MCP2200

USB 2.0 to UART Protocol Converter with GPIO

Features

Universal Serial Bus (USB)
- Supports Full-Speed USB (12 Mb/s)
- Implements USB Protocol Composite Device:
  - Communication Device Class (CDC) for Communications and Configuration
  - Human Interface Device (HID) for I/O control
- 128-Byte Buffer to Handle Data Throughput at Any UART Baud Rate:
  - 64-byte transmit
  - 64-byte receive
- Fully Configurable VID and PID Assignments and String Descriptors
- Bus-Powered or Self-Powered
- USB 2.0 Compliant: TID 40001150

USB Driver and Software Support
- Uses Standard Windows® Drivers for Virtual Com Port (VCP): Windows XP (SP2 or later), Windows Vista, Windows 7, Windows 8, Windows 8.1 and Windows 10
- Configuration Utility for Initial Configuration

Universal Asynchronous Receiver/Transmitter (UART)
- Responds to SET_LINE_CODING Commands to Dynamically Change Baud Rates
- Supports Baud Rates: 300-1000k
- Hardware Flow Control
- UART Signal Polarity Option

General Purpose Input/Output (GPIO) Pins
- Eight General Purpose I/O pins

EEPROM
- 256 Bytes of User EEPROM

Other
- USB Activity LED Outputs (TxLED and RxLED)
- SSPND Output Pin
- USBCFG Output Pin (indicates when the enumeration is completed)
- Operating Voltage: 3.0V-5.5V
- Oscillator Input: 12 MHz
- Electrostatic Discharge (ESD) Protection: >4 kV
  Human Body Model (HBM)
- Industrial (I) Operating Temperature: –40°C to +85°C

Package Types

The device is offered in the following packages:
- 20-lead VQFN (5x5 mm)
- 20-lead SOIC
- 20-lead SSOP

* Includes Exposed Thermal Pad (EP); see Table 1-1.
1.0 FUNCTIONAL DESCRIPTION

The MCP2200 is a USB-to-UART serial converter that enables USB connectivity in applications that have a UART interface. The device reduces external components by integrating the USB termination resistors. The MCP2200 also has 256 bytes of integrated user EEPROM.

The MCP2200 has eight general purpose input/output pins. Four pins have alternate functions to indicate USB and communication status. See Table 1-1 and Section 1.6 “GPIO Module” for details about the pin functions.

**TABLE 1-1: PINOUT DESCRIPTION**

<table>
<thead>
<tr>
<th>Pin Name</th>
<th>VQFN</th>
<th>SSOP, SOIC</th>
<th>Pin Type</th>
<th>Standard Function</th>
<th>Alternate Function</th>
</tr>
</thead>
<tbody>
<tr>
<td>GP0/SSPND</td>
<td>13</td>
<td>16</td>
<td>I/O</td>
<td>General purpose I/O</td>
<td>USB suspend status pin (refer to Section 1.6.1.1 “SSPND Pin Function”)</td>
</tr>
<tr>
<td>GP1/USB-CFG</td>
<td>12</td>
<td>15</td>
<td>I/O</td>
<td>General purpose I/O</td>
<td>USB configuration status pin (refer to Section 1.6.1.2 “USBCFG Pin Function”)</td>
</tr>
<tr>
<td>GP2</td>
<td>11</td>
<td>14</td>
<td>I/O</td>
<td>General purpose I/O</td>
<td></td>
</tr>
<tr>
<td>GP3</td>
<td>6</td>
<td>9</td>
<td>I/O</td>
<td>General purpose I/O</td>
<td></td>
</tr>
<tr>
<td>GP4</td>
<td>5</td>
<td>8</td>
<td>I/O</td>
<td>General purpose I/O</td>
<td></td>
</tr>
<tr>
<td>GP5</td>
<td>4</td>
<td>7</td>
<td>I/O</td>
<td>General purpose I/O</td>
<td></td>
</tr>
</tbody>
</table>
| GP6/RxLED | 3   | 6  | I/O | General purpose I/O | USB receive activity LED output (refer to Section 1.6.1.3 “RxLED Pin Function (IN Message)”)
| GP7/TxLED | 2   | 5  | I/O | General purpose I/O | USB transmit activity LED output (refer to Section 1.6.1.4 “TxLED Pin Function (OUT Message)”)
| CTS | 10  | 13 | I  | Hardware flow control “Clear to Send” input signal | |
| RTS | 8   | 11 | O  | Hardware flow control “Request to Send” output signal | |
| Rx | 9   | 12 | I  | USART RX input | |
| Tx | 7   | 10 | O  | USART TX output | |
| RST | 1   | 4  | I  | Reset input must be externally biased | |
| VDD | 18  | 1  | P  | Power | |
| VSS | 17  | 20 | P  | Ground | |
| OSC1 | 19  | 2  | I  | Oscillator input | |
| OSC2 | 20  | 3  | O  | Oscillator output | |
| D+ | 16  | 19 | I/O | USB D+ | |
| D- | 15  | 18 | I/O | USB D- | |
| Vusb | 14  | 17 | P  | USB power pin (internally connected to 3.3V). Should be locally bypassed with a high-quality ceramic capacitor. | |
1.1 Supported Operating Systems
Windows XP (SP2 and later), Windows Vista, Windows 7, Windows 8, Windows 8.1 and Windows 10 operating systems are supported.

1.1.1 Enumeration
The MCP2200 will enumerate as a USB device after Power-on Reset (POR). The device enumerates as both a Human Interface Device (HID) for I/O control, and a Virtual Com Port (VCP).

1.1.1.1 Human Interface Device (HID)
The MCP2200 enumerates as an HID, so the device can be configured and the I/O can be controlled. A DLL that facilitates I/O control through a custom interface is supplied by Microchip.

1.1.1.2 Virtual Com Port (VCP)
The VCP enumeration implements the USB-to-UART data translation.

1.2 Control Module
The control module is the heart of the MCP2200. All other modules are tied together and controlled via the control module. The control module manages the data transfers between the USB and the UART, as well as the command requests generated by the USB host controller and the commands for controlling the function of the UART and I/O.

1.2.1 Serial Interface
The control module interfaces to the UART and USB modules.

1.2.2 Interfacing to the Device
The MCP2200 can be accessed for reading and writing via USB host commands. The device cannot be accessed and controlled via the UART interface.

1.3 UART Interface
The MCP2200 UART interface consists of the Tx and Rx data signals and the RTS/CTS flow control pins.
The UART is configurable for several baud rates. The available baud rates are listed in Table 1-3.

1.3.1 Initial Configuration
The default UART configuration is 19200, 8, N, 1. The default start-up baud rate can be changed using the Microchip-supplied configuration PC tool.
Alternatively, a custom configuration tool can be created using the Microchip-supplied DLL to set the baud rate as well as other parameters. See Section 2.0 "Configuration" for details.

TABLE 1-2: UART Configurations

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Configuration</th>
</tr>
</thead>
<tbody>
<tr>
<td>Primary Baud Rates</td>
<td>See Table 1-3</td>
</tr>
<tr>
<td>Data Bits</td>
<td>8</td>
</tr>
<tr>
<td>Parity</td>
<td>N</td>
</tr>
<tr>
<td>Stop Bits</td>
<td>1</td>
</tr>
</tbody>
</table>

1.3.2 Get/Set Line Coding
The GET_LINE_CODING and SET_LINE_CODING commands are used to read and set the UART parameters while in operation. For example, HyperTerminal sends the SET_LINE_COMMAND when connecting to the port. The MCP2200 responds by setting the baud rate only. The other parameters (data bits, parity, stop bits) remain unchanged.

1.3.2.1 Rounding Errors
The primary baud rate setting (with the rounding errors) is shown in Table 1-3. If baud rates other than the ones shown in the table are used, the error percentage can be calculated using Equation 1-1 to find the actual baud rate.

