User’s Guide

USB to I²C Host Adapter
ASCII Fast Mode
with iPort Utility Pack Software

www.mcc-us.com
Introduction

The MCC iPort/USB™ USB to I²C Host Adapter with ASCII Fast Mode (#MIIC-204) 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.

This user’s guide describes the installation and operation of the iPort/USB host adapter, Virtual Communication Port (VCP) driver, 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 are 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.

05-JAN-11
# Table of Contents

## Part 1 - iPort/USB USB to I²C Bus Host Adapter

1. Overview .............................................................................................................. 2
   - iPort/USB USB to I²C Bus Host Adapter ........................................................ 2
   - iPort/USB Virtual Communications Port (VCP) .............................................. 2
   - iPort Utility Pack Software ............................................................................ 3
   - iPort/USB Programmer’s Reference .............................................................. 3
   - Packing Slip ..................................................................................................... 3
   - System Requirements .................................................................................... 3

2. Interconnects ....................................................................................................... 3
   - USB Connector ................................................................................................ 4
   - Virtual Communications Port (VCP) ............................................................... 4
   - +5VDC Power Jack .......................................................................................... 4
   - I²C Interface Connector .................................................................................. 5
   - Interrupt (/INT) Signal Connector ................................................................ 6

3. Hardware Configuration ....................................................................................... 6
   - Pull-up Resistors .............................................................................................. 6
   - Connecting to a 3.3 Volt Target System ......................................................... 7
   - Connecting to an SMBus Target System ......................................................... 7

4. ESD (Electrostatic Discharge) Precautions ......................................................... 8
   - Host Computer Grounding ............................................................................ 8
   - Grounding Solutions ....................................................................................... 8

5. Driver Software Set-Up ....................................................................................... 9
   - Driver Install ................................................................................................... 9
   - Driver Update .................................................................................................. 10
   - Driver Uninstall ............................................................................................. 10

6. Hardware Set-Up ................................................................................................ 10
   - USB Connection ............................................................................................. 10
   - I²C Bus Connection ....................................................................................... 10
   - Optional +5VDC Power Connection .............................................................. 11

## Part 2 - iPort Utility Pack for Windows ............................................................... 13
<table>
<thead>
<tr>
<th>Section</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>1 iPort Utility Pack for Windows</td>
<td>15</td>
</tr>
<tr>
<td>iPort Message Center</td>
<td>15</td>
</tr>
<tr>
<td>iPort Message Manager</td>
<td>16</td>
</tr>
<tr>
<td>2 System Requirements</td>
<td>17</td>
</tr>
<tr>
<td>3 iPort Utility Pack Installation</td>
<td>17</td>
</tr>
<tr>
<td>Installing from CD</td>
<td>17</td>
</tr>
<tr>
<td>Installing from the Web</td>
<td>17</td>
</tr>
<tr>
<td>4 iPort Message Center</td>
<td>18</td>
</tr>
<tr>
<td>Message Center Operations</td>
<td>19</td>
</tr>
<tr>
<td>Starting the Message Center</td>
<td>19</td>
</tr>
<tr>
<td>Selecting the Adapter</td>
<td>19</td>
</tr>
<tr>
<td>Select the Communications Port</td>
<td>20</td>
</tr>
<tr>
<td>Options Menu</td>
<td>20</td>
</tr>
<tr>
<td>Establish Adapter Communications Link</td>
<td>20</td>
</tr>
<tr>
<td>Entering or Editing I²C Messages</td>
<td>20</td>
</tr>
<tr>
<td>Set I²C Address</td>
<td>21</td>
</tr>
<tr>
<td>Set Message Read/Write Direction</td>
<td>21</td>
</tr>
<tr>
<td>Specify Repeated Start Messages</td>
<td>21</td>
</tr>
<tr>
<td>Set Time Delay</td>
<td>22</td>
</tr>
<tr>
<td>Specify Write Data or Read Byte Count</td>
<td>22</td>
</tr>
<tr>
<td>Inserting and Deleting Messages</td>
<td>23</td>
</tr>
<tr>
<td>Saving or Loading Message Lists</td>
<td>23</td>
</tr>
<tr>
<td>Send the Message List</td>
<td>23</td>
</tr>
<tr>
<td>Special Event Handling</td>
<td>23</td>
</tr>
<tr>
<td>Slave Not Acknowledging</td>
<td>24</td>
</tr>
<tr>
<td>Command Line Arguments</td>
<td>25</td>
</tr>
<tr>
<td>Set Adapter Type</td>
<td>26</td>
</tr>
<tr>
<td>Set RS-232 Communication Port</td>
<td>26</td>
</tr>
<tr>
<td>Set RS-232 Baud Rate</td>
<td>26</td>
</tr>
<tr>
<td>Set I²C Bus Clock Rate</td>
<td>26</td>
</tr>
<tr>
<td>Enable /INT Signal Monitor</td>
<td>27</td>
</tr>
<tr>
<td>Stop On Busy</td>
<td>27</td>
</tr>
<tr>
<td>Stop On Arbitration Loss</td>
<td>27</td>
</tr>
<tr>
<td>Stop On Slave Negative Acknowledgment</td>
<td>27</td>
</tr>
<tr>
<td>Beep On Busy</td>
<td>27</td>
</tr>
<tr>
<td>Beep On Arbitration Loss</td>
<td>28</td>
</tr>
<tr>
<td>Beep On Slave Negative Acknowledgment</td>
<td>28</td>
</tr>
<tr>
<td>Beep On /INT Assert</td>
<td>28</td>
</tr>
</tbody>
</table>
Quick Start ................................................................. 42

