512kb EEPROM - CAT25512 - Trēo™ Module

Module Features

- On Semiconductor CAT25512
- RoHS Compliant
- Software Library
- NightShade Trēo™ Compatible
- Breakout Headers

CAT25512 Features

(from On Semiconductor)

- 128-byte Page Write Buffer
- Additional Identification Page with Permanent Write Protection
- Self-timed Write Cycle
- Hardware and Software Protection
- Block Write Protection
    - Protect ¼, ½, or Full EEPROM Array
- Low Power CMOS Technology
- 4,000,000 Program/Erase Cycles
- 200 Year Data Retention

Applications

- Non-Volatile Data Storage
- Setting Retention
- Authentication

Trēo™ Compatibility

Electrical

Communication	SPI
Max Current, 3.3V	2mA
Max Current, 5V	0mA

Mechanical

- 25mm x 25mm Outline
- 20mm x 20mm Hole Pattern
- M2.5 Mounting Holes

Description

The CAT25512 Trēo™ Module is a 512kb EEPROM module that that features On Semiconductor's CAT25512 EEPROM IC. It features 512kb of memory, an identification memory page, memory protection levels, and communication speeds up to 10MHz. This module is a part of the NightShade Treo system, patent pending.

Table of Contents

1 Summary

The CAT25512 device is written to and read from using local memory buffers to stage a complete memory transaction. The device is initialized with the begin() method. A write operation is started with the startMemoryWrite() method. The transmit buffer is then loaded with the bytes to be written using the write() method. Finally, the data is written to the EEPROM by calling the endMemoryWrite() method. Similarly, a read operation is performed by calling the requestMemoryRead() method, which reads data from the EEPROM into a local receive buffer. The data is retrieved from the received buffer by using the read() method. The other library methods can be used to enable partial or full write protection and access the 128 byte identification page of the EEPROM.

2 What is Trēo™?

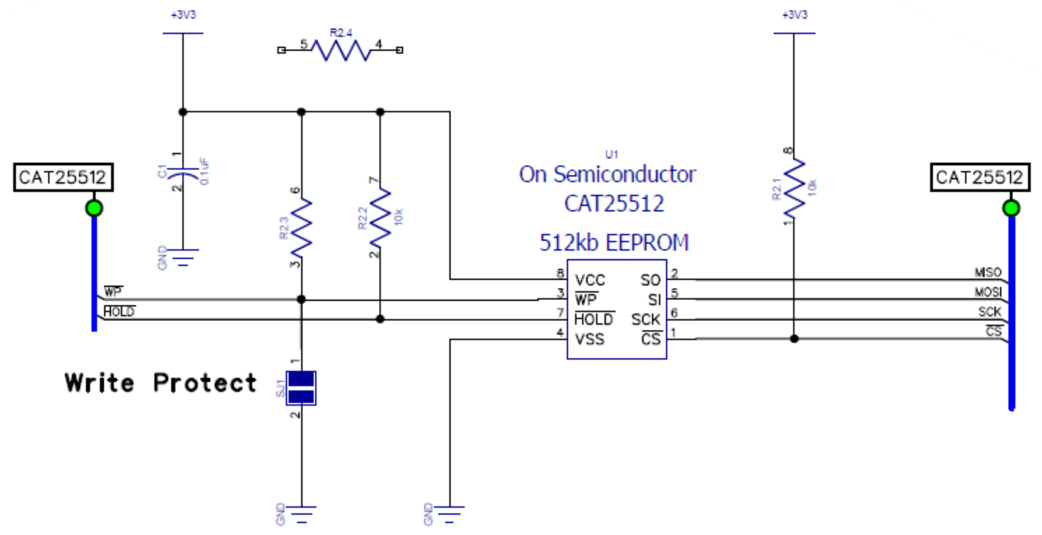
NightShade Trēo is a system of electronic modules that have standardized mechanical, electrical, and software interfaces. It provides you with a way to quickly develop electronic systems around microprocessor development boards. The grid attachment system, common connector/cabling, and extensive cross-platform software library allow you more time to focus on your application. Trēo is supported with detailed documentation and CAD models for each device.

