 V. There are many suitable buffers available; in this example, the 74LVC1G17 is used.

A decoupling capacitor should be placed on the power pin V_{DD}. It is recommended to use a 100nF, 10-25V low-ESR ceramic capacitor.

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

Block D: Signal Conditioning

The buffer, as explained in the previous paragraph, shifts the 3.3 V logic level to 5 V for proper LED operation.

A series resistor—typically between 330 Ω and 470 Ω—is recommended between the buffer and the first addressable LED. This helps reduce signal ringing by adding impedance to better match the trace or wire, thereby damping reflections.

Note: this is not a pull-up or pull-down resistor; it's a series resistor placed directly between the buffer output and the LED's data input.

Block E: Addressable LEDs

The IS3750 can control any number of LEDs in series, as long as the total number of data bytes does not exceed 3,600. This means it supports up to 1,200 3-color LEDs, 900 4-color LEDs, and so on. Typically, addressable LEDs are 3-color.

When using more than a few LEDs, special attention must be given to the power supply design. Poor PCB layout or cabling in the power domain can lead to burnt traces or wires, voltage drops, and incorrect LED brightness or spurious flickering.

Always calculate the total current draw of all LEDs to properly size the power supply. Do not assume that certain colors won't be used and therefore a smaller supply is sufficient. Always assume the worst case: all LED colors on at 100% brightness.

When powering LEDs from an external power supply, make sure the LEDs and the rest of the circuitry controlling the LEDs share the same voltage reference. The GND of the LEDs and the GND of your control circuit must be connected.

6.2. Firmware Example

6.2.1. STM32 Example

This example (ISXMPL3750ex2) demonstrates how to use the IS3750 Addressable LED Controller 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 IS3750.

You can find the complete example at: www.inacks.com/isxmpl3750ex2

You can get the IS3750 evaluation board (Kappa3750Ard) compatible with STM32 Nucleo boards at: www.inacks.com/kappa3750ard

```
#define IS3750_REGISTER_SHOW        0x00
#define IS3750_REGISTER_LED1_RED    0x01
#define IS3750_REGISTER_LED1_GREEN  0x02
#define IS3750_REGISTER_LED1_BLUE   0x03
#define IS3750_REGISTER_LED2_RED    0x04
#define IS3750_REGISTER_LED2_GREEN  0x05
#define IS3750_REGISTER_LED2_BLUE   0x06
#define IS3750_REGISTER_LED3_RED    0x07
#define IS3750_REGISTER_LED3_GREEN  0x08
#define IS3750_REGISTER_LED3_BLUE   0x09

// Sends brightness value to a specific register of the IS3750.
void writeLedRegister(uint16_t registerAddress, uint8_t bright) {
  uint8_t IS3750_I2C_Chip_Address = 0x12 << 1; // STM32 HAL expects 8-bit I2C address
  HAL_I2C_Mem_Write(&hi2c1, IS3750_I2C_Chip_Address, registerAddress, I2C_MEMADD_SIZE_16BIT,
&bright, 1, 1000);
}

// Triggers the IS3750 to update the LED outputs.
void showLEDs(void) {
  uint8_t IS3750_I2C_Chip_Address = 0x12 << 1;
  uint8_t dataToWrite[1] = {1}; // Command to show updated values
  HAL_I2C_Mem_Write(&hi2c1, IS3750_I2C_Chip_Address, IS3750_REGISTER_SHOW,
I2C_MEMADD_SIZE_16BIT, dataToWrite, 1, 1000);
}

// Sets all LED registers to 0 (turns off all LEDs).
void clearAllLedRegisters(void) {
  uint8_t IS3750_I2C_Chip_Address = 0x12 << 1;
  uint8_t dataToWrite[1200 * 3] = {0}; // 3600 zeroed bytes
  HAL_I2C_Mem_Write(&hi2c1, IS3750_I2C_Chip_Address, IS3750_REGISTER_LED1_RED,
I2C_MEMADD_SIZE_16BIT, dataToWrite, sizeof(dataToWrite), 1000);
}

int main(void)
{
  while (1)
  {
    // Show green on LED1
    clearAllLedRegisters();
    writeLedRegister(IS3750_REGISTER_LED1_GREEN, 5);
    showLEDs();
    HAL_Delay(500);

    // Show yellow on LED2 (Red + Green)
    clearAllLedRegisters();
    writeLedRegister(IS3750_REGISTER_LED2_RED, 5);
    writeLedRegister(IS3750_REGISTER_LED2_GREEN, 5);
    showLEDs();
    HAL_Delay(500);

    // Show blue on LED3
    clearAllLedRegisters();
    writeLedRegister(IS3750_REGISTER_LED3_BLUE, 5);
    showLEDs();
    HAL_Delay(500);
  }
}
```

