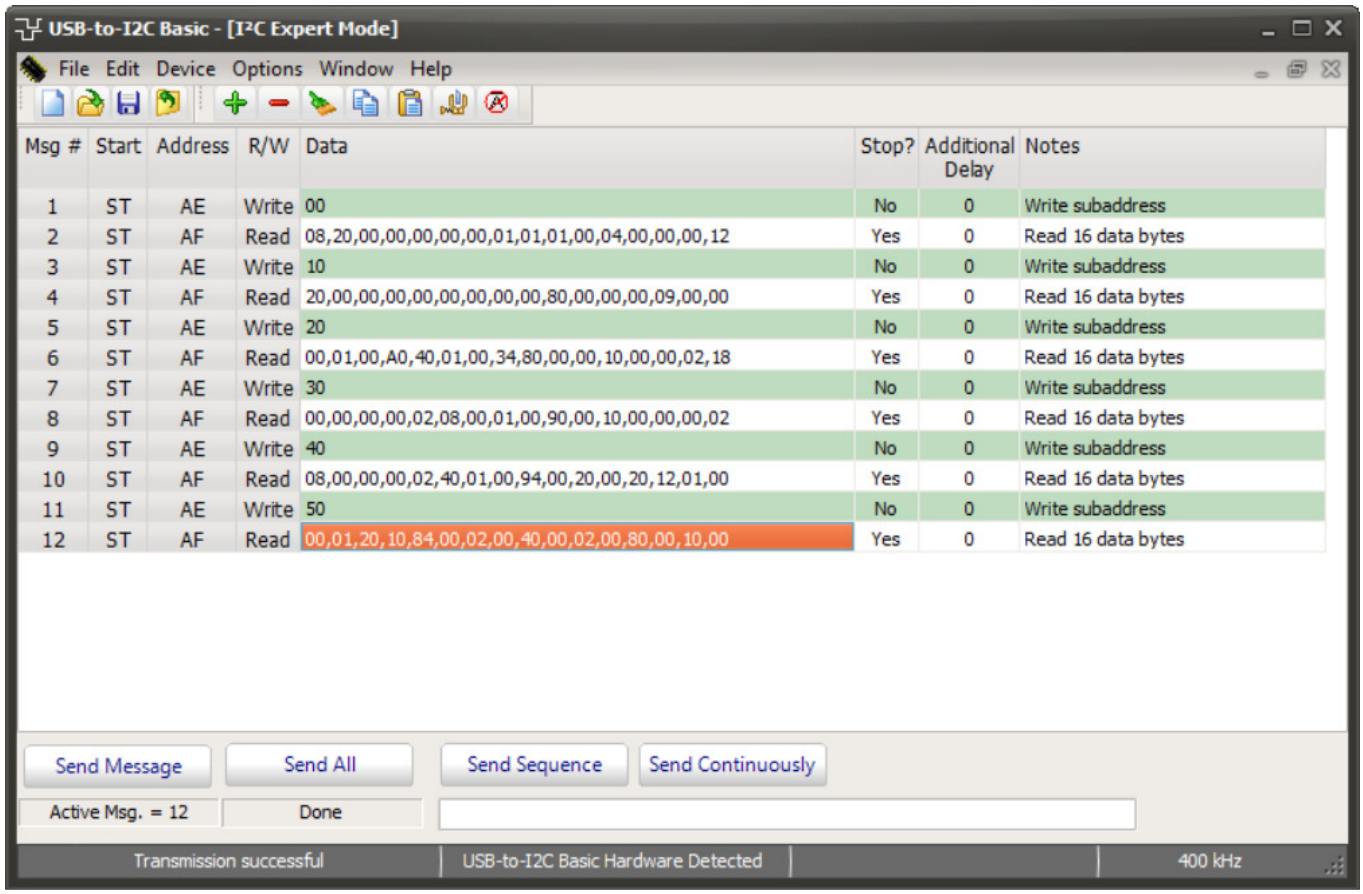
USB-to-I2C©

Software User's Manual for USB-to-I2C Basic



Information provided in this document is solely for use with USB-to-I2C Basic hardware. SB Solutions, Inc. reserves the right to make changes or improvements to this document at any time without notice. We assume no liability whatsoever in the sale or use of this product, including infringement of any patent or copyright. No part of this document may be reproduced or transmitted in any form or by any means, electronic or mechanical, for any purpose, without the express written permission of SB Solutions, Inc.

Microsoft and Windows are registered trademarks of Microsoft Corporation.

Other brand names are trademarks or registered trademarks of their respective owners.

Questions or comments regarding this document should be emailed to support@i2ctools.com

 October 2016

TABLE OF CONTENTS

I²C Protocol

General Characteristics

The I²C protocol allows data to be transferred between devices using two open-drain (or open-collector) bi-directional lines. One line is the serial clock (SCL) and the other is the serial data (SDA). The bus master generates the Start conditions, the clock signals on SCL, as well as the Stop condition. An acknowledge is transmitted on the bus after each byte is sent over the bus.

Bit Transfer

Data on SDA must be stable while SCL is high. The state of SDA when SCL is high determines the logic level of the transmitted data bit.

START and STOP Conditions

Within the procedure of the I²C bus, unique situations arise which are defined as START and STOP conditions. A HIGH to LOW transition on the SDA line while SCL is HIGH is one such unique case. This situation indicates a START condition. A LOW to HIGH transition on the SDA line while SCL is HIGH defines a STOP condition. The master always generates START and STOP conditions. The bus is considered to be busy after the START condition. The bus is considered to be free again a certain time after the STOP condition.

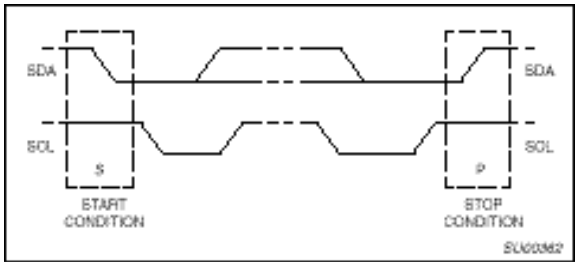


Figure 1. Start and Stop Conditions

I²C Address

The USB-to-I2C software assumes that the I2C address is 8 bits in length. Some manufacturers of I2C devices may use a 7-bit address in their documentation, so you should shift the address of these devices to the left by one bit.

The first seven bits of an I²C transmission make up the slave address. The eighth bit (or the least significant bit) is the R/W bit that determines the direction of the message.

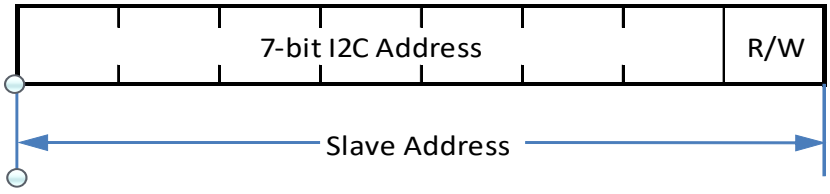


Figure 2. I2C Slave Address

A '0' in the least significant position of the first byte means that the master will WRITE information to the selected slave. A '1' in this position means that the master will READ information from the slave.

When an I²C address is sent, each device in a system compares the first seven bits after the START condition with its own address. If they match, the device considers itself addressed by the master as a slave-receiver or

slave-transmitter, depending on the R/W bit.
When selecting addresses within USB-to-I2C, the software assumes the least significant bit is zero (write). If the I^2C message is a write transmission, the least significant bit will be sent as a '0' while if it is a read, the software will append a '1' in the LSB position.

I^2C Bus Documentation

The complete I^2C Bus specification can be found at
http://www.nxp.com/documents/user_manual/UM10204.pdf

Main Screen

This user manual covers the features of the USB-to-I2C Basic software.
When the USB-to-I2C program starts, a screen, as shown below, will be displayed on the monitor.

