User's Guide

i2cStick™

USB to I²C Bus Host Adapter

with iPort Utility Pack Software

MCC
Micro Computer Control
Small Area Network Specialists

www.mcc-us.com

Introduction

The MCC i2cStick™ USB to I²C Host Adapter (#MIIC-207) allows any Windows 2000, XP, Vista(x86/x64), or 7(x86/x64) PC, with a free USB port or self-powered USB hub, to become an I²C Master or Slave device, transmitting or receiving I²C messages between the PC and one or more I²C devices across an I²C Bus.

What's New!

- Compact shirt-pocket size for on-the-go developers, application engineers, field service engineers, and technicians.
- Fixed-speed communications (115.2K baud).
- Master Transmit Packet-Error Checking (PEC) detection.
- Low-cost I²C Bus connector and cabling.

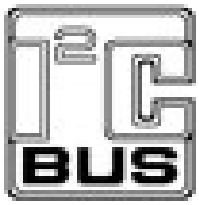
This user's guide describes the installation and operation of the i2cStick host adapter, Virtual Communication Port (VCP) driver, the iPort Utility Pack Software for Windows, and includes the Programmer's Reference for creating custom applications.

Are you new to I²C? Want to know more? We suggest you review "What is I²C?" at www.mcc-us.com/I2CBusTechnicalOverview.pdf.

This MCC product uses NXP (Philips) components and is licensed to use the I²C Bus.

"Purchase of Philips I²C components conveys a license under the Philips' I²C patent to use the components of the I²C system, provided the system conforms to the I²C specifications defined by Philips."

I²C is a trademark of NXP (Philips) Corporation.



DISCLAIMER: Micro Computer Control Corporation makes no representations or warranties with respect to the contents hereof and specifically disclaims any implied warranties of merchantability or fitness for any particular purpose. Further, Micro Computer Control Corporation reserves the right to revise the product described in this publication and to make changes from time to time in the content hereof without the obligation to notify any person of such revisions or changes.

WARNING - Life Support Applications: MCC products are not designed for use in life support appliances, devices, or systems where the malfunction of the product can reasonably be expected to result in a personal injury.

WARNING - Radio Frequency Emissions: This equipment has been tested and found to comply with the limits for a Class A digital device, pursuant to part 15 of the FCC rules. These limits are designed to provide reasonable protection against interference when the equipment is operated in a commercial environment. This equipment generates, uses, and can radiate radio frequency energy and, if not installed and used in accordance with the instruction manual, may cause interference to radio communications. Operation of this equipment in a residential area is likely to cause interference, in which case the user will be required to correct the interference at his own expense.

WARNING - Electrostatic Discharge (ESD) Precautions: Any damage caused by Electrostatic Discharge (ESD) through inadequate earth grounding is NOT covered under the warranty of this product. See the “Electrostatic (ESD) Precautions” section of this guide for more information.

Printed in the United States of America

Table of Contents

Part 1

i2cStick™

USB to I²C Bus Host Adapter

User's Guide

Model: MIIC-207

1 Overview

The MCC i2cStick USB to I^2C Bus Host Adapter (#MIIC-207) allows any Windows 2000, XP, Vista, or 7 PC with a free USB port or self-powered USB hub, to become an I^2C Master or Slave device, transmitting or receiving I^2C messages between the PC and one or more I^2C devices across an I^2C Bus.

i2cStick Product Features:

- Add an I^2C port to ANY Windows 2000, XP, Vista(x86/x64), 7(x86/x64) PC.
- Compatible with USB 2.0 Specifications.
- Works with USB Full or High-Speed systems.
- Supports I^2C Bus Master and Slave, Transmit and Receive operations.
- Four I^2C bit rates up to 400 KHz.
- Compatible with 3.3V to 5V I^2C Bus pull-up voltages.
- Switch-enabled internal I^2C Bus pull-up resistors.
- Based on a High Performance I^2C Bus Co-Processor.
- Optimized for High Bus Throughput, and Low Inter-byte overhead.
- Free I^2C Message Center, Message Manager, and iBurner EEPROM Programming utility software.
- Easy to use Virtual Communications Port (VCP) driver.
- Build custom I^2C Bus applications using our simple ASCII Text Commands, or our free MS.NET Class Library.
- Compatible with iPort/AI, iPort/AFM, iPort/USB applications. No software changes required.

The i2cStick package includes the following items:

1.1 i2cStick USB to I^2C Bus Host Adapter

The i2cStick adapter is a bus-powered USB device that plugs into a host computer's USB port or self-powered USB hub and generates I^2C Bus signals.

1.2 i2cStick Virtual Communications Port (VCP)

The i2cStick Virtual Communications Port (VCP) driver creates a virtual serial port within the i2cStick device. To a Windows software application, the VCP looks just like a legacy RS-232 serial communications port (COMn), allowing standard serial communication programming methods to work unchanged. No special USB programming is required.

1.3 iPort Utility Pack Software

This free software package includes the iPort Message Center and Message Manager applications to help you easily send and receive I²C Bus messages.

1.4 i2cStick Programmer's Reference

This section of the i2cStick User's Guide provides a programmer's guide to creating custom I²C Bus applications. Find additional sample programs and complete projects on our web site's Sample Program page.

1.5 Packing Slip

This package includes the following items:

- i2cStick USB to I²C Bus Host Adapter (#MIIC-207).
- I²C Bus Mini Clip-Lead cable (#I2CMCL).
- i2cStick User's Guide (this document).
- iPort Utility Pack for Windows CD.

1.6 System Requirements

a. A host computer with one free USB port or self-powered USB hub.
b. Windows XP (x86), Vista (x86/x64), 7 (x86/x64).

2 Interconnects

The i2cStick includes two interconnections:

USB Type A

I²C Bus 1x5 2.54 mm (.100")

2.1 USB Connector

The USB Type A connector provides connection from the I²C adapter to a USB port

on the host computer or self-powered USB hub. The i2cStick operates as a low-power (<100 mA) bus-powered USB device.

2.1.1 Virtual Communications Port (VCP)

The i2cStick provides a Virtual Communications Port (VCP) interface via a host computer driver. Install the iPort Utility Pack (available on CD or online) and plug the i2cStick into a USB port. Windows will automatically load the driver and instantly create a new com port (COMn) on your computer. The VCP hides the complexities of a USB physical interface. Application programs running on the host computer communicate with the i2cStick via the standard Windows Communications Application Program Interface (API).

The i2cStick includes the following communication signals VCP interface:

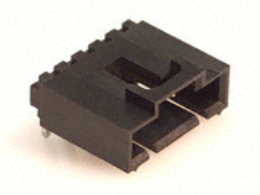
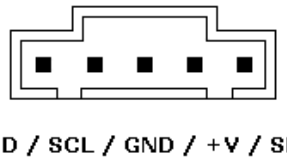
- TX - Transmit Data from the Host Computer to the i2cStick.
- RX - Receive Data from the i2cStick to the Host Computer.
- RTS - Request to Send from the Host Computer to i2cStick.
- CTS - Clear to Send from the i2cStick to the Host Computer.

2.2 I^2C Interface Connector

The i2cStick includes a five wire (1x5) 2.54 mm (.100"), positive locking, shrouded header connector for interfacing to an external I^2C Bus. Lines provided in this connector include I^2C Bus Clock (SCL), I^2C Bus Data (SDA), Ground (GND), Shield (SHD) (optional), and Power (+V) (optional).

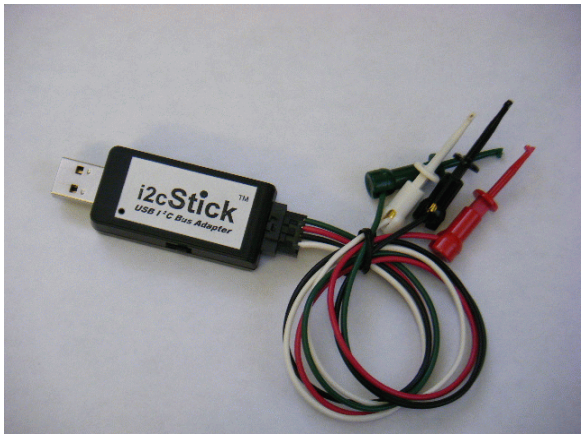
Minimum wiring for I^2C Bus communications include I^2C Bus Clock, Data, and Ground. Use of the Power and Shield wires in the I^2C Interface connector are user optional.

Connect the Power (+V) wire to the target system to power the target system from USB bus power. The maximum available I^2C Bus power is 50 mA at +5V (or +3.3V with 3.3V option).

	
I^2C Bus Connector	I^2C Bus Pinout

An optional I^2C Bus Mini Interface Cable (#I2CMIC) 1x5 2.54 mm (.100") is available to connect the I^2C adapter to an external I^2C Bus.

An I^2C Bus Mini Clip-Lead (#I2CMCL) cable is included to connect the I^2C adapter to a target system.



Each cable wire is color-coded or clearly marked (SCL=C=White, +V=V=Red, SDA=D=Green, Ground=G=Black). +V and Shield (SHD) use are user optional.

3 Hardware Configuration

3.1 Pull-up Resistors

I^2C Bus systems are based on open-collector technology requiring pull-up devices on each signal wire. These pull-up devices usually take the form of pull-up resistors connected to bus power.

The I^2C adapter includes a side-mounted slide switch used to enable or disable internal 5V (or 3.3V) I^2C Bus 1.8K ohm pull-up resistors attached to the SCL and SDA lines. Every I^2C Bus system must have at least one pull-up on the signal lines. Use this switch to configure the pull-up resistors for your system.

3.2 Connecting a 5V i2cStick to a 3.3 Volt Target System

If you are connecting a 5 volt I^2C adapter to a 3.3 volt target system, you should follow these steps BEFORE applying power:

- Shut off the I^2C adapter's internal pull-ups (See Pull-up Resistor section). Use external pull-ups to the target system's 3.3V power. These pull-ups may already be present in the target system.
- Disconnect the I^2C connector +5V wire from the target system. The I^2C adapter will be powered from its USB interface, and the target system will be powered by its own 3.3V power supply.

The I^2C adapter is a 5-volt device. Any signal above 3.3V on the SCL or SDA lines is high enough for the adapter to see a Logical 1.

3.3 Connecting to an SMBus Target System

If you are connecting the I^2C adapter to a SMBus target system, you should follow these steps BEFORE applying power:

- Shut off the I^2C adapter's internal pull-ups (See Pull-up Resistor section).
- Use external SMBus rated (appoximately15k ohms) pull-up resistors. These pull-ups may already be present in the target system.
- Visit our I^2C versus SMBus FAQ page (www.mcc-us.com/I2CSMBusFAQ.htm).
- See the SMBus Specification for additional details.

Special Note for SMBus Users: MCC's I^2C adapters are designed to be I^2C Bus

compatible, not SMBus compatible. Some features of the SMBus protocol not supported include time-outs, device reset, and Packet Error Check byte processing. The non-supported SMBus features may, or may not, permit the use of the I^2C adapter in your SMBus application. Consult the MCC FAQ web page and SMBus Specification for details.

4 ESD (Electrostatic Discharge) Precautions

Electrostatic discharge is defined as the transfer of charge between bodies at different electrical potentials. Electrostatic discharge can change the electrical characteristics of a semiconductor device, degrading or destroying it. Electrostatic discharge also may upset the normal operation of an electronic system, causing equipment malfunction or failure.

When connecting the I^2C adapter to a host computer and a target system, extreme care must be taken to avoid electrostatic discharge. Failure to follow ESD protection procedures when using the I^2C adapter could damage the host computer, I^2C adapter, or the target system, and void product warranty coverage.

4.1 Host Computer Grounding

Case 1 - Desktop and Single-board Computers. The chassis on a desktop or single-board host computer must be connected to earth ground to comply with safety regulations. If the computer chassis is NOT connected to earth ground for some reason (i.e., use of a two-prong power mains plug), the host computer power supply ground will float to some unknown voltage potential.

