

USB-SPI/I2C Protocol Emulator

User's Manual

2013.10

Rev. 1.0



REX-USB61 USB-SPI/I2C Protocol Emulator

| 1. Introduction1-1 $(1-1)$ Specifications of the product1-1 $(1-2)$ Package contents1-3 $(1-3)$ Cable specifications1-4 $(1-4)$ Each mode1-5 $(1-5)$ Connection of a SPI device1-6 $(1-6)$ Connection of a I2C device1-92. Setting up on Windows2-1 $(2-1)$ Setup on Windows 8 x32/8 x64/7 x32/7 x64/Vista x642-1 $(2-2)$ Setting up on Windows 8 x32/8 x64/7 x32/7 x64/Vista x642-5 $(2-2)$ Setting up on Windows Vista x322-3 $(2-3)$ Setting up on Windows XP x32/XP x642-6 $(2-4)$ Confirmation of setting REX-USB612-6 $(2-6)$ Uninstallation on Windows Vista x32/XP x32/XP x642-83. SPI/I2C Control Utility3-1 $(3-1)$ Functions of the Utility3-1 $(3-2)$ Explanation of the Utility3-1 $(3-3)$ Example to control by using this utility3-8 $(3-4)$ Grammar for script description3-13 $(3-5)$ Example of script3-184. API Function reference4-1 | | | |
|---|----------|---|------|
| (1-1) Specifications of the product1-1(1-2) Package contents1-3(1-2) Package contents1-4(1-2) Cable specifications1-4(1-4) Each mode1-5(1-5) Connection of a SPI device1-6(1-6) Connection of a I2C device1-92. Setting up on Windows2-1(2-1) Setup on Windows 8 x32/8 x64/7 x32/7 x64/Vista x642-1(2-2) Setting up on Windows 8 x32/8 x64/7 x32/7 x64/Vista x642-5(2-3) Setting up on Windows XP x32/XP x642-5(2-4) Confirmation of setting REX-USB612-6(2-5) Uninstallation on Windows 8 x32/8 x64/7 x32/7 x64/Vista x642-7(2-6) Uninstallation on Windows Vista x32/XP x32/XP x642-83. SPI/I2C Control Utility3-1(3-1) Functions of the Utility3-1(3-2) Explanation of the Utility3-1(3-3) Example to control by using this utility3-8(3-4) Grammar for script description3-13(3-5) Example of script3-184. API Function reference4-1 | | | |
| (1-1) Specifications of the product1-1(1-2) Package contents1-3(1-2) Package contents1-4(1-2) Cable specifications1-4(1-4) Each mode1-5(1-5) Connection of a SPI device1-6(1-6) Connection of a I2C device1-92. Setting up on Windows2-1(2-1) Setup on Windows 8 x32/8 x64/7 x32/7 x64/Vista x642-1(2-2) Setting up on Windows 8 x32/8 x64/7 x32/7 x64/Vista x642-5(2-3) Setting up on Windows 8 x32/8 x64/7 x32/7 x64/Vista x642-6(2-3) Setting up on Windows XP x32/XP x642-6(2-4) Confirmation of setting REX-USB612-6(2-5) Uninstallation on Windows 8 x32/8 x64/7 x32/7 x64/Vista x642-7(2-6) Uninstallation on Windows Vista x32/XP x32/XP x642-83. SPL/I2C Control Utility3-1(3-1) Functions of the Utility3-1(3-2) Explanation of the Utility3-1(3-3) Example to control by using this utility3-8(3-4) Grammar for script description3-13(3-5) Example of script3-184. API Function reference4-1 | | | |
| (1-1) Specifications of the product1-1 $(1-2)$ Package contents1-3 $(1-2)$ Package contents1-4 $(1-2)$ Cable specifications1-4 $(1-4)$ Each mode1-5 $(1-5)$ Connection of a SPI device1-6 $(1-6)$ Connection of a I2C device1-92. Setting up on Windows2-1 $(2-1)$ Setup on Windows 8 x32/8 x64/7 x32/7 x64/Vista x642-1 $(2-2)$ Setting up on Windows 8 x32/8 x64/7 x32/7 x64/Vista x642-5 $(2-3)$ Setting up on Windows Vista x322-3 $(2-3)$ Setting up on Windows XP x32/XP x642-6 $(2-4)$ Confirmation of setting REX-USB612-6 $(2-5)$ Uninstallation on Windows 8 x32/8 x64/7 x32/7 x64/Vista x642-7 $(2-6)$ Uninstallation on Windows Vista x32/XP x32/XP x642-83. SPL/I2C Control Utility3-1 $(3-1)$ Functions of the Utility3-1 $(3-2)$ Explanation of the Utility3-1 $(3-3)$ Example to control by using this utility3-8 $(3-4)$ Grammar for script description3-13 $(3-5)$ Example of script3-184. API Function reference4-1 | | | |
| (1-1) Specifications of the product1-1 $(1-2)$ Package contents1-3 $(1-2)$ Package contents1-4 $(1-2)$ Cable specifications1-4 $(1-4)$ Each mode1-5 $(1-5)$ Connection of a SPI device1-6 $(1-6)$ Connection of a I2C device1-92. Setting up on Windows2-1 $(2-1)$ Setup on Windows 8 x32/8 x64/7 x32/7 x64/Vista x642-1 $(2-2)$ Setting up on Windows 8 x32/8 x64/7 x32/7 x64/Vista x642-5 $(2-3)$ Setting up on Windows Vista x322-3 $(2-3)$ Setting up on Windows XP x32/XP x642-6 $(2-4)$ Confirmation of setting REX-USB612-6 $(2-5)$ Uninstallation on Windows 8 x32/8 x64/7 x32/7 x64/Vista x642-7 $(2-6)$ Uninstallation on Windows Vista x32/XP x32/XP x642-83. SPL/I2C Control Utility3-1 $(3-1)$ Functions of the Utility3-1 $(3-2)$ Explanation of the Utility3-1 $(3-3)$ Example to control by using this utility3-8 $(3-4)$ Grammar for script description3-13 $(3-5)$ Example of script3-184. API Function reference4-1 | • | | |
| (1-1) Specifications of the product1-1 $(1-2)$ Package contents1-3 $(1-2)$ Package contents1-4 $(1-2)$ Cable specifications1-4 $(1-4)$ Each mode1-5 $(1-5)$ Connection of a SPI device1-6 $(1-6)$ Connection of a I2C device1-92. Setting up on Windows2-1 $(2-1)$ Setup on Windows 8 x32/8 x64/7 x32/7 x64/Vista x642-1 $(2-2)$ Setting up on Windows 8 x32/8 x64/7 x32/7 x64/Vista x642-5 $(2-3)$ Setting up on Windows Vista x322-3 $(2-3)$ Setting up on Windows XP x32/XP x642-6 $(2-4)$ Confirmation of setting REX-USB612-6 $(2-5)$ Uninstallation on Windows 8 x32/8 x64/7 x32/7 x64/Vista x642-7 $(2-6)$ Uninstallation on Windows Vista x32/XP x32/XP x642-83. SPL/I2C Control Utility3-1 $(3-1)$ Functions of the Utility3-1 $(3-2)$ Explanation of the Utility3-1 $(3-3)$ Example to control by using this utility3-8 $(3-4)$ Grammar for script description3-13 $(3-5)$ Example of script3-184. API Function reference4-1 | • | | |
| (1-1) Specifications of the product1-1 $(1-2)$ Package contents1-3 $(1-2)$ Package contents1-4 $(1-2)$ Cable specifications1-4 $(1-4)$ Each mode1-5 $(1-5)$ Connection of a SPI device1-6 $(1-6)$ Connection of a I2C device1-92. Setting up on Windows2-1 $(2-1)$ Setup on Windows 8 x32/8 x64/7 x32/7 x64/Vista x642-1 $(2-2)$ Setting up on Windows 8 x32/8 x64/7 x32/7 x64/Vista x642-5 $(2-3)$ Setting up on Windows Vista x322-3 $(2-3)$ Setting up on Windows XP x32/XP x642-6 $(2-5)$ Uninstallation of setting REX-USB612-6 $(2-5)$ Uninstallation on Windows 8 x32/8 x64/7 x32/7 x64/Vista x642-7 $(2-6)$ Uninstallation on Windows Vista x32/XP x32/XP x642-83. SPL/I2C Control Utility3-1 $(3-1)$ Functions of the Utility3-1 $(3-2)$ Explanation of the Utility3-1 $(3-3)$ Example to control by using this utility3-8 $(3-4)$ Grammar for script description3-13 $(3-5)$ Example of script3-184. API Function reference4-1 | | | |
| (1-1) Specifications of the product1-1 $(1-2)$ Package contents1-3 $(1-2)$ Package contents1-4 $(1-2)$ Cable specifications1-4 $(1-4)$ Each mode1-5 $(1-5)$ Connection of a SPI device1-6 $(1-6)$ Connection of a I2C device1-92. Setting up on Windows | • | | |
| (1-2) Package contents 1-3 (1-3) Cable specifications 1-4 (1-4) Each mode 1-5 (1-4) Each mode 1-6 (1-5) Connection of a SPI device 1-6 (1-6) Connection of a I2C device 1-9 2. Setting up on Windows 2-1 (2-1) Setup on Windows 8 x32/8 x64/7 x32/7 x64/Vista x64 2-1 (2-2) Setting up on Windows Vista x32 2-3 (2-3) Setting up on Windows Vista x32 2-5 (2-3) Setting up on Windows Vista x32 2-5 (2-4) Confirmation of setting REX-USB61 2-6 (2-5) Uninstallation on Windows 8 x32/8 x64/7 x32/7 x64/Vista x64 2-7 (2-6) Uninstallation on Windows 8 vista x32/XP x64 2-8 3. SPI/I2C Control Utility 3-1 (3-1) Functions of the Utility 3-1 (3-2) Explanation of the Utility 3-2 (3-3) Example to control by using this utility 3-8 (3-4) Grammar for script description 3-13 (3-5) Example of script 3-18 4. API Function reference 4-1 | • 1. Int | roduction | 1-1 |
| (1-3) Cable specifications 1-4 (1-4) Each mode 1-5 (1-4) Each mode 1-6 (1-5) Connection of a SPI device 1-6 (1-6) Connection of a I2C device 1-9 2. Setting up on Windows 2-1 (2-1) Setup on Windows & x32/8 x64/7 x32/7 x64/Vista x64 2-1 (2-2) Setting up on Windows & x32/8 x64/7 x32/7 x64/Vista x64 2-5 (2-3) Setting up on Windows XP x32/XP x64 2-5 (2-4) Confirmation of setting REX-USB61 2-6 (2-5) Uninstallation on Windows 8 x32/8 x64/7 x32/7 x64/Vista x64 2-7 (2-6) Uninstallation on Windows Vista x32/XP x32/XP x64 2-8 3. SPI/I2C Control Utility 3-1 (3-1) Functions of the Utility 3-1 (3-2) Explanation of the Utility 3-1 (3-3) Example to control by using this utility 3-8 (3-4) Grammar for script description 3-13 (3-5) Example of script 3-18 4. API Function reference 4-1 | • | (1–1) Specifications of the product | 1-1 |
| (1-4) Each mode 1-5 (1-5) Connection of a SPI device 1-6 (1-6) Connection of a I2C device 1-9 2. Setting up on Windows 2-1 (2-1) Setup on Windows & x32/8 x64/7 x32/7 x64/Vista x64 2-1 (2-2) Setting up on Windows Vista x32 2-3 (2-3) Setting up on Windows Vista x32 2-5 (2-4) Confirmation of setting REX-USB61 2-6 (2-5) Uninstallation on Windows Vista x32/8 x64/7 x32/7 x64/Vista x64 2-7 (2-6) Uninstallation on Windows Vista x32/8 x64/7 x32/7 x64/Vista x64 2-7 (2-6) Uninstallation of the Utility 3-1 (3-1) Functions of the Utility 3-1 (3-2) Explanation of the Utility 3-8 (3-3) Example to control by using this utility 3-8 (3-4) Grammar for script description 3-13 (3-5) Example of script 3-18 4. API Function reference 4-1 | • | (1–2) Package contents | 1-3 |
| (1-5) Connection of a SPI device 1-6 (1-6) Connection of a I2C device 1-9 2. Setting up on Windows 2-1 (2-1) Setup on Windows & x32/8 x64/7 x32/7 x64/Vista x64 2-1 (2-2) Setting up on Windows Vista x32 2-3 (2-3) Setting up on Windows Vista x32 2-5 (2-4) Confirmation of setting REX-USB61 2-6 (2-5) Uninstallation on Windows & x32/8 x64/7 x32/7 x64/Vista x64 2-7 (2-6) Uninstallation on Windows & Vista x32/XP x32/XP x64 2-8 3. SPI/I2C Control Utility 3-1 (3-1) Functions of the Utility 3-1 (3-2) Explanation of the Utility 3-8 (3-3) Example to control by using this utility 3-8 (3-4) Grammar for script description 3-13 (3-5) Example of script 3-18 4. API Function reference 4-1 | • | (1–3) Cable specifications | 1-4 |
| (1-6) Connection of a I2C device 1-9 2. Setting up on Windows | • | (1–4) Each mode | 1-5 |
| 2. Setting up on Windows | • | (1–5) Connection of a SPI device | 1- 6 |
| (2-1) Setup on Windows 8 x32/8 x64/7 x32/7 x64/Vista x64 2-1 (2-2) Setting up on Windows Vista x32 2-3 (2-3) Setting up on Windows XP x32/XP x64 2-5 (2-4) Confirmation of setting REX-USB61 2-6 (2-5) Uninstallation on Windows 8 x32/8 x64/7 x32/7 x64/Vista x64 2-7 (2-6) Uninstallation on Windows Vista x32/XP x32/XP x64 2-8 3. SPI/I2C Control Utility 3-1 (3-1) Functions of the Utility 3-1 (3-2) Explanation of the Utility 3-2 (3-3) Example to control by using this utility 3-8 (3-4) Grammar for script description 3-13 (3-5) Example of script 3-18 4. API Function reference 4-1 | • | (1–6)Connection of a I2C device | 1-9 |
| (2-1) Setup on Windows 8 x32/8 x64/7 x32/7 x64/Vista x64 2-1 (2-2) Setting up on Windows Vista x32 2-3 (2-3) Setting up on Windows XP x32/XP x64 2-5 (2-4) Confirmation of setting REX-USB61 2-6 (2-5) Uninstallation on Windows 8 x32/8 x64/7 x32/7 x64/Vista x64 2-7 (2-6) Uninstallation on Windows Vista x32/XP x32/XP x64 2-8 3. SPI/I2C Control Utility 3-1 (3-1) Functions of the Utility 3-1 (3-2) Explanation of the Utility 3-2 (3-3) Example to control by using this utility 3-8 (3-4) Grammar for script description 3-13 (3-5) Example of script 3-18 4. API Function reference 4-1 | • | | |
| (2-2) Setting up on Windows Vista x322-3(2-3) Setting up on Windows XP x32/XP x642-5(2-4) Confirmation of setting REX-USB612-6(2-5) Uninstallation on Windows 8 x32/8 x64/7 x32/7 x64/Vista x642-7(2-6) Uninstallation on Windows Vista x32/XP x32/XP x642-83. SPI/I2C Control Utility3-1(3-1) Functions of the Utility3-1(3-2) Explanation of the Utility3-8(3-3) Example to control by using this utility3-8(3-4) Grammar for script description3-13(3-5) Example of script3-184. API Function reference4-1 | • 2. Set | - • | 2-1 |
| (2-3) Setting up on Windows XP x32/XP x642-5(2-4) Confirmation of setting REX-USB612-6(2-5) Uninstallation on Windows 8 x32/8 x64/7 x32/7 x64/Vista x642-7(2-6) Uninstallation on Windows Vista x32/XP x32/XP x642-83. SPL/I2C Control Utility3-1(3-1) Functions of the Utility3-1(3-2) Explanation of the Utility3-2(3-3) Example to control by using this utility3-8(3-4) Grammar for script description3-13(3-5) Example of script3-184. API Function reference4-1 | • | (2–1) Setup on Windows 8 x32/8 x64/7 x32/7 x64/Vista x64 | 2- 1 |
| (2-4) Confirmation of setting REX-USB612-6(2-5) Uninstallation on Windows 8 x32/8 x64/7 x32/7 x64/Vista x642-7(2-6) Uninstallation on Windows Vista x32/XP x32/XP x642-83. SPI/I2C Control Utility3-1(3-1) Functions of the Utility3-1(3-2) Explanation of the Utility3-2(3-3) Example to control by using this utility3-8(3-4) Grammar for script description3-13(3-5) Example of script3-184. API Function reference4-1 | • | (2–2) Setting up on Windows Vista x32 | 2-3 |
| (2-5) Uninstallation on Windows 8 x32/8 x64/7 x32/7 x64/Vista x642-7(2-6) Uninstallation on Windows Vista x32/XP x32/XP x642-83. SPI/I2C Control Utility3-1(3-1) Functions of the Utility3-1(3-2) Explanation of the Utility3-2(3-3) Example to control by using this utility3-8(3-4) Grammar for script description3-13(3-5) Example of script3-184. API Function reference4-1 | • | (2–3) Setting up on Windows XP x32/XP x64 | 2- 5 |
| (2-6) Uninstallation on Windows Vista x32/XP x32/XP x642-83. SPI/I2C Control Utility3-1(3-1) Functions of the Utility3-1(3-2) Explanation of the Utility3-2(3-3) Example to control by using this utility3-8(3-4) Grammar for script description3-13(3-5) Example of script3-184. API Function reference4-1 | • | (2-4) Confirmation of setting REX-USB61 | 2-6 |
| 3. SPI/I2C Control Utility 3-1 (3-1) Functions of the Utility 3-1 (3-2) Explanation of the Utility 3-2 (3-3) Example to control by using this utility 3-8 (3-4) Grammar for script description 3-13 (3-5) Example of script 3-18 4. API Function reference 4-1 | • | (2–5) Uninstallation on Windows 8 x32/8 x64/7 x32/7 x64/Vista x64 | 2-7 |
| (3-1) Functions of the Utility3-1(3-2) Explanation of the Utility3-2(3-3) Example to control by using this utility3-8(3-4) Grammar for script description3-13(3-5) Example of script3-184. API Function reference4-1 | • | (2–6) Uninstallation on Windows Vista x32/XP x32/XP x64 | 2- 8 |
| (3-1) Functions of the Utility3-1(3-2) Explanation of the Utility3-2(3-3) Example to control by using this utility3-8(3-4) Grammar for script description3-13(3-5) Example of script3-184. API Function reference4-1 | • | | |
| (3-2) Explanation of the Utility3-2(3-3) Example to control by using this utility3-8(3-4) Grammar for script description3-13(3-5) Example of script3-184. API Function reference4-1 | 3. SP. | I/I2C Control Utility | 3- 1 |
| (3-3) Example to control by using this utility3-8(3-4) Grammar for script description3-13(3-5) Example of script3-184. API Function reference4-1 | • | (3–1) Functions of the Utility | 3-1 |
| (3-4) Grammar for script description3-13(3-5) Example of script3-184. API Function reference4-1 | • | (3–2) Explanation of the Utility | 3-2 |
| (3-5) Example of script 3-18 4. API Function reference 4-1 | • | (3–3) Example to control by using this utility | 3-8 |
| 4. API Function reference 4- 1 | • | (3–4) Grammar for script description | 3–13 |
| | • | (3–5) Example of script | 3–18 |
| | • | | |
| (4-1) Using on VG $4-1$ | 4. AP. | I Function reference | 4-1 |
| • - | • | (4–1) Using on VC | 4-1 |
| (4–2) Using on VB / Visual C# 4– 3 | • | | 4-3 |
| (4–3) List of API functions 4–8 | • | (4–3) List of API functions | |
| (4–4) Detail of API functions 4– 9 | • | (4–4) Detail of API functions | |
| (4–5) Error Codes 4–24 | | (4–5) Error Codes | |
| (4–6) Sample applications 4–25 | • | (4–6) Sample applications | |
| (4–7) How to develop application using this API functions 4–27 | • | (4–7) How to develop application using this API functions | 4–27 |
| | • | | |
| | • | | |
| | • | | |





(1-1) Specifications of the product

REX-USB61 enables you to easily control from a PC a variety of devices with SPI/I2C bus.

