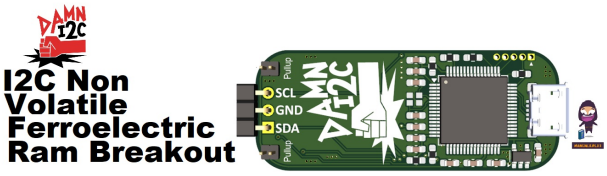
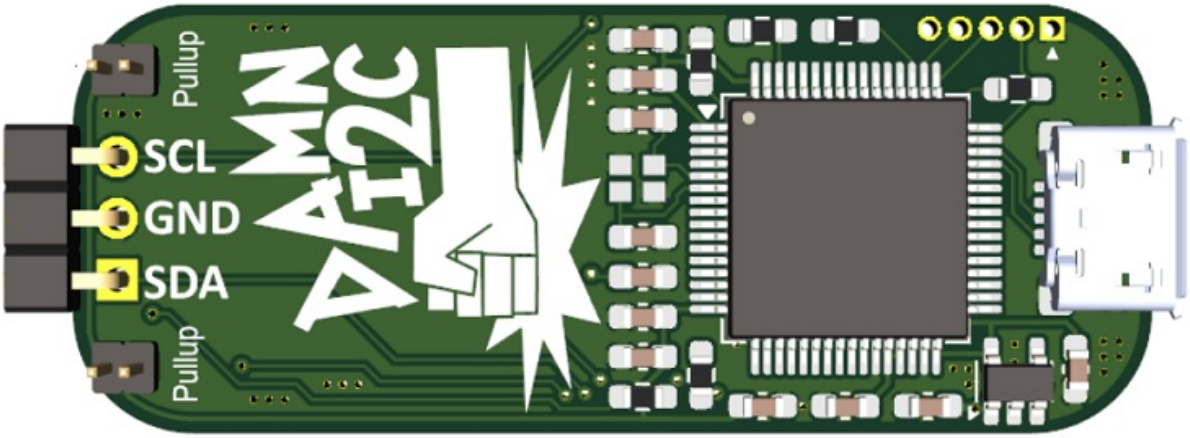


DAMN I2C Non Volatile Ferroelectric Ram Breakout Instruction Manual

Contents

DAMN I2C Non Volatile Ferroelectric Ram Breakout



Specifications

- **Device:** DamnI2C USB Dongle
- **Operating Voltage:** 3.3V
- **Application Pins:** SCL, SDA, GND
- **Software:** Windows application for I2C operations
- **API Support:** Python and other languages with Serial Communications
- **Interface:** USB Serial Device (COMx)

Product Usage Instructions

Setting up DamnI2C Dongle

1. Connect the DamnI2C Dongle to your PC via USB.
2. Connect the application pins (SCL, SDA, GND) to your target circuit.
3. Use the provided software or API to control the dongle.

Using DamnI2C Software

1. Download the software from **www.damntools.com**.
2. Launch the software on your Windows PC.
3. Perform various I2C operations like Bus Scan, Register Read/Write, etc.

API Guide for Custom Applications

Refer to the API Guide for detailed information on integrating DamnI2C with Python or other supported programming languages.

Frequently Asked Questions

Q: What is the operating voltage of the DamnI2C Dongle?

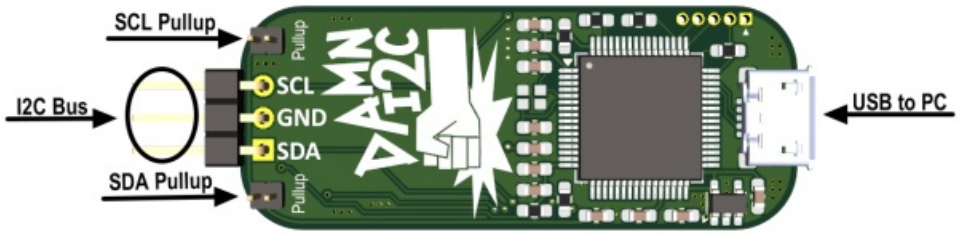
A: The DamnI2C Dongle operates at 3.3V.

Q: Where can I download the DamnI2C software?