Case 2 - Laptop Computers. Laptop computers present special ESD problems. Most laptop computers use an external double-insulated mains power supply which is NOT connected to the mains earth ground. This means that the laptop chassis is floating at some unknown voltage potential.

In either case, upon connection to the I^2C adapter and the target system, the host computer will discharge energy through its serial port to the I^2C adapter, and on to the target system. This discharge could damage the host computer, I^2C adapter, and the target system.

4.2 Grounding Solutions

To avoid damage to the host computer, I^2C adapter, or target system, follow these instructions:

- Wear an earth grounded wrist strap, or discharge any static charge build-up, when handling the I^2C adapter or any target system devices.
- Ensure that both the host computer and target system are connected to a common earth ground point.
- Make sure that all interconnections are made BEFORE applying power to the host computer, I^2C adapter, and target system.
- If you are using a laptop computer or host computer that is NOT connected to mains earth ground, make a hard-wired connection from the host computer (i.e., port connector shell) and the target system ground connector to a common earth ground point.
- Avoid plugging and unplugging system components while the host computer or target system is powered.
- Ensure that any devices connected to the target system are properly grounded to the common earth ground point.
- If unsure how to properly ground system components, seek electrical expert help.

WARNING: Any damage caused by Electrostatic Discharge (ESD) through inadequate earth grounding is NOT covered under the warranty of this product.

5 Driver Software Set-Up

This section provides information on how to install, update, and uninstall the i2cStick software driver.

1. Driver Install

   The i2cStick uses a Virtual Communications Port (VCP) driver that is pre-installed with the iPort Utility Pack CD (See the installation instructions in “Part 2 - iPort Utility Pack for Windows” of this User’s Guide). Pre-installation places the VCP driver into the Windows Driver Store, ready for installation when the i2cStick is first plugged into the host computer.

   After iPort Utility Pack installation, the VCP driver may also be pre-installed with the Driver Install short-cut on the iPort Utility Pack Start menu.

For backup, VCP driver files can be found on the iPort Utility Pack CD, and after iPort Utility Pack software installation, in the Program Files installed i2cStick Driver folder on the host computer.

2. Driver Update

   i2cStick VCP drivers are posted on the MCC website (http://www.mcc-us.com/SoftwareUpgrades-Updates.htm#i2cstick). Use Windows Device Manager (Start | Settings | Control Panel | System | Device Manager | Ports (COM & LPT)) to see the current version of the i2cStick driver installed on your computer, and determine if new driver is available. If a new VCP driver is available, follow website instructions to download and install a driver update on your computer.

3. Driver Uninstall

   i2cStick VCP drivers can be uninstalled using Windows Device Manager (Start | Settings | Control Panel | System | Device Manager | Ports (COM & LPT)), or the Driver Uninstall short-cut on the iPort Utility Pack Start menu.

6. Hardware Set-Up

This section provides information on connecting the i2cStick to your host computer and I^2C Bus target system.

1. USB Connection

   After completing the Driver Installation instructions above, plug the i2cStick adapter into a free USB port on your host computer or self-powered USB hub. If this is the first time the i2cStick is connected to the host computer, Windows will automatically install the VCP driver from the Windows Driver Store and assign the i2cStick to a communications port number (COMn). You can find the ComPort number assigned to the i2cStick by running the iPort Utility Pack Message Center or Message Manager software, and selecting the i2cStick device.

   You can use Windows Device Manager (Start | Settings | Control Panel | System | Device Manager | Ports (COM & LPT)) to find or reassign the ComPort number assigned to the i2cStick.

2. I^2C Bus Connection

   Connect the I^2C Bus cable to the I^2C adapter and your I^2C device. You can make this connection with the i2cMini Interface or i2cMini Clip-Lead cable.

   The I^2C interconnect includes a +5V (or 3.3V) wire. You may not need to, or want to, connect the +V wire to your target system. Refer to the “Hardware Configuration” section for details on pull-up resistors and connecting the optional +V wire.

If you have any questions on I^2C adapter setup and configuration, please visit our FAQ page (http://www.mcc-us.com/faq.htm), or contact our technical support team (support@mcc-us.com).

Part 2

iPort Utility Pack for Windows

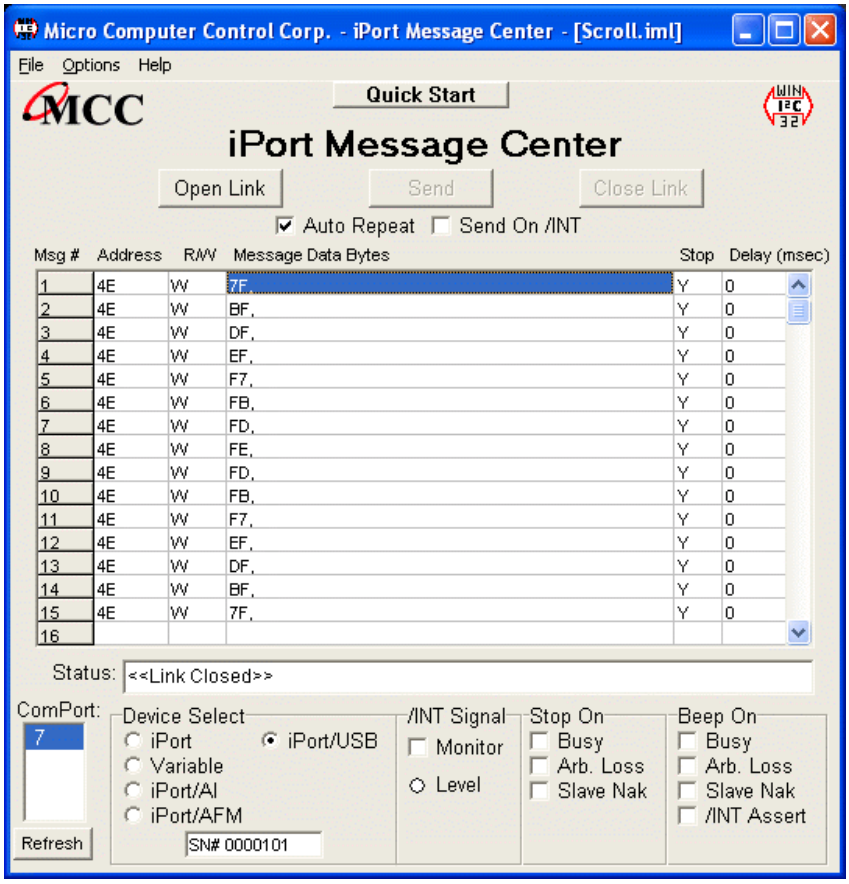
1 iPort Utility Pack for Windows

The iPort Utility Pack for Windows is your express lane to I²C Bus communications. The Utility Pack includes two (2) Windows-based applications (Message Center and Message Manager) that will help you get started sending and receiving I²C Bus messages quickly and easily.

1.1 iPort Message Center

The iPort Message Center, our most popular application, operates with all versions of our I²C Bus Host Adapters. With the Message Center, you can create, save, and automatically execute scripts of I²C Bus messages. I²C Bus message activity includes:

- Master Transmit
- Master Receive

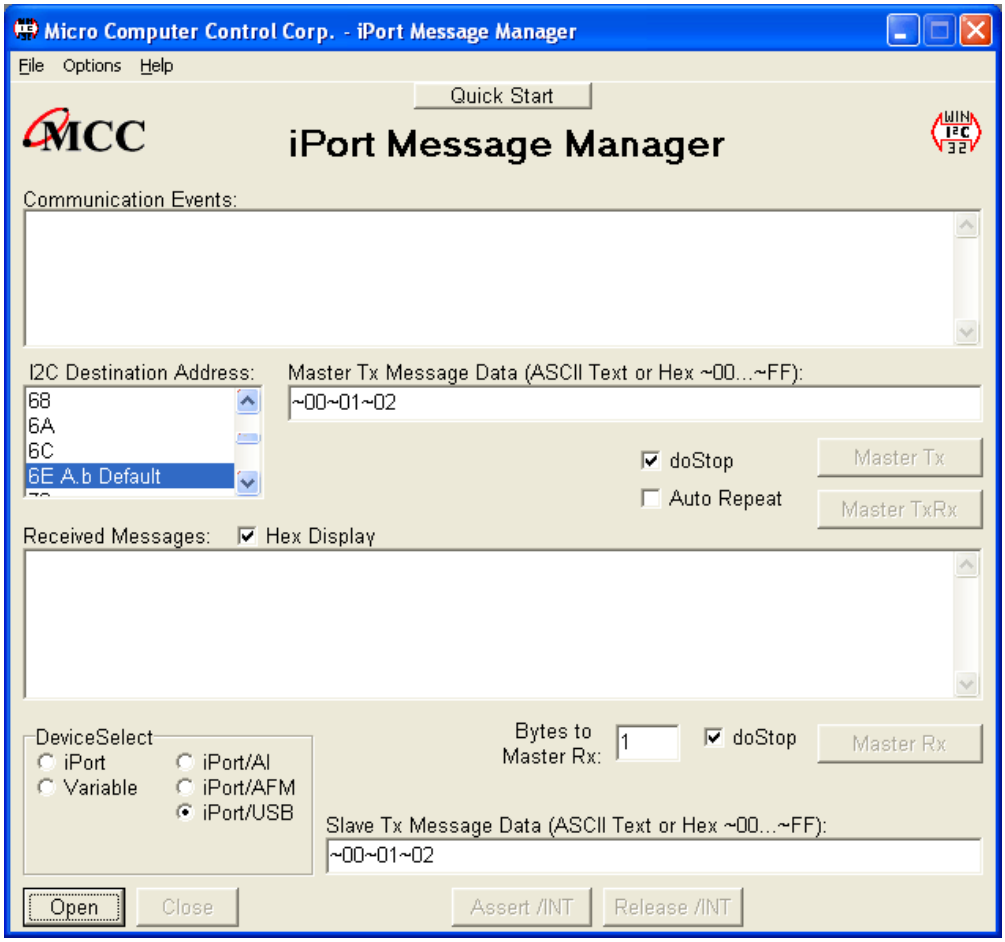


Main Screen (Typical)

1.2 iPort Message Manager

The iPort Message Manager operates with all versions of our I^2C Bus Host Adapters. Using the Message Manager, you can perform all four (4) modes of I^2C Bus message activity, including:

- Master Transmit
- Master Receive
- Slave Transmit
- Slave Receive



Main Screen (Typical)

2 System Requirements

One of the following MCC I^2C Bus adapters:

1. i2cStick (#MIIC-207) USB to I^2C Bus Host Adapter.
2. iPort/USB (#MIIC-204) USB to I^2C Bus Host Adapter.
3. iPort/AFM (#MIIC-203) RS-232 to I^2C Bus Host Adapter with ASCII Fast Mode Interface.
4. iPort/AI (#MIIC-202) RS-232 to I^2C Bus Host Adapter with ASCII Interface
5. iPort (#MIIC-201) Windows to I^2C Bus Host Adapter.
6. iPort DLL/USB (#MIIC-201D/U) I^2C Bus Host Adapter.
7. Variable Clock Rate (#MIIC-201-V) I^2C Bus Host Adapter.

Windows 2000, XP, Vista, 7, or higher.
1 free RS-232 Serial Port, or USB port for USB-based adapters.

3 iPort Utility Pack Installation

3.1 Installing from CD

1. Insert a software distribution CD into your CD drive.
2. If the install program does not start automatically, select Start | Run and type "D:SETUP.EXE." Click OK.
3. Follow instructions on screen.