ASCII Command Interface ........................................... 43
  Synchronous Interface Events .................................... 44
    iPort/USB Reset .................................................... 44
    Status Display .................................................... 45
    Serial Communications Baud Rate ............................ 45
    Close I²C Connection ........................................... 45
    Set Destination I²C Slave Address ............................ 46
    Echo/Prompt Control ............................................ 46
    Serial Communications Flow Control ....................... 46
    I²C General Call Control ...................................... 47
    Hex Only Display Control .................................... 47
    Set iPort/USB’s Own I²C Slave Address .................... 48
    I²C Bus Clock Rate Control .................................. 48
    Command Menu Display ....................................... 48
    Interrupt Signal Control/Status .......................... 49
    Open I²C Connection .......................................... 49
    Master Read Message ......................................... 50
    Slave Transmit Message ...................................... 51
    Master Transmit Message .................................... 52
    Set I²C Bus Time-oOut in msec ............................... 53
    Display Firmware Version ................................... 53
    eXtended Commands .......................................... 53
    Display Tx bYte Count ....................................... 56
  Asynchronous Interface Events ................................ 57
    Slave Transmit Request ...................................... 57
    Slave Receive Complete ..................................... 57
    General Call Receive Complete ............................ 57
    iPort/USB Ready ................................................. 58
    Slave Not Acknowledging .................................... 58
    iPort/USB Busy .................................................. 58
    I²C Bus Arbitration Loss .................................... 58
    I²C Bus Error Detected ...................................... 58
    I²C Bus Time-out Detected .................................. 59
    iPort/USB Connection Closed ............................... 59
    Invalid Command Argument .................................. 59
    Slave Transmit Request Not Active ....................... 59
Part 1

USB to I²C Bus Host Adapter

with

ASCII Fast Mode

User’s Guide

Model: MIIC-204
1 Overview

The MCC iPort/USB USB to I²C Bus Host Adapter with ASCII Fast Mode (#MIIC-204) allows any Windows 2000, XP, Vista, or 7 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.

iPort/USB Product Features:

- Add an I²C port to ANY Windows 2000, XP, Vista(x86/x64), 7(x86/x64) PC.
- Compatible with USB 2.0 Specifications.
- Works with USB and High-Speed systems, peripherals, and cables.
- Supports I²C Bus Master and Slave, Transmit and Receive operations.
- Supports I²C /INT signal generation and detection.
- Compatible with I²C bit rates up to 400 KHz.
- Compatible with 3.3V to 5V I²C Bus voltages.
- Includes switch-enabled internal I²C Pull-Up resistors.
- Based on a High Performance I²C Bus Co-Processor.
- Optimized for High Bus Throughput, and Low Inter-byte overhead.
- Includes our free I²C Message Center and Message Manager Windows Applications. Get on the I²C Bus in Seconds.
- Includes our easy to use Virtual Communications Port (VCP) driver.
- Simple ASCII Text Commands make building custom I²C applications quick and easy. Programmer’s Reference and Sample Programs included.
- Fully Compatible with iPort/AI, iPort/AFM, and i2cStick applications. No software changes required.

The iPort/USB package includes the following items:

1.1 iPort/USB USB to I²C Bus Host Adapter

The iPort/USB adapter is a bus-powered USB device that plugs into a host computer’s USB port or self-powered USB hub and generates I²C Bus signals.

1.2 iPort/USB Virtual Communications Port (VCP)

The iPort/USB Virtual Communications Port (VCP) driver creates a virtual serial port within the iPort/USB 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. The iPort/USB VCP is available to the host computer whenever the iPort/USB is attached to the host computer.

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 iPort/USB Programmer’s Reference

This section of the iPort/USB 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:

- iPort/USB USB to I²C Bus Host Adapter (#MIIC-204).
- USB Interface Cable.
- I²C Clip-Lead Cable.
- I²C Interface Cable.
- I²C Interrupt (/INT) Signal Clip Lead Wire.
- iPort Utility Pack for Windows CD.
- Global Power Supply (Optional supplement power for target system).

1.6 System Requirements

a. A host computer with one free USB port or self-powered USB hub.
b. Windows 2000 (x86), XP, Vista, 7 (x86/x64).

2 Interconnects

The iPort/USB includes four interconnections:
2.1 USB Connector

The USB connector provides connection from the $I^2C$ adapter to a USB port on the host computer or self-powered USB hub. The iPort/USB operates as a high-power (>100 mA) bus-powered USB device, with limited target system power. Additional target system power can be provided via an optional external power supply.