TABLE 1-3: UART Primary Baud Rates

<table>
<thead>
<tr>
<th>Desired Rate</th>
<th>Actual Rate</th>
<th>% Error</th>
</tr>
</thead>
<tbody>
<tr>
<td>300</td>
<td>300</td>
<td>0.00%</td>
</tr>
<tr>
<td>1200</td>
<td>1200</td>
<td>0.00%</td>
</tr>
<tr>
<td>2400</td>
<td>2400</td>
<td>0.00%</td>
</tr>
<tr>
<td>4800</td>
<td>4800</td>
<td>0.00%</td>
</tr>
<tr>
<td>9600</td>
<td>9600</td>
<td>0.00%</td>
</tr>
<tr>
<td>19200</td>
<td>19200</td>
<td>0.00%</td>
</tr>
<tr>
<td>38400</td>
<td>38339</td>
<td>0.16%</td>
</tr>
<tr>
<td>57600</td>
<td>57692</td>
<td>0.16%</td>
</tr>
<tr>
<td>115200</td>
<td>115385</td>
<td>0.16%</td>
</tr>
<tr>
<td>230400</td>
<td>230769</td>
<td>0.16%</td>
</tr>
<tr>
<td>460800</td>
<td>461538</td>
<td>0.16%</td>
</tr>
<tr>
<td>921600</td>
<td>923077</td>
<td>0.16%</td>
</tr>
</tbody>
</table>
1.3.3 CUSTOM BAUD RATES

Custom baud rates are configured by sending the `SET_LINE_CODING` USB command, or by using the DLL. See Section 2.0 “Configuration” for more information.

1.3.4 HARDWARE FLOW CONTROL

Hardware flow control uses the RTS and CTS pins as a handshake between two devices. The RTS pin of one device is typically connected to the CTS of the other device.

RTS is an active-low output that notifies the other device when it is ready to receive data by driving the pin low. The MCP2200 trip point for deasserting RTS (high is 63 characters. This is one character short of “buffer full”.

CTS is an active-low input that notifies the MCP2200 when it is ready to send data. The MCP2200 will check CTS just before loading and sending UART data. If the pin is asserted during a transfer, the transfer will continue. Refer to Figure 1-1.

1.3.4.1 Flow Control Disabled

The buffer pointer does not increment (or reset to zero) if the buffer is full. Therefore, if hardware flow control is not enabled and an overflow occurs (i.e., 65 unprocessed characters received), the new data overwrites the last position in the buffer.

1.4 USB Protocol Controller

The USB controller in the MCP2200 is full-speed USB 2.0 compliant.

- Composite device (CDC + HID):
  - CDC: USB-to-UART communications
  - HID: I/O control, EEPROM access and initial configuration
- 128-byte buffer to handle data throughput at any UART baud rate:
  - 64-byte transmit
  - 64-byte receive
- Fully configurable VID and PID assignments and descriptors (stored on-chip)
- Bus-powered or self-powered

1.5 USB Transceiver

The MCP2200 has a built-in, full-speed USB 2.0 transceiver internally connected to the USB module.

The USB transceiver obtains power from the VUSB pin, which is internally connected to the 3.3V regulator. The best electrical signal quality is obtained when VUSB is locally bypassed with a high-quality ceramic capacitor.

1.5.1 INTERNAL PULL-UP RESISTORS

The MCP2200 devices have built-in pull-up resistors designed to meet the requirements for full-speed USB.

1.5.2 MCP2200 POWER OPTIONS

The following are the main power options for the MCP2200:

- USB Bus-Powered (5V)
- 3.3V Self-Powered

---

**EQUATION 1-1: SOLVING FOR ACTUAL BAUD RATE**

\[
\text{Actual Rate} = \frac{12 \text{ MHz}}{\text{int}(x)}
\]

Where:

\[
x = \frac{12 \text{ MHz}}{\text{Desired Baud}}
\]
1.5.2.1 Internal Power Supply Details

MCP2200 offers various options for power supply. To meet the required USB signaling levels, the MCP2200 device incorporates an internal LDO used solely by the USB transceiver in order to present the correct D+/D- voltage levels.

Figure 1-2 shows the internal connections of the USB transceiver LDO in relation to the VDD power supply rail. The output of the USB transceiver LDO is tied to the VUSB line. A capacitor connected to the VUSB pin is required if the USB transceiver LDO provides the 3.3V supply to the transceiver.

![FIGURE 1-2: MCP2200 INTERNAL POWER SUPPLY DETAILS](image)

The provided VDD voltage has a direct influence on the voltage levels present on the GPIO pins (Rx/Tx and RTS/CTS). When VDD is 5V, all of these pins will have a logical ‘1’ around 5V with the variations specified in Section 3.1 “DC Characteristics”.

For applications that require a 3.3V logical ‘1’ level, VDD must be connected to a power supply providing 3.3V voltage. In this case, the internal USB transceiver LDO cannot provide the required 3.3V of power. It is necessary to also connect the VUSB pin to the 3.3V power supply rail. This way, the USB transceiver is powered-up directly from the 3.3V power supply.

1.5.2.2 USB Bus-Powered (5V)

In Bus Power Only mode, all power for the application is drawn from the USB (Figure 1-3). This is effectively the simplest power method for the device.

![FIGURE 1-3: BUS POWER ONLY](image)

In order to meet the inrush current requirements of the USB 2.0 specifications, the total effective capacitance appearing across VBUS and ground must be no more than 10 µF. If it is not more than 10 µF, some kind of inrush current limiting is required. For more details on inrush current limiting, consider the latest version of the “Universal Serial Bus Specification”.

According to the USB 2.0 specification, all USB devices must also support a low-power Suspend mode. In the USB Suspend mode, devices must consume no more than 500 µA (or 2.5 mA for high-powered devices that are remote wake-up capable) from the 5V VBus line of the USB cable.

The host signals the USB device to enter Suspend mode by stopping all USB traffic to that device for more than 3 ms.

The USB bus provides a 5V voltage. However, the USB transceiver requires 3.3V for the signaling (on the D+ and D- lines).

During USB Suspend mode, the D+ or D- pull-up resistor must remain active, which will consume some of the allowed suspended current budget (500 µA/2.5 mA). The VUSB pin is required to have an external bypass capacitor. It is recommended that the capacitor be a ceramic capacitor between 0.22 µF and 0.47 µF.

Figure 1-4 shows a circuit where MCP2200’s internal LDO is used to provide 3.3V to the USB transceiver. The voltage on the VDD affects the voltage levels on the GPIO pins (Rx/Tx and RTS/CTS). With VDD at 5V, these pins will have a logic ‘1’ of 5V with the variations specified in Section 3.1 “DC Characteristics”.

1.5.2.3 3.3V Self-Powered

Typically, many embedded applications are using 3.3V power supplies. When such an option is available in the target system, MCP2200 can be powered up from the existing 3.3V power supply rail. The typical connections for the MCP2200 are shown in Figure 1-5.

In this example, the MCP2200 has both VDD and VUSB lines tied to the 3.3V rail. These tied connections disable the internal USB transceiver LDO of the MCP2200 to regulate the power supply on VUSB pin. Another consequence is that the ‘1’ logical level on the GPIO pins will be at the 3.3V level, in accordance with the variations specified in Section 3.1 “DC Characteristics”.

1.6 GPIO Module

The GPIO Module is a standard 8-bit I/O port.

1.6.1 CONFIGURABLE PIN FUNCTIONS

The pins can be configured as:

- GPIO – individually configurable general purpose input or output
- SSPND – USB Suspend state
- USBCFG – indicates USB configuration status
- RxLED – indicates USB receive traffic
- TxLED – indicates USB transmit traffic

1.6.1.1 SSPND Pin Function

The SSPND pin (if enabled) reflects the USB state (Suspend/Resume). The pin is active-low when the Suspend state has been issued by the USB host. Likewise, the pin drives ‘high’ after the Resume state is achieved.

This pin allows the application to go into low power mode when USB communication is suspended, and switches to a full active state when USB activity is resumed.

1.6.1.2 USBCFG Pin Function

The USBCFG pin (if enabled) starts out ‘low’ during power-up or after Reset, and goes ‘high’ after the device successfully configures to the USB. The pin will go ‘low’ when in Suspend mode and ‘high’ when the USB resumes.
1.6.1.3 **RxLED Pin Function (IN Message)**
The ‘Rx’ in the pin name refers to the USB host. The RxLED pin is an indicator for USB ‘IN’ messages.
This pin will either pulse low for a period of time (configurable for ~100 ms or ~200 ms), or toggle to the opposite state for every message received (IN message) by the USB host. This allows the application to count messages or provide a visual indication of USB traffic.

1.6.1.4 **TxLED Pin Function (OUT Message)**
The ‘Tx’ in the pin name refers to the USB host. The TxLED pin is an indicator for USB ‘OUT’ messages.
This pin will either pulse low for a period of time (configurable for ~100 ms or ~200 ms), or toggle to the opposite state for every message transmitted (OUT message) by the USB host. This allows the application to count messages or provide a visual indication of USB traffic.

1.7 **EEPROM Module**
The EEPROM module is a 256-byte array of nonvolatile memory. The memory locations are accessed for read/write operations via USB host commands. Refer to Section 2.0 “Configuration” for details on accessing the EEPROM. The memory cells for data EEPROM are rated to endure thousands of erase/write cycles, up to 100K for EEPROM.
Data retention without refresh is conservatively estimated to be greater than 40 years.
The host should wait for the write cycle to complete and then verify the write by reading the byte(s).