6.2.2. Arduino Example

This example (ISXMPL3750ex1) demonstrates how to use the IS3750 Addressable LED Controller chip with an Arduino microcontroller board using the Arduino functions.

You can find the complete example at: www.inacks.com/isxmpl3750ex1

You can get the IS3750 evaluation board (Kappa3750Ard) compatible with Arduino UNO form factor boards at: www.inacks.com/kappa3750ard

```
#include <Wire.h>

// I2C device address of the IS3750 chip:
#define IS3750_I2C_ADDRESS    0x12

// Memory Map:
#define IS3750_REGISTER_SHOW        0x00
#define IS3750_REGISTER_LED1_RED    0x01
#define IS3750_REGISTER_LED1_GREEN  0x02
#define IS3750_REGISTER_LED1_BLUE   0x03
#define IS3750_REGISTER_LED2_RED    0x04
#define IS3750_REGISTER_LED2_GREEN  0x05
#define IS3750_REGISTER_LED2_BLUE   0x06
#define IS3750_REGISTER_LED3_RED    0x07
#define IS3750_REGISTER_LED3_GREEN  0x08
#define IS3750_REGISTER_LED3_BLUE   0x09


void writeLedRegister(uint16_t registerAddress, uint8_t bright) {
  // Start the I2C communications to the IS3750 chip.
  Wire.beginTransmission(IS3750_I2C_ADDRESS);

  // Send the 16-bit Holding Register address (2 bytes).
  Wire.write((registerAddress >> 8) & 0xFF); // High byte.
  Wire.write(registerAddress & 0xFF);        // Low byte.

  // Send the 8-bit data (the brightness).
  Wire.write(bright);

  // End the I2C communications.
  Wire.endTransmission();
}

// This routine updates the LEDs.
void showLeds(void) {
  // Write a '1' to the SHOW register (address 0x00)
  // to trigger rendering based on the current memory map contents.
  writeLedRegister(IS3750_REGISTER_SHOW, 1);
}

// This routine sets all the LED registers to 0.
void clearAllLedRegisters(void) {
  uint16_t i;

  // Write 0 to all LED control registers.
  for (i = 1; i <= 1200; i++) {
    writeLedRegister(i, 0);
  }
}

void setup() {
  Wire.begin(); // Initialize the I2C interface.
}

void loop() {

  // Let's do color green:
  clearAllLedRegisters(); // Clear all memory map.
  writeLedRegister(IS3750_REGISTER_LED1_GREEN, 5); // Set LED1 to green (brightness = 5)
  showLeds();
  delay(500);

  // Let's do color yellow:
  clearAllLedRegisters(); // Clear all memory map.
  // Set LED2 to yellow by combining red and green (brightness = 5 each)
  writeLedRegister(IS3750_REGISTER_LED2_RED, 5);
```

```
  writeLedRegister(IS3750_REGISTER_LED2_GREEN, 5);
  showLeds();
  delay(500);

  // Let's do color blue:
  clearAllLedRegisters();// Clear all memory map.
  writeLedRegister(IS3750_REGISTER_LED3_BLUE, 5); // Set LED3 to blue (brightness = 5)
  showLeds();
  delay(500);

}
```

6.2.3. Raspberry Pi Example

This example (ISXMPL3750ex3) demonstrates how to use the IS3750 Addressable LED Controller chip with an Arduino microcontroller board using the Arduino functions.