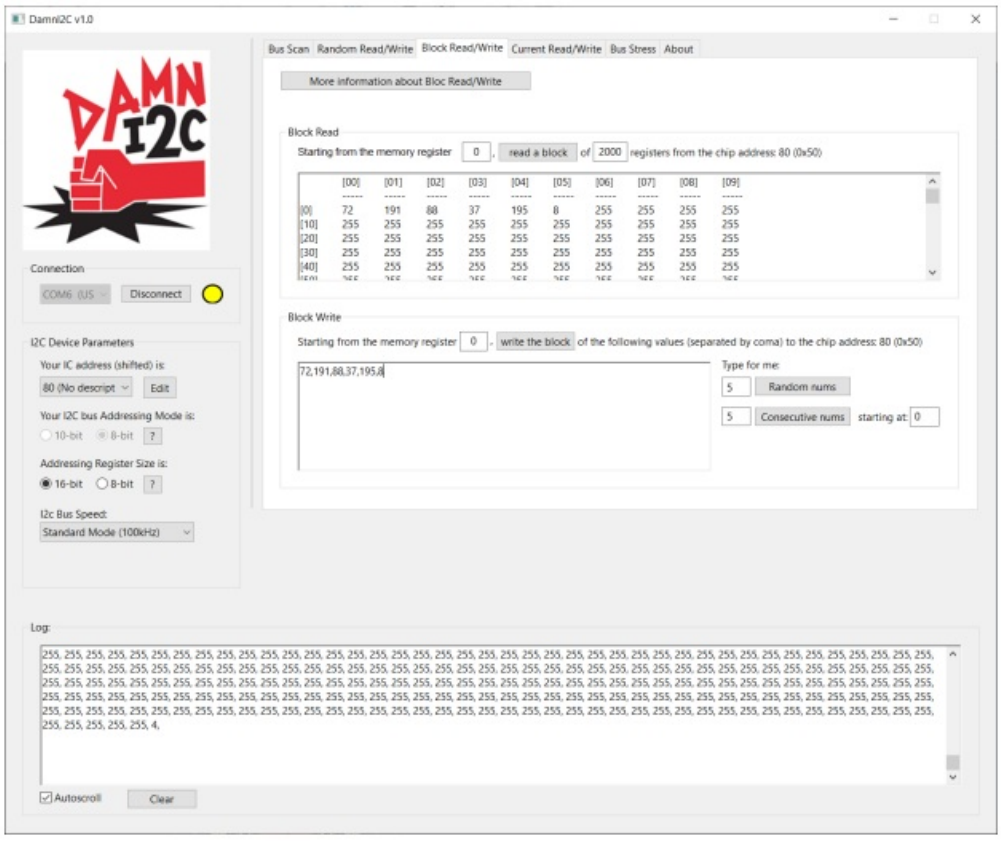
A: You can download the software from **www.damntools.com**.

General Description

- DamnI2C is a PC-controlled I2C-Master device consisting of a USB dongle and accompanying PC software.
- The DamnI2C Dongle has three application pins that should be connected to your target circuit: SCL, SDA, and GND. It operates at 3.3V and includes two jumpers to pull up the SCL and SDA lines if needed.



- The PC software is an extremely user-friendly Windows application that allows you to perform various I2C operations such as Bus Scan, Single Register Read/Write, Register Block Read/Write, and more.
- Customers can either use the provided software or the API documented here to create their applications using Python or any programming language that supports Serial Communications. The dongle appears as a USB Serial Device (COMx).
- The software can be downloaded from **www.damntools.com**.



API Guide

Important considerations

Number Notation

- All the number notations are in decimal except the ones beginning with the notation “0x” which are in

hexadecimal.

I2C Device Address Byte Notation

- All references to the I2C Device Addresses are ‘shifted,’ meaning the R/W bit (LSB) is not included in the Device Address. The Device Address is shifted one position to the right to avoid confusion between read and write operations.
- **Example:** An EEPROM with a Read Address of 160 and a Write Address of 161 has a Shifted I2C Device Address of 80.

I2C Device Address Byte							
Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
1	0	1	0	0	0	0	R/W
80							

I2C Charts Legend

The API Guide includes I2C frame captures showing the results of the operations described in each section.

The I2C chart legend is as follows:

S	I2C Start condition detected
W	Write Bit
Rd	Read Bit
Sr	Repeated Start condition detected
A	Acknowledge Bit
N	Not Acknowledge Bit
P	Stop condition detected

Read Operations

Single Register Read Description

- The Single Register Read operation reads a specific register. The PC must specify the I2C Device Address, the Register to Read, and whether the Register address is 16-bit or 8-bit. A 16-bit Register Addressing is used for EEPROM memories with more than 255 registers, as more than one byte is required to address registers higher than 255. An 8-bit addressing is used for EEPROMs with 255 or fewer registers.
- The DamnI2C dongle can respond with two different function codes: Function Code 2, which returns the read value if the operation is successful, or Function Code 3 if an error occurs. The Error ID is also provided in the event of an error.

For both Request and Response, the frame structure includes:

- **Frame Start:** Represented by the value 1.

- **Frame End:** Represented by the value 4.
- **Frame Size:** A 24-bit value indicating the total number of bytes in the frame.
- These values help identify the start and end of the frame.

Request

Byte index: Byte description Byte value

0	1	2	3	4	5	6	7	8	9
Frame Start	**Frame Size (24-bit)**			**Frame Type**	**I2C Device Address**		**Register to Read**		**Frame End**
	Hi	Mid	Lo		Hi	Lo	Hi	Lo	
1	0	0	10	1 = Random Read Request (8-bit Addressing)	X	X	X	X	4
				10 = Random Read Request (16-bit Addressing)					

Responses

Response 1: Single Register Read Operation Succeed

Byte index:	0	1	2	3	4	5	6
Byte Description:	**Frame Start**	**Frame Size (24-bit)**			**Frame Type**	**Read Value**	**Frame End**
		Hi	Mid	Lo			
Byte value:	1	0	0	7	2 = Random Read Answer Ok	x	4

Response 2: Single Register Read Operation Error

Byte index:	0	1	2	3	4	5	6
Byte Description:	**Frame Start**	**Frame Size (24-bit)**			**Frame Type**	**ID Error**	**Frame End**
		Hi	Mid	Lo			
Byte value:	1	0	0	7	3 = Random Read Answer Error	1: General error 2: Busy 3: Timeout	4

Example: 16-bit addressing Read

- The PC wants to read from a 16-bit addressing EEPROM with the address 0x80 the register 3.