1.8 **RESET/POR**
1.8.1 **RESET PIN**
The RST pin provides a method for triggering an external Reset of the device. A Reset is generated by holding the pin low. These devices have a noise filter in the Reset path which detects and ignores small pulses.

1.8.2 **POWER-ON RESET (POR)**
A POR pulse is generated on-chip whenever VDD rises above a certain threshold. This allows the device to start in the initialized state when VDD is adequate for operation.
To take advantage of the POR circuitry, tie the RST pin through a resistor (1 kΩ to 10 kΩ) to VDD. This will eliminate external RC components usually needed to create a POR delay.
In the self-powered configuration, it is recommended to tie the RST pin to the VBUS line of the USB connector, as in Figure 1-6.

**FIGURE 1-6: CONNECTING THE RST PIN IN A SELF-POWERED CONFIGURATION (RECOMMENDED)**

When the device starts normal operation (i.e., exits the Reset condition), device operating parameters (voltage, frequency, temperature, etc.) must be met to ensure operation. If these conditions are not achieved, the device must be held in Reset until the operating conditions are met.
1.9 Oscillator

The input clock must be 12 MHz to provide the proper frequency for the USB module.

USB full speed is defined as 12 Mb/s. The clock input accuracy is ±0.25% (2,500 ppm maximum).

**FIGURE 1-7: QUARTZ CRYSTAL OPERATION**

![Quartz Crystal Connection Diagram]

Note 1: A series resistor (Rs) may be required for quartz crystals with high drive level.

2: The value of RF is typically between 2 MΩ to 10 MΩ.

**FIGURE 1-8: CERAMIC RESONATOR OPERATION**

![Ceramic Resonator Connection Diagram]

Example: Murata® CSTCE12M0G15L
2.0  CONFIGURATION

The MCP2200 is configured by writing special commands using the HID interface. Configuration can be achieved using the configuration utility provided by Microchip. Alternatively, a custom utility can be developed by using the DLL available on the MCP2200 product page.

2.1  Configuration Utility

The configuration utility provided by Microchip allows the user to configure the MCP2200 to custom defaults. The configuration utility (shown in Figure 2-1) connects to the device’s HID interface, where all of the configurable features can be set.

2.2  Serial String

The MCP2200 is supplied from the factory with a serialized USB serial string.

TABLE 2-1:  CONFIGURATION DESCRIPTIONS

<table>
<thead>
<tr>
<th>Configuration Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Vendor ID (0x04D8)</td>
<td>The USB vendor identification assigned to Microchip by the USB consortium.</td>
</tr>
<tr>
<td>Product ID (0x00DF)</td>
<td>Device ID assigned by Microchip. The device can be used as-is, or Microchip can assign a custom PID by request.</td>
</tr>
<tr>
<td>Baud Rate</td>
<td>Sets the UART baud rate using a list of primary baud rates. See the UART section for details on setting non-primary baud rates.</td>
</tr>
<tr>
<td>IO Config</td>
<td>Individually configures the I/O to inputs or outputs.</td>
</tr>
<tr>
<td>IO Default</td>
<td>Individually configures the output default state for pins configured as outputs.</td>
</tr>
<tr>
<td>Tx/Rx LEDs</td>
<td>Enables/disables the GP6 and GP7 pins to function as USB traffic indicators. Pins are active-low when configured as traffic indicators.</td>
</tr>
<tr>
<td>Hardware Flow Control</td>
<td>Enables/disables CTS and RTS flow control.</td>
</tr>
<tr>
<td>USBCFG Pin</td>
<td>Enables/disables the GP1 pin as a USB configuration status indicator.</td>
</tr>
<tr>
<td>Suspend Pin</td>
<td>Enables/disables the GP0 pin as a USB suspend status pin.</td>
</tr>
<tr>
<td>Invert Sense</td>
<td>Enables/disables the UART lines states:</td>
</tr>
<tr>
<td></td>
<td>- Normal – Tx/Rx idle-high; CTS/RTS active-low</td>
</tr>
<tr>
<td></td>
<td>- Inverted – Tx/Rx idle-low; CTS/RTS active-high</td>
</tr>
<tr>
<td>Manufacturer String</td>
<td>USB manufacturer string.</td>
</tr>
<tr>
<td>Product String</td>
<td>USB product string.</td>
</tr>
</tbody>
</table>
FIGURE 2-1: CONFIGURATION UTILITY

[Image: MCP2200 Configuration Utility interface with menus and options]

- **Vendor ID**: 0x0408
- **Product ID**: 0x00DF
- **Baud Rate**: 115200
- **I/O Config**: 00111111
- **Output Default**: 00000000
- **LED Function**: Blink LEDs, Blink Duration: 100ms

**String Descriptors**
- **Manufacturer**: Microchip Technology Inc.
- **Product**: MCP2200 USB Serial Port Emulator

Writing:
- Manufacturer String Descriptor
- Product String Descriptor
- Configurations
- Device Configured
- VID/PID
- Descriptor block
- Manufacturer String block
- Descriptor block
- Default Settings block
- Device Verified

Connected: [Image: Connected icon]
Reset to Default
2.3 Simple Configuration and I/O DLL

To help the user develop a custom configurator, Microchip provides a DLL that uses Microsoft® .NET Framework 3.5. There is documentation about drivers and utilities on the MCP2200 product page at www.microchip.com (in the Software section) with information on associating the DLL with a Visual C++ project.

Table 2-2 lists the functions provided by the DLL to allow the configuration of the device and control of the I/O.

TABLE 2-2: CONFIGURATION FUNCTIONS

<table>
<thead>
<tr>
<th>Category and Function Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Initialization (Note 1)</td>
<td>void InitMCP2200(VID, PID)</td>
</tr>
<tr>
<td>Configuration (Note 2)</td>
<td>bool ConfigureIO(mask)</td>
</tr>
<tr>
<td></td>
<td>bool ConfigureIODefaultOutput(mask, defaultGpioOutputValue)</td>
</tr>
<tr>
<td></td>
<td>bool fnRxLED (OFF/TOGGLE/BLINKSLOW/BLINKFAST)</td>
</tr>
<tr>
<td></td>
<td>bool fnTxLED (OFF/TOGGLE/BLINKSLOW/BLINKFAST)</td>
</tr>
<tr>
<td></td>
<td>bool fnHardwareFlowControl (ON/OFF)</td>
</tr>
<tr>
<td></td>
<td>bool fnULoad(ON/OFF)</td>
</tr>
<tr>
<td></td>
<td>bool fnSuspend (ON/OFF)</td>
</tr>
<tr>
<td></td>
<td>bool ConfigureMCP2200(mask, baudrate, RxLedMode, TxLedMode, flowCtrl, ULoad, suspend)</td>
</tr>
<tr>
<td></td>
<td>bool ConfigureIO(mask)</td>
</tr>
<tr>
<td>Miscellaneous</td>
<td>String^ GetDeviceInfo(deviceIndex)</td>
</tr>
<tr>
<td></td>
<td>unsigned int GetNoOfDevices()</td>
</tr>
<tr>
<td></td>
<td>int GetSelectedDevice()</td>
</tr>
<tr>
<td></td>
<td>String^ GetSelectedDeviceInfo()</td>
</tr>
<tr>
<td></td>
<td>bool IsConnected()</td>
</tr>
<tr>
<td></td>
<td>int SelectDevice(uiDeviceNo)</td>
</tr>
<tr>
<td></td>
<td>int ReadEEPROM(uiEEPAddress)</td>
</tr>
<tr>
<td></td>
<td>int WriteEEPROM(uiEEPAddress, ucValue)</td>
</tr>
<tr>
<td>I/O Control</td>
<td>bool ClearPin(pinnumber)</td>
</tr>
<tr>
<td></td>
<td>bool SetPin(pinnumber)</td>
</tr>
<tr>
<td></td>
<td>bool ReadPin(pinnumber, *pinvalue)</td>
</tr>
<tr>
<td></td>
<td>int ReadPinValue(pinnumber)</td>
</tr>
<tr>
<td></td>
<td>bool ReadPort(*portValue)</td>
</tr>
<tr>
<td></td>
<td>int ReadPortValue()</td>
</tr>
<tr>
<td></td>
<td>bool WritePort(portValue)</td>
</tr>
<tr>
<td>Summary</td>
<td>bool SimpleIOClass::ClearPin(unsigned int pin) Section 2.3.1.1</td>
</tr>
<tr>
<td></td>
<td>bool SimpleIOClass::ConfigureIO (unsigned char IOMap) Section 2.3.1.2</td>
</tr>
<tr>
<td></td>
<td>bool SimpleIOClass::ConfigureIODefaultOutput(unsigned char ucIOMap, unsigned char ucDefValue) Section 2.3.1.3</td>
</tr>
<tr>
<td></td>
<td>bool SimpleIOClass::ConfigureMCP2200 (unsigned char IOMap, unsigned long BaudRateParam, unsigned int RxLEDMode, unsigned int TxLEDMode, bool FLOW, bool ULOAD, bool SSPND) Section 2.3.1.4</td>
</tr>
<tr>
<td></td>
<td>bool SimpleIOClass::fnHardwareFlowControl (unsigned int onOff) Section 2.3.1.5</td>
</tr>
</tbody>
</table>

Note 1: Prior to any DLL API usage, a call to the InitMCP2200() function is needed. This function is the only initialization function in the presented DLL.