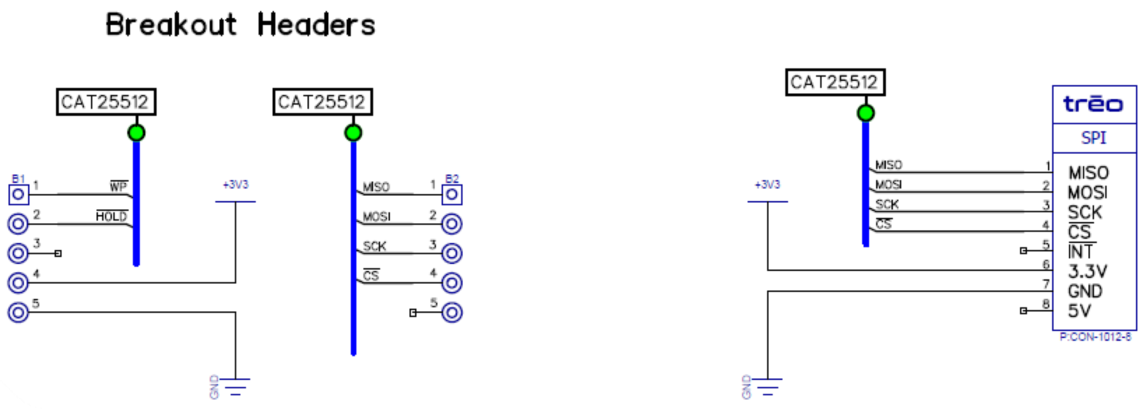
Learn more about Trēo here.

3 Electrical Characteristics

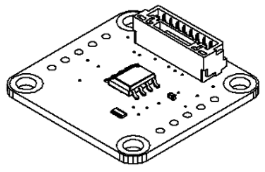
	Minimum	**Nominal**	**Maximum**
Voltages			
$V_{i/o}$ (MISO, MOSI, SCK, $\overline{CS}$)	-0.3V	-	3.6V
$V_{3.3V}$	3.1V	3.3V	3.5V
Communication			
SPI Clock Speed	DC	-	10MHz
Operating Temperature	-25°C	-	+85°C