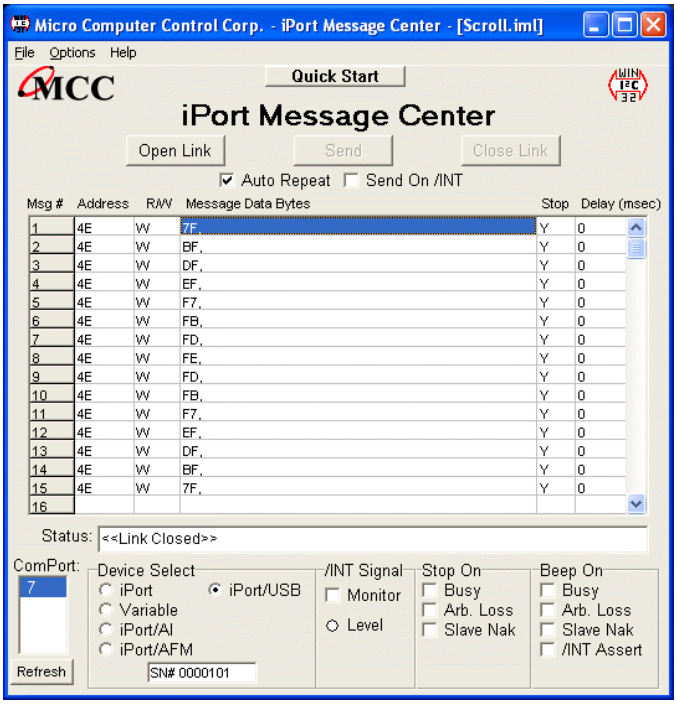
3.2 Installing from the Web

Visit MCC's web site (www.mcc-us.com), and click on the Upgrades/Updates link.

1 Follow the instructions listed on the Upgrade/Update web page for your specific adapter.

4 iPort Message Center

The iPort Message Center supports I²C Master Transmit and Master Receive activities for all MCC I²C Bus host adapters. With this program you can create, save, and execute scripts of I²C Master messages.



Main Screen (Typical)

The iPort Message Center allows a PC to become an I²C Master transmitter or receiving device, sending I²C messages between the PC and one or more I²C devices across an I²C Bus.

The iPort Message Center is designed to be a simple application for experimenting with I²C messages. It provides methods to:

1. Enter/Edit a list of I²C Master Transmit or Receive Messages.
2. Save and/or Load a list of I²C Master messages to/from disk.
3. Transmit the current list of I²C Master messages, with the option to auto-repeat upon completion, send on INT signal assertion (with INT signal supported adapters only), and beep or stop on special I²C Bus events.
4. Use command line arguments to automatically load, send, and save I²C messages from a batch file or another program.

Each I²C message can transfer up to 32 bytes of 8-bit data, with Repeated Start and Time Delay options.

4.1 Message Center Operations

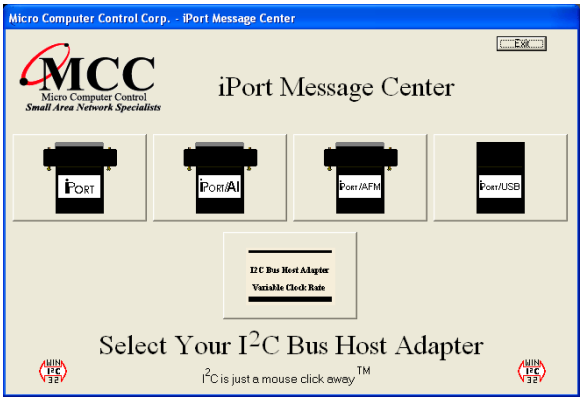
Communicating with another device on the I²C Bus is easy. Just install the software as described in Section 3, then follow these simple steps:

4.1.1 Starting the Message Center

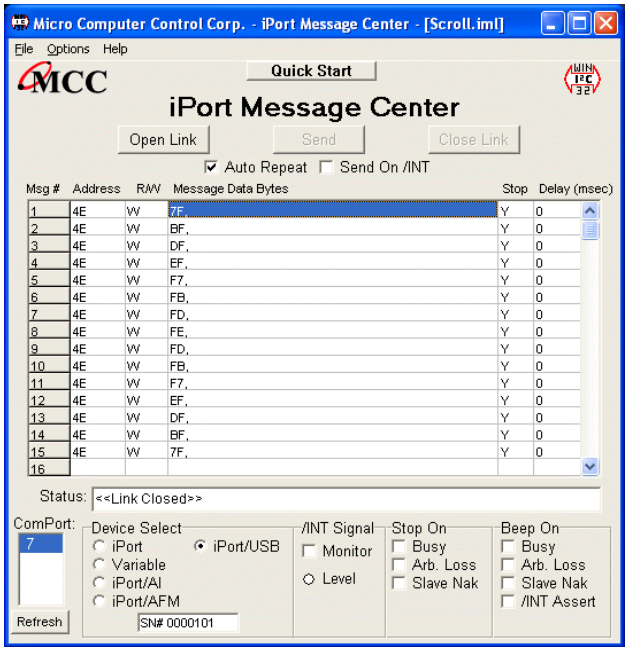
Click, Start │Programs │iPort Utility Pack│iPort Message Center

4.1.2 Selecting the Adapter

Select the I²C adapter you are using by clicking the corresponding adapter image (see Opening Screen), or the Device Select checkbox (see Main Application Screen).



Opening Screen (Typical)



Main Screen (Typical)

4.1.3 Select the Communications Port

Use the “ComPort:” control to select the communication port connected to the I^2C adapter. If a USB-based device is selected, the serial number for the adapter is displayed (Win 2000, XP+ only). In addition to legacy RS-232 ports and USB-based Virtual Communication Ports, Message Center supports USB and network connected local or remote RS-232 ports via the Windows Com driver.

4.1.4 Options Menu

Use the Options menu to override default Baud Rate and I^2C Bus Clock rate settings. Default settings and options are adapter dependant.

4.1.5 Establish Adapter Communications Link

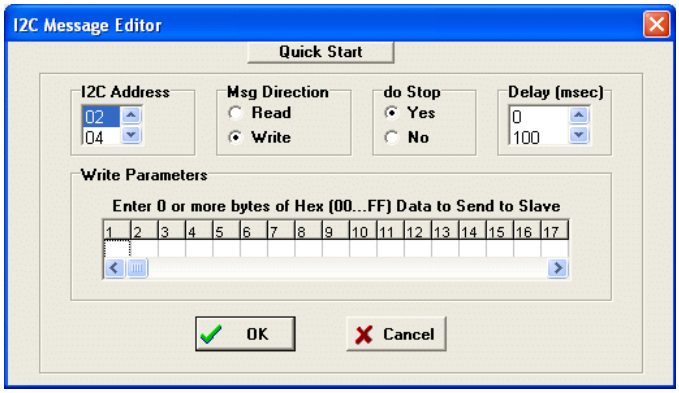
Establish the communications link to the I^2C adapter by clicking the Open Link button.

The Message Center sets the adapter’s own I^2C Slave address to 0xFE. Once the link has opened successfully, you are an active I^2C node. I^2C messages entered into the message spreadsheet can be transmitted upon request. If the link open is not successful, follow the on-screen directions. Make sure the communications port is working, is enabled in the Windows Device Manager, and is not being used by other software.

4.1.6 Entering or Editing I^2C Messages

I^2C messages can be entered with the Message Editor, or a previously recorded message list can be loaded from the File menu.

To enter or edit a message, open the “I^2C Message Editor” screen by double clicking on a message row in the spreadsheet.



Use the I^{2}C Message Editor to:

1. Set I^{2}C Address.

   The I^{2}C Address is the I^{2}C slave address of the slave device being addressed on the bus. All slave addresses are displayed as even numbers (00...FE), representing the 7 most significant bits of the 8-bit slave address transmitted on the bus (aaaa aaa0).

   The I^{2}C adapter automatically supplies the 8th, least significant, Read/Write bit when it sends the slave address across the bus. For master write operations, the Read/Write bit is always transmitted as a logical 0 (aaaa aaa0). For master read operations, the Read/Write bit is always transmitted as a logical 1 (aaaa aaa1).

   Use the I^{2}C Address control to set the slave address of the slave device you want to address on the bus.

2. Set Message Read/Write Direction.

   As a bus master device, the I^{2}C adapter can write data to, or read data from, any device on the bus. Use the Msg Direction control to specify if the current message is a master write, or master read, operation. Upon making your selection, additional Write or Read parameters appear.

3. Specify Repeated Start Messages.

   I^{2}C Bus communications support an operation called Repeated Start. In this operation, a message is sent across the bus beginning with a Start Condition, but without a Stop Condition at the end of the message. The next message sent

across the bus begins with a Start Condition, in this case a Repeated Start.

An I^2C Bus master, that successfully sends a message on the bus, owns the bus until that master sends a message with a terminating Stop Condition. The Repeated Start operation allows the bus master to retain control of the bus while sending one or more messages on the bus. This prevents other bus masters, in a multi-master system, from accessing the bus and interfering with message sequences.

The Message Center supports Repeated Starts with the doStop control. Sending an I^2C message with doStop enabled will cause the message to be terminated with a Stop Condition. Sending an I^2C message with doStop disabled will cause the message to end without a Stop Condition, allowing the next message to be sent with a Repeated Start.

4. Set Time Delay.

   Message Center supports time delays after the completion of a message. Time delays can be used to synchronize or sequence bus messages with the activity of external devices.

5. Specify Write Data or Read Byte Count.

   Enter the hexadecimal data you want to write to a slave receiver device, or the number of data bytes to read from a slave transmitter. Message Center supports up to 32 bytes of 8-bit data per message.

   NOTE: The data you send may have special meaning to the receiving slave device, but to the Message Center, and the I^2C adapter, message data has no special meaning. Consult your slave device's data sheet for details.

Click OK to accept the message and enter it into the spreadsheet.

Master Write messages display the message data in the spreadsheet. Master Read messages display 0xFF placeholders in the spreadsheet. Upon execution, actual data received from a slave transmitter replaces the placeholders in the message spreadsheet.

Repeat above steps for additional messages. The Message Center supports up to 32,000 messages in a list.

4.1.7 Inserting and Deleting Messages

You can insert a new message between existing messages by clicking once on a message below where you want to insert, then press the “Insert” key on your keyboard. The Message Editor also remembers the last message displayed, so double clicking on a blank spreadsheet row will allow you to copy a message. Delete a message by single clicking on the message row and pressing the “Delete” key on your keyboard.

4.1.8 Saving or Loading Message Lists

Message Center I^2C message lists can be saved to, or loaded from, a disk file. To save the current message list, click File|Save on the menu bar. To open an existing message list, click File|Open List on the menu bar.

Message lists are maintained in ASCII text files (*.IML) that can be edited manually or created with a customer-developed program. See message list files for details.

4.1.9 Send the Message List

An I^2C message list can be sent manually, or automatically in response to an INT signal assertion (with INT signal supported adapters only). To send the list manually, click the Send button on the main application screen. To send the list in response to an INT signal assertion (low), enable the “/INT Signal Monitoring” checkbox, and check the “Send on /INT” checkbox. The list will be sent each time the INT signal is asserted.

The Message Center also supports the repeated sending of a message list. If the Auto Repeat checkbox is checked, a message list will automatically repeat upon completion.

4.1.10 Special Event Handling

The Message Center supports the early termination of a message list, and beep on special events. See the “Stop On” and “Beep On” controls on the main application screen of available options.

4.1.11 Slave Not Acknowledging

If you get a “Slave Not Acknowledging” message in the Status window, this could indicate you have the wrong address in the I^2C Destination Address, or the device is not answering to its address. Some slave devices temporarily stop acknowledging their address. Consult the slave device’s data sheet for details.

4.2 Command Line Arguments

The Message Center can be controlled via command line arguments. This feature allows the Message Center to be accessed from a batch file or another program.

Message Center Command Line Arguments	
Command	**Description**
iPort, iPort/AI, iPort/AFM, Variable, iPort/USB, i2cStick	Specify I^2C adapter type.*
COM1...COM99	Specify RS-232 communication port.
BAUD19200, BAUD57600, BAUD115200	Set RS-232 Baud Rate.*
CLOCK12.5K, CLOCK23K, CLOCK86K, CLOCK100K, CLOCK400K, VCLOCK	Set I^2C Bus Clock Rate.*
Monitor/INT	Enable /INT Signal Monitor.*
StopOnBusy	Stop sending on I^2C adapter busy.
StopOnArbLoss	Stop sending on I^2C Bus Arbitration Loss.
StopOnNak	Stop on Slave Negative Acknowledgment.
BeepOnBusy	Beep on I^2C adapter busy.
BeepOnArbLoss	Beep on I^2C Bus arbitration loss.
BeepOnNak	Beep on Slave Negative Acknowledgment.
BeepOn/INT	Beep on /INT signal assert (low).*
AutoLoad	Load I^2C message list file.
AutoSave	Save I^2C message list file.
AutoOpen	Open link to I^2C adapter.
AutoSend	Send I^2C message list.
AutoExit	Exit after sending message list.