[This product comes with SPI/I2C control utility]

This bundled utility can control SPI/I2C, GPO(General Purpose Output) and save a setting file or log file.

For further information, please refer to Chapter 3.

[This product also comes with API library and sample program]

Making an application software with the API library enables you to control the following:

- Can provide a power supply of 3.3V or 5.0V([N.B.] current is under 100mA) from this product to an external device.
- Can provide from 1.8V to 5.0V an input/output level of SPI/I2C/slave port/parallel out port as long as a power supply terminal of this product is provided by a external voltage.
- Can change SPI/I2C, master/slave(SPI is a master only)
- Can specify a frequency of SPI/I2C bus.
- Can output a digital of 4bit at I2C mode.

And this product comes with program source codes by which you can use API library.

(For further information on functions, please refer to (4-4) at Chapter 4. Further information on applications, please refer to (4-6) at Chapter 4.)

[The latest firmware is available through our website]

You can update firmware in order to add or change specifications on this products. The latest firmware and update program is available through our website.

Hardware specifications

| Item | Specifications |
|---------------------|--|
| Host Interface | USB2.0 Full Speed Device |
| Connector | USB mini B connector |
| Voltage | 5V (via USB bus power) |
| Consumption Current | 100mA |
| Derrice Interface | SPI Master Max. frequency 12MHz |
| Device Interface | I2C Master/Slave Frequency 47KHz~1MHz |
| Innut/Output loval | [Output] 3.3V/5V |
| Input/Output level | [Input] 1.8V - 5.0V is enabled with external P/S |
| Dimension | 57(W) x 75(D) x 18(H) mm |
| Weight | Approx.60g (except cable) |
| Operating | Temperature:5~55°C Humidity:20~80% |
| Environment | (non condensing) |

Support Operating System

Windows 8/7/Vista/XP * Works both 32bit OS and 64bit OS

Software

| Item | File | Description |
|--------------------|------------------|------------------------------------|
| Setting file | | Setting file for REX-USB61 |
| for installation | USB61.inf | (Windows Vista x32/XP x32/XP x64) |
| Installer | USB61_Setup.exe | Installer for Windows 8 x32/8 x64/ |
| | CODOI_Detup.exe | 7 x32/7 x64/Vista x64 |
| Utility | Usb61Uty.exe | Utility to control SPI/I2C |
| Semint file | I2C_script.txt | Script file for I2C bus control |
| Script file | SPI_script.txt | Script file for SPI bus control |
| Sample program | EEPROMRWUty | Sample program to send/receive |
| (VC6.0/VB6.0/VB | | SPI/I2C |
| 2005/C#) | I2cSlaveSample | Sample program for I2C slave |
| | usb61api.dll | Library to control SPI/I2C devices |
| Tibrowy | usb61def.h | Header file for Visual C |
| Library | usb61api.bas | Module for Visual Basic |
| | usb61api.vb | Code file for Visual Basic |
| ActiveX control | usb61api.ocx | ActiveX control for REX-USB61 |
| IIminatall utility | UCDC1 uningt ave | Utility to delete INF file |
| Uninstall utility | USB61_uninst.exe | (Windows XP x32/XP x64) |

* REX-USB61 can only use 1 device. On the other hand, REX-USB61M can use multiple devices.

(1-2) Package contents

REX-USB61 package includes:

- ☑ REX-USB61
- CD-ROM
- ☑ USBA mini B cable
- SPI/I2C cable
- \blacksquare Warranty Card



(1-3) Cable Specifications

| The below explains the specifications of the cable bundled wit | h REX-USB61. |
|--|--------------|
|--|--------------|

| Pin | Housing | Cable | <u>C' 1</u> | II |
|---------|-----------|-----------|--------------|--|
| number | color | color | Signal | Usage |
| 1 | Black | Brown | Power | Input/Output of power supply for a target device (Output 5V or 3.3V @100mA) (Input 1.8V - 5V) |
| 2 | Black | Red | Power | Input/Output of power supply for a target device (Output 5V or 3.3V @100mA) (Input 1.8V - 5V) |
| 3 | Black | Orange | 1MHz - SCL | Clock for I2C (401KHz - 1MHz bus voltage 5V only) (Pull-up resistance 10kΩ) |
| 4 | Black | Yellow | 1MHz - SDA | Data signal for I2C (401KHz - 1MHz bus voltage 5V only) (Pull-up resistance 10kΩ) |
| 5 | Black | Green | SCL | Clock for I2C (47KHz-400KHz 1.8-5V) (Pull-up resistance 10kΩ) |
| 6 | Black | Blue | SDA | Data signal for I2C (47KHz-400KHz 1.8-5V) (Pull-up resistance 10kΩ) |
| 7 | Black | Purple | SCK | Clock signal for SPI (12MHz 1.8 - 5V) |
| 8 | Black | Gray | SDO | Data out signal SPI (12MHz 1.8 - 5V) |
| 9 | Black | White | SDI | Data in signal SPI (12MHz 1.8 - 5V) |
| 10 | Black | Black | Reserve | N/A(Don't use) |
| * Don't | use I2C 4 | 401KHz-11 | MHz(Pin#3,4) | and SPI(Pin#7,8,9) at the same time. |

| Pin number | Housing color | Cable color | Signal | Usage |
|---------------|---------------|----------------|--------|--------------------------------------|
| 11 | White(Gray) | Gray | GND | Ground |
| 12 | White(Gray) | Red | GND | Ground |
| 13 | White(Gray) | Orange | DO0 | SS0 for SPI/PORT0 for I2C (1.8 - 5V) |
| 14 | White(Gray) | Yellow | DO1 | SS1 for SPI/PORT1 for I2C (1.8 - 5V) |
| 15 | White(Gray) | Green | DO2 | SS2 for SPI/PORT2 for I2C (1.8 - 5V) |
| 16 | White(Gray) | Blue | DO3 | SS3 for SPI/PORT3 for I2C (1.8 - 5V) |
| 17 | White(Gray) | Purple | GND | Ground |
| 18 | White(Gray) | Gray | GND | Ground |
| 19 | White(Gray) | White | N.C. | N.C. |
| 20 | White(Gray) | Black | N.C. | N.C. |

(1-4) Each mode

The below explains master/slave mode on SPI /I2C bus.

| Bus | | Operation |
|---------|-------------|--|
| CDI Dua | Master mode | This mode can select a slave, send data, |
| SPI Bus | Master mode | display data received from the slave. |
| | Mastannada | This mode can send data to a particular |
| IOC Due | Master mode | address, display data received from the slave. |
| I2C Bus | | This mode can display data received to self |
| | Slave mode | -address, send data to master. |

You can select master mode or slave mode of REX-USB61 by the bundled utility software or API library.

(1-5) Connection of a SPI device

The below explains how to connect an EEPROM with SPI interface.

• Regarding power supply of REX-USB61

In order to provide power supply to a level converter IC on the REX-USB61, <u>it is required to connect the power pin of the REX-USB61 to</u> <u>a power supply of a target device</u>, even if the target device doesn't have power supply.



[Caution]

When connecting/disconnecting a device, never provide power to REX-USB61 nor the device.

(If you provide power to REX-USB61 or the device and connect or disconnect the device, REX-USB61 will be broken.)

SPI Connection(If a target device has power supply)

If a target device has power supply, please disable power supply by utility software or application which uses an API library.

(The library is called usb61_power_control). Refer to (4-4) at Chapter 4.)



SPI Connection(If a target device doesn't have power supply)

If REX-USB61 supply power(3.3V/5.0V) to a target device, please use utility software or application which uses an API library.

(The library is called usb61_power_control). Refer to (4-4) at Chapter 4.)



(1-6) Connection of a I2C device

The below explains how to connect an EEPROM with I2C interface.

• Regarding power supply of REX-USB61

In order to provide power supply to a level converter IC on the REX-USB61, it is required to connect the power pin of the REX-USB61 to a power supply of a target device, even if the target device doesn't have power supply.



[Caution]

When connecting/disconnecting a device, never provide power to REX-USB61 nor the device.

(If you provide power to REX-USB61 or the device and connect or disconnect the device, REX-USB61 will be broken.) If a target device has power supply, please disable power supply by utility software or application which uses an API library.

(The library is called usb61_power_control). Refer to (4-4) at Chapter 4.)



*1 The pull-up resistance on REX-USB61 is $10K\Omega$. If necessary, add pull-up resistance.

I2C connection(If a target device doesn't power supply)

If REX-USB61 supply power(3.3V/5.0V) to a target device, please use utility software or application which uses an API library.

(The library is called usb61_power_control). Refer to (4-4) at Chapter 4.)



*1 The pull-up resistance on REX-USB61 is $10K\Omega$. If necessary, add pull-up resistance.

I2C connection [1MHz-SCL / 1MHz-SDA] (If a target device has power supply)

If a target device has power supply, please disable power supply by utility software or application which uses an API library.

(The library is called usb61_power_control). Refer to (4-4) at Chapter 4.)



* Only after providing power to all devices, set on pull-up resistance.

* If a target device provide power, don't attach pull-up resistance on I2C bus.

I2C connection [1MHz-SCL / 1MHz-SDA] (If a target device doesn't have power supply)

If REX-USB61 supply power(5.0V) to a target device, please use utility software or application which uses an API library.

(The library is called usb61_power_control). Refer to (4-4) at Chapter 4.)



* Only after providing power to all devices, set on pull-up resistance.

*1 The pull-up resistance on REX-USB61 is $10K\Omega$. If necessary, add pull-up resistance.



Turn on the PC and proceed to the below installation before connecting REX-USB61 to the USB port.

X User Account Control Start Do you want to allow the following program to make Win8_7_VistaX64¥USB61_ changes to this computer? Setup.exe at the bundled Program name: RATOC REX-USB61 Installer Verified publisher: RATOC Systems, Inc. CD-ROM. File origin: Hard drive on this computer If user account window appear, Show details Yes No click [Yes]. Change when these notifications appear RATOC REX-USB61 Installer - InstallShield Wizard RATOC REX-USB61 Installer 4 will start. Click [Next]. The InstallShield Wizard will install RATOC REX-USB61 Installer on your computer. To continue, click Next.



Click [Install].



Click [Install] on the Windows Security window.



The set up has finished.

If REX-USB61 is connected to the PC, the installation will automatically finish.

| RATOC REX-USB61 Installer - In | nstallShield Wizard |
|--------------------------------|---|
| | InstallShield Wizard Complete The InstallShield Wizard has successfully installed RATOC REX-USB61 Installer. Click Finish to exit the wizard. |
| | < <u>Back</u> Finish Cancel |

Proceed to (2-4) Confirmation of setting REX- USB61 to confirm the installation has finished properly.

(2-2) Setting up on Windows Vista x32

Turn on the PC and connect REX-USB61 to the USB port. The below hardware wizard will start up. Proceed to the below instruction.

Select [Locate and install driver software (recommended)].



If user account window appear, click [Yes].

Click [Don't search online], as shown right.





Proceed to (2-4) Confirmation of setting REX- USB61 to confirm the installation has finished properly.

(2-3) Setting up on Windows XP x32/XP x64

Turn on the PC and connect REX-USB61 to the USB port. The below hardware wizard will start up. Proceed to the below instruction.

Select [No, not this time] and click [Next].

Insert the bundled CD-ROM

and select [Install the software

automatically(Recommended)]

and click [Next].

has finished.



Click Finish to close the wizard.

Finish

Cancel

The installation of REX-USB61

Proceed to (2-4) Confirmation of setting REX- USB61 to confirm the installation has finished properly.

(2-4) Confirmation of setting REX- USB61

Open [Device Manager].

(X On Windows XP x32/XP x64, open [Control Panel] and [System].

And select the [Hardware] tab and click the [Device manager] button.)

Confirm there is a string of [RATOC REX-USB61] properly under the [SPI/I2C Converter Device].



(2-5) Uninstallation on Windows 8 x32/8 x64/7 x32/7 x64/Vista x64 Start [Programs and Functions].





The uninstallation of REX-USB61 has finished.



Click [Yes].

(2-6) Uninstallation on Windows Vista x32/XP x32/XP x64

To uninstall REX-USB61, you have to delete the driver and INF file. (On Windows Vista, you have to delete the driver only.)

• Delete the driver

Open [Device Manager]. (※ On Windows XPx32/XPx64, open [Control Panel] and [System]. And select the [Hardware] tab

and click the [Device manager] button.)

Right-click the [RATOC REX-USB61] and select [Uninstall].



On Windows Vista x32, put the check mark, as shown right and click [OK].



• Delete INF file

(Windows XPx32/XPx64)

Start [USB61_uninst.exe] at the bundled CD-ROM. ([CD-ROM]:¥USB61_uninst .exe)

When the dialog shown right appear, click [OK].



When the dialog shown right appear, click [OK].

| llShield Wi 💌 |
|---------------|
| alled. |
| ОК |
| |

The uninstallation of REX-USB61 has finished.



Functions of the utility (3-1)

The bundled Usb61Uty.exe can control a target device with SPI or I2C interface and has the following functions:

- Switch operation modes for SPI and I2C
- Control SPI device(Master operation)
- Control I2C device(Master/Slave operation)
- Control PORT pin
- Read/Write setting values
- Save setting files(BIN file format)
- Load setting files
- Save log files(CSV file format)

| | Tabl | le 3-1 Utility Functions |
|---------|-----------|------------------------------------|
| | | Functions |
| | Supply po | ower to a target device |
| Common | | ne interval between data |
| items | Save tran | sfer log files |
| | Switch op | eration modes for SPI and I2C |
| | | Set clock polarity |
| | | Set clock phase |
| | | Set precedent bit |
| | | Set frequency |
| | | Set slave select pin(Max.4) |
| SPI bus | Master | Create transfer data |
| SIIDUS | Master | Edit transfer data |
| | | Send step-by-step transfer data |
| | | Send batch transfer data |
| | | Repeatedly send transfer data |
| | | Save transfer data file |
| | | Read transfer data saved in a file |
| | | Set frequency |
| | | Create transfer data |
| | | Edit transfer data |
| | | Send step-by-step transfer data |
| | Master | Send batch transfer data |
| | Master | Repeatedly send transfer data |
| I2C bus | | Save transfer data file |
| | | Read transfer data saved in a file |
| | | Issue bus reset |
| | | Output PORT pin |
| | | Set frequency |
| | Slave | Set response data to a master |
| | | Set slave address |

(3-2) Explanation of the utility

The below explains screens of the utility and each function.

| SPI Option Slave Sele | ct | | | Samp | ling T | Timir | ng | Po | larity | Ph | ase | Bit c | rder |
|--------------------------|---------|---------|---------------|---------|--------|-------|--------|------|-------------------|---------|-----------------|-------------|------|
| SS0 (| ⊙ SS1 (| 🖱 SS2 (|) SS3 | Æ | R | Ł | F | | Positiv Negati | | Sample Setup | o M ⊙ L: | |
| GPO Option | | | | | | | | | | | | | |
| Port0 | | Port1 | | P | ort2 | | | | Port3 | | | | |
| 🔿 High | | O High | | | 🔵 Hie | | | | 🔿 High | | | C | _ |
| Low | | O Low | | 0 | Lov | N. | | | Low | | | Set | |
| Master Slav | e | | | | | | | | | | | | |
| Transfer Li | st | | | | | | | | F | requenc | y 100 | | • кн |
| Num | Line | Data | | | | | | | | | | Size | |
| 0001 | MOSI | 00 01 | 02 03 | 04 05 0 | 06 07 | 08 | 09 0 | A OB | OC OD | OE OF | | 16 | |
| 0002 | MOSI | 00 01 | 02 03 | 04 05 0 | 06 07 | 08 | 09 0 | A OB | OC OD | 0E 0F | | 16 | |
| 0003 | MOSI | 00 01 | 02 03 | 04 05 0 | 06 07 | 08 | 09 0 | A OB | 0C 0D | OE OF | | 16 | |
| 0004 | MOSI | 00 01 | 02 03 | 04 05 0 | 06 07 | 08 | 09 0 | A OB | OC OD | OE OF | | 16 | |
| 0005 | MOSI | 00 01 | 02 03 | 04 05 0 | 06 07 | 08 | 09 0 | A OB | 0C 0D | OE OF | | 16 | |
| 0006 | MOSI | 00 01 | 02 03 | 04 05 0 | 06 07 | 08 | 09 0 | A OB | 0C 0D | OE OF | | 16 | |
| Send St | en l | Send | A II) | Cor | ntinue | | 1 | | | | 5 | PI Ma | do |
| | | 00.12 | | | | | 1 | | | | 1.01 | 1 1 1 1 1 1 | uc. |
| Transfer Log | | 1158 | 1. 23 | 1122 | | | 111224 | | | | | | |
| time | mode | dir | m/s | freq | ade | dr | size | | data | | | | |
| 15:15:32.77 | | | | | | | | | | | | 100KHz | |
| 15:15:32.78 | | mosi | mas | 100 | | | 16 | | | | 05 06 07 | | |
| 15:15:32.78 | 6 spi | miso | mas | 100 | | - | 16 | 1 | FFFF | FF FF F | F FF FF | FF | |
| | | | | | | | | - | _ | Clear | | Save | |