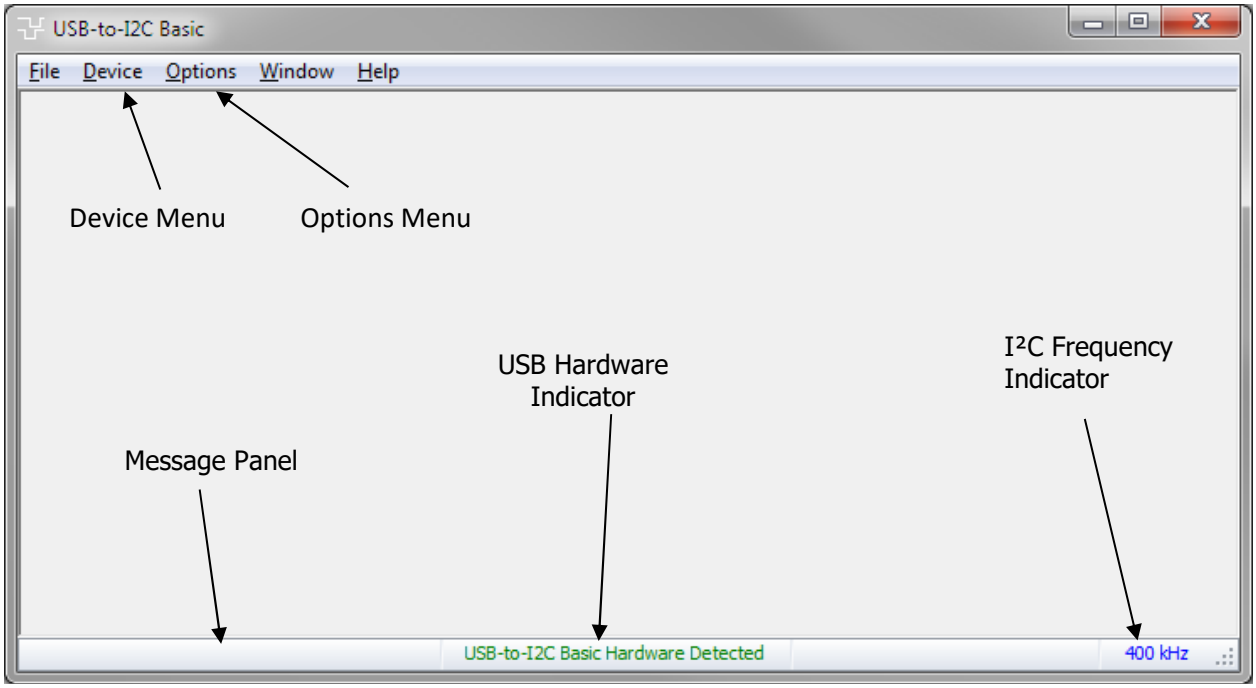


Figure 3. USB-to-I2C Basic overview

Device Menu

The device menu contains a list of I^2C devices supported by the USB-to-I2C software. Selecting the device from this menu may start any of the listed devices. You can have any combination of devices open at one time. Switching between active devices may be accomplished via the Window menu on the main toolbar.

Message Panel

The main screen has a panel that displays messages from the program. It will indicate if the I^2C transmission was successful or if there was a problem encountered. A list of messages is shown below.

Messages:

Transmission successful - the last I^2C transmission was successfully completed.
Address not acknowledged - an I^2C address was successfully transmitted but no slave device acknowledged the address. A STOP condition is sent after the acknowledge clock pulse if no acknowledge is received.
Data not acknowledged - an I^2C address was previously acknowledged but one of the following data bytes was not acknowledged. A STOP condition is sent after the acknowledge clock pulse if no acknowledge is received.
Time-out – the hardware was unable to complete sending the I2C message, usually due to a signal line stuck low.
Hardware not detected- is displayed when there is no USB-to-I2C hardware plugged into the PC's USB port, or if the drivers are not installed correctly.

File Menu

Upon starting the USB-to-I2C Basic software, the File menu contains the Exit and Close commands. When a

device has been selected from the Device Menu, it is possible that the File Menu will also display device specific commands such as “Save As” and “Load”. In User Device mode, previously created device files may be conveniently loaded.

Save Data

Many devices contain the menu item 'Save Data' under the File menu. The data may be recalled by selecting the Load Data item under the File Menu.

Load Data

After data has been stored using the 'Save Data' item in the File menu, it can be recalled by selecting the Load Data item.

Options Menu

The options menu allows you to change the I²C frequency. It also allows the user to reset the I2C bus that has one of the signals stuck low. The reset consists of sending nine clock pulses followed by a Stop condition. If a slave device has become unsynchronized due to noise on the bus, this may help free the bus.

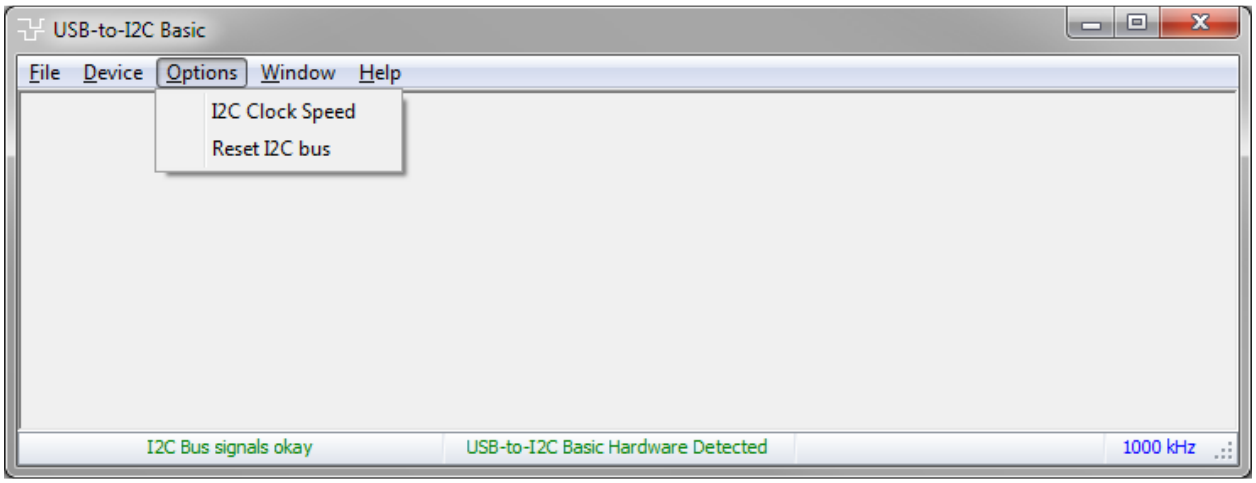


Figure 4. Options Menu

I²C Frequency Menu Item

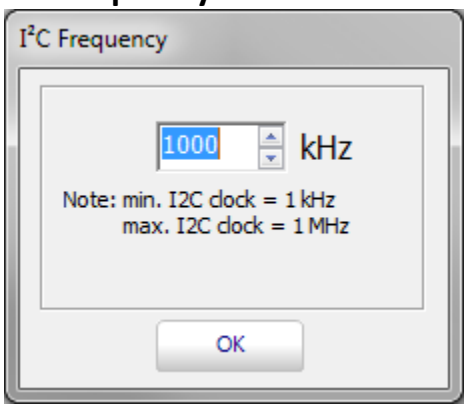


Figure 5. I²C Frequency Selection

The I²C frequency dialog box can be activated by:

a. Selecting I²C Frequency in the Options menu
b. Double-clicking the frequency indicator on the bottom right corner of the application

By choosing this menu item, you will activate a dialog box that shows the current I²C frequency.

The frequency can be changed by entering a value into the edit box or by scrolling the up/down buttons. Pressing the OK button will close the dialog box and will update the I²C frequency panel on the main screen.

Note that a desired frequency may not be available. Therefore, the software will choose the frequency closest to the desired frequency.

Windows Menu

The Windows Menu contains screen commands such as cascade, tile, arrange all icons, and minimize all. If

there are devices active in the program, you will find them listed in this menu. When multiple device types are open, it is easy to move between the device types by clicking on the desired item in this menu.

Frequency Indicator

The frequency at which the hardware is sending I^2C messages over the i2c bus is shown in this box on the main screen. Note that the hardware cannot produce every value you enter so it will set it to the closest available frequency.

Expert Mode

The figure below shows the Expert Mode screen.

