The I2C master protocol

Netzer supports I2C master since version 1.2. The most data transactions are done transparently by Netzer. But a few protocol specifiers are implemented for handling the special states of I2C.

NULL - The termination character

The character *0x00* in the socket communication has special meaning in terminating any pending I2C transmission. If Netzer sends a *0x00* either the bus is busy, a bus collission occurred or a slave does not respond on address or data bytes. In the latter cases the bus is freed with a stop condition automatically. If Netzer receives *0x00* while a I2C transmission is pending, the transmission will be finished with a automatic generated stop condition.



If *0x00* should be sent via the I2C interface it must be escaped with a leading backspace. Therefore also the backslash itself must be escaped.

The start condition and the slave address byte

A start condition is automatically announced on the bus, if Netzer gets the first character via socket. If an error occurs while announcing the start condition Netzer responds with a *0x00*. I2C protocol defines the very first byte after the start condition as slave address. Netzer sends a *0x00* via socket if no slave responds (ACK bit was '1'). Also a stop condition is initiated by Netzer. After the successful (means acknowledged) transmission of the slave address Netzer sends *0xFF* via socket.

General call address

The I2C general call address *0x00* addresses all slaves on the I2C bus. Sending *0x00* as the very first byte is allowed as an exception of the Netzer protocol. The escaping backslash MUST not be used.

The R/W bit

Netzer also checks the R/W bit (LSB) of the slave address character.

Master write

If the R/W bit is **cleared** a transmission from master to slave is initiated.

Each byte received by Netzer via the socket is directly sent to the slave (consider escaping!). In response Netzer sends either '0xFF' as positive acknowledge (ACK bit was '0') or *0x00* as negative acknowledge (ACK bit was '1'). In case of a negative acknowledge Netzer initiates a stop condition on the bus.

Master read

If the R/W bit is **set** a transmission from slave to master is initiated (clock is generated by I2C master).

After the successful transmission of the slave address Netzer waits for further bytes on the socket. On receiving a byte via socket a byte is pulled from the slave and is sent. In case the received byte is *0x00* the transmission is over and Netzer generates a stop condition on the bus. Any other value pulls the next byte from slave.

Repeated start

I2C supports repeated start. This symbol concatenates a stop and start condition while a I2C transmission is pending without freeing the bus inbetween.

The special character *s* (0x73) in socket stream initiates the repeated start on the bus. Therefore *0x73* must be escaped during normal stream transmission to Netzer. Exception to escaping *0x73* is the very first character (slave address) analog to the “General Call address” section.



Receiving *0x73* from Netzer has no special meaning but *0x73* and therefore will not be escaped!

Framing (Since version 1.6)

The protocol was slightly reworked to use it in a more framing like fashion. This simplifys the implementation of the protocol on host.

Each action on the bus is considered like a socket frame terminated by *0x00*. This rule is applied for both directions. So each frame from host is replied by a socket frame.

In case a frame is finished and no more frames wait for processing the socket is flushed. This leads to a higher throughput especially on network socket connections.

From bus to socket

Bus collisions, not acknowedged slave adresses or acknowledged written bytes send the termination character *0x00* to the socket host. The frame is also terminated on bus with the stop condition in

most cases (not on bus collisions).

The socket is in termination state afterwards. In this state no more actions will be initiated on bus. The socket handler only waits for termination character *0x00* from host.

From socket to bus

The termination character stops any pending communication and also terminates any pending frame.

Examples

The following examples presume that there is a connection to Netzer.

Take a look at the following picture for circuiting a I2C EEPROM IC as slave to Netzer. image html EEPROM_Example.gif Connecting a simple EEPROM to Netzer

EEPROM slave address

The EEPROM has the slave address 0x50. Because the LSB is reserved for the R/W indicator the address is shifted left by one: *0xA0*.

Writing to the EEPROM

See the following byte sequence for writing a *0x55* to memory address 0 of the EEPROM:

```
Write per socket to Netzer: 0xA0 0x5C 0x00 0x55 0x00
Answer from EEPROM received from Netzer: 0xFF 0xFF 0xFF 0x00
```

1. *0xA0* is the slave address with the R/W bit cleared, means write access.
2. Netzer returns *0xFF*, indicating the available slave.
3. *0x5C 0x00* transfers the memory address 0 - of course must be backslashed (' ' = 0x5C)
4. Netzer returns *0xFF*, successful transfered memory address.
5. *0x55* is the value written to EEPROM
6. Netzer returns *0xFF*, successful transfered value.
7. *0x00* stops the transmission - I2C bus is released.
8. Socket sends *0x00* to the host - Frame termination

Reading from EEPROM

```
Write per TCP/IP to Netzer: 0xA0 0x5C 0x00 0x73 0xA1 0xFF 0x00
Answer from EEPROM received per TCP/IP from Netzer: 0xFF 0xFF 0xFF 0xFF 0x55
0x78 0x00
```

1. *0xA0* is the slave address with the R/W bit cleared, means write access.
2. Netzer returns *0xFF*, indicating the available slave.
3. *0x5C 0x00* transfers the memory address 0 - of course must be backslashed (' ' = 0x5C)
4. Netzer returns *0xFF*, successful transfered memory address.
5. *0x73* initiates a repeated start on the bus.
6. Netzer returns *0xFF*, successful repeated start.
7. *0xA1* is the slave address with the R/W bit set, means read access.
8. Netzer returns *0xFF*, indicating the available slave.
9. *0xFF* from host: Pull me one byte from slave and acknowledge it.
10. Netzer returns the value read from memory address 0: *0x55*.
11. *0x00* from host: Pull me one more byte from slave and stop.
12. Netzer returns the value read from memory address 1 (auto increment feature of the EEPROM): 0x78.
13. Netzer automatically announces a stop condition after sending the byte.
14. Socket sends *0x00* to the host - Frame termination

From:
http://www.mobacon.de/dokuwiki/ - **MoBaCon**

Permanent link:
http://www.mobacon.de/dokuwiki/doku.php?id=en:netzer:i2c-master&rev=1419159177

Last update: **2025/06/11 20:42**