Fig 3-1. SPI master mode

| SPI/I2C Device | | | | | | | | | |
|--|--------------------|----------------------------|----------|---------------|------|---|---|---------|----------------|
| ile(<u>F)</u> Edit(<u>E</u>) | Device(| D) Option(O |) Help | o(<u>H</u>) | | | | | |
|) 🖬 🖬 🎆 | + 5 | 4 - <i>9</i> B | 18 | | | | | | |
| SPI Option | | | | | | | | | |
| Slave Select | | | Samp | ling Timi | ng | Polarity | Phase | | Bit order |
| 🔘 SS0 🔘 | SS1 C | SS2 🔘 SS3 | | | | Positive Negative | Samp Setup | | MSB LSB |
| GPO Option | | | | | | | | | |
| Port0 | | Port1 | P | ort2 | | Port3 | | | |
| 🔘 High | | 🔘 High | 0 | 🖯 High | | 🔘 High | 4 | | |
| Low | | Low | (| Low | | Low | | | Set |
| Master Slave | | | | | | | | | |
| Response Da | ta | | | | | Slave Addre | 224 | | |
| | | | | | | | | | |
| 00 01 02 03 | | | | | | | | | |
| | 04 00 | 06 07 08 09 0 | A OB OU | ; UD UE | UF 🔺 | 00 | н | | |
| | 04 00 | UE U7 U8 U9 U | A UB UU | ; UD UE | 01- | | | | |
| | 04 00 | 06 U7 U8 U9 U | IA UB UU | ; UD UE | 01 | Master Cloc | | | |
| | 04 00 | UG U7 U8 U9 U | IA UB UL | ; UD UE | UF 🔺 | | k. | ode (Uj | p to 400KHz) |
| | 04 00 | UG U7 U8 U9 U | IA UB UL | ; UD UE | UF | Master Cloc Standard | k or Fast M | | |
| | 04 00 | 06 07 08 09 0 | IA UB UL | ; UD UE | UF * | Master Cloc | k or Fast M | | |
| | . 04 00 | | A UB UL | ; UD UE | UF | Master Cloc Standard | k or Fast M | | |
| Clear | | Enable | A OB OL | ; UD UE | UF * | Master Cloc Standard | k or Fast M | | |
| Clear | | | A OB OC | ; UD UE | UF A | Master Cloc Standard | k or Fast M ed Mode () | Over 4 | DOKHz) |
| Clear | | | A DE OC | ; UD UE | UF * | Master Cloc Standard | k or Fast M ed Mode () | Over 4 | |
| Clear | | | A DE UC | ; UD UE | UF * | Master Cloc Standard | k or Fast M ed Mode () | Over 4 | DOKHz) |
| Clear Transfer Log | | | A DB UC | ; UD UE | UF * | Master Cloc Standard | k or Fast M ed Mode () | Over 4 | DOKHz) |
| | mode | | freq | addr | size | Master Cloc Standard | k or Fast M ed Mode () | Over 4 | DOKHz) |
| Transfer Log | mode | Enable | 1.0 | | * | Master Cloc Standard High-Spe | ik or Fast Mi ed Mode (i | Over 4 | DOKHz) |
| Transfer Log | mode | Enable dir m/s | 1.0 | | * | Master Cloc Standard High-Spe data | k or Fast M ed Mode (r | Over 4 | DOKH2) |
| Transfer Log time 18:37:37,881 | mode i2c i2c | Enable dir m/s | 1.0 | | * | Master Cloc Standard High-Spe data Slave mode | k or Fast M ed Mode (i s Enabled s Frequenc | Over 4 | DOKH2) |
| Transfer Log time 18:37:37,881 18:37:37,897 | mode i2c i2c | Enable dir m/s slave | 1.0 | | * | Master Cloc Standard High-Spe data Slave mode Set I2C Bu | k or Fast M ed Mode (i s Enabled s Frequenc | Over 4 | DOKH2) |
| Transfer Log time 18:37:37,881 18:37:37,897 | mode i2c i2c | Enable dir m/s slave | 1.0 | | * | Master Cloc Standard High-Spe data Slave mode Slave mode Slave mode | k or Fast M ed Mode (Enabled s Frequence Disabled | Over 41 | DOKH2) Mode |
| Transfer Log time 18:37:37,881 18:37:37,897 | mode i2c i2c | Enable dir m/s slave | 1.0 | | * | Master Cloc Standard High-Spe data Slave mode Slave mode Slave mode | k or Fast M ed Mode (i s Enabled s Frequenc | Over 41 | DOKH2) |
| Transfer Log time 18:37:37,881 18:37:37,897 | mode i2c i2c | Enable dir m/s slave | 1.0 | | * | Master Cloc Standard High-Spe data Slave mode Slave mode Slave mode | k or Fast M ed Mode (Enabled s Frequence Disabled | Over 41 | DOKH2) Mode |

Fig 3-3. I2C slave mode

| Slave Select | t | | | Samo | ling Timi | ine | Polarity | Phase | Bit order |
|---------------------------------------|--------|-----------|-------|---------|---------------|-------|--|----------------|-----------|
| @ SS0 (| | D. CCO. L | | (IIII) | | 196 | Positive | | (@) MSB |
| (e) 550 (c) | 1991 (| 0 882 | 0 333 | | | | O Negative | 🔘 Setup | 🖲 LSB |
| PO Option | | | | | | | | | |
| Port0 | | Port1 | | | ort2 | | Port3 | | |
| High Low | | O High | | |) High Low | | High Low | | Set |
| COM | | UN CON | | | U000 | | UWV | | |
| aster Slave | | | | | | | | | |
| Transfer List | | | | | | | Freq | uency 100 | |
| Num 4 | Addr | Dir | Date | | | | | | Stop |
| | 50 | Write | 1.500 | S | 13 04 05 | 06.07 | 08 09 0A 0B | 00.00 | Yes |
| | 50 | Read | | 02 0 | 0 04 00 | 00 01 | 00 00 0H 0D | 00 00 | Yes |
| 0003 5 | 50 | Write | 00 | 01 02 0 | 3 04 05 | 06 07 | 08 09 0A 0B | 0C 0D | Yes |
| | 50 | Read | | | | | | | Yes |
| 0005 5 | 50 | Write | 00 | 01 02 0 | 3 04 05 | 06 07 | 08 09 0A 0B | 0C 0D | Yes |
| 0006 5 | 50 | Read | | | | | | | Yes |
| | | | | | | | | | |
| Send Step | P | Send | 411 | Cor | ntinue | Bu | is Reset | 12 | C Mode |
| ansfer Log | | | | | | | | | |
| time | mode | dir | m/s | freg | addr | size | data | | |
| 18:32:29.454 | | | | noq | GGG | 0120 | I2C Pullup | Enabled | 1 |
| 18:32:29:404 | | | | | | | | er Enabled : { | 50V |
| 18:32:29.485 | | | | | | | Set Interva | | |
| 18:34:35.471 | | write | mas | 100 | 50 | 16 | 00 01 02 03 | 04 05 06 07 | |
| | | | | | | | 1992 - Contract of the second se | lear) | Save Log |

Fig 3-2. I2C master mode

<u>Menu bar</u>

- File(F)
 - Create : Create a new setting file
 - Open : Open a setting file
 - Overwrite : Overwrite a current setting
 - Save a file Save a current setting as a new name
 - End ÷ End application

* All functions except [End] can work on master mode only.

| Edit(E) · Add | : Add a new transfer data to the end of transfer list |
|--------------------|---|
| • Insert | : Insert a new transfer data into the transfer data number selected now |
| • Delete | : Delete a selected transfer list |
| • Erase | : Erase a content of the selected transfer list |
| • Copy | : Copy a content of the selected transfer list |
| • Paste | : Paste a copied content of the transfer list onto a selected number |
| * All functions of | can work on master only. |
| Devices(D) | |
| Switch SPI/I2C | Switch modes between the SPI and I2C bus |

Options(O)

| $Options(\underline{O})$ | |
|--------------------------|---|
| • Setting | : Switch pull-up conditions of the I2C bus signal |
| | Set whether to supply power to devices |
| | Set a voltage of power supply(3.3V, 5.0V) |
| | Set a time interval for each 1 byte |
| • View list/Switch | comints . View list and switch comints |

• View list/Switch scripts : View list and switch scripts

Help(<u>H</u>)

Version information

: Display version of this application

Tool bar

Same as [Create] of [Files] at the menu bar Same as [Open] of [Files] at the menu bar Same as [Overwrite] of [Files] at the menu bar Switch modes between SPI and I2C mode Same as [Add] of [Edit] at the menu bar

Same as [Insert] of [Edit] at the menu bar Same as [Delete] of [Edit] at the menu bar Same as [Erase] of [Edit] at the menu bar Same as [Copy] of [Edit] at the menu bar Same as [Paste] of [Edit] at the menu bar

<u>Controls</u>

SPI Option

- SPI Option
 Slave Select
 Sampling Timing
 Polarity
 Phase
 Set SPI mode
 Set SPI mode
 Set slave select pin
 Select slave select pin
 Set when to sample by which part of a clock
 Select positive polarity or negative polarity
 Select phase. Select sampling or setup.
- Bit order : Select which bit is transmitted first, MSB or LSB.

* Setting of Sampling Timing, Polarity, Phase operate with each other.

GPO Option

- Port0~3 : Set each PORT(Output only)
- High/Low : Set/Display a value at each port
- Set : Output to each Port

Master

| • Transfer List | : Display the content of setting transfer |
|-----------------|--|
| ① Num | : Number of transfer data |
| ② Addr | : Device address |
| 3 Dir | : Direction of transfer. Display Read or Write |
| ④ Line | : Display data line name. Display MOSI or MISO |
| 5 Data | : Display data content |
| 6 Stop | : Display whether to stop condition is issued. |
| ⑦ Size | : Data size |
| • Send | : Transfer selected data |
| • Send All | : Transfer all of setting items at the list view |
| • Continue | : Repeatedly transfer setting items at the list view |
| • Bus Reset | : Issue a bus reset of the I2C bus |

Slave

| • | Slave Address | : Set slave address |
|---|---------------|--|
| | | N.B.) Refer to Page.4-16 for how to appoint an address |
| • | Response Data | : Set data to be returned to a master |
| • | Clear | : Delete returned data |

• Enable : Enable slave operation

Master/Slave common

• Sampling rate : Set/display a sampling rate(frequency).

> You can set a sampling rate(frequency) by 1KHz each. For SPI, a sampling rate(frequency) will be set at an approximate value which can be really set.

(I2C:47KHz - 1MHz / SPI: Up to 12MHz)

* For how to calculate an approximate value,

please refer to a usb61_spi_set_freq0 function at Chapter 4.

| SPI/I2C Bus Clock Frequency | X |
|-----------------------------|---------|
| Set the clock frequency. | |
| 12C Range : 47 to 1000 K | .Hz |
| SPI Range : 1 to 12000 K | Hz |
| Clock Frequency | KHz KHz |
| ОК | Cancel |

: Display a current sampling frequency at the lower left.

: Display a current I2C bus pull-up conditions.

[Setting sample rate]

- : Display a current operating mode Device Mode (SPI Mode or I2C Mode) **Output Volt** : Display a current output at the lower left.
- Freq
- Pull-up

Log

- Transfer Log : Display a log of the content of transfer \bigcirc time : Display time when a log is added(hh:mm:ss:msec) $2 \mod 2$: Display transfer mode for SPI/I2C(SPI/I2C) ③ dir : Display transfer direction(read/write, miso/mosi) ④ m/s : Display master/slave mode(master/slave) 5 freq : Display operating frequency(in KHz) : Display I2C slave address (in Hex number) 6 addr ⑦ size : Display a length of data transfer(in Decimal number) (8) data : Display transfer data(Data after 8 bytes will be omitted)
- Clear
- : Delete the content of transfer log
- Save Log
- : Save a log file(in CSV file format)

3.SPI/I2C Control Utility

Edit window for transfer data

If you double-click a row at the transfer list, the below edit window will be shown.

| Transfer Data Editor | Transfer Data Editor |
|---|---|
| 12C Settings Slave Address 00 H Direction Write Stop Condition Stop | I2C Settings Slave Address 20 H 10bit Direction Write Stop Condition Stop |
| Transfer Data Settings | Transfer Data Settings |
| Transfer Length OK Cancel | Transfer Length 16 OK Cancel |





- I2C setting
 Set transfer setting for the I2C bus
 Slave address
 Set a device address in Hex number
 N.B.) Refer to Page.4-16 for how to set an slave address.
 - ② 10bit : Put a check mark when you set 10 bit address N.B.) Refer to Page.4-16 for how to set an slave address.
 - ③ Transfer direction : Set transfer direction. Set it as Read or Write.
 - ④ Stop condition : Set whether to issue stop condition.

• Transfer data setting : Display a content of transfer data or file name (in Hex number).

- ① Set as a binary \therefore At the edit box, input data which is directly sent
- ② Set from a file : Set data from a binary file
- ③ Select a file Select a binary file
- ④ Length of data transfer : Data size (in Decimal number)

(Max 65535 bytes)

Option setting

You can set the following by selecting $[Option(O)] \rightarrow [Setting]$.

- Set pull-up setting on the I2C bus
- Set to supply power to a target device
- Set an interval between data

| ional Settings Pull-up Control | |
|-----------------------------------|--|
| C Disable | € Enable |
| Power supply to Targ | get Device |
| C Disable | Enable |
| Voltage of Target | 3.3V 💌 |
| Please don't powe | er supply in the case of outside power |
| Interval Setting | |
| During SPI/I2C d each byte. | ata transmission, set the time interval of |
| | des processing time with a thing ance set to a minimum in real time and |
| 0 use | c |
| | OK Cancel |

Fig3-6. Option setting window

- Disable/enable pull-up
 Select whether to set pull-up on the I2C bus line.
 (I2C at 5V, 1MHz pin [401KHz - 1000KHz] only can be selected.)
- Whether to supply power or not Select whether to supply power to a target device. Select from 3.3V or 5.0V.
 N.B. : Don't supply power while an external power supply provide power
- Set an interval

Set a time interval for 1 byte each when sending data.

(3-3) Example to control by using this utility

* The below explanation is an example used ATMEL:AT24C02B、AT25080A

• SPI master mode

[Switch SPI/I2C]

By switching SPI/I2C, SPI Mode can be selected.

[Set a sampling rate(frequency)] Set a sampling rate(frequency) at the Frequency section.

You can set a sampling rate(frequency) by 1KHz each. For SPI, a sampling rate(frequency) will be set at an approximate value which can be really set. (I2C:47KHz - 1MHz / SPI: Up to 12MHz)

For how to calculate an approximate value, please refer to a usb61_spi_set_freq() function at Chapter 4.

[Set to supply power] Supply power by selecting [Option]-[Setting].

[Set an interval] Set a time interval by 1 byte each for sending data.

| PI Option — | | hange SF | PI/12C | ノ |] | | | |
|-----------------|-----|----------|--------|----------|-------------------|------------------------------------|--------------------------|--------------------------|
| Slave Selec | | ° SS2 (| | | ne Timine FLYF | Polarity Positive C Negative | Phase Sample Setup | Bit order MSB CLSB |
| PO Option — | | | | 177.5 | | | | |
| Port0 | | Port1 | | Po | | Port3 | | |
| C High C Low | | C High | | | High Low | C High C Low | | Set |
| laster Slave | 1 | | | | | | | |
| Transfer List | t | | | | (| Frequency | 100 👱 | KH2 |
| Num | ine | Data | | | | | | Size |
| | | | | | | | | |
| Send Ste | P | Send A | 11 _ | Cont | inue | | SF | PI Mode |
| Send Ste | p | Send A | II | Cont | inue | | SF | PI Mode |
| | p | | | Cont | addr size | e data | SF | PI Mode |
| ransfer Log - | | | | | | e data | SF | PI Mode |
| ransfer Log - | 1 | | | | | | Clear | PI Mode |

| ptional Settings | 1 | - | × |
|---|--------------------|------------------|--------|
| Pull-up Control | | | |
| C Disable | 🕫 Enable | | |
| Power supply to Targe | t Device | | |
| C Disable | Enable | | |
| Voltage of Target | 3.3V 💌 | | |
| Please don't power | supply in the eas | e of outside po | ower |
| _ Interval Setting | | | |
| During SPI/I2C dat each byte. | a transmission, se | t the time inter | val of |
| This setting include guaranteeing distar | | | and |
| 0 usec | | | |
| | C | ж [[| Cancel |
| | | | |

Fig3-8. Setting to supply power

Example : Write 11 22 33 44 55 66 77 88 at 50h, and write 8bytes data from 50h.

[Data input(Write / Read)] Double-click an inside of [Transfer List] and input a Hex number. Please see the below example.

(1 row)06h --- Set Write Enable bit (2 row)02h --- Write command 00h 50h --- Address where data is written 11h 22h.. --- Data to be written (8byte) (3 row)03h --- Read command 00h 50h --- Address to be read 00h 00h --- Dummy data for Read(8byte) (8byte data will be read)



Fig3-9. data input

SPI/I2C Device Control Utility for REX-USB61

[Execution(Write / Read)]

By clicking the [Send All] button, the data inside of the [Transfer List] will be sent.