Note 2: The configuration only needs to be set a single time – it is stored in NVM.
### 2.3.1.1 ClearPin

Function: 
`bool SimpleIOClass::ClearPin (unsigned int pin)`

**Summary:** Clears the specified pin.

**Description:** Clears the specified pin to logic '0'.

**Precondition:** This pin must be previously configured as an output via a `ConfigureIO` or `ConfigureIoDefaultOutput` call.

**Parameters:**
- `pin` - The pin number to set (0-7).

**Returns:** This function returns True if the transmission is successful and returns False if the transmission fails.

**Remarks:** None

**EXAMPLE 2-1:**

```c
if (SimpleIOClass::ClearPin (2))
{
    lblStatusBar->Text = "Success";
}
else
    lblStatusBar->Text = "Invalid command " + SimpleIOClass::LastError;
```
2.3.1.2 ConfigureIO

Function:
bool SimpleIOC::ConfigureIO (unsigned char IOMap)

Summary: Configures the GPIO pins for Digital Input or Digital Output.
Description: GPIO Pins can be configured as Digital Input or Digital Output.
Precondition: VID and PID must be previously set via a call to InitMCP2200(VID, PID).
Parameters:
IOMap - a byte that represents a bitmap of the GPIO configuration:
• a bit set to '1' will be a digital input
• a bit set to '0' will be a digital output
• MSB    –        –       –       –       –       –      LSB
  GP7  GP6  GP5  GP4  GP3  GP2  GP1  GP0
Returns: This function returns True if the transmission is successful and returns False if the transmission fails.
Remarks: Error code is returned in LastError.

EXAMPLE 2-2:
if (SimpleIOC::ConfigureIO(0xA5) == SUCCESS)
  lblStatusBar->Text = "Success";
else
  lblStatusBar->Text = "Invalid command " + SimpleIOC::LastError;

2.3.1.3 ConfigureIODefaultOutput

Function:
bool SimpleIOC::ConfigureIODefaultOutput (unsigned char ucIoMap, unsigned char ucDefValue)

Summary: Configures the IO pins for Digital Input, Digital Output and also the default output latch value.
Description: IO Pins can be configured as Digital Input or Digital Output. The default output latch value is received as a parameter.
Precondition: VID and PID must be previously set via a call to InitMCP2200(VID, PID).
Parameters:
1. ucIoMap - a byte that represents a bitmap used to set the GPIOs as either input or output.
   • '1' configures GPIO as input
   • '0' configures GPIO as output
   • MSB    –        –       –       –       –       –      LSB
     GP7  GP6  GP5  GP4  GP3  GP2  GP1  GP0
2. ucDefValue - the default value that will be loaded to the output latch (affects only the pins configured as outputs).
Returns: This function returns True if the transmission is successful and returns False if the transmission fails.
Remarks: Error code is returned in LastError.

EXAMPLE 2-3:
if (SimpleIOC::ConfigureIODefaultOutput(IoMap, DefValue) == SUCCESS)
  lblStatusBar->Text = "Success";
else
  lblStatusBar->Text = "Invalid command " + SimpleIOC::LastError;
2.3.1.4 ConfigureMCP2200

Function: bool SimpleIOClass::ConfigureIoDefaultOutput (unsigned long BaudRateParam, unsigned int RxLEDMode, unsigned int TxLEDMode, bool FLOW, bool ULOAD, bool SSPND)

Summary: Configures the device.
Description: Sets the default GPIO designation, baud rate, TX/RX LED modes, flow control.
Precondition: VID and PID must be previously set via a call to InitMCP2200(VID, PID).

Parameters:
1. IOMap - A byte that represents the input/output state of the pins (each bit may be either a ‘1’ for input or ‘0’ for output.
2. BaudRateParam - the default communication baud rate.
3. RxLEDMode - can take one of the constant values (OFF, ON, TOGGLE, BLINKSLOW, BLINKFAST) to define the behavior of the Rx LED.
   • OFF = 0
   • ON = 1
   • TOGGLE = 3
   • BLINKSLOW = 4
   • BLINKFAST = 5
4. TxLEDMode - can take one of the defined values (OFF, ON, TOGGLE, BLINKSLOW, BLINKFAST) in order to define the behavior of the Tx LED.
5. FLOW - this parameter establishes the default flow control method (False - no HW flow control, True - RTS/CTS flow control).
6. ULOAD - this parameter establishes when the USB has loaded the configuration.
7. SSPND - this parameter establishes when the USB sends the Suspend mode signal.
Returns: This function returns True if the transmission is successful and returns False if the transmission fails.
Remarks: None.

EXAMPLE 2-4:

```cpp
if (SimpleIOClass::ConfigureMCP2200(0x43, 9600, BLINKSLOW, BLINKFAST, false, false, false) == SUCCESS)
    lblStatusBar->Text = "Success";
else
    lblStatusBar->Text = "Invalid command ";
```

2.3.1.5 fnHardwareFlowControl

Function: bool SimpleIOClass::fnHardwareFlowControl (unsigned int onOff)

Summary: Configures the flow control of the MCP2200. The flow control configuration will be stored in NVRAM.
Description: Sets the flow control to HW flow control (RTS/CTS) or no flow control.
Precondition: VID and PID must be previously set via a call to InitMCP2200(VID, PID).

Parameters: onOff:
   • ‘1’ if HW flow control is required
   • ‘0’ if no flow control is required
Returns: This function returns True if the transmission is successful and returns False if the transmission fails.
Remarks: Error code is returned in LastError.

EXAMPLE 2-5:

```cpp
if (SimpleIOClass::fnHardwareFlowControl(1) == SUCCESS)
    lblStatusBar->Text = "Success";
else
    lblStatusBar->Text = "Invalid command "+ SimpleIOClass::LastError;
```
2.3.1.6 fnRxLED
Function:
bool SimpleIOClass::fnRxLED (unsigned int mode)

Summary: Configures the Rx LED mode. Rx LED configuration will be stored in NVRAM.
Description: Sets the Rx LED mode to one of the possible values.
Precondition: VID and PID must be previously set via a call to InitMCP2200(VID, PID).
Parameters: 
mode (constant): OFF, TOGGLE, BLINKSLOW, BLINKFAST
Returns: This function returns True if the transmission is successful and returns False if the transmission fails.
Remarks: Error code is returned in LastError.

EXAMPLE 2-6:
```c
define fnRxLED (BLINKFAST) == SUCCESS
   lblStatusBar->Text = "Success";
else
   lblStatusBar->Text = "Invalid command " + SimpleIOClass::LastError;
```

2.3.1.7 fnSetBaudRate
Function:
bool SimpleIOClass::fnSetBaudRate (unsigned long BaudRateParam)

Summary: Configures the device’s default baud rate. The baud rate value will be stored in NVRAM.
Description: Sets the desired baud rate and it stores it into the device’s NVRAM.
Precondition: VID and PID must be previously set via a call to InitMCP2200(VID, PID).
Parameters: 
BaudRateParam - the desired baud rate value
Returns: This function returns True if the transmission is successful and returns False if the transmission fails.
Remarks: Error code is returned in LastError. This function is used only to set the default power-up baud rate
value. When used with a terminal program, there is no need to call this function to change the baud rate. Changing the baud rate from the terminal program will send the appropriate CDC packet that
will change the communication’s baud rate without the need to call this function.