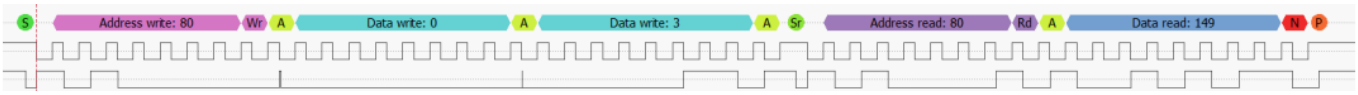
Request

- 1, 0, 0, 10, 10, 0, 80, 0, 3, 4

Response

- 1, 0, 0, 7, 2, 149, 4
- The read value is 149.

I2C Output



Example: 8-bit addressing Read

- The PC wants to read from an 8-bit addressing EEPROM with the address 0x80 the register 3.

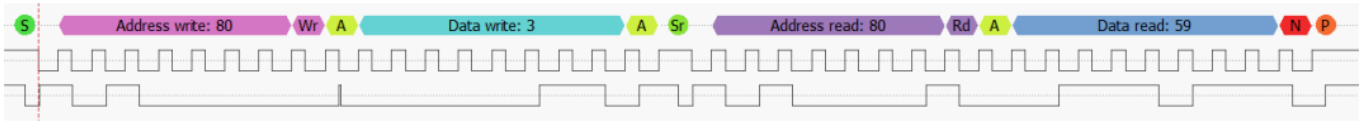
Request

- 1, 0, 0, 10, 1, 0, 80, 0, 3, 4

Answer

- 1, 0, 0, 7, 2, 59, 4
- The read value is 59.

I2C Output



Block Read Description

- The Block Read operation performs a massive read of the contiguous number of registers. The PC must specify the I2C Device ID, the starting register, the total amount of registers to read (1 to 2000), and whether the register addressing is 16-bit or 8-bit.
- A 16-bit register address is used for EEPROM memories with more than 255 registers, as more than one byte is required to address registers higher than 255. An 8-bit address is used for EEPROMs with 255 or fewer registers.
- **The DamnI2C dongle can respond with two different function codes:** Function Code 8, which returns the read value if the operation is successful, or Function Code 9 if an error occurs. The Error ID is also provided in the event of an error.

For both Request and Response, the frame structure includes:

- **Frame Start:** Represented by the value 1.
- **Frame End:** Represented by the value 4.
- **Frame Size:** A 24-bit value indicating the total number of bytes in the frame.
- These values help identify the start and end of the frame.

Request

Byte index:	0	1	2	3	4	5	6	7	8	9	10	11
Byte Description:	**Frame Start**	**Frame Size (24-bit)**			**Frame Type**	**I2C Device Address**		**Start Register**		**Quantity of Registers (1 to 2000)**		**Frame End**
		Hi	**Mid**	**Lo**		Hi	Lo	Hi	Lo	Hi	Lo	
Byte value:	1	0	0	12	7 = Block Read Request (8-bit Addressing)	x	x	x	x	x	x	4
					12 = Block Read Request (16-bit Addressing)							
					41 = Block Read In Bus Stress Mode (8-bit Addressing)							
					42 = Block Read In Bus Stress Mode (16-bit Addressing)							
					43 = Stop Bus Stress Mode							

Responses

Response 1: Block Read Operation Succeed

Byte index:	0	1	2	3	4	5	6	7	…	Last
Byte Description:	**Frame Start**	**Frame Size (24-bit)**			**Frame Type**	**Start Register**		**Data**		**Frame End**
		Hi	Mid	Lo		Hi	Lo			
Byte value:	1	[Quantity of Registers of the Bloc Read Request] + 8			8 = Block Read Answer Ok	x	x	Read Value 1	Read Value n	4

Response 2: Block Read Operation Error

Byte index:	0	1	2	3	4	5	6
Byte Description:	**Frame Start**	**Frame Size (24-bit)**			**Frame Type**	**ID Error**	**Frame End**
		Hi	Mid	Lo			
Byte value:	1	0	0	7	9 = Block Read Answer Error	1: General error 2: Busy 3: Timeout	4

Example: 16-bit addressing Block Read

- The PC wants to read from a 16-bit addressing EEPROM with the address 0x80, the registers 300, 301, 302.

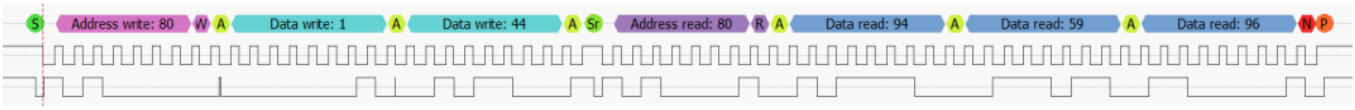
Request

- To perform the operation, we should indicate the start address is 300 and that the number of bytes to read is 3:
- 1, 0, 0, 12, 12, 0, 80, 1, 44, 0, 3, 4

Response

- 1, 0, 0, 11, 8, 1, 44, 94, 59, 96, 4
- The read values are 94, 59, and 96.