* Adapter specific commands. See command details below.

Command Line Syntax: imsgctr.exe AdapterType argument-list

Example: imsgctr.exe iPort/AFM adctest01.iml AutoOpen AutoSend AutoExit

4.2.1 Set Adapter Type

i2cStick	i2cStick (#MIIC-207)
iPort/USB	iPort/USB (#MIIC-204)
iPort/AFM	iPort/AFM (#MIIC-203)
iPort/AI	iPort/AI (#MIIC-202)
iPort	iPort (#MIIC-201)
Variable	Variable Clock (#MIIC-201-V)

The Adapter Type argument should be the first argument in the argument list as it controls the availability of other arguments. If the Adapter Type is not specified, the startup adapter selection screen will be presented.

4.2.2 Set RS-232 Communication Port

1st Available ComPort (Default)
COM1...COM99

Set the RS-232 communications port attached to the I^2C adapter.

4.2.3 Set RS-232 Baud Rate

BAUD19200	(Default*)
BAUD57600	(iPort/AFM, iPort/USB, i2cStick* ONLY)
BAUD115200	(iPort/AFM, iPort/USB, i2cStick* ONLY)

Set the RS-232 Baud Rate. *i2cStick internally re-maps to 115.2K baud.

4.2.4 Set I^2C Bus Clock Rate

CLOCK12.5K	(iPort ONLY)
CLOCK23K	(iPort/AFM, iPort/USB, i2cStick ONLY)
CLOCK86K	(iPort/AFM, iPort/USB, i2cStick ONLY)
CLOCK100K	(iPort, iPort/AI, iPort/AFM, iPort/USB, i2cStick, Default)
CLOCK400K	(iPort/AFM, iPort/USB, i2cStick ONLY)
VCLOCK=nnnHz	(Variable ONLY. nnn=451...57787)

Set the I^2C Bus Clock Rate to the specified value. The defaults rate for the Variable Clock adapter is 451Hz. The Variable Clock adapter does not support all rates within the specified range. The Message Center will adjust the specified rate to the

nearest available supported rate.

4.2.5 Enable /INT Signal Monitor

Monitor/INT (on INT signal supported adapters only. Default=OFF)

Enable /INT signal monitoring.

4.2.6 Stop On Busy

StopOnBusy (Default=OFF)

Stop sending I^2C messages if the adapter returns a "Busy" response to the host computer.

4.2.7 Stop On Arbitration Loss

StopOnArbLoss (Default=OFF)

Stop sending I^2C messages if the adapter returns a "Bus Arbitration Loss" response to the host computer. Bus Arbitration Loss occurs when another I^2C Bus master wins arbitration while the adapter is attempting to become a bus master.

4.2.8 Stop On Slave Negative Acknowledgment

StopOnNak (Default=OFF)

Stop sending I^2C messages if the adapter returns a "Slave Not Acknowledging" response to the host computer. Slave Not Acknowledging occurs when the adapter is attempting to become a bus master and no slave device acknowledges the transmitted slave address.

4.2.9 Beep On Busy

BeepOnBusy (Default=OFF)

Generate a host computer beep if the adapter returns a "Busy" response to the host computer.

4.2.10 Beep On Arbitration Loss

BeepOnArbLoss (Default=OFF)

Generate a host computer beep if the adapter returns a "Bus Arbitration Loss" response to the host computer. Bus Arbitration Loss occurs when another I^2C Bus master wins arbitration while the adapter is attempting to become a bus master.

4.2.11 Beep On Slave Negative Acknowledgment

BeepOnNak (Default=OFF)

Generate a host computer beep if the adapter returns a "Slave Not Acknowledging" response to the host computer. Slave Not Acknowledging occurs when the adapter is attempting to become a bus master and no slave device acknowledges the transmitted slave address.

4.2.12 Beep On /INT Assert

BeepOn/INT (on INT supported adapters only. Default=OFF)

Generate a host computer beep if the adapter returns an "/INT Signal Assert" response to the host computer. /INT Signal Assert occurs if /INT Signal Monitoring is enabled and a high to low transition is detected on the adapter /INT signal connector.

4.2.13 Load I^2C Message List File

AutoLoad=filename
AutoLoad="file name"
filename.iml
"file name.iml"

Automatically open file with extension .IML and load messages into Message Center spreadsheet.

4.2.14 Saved I²C Message List File

AutoSave=filename
AutoSave="file name"

Automatically save message list to the specified file upon executing AutoExit. Use to save message data read from a slave transmitter device.

4.2.15 Auto Open

AutoOpen Auto Open Link to I²C Adapter

Open link to the adapter.

4.2.16 Auto Send

AutoSend Auto Send I²C Message List

Send I²C messages loaded with the AutoLoad command.

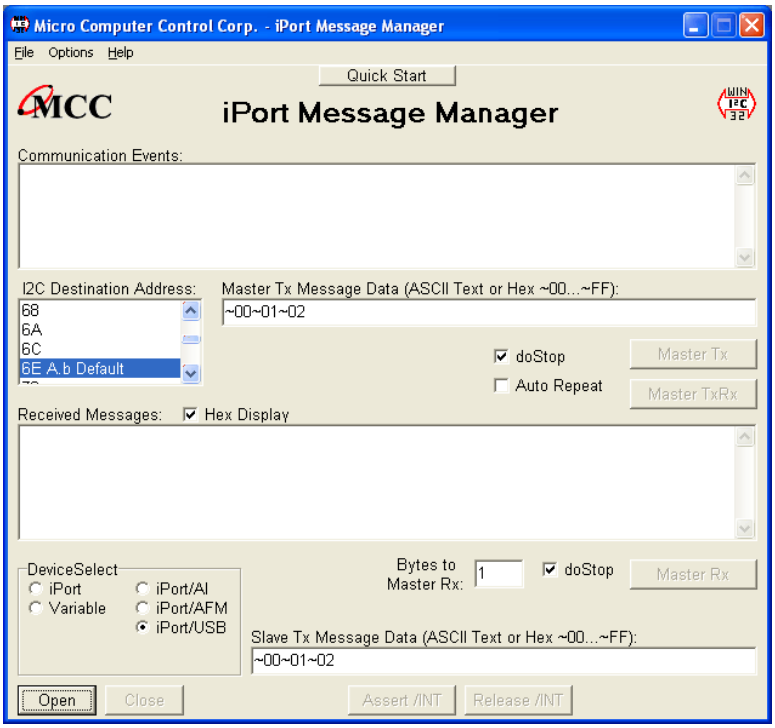
4.2.17 Auto Exit

AutoExit Auto exit after sending the message list.

Message Center will auto exit after sending the last message in the I²C message list.

5 iPort Message Manager

The iPort Message Manager supports I²C Master and Slave, Transmit and Receive activities for all MCC I²C Bus host adapters, allowing a PC to become an I²C Master or Slave device, transmitting or receiving I²C messages between the PC and one or more I²C devices across an I²C Bus.



Main Screen (Typical)

The Message Manager is designed to be a simple application for experimenting with I²C messages. Message Manager provides methods to:

1. Set the I²C adapter's own I²C Slave address, General Call Enable, and other operating parameters.
2. Master Transmit ASCII text or Hex (00...FF) data to a specified I²C Slave Receiver device.
3. Master Receive data from a specified I²C Slave device.
4. Perform Master Read after Write operations.
5. Slave Transmit data to a requesting I²C Master device.
6. Display Master or Slave Receive data in hexadecimal or ASCII.
7. Display I²C Bus communication events.
8. Assert or release the INT signal (on supported adapters only).

5.1 Message Manager Operations

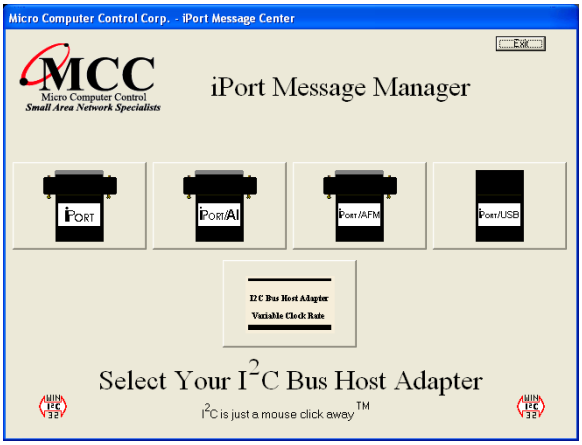
Communicating with another device on the I²C Bus is easy. Just install the software as described in Section 3, then follow these simple steps:

5.1.1 Starting the Message Manager

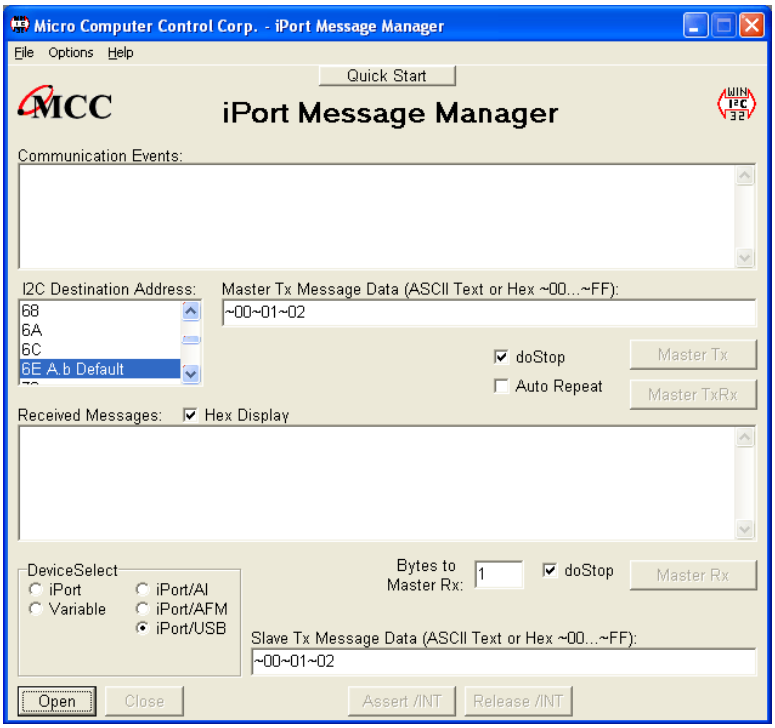
Click, Start │Programs │iPort Utility Pack│iPort Message Manager

5.1.2 Select the Adapter

Select the I²C adapter you are using by clicking the corresponding adapter image (see Opening Screen), or the Device Select checkbox (see Main Screen).



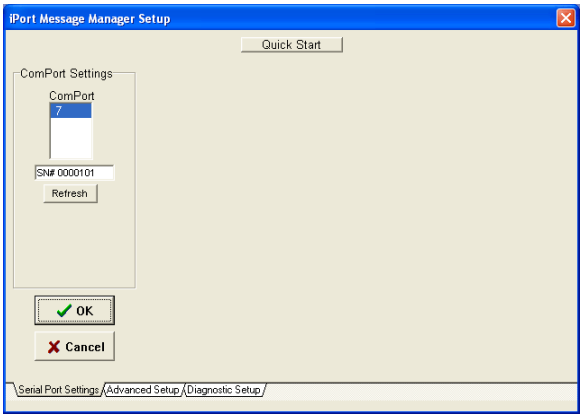
Opening Screen (Typical)



Main Screen (Typical)

5.1.3 Establish Adapter Communications Link

On the main screen, click the Open button to view the Set Up Screen. Three levels of setup options are available, Basic, Advanced, and Diagnostic. Only Basic setup is required.