2.1.1 Virtual Communications Port (VCP)

The iPort/USB 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 iPort/USB 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 iPort/USB via the standard Windows Communications Application Program Interface (API).

Communication signals included in the VCP interface include:

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

2.2 +5VDC Power Jack

The iPort/USB is a bus-powered USB device. When deriving power from the Universal Serial Bus only, a limited amount of power is available to a target system via +5V wire in the $I^2C$ Interface Connector. Additional power for a target system can be provided via the +5VDC Power Jack. When connected to the +5VDC Power Jack, an optional external power supply will remove all $I^2C$ Bus target system power load from the USB and replace it with power from the external power...
supply. See the “Hardware Setup” section for additional details.

2.3 I²C Interface Connector

The I²C adapter includes a four wire, positive locking, modular receptacle connector (see Appendix A) for interfacing to an external I²C Bus. Lines provided include I²C Clock (SCL), I²C Data (SDA), Ground, and +5V.

Minimum wiring for I²C Bus communications include I²C Bus Clock, Data, and Ground.

Connect the +5V (V) wire to the target system to power the target system from either:

1. USB derived power. Maximum available target system power is 100 mA at +5V.

2. A +5VDC Power Jack connected optional external +5V regulated power supply. Maximum available target system power is 150 mA at +5V.

An I²C Clip-Lead cable is provided to connect the I²C adapter to a target system. Each clip-lead is clearly marked (C=SCL, V=+5VDC, D=SDA, G=Ground).

An I²C Interface Cable (White=SCL, Red=+5VDC, Green=SDA, Black=Ground) is also provided to connect the I²C adapter to an external I²C Bus. Since there is no
standard I^2C Bus connector, you may want to cut off one end of the cable and add a connector compatible with your target system. Additional I^2C Interface Cables, Clip-Lead Cables, and modular connectors are available from MCC (see Appendix A).

2.1 Interrupt (/INT) Signal Connector

The Interrupt (/INT) signal is an extra pin found on some I^2C devices. The /INT signal allows a slave-only device to attract the attention of a bus master device by lowering the /INT signal to a logical low voltage level.

The iPort/USB provides an open-drain /INT signal which can be connected to a corresponding pin on a master or slave device. The /INT signal allows the iPort/USB to participate in interrupt signaling between master or slave devices.

An interrupt output (/INT=low) is generated upon receiving an iNterrupt Assert command from the host computer. The interrupt signal is released (/INT = high) when the iPort/USB is addressed as an I^2C Bus slave device, or an iNterrupt Release command is received from the host computer.

Interrupt monitoring is enabled upon receiving an Enable /INT Monitor command from the host computer. Interrupt monitoring causes the iPort/USB to monitor the /INT signal level, and send a notification to the host computer when the /INT signal changes state.

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 slide switch used to enable or disable internal +5VDC I^2C Bus 1.8K ohm pull-up resistors attached to the SCL, SDA, and /INT 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 to a 3.3 Volt Target System
If you are connecting the I^2^C adapter to a 3.3 volt target system, you should follow these steps BEFORE applying power:

• Shut off the iPort/USB 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^2^C Bus connector +5V wire from the target system. The iPort/USB will be powered from the USB, and the target system will be powered by its own 3.3V power supply.

The I^2^C adapter is a 5-volt device. Any signal at or above 3.3V on the SCL, SDA, and /INT 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^2^C adapter to a SMBus target system, you should follow these steps BEFORE applying power:

• Shut off the iPort/USB internal pull-ups (See Pull-up Resistor section).
• Use external SMBus rated (approximately 15k ohms) pull-up resistors. These pull-ups may already be present in the target system.
• Visit our I^2^C 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^2^C adapters are designed to be I^2^C 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^2^C 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^2^C 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 I²C adapter could damage the host computer, I²C 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 I²C adapter and the target system, the host computer will discharge energy through its serial port to the I²C adapter, and on to the target system. This discharge could damage the host computer, I²C adapter, and the target system.

4.2 Grounding Solutions

To avoid damage to the host computer, I²C adapter, or target system, follow these instructions:

- Wear an earth grounded wrist strap, or discharge any static charge build-up, when handling the I²C 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, I²C 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 iPort/USB software driver.

1. Driver Install

The iPort/USB 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 iPort/USB 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 iPort/USB Driver folder on the host computer.

2. Driver Update

iPort/USB VCP drivers are posted on the MCC website (http://www.mcc-us.com/SoftwareUpgrades-Updates.htm#iPortUSB). Use Windows Device Manager (Start | Settings | Control Panel | System | Device Manager | Ports (COM & LPT)) to see the current version of the iPort/USB 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

iPort/USB VCP drivers can be uninstalled using Windows Device Manager
6. Hardware Set-Up

This section provides information on connecting the I²C adapter to your host computer and I²C Bus target system.

1. USB Connection

After completing the Driver Installation instructions above, plug the iPort/USB adapter into a free USB port on your host computer or self-powered USB hub. If this is the first time the iPort/USB is connected to the host computer, Windows will automatically install the VCP driver from the Windows Driver Store and assign the iPort/USB to a communications port number (COMn).