I2C Output



Example: 8-bit addressing Block Read

- The PC wants to read from a 16-bit addressing EEPROM with the address 0x80, the registers 20, 21, 22.

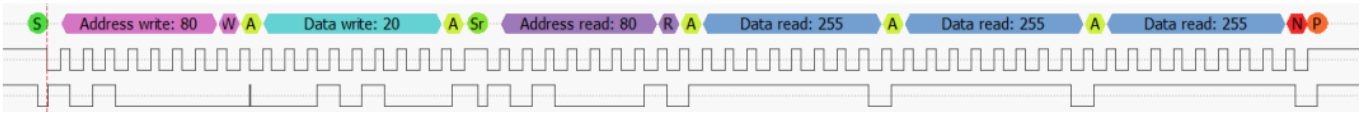
Request

- 1, 0, 0, 12, 7, 0, 80, 0, 20, 0, 3, 4

Response

- 1, 0, 0, 11, 8, 0, 20, 255, 255, 255, 4
- The read values are 255, 255, 255.

I2C Output



Current Address Read Description

- The Current Address Read operation performs a read wherever the register addressing pointer is. The PC must specify only the I2C Device ID and whether the register addressing is 16-bit or 8-bit. A 16-bit register address is used for EEPROM memories with more than 255 registers, as more than one byte is required to address registers higher than 255. An 8-bit address is used for EEPROMs with 255 or fewer registers.
- **The DamnI2C dongle can respond with two different function codes:** Function Code 34, which returns the read value if the operation is successful, or Function Code 35 if an error occurs. The Error ID is also provided in the event of an error.

For both Request and Response, the frame structure includes:

- **Frame Start:** Represented by the value 1.
- **Frame End:** Represented by the value 4.
- **Frame Size:** A 24-bit value indicating the total number of bytes in the frame.
- These values help identify the start and end of the frame.

Request

Byte index: Byte description Byte value

0	1	2	3	4	5	6	7
Frame Start	**Frame Size (24-bit)**			**Frame Type**	**I2C Device Address**		**Frame End**
	Hi	Mid	Lo		Hi	Lo	
1	0	0	8	26 = Current Address Read Request (8-bit Addressing)	X	X	4
				27 = Current Address Read Request (16-bit Addressing)			

Response

Response 1: Current Address Read Operation Succeed

Byte index: Byte description Byte value

0	1	2	3	4	5	6
Frame Start	**Frame Size (24-bit)**			**Frame Type**	**Read Value**	**Frame End**
	Hi	Mid	Lo			
1	0	0	7	34 = Current Address Read Answer Ok	x	4

Response 2: Current Address Read Operation Error

Byte index: Byte description Byte value

0	1	2	3	4	5	6
Frame Start	**Frame Size (24-bit)**			**Frame Type**	**ID Error**	**Frame End**
	Hi	Mid	Lo			
1	0	0	7	35 = Current Address Read Answer Error	1: General error 2: Busy 3: Timeout	4

Example: Current Address Read

- The PC wants to read the current address from an EEPROM with the address 0x80.

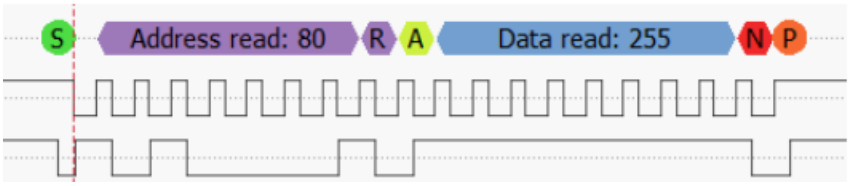
Request

- 1, 0, 0, 8, 27, 0, 80, 4

Response

- 1, 0, 0, 7, 34, 255, 4
- The read value is 255.

I2C Output



Write Operations

Single Register Write Description

- Single Register Write operation performs a write of a specific value. The PC must specify the I2C Device Address, the Register to Write, and whether the Register address is 16-bit or 8-bit. A 16-bit Register Addressing is used for EEPROM memories with more than 255 registers, as more than one byte is required to address registers higher than 255. An 8-bit addressing is used for EEPROMs with 255 or fewer registers.
- The DamnI2C dongle can respond with two different function codes: Function Code 5, which returns the read value if the operation is successful, or Function Code 6 if an error occurs.
- The Error ID is also provided in the event of an error.

For both Request and Response, the frame structure includes:

- **Frame Start:** Represented by the value 1.
- **Frame End:** Represented by the value 4.
- **Frame Size:** A 24-bit value indicating the total number of bytes in the frame.
- These values help identify the start and end of the frame.