The sent/received data will be displayed at the [Transfer Log].



Fig3-10. Execution

| Switch SPI/I2C] | 😸 SPI/I2C Device Control Utility for REX-USB61 |
|---|---|
| | File(F) Edit(E) Device(D) Option(O) Help(H) |
| Set a sampling rate(frequency)] | |
| <u>Set to supply power</u> | SPI Option Slave Select Sampling Timing Polarity Phase Bit order |
| Set an interval] | C SSD |
| ike the procedure described at | GPO Option |
| 1 | Port0 Port1 Port2 Port3 |
| age.3-8,switch modes into I2C Mode | CHigh CHigh CHigh CLow CLow CLow Set |
| nd set a sampling rate(frequency) / | Master Slave |
| ower supply / interval. (Master tab | |
| hould be selected) | Transfer List Frequency 100 KHz |
| | Num Addr Dir Data Stop |
| 1 Slave address (R/W bit is not included) | Num Addr Dir Data Stop |
| (R/W bit is not included) | Num Addr Dir Data Stop |
| (R/W bit is not included) | Num Addr Dir Data Stop |
| (R/W bit is not included) Example : In case of 50h] | Send Step Send All Continue Bus Reset I2C Mode |
| (R/W bit is not included) Example : In case of 50h] | Send Step Send All Continue Bus Reset I2C Mode |
| (R/W bit is not included) Example : In case of 50h] | Send Step Send All Continue Bus Reset I2C Mode Transfer Log [time mode dir m/s freq addr size data [20302167 i2c |
| (R/W bit is not included) Example : In case of 50h] | Send Step Send All Continue Bus Reset I2C Mode Transfer Log Time mode dir m/s freq addr size data |
| (R/W bit is not included) Example : In case of 50h] | Send Step Send All Continue Bus Reset I2C Mode Transfer Log Image: Send All Continue Bus Reset I2C Mode Transfer Log Image: Send All Continue Bus Reset I2C Mode Transfer Log Image: Send All Continue Bus Reset I2C Mode Transfer Log Image: Send All Image: Send All Image: Send All Z03302167 Image: Send All Image: Send All Image: Send All |
| Example : In case of 50h] | Send Step Send All Continue Bus Reset I2C Mode Transfer Log Image: Send All Continue Bus Reset I2C Mode Transfer Log Image: Send All Continue Bus Reset I2C Mode Transfer Log Image: Send All Continue Bus Reset I2C Mode Transfer Log Image: Send All Image: Send All Image: Send All Z03302167 Image: Send All Image: Send All Image: Send All |
| (R/W bit is not included) Example : In case of 50h] 0 1 0 1 0 0 0 R/W | Send Step Send All Continue Bus Reset I2C Mode Transfer Log Imme mode dir m/s freq addr size data 203302167 i2c I2C Pullup Enabled Target Power Enabled 50V 203302183 Continue Set Interval : 0 usec |

Example : Write 11 22 33 44 at the address of 00h for a device of slave address 50h, and write 4byte data from the address of 00h for the device of address 50h.

[Data input(Write / Read)]

Double-click an inside of [Transfer List]

- and set each item.
- slave address --- Set 7bit
- *1 For setting, refer to [Slave address], as described above.

(If you set data as 10bit, put a check mark at the [10bit])

• Transfer direction --- Select Write / Read

• Stop condition --- Set an issue of stop condition

• Transfer data --- Set data in Hex number

• Length of data transfer --- When writing, the length of data transfer will be displayed automatically and when reading, set data size which will be read.(Unit:Byte)

| Slave Address | 50 н Г 10Ыт | |
|---------------------------------|-------------|---|
| Direction Wri | te 💌 | |
| Stop Condition Sto | p 💌 | |
| Fransfer Data Setting Binary | r File | |
| 10 11 22 33 44 | | * |
| | | |
| | | Ŧ |
| Transfer Length | 5 | Ŧ |

Please see the below sample.

(1 row) < Write >

- Slave address --- 50 h (Set as 7bit)
- Data direction --- Write
- Stop condition ---Yes
- Transfer data
 00h --- Address to be written
 11h 22h.. --- Data to be written
- (2 row) <Write for Read>
- Slave address --- 50 h (Set as 7bit)
- Data direction --- Write
- Stop condition --- No
- Transfer data 00h --- Address to be read

(3 row) <Read>

- Slave address --- 50 h (Set as 7bit)
- Data direction --- Read
- Stop condition --- Yes
- Length of data transfer --- 4 (This won't display)

| 🕫 SS0 | C SS1 (| nsszinis | | ling Timing | Polarity Positive Negative | | |
|--------------------------|---|--------------------------|------------|--------------------------|----------------------------------|-----------------------------|---------------------|
| GPO Option | | | | | | | |
| Port0 ⊂ High € Low | | Port1 C High @ Low | (| 'ort2 ⊂ High € Low | Port3 ⊂ High ເ⊂ Low | | Set |
| Master Sla | ave | | | | | | |
| Transfer L | ist | | | | Frequency 1 | 00 🗸 | KHz |
| Hum | Addr | | Data | | | | Stop |
| 0001 | 50 50 | | 00 11 22 3 | | | | Yes |
| 0003 | 50 | Write Read | 00 | | | | No Yes |
| Send S | 50 | | | ntinue | Bus Reset | | Yes |
| Send S | 50 | Read | | | Bus Reset | | No |
| | 50 | Read | Con | | | | Yes |
| Send S | 50 Step @ | Read Send All | Con | ntinue | e data 12C Pullup | Enabled ver Enabled : 5. | No Yes O Mode |

Fig3-13. Data input

[Execution(Write / Read)]

By clicking the [Send All] button, data inside of the [Transfer List] will be sent.

The sent/received data will be displayed at the [Transfer Log].

| Slave Sel | | | | | | | | |
|--|----------------------------------|---------------|--------------------|------------|---------|--|----------------------------|-----------|
| blave bel | ect | | Sa | mpling Tim | | Polarity | Phase | Bit order |
| C 880 | C SS1 | C SS2 C | SS3 🗄 | | | Positive Negative | C Sample | C MSB |
| O Option | | | | | | | | |
| Port0 | | Port1 | | Port2 | | Port3 | | |
| C High | | C High | | C High | | C High | | 0.1 |
| • Low | | | | ← Low | | C Low | | Set |
| ster Sla | ive | | | | | | | |
| ransfer L | ist | | | | | Frequency 1 | 00 👱 | KHz |
| Num | Addr | Dir | Data | | | | | Stop |
| 0001 | 50 | Write | 00 11 2 | 2 33 44 | | | | Yes |
| 0002 | 50 50 | Write Read | 00 | | | | | No Yes |
| | | | | | | | | |
| C 10 | itep 🚺 | Send All | | Continue | В | is Reset | 12 | C Mode |
| Send S | | | | | 305 | | | |
| ansfer Lo | | dır r | n/s fre | q addr | size | data | | |
| ansfer Loe ime | mode | | | | | Set Interva | I:0 usec is Frequency : | 100/10 |
| insfer Lor ime 8-37:26:49 | 94 | | | | 5 | 00 11 22 3 | | |
| insfer Lo ime 0:37:26:49 0:37:39:09 | 94 99 i2c | write r | nas 100 | 1 50 | | | | |
| insfer Lo ime 0:37:26.49 0:37:39.09 0:37:39.11 0:37:39.13 | 94 99 i2c 14 i2c 30 i2c | | nas 100 nas 100 | 50 | 1 | 00 | | |
| insfer Log ime 0:37:26.49 0:37:39.09 0:37:39.11 | 94 99 i2c 14 i2c 30 i2c | write r | | 50 | | 00 0A 2F 2F 3 | 2F | |

Fig3-14. Execution

• I2C slave mode

| Example : Read data transferred to slave | e address 50h |
|---|---|
| [Switch SPI/I2C] | SPI/I2C Device Control Utility |
| [Set a sampling rate(frequency)] | File(E) Edit(E) Device(D) Option(D) Help(H) Image: |
| [Set to supply power] [Set an interval] | SPI Option. Slave Select Sampling Timing Polarity Phase Bit order Image: State St |
| Like the procedure described at Page.3-8,switch modes into I2C Mode | GPO Option Port0 Port1 Port2 Port3 C High C High C High C High C Low C Low C Low Set Master Slave |
| and set a sampling rate(frequency) / power supply/interval. (Master tab should be selected) | Response Data |
| | Standard or Fast Mode (Up to 400KHz) C High-Speed Mode (Over 400KHz) |
| Set a slave address as 7bit at the [Slave Address] and click the [Enable] button. | Clear Stop Transfer Log |
| When data is sent from a master, Read data will be displayed at the [Response Data] and [Transfer log]. | Clear Save Log Output Volt 50V Freq 100 KHz Pullup - Enabled |

Fig3-15. I2C slave mode setting
(3-4) Grammar for script description

This utility can treat a script file where a description of a device access is written. You can describe a comment at the script file. By using the script file, you can access a device. You need to describe a script by the following rule:

◆ Common command for SPI/I2C

The following is a common command for both SPI and I2C.

• <u>Definition of values</u>

Describe values as Decimal number or Hex number.

• The values ranges from $0\sim 65536$ and if the values is Hex number, put [h] or [H] at the end of the values. If you describe the values consecutively, put [,] between a value and another value.

• Definition of characters

This script file doesn't distinguish a small letter and a large letter of alphabet. You may write Japanese comments.

• <u>Grammar</u>

Be sure to put space(one-byte) or TAB between a command and another command, or a value and another value.

Two-byte characters are a grammatical error.

| Command | # | | | | | | | |
|-----------|---|--|--|--|--|--|--|--|
| Meaning | The sentence after [#] is treated as a comment. | | | | | | | |
| Parameter | None | | | | | | | |
| | | | | | | | | |
| Command | MODE= | | | | | | | |
| Meaning | Set mode for SPI, or I2C. | | | | | | | |
| | There is not a default value for this MODE command. | | | | | | | |
| | If you don't set a mode, it is a grammatical error. | | | | | | | |
| | After setting, you can not change modes halfway. | | | | | | | |
| Parameter | SPI | | | | | | | |
| | I2C | | | | | | | |
| | | | | | | | | |
| Command | FREQUENCY= | | | | | | | |
| Meaning | Set a sampling rate(frequency). | | | | | | | |
| | You can set a sampling rate(frequency) by 1KHz each. | | | | | | | |
| | For SPI, a sampling rate(frequency) will be set at an approximate value which | | | | | | | |
| | can be really set. (I2C:47KHz - 1MHz / SPI: Up to 12MHz) | | | | | | | |
| | * For how to calculate an approximate value, please refer to a | | | | | | | |
| | usb61_spi_set_freq0 function at Chapter 4. | | | | | | | |
| | If you don't set Frequency, the following is a default value. | | | | | | | |
| | Mode Sampling rate (Frequency) | | | | | | | |
| | SPI 100KHz | | | | | | | |
| | I2C 100KHz | | | | | | | |
| | You can change sampling rate(frequency) any time. | | | | | | | |
| Parameter | You can set the below setting for SPI,I2C. | | | | | | | |
| | Mode Setting value | | | | | | | |
| | SPI 1 - 12000 | | | | | | | |
| | I2C 47 - 1000 | | | | | | | |

Fig.3-2 Common command table

| Command | INTERVAL= | | | | | | | |
|--|--|--|--|--|--|--|--|--|
| Meaning | Set an interval of waiting time which is inserted into bytes of data to be sent.(Unit: micro second) | | | | | | | |
| | If you don't set this value, a default value is 0. | | | | | | | |
| Parameter | Parameter Set a value ranging from 0 to 65535.(0 – 65535 micro seconds) | | | | | | | |
| * An actual interval time include process time, so it will be longer than the interval set here. | | | | | | | | |

| Command | POWER= | | | | | | | | |
|--------------------|--|--|--|--|--|--|--|--|--|
| Meaning | Supply power set at a parameter. | | | | | | | | |
| | You can change power supply any time. | | | | | | | | |
| | If you don't set this item, a default value is Output OFF. | | | | | | | | |
| Parameter | Output Setting value | | | | | | | | |
| | Output OFF OFF | | | | | | | | |
| | Output 3.3V ON3 | | | | | | | | |
| | Output 5.0V ON5 | | | | | | | | |
| Command | WAIT= | | | | | | | | |
| Meaning | Set a waiting time until a next command is executed. | | | | | | | | |
| liteaning | Unit is 100 milliseconds. (100 milliseconds - 60 seconds) | | | | | | | | |
| Parameter | Set a value ranging from 1 to 600. (100milliseconds - 60 seconds) | | | | | | | | |
| | | | | | | | | | |
| Command | REPEAT=nn | | | | | | | | |
| Meaning | Repeat a command written in {} after REPEAT command by a number set in | | | | | | | | |
| | this command. | | | | | | | | |
| | If this {} is not written, only a next command written right after this command | | | | | | | | |
| | will be repeated. * For how to use this command, please refer to Page.3-17. | | | | | | | | |
| Parameter | nn=1 - 65536{} | | | | | | | | |
| Farameter | IIII-1 - 05050() | | | | | | | | |
| Command | PULLUP= | | | | | | | | |
| Meaning | Set a pull-up setting for SDA, SCL signal line. | | | | | | | | |
| | A default value is pull-up(ON), and as long as voltage of power | | | | | | | | |
| | supply:5V,frequency:1MHz is set, you can set off pull-up. | | | | | | | | |
| Parameter | ON or OFF | | | | | | | | |
| Common d | FILEn | | | | | | | | |
| Command Meaning | Set a file number as n, and you can set 5 files at maximum. | | | | | | | | |
| Meaning | From a file embraced by "(double-quotation), data will be sent/received. Data is | | | | | | | | |
| | treated as binary data. | | | | | | | | |
| | To set a file, appoint a file name instead of path. Please note error happens if the | | | | | | | | |
| | file doesn't exist at the same directory when sending data. When receiving data, | | | | | | | | |
| | a new file will be created. | | | | | | | | |
| Parameter | n=1 - 5 | | | | | | | | |
| | "file name" | | | | | | | | |
| L | | | | | | | | | |

| Command | END |
|-----------|---|
| Meaning | Execute the script until END. |
| | The content described after END isn't executed. |
| | (Reading script will stop at END command) |
| Parameter | None |

• Command for I2C only

| | Fig.3-3 I2C command table | | | | | | | |
|----------------------|--|--|--|--|--|--|--|--|
| Command | ADDRESSMODE= | | | | | | | |
| Meaning | Set I2C address to 7 bit mode or 10 bit mode. (Default value is 7 bit mode) | | | | | | | |
| Parameter | 7 or 10 | | | | | | | |
| Command | ADDRESS= | | | | | | | |
| Meaning | Set I2C address | | | | | | | |
| 0 | You can change address any time, but if [READ] or [WRITE] command is | | | | | | | |
| | written before setting an address, it is a grammatical error. | | | | | | | |
| Parameter | 0-1023 | | | | | | | |
| Command | READ | | | | | | | |
| Meaning | Read bytes set by this command. | | | | | | | |
| Parameter | xxH | | | | | | | |
| | Set bytes which will be read. | | | | | | | |
| | 1 - 65536 | | | | | | | |
| Command | READF | | | | | | | |
| Meaning | Read bytes set by this command and save the data as a file. | | | | | | | |
| | The data is saved as a file name described as FILEn. | | | | | | | |
| | Add the data to the existing file if a file already exist. | | | | | | | |
| Parameter | xxH FILEn | | | | | | | |
| | Set bytes to be read, a file to be saved (Bytes:1 - 65536) | | | | | | | |
| Command | WRITE | | | | | | | |
| Meaning | Write data set by this command. If more than one data need to be written, | | | | | | | |
| D (| separate each data by comma. | | | | | | | |
| Parameter | xxH, xxH, Set data to be written by bytes. | | | | | | | |
| ~ | | | | | | | | |
| Command | WRITEF Sand data from a file. Data is treated by a binary data | | | | | | | |
| Meaning | Send data from a file. Data is treated by a binary data. Data to be written is read from a file specified as FILEn. | | | | | | | |
| Parameter | FILEn | | | | | | | |
| 1 arameter | Set a file to be written | | | | | | | |
| 0 | | | | | | | | |
| Command Meaning | Send stop bit. | | | | | | | |
| Parameter | None | | | | | | | |
| | | | | | | | | |
| Command | RESET Issue a reset to bus(send STOP bit) | | | | | | | |
| Meaning Parameter | None | | | | | | | |
| | | | | | | | | |
| Command | GPO= | | | | | | | |
| Meaning | Set a port output to DO0 - DO3(#13 - #16 pin) | | | | | | | |
| Parameter | Set 1 for output bit. | | | | | | | |
| | Set 0 - 15 when describing in Decimal number. | | | | | | | |
| | Set 0h - Fh when describing in Hex number. | | | | | | | |
| | Bit3 Bit2 Bit1 Bit0 | | | | | | | |
| | Bit3Bit2Bit1Bit0DO3DO2DO1DO0 | | | | | | | |
| | | | | | | | | |

◆ Command for SPI only

| Fig.3-4 SPI co | mmand table |
|----------------|-------------|
|----------------|-------------|

| Command | SS=n | | | | | | | |
|-----------|---|---------------------------------|--------------|---------|---|--|--|--|
| Meaning | Set slave select pin. Default value is 0. | | | | | | | |
| | Parameter 0 1 2 3 | SSx SS0 SS1 SS2 SS3 | | | | | | |
| Parameter | n=0 - 3 | | | | | | | |
| Command | SAMPLING=1 | ı | | | | | | |
| Meaning | Set bus sampl | | Default valu | e is 0. | | | | |
| | Parameter | Samplin | ng edge | Figure | 1 | | | |
| | 0 | Rising edge | | | | | | |
| | 1 | Falling edge | е | | | | | |
| | 2 | Falling edge | е | | | | | |
| | 3 | Rising edge | | | | | | |
| Parameter | n=0 - 3 | | | | | | | |
| Command | FB= | | | | | | | |
| Meaning | Set a first bit. Default value is MSB. | | | | | | | |
| Parameter | MSB or LSB | | | | | | | |
| Command | SSSET | | | | | | | |
| Meaning | Set Low for slave select signal set by SS command. | | | | | | | |
| Parameter | None | | | | | | | |
| Command | SSRESET | | | | | | | |
| Meaning | Set High for slave select signal set by SS command. | | | | | | | |
| Parameter | None | | | | | | | |

| Other functions | | | | | | |
|-----------------|---|--|--|--|--|--|
| For SPI, there | e isn't any particular command for Read/Write, and write a described value. SPI | | | | | |
| by its nature | write and read at the same time, so to read only isn't allowed. | | | | | |
| Meaning | Read data is saved as a file set in FILEn.(If FILEn is specified) | | | | | |
| | Add data if an existing file is set. | | | | | |
| Parameter | xxH, xxH, FILEn | | | | | |
| | Set data to be written by bytes, and data to be read is saved as a | | | | | |
| | File. | | | | | |
| Meaning | Write data from a file set by FILEm. | | | | | |
| | Read Data is saved as a file set by FILEn.(If FILEn is specified) | | | | | |
| | Add data if an existing file is set. | | | | | |
| Parameter | FILEm FILEn | | | | | |
| | Set data to be written as a file, and save data to be read as a file. | | | | | |

♦ How to use REPEAT command

This section explains REPEAT script and inside process of $\{\ \}$ and STOP.

| Script code | Explanation of function |
|---------------------------------------|---|
| REPEAT=10 READ 1 STOP | After receiving 10 bytes of data, STOP condition is sent. |
| REPEAT=10 { READ 1 STOP } | Repeat the following 10 times: [Send STOP condition by 1 byte each] |
| REPEAT=10 READ 1 STOP | STOP condition is sent after receiving 10 bytes of data. |
| REPEAT=10 { READ 1 } STOP | STOP condition is sent after receiving 10 bytes of data. |
| REPEAT=10 { READ 1 STOP } | Repeat the following 10 times: [STOP condition is sent after receiving 1 byte of data] |

(3-5) Example of script

The below is an explanation of how to use a script file.