You can find the ComPort number assigned to the iPort/USB by running the iPort Utility Pack Message Center or Message Manager software, and selecting the iPort/USB 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 iPort/USB.

2. I²C Bus Connection

Connect the I²C Bus cable to the I²C adapter and your I²C device. You can make this connection with the included I²C Clip-Lead cable or I²C Interface Cable.

The I²C interconnect includes a +5V wire. You may not need to, or want to, connect the +5V wire to your target system. Refer to the “+5VDC Power Jack” and “Hardware Configuration” sections for details on pull-up resistors and connecting the optional +5V wire.

3. Optional +5VDC Power Connection

Optionally connect the external power supply to the power jack. iPort/USB power is derived from the USB, with a limited amount of power available to the target system via the I²C connector. Use of the external power supply
provides additional power to the target system via the I²C connector. See “+5VDC Power Jack” section for additional details.

If you have any questions on I²C 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).
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 \(^2\)C Bus Host Adapters. Using the Message Manager, you can perform all four (4) modes of \(^2\)C Bus message activity, including:

- Master Transmit
- Master Receive
- Slave Transmit
- Slave Receive
2 System Requirements

One of the following MCC I²C Bus adapters:

1. i2cStick (#MIIC-207) USB to I²C Bus Host Adapter.
2. iPort/USB (#MIIC-204) USB to I²C Bus Host Adapter.
4. iPort/AI (#MIIC-202) RS-232 to I²C Bus Host Adapter with ASCII Interface
5. iPort (#MIIC-201) Windows to I²C Bus Host Adapter.
6. iPort DLL/USB (#MIIC-201D/U) I²C Bus Host Adapter.
7. Variable Clock Rate (#MIIC-201-V) I²C 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.
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.

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 I\textsuperscript{2}C 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 I\textsuperscript{2}C adapter you are using by clicking the corresponding adapter image (see Opening Screen), or the Device Select checkbox (see Main Application Screen).
4.1.3 Select the Communications Port

Use the “ComPort:” control to select the communication port connected to the I\textsuperscript{2}C 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 iPort/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\textsuperscript{2}C 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\textsuperscript{2}C adapter by clicking the Open Link button.

The Message Center sets the adapter’s own I\textsuperscript{2}C Slave address to 0xFE. Once the link has opened successfully, you are an active I\textsuperscript{2}C node. I\textsuperscript{2}C 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\textsuperscript{2}C Messages

I\textsuperscript{2}C 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\textsuperscript{2}C Message Editor” screen by double clicking on a message row in the spreadsheet.
Use the I²C Message Editor to:

1. Set I²C Address.

The I²C 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).

Use the I²C 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 I²C 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.


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 Center supports Repeated Starts with the doStop control. Sending an I²C message with doStop enabled will cause the message to be terminated with a Stop Condition. Sending an I²C 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 I²C 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 I²C 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 I²C 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 I²C 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.

<table>
<thead>
<tr>
<th>Command</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>iPort, iPort/AI, iPort/AFM, Variable, iPort/USB, i2cStick</td>
<td>Specify I²C adapter type.*</td>
</tr>
<tr>
<td>COM1...COM99</td>
<td>Specify RS-232 communication port.</td>
</tr>
<tr>
<td>BAUD19200, BAUD57600, BAUD115200</td>
<td>Set RS-232 Baud Rate.*</td>
</tr>
<tr>
<td>CLOCK12.5K, CLOCK23K, CLOCK86K, CLOCK100K, CLOCK400K, VCLOCK</td>
<td>Set I²C Bus Clock Rate.*</td>
</tr>
<tr>
<td>Monitor/INT</td>
<td>Enable /INT Signal Monitor.*</td>
</tr>
<tr>
<td>StopOnBusy</td>
<td>Stop sending on I²C adapter busy.</td>
</tr>
<tr>
<td>StopOnArbLoss</td>
<td>Stop sending on I²C Bus Arbitration Loss.</td>
</tr>
<tr>
<td>StopOnNak</td>
<td>Stop on Slave Negative Acknowledgment.</td>
</tr>
<tr>
<td>BeepOnBusy</td>
<td>Beep on I²C adapter busy.</td>
</tr>
<tr>
<td>BeepOnArbLoss</td>
<td>Beep on I²C Bus arbitration loss.</td>
</tr>
<tr>
<td>BeepOnNak</td>
<td>Beep on Slave Negative Acknowledgment.</td>
</tr>
<tr>
<td>BeepOn/INT</td>
<td>Beep on /INT signal assert (low).*</td>
</tr>
<tr>
<td>AutoLoad</td>
<td>Load I²C message list file.</td>
</tr>
<tr>
<td>AutoSave</td>
<td>Save I²C message list file.</td>
</tr>
<tr>
<td>AutoOpen</td>
<td>Open link to I²C adapter.</td>
</tr>
<tr>
<td>AutoSend</td>
<td>Send I²C message list.</td>
</tr>
<tr>
<td>AutoExit</td>
<td>Exit after sending message list.</td>
</tr>
</tbody>
</table>

* 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 I²C 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 I²C Bus Clock Rate

CLOCK12.5K    (iPort ONLY)
CLOCK23K      (iPort/AFM, iPort/USB, i2cStick ONLY)
CLOCK86K      (iPort/AFM, iPort/USB, i2cStick ONLY)
CLOCK100K     (I²C, iPort/AI, iPort/AFM, iPort/USB, i2cStick, Default)
CLOCK400K     (iPort/AFM, iPort/USB, i2cStick ONLY)
VCLOCK=nnnHz  (Variable ONLY. nnn=451...57787)

Set the I²C 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 supported adapters only. Default=OFF)

Enable /INT signal monitoring.