Request

Byte index: Byte description Byte value

0	1	2	3	4	5	6	7	8	9
Frame Start	**Frame Size (24-bit)**			**Frame Type**	**I2C Device Address**		**Register to Write**		**Frame End**
	Hi	Mid	Lo		Hi	Lo	Hi	Lo	
1	0	0	11	4 = Random Write Request (8-bit Addressing)	X	X	X	X	4
				11 = Random Write Request (16-bit Addressing)					

Response

Response 1: Single Register Write Operation Succeeds

Byte index:	0	1	2	3	4	5	6
Byte Description:	**Frame Start**	**Frame Size (24-bit)**			**Frame Type**	**Data**	**Frame End**
		Hi	Mid	Lo			
Byte value:	1	0	0	7	5 = Random Write Answer OK	Don't care	4

Response 2: Single Register Write Operation Error

Byte index:	0	1	2	3	4	5	6
Byte Description:	**Frame Start**	**Frame Size (24-bit)**			**Frame Type**	**ID Error**	**Frame End**
		Hi	Mid	Lo			
Byte value:	1	0	0	7	6 = Random Write Answer Error	1: General error 2: Busy 3: Timeout	4

Example: 16-bit addressing Single Register Write

- The PC wants to write the value 12 to address 300 of a 16-bit addressing EEPROM with a Device Address of 80.

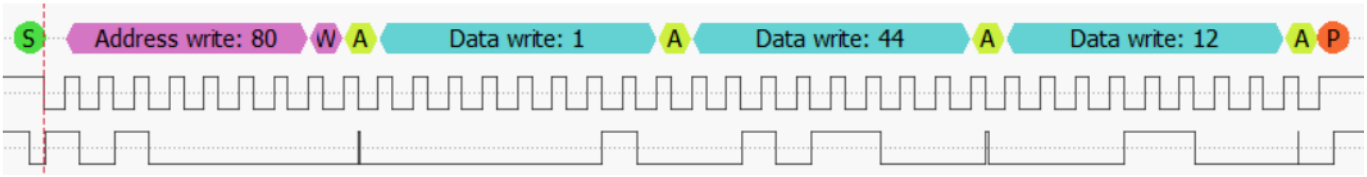
Request

- 1, 0, 0, 11, 11, 0, 80, 1, 44, 12, 4

Answer

- 1, 0, 0, 7, 5, 0, 4
- Operation Succeed.

I2C Output



Example: 8-bit addressing Single Register Write

- The PC wants to write the value 12 to address 50 of an 8-bit addressing EEPROM with a Device Address of 80.

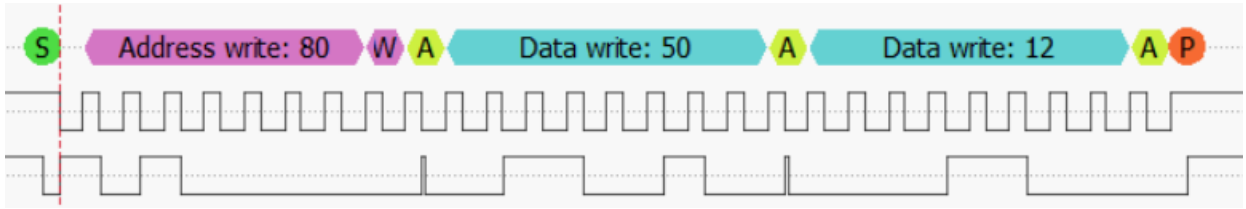
Request

- 1, 0, 0, 11, 4, 0, 80, 0, 50, 12, 4

Response

- 1, 0, 0, 7, 5, 0, 4
- Operation Succeed.

I2C Output



Block Write Description

- The Block Write operation performs a massive write of the contiguous number of registers. The PC must specify the I2C Device ID, the starting register, the register values to write (1 to XXX), and whether the register addressing is 16-bit or 8-bit.
- A 16-bit register address is used for EEPROM memories with more than 255 registers, as more than one byte is required to address registers higher than 255. An 8-bit address is used for EEPROMs with 255 or fewer registers.
- The DamnI2C dongle can respond with two different function codes: Function Code 24, which returns the read value if the operation is successful, or Function Code 25 if an error occurs. The Error ID is also provided in the event of an error.

For both Request and Response, the frame structure includes:

- **Frame Start:** Represented by the value 1.
- **Frame End:** Represented by the value 4.
- **Frame Size:** A 24-bit value indicating the total number of bytes in the frame.
- These values help identify the start and end of the frame.

Request

Byte index: Byte description Byte value

0	1	2	3	4	5	6	7	8	9	10	n	Last
Frame Start	**Frame Size (24-bit)**			**Frame Type**	**I2C Device Address**		**Starting Register**		**Values**			**Frame End**
	Hi	Mid	Lo		Hi	Lo	Hi	Lo	1st	2nd	n	
1	0	0	10	20 = Block Write Request (8-bit Addressing)	x	x	x	x	x	x	x	4
				21 = Block Write Request (16-bit Addressing)								

Response

Response 1: Bloc Write Operation Succeed

Byte index:	0	1	2	3	4	5	6
Byte Description:	**Frame Start**	**Frame Size (24-bit)**			**Frame Type**	**Read Value**	**Frame End**
		Hi	Mid	Lo			
Byte value:	1	0	0	7	24 = Block Write Answer Ok	x	4

Response 2: Block Write Operation Error

Byte index:	0	1	2	3	4	5	6
Byte Description:	**Frame Start**	**Frame Size (24-bit)**			**Frame Type**	**ID Error**	**Frame End**
		Hi	Mid	Lo			
Byte value:	1	0	0	7	25 = Block Write Answer Error	1: General error 2: Busy 3: Timeout	4

Example: 16-bit addressing Block Write

- The PC wants to write the values 12, 13, 14, and 15 to the Starting Address 300 of a 16-bit addressing

EEPROM with a Device Address of 80.