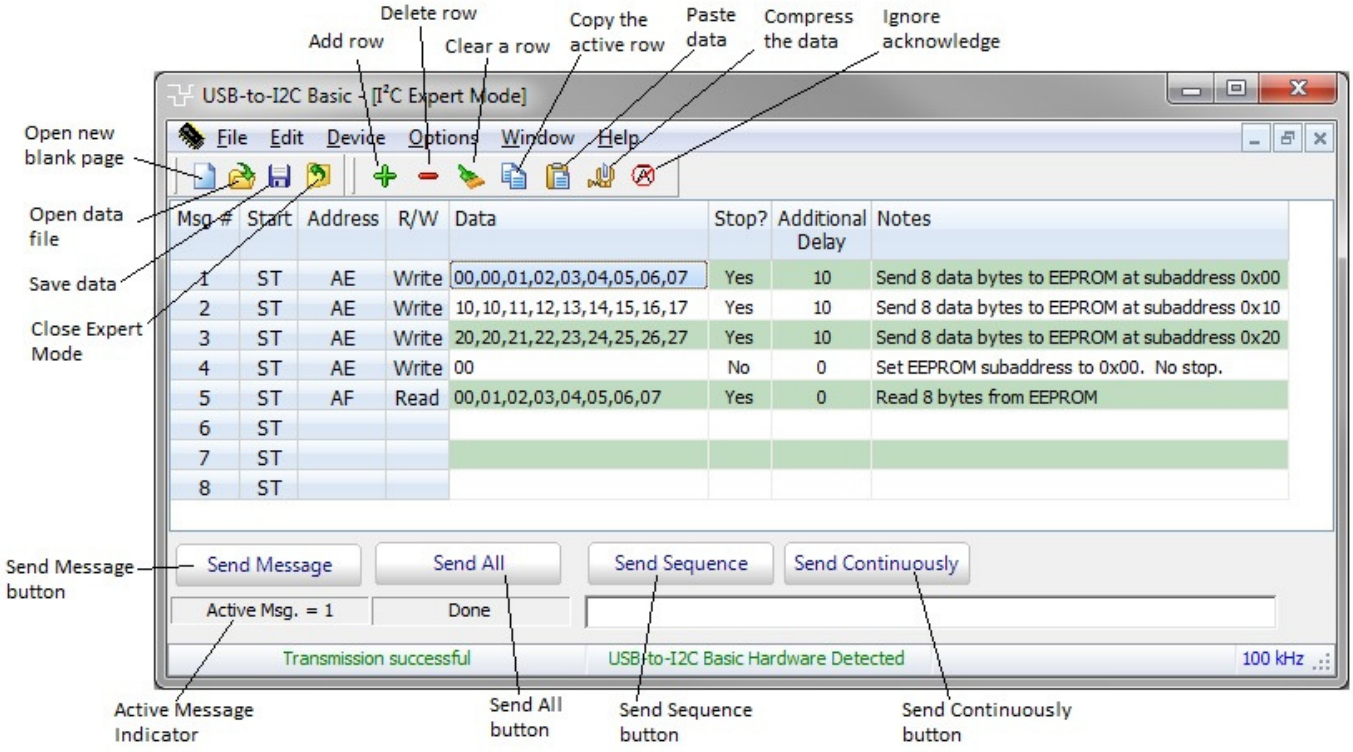


Figure 6. Expert Mode Overview

Open New Page

Pressing this button opens a new blank page. There will be 8 empty rows (messages). Selecting 'New' from the 'File' menu while the Expert Mode is active will perform the same function.

Open Data File

A previously saved data file can be recalled by pressing the Open Data File button or by selecting Open from the File menu while the Expert Mode is active. A dialog box will be displayed allowing the user to navigate to the appropriate directory.

Save Data

The current data will be saved when this button is pressed. The user specifies the name and location of the file in a dialog box that is displayed after the button is pressed. A dialog box will be displayed which allows the user to navigate to the appropriate directory.
The user can also perform the same function by selecting Save from the File menu while the Expert Mode screen is active.

Close Expert Mode

The Expert Mode screen is closed but USB-to-I2C Basic will not be terminated.

Add a Row

Inserts a new (blank) row after the active row. You can also use the Ctrl+Insert keyboard shortcut to insert a new row. The total number of rows is limited to 64.

Delete a Row

Deletes the Active Row (active message). You can also use the Ctrl+Del keyboard shortcut to delete the current row.

Clear the current row

The current row (message) will be cleared. The row will not be deleted but will appear blank.
You can also use the Shift+Del keyboard shortcut to clear the current row.

Copy the Current Row

The current row (message) will be copied. Use the Paste command to paste it to a different row.
You can also use the Windows Ctrl+C keyboard shortcut to copy the current row.

Paste Data

Previously copied data will be pasted into the current row (message).
You can also use the standard Windows Ctrl+V keyboard shortcut to paste the clipboard into the current row.

Compress Data

All blank rows will be eliminated from the display. Here is an example of a display before compress:

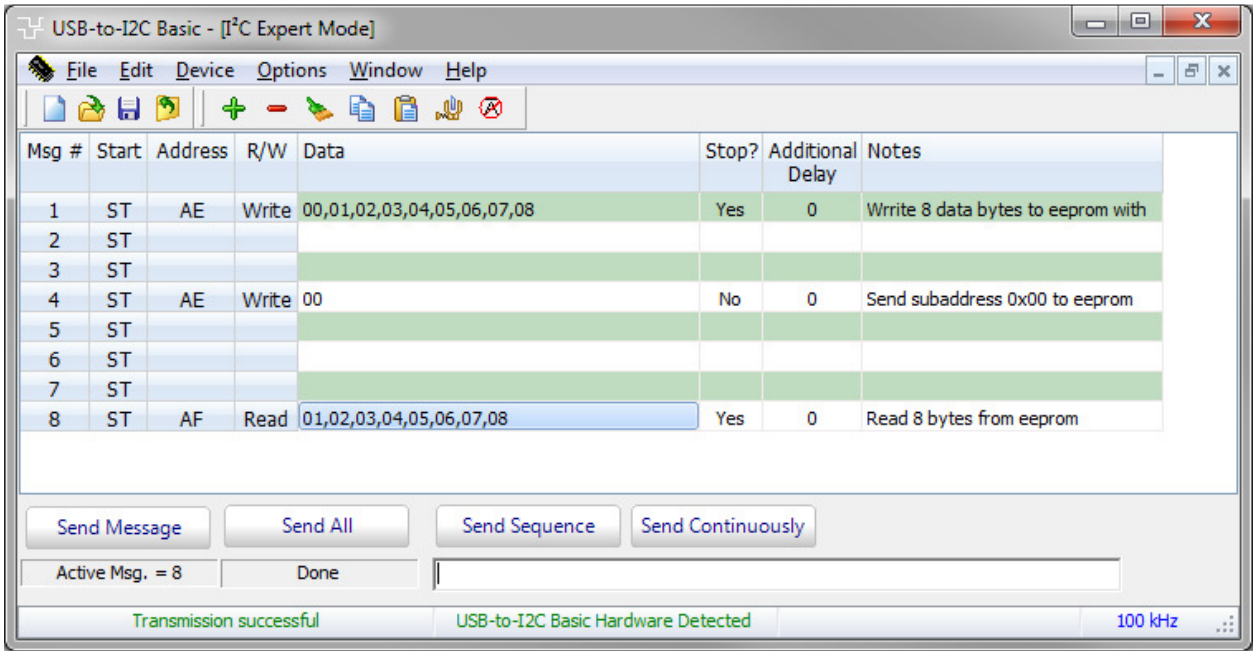


Figure 7. Expert mode before compress

And here is the same screen after the compress:

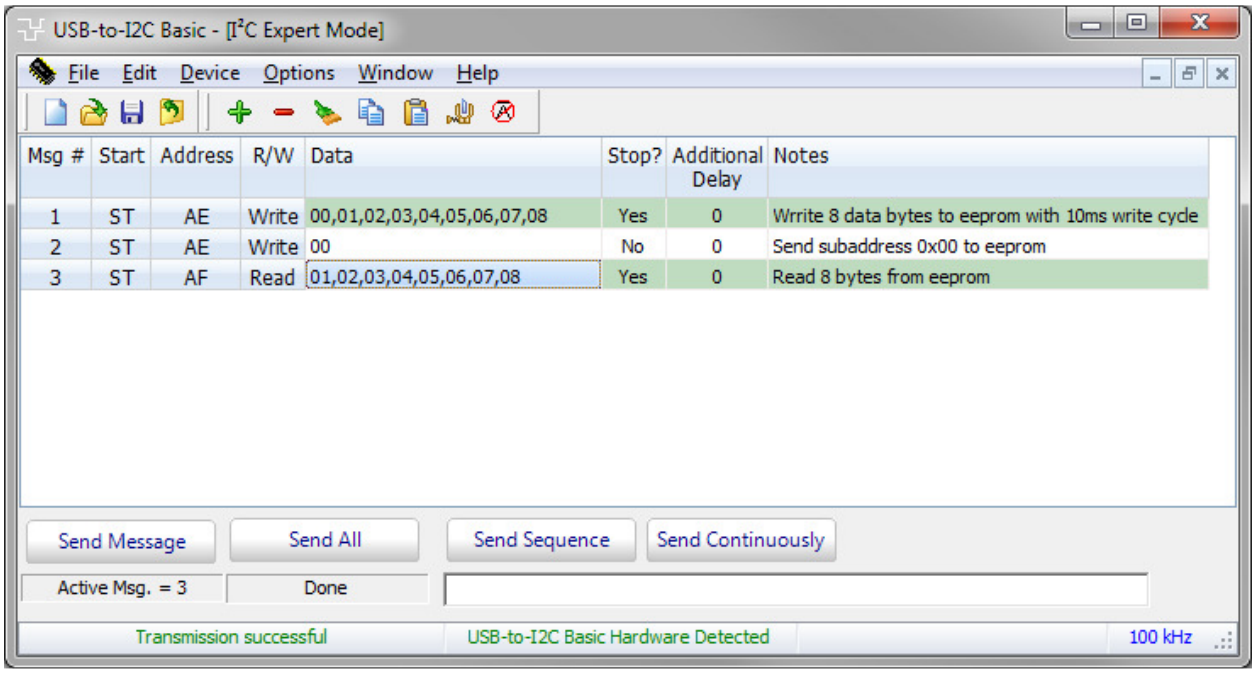


Figure 8. Expert mode after compress

It is not required to perform a compress but it does speed up the message transfer process since the application does not need to evaluate blank rows to see if there is data to be sent.

Send Message

The current message will be sent when this button is pressed. The current message number is shown below the Send Message button. To change the active message to be sent, single-click on the row of the data to be sent.

Send All

All the valid messages on the screen will be sent in order of the row number. The action will be performed one time. A message is valid if there is a minimum of an address within the message.
Since the program tests for a valid message on each line within the message grid before sending the message, it is recommended (not required) to compress the data (see Compress Data above) to speed up the transfer.

Send Sequence

A sequence of messages will be sent when the Send Sequence button is pressed.
The sequence editor is invoked by double-clicking on the sequence display. The sequence length can be up to 64 messages in length. The sequencer is limited to using messages 1 through 64.
The Sequence Editor is shown below.

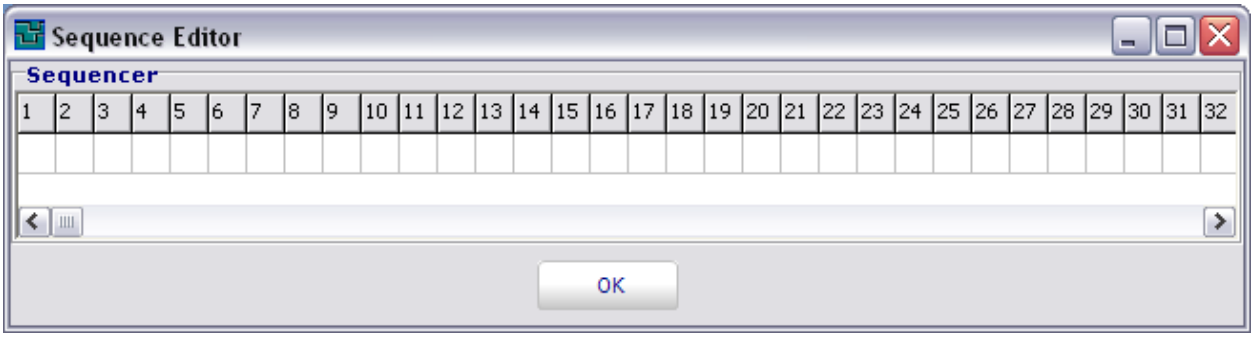


Figure 9. Sequence Editor

Message Editor

The I2C message cannot be edited directly in the Expert mode screen. Instead, an Expert Mode Editor is brought up either by double-clicking on a message or when the user attempts to type directly into one of the rows (messages) in the Expert Mode screen.

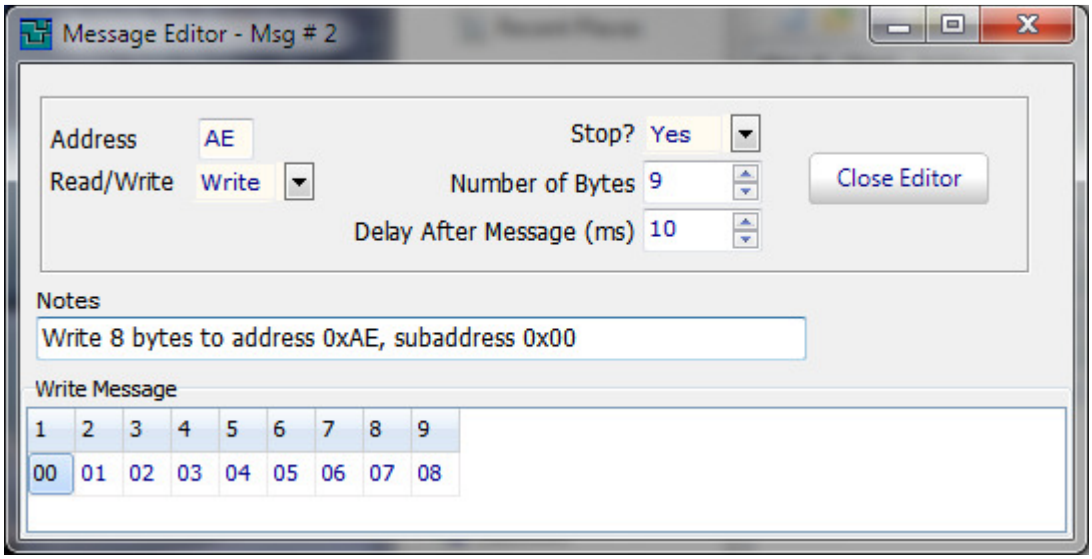


Figure 10. Message Editor

Message Number

The message being edited is shown at the top of the message editor screen.

Delay after message

A delay, measured in milliseconds, can be inserted after a message.

Device Address

The I2C slave address is entered in the address box in hexadecimal notation. The least significant bit of the address is not important (can be a '1' or '0') since the Expert Mode will ensure that this is appropriate for the read/write transaction when the message is actually transmitted. Note that this is an 8-bit address value (see I2C Protocol section for details).

Read/Write Selection

The user can select a Read or Write transaction from the drop-down selection box. If a Read is chosen, then the 'Number of Bytes to Read' box will be shown and the data entry area will be hidden. If a Write is chosen, the 'Number of Bytes to Read' will be hidden and the data entry area will be shown.

Stop?

Sending a Stop condition after a message is optional. Normally, it is advisable to send the Stop condition. If a Stop condition is not sent, the clock line will be held low until the next message is sent. If a Stop is not sent, the next message will begin with a Restart condition rather than a Start condition.

Message Data

The Message Data area contains the location where the user can enter up to 49 data bytes in hexadecimal format. Blank data bytes will be ignored.

Notes

The information in the Notes section is optional, and is not used by the USB-to-I2C Basic software.

User Definable Device

The User Definable Device allows you to define your own I²C device and then enables the user to change the values of the individual cells within the grid using various controls such as sliders and spin controls.