Basic Set Up Screen

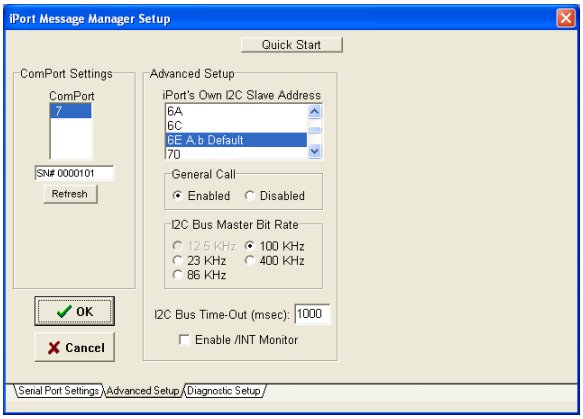
5.1.3.1 Basic Setup

Use the "ComPort" control to select the communication port connected to the I^2C adapter. If a USB-based is selected, the serial number for the selected adapter is displayed (Win 2000, XP+ only). In addition to RS-232 and USB-based adapters, Message Manager supports USB and network connected local or remote RS-232 ports via the Windows Com driver.

Select from the list of available baud rates. Then click OK.

After a few moments, the Communication Events window on the Main Application screen should report "I^2C Open Successful."

If open is not successful, follow the on-screen instructions. Make sure the communications port is working, is enabled in the Windows Device Manager, and is not being used by other software. Additional communication port open information is available in the log file. See Diagnostic Setup options.



Advanced Set Up Screen

5.1.3.2 Advanced Setup

On the Advanced Setup screen you can set the following parameters:

Adapter's Own I^2C Slave Address

Select the I^2C adapter's own slave address. The adapter will acknowledge messages sent to this slave address. The default address is 0x6E.

General Call Enable

General Call Enable allows the I^2C adapter to respond as a slave receiver to the I^2C General Call Address (0x00). General Call is used by a master to broadcast an I^2C message to multiple devices. The default value is enabled.

I^2C Bus Master Bit Rate

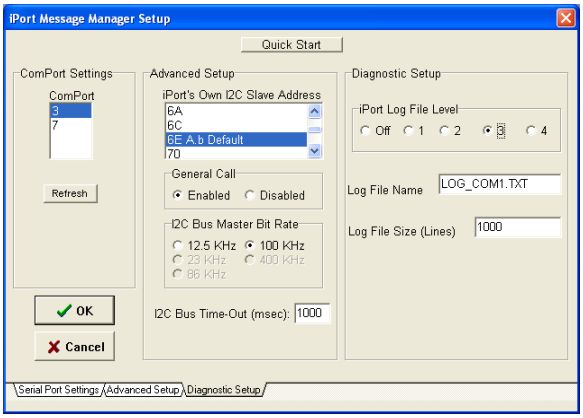
Select I^2C Bus speed during master operations. 100kHz is standard mode. 400kHz is fast mode. Available rates are I^2C adapter dependant.

I^2C Bus Time-Out

Specify how long the I^2C adapter will wait before reporting an I^2C Bus inter-byte time-out (0 = no time-out, 1 to 32767 milliseconds, iPort/AI fixed at 1 second).

Enable INT Signal Monitor (on supported adapters)

Enables monitoring of the INT signal state. INT state changes are reported in the main screen Communications Events window.



Diagnostic Set Up Screen

5.1.3.3 Diagnostic Setup (on supported adapters)

On the Diagnostic Set-up screen you can set the following parameters:

Log File Level

A log file is available for troubleshooting communication problems between the host computer and the I^2C adapter. The log file is an ASCII text file viewable with any text editor. Select logging level. Level 1 provides minimum information. Level 4 provides maximum information.

Log File Name

Specify a log file name. Unless a path is specified, the log file will be created in the current working directory.

Log File Size

Specify log file length in lines. The log file overwrites earlier entries upon reaching the specified number on lines.

5.1.4 Sending I²C Messages

5.1.4.1 Master Operations

5.1.4.1.1 Specifying the Destination Address

The Destination Address is the I²C slave address of the slave device being addressed on the bus. All slave addresses are displayed as even numbers (00...FE), representing the 7 most significant bits of the 8-bit slave address transmitted on the bus (aaaa aaa0).

The I²C adapter automatically supplies the 8th, least significant, Read/Write bit when it sends the slave address across the bus. For master write operations, the Read/Write bit is always transmitted as a logical 0 (aaaa aaa0). For master read operations, the Read/Write bit is always transmitted as a logical 1 (aaaa aaa1).

On the main screen, use the I²C Destination Address list control to set the slave address of the slave device you want to address on the bus.

5.1.4.1.2 Repeated Start Messages

I²C Bus communications support an operation called Repeated Start. In this operation, a message is sent across the bus beginning with a Start Condition, but without a Stop Condition at the end of the message. The next message sent across the bus begins with a Start Condition, in this case a Repeated Start.

An I²C Bus master, that successfully sends a message on the bus, owns the bus until that master sends a message with a terminating Stop Condition. The Repeated Start operation allows the bus master to retain control of the bus while sending one or more messages on the bus. This prevents other bus masters, in a multi-master system, from accessing the bus and interfering with message sequences.

The Message Manager supports Repeated Starts with the doStop checkbox. Sending an I²C message with doStop checked will cause the message to be terminated with a Stop Condition. Sending an I²C message with doStop unchecked will cause the message to end without a Stop Condition, allowing the next message to be sent with a Repeated Start.

5.1.4.1.3 Auto Repeat

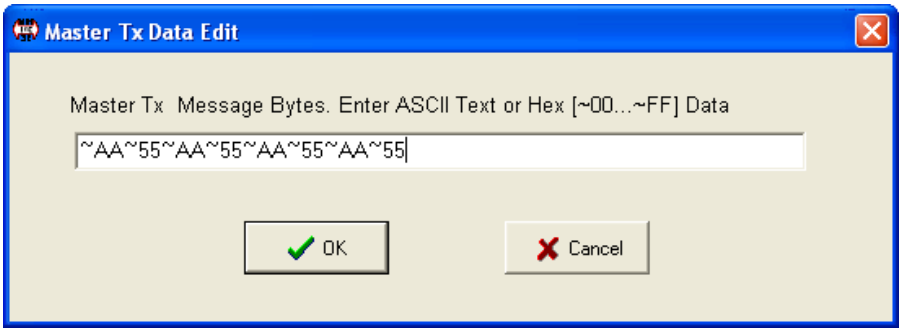
The situation often arises, where you would like to automatically repeat a master message operation.

The Message Manager supports auto-repeat with the Auto Repeat checkbox. You can automatically repeat a master operation by checking the Auto Repeat control before clicking the Master Tx, Master Rx, or Master TxRx buttons. The master operation repeats until the Auto Repeat control is unchecked.

5.1.4.1.4 Master Transmitting Data

Specifying Master Tx Message Bytes

Master Tx Message Bytes is the ASCII or Hexadecimal data you want to transmit to a slave receiver device. With the Message Manager, entering master transmit data is easy. On the main application screen, click on the Master Tx Message Bytes box to open the data editor.



In the data editor, enter one or more ASCII text characters or hexadecimal data bytes. Each hexadecimal byte is entered as two ASCII-Hex characters (00 to FF) preceded by a tilde (~) character. ASCII text and hex data can be intermixed, as long as each hex byte is preceded by a tilde.

For example, to enter hex data bytes 0x00, 0x01, and 0x02, enter the characters ~00~01~02 into the text box.

Each iPort Message Manager I^2C message can include up to 80 bytes of 8-bit ASCII binary data.

NOTE: The data you send may have special meaning to the receiving slave device, but to the Message Manager, and the I^2C adapter, message data has no special meaning. Consult your slave device's data sheet for details.

Click OK to accept the data.

Sending Master Transmit Messages

Click the Master Tx button to write the specified Master Tx Data Bytes to the selected destination slave device. If Auto Repeat is checked, the message will automatically repeat upon completion.

The Communications Events window on the main screen should report “Master Tx Complete.” If this message does not appear, check the slave device address, connections, and power.

If you get a “Slave Not Acknowledging” message in the Communications Events window, this could mean you have the wrong address in the I^2C Destination Address, or the device is not answering to its address. Consult your slave device’s data sheet for details.

5.1.4.1.5 Master Receive Data

Specifying Data to Read

On the lower part of the main screen, set the Bytes to MasterRx edit box to the number of bytes you want to read. For example: Set this to 1 to read a single byte. Click on the MasterRx button to read the data from the selected slave device.

Data received from the slave is displayed in the Received Messages text box on the main screen. The Communications Events window should report “Master Rx Transfer Complete.” If this message does not appear, check the slave device address, connections, and power.

If you get a “Slave Not Acknowledging” message in the Communications Events window, this could mean you have the wrong address in the I^2C Destination Address, or the device is not answering to its address. Consult your slave device’s data sheet for details.

Negative Acknowledge Last Byte

On supported adapters, the doNak checkbox gives you the option to acknowledge, or negatively acknowledge, the last byte read from a slave device. Some Slave Transmitter Devices require a negative acknowledgment on the final byte read from the slave device. I^2C adapters not supporting this option automatically negatively acknowledge the last byte read.

5.1.4.1.6 Master Transmit and Receive

The Master TxRx button sends a master write message with no Stop Condition, immediately followed by a Repeated Start master read message with Stop.

5.1.4.2 Slave Operations

In addition to performing I^2C Bus master operations, the Message Manager can also perform I^2C bus slave transmit and receive operations.

5.1.4.2.1 Slave Transmit Data

Slave transmit data is entered in the Slave Tx Message Bytes text box control on the main screen. Data in this text box is automatically sent to a requesting master upon receiving a slave transmit request.

Like Master Transmit data, Slave Transmit data is entered with the data editor. To enter data to be transmitted, click on the Slave Tx Message Bytes text box to open the data editor. See “Specifying Master Tx Message Bytes” section for data entry details.

5.1.4.2.2 Slave Receive Data

Data bytes received from a Master Transmitter are automatically displayed in the main application screen Received Message window. Received data is displayed in ASCII printable, or hexadecimal (~00 to ~FF) formats. Use the Hex-Display checkbox to force ASCII printable data to display in hexadecimal format .

6 Uninstalling Software Components

Software components include the iPort Utility Pack for Windows, and for USB-based adapters, the Virtual Communications Port (VCP) Device Driver. The following instructions can be used to remove either or both software components from your computer.

6.1 Uninstalling iPort Utility Pack for Windows

To uninstall the iPort Utility Pack for Windows software, use the Windows Control Panel “Programs and Features” (formerly “Add or Remove Programs”) utility. Note that uninstalling the iPort Utility Pack for Windows software does not uninstall the device or driver software.

6.2 Uninstalling VCP Device Driver

The VCP Device Driver can be uninstalled using Windows Device Manager (Start | Settings | Control Panel | System | Device Manager | Ports (COM & LPT)), or the Driver Uninstall short-cut on the iPort Utility Pack Start menu.

Part 3

Programmer's Reference

ASCII Command Interface Definitions

Programmer's
Quick Start

Creating a custom i2cStick program is easier if you know what to expect. Follow these steps to manually control the i2cStick from your computer's keyboard and screen.

1 Install the i2cStick as directed in the "Hardware Set-Up" section of this User's Guide.

2 Use a terminal emulator program, like Windows' Hyperterminal, to start communicating with the I^2C adapter. Remember to select the correct Com Port (COM1, COM2,…) and set the communication parameters to 19200, 57600, or 115200 Baud*, 8 Data Bits, No Parity, and 1 Stop Bit. (*i2cStick internally re-maps all three baud rates to 115200 baud)

3 Enter **//[CR]** to get an i2cStick Status Report. Note: All i2cStick commands are terminated with a Carriage Return ([CR]) character. On most terminal emulators, press the Enter key.

4 Enter **/F0[CR]** (XON/XOFF) or **/F1[CR]** (RTS/CTS) to set i2cStick's communications Flow Control to match your terminal.