From [Option]-> [List View/Script Change], show script description mode.

The function of each button is as follows: [Load] --- Read a script file. [Save] --- Save a script file. [Clear]--- Erase a shown content. [Execute]---Execute a script. [Stop]--- Stop executing script.

A result of execution shows at [Transfer Log].

| Transfer Log time mode dir. m/6 frey addr. size data 16/4319522 fi2c read mas 100 50 4 83 67 83 43 r64312538 fi2c write mas 100 50 9 08 00 01 02 03 (figure 100) | rice | e C | ont | roll | Jtility | for RE | X-USB6 | 1 | 1 | |
|---|-----------------------|-----------|---------------------|------------|---------|---------------------------|-------------------------------------|-----------------------|-----------------|--|
| Image: Sample of I2C Script AMME1: AT24C01A Serial EEPROM Input/Output MODE=12C Image: Strength Serial EEPROM Input/Output MODE=12C "write.bin" Ital file to save received data INTERVAL=20 INTERVAL=20 Interval between data transmiss FREQUENCY=100 POWER=0NS External power supply = 5V PULLUP=0N ADDRESSMODE=7 ADDRESSMODE=7 ADDRESSMODE=7 ADDRESS=50h If From here, access to a device If Write 4 bytes data from address 0008h WRITE 08h WRITE 08h WRITE 08h WRITE 08h If Read 4 bytes MUTH 08h WRITE 08h WRITE 08h WRITE 08h WRITE 08h If Read 4 bytes If Read 4 bytes Image: Confirm data is written properly If Read 4 bytes Image: Confirm data is written properly If Read 4 bytes Image: Confirm data is written properly Image: Confirm data is written properly |) | D | evic | e(D) |) 0 | ption((|) Hel | р(<u>Н</u>) | | |
| i ATMEL: AT24C01A Serial EEPROM Input/Output dODE=12C "read.bin" # I2C mode "ILE1 "read.bin" # file to be sent "ILE2 "read.bin" # file to save received data INTERVAL=20 # Time interval between data transmiss FREQUENCY=100 # Set sampling rate(frequency) at 100Kl POWEN=ON6 # External power supply = 5V PULLUP=ON # SCL SOA line pull-up ADDRESSMODE=7 # Address mode = 7bit ADDRESS=50h # Slave address = 08h # Write 4 bytes data from address 0008h WRITE 08h # Write adata STOP # Confirm data is written properly # Read 4 bytes 4 Unite 108h # Read address = 08h READ 4 # Read 4 bytes 4 Unite 108h # Read address = 08h READ 4 # Read 4 bytes 4 Unite 108h # Read address = 08h READ 4 # Read 4 bytes 4 Unite 108h # Read address = 08h READ 4 # Read 4 bytes 4 Unite 108h # Read address = 08h READ 4 # Read 4 bytes 4 Unite 108h # Read 4 bytes | 6 | | ÷ | 噑 | = | <i>4</i> | | | | |
| ILE1 "write.bin" # data file to be sent ILE2 "read.bin" # file to save received data INTERVAL=20 # Time interval between data transmiss FREQUENCY=100 # Set sampling rate(frequency) at 100Kl POWER=005 # External power supply = 5V PULLUP=0N # SQL_SOA line pull=up ADDRESSMODE=7 # Address mode = 7bit ADDRESS=50h # Slave address = 08h Write 4 bytes data from address 0008h Write address = 08h WRITE 08h # Write data STOP # STOP # Confirm data is written properly # Read address = 08h WRITE 08h # Read address = 08h READ # Read address = 08h Transfer Log Load Save Clear Exe Transfer Log Im mas 100 50 4 83 67 83 43 | | | | | al EE | | | | | |
| FREQUENCY=100 # Set sampling rate(frequency) at 100KI POWER=0N5 # External power supply = 5Y PULLUP=0N # Sol_SDA line pull-up ADDRESS=500 # Sol_SDA line pull-up ADDRESS=500 # Slave address = 50h # From here, access to a device # Write 4 bytes data from address 0008h WRITE 08h WRITE 08h WRITE 08h WRITE 100 koln, 02h, 03h # STOP # Confirm data is written properly # Read 4 bytes V # WRITE 08h # Read 4 bytes * * * * * * | it: ad | e.b | oin″ In″ | | | # | data f | ile to | | |
| POWER-ON6 | ER' | VAL | =20 | | | # | Time i | nterval | betwe | een data transmission = 20 micro se |
| ADDRESS=50h | ER | =01 | 15 | 00 | | # | Extern | al powe | r supp | bly = 5V |
| # # Write 4 bytes data from address 0008h WRITE 08h # Write address = 08h WRITE 00h,01h,02h,03h # Write data STOP # Confirm data is written properly # Read 4 bytes data from address 0008h WRITE 08h # Read address = 08h READ 4 # Read 4 bytes | | | | | | ŧ | Addres Slave | s mode address | = 7bit = 50h | 1 |
| WRITE 08h # Write address = 08h WRITE 00h,01h,02h,03h # write data STOP # Confirm data is written properly # Read 4 bytes data from address = 000h WRITE 08h # Read address = 08h READ 4 # Read 4 bytes < | ro | m F | nere | , ac | ccess | : to a | device | | | |
| Load Save Clear Exe Transfer Log | P ont eai TE | fir d4 | rm d 4 by 08h | ata tes | is v | # vritten from # | STOP proper address Read a | ly 0008h ddress | = 08h | |
| Transfer Log mode dir. m/s freq addi size dets 16/3/12/822 12c read mas 100 50 4 83 67 83 43 16/4/3/12/838 12c read mas 100 50 9 08 00 01 02 03 0 | | | | | | | | m | | |
| 164312522 12c read mas 100 50 4 83 67 83 43 164312538 i2c write mas 100 50 9 08 00 01 02 03 0 | n | | Ç | | Loa | | Sa | ave | | Clear Execute Sto |
| 16:43:12:538 i2c write mas 100 50 9 08:00:01:02:03:0 | | | | d | ir | m/o | freq | addr | size | data |
| | | | | | | | | | | 83 67 83 43 08 00 01 02 03 04 05 06 |
| | 53 | i2 | 2c | W | rite | mas | 100 | 50 | 1 | 08 |
| 16:43:12:569 i2c read mas 100 50 4 2C 20 83 41 10:42:12:569 Target Power Di | | 12 | 2c | re | ead | mas | 100 | 50 | 4 | 2C 20 83 41 Target Power Disabled |
| Clear | - | | | | | | | | | Clear Save Lo |
| | | | | | | | | | | |

Fig3-16. Example of script

* You can make and edit a script file with a text editor because script files are text files.

The below describes script samples(Write/Read) to control an I2C and SPI device. (The script files are included at the bundled CD-ROM. I2C_script.txt/ SPI_script.txt)

• <u>Sample of I2C script</u>: (Write 4bytes data(00h 01h 02h 03h) from the address of 08h of the device at slave address 50h, and read the data to confirm the data is written properly. And also, write data on the file, and read the data to a file to confirm the data is written. Sampling rate(frequency) is 100KHz/External power supply is 5V)

| # Sample of I2C script | | | | | | |
|--|--|--|--|--|--|--|
| # ATMEL: AT24C01A Serial EEP MODE=I2C | ROM Input/Output #I2C mode | | | | | |
| MODE-120 | # 120 mode | | | | | |
| FILE1 "write.bin" | # data file to be sent | | | | | |
| FILE2 "read.bin" | # file to save received data | | | | | |
| INTERVAL=20 | # Time interval between data transmission 20μ sec. | | | | | |
| FREQUENCY=100 | # Set sampling rate(frequency) at 100KHz. | | | | | |
| POWER=ON5 | # External power supply 5V | | | | | |
| PULLUP=ON | # SCL,SDA line pull-up | | | | | |
| ADDRESSMODE=7 | # Address mode 7 bit | | | | | |
| ADDRESS=50h | # Slave address 50h | | | | | |
| # | | | | | | |
| # From here, access to a # | device | | | | | |
| ** | | | | | | |
| # Write 4 bytes data from | | | | | | |
| WRITE 08h | # Write address 08h | | | | | |
| WRITE 00h,01h,02h,03 STOP | # STOP | | | | | |
| 5101 | # 5101 | | | | | |
| # Confirm data is writte | | | | | | |
| # Read 4 bytes data from | | | | | | |
| WRITE 08h | # Read address 08h | | | | | |
| READ 4 | # Read 4 bytes | | | | | |
| STOP | # STOP | | | | | |
| # Read data of FILE1 fro | | | | | | |
| WRITE 08h | # Write address 08h | | | | | |
| WRITEF FILE1 STOP | # Write data(4 bytes of binary data at FILE1) # STOP | | | | | |
| 5101 | # 5101 | | | | | |
| # Confirm data is writte | | | | | | |
| | ddress 0008h onto FILE2 | | | | | |
| WRITE 08h READF 04h FILE2 | # Read address 08h # Conv. 4 butes read data onto FU F2 | | | | | |
| STOP | # Copy 4 bytes read data onto FILE2 # STOP | | | | | |
| POWER=OFF | # External power supply 0V | | | | | |
| END | | | | | | |
| | | | | | | |

• Sample of SPI script : (Write 4bytes data(00h 01h 02h 03h) from the address of 15h, and read the data to confirm the data is written properly. And also, write data on the file, and read the data to a file to confirm the data is written. Sampling rate(frequency) is 3MHz/External power supply is 5V)

| # Sample of SPI s | | |
|-------------------|-----------------------|--|
| # ATMEL: AT250 | 980 Serial EEPROM | Input/Output |
| MODE=SPI | # SPI mode | 9 |
| FILE1 | "write.bin" | # data file to be sent |
| FILE2 | "read.bin" | # file to save received data |
| POWER | =ON5 | # External power supply 5V |
| INTERV | /AL=20 | # Time interval between data transmission 20μ sec. |
| FREQU | ENCY=3000 | # Set sampling rate(frequency) at 3MHz. |
| SAMPL | | # Specify an edge to renew data |
| FB=MSI | В | # Set a bit order |
| SS=0 | | # Select slave select pin as 0 |
| # | | |
| | here, access to a dev | ice |
| # | | |
| # Write | 4 bytes data from ad | dress 0015h |
| SSSET | | # Activate SS signal at the Low level |
| 06h | | # Operation code WREN |
| SSRESE | ET | #Activate SS signal at the High level |
| SSSET | | #Activate SS signal at the Low level |
| 02h, 15h | | # Operation code WRITE $+$ write address |
| | ,02h,03h | # Write data |
| SSRESE | ΣT | # Activate SS signal at the High level |
| # Confir | m data is written pr | operly |
| # Read 4 | bytes data from ad | dress 0015h |
| SSSET | | #Activate SS signal at the Low level |
| 03h,15h | | # Operation code READ+ read address |
| REPEAT | [=4 | # Repeat the next command 4 times |
| 00h | | # Read 1 byte of dummy data |
| SSRESE | ET | # Activate SS signal at the High level |
| # Write | data on FILE1 from | address 0015h |
| SSSET | | #Activate SS signal at the Low level |
| 06h | | # Operation code WREN |
| SSRESE | ΣT | # Activate SS signal as the High level |
| SSSET | | # Activate SS signal at the Low level |
| 02h, 15h | ,00h | # Operation code WRITE+ write address |
| FILE1 | | # Write data on FILE1 |
| SSRESE | ΣT | #Activate SS signal at the High level |
| | | # (Continue to the following page) |

| # (Continue from the previous page |) | | | | |
|--------------------------------------|--|--|--|--|--|
| # Confirm data is written p | # Confirm data is written properly | | | | |
| # Copy data read from add | ress 0015h onto FILE2 | | | | |
| SSSET | # Activate SS signal at the Low level | | | | |
| 03h,15h,00h | # Operation code READ+ read address | | | | |
| FILE1 FILE2 | # Write dummy data from FILE1 | | | | |
| | # Save date read to FILE2 | | | | |
| SSRESET | SSRESET # Activate SS signal at the High level | | | | |
| POWER=OFF # External power supply 0V | | | | | |
| END | | | | | |
| | | | | | |

4. API function reference

(4-1) Using on VC

This API functions is a library software to support software development using REX-USB61.

By using the API functions, it will be possible to incorporate the application program own control SPI/I2C target device.

The header file (usb61def.h) and the library file (usb61api.lib, usb61spi.dll) are provided to use the library functions on VC++.

Add these files to your project, then call the library functions.

The declaration of importing library functions is as follows (excerpt from usb61def.h):

* For a description of user defined types, please see the header file usb61def.h.

#define USB61LIB_API __declspec(dllimport)

USB61LIB_API HANDLE WINAPI usb61_open(RS_STATUS *pStatus);

USB61LIB_API RS_STATUS WINAPI usb61_close(HANDLE hUsb61Device);

USB61LIB_API RS_STATUS WINAPI

usb61_power_control(HANDLE hUsb61Device, UINT fPowerState);

USB61LIB_API RS_STATUS WINAPI

 $usb61_mode_change(\ HANDLE\ hUsb61Device,\ UINT\ fDeviceMode,$

USHORT i2cSlaveAddr);

USB61LIB_API RS_STATUS WINAPI

usb61_set_interval(HANDLE hUsb61Device, USHORT IntervalCnt);

USB61LIB_API RS_STATUS WINAPI

usb61_gpo_write(HANDLE hUsb61Device, UINT fPortVal);

USB61LIB_API RS_STATUS WINAPI

usb61_get_fw_version(HANDLE hUsb61Device, UCHAR* pFWMajorVer, UCHAR* pFWMinorVer);

USB61LIB_API RS_STATUS WINAPI

usb61_get_dll_version(HANDLE hUsb61Device, UCHAR* pDllMajorVer, UCHAR* pDllMinorVer);

USB61LIB_API RS_STATUS WINAPI

usb61_get_hw_info(HANDLE hUsb61Device, RS_HARDWARE_INFO pHardwareInfo); USB61LIB_API RS_STATUS WINAPI

usb61_i2c_pullup(HANDLE hUsb61Device, RS_I2C_PULLUP fI2cPullup); USB61LIB_API RS_STATUS WINAPI usb61_i2c_bus_reset(HANDLE hUsb61Device); (Continue to the following page) USB61LIB_API RS_STATUS WINAPI usb61_i2c_set_freq(HANDLE hUsb61Device, RS_I2C_FREQ fI2cFreq); USB61LIB_API RS_STATUS WINAPI usb61_i2c_set_freq_ex(HANDLE hUsb61Device, USHORT Frequency, USHORT *pActualFrequency); USB61LIB_API RS_STATUS WINAPI usb61_i2c_read_master(HANDLE hUsb61Device, USHORT SlaveAddress, UINT fI2cOption, USHORT ReadBytes, UCHAR *pReadBuf); USB61MLIB_API RS_STATUS WINAPI usb61_i2c_read_master_ex(HANDLE hUsb61Device, USHORT SlaveAddress, UINT fI2cOption, USHORT ReadBytes, UCHAR *pReadBuf); USB61LIB_API RS_STATUS WINAPI usb61_i2c_write_master(HANDLE hUsb61Device, USHORT SlaveAddress, UINT fI2cOption, USHORT WriteBytes, UCHAR *pWriteBuf); USB61LIB_API RS_STATUS WINAPI usb61_i2c_read_slave(HANDLE hUsb61Device, RS_NOTIFY_TYPE nType, void (CALLBACK EXPORT* lpfnReadEvent) (USHORT ReadBytes, UCHAR *pReadBuf), HWND hWnd); USB61LIB_API RS_STATUS WINAPI usb61_i2c_set_response_data(HANDLE hUsb61Device, USHORT ResponseBytes, UCHAR *pResponseBuf); USB61LIB_API RS_STATUS WINAPI usb61_spi_set_freq(HANDLE hUsb61Device, UINT fDataMode, USHORT Frequency, USHORT *pActualFrequency); USB61LIB_API RS_STATUS WINAPI usb61 spi transmit master(HANDLE hUsb61Device, RS SPI SS fSlaveSelect, USHORT TransmitSize, UCHAR *pSendBuf, UCHAR *pRecvBuf); USB61LIB API RS STATUS WINAPI usb61 spi transmit master hold ss(HANDLE hUsb61Device, RS_SPI_SS fSlaveSelect, USHORT TransmitSize, UCHAR *pSendBuf, UCHAR *pRecvBuf);

(4-2) Using on VB / Visual C#

To use an ActiveX component that is attached to a product from application of Visual C# and Visual BASIC, you need to register your ActiveX by following method.