EXAMPLE 2-7:
```c
define fnSetBaudRate(9600) == SUCCESS
   lblStatusBar->Text = "Success";
else
   lblStatusBar->Text = "Invalid command " + SimpleIOClass::LastError;
```

2.3.1.8 fnSuspend
Function:
bool SimpleIOClass::fnSuspend (unsigned int onOff)

Summary: Configures the GP0 pin of the MCP2200 to show the status of the USB Suspend/Resume states.
Description: When the GP0 is designated to show the USB Suspend/Resume states, the pin will go ‘low’ when the
Suspend state is issued, or will go ‘high’ when the Resume state is on.
Precondition: VID and PID must be previously set via a call to InitMCP2200(VID, PID).
Parameters: 
  onOff:
• ‘1’ GP0 will reflect the USB Suspend/Resume states
• ‘0’ GP0 will not reflect the USB Suspend/Resume states (can be used as GPIO)
Returns: This function returns True if the transmission is successful and returns False if the transmission fails.
Remarks: Error code is returned in LastError.
EXAMPLE 2-8:

```cpp
if (SimpleIOClass::fnSuspend(1) == SUCCESS)
    lblStatusBar->Text = "Success";
else
    lblStatusBar->Text = "Invalid command “ + SimpleIOClass::LastError;
```

2.3.1.9 fnTxLED

**Function:**
bool SimpleIOClass::fnTxLED (unsigned int mode)

**Summary:** Configures the Tx LED mode. Tx LED configuration will be stored in NVRAM.

**Description:** Sets the Tx LED mode to one of the possible values.

**Precondition:** VID and PID must be previously set via a call to InitMCP2200(VID, PID).

**Parameters:**
- `mode` (constant): OFF, TOGGLE, BLINKSLOW, BLINKFAST

**Returns:** This function returns True if the transmission is successful and returns False if the transmission fails.

**Remarks:** Error code is returned in LastError.

EXAMPLE 2-9:

```cpp
if (SimpleIOClass::fnTxLED (BLINKSLOW) == SUCCESS)
    lblStatusBar->Text = "Success";
else
    lblStatusBar->Text = "Invalid command “ + SimpleIOClass::LastError;
```

2.3.1.10 fnULoad

**Function:**
bool SimpleIOClass::fnULoad (unsigned int onOff)

**Summary:** Configures the GP1 pin of the MCP2200 to show the configuration status of the USB.

**Description:** When the GP1 is designated to show the USB configuration status, the pin will start ‘low’ (during power-up or after Reset), and it will go ‘high’ after the MCP2200 is successfully configured by the host.

**Precondition:** VID and PID must be previously set via a call to InitMCP2200(VID, PID).

**Parameters:**
- `onOff`:
  - ‘1’ GP1 will reflect the USB configuration status
  - ‘0’ GP1 will not reflect the USB configuration status (can be used as GPIO)

**Returns:** This function returns True if the transmission is successful and returns False if the transmission fails.

**Remarks:** Error code is returned in LastError.

EXAMPLE 2-10:

```cpp
if (SimpleIOClass::fnULoad(1) == SUCCESS)
    lblStatusBar->Text = "Success";
else
    lblStatusBar->Text = "Invalid command “ + SimpleIOClass::LastError;
```
2.3.1.11 GetDeviceInfo

Function:
String^ SimpleIOClass::GetDeviceInfo (unsigned int uiDeviceNo)

Summary: Returns the path name for one of the connected devices.
Description: The function will return the path name for the given device ID.
Precondition: At least one call to the InitMCP2200() is required in order to initiate a DLL search for the compatible devices. VID and PID must be previously set via a call to InitMCP2200(VID, PID).
Parameters: uiDeviceNo: The device ID for which the path information is needed. Can have a value between 0 and the number of devices minus 1.
Returns: This function returns a string containing the path name of the given device id.
- In the case the given ID is out of range, the function will return the "Device Index Error" string.
- In the case the device for which the path name is required is not connected anymore, the return string will be "Device Not Connected".
Remarks: None.

EXAMPLE 2-11:
        lblStatusBar->Text = SimpleIOClass::GetDeviceInfo(0);

2.3.1.12 GetNoOfDevices

Function:
unsigned int SimpleIOClass::GetNoOfDevices(void)

Summary: The function returns the number of available devices present in the system.
Description: The function returns the number of HID devices (with the given VID/PID) connected to the system.
Precondition: At least one call to the InitMCP2200() is required in order to initiate a DLL search for the compatible devices. Also, in order to know the actual number of devices connected to the system, call the SimpleIOClass::IsConnected() function. VID and PID must be previously set via a call to InitMCP2200(VID, PID).
Parameters: None.
Returns: This function returns the number of HID devices with the given VID/PID (as parameters of the SimpleIOClass::InitMCP2200() function).
Remarks: Call the SimpleIOClass::IsConnected() function prior to the call of this function in order to have the most recent number of devices that are present in the system.

EXAMPLE 2-12:
        SimpleIOClass::IsConnected(); //call this function to refresh the number of
        //the devices present in the system
        lblStatusBar->Text = SimpleIOClass::GetNoOfDevices();

2.3.1.13 GetSelectedDevice

Function:
int SimpleIOClass::GetSelectedDevice(void)

Summary: Returns the ID of the selected device.
Description: The function returns the ID of the current selected device.
Precondition: At least one call to the InitMCP2200() is required in order to initiate a DLL search for the compatible devices. VID and PID must be previously set via a call to InitMCP2200(VID, PID).
Parameters: None.
Returns: This function returns the ID of the current selected device. Its value can range from 0 to the number of devices minus 1.
Remarks: None.
EXAMPLE 2-13:

```c
lblStatusBar->Text = SimpleIOClass::GetSelectedDevice();
```

2.3.1.14 GetSelectedDeviceInfo

Function:
`String^ SimpleIOClass::GetSelectedDeviceInfo(void)`

Summary: Returns the selected device path name.
Description: The function returns a string containing the unique path name of the selected device.
Precondition: At least one call to the `InitMCP2200()` is required in order to initiate a DLL search for the compatible devices. VID and PID must be previously set via a call to `InitMCP2200(VID, PID)`.
Parameters: None.
Returns: This function returns a string containing the unique path name of the selected device.
Remarks: The default selected device is the first one that the DLL finds. If the user wants to retrieve other devices path names (assuming more than one device is present in the system), a call to `SimpleIOClass::SelectDevice(deviceNo)` is required.

EXAMPLE 2-14:

```c
lblStatusBar->Text = SimpleIOClass::GetSelectedDeviceInfo(void);
```

2.3.1.15 InitMCP2200

Function:
`void SimpleIOClass::InitMCP2200 (unsigned int VendorID, unsigned int ProductID)`

Summary: Configures the Simple IO class for a specific Vendor and Product ID.
Description: Sets the Vendor and Product ID used for the project.
Precondition: None.
Parameters: 1. Vendor ID - assigned by USB IF (www.usb.org)
2. Product ID - assigned by the Vendor ID Holder
Returns: None.
Remarks: Call this function before any other calls, to set the Vendor and Product IDs.

EXAMPLE 2-15:

```c
InitMCP2200 (0x4D8, 0x00DF);
```

2.3.1.16 IsConnected

Function:
`bool SimpleIOClass::IsConnected()`

Summary: Checks with the OS if the current VID/PID device is connected.
Description: Checks if a MCP2200 device is connected to the computer. If so, it returns True; otherwise, the result will be False.
Precondition: VID and PID must be previously set via a call to `InitMCP2200(VID, PID)`.
Parameters: None.
Returns: True - if at least one device is connected to the host.
         False - if there are no devices connected to the host.
Remarks: No actual communication with the end device is occurring. The function inquires the OS if the specified VID/PID was enumerated.
EXAMPLE 2-16:

```cpp
unsigned int rv;
if (SimpleIOClass::IsConnected ())
{
    lblStatusBar->Text = “Device connected”;
}
else
    lblStatusBar->Text = “Device Disconnected”;`n```

2.3.1.17 ReadEEPROM

Function:
```cpp
int SimpleIOClass::ReadEEPROM (unsigned int uiEEPAddress)
```

Summary: Reads a byte from the EEPROM.

Description: Reads a byte from the EEPROM at the given address.

Precondition: At least one call to the InitMCP2200() is required in order to initiate a DLL search for the compatible devices. VID and PID must be previously set via a call to InitMCP2200(VID, PID).

Parameters:
- `uiEEPAddress`: the EEPROM address location we need to write to (must be from 0 to 255, inclusively).

Returns:
- This function returns any positive value as being the EEPROM’s location value:
  - `E_WRONG_ADDRESS (-3)` - in case the given EEPROM address is out of range
  - `E_CANNOT_SEND_DATA (-4)` - in case the function cannot send the command to the device

Remarks: None.

EXAMPLE 2-17:

```cpp
int iRetValue = SimpleIOClass::ReadEEPROM(0x01);
if (iRetValue >= 0)
{
    lblStatusBar->Text = “Success”;
}
else
    lblStatusBar->Text = “Error reading to EEPROM” + SimpleIOClass::LastError;`n```

2.3.1.18 ReadPin

Function:
```cpp
bool SimpleIOClass::ReadPin (unsigned int pin, unsigned int *returnvalue)
```

Summary: Reads the specified pin.