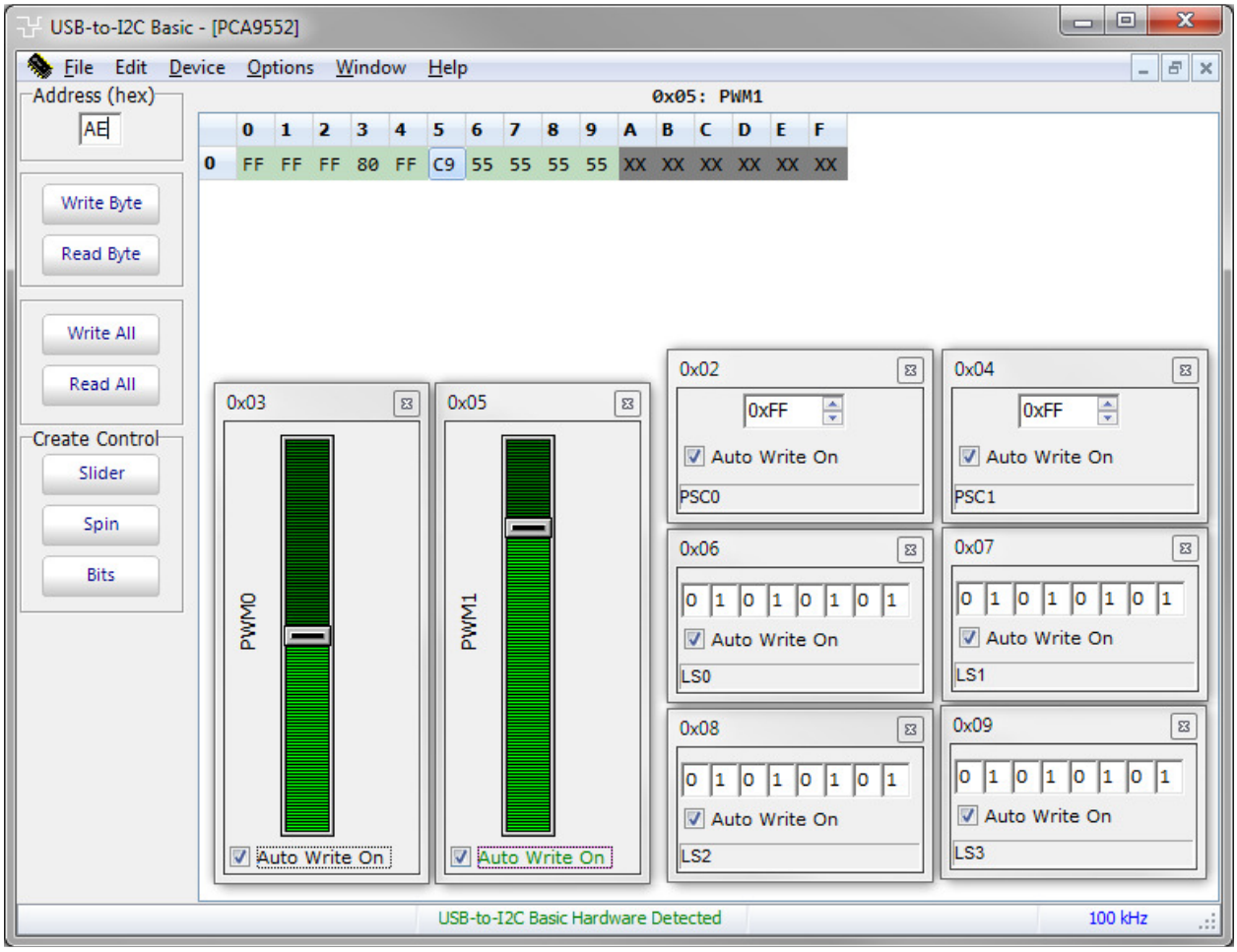


Figure 11. User Definable Mode

When the User Definable Device is first opened, a 16-byte device grid is shown on the screen but does not have any names associated with the data, and all the data bytes are set to 0xFF. A previously defined device may be loaded by selecting it from the Most Recently Used files at the bottom of the File menu. If devices have not yet been defined, there will be no devices shown below the File/Exit menu item.

To begin the definition process, you may right click with your mouse on any data cell in the grid or select 'Current Device' or 'Current Cell' from the Edit menu. The Reset item in the Edit menu will clear all the register names to 'Undefined Register' and set the data values to 0xFF.

Clicking on the desired cell and doing one of the following can change the data value of the individual cells:

1. Typing in a two-digit hexadecimal number.
2. Assigning a slider to the active cell by pressing the Slider button.
3. Assigning a spin control to the active cell by pressing the Spinner button.
4. Assigning a Bit-wise control to the active cell by pressing the Bit-wise button.

Items 2, 3, and 4 can also be achieved by positioning the cursor over the desired cell and right clicking on the grid to select the appropriate control from the pop-up menu.

Define New Device

This is similar to the Edit Current Device (explained below). This menu selection allows the user to start a new device from scratch. All register names are undefined and all default values are 0xFF.

Open Device Definition File

A previously saved device definition file (.def) can be recalled by selecting this menu item. The device definition file contains the Device Name, Device Address, Register Names, and Register Values.

Save

Device definition files can be saved to disk by selecting the Save menu item.

Save As…

Use the Save As dialog box to change the definition file name or to save the definition file in a new location. If the file name already exists, USB-to-I2C asks if you want to replace the existing file.

Save Registers in text format

The device definition files are not in a format that can easily be used by the user, therefore, USB-to-I2C allows you to save the information in a text format (extension .txt). Users can then open and edit this file with any word processor such as Notepad or WordPad. The text files are for the user's information only and cannot be read by USB-to-I2C.

Print Device Data

A print out of the register definitions can be obtained by selecting this option from the File menu.

Data Grid

The grid consists of rows and columns. Each cell within the grid contains a two-digit hexadecimal number. Each cell corresponds to a physical byte location within the I^2C device. For example, in the diagram shown above, cell 0x07 is highlighted (row 0, column 7). This translates to address 7 (decimal) in the device (assuming the first byte is address 0x00). The data may be changed by entering hexadecimal numbers from your keyboard. Non-valid keys will be ignored. In order to edit the entire grid, including the name and default values of the registers, you may right click on the grid or select the appropriate item from the Edit menu.

The individual cells within the grid will be blue if the cell's subaddress is greater than the maximum number of registers defined for the active device. The number of device registers may be changed at any time in the 'Edit Current Device' screen.

Edit Menu

The Edit menu is available only when the User Definable Device screen is active.

The user can select from one of the three menu items:

1. Current Device - brings up a screen showing the all the register data for the active device.
2. Current register - allows registers to be changed one at a time.
3. Reset grid - all register data will be set to 0xFF and the register descriptions will be 'Undefined Register'

Edit Current Register

If you want to adjust one register in the grid, use the 'Edit Current Register' screen. This screen can be started by right clicking on the User Definable Device grid or by selecting Current Register from the Edit menu. The name of the register and the initial value displayed, when the definition file is first opened, can be changed here. Note that the register name changes are not saved until the 'Save' or 'Save As.. .' item (under the File menu) is selected.

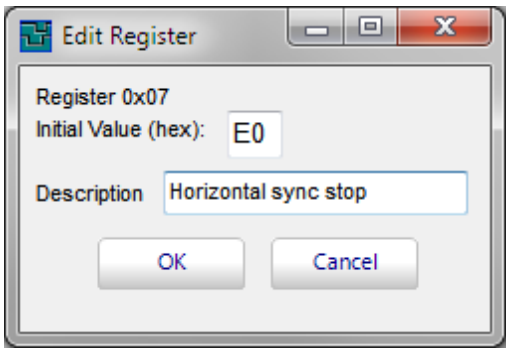


Figure 12. Edit Current Register

Edit Current Device

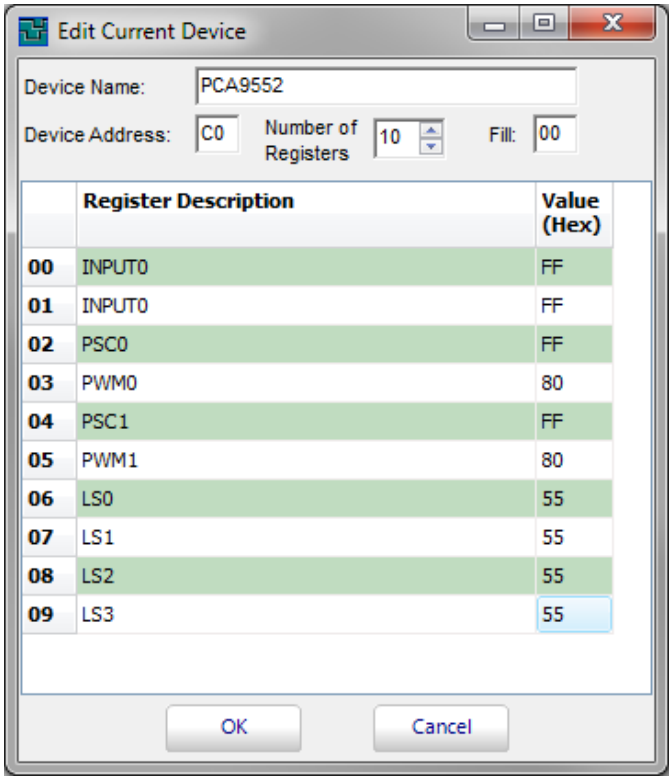


Figure 13. Edit Current Device

Editing the current device may be accomplished by clicking on 'Edit Current Device' from the Edit menu, or by right clicking on the grid within the User Definable Device mode of USB-to-I2C. The screen, shown above will be displayed when either method is invoked.

