# MSC I2C

## RHProxy Interface

for

## Windows 10 (x64) / Windows 11 (x64)

# Users Guide

**V1.01**

**DLL Version 1.0.1.0**

**June 7th, 2023**

AVNET EMBEDDED

Avnet Embedded GmbH
Ludwig-Erhard-Str. 10c
D-85375 Neufahrn

# Preface

## Copyright Notice

## Important Information

This documentation is intended for qualified audience only. The product described herein is not an end user product. It was developed and manufactured for further processing by trained personnel.

## Disclaimer

Although this document has been generated with the utmost care no warranty or liability for correctness or suitability for any particular purpose is implied. The information in this document is provided "as is" and is subject to change without notice.

## Trademarks

All used product names, logos or trademarks are property of their respective owners.

## Certification

Avnet Embedded GmbH is certified according to DIN EN ISO 9001:2015 standards.

## Life-Cycle-Management

Avnet Embedded GmbH products are developed and manufactured according to high quality standards. Our life-cycle-management assures long term availability through permanent product maintenance. Technically necessary changes and improvements are introduced if applicable. A product-change-notification and end-of-life management process assures early information of our customers.

## Product Support

Avnet Embedded GmbH engineers and technicians are committed to provide support to our customers whenever needed.

Before contacting Technical Support of Avnet Embedded GmbH, please consult the respective pages on our web site at embedded.avnet.com/support for the latest documentation, drivers and software downloads.

If the information provided there does not solve your problem, please contact our Avnet Embedded GmbH Technical Support team as follows:

| | |
|---|---|
| Phone: | +49 8165 906 - 200 |
| Email: | support.boards@avnet.eu |

## Content

# 1 General Information

## 1.1 Revision History

| Rev. | Date | Description |
|---|---|---|
| X1.00a | April 9th, 2019 | first version |
| X1.00b | November 25th, 2019 | Same functionality as X1.00a but generated with VS 2019. |
| X1.00c | February 26th, 2020 | Same functionality as X1.00b, only PRODUCTVERSION adjusted to EAPI release X6.00c and FILEVERSION incremented. |
| X1.00d | November 4th, 2020 | Same functionality as X1.00c, only PRODUCTVERSION adjusted to EAPI release X6.00d and FILEVERSION incremented. |
| X1.00e | November 27th, 2020 | Same functionality as X1.00d, only PRODUCTVERSION adjusted to EAPI release X6.00e and FILEVERSION incremented. |
| V1.00 | May 24th, 2021 | Same functionality as X1.00e, only PRODUCTVERSION adjusted to EAPI release V6.00 and FILEVERSION adjusted for driver release version V1.00. |
| X1.01a | September 13th, 2022 | From a Win32 app it was not possible to read data from the I2C bus. However, a "scan" of the I2C bus was successful and the available I2C devices were reported. |
| X1.01b | September 29th, 2022 | Same functionality as X1.01a, only PRODUCTVERSION adjusted to EAPI release X6.01b and FILEVERSION adjusted for driver release version X1.01b. |
| X1.01c | October 24th, 2022 | Same functionality as X1.01b, only PRODUCTVERSION adjusted to EAPI release X6.01c and FILEVERSION adjusted for driver release version X1.01c. |
| X1.01d | April 11th, 2023 | Same functionality as X1.01c.<br>In the header file 'msci2c.h' the names of the command/offset parameters of the commands read1, read2, write1 and write2 have been changed, to make it clear how they are used.<br>PRODUCTVERSION adjusted to EAPI release X6.01d and FILEVERSION adjusted for driver release version X1.01d. |
| V1.01 | June 7th, 2023 | Same functionality as X1.01d, only PRODUCTVERSION adjusted to EAPI release V6.01 and FILEVERSION adjusted for driver release version V1.01. |

## 1.2 Introduction

To allow I2C accesses under Windows 10/11 Microsoft introduced the “Simple Peripheral Bus” interface driver model (SPB), see:

https://docs.microsoft.com/en-us/windows-hardware/design/component-guidelines/simple-peripheral-bus--spb-

This driver model allows to write a peripheral driver for a specific I2C device.