5 Enter **/Ixx[CR]** (xx = 02…FE even) to set i2cStick's Own I^2C Slave Address.

6 Enter **/O[CR]** to Open the i2cStick Connection. The i2cStick does not need to be connected to an I^2C Bus to open a connection.

7 Enter **/Dxx[CR]** (xx = 00…FE even) to select a Destination I^2C Slave Address

8 Enter **/Ttext[CR]** (text = ASCII or Hex-Equivalent ~00…~FF) to Master Transmit a message to the current Destination I^2C Slave device

9 Enter **/Rn[CR]** (n = 0…32767) to Master Read a message from the current Destination I^2C Slave device.

i2cStick
ASCII Command Interface

Note: [CR] = Carriage Return Code or Enter Key.
Syntax: [Select], (Optional), xx = [00..FE], n = [0..32767]

Command	Description
Ctrl/R,Ctrl/R,Ctrl/R	**i2cStick Reset** This command resets the i2cStick to its default state.
//[CR]	**Status Display** Display i2cstick status information.
/B[0|1|2][CR]	**Serial Communication Baud Rate Control** Set the serial communication baud rate (0 = 19.2K, 1 = 57.6K, 2 = 115.2K Baud). Note: For backward compatibility only. i2cStick internally re-maps all three baud rates to 115.2K.
/C[CR]	**Close I^2C Connection** Disconnect from the I^2C Bus.
/Dxx[CR]	**Set Destination I^2C Slave Address** Set the destination I^2C Slave Address for subsequent Master Transmit or Receive operations.
/E[0|1][CR]	**Echo/Prompt Control [0 = Off, 1 = On]** Enable/Disable data entry echo and prompts.
/F[0|1][CR]	**Flow Control [0 = XON/XOFF, 1 = RTS/CTS]** Select serial communication handshaking protocol.
/G[0|1][CR]	**I^2C General Call Control [0 = Disabled, 1 = Enabled]** Enables/Disables i2cStick response to I^2C Bus General Call (00) messages.
/H[0|1][CR]	**Hex Only Display Control [0 = Disabled, 1 = Enabled]** Controls display format of received message data.
/Ixx[CR]	**Set i2cStick's Own I^2C Slave Address** Sets i2cStick's own I^2C Slave Address. i2cStick will respond to I^2C Bus messages sent to this address.
/K[0|1|2|3][CR]	**I^2C Bus Clock Rate Control** Set I2C Bus Clock Rate Control (0=23, 1=86, 2=100, 3=400 KHz)
/M[CR]	**Command Menu Display** Displays i2cStick's Command Menu
/N([0|1|A|R])[CR]	**iNterrupt Signal Monitor/Control/Status** Sets Monitor/Control/Status of INT line [0 = Disable, 1 = Enable, A = Assert, R = Release, CR=Status]. For backward compatibility only. INT signal not supported.
/O[CR]	**Open I^2C Connection** Activates i2cStick as an I^2C device attached to the bus.

/(*)Rnnnn[CR]	**Master Read Message** Read the specified number of data bytes from the current Destination I^2C Slave device. * = No Stop for Repeated Start.
/Stext[CR]	**Slave Transmit Message** Write the specified data bytes to a requesting I^2C Master Receiver device.
/(*)Ttext[CR]	**Master Transmit Message** Master Transmit the specified data bytes to the current Destination I^2C Slave device. * = No Stop for Repeated Start.
/Un[CR]	**I^2C Bus Time-oUt** Set I^2C Bus Time-oUt in msec (0=Disable)
/V[CR]	**Display i2cStick Firmware Version** (Major XX.XX Minor)
/X[CR]	**eXtended Commands** (See Prompt or User's Guide)
/(*)Y[CR]	**Display Tx bYte Count** Display number of data bytes last sent to slave device. * = Also display last received Acknowledgment bit from slave device.

Synchronous Interface Events

Synchronous Events are those i2cStick interface activities initiated by the Host computer.

i2cStick Reset

Reset i2cStick to its default state.

The reset command consists of three (3) sequential Ctrl/R characters. Ctrl/R is the character code Decimal 18 and Hexadecimal 0x12. When using a terminal emulator program, you can generate a Ctrl/R by holding down the Ctrl key and pressing the R key.

Note: It is recommended that the Host computer turn off all serial port flow control before sending this command to override any flow control from the I^2C adapter that could block the transmission. Flow control should be enabled once the response is received.

Command: Ctrl/R,Ctrl/R,Ctrl/R ‘i2cStick Reset
Response. * ‘i2cStick Ready

Default Setting: None

Status Display

Display i2cStick status.

Command: //[CR] 'Status Display
Response (Typical):
i2cStick I²C Host Adapter Vxx.xx
Copyright © xxxx, Micro Computer Control Corp.
Visit our Web Site at: http://www.mcc-us.com

Serial Communications Baud Rate (115.2kHz)
Destination I²C Slave Address (xxH)
Echo/Prompt (Disabled)
Flow Control (XON/XOFF)
Hex Only Display (Enabled)
I²C Connection (Closed)
General Call (Enabled)
iPort's own Slave Address (xxH)
I2C Bus Clock Rate (100kHz)
iNterrupt Signal (Released)
I2C Bus Time-oUt (10000 msec)

Serial Communications Baud Rate
This command sets the serial communications baud rate.
(0=19.2k, 1=57.6k, 2= 115.2k)

Command: /B[0|1|2][CR] 'Set Serial Com Baud Rate
Response 1: /BC0[CR] 'Baud Change Complete
Response 2: /BC1[CR] 'Baud Change Complete
Response 3: /BC2[CR] 'Baud Change Complete
Response 3: /I89[CR] 'Invalid Command Argument

Default Setting: /B0[CR]

NOTE: The Serial Communications Baud Rate command is supported for backward compatibility only. i2cStick internally re-maps all three baud rates to 115.2K baud.

Close I²C Connection

Disconnect i2cStick from the I²C Bus.

Command: /C[CR] 'Close I²C Connection
Response: /CCC[CR] 'Close Connection Complete
Default Setting: 'Closed

Set Destination I²C Slave Address

Set the destination I²C Slave Address (Hex 0,2...FE) for all subsequent Master Transmit or Receive operations.

Command: /Dxx[CR] 'Set Destination I²C Slave Address
Response 1: * 'i2cStick Ready
Response 2: /I89[CR] 'Invalid Command Argument
Default Setting: 00

Echo/Prompt Control

This command enables or disables data entry echo and prompts used as feedback to manual operations from a computer terminal.

Command: /E[0|1][CR] 'Echo/Prompt Control [0 = Off, 1 = On]
Response: * 'i2cStick Ready
Default Setting: Off

Serial Communications Flow Control

Select the serial communication handshaking protocol to be used in communicating with the Host computer.

i2cStick implements either XON/XOFF (by default) or RTS/CTS flow control protocols. Flow control is used by the i2cStick to limit character flow to and from the Host computer to avoid overflowing internal communication buffers and lost data.

The XON/XOFF protocol inserts characters directly into the ASCII data stream. XON (Hexadecimal 0x11) is used to enable the flow of data. XOFF (Hexadecimal 0x13) is used to stop the flow of data.

The RTS/CTS protocol uses two additional wires in the cable connecting

communicating devices. The RTS wire is an output signal. It indicates that the device generating the signal has buffer space available, and can receive data. The CTS wire is an input signal. It indicates that the other device has buffer space available, and can receive more data.

In general, XON/XOFF requires a minimal three-wire connection, Ground, Transmit Data, and Receive Data. This protocol does insert control characters into the stream of data, and may not be appropriate for all Host systems. If supported, these control characters are normally automatically stripped out of the data stream by Host communication driver software, and are not visible at the application program level.

The RTS/CTS protocol requires a serial port, cabling, and Host communication driver software that supports the additional control signals.

Command: /F[0|1][CR]	Flow Control [0 = XON/XOFF, 1 = RTS/CTS]
Response: *	'i2cStick Ready
Default Setting:	XON/XOFF

I^2C General Call Control

Enables or disables i2cStick response to I^2C Bus General Call (Address x00) messages.

Command: /G[0|1][CR]	'I^2C General Call [0 = Disabled, 1 = Enabled]
Response: *	'i2cStick Ready
Default Setting:	Enabled

Hex Only Display Control

Controls Hex Only (~00...~FF) output of Master or Slave received data.

When enabled, all received I^2C message data bytes are displayed in Hex (~00…~FF) format. When disabled, received I^2C message data bytes representing ASCII printable characters (x20...x7F) are displayed as their ASCII printable character. Non-ASCII printable data bytes are always displayed in Hex (~00…~FF) form.

Command: /H[0|1][CR]	'Hex Only Display [0 = Disabled, 1 = Enabled]
Response: *	'i2cStick Ready

Default Setting: Enabled

Set i2cStick's Own I²C Slave Address

Sets i2cStick's own I²C Slave Address (Hex 2...FE). Subsequent I²C messages to this address will cause i2cStick to become an active Slave device on the bus.

Command: /Ixx[CR] 'Set i2cStick's Own I²C Slave Address
Response 1: * 'i2cStick Ready
Response 2: /I89[CR] 'Invalid Command Argument
Default Setting: 6E

I²C Bus Clock Rate Control

Set the I²C Bus master clock rate. (0=23, 1=86, 2=100, 3=400 KHz)

Command: /K[0|1|2|3][CR] 'Set i2cStick's Clock Rate
Response 1: * 'i2cStick Ready
Default Setting: /K2[CR]

The i2cStick I²C Bus master clock rate is controlled by the frequency of the oscillator used in the adapter. The oscillator frequency has been selected to give accurate RS-232 baud rates, as the RS-232 baud rate must exactly match the rate used by the host computer. Actual master I²C clock rates are close to, but not faster than, the stated rates. Slave I²C clock rates are driven by the external master device, with possible clock-stretching as required to store or retrieve message data.

Command Menu Display

Display i2cStick's command menu.

Command: /M[CR] 'Command Menu Display
Response (Typical):
i2cStick Command Menu Syntax: [Select], (Optional), xx=[00..FE], n=[1..32767]

//	Status Display
/B[0|1|2]	RS-232 Baud Rate Control (0=19.2, 1=57.6, 2=115.2KHz)
/C	Close I2C Connection
/Dxx	Set Destination I2C Slave Address
/E[0|1]	Echo/Prompt Control (0=Disable, 1=Enable)

/F[0|1] Flow Control (0=XON/XOFF, 1=RTS/CTS)
/G[0|1] General Call Control (0=Disable, 1=Enable)
/H[0|1] Hex Only Display Control (0=Disable, 1=Enable)
/Ixx Set i2cStick's Own I2C Slave Address
/K[0|1|2|3] I2C Bus Clock Rate Control (0=23, 1=86, 2=100, 3=400 KHz)
/M Menu Display
/N([0|1|A|R]) iNterrupt Signal Monitor/Control/Status
(0=Disable, 1=Enable / A=Assert, R=Release / <CR>=Status)
/O Open I2C Connection
/(*)Rn Master Rx Message *=No Stop
/S(text) Slave Tx Message
/(*)T(text) Master Tx Message *=No Stop
/Un Set I2C Bus Time-oUt in msec 0=Disable)
/V Display Firmware Version (Major XX.XX Minor)
/X[...]... Extended Cmds (See Prompt or User's Guide)
/(*)Y Display Tx bYte Count *= with last received Ack bit

Interrupt Signal Control/Status

The i2cStick does not support the INT signal. The following commands and responses are provided for backward compatibility only.

Control
Command: /N0[CR] Disable Monitor
/N1[CR] Enable Monitor
/NA[CR] Assert INT Signal
/NR[CR] Release INT Signal

Status
Command: /N[CR] Status
Response: /NSA INT Asserted
/NSR INT Released

Response: * 'i2cStick Ready
Default Setting: /N0, /NR

Open I^2C Connection

Activates i2cStick as an active device on the I^2C Bus.