Device Name: the name entered in this box will be shown in the title bar of USB-to-I2C when the definition file is opened.

Device Address: is the device I^2C address that will be displayed in the Address box when the definition file is opened. Note that only even addresses are valid here. USB-to-I2C will append the appropriate R/W bit at the end of the address depending upon the operation to be performed (the last bit will be a '1' if it is a read operation and a '0' if it is a write).

Number of Registers: Enter the number of registers the device contains. The size of the data entry grid will be modified to accommodate the number of registers.

Fill: This box should be modified only if you want to initialize all the registers to one particular value.

OK button: the data entered by the user in the 'Edit Current Device' screen will be transferred to the User Definable Device screen.

Cancel button: the editing session will be closed and no changes to the User Definable Device screen will occur.

Note that any changes are not saved until the 'Save' or 'Save As…' menu item is selected.

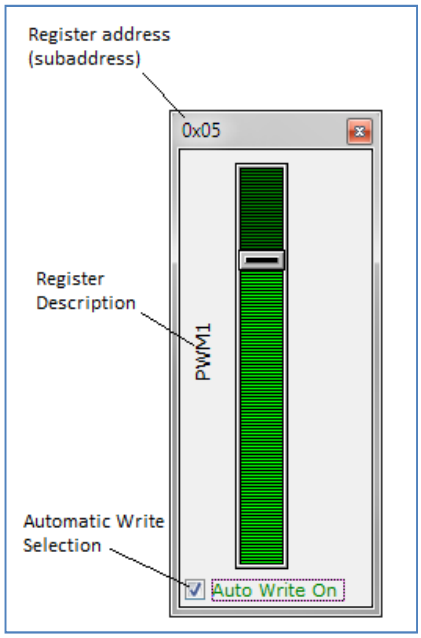


Figure 14. Slider Control

Slider Control

The Slider Control is activated when the user presses the Slider button on the User Defined Device screen or by right clicking the grid and then selecting 'Change Active Register with Slider Control' from the pop-up menu.

The subaddress of the active cell in the grid will be assigned to the Slider. The subaddress is shown in the upper left corner. Moving the slider bar up and down will cause the value of that cell to be changed. If Auto Write On is checked, then the contents of the cell will be transmitted to the device subaddress when it is changed. If an error is encountered while transmitting using Auto Write, USB-to-I2C will turn off Auto Write and it will be up to the user to fix the error condition and re-enable Auto Write.

It should be noted that the Slider Control always stays on top of all other devices within USB-to-I2C.

Spin Control

The Spin Control is activated when the user presses the Spinner button on the User Defined Device screen or by right clicking the grid and then selecting 'Increment/Decrement Current Register with a Spin Control' from the pop-up menu.

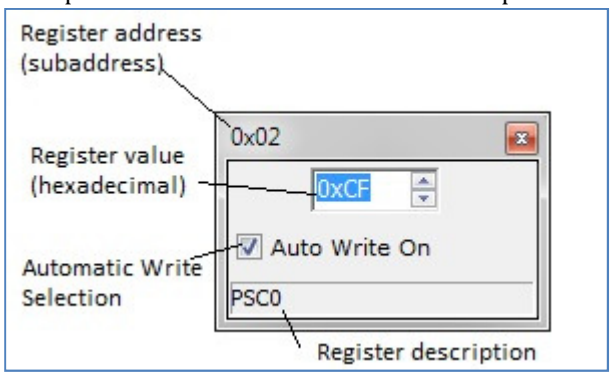


Figure 15. Spin Control

The subaddress of the active cell in the grid will be assigned to the Spin Control. The subaddress is shown in the upper left corner. Clicking on the spin control's up or down arrow will cause the value of that cell to be incremented or decremented. If Auto Write On is checked, then the contents of the cell will be transmitted to the device subaddress when it is changed. . If an error is encountered while transmitting using Auto Write, USB-to-I2C will turn off Auto Write and it will be up to the user to fix the error condition and re-enable Auto Write.

It should be noted that the Spin Control always stays on top of all other devices within USB-to-I2C.

Bit Control

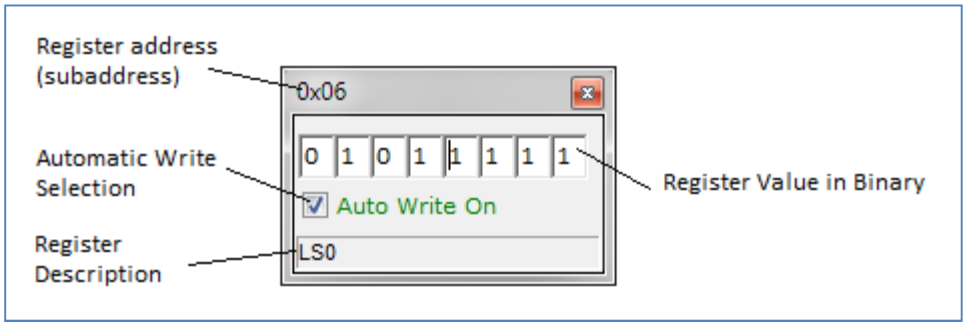


Figure 16. Bit Control

The Bit Control is activated when the user presses the Bit Control button on the User Defined Device screen or by right clicking the grid and then selecting 'Use Bit Control to Change Active Register' from the pop-up menu.

The subaddress of the active register in the grid will be assigned to the Bit Control. The subaddress is shown in the upper left corner. Clicking on any of the eight edit boxes will cause the value of that bit to be inverted. If Auto Write On is checked, then the contents of the cell will be transmitted to the device subaddress when it is changed.

It should be noted that the Bit Control always stays on top of all other devices within USB-to-I2C.

Memory Devices (EEPROM, RAM, FRAM)

USB-to-I2C Basic supports I2C EEPROM memory devices from 128 bit (16 bytes), up to 16kb (2k byte). Upon starting any memory type device, you will see a screen similar to the one shown below.