4.2.6 Stop On Busy

StopOnBusy (Default=OFF)

Stop sending I²C messages if the adapter returns a "Busy" response to the host computer.

4.2.7 Stop On Arbitration Loss

StopOnArbLoss (Default=OFF)

Stop sending I²C messages if the adapter returns a "Bus Arbitration Loss" response to the host computer. Bus Arbitration Loss occurs when another I²C 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 I²C 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 I²C 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 I²C 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 I\(^2\)C 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 I\(^2\)C Adapter

Open link to the adapter.

4.2.16 Auto Send

AutoSend Auto Send I\(^2\)C Message List

Send I\(^2\)C 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 I\(^2\)C message list.
5 iPort Message Manager

The iPort Message Manager supports \( I^2C \) Master and Slave, Transmit and Receive activities for all MCC \( I^2C \) Bus host adapters, allowing a PC 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.

Main Screen (Typical)

The Message Manager is designed to be a simple application for experimenting with \( I^2C \) messages. Message Manager provides methods to:

1. Set the \( I^2C \) adapter’s own \( I^2C \) Slave address, General Call Enable, and other operating parameters.
2. Master Transmit ASCII text or Hex (00...FF) data to a specified \( I^2C \) Slave Receiver device.
3. Master Receive data from a specified \( I^2C \) Slave device.
4. Perform Master Read after Write operations.
5. Slave Transmit data to a requesting \( I^2C \) Master device.
6. Display Master or Slave Receive data in hexadecimal or ASCII.
7. Display \( I^2C \) 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).
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.

5.1.3.1 Basic Setup

Use the “ComPort” control to select the communication port connected to the I²C adapter. If an iPort/USB is selected, the serial number for the selected adapter is displayed (Win 2000, XP+ only). In addition to RS-232 and iPort/USB 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 “I²C 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.
5.1.3.2 Advanced Setup

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

**Adapter’s Own I²C Slave Address**

Select the I²C 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 I²C adapter to respond as a slave receiver to the I²C General Call Address (0x00). General Call is used by a master to broadcast an I²C message to multiple devices. The default value is enabled.

**I²C Bus Master Bit Rate**

Select I²C Bus speed during master operations. 100kHz is standard mode. 400kHz is fast mode. Available rates are I²C adapter dependant.

**I²C Bus Time-Out**

Specify how long the I²C adapter will wait before reporting an I²C Bus inter-byte time-out (0 = no time-out, 1 to 32767 milliseconds, iP/Port/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.
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 I²C 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 I²C 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 I²C 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 I²C 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 I²C 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. I²C 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 I²C Bus master operations, the Message Manager can also perform I²C 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 iPort/USB program is easier if you know what to expect. Follow these steps to manually control the iPort/USB from your computer’s keyboard and screen.

1. Install the iPort/USB 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 I²C adapter. Remember to select the correct Com Port (COM1, COM2,…) and set the communication parameters to 19200 Baud, 8 Data Bits, No Parity, and 1 Stop Bit.

3. Enter //\[CR\] to get an iPort/USB Status Report. Note: All iPort/USB 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 iPort/USB’s communications Flow Control to match your terminal.

5. Enter /Ixx[CR] (xx = 02…FE even) to set iPort/USB’s Own I²C Slave Address.

6. Enter /O[CR] to Open the iPort/USB Connection. The iPort/USB does not need to be connected to an I²C Bus to open a connection.

7. Enter /Dxx[CR] (xx = 00…FE even) to select a Destination I²C Slave Address.

8. Enter /Ttext[CR] (text = ASCII or Hex-Equivalent ~00…~FF) to Master Transmit a message to the current Destination I²C Slave device.