Command: /O[CR] 'Open I^2C Connection

Response: /OCC[CR] 'Open Connection Complete
Default Setting: Closed

Master Read Message

This command causes i2cStick to read the specified number of data bytes from the currently selected Destination I^2C Slave Address with or without generating an I^2C Stop condition after the last byte is received.

Enter Byte Count (Decimal 0...32767) then Press Enter, or ESCape to Cancel. A Byte Count of Zero (0) represents a Variable Length message, where the first byte read from the I^2C Slave device indicates the number of additional trailing bytes that are available to read. The i2cStick automatically reads the first byte, then the additional bytes as specified by the first byte. All message bytes including the Length byte are returned to the Host computer.

The received text is a representation of the data bytes within the Master Receive message. The format of this data is controlled by the current setting of the Hex Only Display Control.

If the slave device acknowledges its I^2C Slave Address, the specified number of bytes are read. The i2cStick acknowledges all bytes read except the last. If not disabled, the message is then terminated with an I^2C Stop condition.

Sending Master Receive messages with No Stop allows the Master to retain exclusive control of the I^2C Bus until it finally sends a Stop. During this time, the Master can send additional (Repeated Start) Master Transmit or Master Receive messages to the same or other I^2C Slave devices.

Command: /(*)Rnnnn[CR] 'Master Read Message (* = No Stop)

Response 1: /MRCtext[CR] 'Master Read Complete
Response 2: /SNA[CR] 'Slave Not Acknowledging
Response 3: /I81[CR] 'i2cStick is Busy, Command Ignored
Response 4: /I83[CR] ' I^2C Arbitration Loss Detected
Response 5: /I88[CR] 'Connection Not Open
Response 6: /I89[CR] 'Invalid Command Argument
Default Setting: None

Slave Transmit Message

This command should be issued to i2cStick in response to a Slave Transmit Request (/STR). This command causes i2cStick to write the specified data bytes to the requesting I^2C Master Receiver device.

Enter Message Bytes (1 or more Printable ASCII or Hex-equivalent ~00..~FF), then Press Enter, or ESCape to Cancel.

Note 1: Upon receiving a Slave Transmit request from a Master Receiver device on the I^2C Bus, the i2cStick outputs a Slave Transmit Request to its Host device, and initiates an I^2C Clock Stretch (SCL Low) until a Slave Transmit command is received from the Host computer. While clock stretching, no other messages can be transmitted on the I^2C Bus.

Note 2: The tilde (~) character and the Carriage Return (CR) characters are used as special marker characters within all i2cStick transmitted text messages. These characters may not be used within the text of a message, but must be replaced by the following "Hex equivalent" characters:

Tilde replaced by "~7E"
Carriage Return replaced by "~0D"

i2cStick automatically translates "Hex equivalent" characters to their single-byte value for transmission across the I^2C Bus.

All entered data bytes are transmitted to the requesting Master Receiver device. Slave Transmit stops upon receiving the first negative acknowledgment (Nack) from the Master Receiver.

Command: /Stext[CR]	'Slave Transmit Message
Response 1: /STC[CR]	'Slave Transmit Complete
Response 2: /I88[CR]	'Connection Not Open
Response 3: /I8A[CR]	‘Slave Transmit Request Not Active, Cmd Ignored
Default Setting:	None

Examples:

/Sabcd1234[CR] ‘ASCII Printable characters "abcd1234"
/S~00~01~02[CR] ‘Binary data bytes 00, 01,02
/Sab~7Ecd[CR] ‘Tilde embedded in ASCII Printable characters
/S12~0D24[CR] ‘Carriage Return embedded in ASCII Printable characters

Master Transmit Message

Write the specified data bytes to the currently selected Destination I^2C Slave Address with or without generating an I^2C Stop condition after the last byte is transmitted.

Enter Message Bytes (0 or more Printable ASCII or Hex-equivalent ~00..~FF), then Press Enter, ESCape to Cancel.

Note: The tilde (~) character and the Carriage Return (CR) characters are used as special marker characters within all i2cStick transmitted text messages. These characters may not be used within the text of a message, but must be replaced by the following "Hex-equivalent" characters:

Tilde replaced by "~7E"
Carriage Return replaced by "~0D"

i2cStick automatically translates "Hex equivalent" characters to their single-byte value for transmission across the I^2C Bus.

All entered data bytes are transmitted to the Destination I^2C Slave Receiver device. Master Transmit stops upon receiving the first negative acknowledgment (Nack) from the Slave Receiver. If not disabled, the message is then terminated with an I^2C Stop condition.

Sending Master Transmit messages with No Stop allows the Master to retain exclusive control of the I^2C Bus until it finally sends a Stop. During this time, the Master can send additional (Repeated Start) Master Transmit or Master Receive messages to the same or other I^2C Slave devices.

Note: See the Display Tx bYte Count command (/Y) for additional information on the last completed Master Transmit message.

Command: /(*)Ttext[CR] 'Master Transmit Message (* = No Stop)
Response 1: /MTC[CR] 'Master Transmit Complete
Response 2: /SNA[CR] 'Slave Not Acknowledging
Response 3: /I81[CR] 'i2cStick is Busy, Command Ignored
Response 4: /I83[CR] ' I^2C Arbitration Loss Detected
Response 5: /I88[CR] 'Connection Not Open
Default Setting: None

Examples:

/Tabcd1234[CR] ‘ASCII Printable characters "abcd1234"
/T~00~01~02[CR] ‘Binary data bytes 00, 01,02
/*T~00~01~02[CR] ‘Binary data bytes 00, 01,02 with No Stop
/Tab~7Ecd[CR] ‘Tilde embedded in ASCII Printable characters
/T12~0D24[CR] ‘Carriage Return embedded in ASCII Printable characters

Set I^2C Bus Time-oUt in msec

Set bus time-out in milliseconds.

The i2cStick reports a bus time-out if no inter-byte bus activity for the specified time occurs within an I^2C Bus message.

Command: /Unnnnn[CR] 'I^{2}C Bus time-oUt (nnnn = 0 (disable)...32000 msec)
Response: * 'i2cStick Ready
Default Setting: 10000 msec (10 seconds)

Display Firmware Version

Display i2cStick firmware version

Command: /V[CR] 'Firmware Version
Response: /VCCXX.XX[CR] '(Major XX.XX Minor)

eXtended Commands

The eXtended commands are used to generate "out-of-spec" signaling. eXtended commands cannot use the adapter’s I^2C hardware to control the SCL and SDA lines, as the I^2C hardware only generates I^2C compatible signals. The eXtended

commands use firmware to "bit-bang" the SCL and SDA lines. This firmware cannot operate as fast as the hardware, and it can be interrupted at any time by adapter internal interrupts. The eXtended commands run directly off the command characters as they are received on the serial link. Speed of execution of eXtended commands is controlled by the serial link communication rate, the execution speed of the firmware, delays caused by execution interruptions that may occur while a command is executing, and I^2C Bus clock-stretching by external slave devices.

The following commands manipulate the I^2C Clock (SCL) and data (SDA) lines.

Command: /X[S|~xx|R|r|P|0|1|?|D|d|C|c|L|A| |"]..., then Press Enter or ESCape
Enter /X followed by zero or more sub-commands, the [CR]

Response: /XCC(see commands below)[CR]

High Level Sub-Commands:

S = Send Start
~xx = Send Byte (xx = 00...FF)(response = A or N)
R = Read Byte with Ack (response = ~xx)
r = Read Byte with Nak (response = ~xx)
P = Send Stop

Mid Level Sub-Commands:

0 = Send 0 Bit
1 = Send 1 Bit
? = Read Bit (response = 0 or 1)

Low Level Sub-Commands:

D = Set SDA High
d = Set SDA Low
C = Set SCL High
c = Set SCL Low
L = Read SCL (response = 0 or 1)
A = Read SDA (response = 0 or 1)

Miscellaneous Sub-Commands:

space = no action
"comment" = no action

Examples:

Master transmit three bytes to slave address 0x4e using high level, mid level, and low level sub-commands.

High Level Command:/X S ~4e ~01 ~02 ~03 P [CR]
High Level Response: /XCCAAAA[CR]

Mid Level Command:/X S 01001110 ? 00000001 ? 00000010 ? 00000011 ? P [CR]
Mid Level Response: /XCC0000[CR]

Low Level Command:/X dc dCcDCcdCcdCcDCcDCcDCcdCc DCAc dCcdCcdCcdCcdCcdCcdCcDCc DCAcdCcdCcdCcdCcdCcdCcDCcdCc DCAc dCcdCcdCcdCcdCcdCcdCcDCc DCAc dCD[CR]
Low Level Response: /XCC0000[CR]

Master read three bytes from slave address 0x4F. First two bytes are acknowledged by master.

Command: /X S ~4f Rrr P [CR]
Response: /XCCA~xx~xx~xx[CR] ‘(xx = 00...FF)

Master transmit a Write WCR command to a Xicor X9241 at slave address 0x50. WCR data is 0x00.

Command: /X S ~50 ~a0 ~00 P [CR]
Response: /XCCAAA[CR]

Master transmit a Write WCR command to a Xicor X9241 at slave address 0x50. WCR data is 0x3f.

Command: /X S ~50 ~a0 ~3f P [CR]
Response: /XCCAAA[CR]

Issue a Read WCR command to a Xicor X9241 at slave address 0x50.

Command: /X S ~50 ~90 ~R P [CR]
Response: /XCCAA~xx[CR] ‘(xx = 00...FF)

Issue an Increment Wiper command to a Xicor X9241 at slave address 0x50.

Command: /X S ~50 ~20 1 P [CR]
Response: /XCCAA[CR]

Issue a Decrement Wiper command to a Xicor X9241 at slave address 0x50.

Command: /X S ~50 ~20 0 P [CR]
Response: /XCCAA[CR]

Display Tx bYte Count

Returns the number of bytes received by the slave device in the last master transmit message, with an option to received the state of the last received Acknowledgment bit.

Note: The byte count and last received acknowledgment bit state can be used for SMBus Packet Error Control (PEC) error detection. See SMBus v1.1+ specifications for details.

Command: /(*)Y[CR] ‘Tx bYte Count (* = with last received Ack bit)
Response: /TBCn[CR] 'n =00000...32767
Response: /TBCn(A|N)[CR] 'n =00000...32767, A = ACK, N = NACK

Asynchronous Interface Events

Asynchronous Events are those i2cStick interface activities initiated by the i2cStick I^2C Host Adapter in response to activities on the I^2C Bus.

Slave Transmit Request

This event is caused by the reception of an I^2C Bus Slave Transmit message directed at the current i2cStick's own Slave address.

Prompt: /STR[CR] 'Slave Transmit Request
Command: /Stext[CR] 'Slave Transmit Text

The normal Host computer response is to send a Slave Transmit (/Stext[CR]) command.

Note: Upon receiving a Slave Transmit request from a Master Receiver device on the I^2C Bus, i2cStick outputs a Slave Transmit Request to its Host device, and initiates an I^2C Clock Stretch (SCL Low) until a Slave Transmit Text command is received from the Host computer. While clock stretching, no other messages can be transmitted on the I^2C Bus.

Slave Receive Complete

This event is caused by the reception of an I^2C Bus Slave Receive message directed at the current i2cStick's own Slave address.

The received text is a representation of the data bytes within the Slave Receive message. The format of this data is controlled by the current setting of the Hex Only Display Control.

Prompt: /SRCtext[CR] 'Slave Receive Complete
Command: None Required

General Call Receive Complete

This event is caused by the reception of an I^2C Bus Slave Receive message directed at the I^2C General Call Address (00), when i2cStick's General Call recognition is enabled.

The received text is a representation of the data bytes within the Slave Receive message. The format of this data is controlled by the current setting of the Hex Only Display Control.

Prompt: /GRCtext[CR] ‘General Call Receive Complete
Command: None Required

i2cStick Ready
Prompt: * ‘i2cStick Ready

Cause: i2cStick is ready for the next Host command.