With Windows 10 version 1709 Microsoft introduced the RHProxy driver, which allows SPB accesses directly from user mode, see:

https://docs.microsoft.com/de-de/windows/uwp/devices-sensors/enable-usermode-access

With this method you can access the I2C busses directly from your user mode application.

To simplify the I2C access over the RHProxy driver Avnet Embedded GmbH provides the interface DLL (MSCI2C64.DLL) with easy-to-use I2C read/write functions.

# 2 Technical Description

## 2.1 Setup Procedure

There is no specific installation procedure for the MSCI2C64.DLL. Just copy the provided MSCI2C.ZIP to a folder, where the application can find the DLL and unpack the ZIP file.

Prerequisites:

- Windows 10 (x64) 1709 or later or Windows 11 (x64)
- BIOS version with ACPI definitions which enable the RHProxy driver to be loaded and started. Note, the ACPI definitions support 7-Bit I2C addresses only (no 10-Bit I2C addresses).
- BIOS Setup: User I2C Support -> Controller-based
- An I2C controller driver provided by the SOC hardware vendor

## 2.2 Using the MSCI2C64 DLL

Using a Microsoft development environment your application links against the x64 import library ‘MSCI2C64.LIB’ and includes the header ‘MSCI2C.H’. Then you can call the interface functions declared in the header file.

It must be ensured that the ‘MSCI2C64.DLL’ resides in a folder, where it can be found by the application.

It is also possible to use an x86 application. In this case you must link against the x86 library ‘MSCI2C.LIB’ which results in using the x86 DLL MSCI2C.DLL.

## 2.3 Code samples (error handling omitted)

### 2.3.1 Get information about the available I2C busses

```
// get information about the available I2C busses:

unsigned int nbrI2CControllers;

char *pI2CBusses = ListI2Controllers(&nbrI2CControllers);
```

Search through the returned I2C bus names for the I2C bus name you like to access.
Return an error if you can't find the desired I2C bus name.
Else you can use the I2C bus name as first parameter of the following read/write functions.

### 2.3.2 Read from I2C devices (8 bit offset)

```
// read 128 bytes from the "user" I2C bus, I2C device address 0x57, starting at device offset 0x10:

bool ret;
unsigned int i2cAddr = 0x57;     // 7 bit I2C address (without read/write bit)
insigned int i2cSpeed = 400;     // I2C speed (100 or 400)
unsigned int offset = 0x10;      // offset within the device
unsigned int nbrBytes = 128      // number of bytes to read
UCHAR readBuf[128];              // the read buffer
unsigned int bytesRead;          // number of bytes read (returned by read1())

ret = read1("user", i2cAddr, i2cSpeed, offset, nbrBytes, readBuf, &bytesRead);
```

Check the return code and compare bytesRead with nbrBytes….

### 2.3.3 Read from I2C devices (16 bit offset)

```
// read 128 bytes from the "user" I2C bus, I2C device address 0x56, starting at device offset 0x210:

bool ret;
unsigned int i2cAddr = 0x56;     // 7 bit I2C address (without read/write bit)
insigned int i2cSpeed = 400;     // I2C speed (100 or 400)
unsigned int offset = 0x210;     // offset within the device
unsigned int nbrBytes = 128      // number of bytes to read
UCHAR readBuf[128];              // the read buffer
unsigned int bytesRead;          // number of bytes read (returned by read2())
```

*unsigned int cmdOffs_MSB;*
*unsigned int cmdOffs_LSB;*

*cmdOffs_MSB = (offset & 0xff00) >> 8;*
*cmdOffs_LSB = offset & 0xff;*

*ret = read2("user", i2cAddr, i2cSpeed, cmdOffs_MSB, cmdOffs_LSB, nbrBytes, readBuf, &bytesRead);*

Check the return code and compare bytesRead with nbrBytes….