(1) Registration of ActiveX

Install the driver in Chapter 2 Windows Setup.

The DLL and ActiveX will be copied automatically.

For using the usb61api.ocx on VB, use the tool "Regsvr32.exe" that is attached to the Visual BASIC.

The "Regsvr32.exe" is 32bit console application. Therfore you must run it on command prompt.

When register "usb61api.ocx", enter on command prompt as the follows:

> regsvr32 usb61api.ocx

* On Windows 7/Vista, you have to start the command which run as administrator.

| RegSvr32 | 2 | × |
|----------|---|----|
| ٩ | DllRegisterServer in usb61api.ocx succeeded | d. |
| | OK | |

The message of registration success.

(2) Unregistration of ActiveX

When unregister it, enter on command prompt as the follows: > regsvr32 /u usb61spi.ocx

| RegSvr32 | × |
|----------|--|
| ٩ | DllUnregisterServer in usb61api.ocx succeeded. |
| | ОК |

The message of unregistration success.

4. API function reference

(3) How to reference ActiveX on VB6

Create new project.

menu.

Check-in the



X Components Select the component Controls Designers Insertable Objects with the Project PrintUI Objects 1.0 Type Library OIE rdpcomapi 1.0 Type Library * FE RelMon 1.0 Type Library System Monitor Control "usb61spi ActiveX 999 99 13 -✓ Usb61api ActiveX fRf^{*}fgf[f Control Module" in VB 6 Application Wizard ð 5 た. VB 6 Data Form Wizard the list of controls. VB 6 MSChart Wizard VCWiz 8.0 Type Library Click "OK" button. VHtmlInput 1.0 Type Library E VisModelBrowser vjshost 1.0 Type Library Browse WfcHost 1.0 Type Library 1 111 Þ Selected Items Only Usb61api ActiveX fRf"fqf[f<f,fWf...[f< Location: C:\Windows\system32\usb61api.ocx OK Cancel Apply

Then the usb61api Active X component is added.



4. API function reference

Page.4-5

Select the usb61api Active X component that was added, and then paste the project to the form. To prevent appear on the run-time, set the "Visible" in the property of the object to false.

Double-click the object, then appear the subroutine "Sub Usb61api1_OnEvnet Msg(...)" that called at when event occurs.

See the description of the "Detail of API functions".



(4)How to reference ActiveX on VB.NET / Visual C#

Create new project.



Select the [Tool] -[Choose Toolbox Items...] - [COM Components] in the menu. Check the [Usb61apiControl]. Then click the "OK" button.

Choose Toolbox Items ? X .NET Framework Components COM Components Name Path Library SDProjWiz2 Class E:\Program Files\Microsoft Visual Studio ... SelectFile Class C:\Windows\system32\CompatUI.dll CompatUI 1.0 Typ... SysColorCtrl class C:\Windows\system32\cic.dll cic 1.0 Type Library System Monitor Control C:\Windows\System32\sysmon.ocx System Monitor C... Tabular Data Control C:\Windows\system32\tdc.ocx TaskSymbol Class C:\Windows\system32\mmcndmgr.dll NodeMgr 1.0 Type... Usb(VCMacroPicker Class E:\Program Files\Microsoft Visual Studio ... VideoRenderCtl Class advd.dll VisModelBrowser.VMBrowser E:\Program Files\Microsoft Visual Studio... VisModelBrowser VJSHostCtl Class C:\Windows\Microsoft.NET\Framework\.. vishost 1.0 Type Li... Usb61api Control Browse... Language Neutral Language: OCX Version: 1.0 OK Cancel Reset

Confirm the component is registered, then paste to the form.

To prevent appear on the run-time, set the "Visible" in the property of the object to false.



4. API function reference

Double-click the object, then appear the subroutine "Sub Usb61api1_OnEvnet Msg(...)" that called at when event occurs.

See the description of the "Detail of API functions".

| WindowsApplication1 - | | | | |
|---|---|--|-----------------------|--------------------|
| _Eile _Edit _View _Refac | tor <u>P</u> roject <u>B</u> uild <u>D</u> ebug D <u>a</u> ta <u>T</u> ools ∦ 📬 🐏 🔊 → 🖓 → 💭 → 📮 ▶ Debu 🗮 ≌ 🗔 🖓 🗣 📮 📮 🥷 👰 💂 | | <u>+</u> elp - ₩ | |
| Solution Exp + 4 × | Form1.cs* WindowsApplication1* Form1.cs [[| Design1* Start Page | | + X |
| The Line was lines at the | <pre>\$ WindowsApplication1.Form1 { public partial class Form1 { public Form1() { InitializeComponent } } }</pre> | ▼ svusb61api1_ Form | OnEventMsg(object ser | nder, AxUSB61API < |
| USB6 ISB6 ISB6 Form1.cs | <pre>private void axUsb6lap:</pre> | i1_OnEventMsg(objed | t sender, AxUSE | GIAPILIDDU |
| USB6 Form.cs Sorm Form | | i1_OnEventMsg(objec | ot sender, AxUSE | 361APILibDt |
| | | il_OnEventMsg(objed | ot sender, AxUSE | |
| - USB6 Form - Form - S Ø? Cl Pr < | (-) -) -) -) -) -) -) -) -) -) | i1_OnEventMsg(objec | ot sender, AxUSE | 361APIL1bD(|
| · · · · · · · · · · · · · · · · · · · | (-) -) -) -) -) -) -) -) -) -) | i1_OnEventMsg(objec | tt sender, AxUSE | |
| Forml.ct Forml.ct Forml.ct Forml.ct Forml.ct Forml.ct Formlet Formlet | (-) -) -) -) -) -) -) -) -) -) | i1_OnEventMsg(objec | | • # × |
| S ♥ Clin Pr < | ()) s () 0 Messages | | | • # × |
| S @ Cl Pr < O Errors 10 OWarning | (-) -) -) -) -) -) -) -) -) -) | st Bench | | • # × |

(4–3) List of API functions

The list of API functions is as the below:

| Table 4-1. API Function Names and Descriptions | | | |
|--|---|--|--|
| Function name | Description | | |
| usb61_open() | Open the REX-USB61 device | | |
| usb61_close() | Close the REX-USB61 device. | | |
| usb61_power_control() | Control the supply power to the device | | |
| usb61_get_fw_version() | Get the version number of firmware | | |
| usb61_get_dll_version() | Get the version number of DLL | | |
| usb61_get_hw_info() | Retrieve the hardware information for the SPI/I2C bus operation | | |
| usb61_mode_change() | Configure SPI/I2C mode | | |
| | and Master/Slave operation | | |
| usb61_set_interval() | Configure the interval time of sending byte data to the SPI/I2C bus | | |
| usb61_gpo_write() | Output to the GPO pin on I2C mode | | |
| usb61_i2c_pullup() | Set pullup on I2C bus | | |
| | (Each pin of SDA and SCL) | | |
| usb61_i2c_bus_reset() | Reset I2C bus | | |
| usb61_i2c_set_freq0 | Configures I2C bus frequency | | |
| usb61_i2c_set_freq_ex() Configures the I2C bus frequen | | | |
| | kilohertz | | |
| usb61_i2c_read_master() | Read a stream of bytes from the I2C slave device | | |
| usb61_i2c_read_master_ex0 | Read a stream of bytes from the I2C slave | | |
| | device with sub-address | | |
| usb61_i2c_write_master() | Write a stream of bytes to the I2C slave device | | |
| usb61_i2c_read_slave() | Read a stream of bytes from the I2C | | |
| | master device | | |
| usb61_i2c_set_response_data() | Set the data for sending to master device on I2C slave mode | | |
| usb61_spi_set_freq() | Set the SPI bus frequency in kilohertz | | |
| usb61_spi_transmit_master() | Write a stream of bytes to the downstream | | |
| _ | SPI slave device | | |
| | *After write, set SS line status to High | | |
| usb61_spi_transmit_master_hold_ss0 | Write a stream of bytes to the downstream | | |
| | SPI slave device | | |
| | *After write, not set SS line status to High | | |
| | write, not set so into status to ingh | | |

(4-4) Detail of API functions

The detail of API functions is as the below. (See the VB6 sample "EEPROMRWUty" and the VB/C# sample "EEPROMRWUtyCS", for the calling method and the definition of function without the use of ActiveX on VB/C#.)

<u>General Functions</u>

| | 1 | | | |
|-------------|-----------|---|----|--|
| Definition | VC | HANDLE usb61_open(RS_STATUS *pStatus); | | |
| | VB | Function Usb61Open (pStatus As Long) As Long | | |
| | VB.NET | Function Usb61Open (ByRef pStatus As Integer) As Intege | er | |
| Description | Open the | REX-USB61 device. Start for using the REX-USB61 device. | | |
| Parameters | [OUT] pSt | tatus : RS_SUCCESS Function call succeeded. | | |
| | | Error code (refer. 4-5) Function call failed. | | |
| Return | Handle of | f an REX-USB61 device Function call succeeded. | | |
| Values | INVALID | P_HANDLE_VALUE Function call failed. | | |

| Definition | VC | RS_STATUS usb61_close(HANDLE hUsb61Device); | | |
|-------------|--|---|--|--|
| | VB | Function Usb61Close (ByVal hUsb61Device As Long) As Long | | |
| | VB.NET | Function Usb61Close (ByVal hUsb61Device As Integer) As Integer | | |
| Description | Close the REX-USB61 device. Finish using the REX-USB61 device. | | | |
| Parameters | [IN] hUsb61Device : handle of an REX-USB61 device | | | |
| Return | RS_SUCC | ESS Function call succeeded. | | |
| Values | Error code | e (refer. 4-5) Function call failed. | | |

| Definition | VC | RS_STATUS usb61_power_control(HANDLE hUsb61Device, UINT fPowerState); Function Usb61PowerControl (ByVal hUsb61Device As Long, ByVal fPowerState As Long) As Long | | |
|-------------|--|---|---|--|
| | VB | | | |
| | VB.NET | Function Usb61Power | Control (ByVal hUsb61Device As Integer, ByVal fPowerState As Integer) As Integer | |
| Description | Control th | e supply power | to the device | |
| Parameters | [IN] fPow Set the va following: RS_PWI And descr RS_PWI RS_PWI | erState Ena pow lue of bit oper RCTRL_ON, RS ibe the value of RCTRL_OFF RCTRL_ON R | dle of an REX-USB61 device bled / Disabled supply power and the voltage of er ation by using the defined symbol as the 5_OUTPUT_3_3V, RS_OUTPUT_5_0V bit-mask as the below: Disable supply power. S_OUTPUT_3_3V Enable supply power and the voltage is 3.3V. S_OUTPUT_5_0V Enable supply power and the voltage is 5.0V. | |
| Return | RS_SUCC | ESS | Function call succeeded. | |
| Values | Error code | e (refer. 4-5) | Function call failed. | |

| Definition | VC | RS_STATUS usb61_get_fw_version(HANDLE hUsb61Device, UCHAR *pFwMajorVer, UCHAR *pFwMinorVer); | | | |
|-------------|--|---|--|--|--|
| | VB | Function Usb61GetFwVersion(ByVal hUsb61Device As Long, pFWMajorVer As Byte, pFWMinorVer As Byte) As Long | | | |
| | VB.NET | Function Usb61GetFwVersion(ByVal hUsb61Device As Integer, ByRef pFWMajorVer As Byte, ByRef pFWMinorVer As Byte) As Integer | | | |
| Description | Get the version number of firmware | | | | |
| Parameters | [IN] hUsb61Device : handle of an REX-USB61 device | | | | |
| | [OUT] *pFwMajorVer : Pointer to majar version number of Firmware | | | | |
| | (Hex-decimal) | | | | |
| | [OUT] *p] | UT] *pFwMinorVer: Pointer to minor version number of Firmware | | | |
| | | (Hex-decimal) | | | |
| Return | RS_SUCC | ESS Function call succeeded. | | | |
| Values | Error code | (refer. 4-5) Function call failed. | | | |

| Definition | VC | RS_STATUS | | |
|--------------|--|--|--|--|
| | | usb61_get_dll_version(HANDLE hUsb61Device, | | |
| | | UCHAR *pDllMajorVer, UCHAR *pDllMinorVer); | | |
| | VB | Function | | |
| | VD | Usb61GetDllVersion(ByVal hUsb61Device As Long, | | |
| | | pDllMajorVer As Byte, | | |
| | | | | |
| | | pDllMinorVer As Byte) As Long | | |
| | VB.NET | Function | | |
| | | Usb61GetDllVersion(ByVal hUsb61Device As Integer, | | |
| | | ByRef pDllMajorVer As Byte, | | |
| | | ByRef pDllMinorVer As Byte) As Integer | | |
| Decomination | Catthe | | | |
| Description | Get the version number of DLL. | | | |
| Parameters | [IN] hUsb61Device : handle of an REX-USB61 device | | | |
| | [OUT] *p | [OUT] *pDllMajorVer : Pointer to majar version number of DLL | | |
| | (Hex-decimal) | | | |
| | [OUT] *pDllMinorVer : Pointer to minor version number of DLL | | | |
| | (Hex-decimal) | | | |
| | (IIex uecilial) | | | |
| Return | RS_SUC | CESS Function call succeeded. | | |
| Values | Error cod | de (refer. 4-5) Function call failed. | | |
| | | | | |

| Definition | VC | RS_STATUS | | | |
|-------------|--|--|--|--|--|
| | | usb61_get_hw_info(HANDLE hUsb61Device, | | | |
| | | PRS_HARDWARE_INFO pHardwareInfo) | | | |
| | VB | Function Usb61GetHwInfo(ByVal hUsb61Device As Long, | | | |
| | | | | | |
| | | pHardwareInfo As Byte) As Long | | | |
| | VB.NET | Function | | | |
| | | Usb61GetHwInfo(ByVal hUsb61Device As Integer, | | | |
| | | ByRef pHardwareInfo As Object) As Integer | | | |
| Description | Retrieve t | he hardware information for the SPI/I2C bus operation | | | |
| Parameters | [IN] } | uUsb61Device : handle of an REX-USB61 device | | | |
| | [OUT] pH | ardwareInfo : pointer to _RS_HARDWARE_INFO structure | | | |
| | The _RS_ | HARDWARE_INFO structure is described bellow: | | | |
| | UC US UC | UCHAR DeviceMode; // SPI/I2C mode UCHAR MasterSlaveAct; // Master/Slave operation USHORT Frequency; // frequency of interface UCHAR OutputVolt; // Output voltage for target device S_HARDWARE_INFO, *PRS_HARDWARE_INFO; ARDWARE_INFO structure is defined in usb61def.h. | | | |
| | _RS_HAR | | | | |
| | sample co | mple code for VB as the dellow: Dim pHardWareBuf() As Byte Dim HardWareInfo As RS_HARDWARE_INFO | | | |
| | | | | | |
| | ReDim pHardWareBuf(10) As Byte rsStatus = Usb61api.Usb61GetHwInfo(m_hDeviceHandle, pHardWareBuf) If rsStatus <> RS_SUCCESS Then 'error process Else | | | | |
| | End If | HardWareInfo.DeviceMode = pHardWareBuf(0) HardWareInfo.MasterSlaveAct = pHardWareBuf(1) HardWareInfo.Frequency = pHardWareBuf(3)*256 + pHardWareBuf(2) HardWareInfo.OutputVolt = pHardWareBuf(4) | | | |
| Return | RS_SUCC | ESS Function call succeeded. | | | |
| Values | Error code | e (refer. 4-5) Function call failed. | | | |

| Definition | VC | RS_STATUS usb61_mode_change(HANDLE hUsb61Device, UINT fDeviceMode, USHORT i2cSlaveAddr); | | |
|-------------|---|--|---|---|
| | VB | Function Usb61ModeChange(ByVal hUsb61Device As Long, ByVal fDeviceMode As Long, ByVal i2cSlaveAddr As Integer) As Long | | |
| | VB.NET | Function Usb61ModeChange(ByVal hUsb61Device As Integer, ByVal fDeviceMode As Integer, ByVal i2cSlaveAddr As Short) As Integer | | |
| Description | Configure | SPI/I2C n | node and Master/Sla | ave operation. |
| Parameters | [IN] hUsb | 61Device : | Handle of an REX- | USB61 device |
| | [IN] fDeviceMode : Device mode setting bits | | | |
| | Set the followir | value of bit operation by using the defined symbol as the ng: | | |
| | | RS_DEVN RS_DEVN | AODE_SPI AODE_I2C AODE_MASTER AODE_SLAVE | SPI mode I2C mode Master operation Slave operation |
| | Example: | | | |
| | | RS_DEVMODE_SPI RS_DEVMODE_MASTER (SPI master) [IN] i2cSlaveAddr : Address of I2C target device, when set I2C slave mode. | | |
| | [IN] i2cSla | | | |
| Return | RS_SUCC | ESS | F | unction call succeeded. |
| Values | Error code | e (refer. 4- | 5) I | Function call failed. |

| Definition | VC | RS_STATUS | | |
|--------------|--|---|--|--|
| 2 0111101011 | | usb61_set_interval(HANDLE hUsb61Device, | | |
| | | USHORT IntervalCnt); | | |
| | VB | Function | | |
| | | Usb61SetInterval(ByVal hUsb61Device As Long, | | |
| | | ByVal IntervalCnt As Long) As Long | | |
| | VB.NET | Function | | |
| | | Usb61SetInterval(ByVal hUsb61Device As Integer, | | |
| | | ByVal IntervalCnt As Integer) As Integer | | |
| Description | Configure the interval time of sending byte data to the SPI/I2C bus. | | | |
| I I I | (a micro-s | (a micro-second unit) | | |
| | * The act | * The actual interval is longer than the time set, because includes | | |
| | processing time. | | | |
| | (If do not call this function, then actual interval is 0 micro-second) | | | |
| | * Need to call usb61_mode_change(), before calling this function. | | | |
| Parameters | [IN] hUst | 61Device : handle of an REX-USB61 device | | |
| 1 arameters | [IN] Inter | valCnt : Interval for send data (micro-second: 0 - 65535) | | |
| Return | RS_SUCC | ESS Function call succeeded. | | |
| Values | Error cod | e (refer. 4-5) Function call failed. | | |

| Definition | VC | RS_STATUS usb61_i2c_pullup(HANDLE hUsb61Device, RS_I2C_PULLUP fI2cPullup); | |
|-------------|--|--|--|
| | VB | Function Usb61I2cPullup(ByVal hUsb61Device As Long, ByVal fI2cPullup As Integer) As Long | |
| | VB.NET | Function Usb61I2cPullup(ByVal hUsb61Device As Integer, ByVal fI2cPullup As Short) As Integer | |
| Description | Set pullup on I2C bus. (Each pin of SDA and SCL) | | |
| Parameters | [IN] hUsb61Device : handle of an REX-USB61 device | | |
| | [IN] fI2cPullup : pullup setting | | |
| | Set the value by using the defined symbol as the following: | | |
| | RS_I2C_PULLUP_DISABLE Not set pull-up the pin SCL and SDA. RS_I2C_PULLUP_ENABLE Set pull-up the pin of SCL and SDA. | | |
| | * When 1 mode) | 2C, SPI mode, always set ENABLE.(Can select only 1MHz I2C | |
| Return | RS_SUC | CESS Function call succeeded. | |
| Values | Error code (refer. 4-5) Function call failed. | | |