Description: Reads the specified pin and returns the value in returnvalue. If the pin has been configured as a digital input, the return value will be either ‘0’ or ‘1’.

Precondition: Must be previously configured as an input via a ConfigureIO call. VID and PID must be previously set via a call to InitMCP2200(VID, PID).

Parameters:
- `pin`: the pin number to set (0-7)
- `returnvalue`: the value read on the pin (‘0’ or ‘1’)

Returns:
- This function returns True if the transmission is successful and returns False if the transmission fails.

Remarks: None.

EXAMPLE 2-18:

```cpp
unsigned int rv;
if (SimpleIOClass::ReadGPIOn (0, &rv))
{
    lblStatusBar->Text = “Success”;
}
else
    lblStatusBar->Text = “Invalid command ” + SimpleIOClass::LastError;`n```
2.3.1.19 ReadPinValue

Function:
int SimpleIOClass::ReadPinValue(unsigned int pin)

Summary: Reads the specified pin.
Description: Reads the specified pin and returns the value as the return value. If the pin has been configured as a
digital input, the return value will be either '0' or '1'. If an error occurs, the function will return a value of 0x8000.
Precondition: Must be previously configured as an input via a ConfigureIO call.
VID and PID must be previously set via a call to InitMCP2200(VID, PID).
Parameters: pin - the pin number to set (0-7)
Returns: This function returns the read value of the pin, or returns a value of 0x8000, if an error occurs.
Remarks: None.

EXAMPLE 2-19:

unsigned int rv;
if (SimpleIOClass::ReadPinValue(0) != 0x8000)
    {
        lblStatusBar->Text = "Success";
    }
else
    lblStatusBar->Text = "Invalid command " + SimpleIOClass::LastError;

2.3.1.20 ReadPort

Function:
bool SimpleIOClass::ReadPort(unsigned int *returnvalue)

Summary: Reads the GPIO port as digital input.
Description: Reads the GPIO port and returns the value in returnvalue. This provides a means to read all pins
simultaneously, instead of one-by-one.
Precondition: Must be previously configured as an input via a ConfigureIO call.
VID and PID must be previously set via a call to InitMCP2200(VID, PID).
Parameters: • pin - the pin number to set (0-7)
        • returnvalue - the value read on the pin ('0' or '1')
Returns: This function returns True if the transmission is successful and returns False if the transmission fails.
Remarks: Pins configured for output return the current state of the port. Pins configured as input read as zero.

EXAMPLE 2-20:

unsigned int rv;
if (SimpleIOClass::ReadGPIOPort (0, &rv))
    {
        lblStatusBar->Text = "Success";
    }
else
    lblStatusBar->Text = "Invalid command " + SimpleIOClass::LastError;
2.3.1.21 ReadPortValue

Function:
int SimpleIOClass::ReadPortValue()

Summary: Reads the GPIO port as digital input.
Description: Reads the GPIO port and returns the value of the port. This provides a method to read all pins simultaneously, instead of one-by-one. In case of an error, the returned value will be 0x8000.
Precondition: Must be previously configured as an input via a ConfigureIO call. VID and PID must be previously set via a call to InitMCP2200(VID, PID).
Parameters: None.
Returns: This function returns True if the transmission is successful and returns False if the transmission fails.
Remarks: Pins configured for output return the current state of the port. Pins configured as input read as zero.

EXAMPLE 2-21:

```cpp
int rv;
rv = SimpleIOClass::ReadPortValue();
if (rv != 0x8000)
{
    lblStatusBar->Text = "Success";
}
else
    lblStatusBar->Text = "Invalid command " + SimpleIOClass::LastError;
```

2.3.1.22 SelectDevice

Function:
int SimpleIOClass::SelectDevice(unsigned int uiDeviceNo)

Summary: Selects one of the active devices in the system.
Description: The function is used to select one of the detected devices in the system as the “active device”.
Precondition: At least one call to the InitMCP2200() is required in order to initiate a DLL search for the compatible devices. Also, in order to know the actual number of devices in the system, call the SimpleIOClass::IsConnected() function. VID and PID must be previously set via a call to InitMCP2200(VID, PID).
Parameters: uiDeviceNo - the ID of the device to be selected (can have a value between 0 and the number of devices minus 1).
Returns: This function returns '0' in case of selection success, otherwise it will return:
• E_WRONG_DEVICE_ID (-1) for a device ID that is out of range
• E_INACTIVE_DEVICE (-2) for an inactive device.
Remarks: Call the SimpleIOClass::IsConnected() prior to the call of this function in order to have the most recent number of devices that are present in the system.

EXAMPLE 2-22:

```cpp
int iResult;
iResult = SimpleIOClass::SelectDevice(1);
if (iResult == 0)
{
    lblStatusBar->Text = "Success";
}
else
   lblStatusBar->Text = "Error selecting device";
```
2.3.1.23  SetPin

Function:
bool SimpleIOClass::SetPin(unsigned int pin)

Summary: Sets the specified pin.
Description: Sets the specified pin to logic '1'.
Precondition: Must be previously configured as an output via a ConfigureIO or ConfigureIoDefaultOutput call.
VID and PID must be previously set via a call to InitMCP2200 (VID, PID).
Parameters: pin - the pin number to set (0-7)
Returns: This function returns True if the transmission is successful and returns False if the transmission fails.
Remarks: None.

EXAMPLE 2-23:

```cpp
if (SimpleIOClass::SetPin (2))
{
    lblStatusBar->Text = "Success";
}
else
    lblStatusBar->Text = "Invalid command " + SimpleIOClass::LastError;
```

2.3.1.24  WriteEEPROM

Function:
int SimpleIOClass::WriteEEPROM(unsigned int uiEEPAddress, unsigned char ucValue)

Summary: Writes a byte into the MCP2200 device’s EEPROM.
Description: Writes a byte at the given address into the internal 256 bytes EEPROM.
Precondition: At least one call to the InitMCP2200() is required in order to initiate a DLL search for the compatible devices. VID and PID must be previously set via a call to InitMCP2200 (VID, PID).
Parameters:  • uiEEPAddress - the EEPROM address location to write to (must be from 0 to 255 inclusively).
             • ucValue - the byte value required for writing to the given location.
Returns: This function returns '0' if the write command was successfully sent to the device, otherwise it returns:
             • E_WRONG_ADDRESS (-3) in case the given EEPROM address is out of range
             • E_CANNOT_SEND_DATA (-4) in case the function cannot send the command to the device.
Remarks: The function will send the write EEPROM command, but has no confirmation whether the EEPROM location was actually written. In order to verify the correctness of the EEPROM write, the user can issue a SimpleIOClass::ReadEEPROM() and check if the returned value matches the written one.

EXAMPLE 2-24:

```cpp
int iRetValue = SimpleIOClass::WriteEEPROM(0x01, 0xAB);
if (iRetValue == 0)
{
    lblStatusBar->Text = "Success";
}
else
    lblStatusBar->Text = "Error writing to EEPROM" + SimpleIOClass::LastError;
```
2.3.1.25 WritePort
Function:
bool SimpleIOClass::WritePort(unsigned int portValue)

Summary: Writes a value to the GPIO port.
Description: Writes the GPIO port. This provides a means to write all pins simultaneously, instead of one-by-one.
Precondition: Must be previously configured as an output via a ConfigureIO call. VID and PID must be previously set via a call to InitMCP2200(VID, PID).
Parameters: portValue - byte value to set on the port.
Returns: This function returns True if the transmission is successful and returns False if the transmission fails.
Remarks: None.

EXAMPLE 2-25:

```
if (SimpleIOClass::WritePort (0x5A))
{
    lblStatusBar->Text = "Success";
}
else
    lblStatusBar->Text = "Invalid command " + SimpleIOClass::LastError;
```
3.0 Electrical Characteristics

Absolute Maximum Ratings (†)

Ambient temperature under bias ........................................................................................................ –40°C to +85°C
Storage temperature ....................................................................................................................... –65°C to +150°C
Voltage on VDD with respect to VSS .................................................................................................. –0.3V to +6.0V
Voltage on MCLR with respect to Vss ................................................................................................ –0.3V to +9.0V
Voltage on VUSB(1) pin with respect to VSS ....................................................................................... –0.3V to +4.0V
Voltage on D+ and D- pins with respect to VSS ..................................................................... –0.3V to (VUSB + 0.3V)
Voltage on all other pins with respect to VSS ........................................................................... –0.3V to (VDD + 0.3V)
Total power dissipation(2) ....................................................................................................................... 800 mW
Maximum current out of VSS pin .............................................................................................................................. 95 mA
Maximum current into VDD pin ......................................................................................................................... 95 mA
Clamp current, IK (VPIN < 0 or VPIN > VDD) ...................................................................................... ± 20 mA
Maximum output current sunk by any I/O pin.................................................................................... 25 mA
Maximum output current sourced by any I/O pin ........................................................................ 25 mA
Maximum current sunk by all ports .................................................................................................................... 90 mA
Maximum current sourced by all ports ................................................................................................. 90 mA

Note 1: VUSB must always be ≤ VDD + 0.3V.
2: Power dissipation is calculated as follows: PDIS = VDD x {IDD – Σ IOH} + Σ {(VDD – VOH) x IOH} + Σ(VOL x IOL).

† NOTICE: Stresses above those listed under “Absolute Maximum Ratings” may cause permanent damage to the device. This is a stress rating only and functional operation of the device at those or any other conditions above those indicated in the operation listings of this specification is not implied. Exposure above maximum rating conditions for extended periods may affect device reliability.
### 3.1 DC Characteristics

<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>D001</td>
<td>Supply Voltage</td>
<td>VDD</td>
<td>3.0</td>
<td></td>
<td>5.5</td>
<td>V</td>
<td></td>
</tr>
<tr>
<td>D002</td>
<td>Power-on Reset Release Voltage</td>
<td>VPOR</td>
<td></td>
<td>1.6</td>
<td></td>
<td>V</td>
<td></td>
</tr>
<tr>
<td>D003</td>
<td>Power-on Reset Rearm Voltage</td>
<td>VDD</td>
<td></td>
<td>0.8</td>
<td></td>
<td>V</td>
<td></td>
</tr>
<tr>
<td>D003</td>
<td>VDD Rise Rate to Ensure Power-on Reset</td>
<td>SVDD</td>
<td>0.05</td>
<td></td>
<td></td>
<td>V/ms</td>
<td>Design guidance only, Not tested</td>
</tr>
<tr>
<td>D004</td>
<td>Supply Current</td>
<td>IDD</td>
<td></td>
<td></td>
<td></td>
<td>mA</td>
<td></td>
</tr>
<tr>
<td></td>
<td>VDD = 3.0V</td>
<td>—</td>
<td>10</td>
<td>12</td>
<td></td>
<td>mA</td>
<td>Fosc = 12 MHz, (330 nF on VUSB)</td>
</tr>
<tr>
<td></td>
<td>VDD = 5.0V</td>
<td>—</td>
<td>13</td>
<td>15</td>
<td></td>
<td>mA</td>
<td></td>
</tr>
<tr>
<td>D005</td>
<td>Standby Current</td>
<td>IDDS</td>
<td>—</td>
<td>9</td>
<td></td>
<td>µA</td>
<td></td>
</tr>
</tbody>
</table>

#### Input Low-Voltage

<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>D031</td>
<td>Schmitt Trigger (GPIO TTL (CTS pin))</td>
<td>VIL</td>
<td>—</td>
<td>—</td>
<td>0.2</td>
<td>Vdd</td>
<td>3.0V ≤ VDD ≤ 5.5V</td>
</tr>
<tr>
<td>D041</td>
<td>Schmitt Trigger (GPIO TTL (RTS pin))</td>
<td>VIH</td>
<td>0.8</td>
<td>Vdd</td>
<td></td>
<td>V</td>
<td>3.0V ≤ VDD ≤ 5.5V</td>
</tr>
</tbody>
</table>

#### Input High-Voltage

<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>D060</td>
<td>GPIO, CTS RST</td>
<td>IIL</td>
<td>—</td>
<td>±50</td>
<td>±100</td>
<td>nA</td>
<td>VSS ≤ VPIN ≤ VDD, pin at high Z</td>
</tr>
<tr>
<td></td>
<td>OSC1</td>
<td>—</td>
<td>±50</td>
<td>±200</td>
<td>nA</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

#### Output Low-Voltage

<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>D080</td>
<td>GPIO, UART Tx/Rx</td>
<td>VOL</td>
<td>—</td>
<td>—</td>
<td>0.6</td>
<td>V</td>
<td>IOL = 8.0 mA, VDD = 5.0V</td>
</tr>
<tr>
<td></td>
<td></td>
<td>—</td>
<td>—</td>
<td>0.6</td>
<td>V</td>
<td>IOL = 6.0 mA, VDD = 3.3V</td>
<td></td>
</tr>
</tbody>
</table>

#### Output High-Voltage

<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>D090</td>
<td>GPIO, UART Tx/Rx</td>
<td>VOH</td>
<td>VDD – 0.7</td>
<td>—</td>
<td>—</td>
<td>V</td>
<td>IOL = -3.5 mA, VDD = 5.0V</td>
</tr>
<tr>
<td></td>
<td></td>
<td>VDD – 0.7</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>V</td>
<td>IOL = -3.0 mA, VDD = 3.3V</td>
</tr>
</tbody>
</table>

#### Capacitive Loading Specifications on Output Pins

<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>D101</td>
<td>OSC2</td>
<td>—</td>
<td>—</td>
<td>15</td>
<td></td>
<td>pF</td>
<td></td>
</tr>
<tr>
<td>D102</td>
<td>GPIO</td>
<td>—</td>
<td>—</td>
<td>50</td>
<td></td>
<td>pF</td>
<td></td>
</tr>
</tbody>
</table>

**Note 1:** This parameter is characterized, but not tested.
FIGURE 3-1: POR AND POR REARM WITH SLOW RISING VDD

Note 1: When NPOR is low, the device is held in Reset.
2: TPOR 1 µs typical.
3: TVLOW 2.7 µs typical.

TABLE 3-1: USB MODULE SPECIFICATIONS

<table>
<thead>
<tr>
<th>DC Characteristics</th>
<th>Operating Conditions (unless otherwise indicated): 3.0V ≤ VDD ≤ 5.5V at −40°C ≤ TA ≤ +85°C (I-Temp)</th>
</tr>
</thead>
<tbody>
<tr>
<td>D313</td>
<td>USB Voltage</td>
</tr>
<tr>
<td>D314</td>
<td>Input Leakage on Pin</td>
</tr>
<tr>
<td>D315</td>
<td>Input Low Voltage for USB Buffer</td>
</tr>
<tr>
<td>D316</td>
<td>Input High Voltage for USB Buffer</td>
</tr>
<tr>
<td>D318</td>
<td>Differential Input Sensitivity</td>
</tr>
<tr>
<td>D319</td>
<td>Differential Common Mode Range</td>
</tr>
<tr>
<td>D320</td>
<td>Driver Output Impedance(1)</td>
</tr>
<tr>
<td>D321</td>
<td>Voltage Output Low</td>
</tr>
<tr>
<td>D322</td>
<td>Voltage Output High</td>
</tr>
</tbody>
</table>

Note 1: The D+ and D- signal lines have been built-in impedance matching resistors. No external resistors, capacitors or magnetic components are necessary on the D+/D- signal paths between the MCP2200 family device and the USB cable.