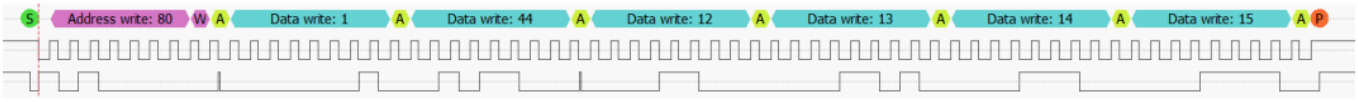
Request

- 1, 0, 0, 14, 21, 0, 80, 1, 44, 12, 13, 14, 15, 4

Response

- 1, 0, 0, 7, 24, 0, 4
- Operation Succeed.

I2C Output



Example: 16-bit addressing Block Write

- The PC wants to write the values 12, 13, 14, and 15 to the Starting Address 300 of an 8-bit addressing EEPROM with a Device Address of 80.

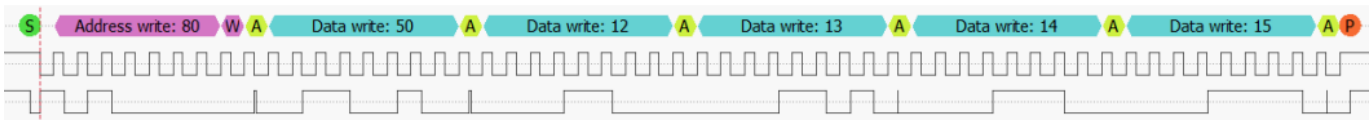
Request

- 1, 0, 0, 14, 20, 0, 80, 0, 50, 12, 13, 14, 15, 4

Response

- 1, 0, 0, 7, 24, 0, 4
- Operation Succeed.

I2C Output



Current Address Write Description

- The Current Address Write operation performs a write where the register addressing pointer is. The PC must specify the I2C Device ID, the data to write, and whether the register addressing is 16-bit or 8-bit.
- A 16-bit register address is used for EEPROM memories with more than 255 registers, as more than one byte is required to address registers higher than 255. An 8-bit address is used for EEPROMs with 255 or fewer registers.
- The DamnI2C dongle can respond with two different function codes: Function Code 36, which returns the read value if the operation is successful, or Function Code 37 if an error occurs. The Error ID is also provided in the event of an error.

For both Request and Response, the frame structure includes:

- **Frame Start:** Represented by the value 1.
- **Frame End:** Represented by the value 4.
- **Frame Size:** A 24-bit value indicating the total number of bytes in the frame.
- These values help identify the start and end of the frame.

Request

Byte index: Byte description Byte value

0	1	2	3	4	5	6	7	8
Frame Start	**Frame Size (24-bit)**			**Frame Type**	**I2C Device Address**		**Data to Write**	**Frame End**
	Hi	Mid	Lo		Hi	Lo		
1	0	0	12	30 = Current Address Write Request (8-bit Addressing)	x	x	x	4
				31 = Current Address Write Request (16-bit Addressing)				

Responses

Response 1: Current Address Write Operation Succeed

Byte index:	0	1	2	3	4	5	6	7	n	Last Byte
Byte Description:	**Frame Start**	**Frame Size (24-bit)**			**Frame Type**	**Start Register**		**Data**		**Frame End**
		Hi	Mid	Lo		Hi	Lo			
Byte value:	1	[Quantity of Registers of the Bloc Read Request] + 8			36 = Current Address Write Answer Ok	x	x	Read Value 1	Read Value n	4

Response 2: Current Address Write Operation Error

Byte index:	0	1	2	3	4	5	6
Byte Description:	**Frame Start**	**Frame Size (24-bit)**			**Frame Type**	**ID Error**	**Frame End**
		Hi	Mid	Lo			
Byte value:	1	0	0	7	37 = Current Address Write Answer Error	1: General error 2: Busy 3: Timeout	4

Example: Current Address Write

- The PC wants to write the value 15 to the current location of the address pointer. This operation is risky because the exact write location is not explicitly known. Additionally, EEPROMs do not support this type of operation.