You can find the complete example at: www.inacks.com/isxmpl3750ex3

You can get the IS3750 evaluation board (Kappa3750Ard) compatible with Arduino UNO form factor boards at: www.inacks.com/kappa3750rasp

```
from smbus2 import SMBus, i2c_msg
import time

I2C_BUS = 1  # Use 1 for most Raspberry Pi models
DEVICE_ADDRESS = 0x12  # 7-bit I2C address of the IS3750

# IS3750 register map
REGISTER_SHOW = 0x00
REGISTER_LED1_RED = 0x01
REGISTER_LED1_GREEN = 0x02
REGISTER_LED1_BLUE = 0x03
REGISTER_LED2_RED = 0x04
REGISTER_LED2_GREEN = 0x05
REGISTER_LED2_BLUE = 0x06
REGISTER_LED3_RED = 0x07
REGISTER_LED3_GREEN = 0x08
REGISTER_LED3_BLUE = 0x09

def write_register(start_register, data_bytes):
    """
    Write a block of data starting at a 16-bit register address.

    :param start_register: The 16-bit register address to start writing to.
    :param data_bytes: A list of bytes to write.
    """
    high_addr = (start_register >> 8) & 0xFF
    low_addr = start_register & 0xFF
    with SMBus(I2C_BUS) as bus:
        msg = i2c_msg.write(DEVICE_ADDRESS, [high_addr, low_addr] + data_bytes)
        bus.i2c_rdwr(msg)

def show_leds():
    """Send the 'show' command to apply the LED updates."""
    write_register(REGISTER_SHOW, [1])

def clear_all_led_registers():
    """Clear all LED registers by sending 3600 zero bytes."""
    data = [0] * (1200 * 3)
    write_register(REGISTER_LED1_RED, data)

# Example usage loop
while True:
    clear_all_led_registers()
    write_register(REGISTER_LED1_GREEN, [5])
    show_leds()
    time.sleep(1)

    clear_all_led_registers()
    write_register(REGISTER_LED2_RED, [5])
    write_register(REGISTER_LED2_GREEN, [5])
    show_leds()
    time.sleep(1)

    clear_all_led_registers()
    write_register(REGISTER_LED3_RED, [5])
    show_leds()
    time.sleep(1)
```