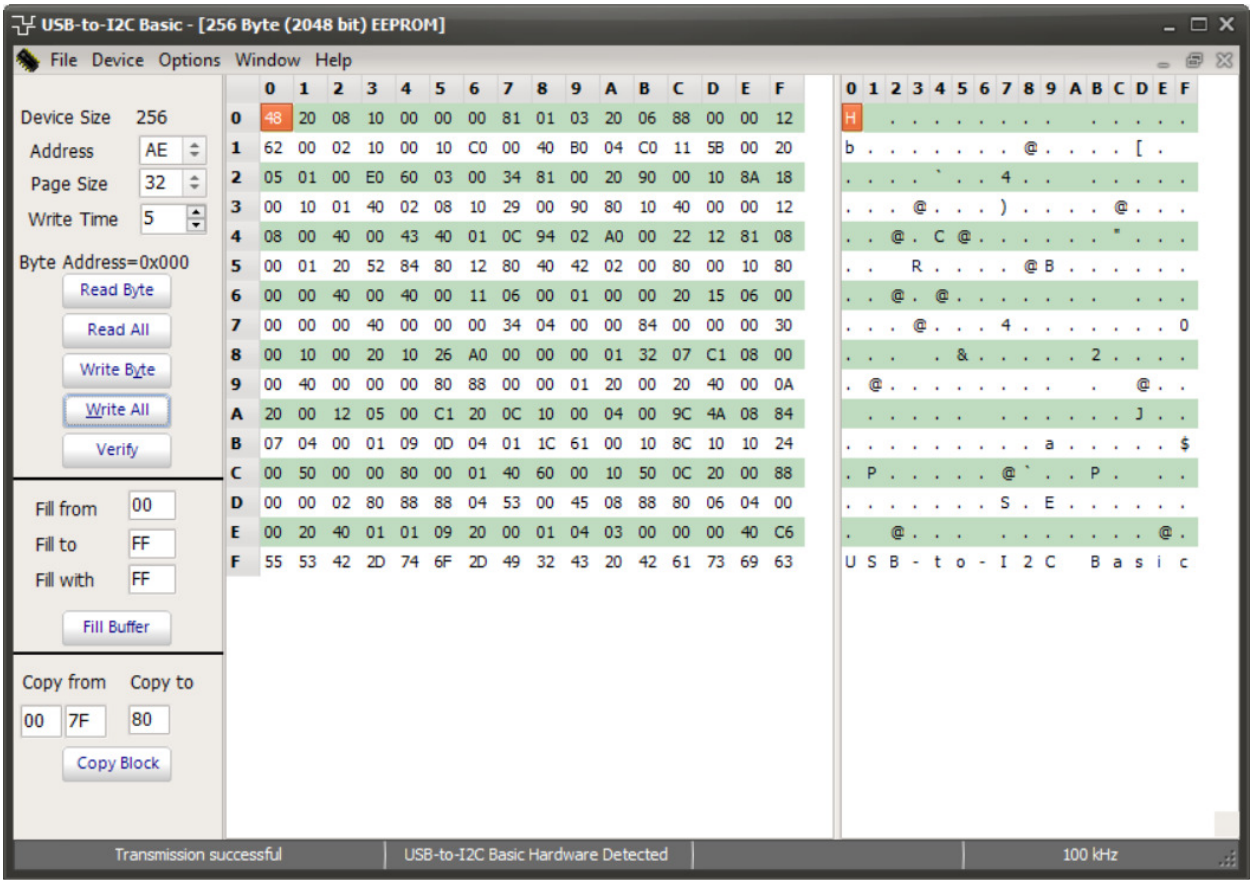


Figure 17. EEPROM Device

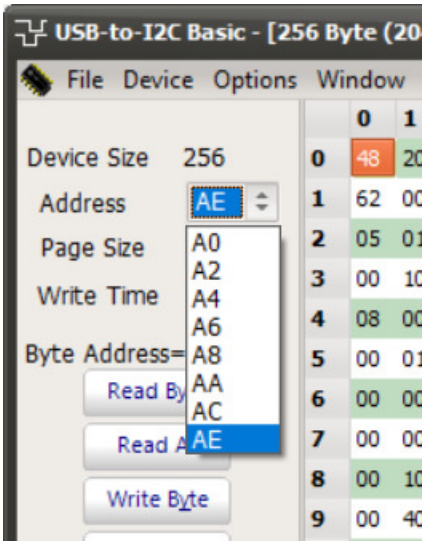


Figure 18. I²C Address Selection

I²C Address

A drop-down menu is provided which allows the user to select a valid address for the selected device type.

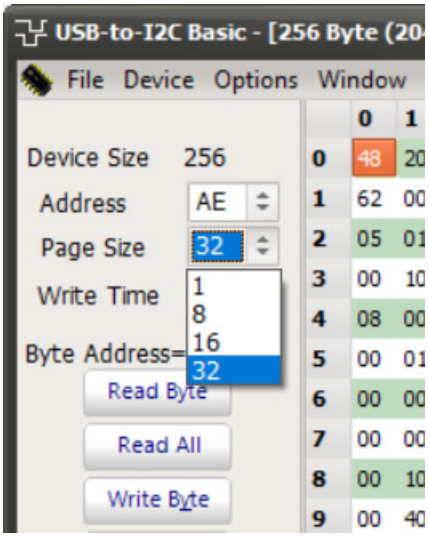
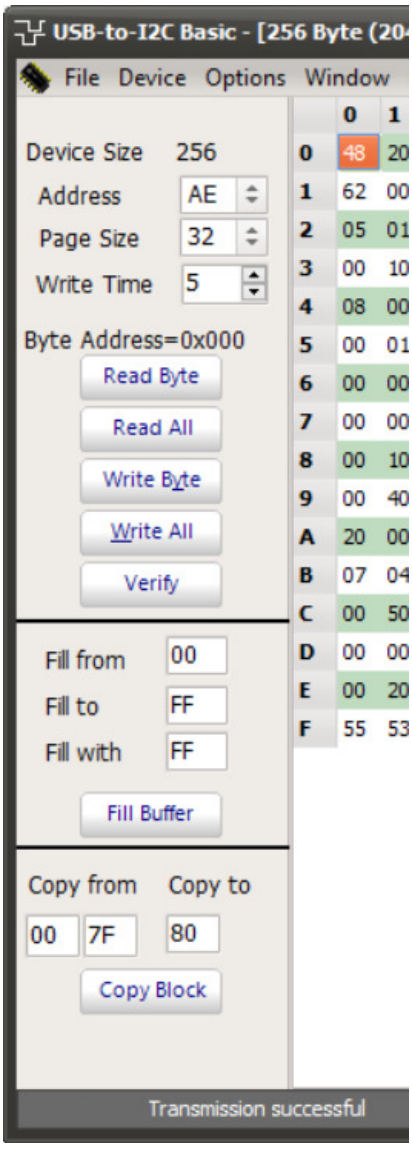


Figure 19. Page Size Selection

Write Page Size Selection

The Page Write Size defines the number of bytes that may be written in a single erase/write programming cycle. Smaller devices generally use an 8 or 16-byte pages but may be up to 128 bytes per page. Check the device datasheet to find the appropriate page write size for the device you are programming. If you do not know the page size, use a small page size such as 8 bytes.



Erase/Write Cycle Time

Programming software must allow a certain period of time to elapse after writing a block of data to an EEPROM. This time is device dependent but is normally between 5ms and 40ms. A STOP condition must be performed before the erase/write cycle commences. Most EEPROMs will not respond to their address until the erase/write cycle has been completed.

Data Grid

The data grid consists of rows and columns. Each cell within the grid contains a two-digit hexadecimal number. Each cell corresponds to a physical byte location within the memory device.
The program calculates the physical address for you and displays it at the left side of the screen in the box labeled 'Byte Address' or 'Subaddress'.

Byte Address (Subaddress)

The byte address (sometimes called subaddress or word address) is a pointer to a register or memory location within the I^2C device. To access this location, the software will send out the device I^2C address, followed by this byte address, followed by the read or write data. The program displays the byte address of the active cell of the memory grid in both hexadecimal and decimal notation.

Read Byte Button

Pressing the 'Read Byte' button initiates a read from the I^2C device. The program begins the transmission by writing the device address, followed by the current byte address. A Repeated Start is then generated, followed by the device's read address, and finally a read of a single data byte. The result of each byte read is immediately entered in

the appropriate cell in the grid.

Read All Button

Pressing the 'Read All' button initiates a read from the I²C device. The program begins the transmission by writing the device address, then the byte address 0x00. A Repeated Start is then generated, followed by sequential reads of the entire device.

Write Byte Button

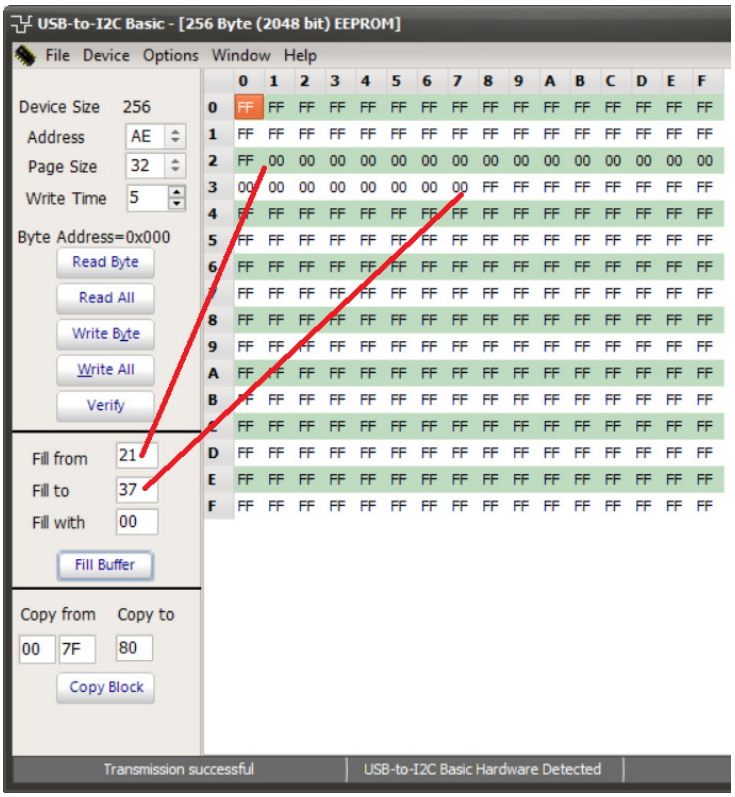
Pressing the 'Write Byte' button initiates a Write to the I²C device. The program begins the transmission by writing the slave address and then the word address of the currently active cell in the grid. The selected data byte is then sent. In addition to pressing the 'Write Byte' button, you may press the <Alt> and <y> keys simultaneously and achieves the same results.