GPO (Only on I2C mode)

| Definition | VC | RS_STATUS usb61_gpo_write(HANDLE hUsb61Device, UINT fPortVal); | | |
|-------------|---|---|--|--|
| | VB | VB Function Usb61GpoWrite(ByVal hUsb61Device As Long, ByVal fPortVal As Long) As Long | | |
| | VB.NET | Function Usb61GpoWrite(ByVal hUsb61Device As Integer, ByVal fPortVal As Integer) As Integer | | |
| Description | Output to | o the GPO pin on I2C mode. | | |
| Parameters | [IN] h | Usb61Device : handle of an REX-USB61 device | | |
| | [IN] fF | PortVal : a bitmask specifying which outputs to GPO pin. | | |
| | GPO line location of bit mask by using the defined symbol as the following: RS_GPO_NONE Set Low(=0) to all port RS_GPO_PORT0 Set High(=1) to PORT0 RS_GPO_PORT1 Set High(=1) to PORT1 RS_GPO_PORT2 Set High(=1) to PORT2 RS_GPO_PORT3 Set High(=1) to PORT3 For setting to multiple GPO port at the same time, bit operation as the | | | |
| | following example: RS_C | : FPO_PORT0 RS_GPO_PORT1 output PORT1 and PORT2 | | |
| Return | RS_SUC | CESS Function call succeeded. | | |
| Values | Error cod | le (refer. 4-5) Function call failed. | | |

General on <u>I2C mode</u>

| Definition | VC | RS_STATUS usb | 61_i2c_bus_reset(HANDLE hUsb61Device); |
|-------------|-------------------------------------|---------------------|--|
| | VB | Function | |
| | | Usb6112cBusRe | set(ByVal hUsb61Device As Long) As Long |
| | VB.NET | Function | |
| | , 201021 | Usb61I2cBusRe | set(ByVal hUsb61Device As Integer) |
| | | | As Integer |
| Description | Reset I2 | C bus. Set the Stop | condition to the I2C bus |
| Parameters | [IN] | hUsb61Device : ha | ndle of an REX-USB61 device |
| Return | RS_SUCCESS Function call succeeded. | | Function call succeeded. |
| neturn | 100_000 | | |

| Definition | VC | RS_STATUS | |
|-------------|------------------------------|---|----------------|
| | | usb61_i2c_set_freq(HANDLE hUsb61Device, | |
| | | RS_I2C_FREQ fI2 | cFreq); |
| | VB | Function | |
| | VD | Usb61I2cSetFreq(ByVal hUsb61Device As Long, | |
| | | ByVal fI2cFreq As Integer) A | s Long |
| | VB.NET | Function | 0 |
| | VD.NE1 | Usb61I2cSetFreq(ByVal hUsb61Device As Integer, | |
| | | ByVal fI2cFreq As Short) As | Integer |
| D : .: | | | |
| Description | Configures I2C bus frequency | | |
| Parameters | [IN] | hUsb61Device : handle of an REX-USB61 device | |
| | [IN] | fI2cFreq : the frequency of I2C bus | |
| | | | · 11 · · · · · |
| | enumera | ted type of freqency by using the defined symbol as the f | ollowing |
| | RS 120 | FREQ 1M 1MHz | |
| | _ | FREQ 400K 400KHz | |
| | _ | - ·- | |
| | RS_12C | FREQ_100K 100KHz | |
| Return | RS_SUC | CESS Function call succeeded. | |
| Values | Error cod | le (refer. 4-5) Function call failed. | |

| Definition | VC RS_STATUS | | |
|-------------|--|--|--|
| | usb61_i2c_set_freq_ex(HANDLE hUsb61Device, | | |
| | USHORT Frequency, USHORT *pActualFrequency); | | |
| | VB Function | | |
| | Usb61I2cSetFreqEx(ByVal hUsb61Device As Long, | | |
| | ByVal Frequency As Long, | | |
| | pActualFrequency As Long) As Long | | |
| | VB.NET Function | | |
| | Usb61I2cSetFreqEx(ByVal hUsb61Device As Integer, | | |
| | ByVal Frequency As Integer, | | |
| | ByRef pActualFrequency As Integer) As Integer | | |
| Description | Configures the I2C bus frequency in kilohertz. | | |
| | Can be set from 47 to 100KHz. | | |
| | The actual frequency value to be set return to the pActualFrequency. | | |
| Parameters | [IN] hUsb61Device : Handle of an REX-USB61 device | | |
| | [IN] Frequency : The frequency to request on I2C bus | | |
| | [OUT] pActualFrequency : The actual frequency value to be set | | |
| Return | RS_SUCCESS Function call succeeded. | | |
| Values | Error code (refer. 4-5) Function call failed. | | |

I2C bus operation on I2C bus master mode

| Definition | VC | RS_STATUS | | |
|-------------|---|---|--|--|
| | | usb61_i2c_read_master(HANDLE hUsb61Device, | | |
| | | USHORT SlaveAddress, UINT fl2cOption, | | |
| | | USHORT ReadBytes, UCHAR *pReadBuf); | | |
| | VB | Function | | |
| | | Usb61I2cReadMaster(ByVal hUsb61Device As Long, | | |
| | | ByVal SlaveAddress As Integer, ByVal fI2cOption As Long, | | |
| | | ByVal ReadBytes As Integer, pReadBuf As Byte) As Long | | |
| | VB.NET | Function | | |
| | | Usb61I2cReadMaster(ByVal hUsb61Device As Integer, | | |
| | | ByVal slaveAddress As Short, ByVal fI2cOption As Integer, | | |
| | | ByVal readBytes As Short, ByRef pReadBuf As Object) | | |
| | | As Integer | | |
| Description | Read a stream of bytes from the I2C slave device. | | | |
| Parameters | [IN] hUsb61Device : handle of an REX-USB61 device | | | |
| | [IN] SlaveAddress : the slave from which to read. See the below | | | |
| | [IN] fI2cOption : special operation as described in "Table 4-2" and below | | | |
| | [IN] ReadBytes : the number of bytes to read | | | |
| | [OUT] pI | ReadBuf : pointer to data to read. | | |
| Return | RS_SUC | CESS Function call succeeded. | | |
| Values | Error cod | le (refer. 4-5) Function call failed. | | |

*Slave address:

Specify the slave address in 7bits or 10bits, not includes R/W bit.

| Definition | VC | RS_STATUS | |
|-------------|---|---|--|
| Deminion | | usb61_i2c_read_master_ex(HANDLE_hUsb61Device, | |
| | | USHORT SlaveAddress, USHORT SubAddress, | |
| | | UINT fI2cOption, USHORT ReadBytes, UCHAR *pReadBuf); | |
| | VB | Function | |
| | 1D | Usb61I2cReadMasterEx(ByVal hUsb61Device As Long, | |
| | | ByVal SlaveAddress As Integer, | |
| | | ByVal SubAddress As Integer, ByVal fI2cOption As Long, | |
| | | ByVal ReadBytes As Integer, | |
| | | pReadBuf As Byte) As Long | |
| | VB.NET | Function | |
| | | Usb61I2cReadMasterEx(ByVal hUsb61Device As Integer, | |
| | | ByVal slaveAddress As Short, ByVal subAddress As Short, ByVal fI2cOption As Integer, ByVal readBytes As Short, | |
| | | ByRef pReadBuf As Object) As Integer | |
| Description | Read a stream of bytes from the I2C slave device with sub-address. | | |
| | It is different from the "usb61_i2c_read_master" function that write data | | |
| | before fo | or reading with the specifying the calling position (specifying | |
| | sub-addr | ress) on inside function. | |
| Parameters | [IN] h | Usb61Device : handle of an REX-USB61 device | |
| | [IN] SI | aveAddress : the slave from which to read. | |
| | | See *Slave address in Page4-16. | |
| | [IN] St | ubAddress :Sub address (supports 2 bytes-address) | |
| | [IN] fI | 2cOption is special operation as described in "Table 4-2" and below | |
| | [IN] R | eadBytes : the number of bytes to read | |
| | [OUT] pl | ReadBuf : pointer to data to read. | |
| Return | RS_SUC | CESS Function call succeeded. | |
| Values | Error coo | de (refer. 4-5) Function call failed. | |

| Definition | VC | RS_STATUS usb61_i2c_write_master(HANDLE hUsb61Device, | | |
|-------------|---|--|--|--|
| | | USHORT SlaveAddress,UINT fl2cOption, | | |
| | | USHORT WriteBytes, UCHAR *pWriteBuf); | | |
| | VB | Function | | |
| | ۷D | Usb61I2cWriteMaster(ByVal hUsb61Device As Long, | | |
| | | ByVal SlaveAddress As Integer, | | |
| | | ByVal fI2cOption As Long, | | |
| | | ByVal WriteBytes As Integer, | | |
| | | ByVal pWriteBuf As Byte) As Long | | |
| | VB.NET | Function | | |
| | | Usb61I2cWriteMaster(ByVal hUsb61Device As Integer, | | |
| | | ByVal slaveAddress As Short, ByVal fileOntion As Integer | | |
| | | ByVal fI2cOption As Integer, ByVal writeBytes As Short, | | |
| | | ByVal pWriteBuf As Object) As Integer | | |
| Description | Write a | e a stream of bytes to the I2C slave device | | |
| Parameters | [IN] hUsb61Device : handle of an REX-USB61 device | | | |
| | [IN] SlaveAddress : the slave from which to read. | | | |
| | | See *Slave address in Page4-16. | | |
| | [IN] fI | 2cOption : special operation as described in "Table 4-2" and below | | |
| | [IN] W | riteBytes : the number of bytes to write | | |
| | [IN] p | WriteBuf : pointer to data to write | | |
| Return | RS_SU | CCESS Function call succeeded. | | |
| Values | Error co | ode (refer. 4-5) Function call failed. | | |

Table 4-2. Special operation on I2C bus

| Literal Name | Value | Description |
|------------------------|-------|--|
| RS_I2C_FLAG_NONE | 0x00 | No flags. |
| RS_I2C_FLAG_10BIT_ADDR | 0x01 | For 10-bits address device |
| RS_I2C_FLAG_STOP | 0x02 | Set before issue the stop condition |
| RS_I2C_FLAG_1BYTE_SA | 0x04 | Send 1 byte sub-address before reading data |
| RS_I2C_FLAG_2BYTE_SA | 0x0C | Send 2 bytes sub-address before reading data |

I2C bus operation on I2C bus slave mode

| VC | RS_STATUS | | |
|---|---|--|----------|
| ve | usb61_i2c_read_slave(HANDLE hUsb61Device, | | |
| | RS_NOTIFY_TYPE nType, void (CALLBACK EXPORT* lpfnReadEvent) | | |
| | (USHORT ReadBytes, UCHAR *pReadBuf), | | |
| | HWND hWnd); | | |
| VB Function | | | |
| | Usb61I2cReadSlave(ByVal hUsb61Device As Long, ByVal nType As Integer) As Long | | |
| VB.NET | Function | | |
| | Usb61I2cReadSlave(ByVal hUsb61Device As Integer, ByVal nType As Short) As Integer | | |
| Read a s | tream of bytes from the I2C master device. | | |
| In backg | round, waiting until it receives the data from master device, after call this | | |
| function | | | |
| The com | pletion of receiving the data, it is notified to the application via callback | | |
| function | | | |
| Before ca | alling this function, have to call the "usb61_I2c_set_response_data()" | | |
| function | tion to set the data for sending to master device in advance. | | |
| On Visua | visual Basic, by using ActiveX control, as user-defined-message | | |
| "WM_US | USB61_MSG" is notified. | | |
| [IN] hl | Jsb61Device : handle of an REX-USB61 device | | |
| [IN] n7 | Type : notification method | | |
| enumera | ated type of notification method by using the defined symbol as the following: | | |
| | _NOTIFY_CALLBACK notified by callback function (only VC) _NOTIFY_USER_MSG notified by user message | | |
| [IN] lp | fnReadEvent : callback function which notify to application | | |
| 'lpfnReadEvent' callback function supplied by the upper application is set as the argument. The name of 'lpfnReadEvent' callback function does not have to be 'ReadIsComple but it must be defined as follows: | | | |
| | | | void CAI |
| [IN] hW | Vnd : window handle which notify user message | | |
| | if not notify user message, set NULL | | |
| RS_SUC | CESS Function call succeeded. | | |
| Error co | de (refer. 4-5) Function call failed. | | |
| | VB.NET Read a s In backg function The com function Before ca function On Visua "WM_US [IN] hU [IN] hU [IN] hU (IN] hU lpfnRea argumen The nam but it mu void CAI [IN] hV RS_SUC | | |

| Definition | VC | No use on VC |
|------------------|-------------------------------------|---|
| | VB | Function Usb61GetData(ByVal wParam As Long, ByVal lParam As Long, pBuf As Byte) As Long |
| | VB.NET | Function Usb61GetData(ByVal wParam As Integer, ByVal lParam As Integer, ByRef pBuf As Object) As Integer |
| Description | Get th "WM_US | e data by using the "Usb61I2cReadSlave" function, when the SB61_MSG" message is posted |
| Parameters | ' Statu Dim ra Dim p ReDin | ram : address of the data to read pBuf : pointer to the data to read e on VB: ub Usb61api_OnEventMsg(ByVal wParam As Long, ByVal lParam As Long) |
| Return Values | RS_SUC | CESS Function call always succeeded. |

| Definition | VC | RS_STATUS usb61_i2c_set_response_data(HANDLE hUsb61Device, USHORT ResponseBytes, UCHAR *pResponseBuf); | | |
|-------------|---|---|--|--|
| | VB | Function Usb61I2cSetResponseData(ByVal hUsb61Device As Long, ByVal ResponseBytes As Integer, ByVal pResponseBuf As Byte) As Long | | |
| | VB.NET | Function Usb61I2cSetResponseData(ByVal hUsb61Device As Integer, ByVal responseBytes As Short, ByVal pResponseBuf As Object) As Integer | | |
| Description | Set the data for sending to master device on I2C slave mode | | | |
| | When ree | When receive data from master device, send the data pre-set for master | | |
| Parameters | [IN] hUsb61Device : handle of an REX-USB61 device | | | |
| | [IN] | ResponseBytes it is number of bytes for sending to master device | | |
| | [IN] | pResponseBuf : pointer to the data for sending to master device | | |
| Return | RS_SUC | CESS Function call succeeded. | | |
| Values | Error coo | de (refer. 4-5) Function call failed. | | |