6.3. Troubleshooting

Visual LED Problems

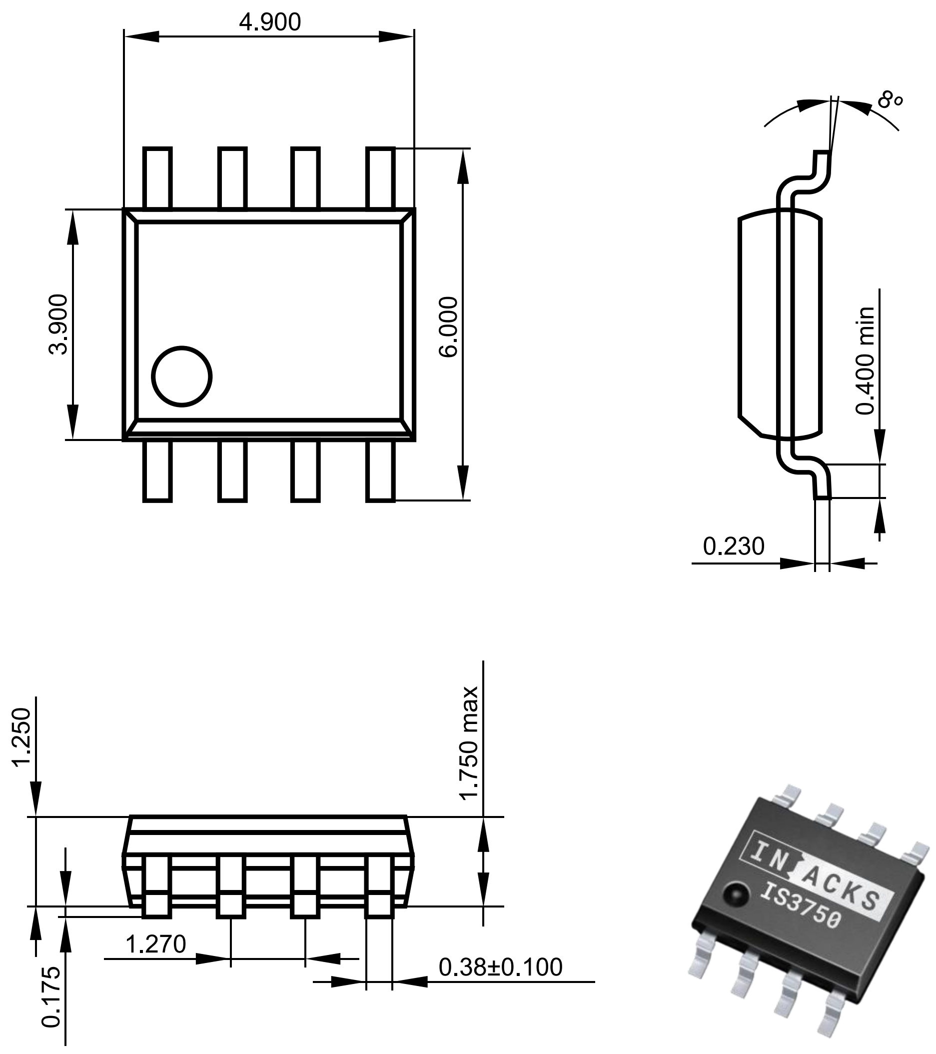
Addressable LEDs are susceptible to a wide variety of problems due to incorrect power supply design, electrical noise, and other factors. This troubleshooting section does not aim to cover every possible problem, but rather to address the most common ones.

Problems	**LEDs are displaying an orange-red color instead of the correct colors** **Wrong LED sequence or pattern**
Possible Causes and Solution #1	Your LEDs are drawing more current than your power supply can provide. Calculate the total current needed in the worst case (all LEDs at full brightness—red, green, and blue at 100%), and ensure your power supply can deliver at least that amount.
Possible Causes and Solution #2	You're losing voltage across the power wires due to their resistance and the high current draw. Use thicker wires/traces, and power the LED strip/PCB from multiple injection points. Run several positive and negative wires/traces from the power supply to different spots along the LED strip/PCB to minimize voltage drop.

Problem	**Random color changes or glitches in the LED pattern**
Possible Causes and Solution #1	The GND of the power supply used for the LEDs is not connected to the GND of the control electronics (PCB). If it is safe to do so, connect the grounds together. This ensures that both the power supply and the control circuit share a common voltage reference. Important: Always evaluate carefully before connecting different power domains to avoid ground loops or potential damage.

7. Mechanical

SO8N Package



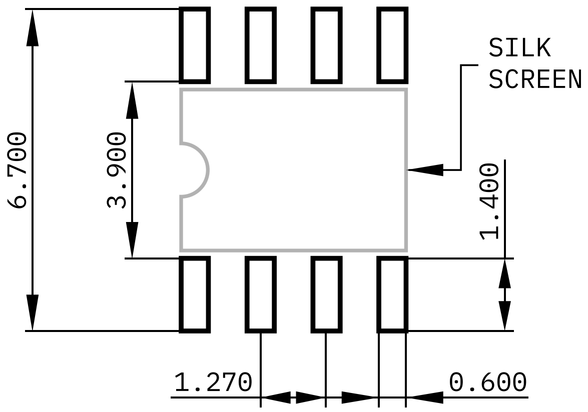
Units: millimeters

INTEGRATED SILICON STACKS

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

Evadrw0081r0

SO8N Recommended Footprint



Units: millimeters

INTEGRATED
SILICON STACKS

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

Evadrw0025r2

Content

Appendix

Revision History

Date	Revision Code	Description
June 2025	ISDOC132**B**	Color name correction in *Figure 3*, column *Description*
May 2025	ISDOC132**A**	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

All referenced brands, product names, service names and trademarks are 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.