Write All Button

Pressing the 'Write All' button initiates a Write to the I²C device. The program begins the transmission by writing the byte address 0x00 to the device and sequentially writes the entire device.
For a RAM type device, the data is sent in one long message. In the case of an EEPROM, the software will send one page of data (usually 8 bytes but check the datasheet for the particular device you are addressing) followed by a STOP condition. Following the STOP condition, the program waits a length of time, determined by the Erase/Write cycle time of the device (again check the datasheet for the device you are programming), before writing another page to the device. In addition to pressing the 'Write All' button, you may press the <Alt> and <w> keys simultaneously to achieve the same results.
After the completion of the write cycle, USB-to-I2C Basic will read the entire device to verify that the contents of the device match the data that was sent. An error message will be displayed if the data read from the device does not match the data in the on-screen buffer.

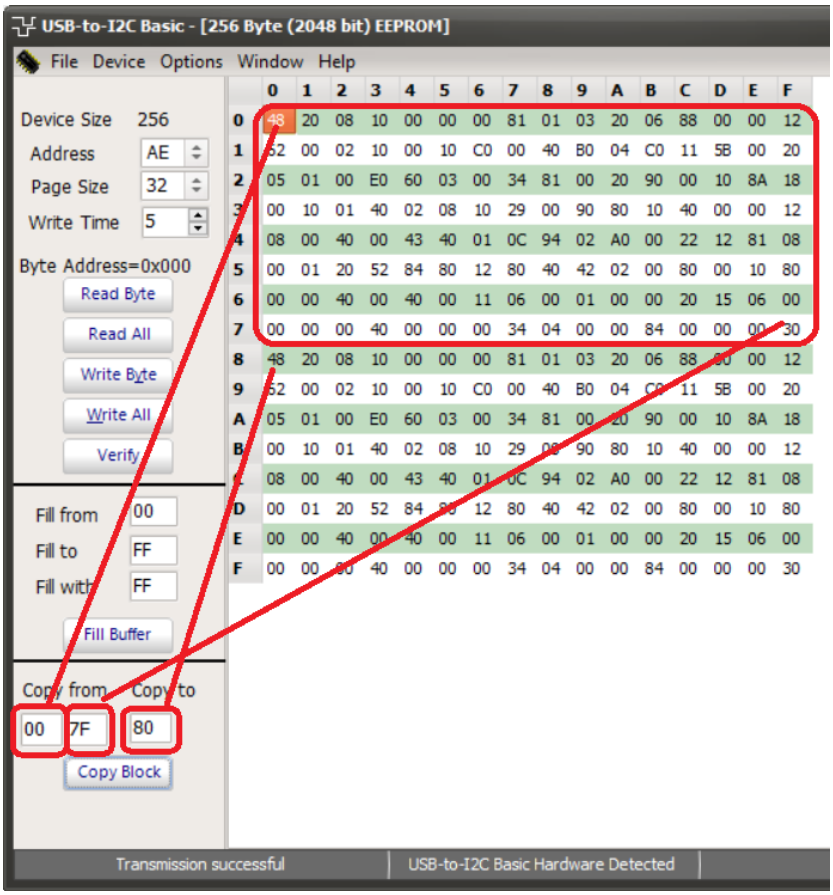
Verify Button

Pressing the 'Verify' button initiates a read of the entire EEPROM. After reading the contents of the EEPROM, USB-to-I2C Basic will compare the contents with the values in the grid. An error will be flagged if the contents of the EEPROM do not match the contents of the USB-to-I2C Basic on-screen grid.



Fill Buffer

The grid will be filled with the two-digit hexadecimal number found in the 'Fill with' edit box when the Fill Buffer button is pressed. The fill can be constrained to the addresses found in the 'Fill from' to the 'Fill to' edit boxes. No information will be sent over the I²C bus.

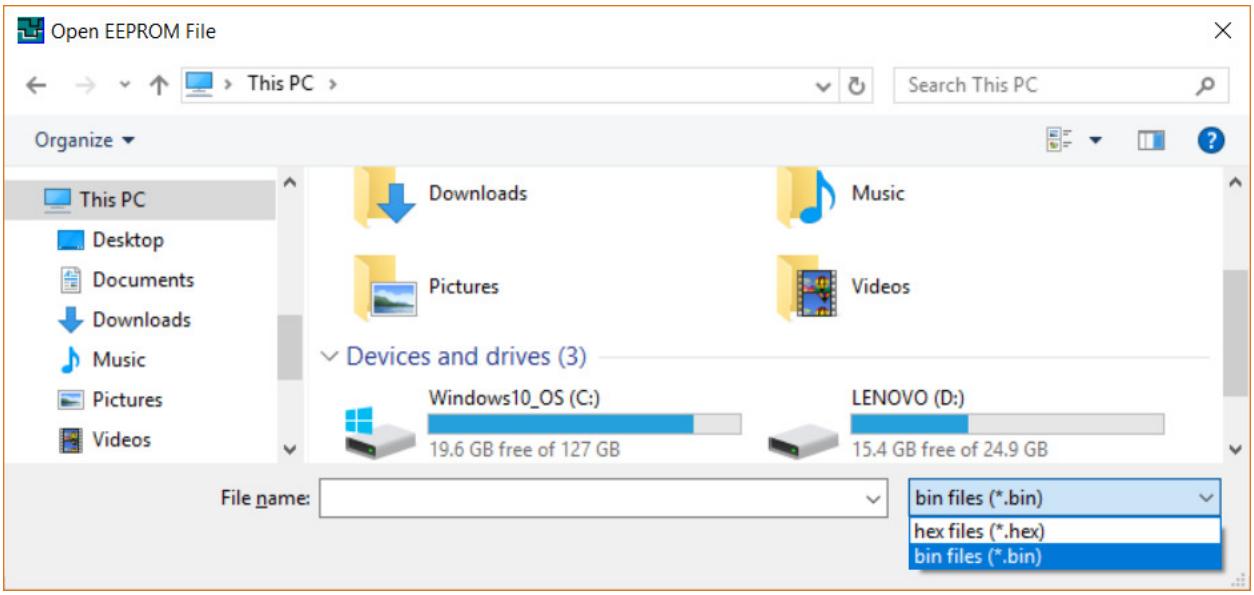


Copy Block

It is possible to copy a block of data from one on-screen buffer area to another using the Copy Block function. Simply define the data to be copied using the 'Copy From' edit boxes and then enter the address where the data is to be copied to. The ending address is calculated automatically by the software and is not editable by the user. Pressing the Copy Block button completes the copying process.

Open and Save Data

Data can be saved for later use in one of two formats:
hex: standard Intel hex file format
bin: binary file.



TROUBLESHOOTING

- The USB-to-I2C Basic has no pull-up resistors on the I²C bus lines. Ensure that the user hardware has pull-up resistors, or add two pull-up resistors to the bottom side of the USB-to-I2C Basic board. See Hardware User's Manual for details.

- The USB-to-I2C software will not function on Windows 95, Windows 98 (First edition), or NT systems, since these operating systems did not support USB.

- The USB-to-I2C software requires that Microsoft GDI+ is installed on your PC. It is normally installed by Windows, however, if you receive an error message saying that the GDI dll is missing, you can download it from the Microsoft website:
  Microsoft GDI+ download

- USB-to-I2C hardware monitors the communications on the I²C bus for proper operation of connected peripherals; any errors on the bus are detected and reported by the software. Bus communication is stopped if errors are detected and can be resumed when the (hardware) problem is corrected and the transmission retried.

- Check for new versions of the software at https://www.i2ctools.com/manuals-and-downloads/

- The firmware for the Basic hardware can be upgraded by the user. The latest version information can be found on the i2ctools website. Email a request to support@i2ctools.com with your serial number information to receive the update.

If all else fails, email a description of the problem you are having to us at support@i2ctools.com. Note that all technical support requests must begin with an email to this email address.

We are interested in receiving feedback from our customers. Is there is a feature that should be added to make this tool better? Please send your requests and comments to support@i2ctools.com.