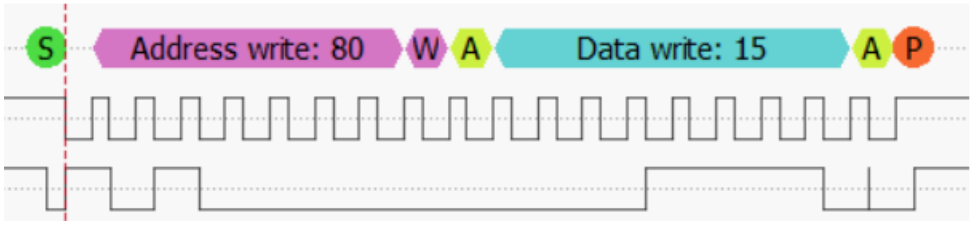
Request

- 1, 0, 0, 9, 30, 0, 80, 15, 4

Response

- 1, 0, 0, 7, 36, 0, 4
- Operation Succeed.

I2C Output



Other Operations

I2C Bus Scan Description

This operation scans the I2C Device Addresses from 0 to 127 to check for available chips on the bus.

The DamnI2C Dongle can respond with two different function codes:

1. **Function Code 18:** This code indicates that the operation succeeded and returns an array of 127 bytes, each representing one of the 127 possible I2C device addresses on the bus. The first byte corresponds to I2C device address 0, the second byte to address 1, and so on. A value of 1 in the array indicates an error, meaning the device was not detected, while a value of 0 means the device was detected.
2. **Function Code 19:** This code indicates that the bus scan operation could not be completed. Note that Function Code 19 should not be confused with a successful bus scan that found no I2C devices; in the latter case, the scan operation succeeds, but no devices are detected.

For both Request and Response, the frame structure includes:

- **Frame Start:** Represented by the value 1.
- **Frame End:** Represented by the value 4.
- **Frame Size:** A 24-bit value indicating the total number of bytes in the frame.

These values help identify the start and end of the frame.

Request

Byte index:	0	1	2	3	4	5	6
Byte Description:	**Frame Start**	**Frame Size (24-bit)**			**Frame Type**	**Data**	**Frame End**
		Hi	Mid	Lo			
Byte value:	1	0	0	7	17 = Bus Scan Request	Don’t care	4

Responses

Response 1: Bus Scan Operation Succeeds

0	1	2	3	4	5	6	7	8	9	10	n	Last byte
Frame Start	**Frame Size (24-bit)**			**Frame Type**	**I2C Device Address**		**Start Register**		**I2C Device Detected Value**			**Frame End**
	Hi	Mid	Lo		Hi	Lo	Hi	Lo	Device 0	Device 1	Device N	
1	0	0	134	18 = Bus Scan Answer Ok	x	x	x	x	0: I2C Device ID Detected 1: I2C Device ID Not Detected			4

Response 2: Bus Scan Operation Error

Byte index:	0	1	2	3	4	5	Last byte
Byte Description:	**Frame Start**	**Frame Size (24-bit)**			**Frame Type**	**Error ID**	**Frame End**
		Hi	Mid	Lo			
Byte value:	1	0	0	7	19 = Bus Scan Answer Error	1: General error 2: Busy 3: Timeout	4

Example I2C Bus Scan

- The PC wants to discover which I2C devices are present on the bus.

Request

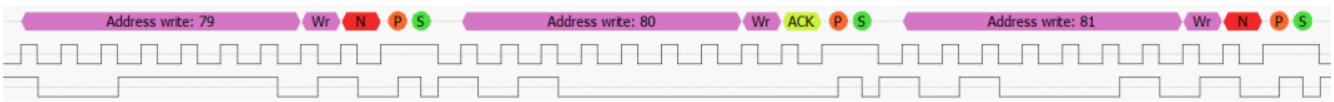
- 1, 0, 0, 7, 17, 0, 4

Response

- 1, 0, 0, 134, 18, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1,
- 1, 1, 1, 1, 1, 1, 1, 1, 0, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 4
- All the 'ones' after the number 18 in the response correspond to the tested I2C device addresses that did not respond to the call, indicating they were not present on the bus.
- A 'zero' appears at position 80, indicating that a chip is responding at I2C device address 80. Therefore, only one chip is present on the bus at that address.

I2C Output

- The full frame capture is not shown due to the large amount of data. Instead, only the relevant part of the frame where address 80 is detected is shown.
- I2C devices at addresses 79 and 81 are not detected, making it highly likely that there are no chips with those addresses. The device at I2C address 80 is detected, indicating that a chip with this address exists on the bus.



Get DamnI2C Dongle Status Description

- This operation checks if the DamnI2C Dongle is responding or not. It can be understood as the ping command in the networking world.

For both Request and Response, the frame structure includes:

- **Frame Start:** Represented by the value 1.
- **Frame End:** Represented by the value 4.
- **Frame Size:** A 24-bit value indicating the total number of bytes in the frame.
- These values help identify the start and end of the frame.

Request

Byte index: Byte description Byte value

0	1	2	3	4	5	6
Frame Start	Frame Size (24-bit)			Frame Type	Data	Frame End
	Hi	Mid	Lo			
1	0	0	7	15 = DamnI2C Dongle Status Request	Don’t care	4

Response

Byte index:	0	1	2	3	4	5	6	7	8 to 19	20
Byte Description:	Frame Start	Frame Size (24-bit)			Frame Type	Data				Frame End
		Hi	Mid	Lo		State	FW Version	HW Version	Not used	
Byte value:	1	0	0	21	16 = DamnI2C Dongle status Request	0: Ready	x	x	Don’t care	4

Example DamnI2C Dongle Status Request

- 1, 0, 0, 7, 15, 0, 4

Response

- 1, 0, 0, 7, 16, 0, 4
- A dongle is Ready.