9. Enter /Rn[CR] (n = 0…32767) to Master Read a message from the current Destination I²C Slave device.
<table>
<thead>
<tr>
<th>Command</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Ctrl/R,Ctrl/R,Ctrl/R</td>
<td><strong>iPort/USB Reset</strong>&lt;br&gt;This command resets the iPort/USB to its default state.</td>
</tr>
<tr>
<td>//[CR]</td>
<td><strong>Status Display</strong>&lt;br&gt;Display iPort/USB status information.</td>
</tr>
<tr>
<td>/B[0</td>
<td>1</td>
</tr>
<tr>
<td>/C[CR]</td>
<td><strong>Close I²C Connection</strong>&lt;br&gt;Disconnect from the I²C Bus.</td>
</tr>
<tr>
<td>/Dxx[CR]</td>
<td><strong>Set Destination I²C Slave Address</strong>&lt;br&gt;Set the destination I²C Slave Address for subsequent Master Transmit or Receive operations.</td>
</tr>
<tr>
<td>/E[0</td>
<td>1][CR]</td>
</tr>
<tr>
<td>/F[0</td>
<td>1][CR]</td>
</tr>
<tr>
<td>/G[0</td>
<td>1][CR]</td>
</tr>
<tr>
<td>/H[0</td>
<td>1][CR]</td>
</tr>
<tr>
<td>/Ixx[CR]</td>
<td><strong>Set iPort/USB’s Own I²C Slave Address</strong>&lt;br&gt;Sets iPort/USB's own I²C Slave Address. iPort/USB will respond to I²C Bus messages sent to this address.</td>
</tr>
<tr>
<td>/K[0</td>
<td>1</td>
</tr>
<tr>
<td>/M[CR]</td>
<td><strong>Command Menu Display</strong>&lt;br&gt;Displays iPort/USB’s Command Menu</td>
</tr>
<tr>
<td>/N([0</td>
<td>1][A</td>
</tr>
<tr>
<td>/O[CR]</td>
<td><strong>Open I²C Connection</strong>&lt;br&gt;Activates iPort/USB as an I²C device attached to the bus.</td>
</tr>
<tr>
<td>/(*)Rnnnn[CR]</td>
<td><strong>Master Read Message</strong>&lt;br&gt;Read the specified number of data bytes from the current Destination I²C Slave device. * = No Stop for Repeated Start.</td>
</tr>
<tr>
<td>Command</td>
<td>Description</td>
</tr>
<tr>
<td>---------</td>
<td>-----------------------------------------------------------------------------</td>
</tr>
</tbody>
</table>
| /Stext[CR] | **Slave Transmit Message**  
Write the specified data bytes to a requesting I2C Master Receiver device. |
| /(*)Ttext[CR] | **Master Transmit Message**  
Master Transmit the specified data bytes to the current Destination I2C Slave device. * = No Stop for Repeated Start. |
| /Un[CR] | **I2C Bus Time-Out**  
Set I2C Bus Time-out in msec (0=Disable) |
| /V[CR] | **Display Firmware Version**  
(Major XX.XX Minor) |
| /X[CR] | **eXtended Commands**  
(See Prompt or User’s Guide) |
| /Y[CR] | **Display Tx bYte Count** (Requires Firmware V2.0 or later) |

**Synchronous Interface Events**

Synchronous Events are those iPort/USB interface activities initiated by the Host computer.

**iPort/USB Reset**

Reset iPort/USB 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 I2C adapter that could block the transmission. Flow control should be enabled once the response is received.

Command: Ctrl/R,Ctrl/R,Ctrl/R ‘iPort Reset
Response. * ‘iPort/USB Ready
Default Setting: None

**Status Display**

Display iPort/USB status.
Command: //[CR] 'Status Display
Response:
  iPort/USB I²C Host Adapter w/ASCII Fast Mode Interface Vxx.xx
  Copyright © xxxx, Micro Computer Control Corp.
  Visit our Web Site at: http://www.mcc-us.com

Serial Communications Baud Rate (19.2kHz)
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]

**Close I²C Connection**

Disconnect iPort/USB 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: *  'iPort/USB 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: *  'iPort/USB Ready
Default Setting: Off

**Serial Communications Flow Control**

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

iPort/USB implements either XON/XOFF (by default) or RTS/CTS flow control protocols. Flow control is used by the iPort/USB 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: *    'iPort/USB Ready
Default Setting:  XON/XOFF

I²C General Call Control

Enables or disables iPort/USB response to I²C Bus General Call (Address x00)
messages.

Command: /G[0|1][CR]  'I²C General Call [0 = Disabled, 1 = Enabled]
Response: *    'iPort/USB Ready
Default Setting:  Enabled

Hex Only Display Control

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

When enabled, all received I²C message data bytes are displayed in Hex
(~00…~FF) format. When disabled, received I²C 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: *    'iPort/USB Ready
Default Setting:  Enabled

Set iPort/USB’s Own I²C Slave Address

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

Command: /Ixx[CR]    'Set iPort/USB’s Own I²C Slave Address
Response 1: *           'iPort/USB 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 iPort/USB's Clock Rate
Response 1: *           'iPort/USB Ready
Default Setting: /K2[CR]

The iPort/USB 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 iPort/USB’s command menu.

Command: /M[CR]       'Command Menu Display
Response:
iPort/USB 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 iPort/USB’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)
Open I2C Connection

Master Rx Message *=No Stop
Slave Tx Message
Master Tx Message *=No Stop
Set I2C Bus Time-out in msec 0=Disable
Display Firmware Version (Major XX.XX Minor)
Extended Cmds (See Prompt or User's Guide)
Display Tx Byte Count

Interrupt Signal Control/Status

The INT signal allows the iPort/USB to participate in INT master and/or slave communications.

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: *  'iPort/USB Ready
Default Setting: /N0, /NR

Open I2C Connection

Activates iPort/USB as an active device on the I2C Bus.