SPI bus operation on SPI bus master mode

| Definition | VC | RS_STATUS usb61_spi_set_freq(HANDLE hUsb61Device, UINT fDataMode, USHORT Frequency, USHORT *pActualFrequency); | |
|-------------|--|--|--|
| | VB | Function Usb61SpiSetFreq(ByVal hUsb61Device As Long, ByVal fDataMode As Long, ByVal Frequency As Long, pActualFrequency As Long) As Long | |
| | VB.NET | Function Usb61SpiSetFreq(ByVal hUsb61Device As Integer, ByVal fDataMode As Integer, ByVal frequency As Integer, ByRef pActualFrequency As Integer) As Integer | |
| Description | Set the S | SPI bus frequency in kilohertz. | |
| | Can be s | set from 1 to 12000KHz. | |
| | The app | roximate value of frequency that can be set is calculated from the | |
| | 'Frequer | ncy' parameter. | |
| | The actu | al frequency value to be set, returns to the 'pActualFrequency' parameter. | |
| | Note: Th | he approximate value of frequency will be calculated as the follows: | |
| | The X is paramet | the integer part of the value that 6024 divided by the 'Frequency' ser. | |
| | Y = inte | is greater than or equal to 1020, [$X \ge 1020$] eger of (X / 16) | |
| | *pAcuti | ualFrequency = integer of 6024 / (Y * 16) | |
| | The Y is the integer part of the value that X divided by 16. The integer part of the value which 6024 divided by 16 multiple of Y, will set to the 'pAcutualFrequency' parameter. | | |
| | If the X is greater than or equal to 256 and smaller than 1020 [256 =< X < 102 Y = integer of (X / 4) *pAcutualFrequency = integer of 6024 / (Y * 4) | | |
| | The Y is the integer part of the value that X divided by 4. The integer part of the value which 6024 divided by 4 multiple of Y, will set to t 'pAcutualFrequency' parameter. | | |
| | If the X is smaller than 256 [X < 256] | | |
| | *pAcutualFrequency = integer of 6024 / X | | |
| | The integer part of the value which 6024 divided by X, will set to the 'pAcutualFrequency' parameter. | | |
| | When the 'Frequency' parameter is 1, 750, 300, 12000(KHz), these frequency has special setting value. | | |
| | Therefore, the same value as the Frequency is returned to the 'pActualFrequency' parameter. When the 'Frequency' parameter is greater than or equal to 3013(KHz), 12000(KHz) will set to the 'pAcutualFrequency' parameter. | | |
| | | | |
| | | | |
| | | | |

| Parameters | [IN] hUsb61Device : handle of an REX-USB61 device | | | |
|------------|---|--|--|--|
| | [IN] fDataMode : a bit mask specifying which operation mode for sending on SPI bus. | | | |
| | The clock polarity and the clock edge determine the value of bit mask. | | | |
| | The bit mask by using the defined symbol as the following:RS_SPI_PHASE_SETUP_SAMPLEsampling on rising edgeRS_SPI_PHASE_SAMPLE_SETUPsampling on falling edgeRS_SPI_POLARITY_POSITIVEclock polarity is positiveRS_SPI_POLARITY_NEGATIVEclock polarity is negativeRS_SPI_MSB_FIRSTMBS firstRS_SPI_LSB_FIRSTLSB firstexample: | | | |
| | RS_SPI_PHASE_SETUP_SAMPLE RS_SPI_POLARITY_POSITIVE RS_SPI_MSB_FIRST | | | |
| | [IN] Frequency : the frequency to request on SPI bus | | | |
| | [OUT] pActualFrequency: the actual frequency value to be set | | | |
| Return | RS_SUCCESS Function call succeeded. | | | |
| Values | Error code (refer. 4-5) Function call failed. | | | |

| Definition | VC | RS_STATUS usb61_spi_transmit_master(HANDLE hUsb61Device, RS_SPI_SS fSlaveSelect, USHORT TransmitSize, UCHAR *pSendBuf, UCHAR *pRecvBuf); | | |
|-------------|---|--|--|--|
| | VB | Function Usb61SpiTransmitMaster(ByVal hUsb61Device As Long, ByVal fSlaveSelect As Integer, ByVal TransmitSize As Integer, ByVal pSendBuf As Byte, pRecvBuf As Byte) As Long | | |
| | VB.NET | Function Usb61SpiTransmitMaster(ByVal hUsb61Device As Integer, ByVal fSlaveSelect As Short, ByVal transmitSize As Short, ByVal pSendBuf As Object, ByRef pRecvBuf As Object) As Integer | | |
| Description | Write a | stream of bytes to the downstream SPI slave device and read back dummy | | |
| | data. | | | |
| | After wr | After write, set SS line status to High. | | |
| Parameters | [IN] | I] hUsb61Device : handle of an REX-USB61 device | | |
| | [IN] | fSlaveSelect : Pin number for slave select | | |
| | enumerated values specifying pin number the bellow: | | | |
| | RS_SPI_SS0 Slave select pin number 0 | | | |
| | | RS_SPI_SS1 Slave select pin number 1 RS_SPI_SS2 Slave select pin number 2 | | |
| | RS | _SPI_SS3 Slave select pin number 3 | | |
| | [IN] Tra | nsmitSize : the number of bytes to write | | |
| | [IN] pSe | ndBuf : pointer to write data | | |
| | [OUT] p | RecvBuf : pointer to read back data | | |
| Return | RS_SUC | CESS Function call succeeded. | | |
| Values | Error co | de (refer. 4-5) Function call failed. | | |

| Definition | VC | RS_STATUS usb61_spi_transmit_master_hold_ss(HANDLE hUsb61Device, RS_SPI_SS fSlaveSelect, USHORT TransmitSize, UCHAR *pSendBuf, UCHAR *pRecvBuf); | | |
|-------------|--|--|--|--|
| | VB | Function Usb61SpiTransmitMasterHoldSS(ByVal hUsb61Device As Long, ByVal fSlaveSelect As Integer, ByVal TransmitSize As Integer, ByVal pSendBuf As Byte, pRecvBuf As Byte) As Long | | |
| | VB.NET | Function Usb61SpiTransmitMasterHoldSS(ByVal hUsb61Device As Integer, ByVal fSlaveSelect As Short, ByVal transmitSize As Short, ByVal pSendBuf As Object, ByRef pRecvBuf As Object) As Integer | | |
| Description | | stream of bytes to the downstream SPI slave device and read back dummy | | |
| | data. | its not set CS line status to High | | |
| | | After write, not set SS line status to High. | | |
| Description | | For setting SS line to High, call usb61_gpo_write function. | | |
| Parameters | | [IN] hUsb61Device : handle of an REX-USB61 device | | |
| | [IN] | fSlaveSelect : Pin number for slave select | | |
| | enumerated values specifying pin number the bellow: | | | |
| | RS_SPI_SS0 Slave select pin number 0 RS_SPI_SS1 Slave select pin number 1 | | | |
| | RS_SPI_SS2 Slave select pin number 2 | | | |
| | RS_SPI_SS3 Slave select pin number 3 | | | |
| | [IN] TransmitSize : the number of bytes to write | | | |
| | [IN] pSendBuf : pointer to write data | | | |
| | [OUT] pRecvBuf : pointer to read back data | | | |
| Return | _ | UCCESS Function call succeeded. | | |
| Values | Error co | de (refer. 4-5) Function call failed. | | |

(4-5) Error Codes

| Literal Name | | |
|-----------------------------|-------|--|
| Literal Name | Value | Description Function call succeeded |
| RS_SUCCESS | 0 | ok |
| RS_OK | | device is found |
| RS_DEVICE_FOUND | 0 | |
| RS_DEVICE_CONNECT | 0 | device is connected |
| RS_UNABLE_TO_LOAD_LIBRARY | -1 | unable to load library |
| RS_UNABLE_TO_LOAD_DRIVER | -2 | unable to load REX-USB61 |
| | 0 | driver |
| RS_UNABLE_TO_LOAD_FUNCTION | -3 | unable to call function |
| RS_INCOMPATIBLE_LIBRARY | -4 | incompatible library version |
| RS_INCOMPATIBLE_DEVICE | -5 | incompatible device |
| RS_COMMUNICATION_ERROR | -6 | communication error |
| RS_UNABLE_TO_OPEN | -7 | unable to open device |
| RS_UNABLE_TO_CLOSE | -8 | unable to close device |
| RS_INVALID_HANDLE | -9 | invalid device handle |
| RS_CONFIG_ERROR | -10 | configuration error |
| RS_TIMEOUT | -11 | time out |
| RS_OUT_OF_RANGE | -12 | out of range |
| RS_DEVICE_NOT_FOUND | -20 | device not found |
| RS_DEVICE_NOT_CONNECT | -21 | device not connected |
| RS_DEVICE_OPEN_EXIST | -22 | device already opened |
| RS_I2C_NOT_AVAILABLE | -100 | I2C bus not available |
| RS_I2C_NOT_ENABLED | -101 | I2C not enabled |
| RS_I2C_READ_ERROR | -102 | I2C read error |
| RS_I2C_WRITE_ERROR | -103 | I2C wrtie error |
| RS_I2C_BAD_CONFIG | -104 | I2C bad configuration |
| RS_I2C_TIMEOUT | -105 | I2C bus timeout |
| RS_I2C_DROPPED_EXCESS_BYTES | -106 | I2C dropped excess bytes |
| RS_I2C_BUS_ALREADY_FREE | -107 | I2C bus already free |
| RS_I2C_WRITE_COLLISION | -108 | I2C write collision |
| RS_I2C_READ_OVERFLOW | -109 | I2C read overflow |
| RS_I2C_NACK_DETECT | -110 | I2C no ack detected |
| RS_I2C_OUTRANGE | -111 | I2C out of range |
| RS_SPI_NOT_AVAILBLE | -200 | SPI bus not available |
| RS_SPI_NOT_ENABLED | -201 | SPI bus not enabled |
| RS_SPI_WRITE_ERROR | -202 | SPI write error |
| RS_SPI_READ_ERROR | -203 | SPI read error |
| RS_SPI_BAD_ CONFIG | -204 | SPI bad configuration |
| RS_SPI_TIMEOUT | -205 | SPI bus timeout |
| RS_SPI_DROPPED_EXCESS_BYTES | -206 | SPI dropped excess bytes |
| RS_SPI_WRITE_OVERFLOW | -207 | SPI write overflow |
| RS_SPI_OUTRANGE | -208 | SPI out of range |
| RS_GPO_NOT_AVAILABLE | -300 | GPO port not available |
| RS_FAILURE | -400 | general error |

* A positive value of except above is error code of Win32.

(4-6) Sample Applications

REX-USB61 includes a sample application of reference of the application development.

The "EEPROM R / W Utility" sample application that can read and write for the EEPROM (ATMEL AT24C02B, AT25080A) with I2C interface or SPI. It includes in the "EEPROMRWUty" folder.

The "I2cSlaveSample" sample application can work as I2C slave for the REX-USB61.

It includes in the "I2CSlaveSample" folder.

[Description of EEPROM R/W Utility]

| Select SPI/I2C Bus | select mode SPI or I2C |
|----------------------------|---|
| Direction | Read or Write |
| Operation Frequency | Enter the frequency for setting |
| Actual Frequency | Display the frequency that calculated from the above value. |
| Interval C | onfigure Interval of 1 byte for sending |
| I2C Target address | Enter I2C target address |
| EEPROM Address | Start position for reading or writing |
| Transfer length | Transfer length for reading or writing |
| Write Data | Transfer data for writing |
| Read Data | Display received data |
| Execute | Start transfer by the above settings |

| Write Data | ⊤Transfer Settings |
|------------|---|
| | Select SPI/I2C Bus SPI C I2C Direction (Read/Write) Read C Write Operation Frequency Actual Frequency 100 KHz KHz Interval (time interval of each byte) usec I2C ??????????? H |
| Read Data | EEPROM Address (Read/Write position) 00 H Transfer length (Read/Write bytes) 256 Execute Execute |
| | Target Device SPI EEPROM : I2C EEPROM : AT25080A AT24C02B Compatible Compatible device: device: AT25160A AT24C01A AT25320A AT24C02 AT25640A AT24C02 AT24C08A AT24C16A |

[View of EEPROM R/W Utility]

See the source code for programming.

And refer to the specification of the EEPROM which the EEPROM maker provides.

[Description of I2CSlaveSample]

| I2C Slave address | Set I2C slave address | |
|-----------------------------------|---------------------------------------|--|
| Operation Frequency of I2C Master | | |
| | Select frequency of I2C master device | |
| Response data | Set response data for I2C master | |

- Execute Start for I2C slave Receive data
 - Display receive data from I2C master
- Clear Erase displayed receive data

| laster | | | |
|-------------------|-----------------------------|---|--|
| le (Operation fre | quency | <=400Kł | Hz) |
| peration frequer | ncy > 40 | OKHz) | |
| r | | | |
| | | | * |
| | | | |
| | | | |
| | 1.5 | | Ŧ |
| Clear | | Start | - |
| Clear | | Start | + |
| | | Start | - |
| Clear | | Start | + |
| | | Start | - |
| | | Start | + |
| | | Start | * |
| | | Start | * |
| | | Start | |
| 0 | Master de (Operation fre | Master de (Operation frequency Operation frequency > 40 | Master de (Operation frequency <=400Kl Operation frequency > 400KHz) |

[View of I2CSlaveSample]

See the source code for programming.

(4-7) How to develop application using this API functions

This section describes how to create a control application using the REX-USB61 API functions.

It is an example in C++. If you want to know in other programming language or detail, refer to the source code of the sample programs.

Example - EEPROM R/W Utility [I2C] :

Output a byte 'FF' hex-decimal data from SDA line. (C++) It does not include error handling.

| HANDLE hDeviceHandle; | // Device handle |
|-------------------------------------|--|
| BYTE DeviceAddr; | // Device address |
| WORD DataLen; | // Transfer length |
| USHORT i2cFreq; | // Operating frequency |
| WORD ActualFreq; | // Actual frequency |
| USHORT IntervalCnt; | // Interval |
| BYTE Data; | // Data for writing |
| | |
| // Get device handle | |
| hDeviceHandle = usb61_open(&r | sStatus); |
| // Supply 5.0V power to target de | vice. |
| | arget device without from REX-USB61, |
| // Set RS_PWRCTRL_OFF | |
| | ndle, RS_PWRCTRL_ON RS_OUTPUT_5_0V); |
| | |
| // Set I2C master mode | |
| usb61_mode_change(hDeviceHar | ndle, RS_DEVMODE_I2C RS_DEVMODE_MASTER, NULL); |
| | |
| // Set Interval (After I2C mode ch | nanged) |
| usb61_set_interval(hDeviceHand | lle, IntervalCnt); |
| | |
| // Set I2C bus pull-up | |
| usb61_i2c_pullup(hDeviceHandl | e, RS_I2C_PULLUP_ENABLE); |
| // Set frequency | |
| usb61_i2c_set_freq_ex(hDeviceH | landle i2cFreq &ActualFreq); |
| accor | |
| // Output a byte 'FF' hex-decimal | data from SDA line |
| // Set target device address to the | e "DeviceAddr". Do not include R/W bit, |
| // Set x00 to Device address. | |
| DeviceAddr = 0x00; // Device a | ddress = 0x00 |
| Data $= 0xFF;$ // A byte 'F | FF hex-decimal data |
| DataLen = 1; // length of | f data = 1 |
| usb61_i2c_write_master(hDevice | eHandle, DeviceAddr, RS_I2C_FLAG_STOP, DataLen, &Data); |
| | |
| // Finish using the REX-USB61 d | levice |
| usb61_close(hDeviceHandle); | |
| hDeviceHandle = NULL; | |

Example - EEPROM R/W Utility [SPI] :

Output a byte 'FF' hex-decimal data from SDO line. (C++) $\,$

It does not include error handling.

| HANDLE hDeviceHandle; | // Device handle |
|--|--|
| WORD DataLen; | // Transfer length |
| USHORTspiFreq; | // Operating frequency |
| WORD ActualFreq; | // Actual frequency |
| BYTE pWriteBuf; | // Store data for writing |
| BYTE pReadBuf; | // Store data for reading |
| UINT uiFlag; | // bit combination of clock leading or trailing, polarity |
| // Get device handle | |
| hDeviceHandle = usb61_open(| (&rsStatus); |
| // Supply 5.0V power to target | device. |
| | n target device without from REX-USB61, |
| // Set RS_PWRCTRL_OFF | |
| usb61_power_control(hDevice) | Handle, RS_PWRCTRL_ON RS_OUTPUT_5_0V); |
| // Set SPI master mode | |
| usb61_mode_change(hDeviceF | Handle, RS_DEVMODE_SPI RS_DEVMODE_MASTER, NULL); |
| // Set Interval (After SPI mode | e changed) |
| usb61_set_interval(hDeviceHa | andle, IntervalCnt); |
| // Set frequency | |
| | SAMPLE_SETUP RS_SPI_POLARITY_POSITIVE |
| RS_SPI_MSB_F | |
| usb61_spi_set_treq(hDeviceH | andle, uiFlag, spiFreq, &ActualFreq); |
| // Send the Write Enable comm | nand(0x06) |
| pWriteBuf[0] = 0x06; // WRE | N command |
| | fer length |
| usb61_spi_transmit_master(h | DeviceHandle, RS_SPI_SS0, DataLen, pWriteBuf, pReadBuf); |
| // Output a byte 'FF' hex-decim | nal data from SDO line |
| | + EEPROM address(0x0000) + data for writing(0xff)] 4bytes |
| pWriteBuf[0] = 0x02 // Write | |
| | ROM address(upper byte) |
| • | ROM address(lower byte) |
| 1 | for writing |
| | fer length |
| usb61_spi_transmit_master(h // In reading, after the fourth of | nDeviceHandle, RS_SPI_SS0, DataLen, pWriteBuf, pReadBuf); |
| " in reading, after the fourth (| n oyou are uava. |
| // Finish using the REX-USB6 | 1 device |
| usb61_close(hDeviceHandle); | |
| hDeviceHandle = NULL; | |

<u>Example – I2cSlaveSample [I2C] :</u>

This program operates as a slave device for REX-USB61.

When reading from the I2C master device, send the response data which prepared in advance to the I2C master device.

The transmission of data from the I2C master device is notified by the event of receipt or the callback function.

And it is displayed on the application program. (C++)

It does not include error handling.

| TANDLE I | nDeviceHandle; | // Device handle |
|----------------------------|----------------------------------|--|
| char | csSlaveAddr[16]; | // Character string for I2C slave address |
| ULONG | ulSlaveAddr; | // I2C slave address |
| char | *stopstring; | |
| USHORT | usFreq; | // Operating frequency |
| USHORT | usActualFreq; | // Actual frequency |
| BYTE Resp | oonseBuf[255]; | // response data |
| WORD Res | ponseBytes; | // length of response data |
| // Get devic | e handle | |
| hDeviceHa | ndle = usb61_open(& | krsStatus); |
| // Supplies | a power supply from | 12C master |
| // REX-USE | 361 does not supply | power |
| usb61_powe | er_control(hDeviceH | Iandle, RS_PWRCTRL_OFF); |
| // I2C salve | address | |
| GetDlgIten | nText(hwnd, IDE_SI | LAVE_ADDRESS, csSlaveAddr, sizeof(csSlaveAddr)); |
| ulSlaveAdd | ir = strtoul(csSlaveA | addr, &stopstring, 16); |
| // Set I2C sl | lave mode | |
| usb61_mod | e_change(hDeviceH | andle, RS_DEVMODE_I2C RS_DEVMODE_SLAVE, |
| | (USHOR | T)ulSlaveAddr); |
| // Enable pu | all-up | |
| usb61_i2c_p | oullup(hDeviceHand | lle, (RS_I2C_PULLUP)fPullup); |
| // Set freque | encv | |
| - | - | Handle, usFreq, &usActualFreq); |
| // Set resp | onse data | |
| // usb61_i2c | e_set_response_data | (hDeviceHandle, ResponseBytes, ResponseBuf); |
| | c 1 | as I2C slave |
| // Waiting | for receving data | |
| | for receving data vice handle | |
| | vice handle | |
| // Save dev g_hwnd = hw | vice handle wnd; | notified the event of receipt |

(When noitfied by event of receipt, receive user-message and display the transmission of data from the I2C master device.)



(C) RATOC Systems, Inc. All rights reserved.