Slave Not Acknowledging

Prompt: /SNA[CR] ‘Slave Not Acknowledging

Cause: There is no response (I^2C Slave Address Acknowledgment) during a Master Transmit or Receive operation from an I^2C Slave device at the current Destination I^2C Address.

i2cStick Busy

Prompt: /I81[CR] ‘i2cStick Busy

Cause: The host computer attempted a Master operation while i2cStick was busy. The host computer should wait for any previously issued command to complete, process any pending slave events, and retry the last command.

I^2C Bus Arbitration Loss

Prompt: /I83[CR] ‘I^2C Arbitration Loss Detected

Cause: i2cStick lost I^2C Bus Arbitration to another bus master device while Master Transmitting or Master Receiving an I^2C message. Host should process any active slave events and repeat the last command.

I^2C Bus Error Detected

Prompt: /I84[CR] ‘I^2C Bus Error Detected

Cause: i2cStick has detected an error condition on the I^2C Bus. The host computer should retry the last command or issue an i2cStick Reset command.

I^2C Bus Time-out Detected

Prompt: /I85[CR] ‘I^2C Bus Time-out Detected

Cause: i2cStick issues this response when it detects a byte transfer delay greater than the specified I^2C Bus Time-oUt period. No corrective action is taken by the i2cStick regarding I^2C Bus activity. No host computer response is required, but this event can be used to detect possible bus problems.

i2cStick Connection Closed

Prompt: /I88[CR] ‘i2cStick Connection is Closed.

Cause: The host computer is attempting to perform an I^2C Bus message operation while the i2cStick Connection is Closed. The host computer should issue an Open I^2C Connection command before attempting to perform I^2C Bus message operations.

Invalid Command Argument

Prompt: /I89[CR] ‘Invalid Command Argument Detected

Cause: This event normally indicates the value of a host command argument was out of range. The host should reissue command with correct arguments.

Slave Transmit Request Not Active

Prompt: /I8A[CR] ‘Slave Transmit Request Not Active

Cause: This event indicates the host attempted to issue a Slave Transmit Text command when no Slave Transmit Request was present.

Invalid i2cStick Command

Prompt: /I8F[CR] ‘Invalid i2cStick Command

Cause: This event normally indicates that an invalid command was issued by the

host. The host should reissue the correct command.

i2cStick Receive Buffer Overflow

Prompt: /I90[CR] ‘i2cStick Serial Receive Buffer Overflow

Cause: This event normally indicates that data sent to the i2cStick via the serial port has been lost. Check the host computer’s Serial Port Flow Control (XON/XOFF, or Hardware) to make sure it matches current i2cStick Flow Control. Also, check if the host computer’s FIFO buffers in its 16550 UART are enabled. If so, reduce or disable Transmit Data Buffering. On Windows-based host computers, see the Device Manager, COM port, Advanced Settings. You may need to power down the host computer for any FIFO change to take effect.

Example Code

The following examples are written in MS Visual Basic V3 for Windows using the serial communications control (MSCOMM.VBX). It can be used as a guide in implementing i2cStick interface programs in other programming languages and operating environments.

Note: Sample code is also available online at: www.mcc-us.com

i2cStick Reset

```
Comm1.Output = Chr$(18)        'Ctrl/R
Comm1.Output = Chr$(18)        'Ctrl/R
Comm1.Output = Chr$(18)        'Ctrl/R
```

i2cStick Initialization

```
Comm1.Output = "/f0"        'Set i2cStick XON/XOFF Flow Control
Comm1.Output = Chr$(13)

Comm1.Output = "/i70"         'Set i2cStick's Own Slave Address
Comm1.Output = Chr$(13)

Comm1.Output = "/d4e"         'Set Destination Slave Address
Comm1.Output = Chr$(13)

Comm1.Output = "/o"           'Open I²C Connection
Comm1.Output = Chr$(13)
```

Master Transmit Message

```
Comm1.Output = "/T~00~01" 'Send Master Tx Command
Comm1.Output = Chr$(13)     'Terminate Command
```

Master Receive Message

```
Comm1.Output = "/R10"         'Send Master Rx Command
Comm1.Output = Chr$(13)     'Terminate Command
```

Communication Event Processing

```
Static Sub Comm1_OnComm ()
Static LineBuf$

While Comm1.InBufferCount
   Msg$ = Comm1.Input ' Get Comm input character
   CharIn$ = Msg$

   If Msg$ = Chr$(13) Then Msg$ = ""    ' Remove CR
   If Msg$ = Chr$(10) Then Msg$ = ""    ' Remove LF
   If Msg$ = "*" Then                ' if i2cStick Ready
      Msg$ = "****"                  ‘ Substitute Token
      CharIn$ = Chr$(13)             ‘ Terminate Line
   End If
   LineBuf$ = LineBuf$ + Msg$         'Add new text to line buffer

   If CharIn$ = Chr$(13) Then         ' if Carriage Return detected
      iPortResp$ = Left$(LineBuf$, 4)     'Isolate Response Code

      ' Test for i2cStick Synchronous Interface Events

      If (StrComp(iPortResp$, "/OCC") = 0) Then
         ' Open Connection Complete Processing
         TextBox.Text = "/OCC Open Connection Complete"

      ElseIf (StrComp(iPortResp$, "/MTC") = 0) Then
         ' Master Transmit Complete Processing
         TextBox.Text = "/MTC Master Tx Complete"

      ElseIf (StrComp(iPortResp$, "/MRC") = 0) Then
         ' Master Rx Complete Processing
         TextBox.Text = LineBuf$   'Update Display

      ElseIf (StrComp(iPortResp$, "/STC") = 0) Then
         ' Slave Tx Complete Processing
         TextBox.Text = "/STC Slave Tx Complete"

      ElseIf (StrComp(iPortResp$, "/CCC") = 0) Then
         ' Close Connection Complete Processing
         TextBox.Text = "/CCC Close Connection Complete "
```

```
ElseIf (StrComp(iPortResp$, "/BC0") = 0) Then
    ' i2cStick Baud Change 0 {19.2K}
    TextBox.Text = "i2cStick Baud Change 0 {19.2K} "

ElseIf (StrComp(iPortResp$, "/BC1") = 0) Then
    ' i2cStick Baud Change 1 {57.6K}
    TextBox.Text = "i2cStick Baud Change 1 {57.6K} "

ElseIf (StrComp(iPortResp$, "/BC2") = 0) Then
    ' i2cStick Baud Change 2 {115.2K}
    TextBox.Text = "i2cStick Baud Change 0 {115.2K} "

' Test for i2cStick Asynchronous Interface Events

ElseIf (StrComp(iPortResp$, "/SRC") = 0) Then
    ' Slave Rx Complete Processing
    TextBox.Text = LineBuf$       'Update Display

ElseIf (StrComp(iPortResp$, "/GRC") = 0) Then
    ' General Call Rx Complete Processing
    TextBox.Text = LineBuf$       'Update Display

ElseIf (StrComp(iPortResp$, "/STR") = 0) Then
    ' Slave Tx Request Processing
    Comm1.Output = "/S~00~01" 'Send Slave Tx Msg
    Comm1.Output = Chr$(13)     'Terminate Command
    TextBox.Text = LineBuf$       'Update Display

' Test for i2cStick Response Messages

ElseIf (StrComp(iPortResp$, "****") = 0) Then
    TextBox.Text = "* i2cStick Ready" 'Update Display

ElseIf (StrComp(iPortResp$, "/SNA") = 0) Then
    TextBox.Text = "/SNA Slave Not Acknowledging"

ElseIf (StrComp(iPortResp$, "/I81") = 0) Then
    TextBox.Text = "/I81 i2cStick Busy" 'Update Display

ElseIf (StrComp(iPortResp$, "/I83") = 0) Then
```

```
            TextBox.Text = "/I83 Arbitration Loss" 'Update Display

        ElseIf (StrComp(iPortResp$, "/I84") = 0) Then
            TextBox.Text = "/I84 I2C Bus Error Detected"

        ElseIf (StrComp(iPortResp$, "/I85") = 0) Then
            TextBox.Text = "/I85 I2C Bus Time-out Detected"

        ElseIf (StrComp(iPortResp$, "/I88") = 0) Then
            TextBox.Text = "/I88 i2cStick Connection Closed"

        ElseIf (StrComp(iPortResp$, "/I89") = 0) Then
            TextBox.Text = "/I89 Invalid Command Argument"

        ElseIf (StrComp(iPortResp$, "/I8A") = 0) Then
            TextBox.Text = "/I8A Slave Tx Request Not Active"

        ElseIf (StrComp(iPortResp$, "/I8F") = 0) Then
            TextBox.Text = "/I8F Invalid i2cStick Command"

        ElseIf (StrComp(iPortResp$, "/I90 = 0) Then
            TextBox.Text = "/I90 i2cStick Rx Buffer Overflow”

        Else
            TextBox.Text = LineBuf$   'Other Update Display
        End If
    LineBuf$ = ""
    End If
Wend
End Sub
```

i2cStick Revision Report

This section defines revisions and changes made to the i2cStick interface:

Revision: 1.00

1 Initial Release

i2cStick UG Update 03-APR-2012

- Correct I^2C Bus Mini Clip lead cable included, Mini Interface cable available.
- Correct Appendix A Molex part number and cable length.
- Add I^2C Bus Mini CAB (#I2CMCAB) cable.

Additional Information

For additional information on the I^2C Bus, please refer to the following:

“What is I^2C?”
www.mcc-us.com/I2CBusTechnicalOverview.pdf

“Frequently Asked Questions (FAQ)”
www.mcc-us.com/faq.htm

"The I^2C and How to Use It"
www.mcc-us.com/i2chowto.htm

Appendix A - I²C Connector Information

I²C Bus Interface Connector and Plug Information

The i2cStick uses the following 1x5 2.54 mm (.100") pitch, 0.64 mm (.025") square pin, header and plug assemblies for the I²C Bus interface.

I²C Header

Molex C-Grid® SL™ 70553 Header

Molex Part # 70553-0004

I²C Plug Housing

Molex C-Grid® SL™ 70066 Crimp Housing

Molex Part # 50-57-9405

Molex C-Grid® SL™ 70058 Crimp Terminal

Molex Part # 16-02-0102

The following I²C Cables are available from MCC

MCC Part #	I2CMIC	I²C Mini Interface Cable 0.6 m (2')
MCC Part #	I2CMCL	I²C Mini Clip Lead Cable 0.3 m (1')
MCC Part #	I2CMCAB	I²C Mini CAB Cable 0.6 m (2')

Information

FCC Statement

DECLARATION OF CONFORMITY WITH FCC RULES FOR ELECTROMAGNETIC COMPATIBILITY

We, Micro Computer Control Corporation, of 83 Princeton Avenue #1D / PO Box 275, Hopewell, New Jersey 08525 USA, declare under our sole responsibility that the product:

i2cStick (#MIIC-207)

to which this declaration relates:
Complies with Part 15 of the FCC Rules. Operation is subject to the following two conditions; (1) this device may not cause harmful interference, and (2) this device must accept any interference received, including interference that may cause undesired operation.

Test Laboratory Information:

MET Laboratories, Inc.
Test Report Number: EMC29574-FCC
Test Report Date: November 4, 2010
Technical file held by: Micro Computer Control Corporation, 83 Princeton Avenue #1D / PO Box 275, Hopewell, New Jersey 08525 USA, or its applicable authorized distributor or representative.

CE Declaration of Conformity

We, Micro Computer Control Corporation, of 83 Princeton Avenue #1D / PO Box 275, Hopewell, New Jersey 08525 USA, declare under our sole responsibility that the **i2cStick (#MIIC-207)**, to which this declaration relates, is in conformity with General Emissions Standard EN55022 (CISPR22): 2006 Class A, and General Immunity Standard EN 55024: 1998 + A1:2001 + A2: 2003.

Test Laboratory Information:

MET Laboratories, Inc.
Test Report Number: EMC29574-EURO
Test Report Date: November 4, 2010
Technical file held by: Micro Computer Control Corporation, 83 Princeton Avenue #1D / PO Box 275, Hopewell, New Jersey 08525 USA, or its applicable authorized distributor or representative.