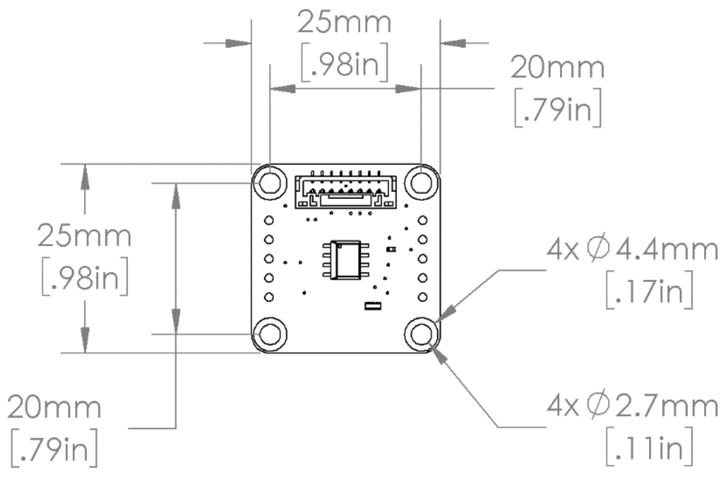
4 Electrical Schematic

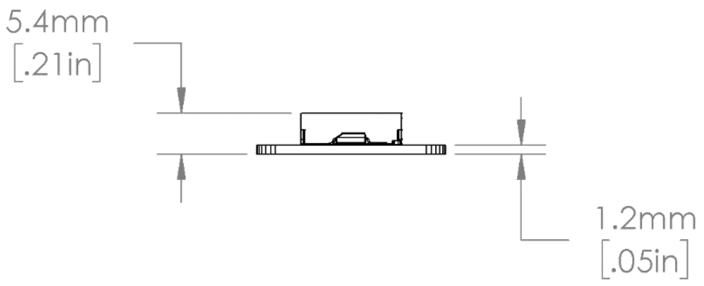




5 Mechanical Outline







6 Example Arduino Program

```
/**********************************************************
  CAT25512_EEPROM - NightShade_Treo by NightShade Electronics

  This sketch demonstrates the functionality of the
  NightShade Trēo CAT25512 EEPROM module. (NSE-1138-1) It
  performs write  and read cycles, playing "telephone" with
  the EEPROM.

  Created by Aaron D. Liebold
  on February 15, 2021

  Links:
  NightShade Trēo System: https://nightshade.net/treo
  Product Page: https://nightshade.net/product/treo-512kb-eeprom-cat25512/

  Distributed under the MIT license
  Copyright (C) 2021 NightShade Electronics
  https://opensource.org/licenses/MIT
**********************************************************/

#include <NightShade_Treo.h>

NightShade_Treo_CAT25512 eeprom(1, 10);

char buffer[128] =  "Only 128 bytes can be read or written to the CAT25512 EEPROM at a
time. This is called a page write/read...................128B";

void setup() {
  Serial.begin(115200);
  Serial.print("Initial character string: \"");
  Serial.print(buffer);
  Serial.println('\"');
  eeprom.begin();
}

void loop() {
  Serial.println("\nWriting last string to memory...");
  eeprom.startMemoryWrite(0x00);
  for (int x = 0; x < 128; ++x) eeprom.write(buffer[x]);
  eeprom.endMemoryWrite();
  Serial.println("Clearing software buffer.");
  for (int x = 0; x < 128; ++x) buffer[x] = 0;
  Serial.println("Buffer cleared.");
  Serial.println("Reading memory.");
  eeprom.requestMemoryRead(0x00, 128);
  for (int x = 0; x < 128; ++x) buffer[x] = eeprom.read();
  Serial.println("String read from EEPROM:");
  for (int x = 0; x < 128; ++x) Serial.write(buffer[x]);
  Serial.println();
  delay(500);
}
```

7 Library Overview (C++ & Python)

C++ Class

NightShade_Treo_CAT25512<classObject>();

Python Module

<classObject> = NightShade_Treo.CAT25512()

7.1 Constructors

NightShade_Treo_CAT25512(int spiPort, chipSelectPin, uint32_t spiClockSpeed)

Creates a CAT25512 object.

Arguments:

spiPort	Integer of the SPI port used (e.g. 0 = "/dev/spi_0")
chipSelectPin	Number of the pin connected to the chip select
spiClock	Desired clock speed for the bus

Returns:

Nothing

NightShade_Treo_CAT25512(int spiPort, int chipSelectPin)

Creates a CAT25512 object assuming the default clock speed.

Arguments:

port	Integer of the I2C port used. (e.g. 0 = "/dev/i2c_0")

Returns:

Nothing

7.2 Methods

begin()

Initializes the CAT25512 and enables write operations.

Arguments:

None

Returns:

Error	0 = Success

startMemoryWrite(int startAddress)

Clears the local transmit buffer to start a new write operation and enables write operations.

Arguments:

startAddress	EEPROM write address (0 – 0xFFFF)

Returns:

Error	0 = Success

write(uint8_t byte)

Appends a byte to the local transmit buffer.

Arguments:

byte	byte of data to be written

Returns:

Error	0 = Success

txAvailable()

Returns the number of bytes in the local transmit buffer.

Arguments:

None

Returns:

Error	0 = Success

endMemoryWrite()

Writes the data in the local transmit buffer to the EEPROM at the startAddress set with the startMemoryWrite() method.

Arguments:

None

Returns:

Error	0 = Success

requestMemoryRead(uint16_t startAddress, int numberOfBytes)

Reads a block of data from the EEPROM into the local receive buffer.

Arguments:

startAddress	first address of the memory read
numberOfBytes	number of bytes to read sequentially from the start address

Returns:

Error	0 = Success

rxAvailable()

Returns the number of bytes available in the local receive buffer.

Arguments:

None

Returns:

Error	0 = Success

read()

Returns the next byte from the local receive buffer.

Arguments:

None

Returns:

Next byte in receive local buffer (uint8_t)

enableHardwareWriteProtection(int enable)

The setting enables write protection to be enabled by the external $\overline{WP}$ pin.

Arguments:

enable	true/false

Returns:

Error	0 = Success

enableIdentificationPage(int enable)

When the identification page is enabled all of the memory functions access the 128 byte identification page rather than the main memory array. (Addresses 0x00 – 0x7F)

Arguments:

enable	true/false

Returns:

Error	0 = Success

permanentlyLockIdentificationPage(int enable)

Calling this method PERMENTANTLY protects the identification page from any write operations.

Arguments:

enable	true/false (CAUTION! This cannot be undone.)

Returns:

Error	0 = Success

setProtectionLevel(int protectionLevel)

The EEPROM memory can be write protected in block with different protection levels.

Arguments:

protectionLevel	0: No Protection	
	1: Quarter Array Protection	0xC000 – 0xFFFF
	2: Half Array Protection	0x8000 – 0xFFFF
	3: Full Array Protection	0x0000 – 0xFFFF

Returns:

Error	0 = Success

writeStatusReg(uint8_t regValue)

Write the Status register value.

Arguments:

regValue	Status register value

Returns:

Error	0 = Success

readStatusReg()

Read the current value of the Status register.

Arguments:

None

Returns:

Status register value (uint8_t)