Command: /O[CR]  'Open I2C Connection
Response: /OCC[CR]  'Open Connection Complete
Default Setting:  Closed

Master Read Message

This command causes iPort/USB to read the specified number of data bytes from the currently selected Destination I2C Slave Address with or without generating an I2C 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 I²C Slave device indicates the number of additional trailing bytes that are available to read. The iPort/USB 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 I²C Slave Address, the specified number of bytes are read. The iPort/USB acknowledges all bytes read except the last. If not disabled, the message is then terminated with an I²C Stop condition.

Sending Master Receive messages with No Stop allows the Master to retain exclusive control of the I²C 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 I²C 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]      'iPort/USB is Busy, Command Ignored
Response 4: /I83[CR]      'I²C Arbitration Loss Detected
Response 5: /I88[CR]      'iPort Connection Not Open
Response 6: /I89[CR]      'Invalid Command Argument
Default Setting: None

**Slave Transmit Message**

This command should be issued to iPort/USB in response to a Slave Transmit Request (/STR). This command causes iPort/USB to write the specified data bytes to the requesting I²C 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 I²C Bus, the iPort/USB outputs a Slave Transmit Request to its Host device, and initiates an I²C 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 I²C Bus.

Note 2: The tilde (~) character and the Carriage Return (CR) characters are used as special marker characters within all iPort/USB 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"

iPort/USB automatically translates "Hex equivalent" characters to their single-byte value for transmission across the I²C 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]  'iPort Connection Not Open
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 I²C Slave Address with or without generating an I²C 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 iPort/USB 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"

iPort/USB automatically translates "Hex equivalent" characters to their single-byte
value for transmission across the I²C Bus.

All entered data bytes are transmitted to the Destination I²C 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 I²C
Stop condition.

Sending Master Transmit messages with No Stop allows the Master to retain
exclusive control of the I²C 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 I²C Slave devices.

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]    'iPort/USB is Busy, Command Ignored
Response 4: /I83[CR]    'I²C Arbitration Loss Detected
Response 5: /I88[CR]    'iPort 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\(^2\)C Bus Time-oUt in msec

Set bus time-out in milliseconds.

The iPort/USB reports a bus time-out if no inter-byte bus activity for the specified time occurs within an I\(^2\)C Bus message.

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

Display Firmware Version

Display 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\(^2\)C hardware to control the SCL and SDA lines, as the I\(^2\)C hardware only generates I\(^2\)C 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\(^2\)C Bus clock-stretching by external slave devices.

The following commands manipulate the I\(^2\)C 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 Response: /XCC0000[CR]

Low Level Command:/X dc dCcDCcdCcdCeDCcDCcDCcCeDCcCeDc DCACe 
dCcdCcdCcdCcdCcdCeDCcCeDCcCeDCcCeDCcCeDcCDcCeDCcCeDCcCeDCcCeDCcCeDcCD[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: /XCCA~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.
Command: /Y[CR]    'Tx bYte Count
Response: /TBCn[CR] 'n =00000...32767
Asynchronous Interface Events

Asynchronous Events are those iPort/USB interface activities initiated by the iPort/USB I²C Host Adapter in response to activities on the I²C Bus.

Slave Transmit Request

This event is caused by the reception of an I²C Bus Slave Transmit message directed at the current iPort/USB’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 I²C Bus, iPort/USB outputs a Slave Transmit Request to its Host device, and initiates an I²C 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 I²C Bus.

Slave Receive Complete

This event is caused by the reception of an I²C Bus Slave Receive message directed at the current iPort/USB’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 I²C Bus Slave Receive message directed at the I²C General Call Address (00), when iPort/USB’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

**iPort/USB Ready**

Prompt: * ‘iPort/USB Ready

Cause: iPort/USB is ready for the next Host command.

**Slave Not Acknowledging**

Prompt: /SNA[CR] ‘Slave Not Acknowledging

Cause: There is no response (I²C Slave Address Acknowledgment) during a Master Transmit or Receive operation from an I²C Slave device at the current Destination I²C Address.

**iPort/USB Busy**

Prompt: /I81[CR] ‘iPort/USB Busy

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

**I²C Bus Arbitration Loss**

Prompt: /I83[CR] ‘I²C Arbitration Loss Detected

Cause: iPort/USB lost I²C Bus Arbitration to another bus master device while Master Transmitting or Master Receiving an I²C message. Host should process any active slave events and repeat the last command.

**I²C Bus Error Detected**

Prompt: /I84[CR] ‘I²C Bus Error Detected
Cause: iPort/USB has detected an error condition on the I²C Bus. The host computer should retry the last command or issue an iPort/USB Reset command.