I2C Output

- This operation does not perform any I2C action.

Configure I2C Speed Description

- This operation changes the I2C Bus speed.
- The DamnI2C dongle can respond with two different function codes: Function Code 39, which returns the read value if the operation is successful, or Function Code 40 if an error occurs. The Error ID is also provided in the event of an error.

For both Request and Response, the frame structure includes:

- **Frame Start:** Represented by the value 1.
- **Frame End:** Represented by the value 4.
- **Frame Size:** A 24-bit value indicating the total number of bytes in the frame.
- These values help identify the start and end of the frame.

Request

Byte index: Byte description Byte value

0	1	2	3	4	5	6
Frame Start	Frame Size (24-bit)			Frame Type	Data	Frame End
	Hi	Mid	Lo			
1	0	0	7	38 = Configure I2C Speed	0: 100kHz 1: 400kHz 2: 1000kHz	4

Response

Response 1: Configure I2C Bus Speed Operation Succeed

Byte index: Byte description Byte value

0	1	2	3	4	5	6
Frame Start	Frame Size (24-bit)			Frame Type	Data	Frame End
	Hi	Mid	Lo			
1	0	0	7	39 = Configure I2C Speed Answer Ok	0: 100kHz 1: 400kHz 2: 1000kHz	4

Response 2: Configure I2C Bus Speed Operation Error

Byte index: Byte description Byte value

0	1	2	3	4	5	6
Frame Start	Frame Size (24-bit)			Frame Type	Data	Frame End
	Hi	Mid	Lo			
1	0	0	7	40 = Configure I2C Speed Answer Error	Don’t care	4

Example DamnI2C Dongle Status Request

- 1, 0, 0, 7, 38, 1, 4

Response

- 1, 0, 0, 7, 39, 1, 4
- Operation Succeed.

I2C Output

- This operation does not perform any I2C action.

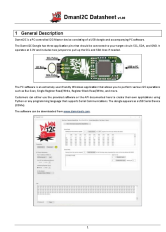
Operation Codes

Code	Description
1	Random Read Request (8-bit Address)
2	Random Read Answer Ok
3	Random Read Answer Error
4	Random Write Request (8-bit Register Address)
5	Random Write Answer OK
6	Random Write Answer Error
7	Block Read Request (8-bit Device Address, 8-bit Register Address)
8	Block Read Answer Ok
9	Block Read Answer Error

10	Random Read Request (16-bit Register Address)
11	Random Write Request (16-bit Register Address)
12	Block Read Request (8-bit Device Address, 16-bit Register Address)
13	Block Read Request (10-bit Device Address, 8-bit Register Address) NOT IMPLEMENTED
14	Block Read Request (10-bit Device Address, 16-bit Register Address) NOT IMPLEMENTED
15	DamnI2C Status Request
16	DamnI2C Status Answer
17	Bus Scan Request
18	Bus Scan Answer Ok
19	Bus Scan Answer Error
20	Block Write Request (8-bit Device Address, 8-bit Register Address)
21	Block Write Request (8-bit Device Address, 16-bit Register Address)
22	Block Write Request (10-bit Device Address, 8-bit Register Address) NOT IMPLEMENTED
23	Block Write Request (10-bit Device Address, 16-bit Register Address) NOT IMPLEMENTED
24	Block Write Answer Ok
25	Block Write Answer Error
26	Current Address Read Request (8-bit Register Address)
27	Current Address Read Request (16-bit Register Address)
28	Current Address Read Request (10-bit Device Address, 8-bit Register Address) NOT IMPLEMENTED
29	Current Address Read Request (10-bit Device Address, 16-bit Register Address) NOT IMPLEMENTED
30	Current Address Write Request (8-bit Device Address, 8-bit Register Address)
31	Current Address Write Request (8-bit Device Address, 16-bit Register Address)
32	Current Address Write Request (10-bit Device Address, 8-bit Register Address) NOT IMPLEMENTED
33	Current Address Write Request (10-bit Device Address, 16-bit Register Address) NOT IMPLEMENTED

34	Current Address Read Answer Ok
35	Current Address Read Answer Error
36	Current Address Write Answer Ok
37	Current Address Write Answer Error
38	Configure I2C Speed Request
39	Configure I2C Speed Response Ok
40	Configure I2C Speed Response Error
41	Block Read In Bus Stress Mode (8-bit Device Address, 8-bit Register Address)
42	Block Read In Bus Stress Mode (8-bit Device Address, 16-bit Register Address)
43	Stop Bus Stress Mode

Documents / Resources

	DAMN I2C Non Volatile Ferroelectric Ram Breakout [pdf] Instruction Manual I2C Non Volatile Ferroelectric Ram Breakout, I2C, Non Volatile Ferroelectric Ram Breakout, Ferroelectric Ram Breakout, Ram Breakout

References

- damntools.com
- damntools.com
- User Manual