I²C Bus Time-out Detected

Prompt: /I85[CR] ‘I²C Bus Time-out Detected

Cause: iPort/USB issues this response when it detects a byte transfer delay greater than the specified I²C Bus Time-out period. No corrective action is taken by the iPort/USB regarding I²C Bus activity. No host computer response is required, but this event can be used to detect possible bus problems.

iPort/USB Connection Closed

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

Cause: The host computer is attempting to perform an I²C Bus message operation while the iPort/USB Connection is Closed. The host computer should issue an Open I²C Connection command before attempting to perform I²C 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 iPort/USB Command

Prompt: /I8F[CR] ‘Invalid iPort/USB Command

Cause: This event normally indicates that an invalid command was issued by the
host. The host should reissue the correct command.

**iPort/USB Receive Buffer Overflow**

**Prompt:** /I90[CR] ‘iPort/USB Serial Receive Buffer Overflow

**Cause:** This event normally indicates that data sent to the iPort/USB 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 iPort/USB 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 iPort/USB interface programs in other programming languages and operating environments.

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

iPort/USB Reset
Comm1.Output = Chr$(18) 'Ctrl/R
Comm1.Output = Chr$(18) 'Ctrl/R
Comm1.Output = Chr$(18) 'Ctrl/R

iPort/USB Initialization
Comm1.Output = "/f0" 'Set iPort/USB XON/XOFF Flow Control
Comm1.Output = Chr$(13)

Comm1.Output = "/i70" 'Set iPort/USB’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
    MsgBox$ = Comm1.Input ' Get Comm input character
    CharIn$ = MsgBox$

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

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

    ' Test for iPort/USB 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 "
62
ElseIf (StrComp(iPortResp$, "/BC0") = 0) Then
' iPort/USB Baud Change 0 {19.2K}
    TextBox.Text = "iPort/USB Baud Change 0 {19.2K} "

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

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

' Test for iPort/USB 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 iPort/USB Response Messages

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

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

ElseIf (StrComp(iPortResp$, "/I81") = 0) Then
    TextBox.Text = "/I81 iPort/USB 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 iPort/USB 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 iPort/USB Command"

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

Else
    TextBox.Text = LineBuf$ 'Other Update Display
End If

TextBox.Text = LineBuf$ = ""

End If

End Sub
iPort/USB Revision Report

This section defines revisions and changes made to the iPort/USB interface:

Revision: 1.00

1 Initial Release

Additional Information

For additional information on the I²C Bus, please refer to the following:

“What is I²C?”
www.mcc-us.com/I2CBusTechnicalOverview.pdf

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

"The I²C and How to Use It"
www.mcc-us.com/i2chowto.htm

"80C51-Based 8-Bit Microcontroller" Data Handbook.
Philips Semiconductors, Tel. (800)227-1817

"I²C Peripherals for Microcontrollers" Data Handbook.
Philips Semiconductors, Tel. (800)227-1817
Appendix A - I²C Connector Information

Interface Connector and Plug Information

MCC uses two (2) different connectors and plug assemblies. We have found these parts to be compatible.

I²C Receptacle Connectors

Molex SEMCONN ACCESS.bus Receptacle Connector

Molex Part # 15-83-0064

AMP SDL (Shielded Data Link) Connectors for ACCESS.bus

AMP Part # 4-943197-1

I²C Plug Connectors

Molex SEMCONN ACCESS.bus Plug

Molex Part # 15-83-1564

AMP SDL (Shielded Data Link) Plug for ACCESS.bus

Bush Amp Part # 520851-1
Ferrule Amp Part # 520433-1
SDL (Shell) Amp Part # 520461-1
SDL (Shell) Amp Part # 520460-1
SDL Amp Part # 4-520424-1

The following I²C Cables are available from MCC

MCC Part # CAB4 I²C Interface Cable, 48 inches (4ft)
MCC Part # CAB8 I²C Interface Cable, 96 inches (8ft)
MCC Part # CABCL I²C and SMBus Clip Lead Cable
Declaration of Conformity

This Declaration of Conformity is issued by the indicated company which is solely responsible for the declared compliance.

Product(s): iPort/USB and iPort/ DLL/USB
Product Part Number(s): MIIC-204 and MIIC-201D/U
Product Description: I2C Bus Host Adapter


Compliant Standards:

EN 55022 : 1998
Emissions Standard
Conducted Emissions (Class B)
Radiated Emissions (Class B)

EN 55024 : 1998
Immunity Standard
Immunity to Radiated Electromagnetic Fields
Immunity to Fast Transient Bursts - AC Power Lines
Immunity to Conducted Field - AC Power Lines
Immunity to Voltage Dips - AC Power Lines
Immunity to Electrostatic Discharge

Test Laboratory Information:

Cass Industries Ltd., Blackbrook Trading Estate, Weybrook Road, Manchester M19 2QD, ENGLAND.
Test Report Number: CI02780
Test Report Date: April 6th, 2006
Technical file held by: Micro Computer Control Corporation, 17 Model Avenue / PO Box 275, Hopewell, New Jersey 08525 USA, or its applicable authorized distributor or representative.

Responsible Company: Micro Computer Control Corporation, 17 Model Avenue / PO Box 275, Hopewell, New Jersey 08525 USA, or its applicable authorized distributor or representative.

Signature of Authorized Representative:

Edward Thompson

Name: Edward Thompson
Title: President, Micro Computer Control Corporation
Date: 09-JAN-07