## TABLE 3-2: THERMAL CONSIDERATIONS

**Standard Operating Conditions (unless otherwise stated)**

Operating temperature: \(-40^\circ C \leq T_A \leq +85^\circ C\) (I-Temp)

<table>
<thead>
<tr>
<th>Param. No.</th>
<th>Sym.</th>
<th>Characteristic</th>
<th>Typ.</th>
<th>Units</th>
<th>Conditions</th>
</tr>
</thead>
<tbody>
<tr>
<td>TH01</td>
<td>(\theta_{JA})</td>
<td>Thermal Resistance Junction to Ambient</td>
<td>36.1</td>
<td>(^\circ C/W)</td>
<td>20-pin VQFN 5x5 mm package</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>85.2</td>
<td>(^\circ C/W)</td>
<td>20-pin SOIC package</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>108.1</td>
<td>(^\circ C/W)</td>
<td>20-pin SSOP package</td>
</tr>
<tr>
<td>TH02</td>
<td>(\theta_{JC})</td>
<td>Thermal Resistance Junction to Case</td>
<td>1.7</td>
<td>(^\circ C/W)</td>
<td>20-pin VQFN 5x5 mm package</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>24</td>
<td>(^\circ C/W)</td>
<td>20-pin SOIC package</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>24</td>
<td>(^\circ C/W)</td>
<td>20-pin SSOP package</td>
</tr>
<tr>
<td>TH03</td>
<td>(T_{JMAX})</td>
<td>Maximum Junction Temperature</td>
<td>150</td>
<td>(^\circ C)</td>
<td></td>
</tr>
<tr>
<td>TH04</td>
<td>PD</td>
<td>Power Dissipation</td>
<td>—</td>
<td>W</td>
<td>(P_D = P_{INTERNAL} + P_{I/O})</td>
</tr>
<tr>
<td>TH05</td>
<td>(P_{INTERNAL})</td>
<td>Internal Power Dissipation</td>
<td>—</td>
<td>W</td>
<td>(P_{INTERNAL} = I_{DD} \times V_{DD}(1))</td>
</tr>
<tr>
<td>TH06</td>
<td>(P_{I/O})</td>
<td>I/O Power Dissipation</td>
<td>—</td>
<td>W</td>
<td>(P_{I/O} = \Sigma (I_{OL} \times V_{OL}) + \Sigma (I_{OH} \times (V_{DD} - V_{OH})))</td>
</tr>
<tr>
<td>TH07</td>
<td>PDER</td>
<td>Derated Power</td>
<td>—</td>
<td>W</td>
<td>(P_{DER} = P_{DMAX} \times (T_J - T_A)/\theta_{JA}(2,3))</td>
</tr>
</tbody>
</table>

**Note 1:** \(I_{DD}\) is the current to run the chip alone without driving any load on the output pins.

**Note 2:** \(T_A = \) Ambient Temperature.

**Note 3:** \(T_J = \) Junction Temperature.
3.2 AC Characteristics

3.2.1 TIMING PARAMETER SYMBOLOGY
The timing parameter symbols have been created in one of the following formats:

<table>
<thead>
<tr>
<th>TppS2ppS</th>
<th>TppS</th>
</tr>
</thead>
<tbody>
<tr>
<td>T</td>
<td>T</td>
</tr>
<tr>
<td>F</td>
<td>Frequency</td>
</tr>
<tr>
<td>E</td>
<td>Error</td>
</tr>
</tbody>
</table>

Lowercase letters (pp) and their meanings:

<table>
<thead>
<tr>
<th>pp</th>
<th>Meaning</th>
</tr>
</thead>
<tbody>
<tr>
<td>io</td>
<td>Input or Output pin</td>
</tr>
<tr>
<td>rx</td>
<td>Receive</td>
</tr>
<tr>
<td>bitclk</td>
<td>RX/TX BITCLK</td>
</tr>
<tr>
<td>drt</td>
<td>Device Reset Timer</td>
</tr>
</tbody>
</table>

Uppercase letters and their meanings:

<table>
<thead>
<tr>
<th>S</th>
<th>Meaning</th>
</tr>
</thead>
<tbody>
<tr>
<td>F</td>
<td>Fall</td>
</tr>
<tr>
<td>H</td>
<td>High</td>
</tr>
<tr>
<td>I</td>
<td>Invalid (high-impedance)</td>
</tr>
<tr>
<td>L</td>
<td>Low</td>
</tr>
<tr>
<td>P</td>
<td>Period</td>
</tr>
<tr>
<td>R</td>
<td>Rise</td>
</tr>
<tr>
<td>V</td>
<td>Valid</td>
</tr>
<tr>
<td>Z</td>
<td>High-impedance</td>
</tr>
</tbody>
</table>

3.2.2 TIMING CONDITIONS
The operating temperature and voltage specified in Table 3-3 apply to all timing specifications unless otherwise noted. Figure 3-2 specifies the load conditions for the timing specifications.

TABLE 3-3: TEMPERATURE AND VOLTAGE SPECIFICATIONS - AC

<table>
<thead>
<tr>
<th>AC CHARACTERISTICS</th>
<th>Standard Operating Conditions (unless otherwise stated)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Operating temperature:</td>
<td>(-40^\circ C \leq TA \leq +85^\circ C)</td>
</tr>
<tr>
<td>Operating voltage (V_{DD}) range as described in DC spec, Section 3.1 “DC Characteristics”</td>
<td></td>
</tr>
</tbody>
</table>

FIGURE 3-2: LOAD CONDITIONS FOR DEVICE TIMING SPECIFICATIONS

Pin

\[50\ pF\ (15\ pF\ for\ \text{OSC2})\]
### 3.2.3 TIMING SPECIFICATIONS

**TABLE 3-4: RESET, OSCILLATOR START-UP TIMER AND POWER-UP TIMER PARAMETERS**

<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>30</td>
<td>TRST</td>
<td>MCLR Pulse Width (low)</td>
<td>2</td>
<td>—</td>
<td>—</td>
<td>μs</td>
<td></td>
</tr>
<tr>
<td>31</td>
<td>TPWRT</td>
<td>Power-Up Timer</td>
<td>40</td>
<td>65</td>
<td>140</td>
<td>ms</td>
<td></td>
</tr>
<tr>
<td>32</td>
<td>TOST</td>
<td>Oscillator Start-Up Time</td>
<td>—</td>
<td>1024</td>
<td>—</td>
<td>TOST</td>
<td></td>
</tr>
</tbody>
</table>

**Note 1:** These parameters are characterized but not tested.
**Note 2:** Data in “Typ.” column is at 5V, 25°C unless otherwise stated. These parameters are for design guidance only and are not tested.
4.0 PACKAGING INFORMATION

4.1 Package Marking Information

Legend:

XX...X  Customer-specific information
Y  Year code (last digit of calendar year)
YY  Year code (last 2 digits of calendar year)
WW  Week code (week of January 1 is week '01')
NNN  Alphanumeric traceability code
@3  Pb-free JEDEC® designator for Matte Tin (Sn)
*  This package is Pb-free. The Pb-free JEDEC® designator @3 can be found on the outer packaging for this package.

Note: In the event the full Microchip part number cannot be marked on one line, it will be carried over to the next line, thus limiting the number of available characters for customer-specific information.
20-Lead Plastic Quad Flat, No Lead Package (MQ) – 5x5x1.0 mm Body [VQFN] With 0.40 mm Contact Length

Note: For the most current package drawings, please see the Microchip Packaging Specification located at http://www.microchip.com/packaging
20-Lead Plastic Quad Flat, No Lead Package (MQ) – 5x5x1.0 mm Body [VQFN] With 0.40 mm Contact Length

### Notes:
- For the most current package drawings, please see the Microchip Packaging Specification located at [http://www.microchip.com/packaging](http://www.microchip.com/packaging)

### Units

<table>
<thead>
<tr>
<th>Dimension Limits</th>
<th>MIN</th>
<th>NOM</th>
<th>MAX</th>
</tr>
</thead>
<tbody>
<tr>
<td>Number of Terminals</td>
<td>N</td>
<td>20</td>
<td></td>
</tr>
<tr>
<td>Pitch</td>
<td>e</td>
<td>0.65 BSC</td>
<td></td>
</tr>
<tr>
<td>Overall Height</td>
<td>A</td>
<td>0.80</td>
<td>0.90</td>
</tr>
<tr>
<td>Standoff</td>
<td>A1</td>
<td>0.00</td>
<td>0.02</td>
</tr>
<tr>
<td>Contact Thickness</td>
<td>(A3)</td>
<td>0.20 REF</td>
<td></td>
</tr>
<tr>
<td>Overall Length</td>
<td>D</td>
<td>5.00 BSC</td>
<td></td>
</tr>
<tr>
<td>Exposed Pad Length</td>
<td>D2</td>
<td>3.15</td>
<td>3.25</td>
</tr>
<tr>
<td>Overall Width</td>
<td>E</td>
<td>5.00 BSC</td>
<td></td>
</tr>
<tr>
<td>Exposed Pad Width</td>
<td>E2</td>
<td>3.15</td>
<td>3.25</td>
</tr>
<tr>
<td>Contact Width</td>
<td>b</td>
<td>0.25</td>
<td>0.30</td>
</tr>
<tr>
<td>Contact Length</td>
<td>L</td>
<td>0.35</td>
<td>0.40</td>
</tr>
<tr>
<td>Contact-to-Exposed Pad</td>
<td>K</td>
<td>0.20</td>
<td>-</td>
</tr>
</tbody>
</table>

### Notes:
1. Pin 1 visual index feature may vary, but must be located within the hatched area.
2. Package is saw singulated
3. Dimensioning and tolerancing per ASME Y14.5M
   - BSC: Basic Dimension. Theoretically exact value shown without tolerances.
   - REF: Reference Dimension, usually without tolerance, for information purposes only.
20-Lead Plastic Quad Flat, No Lead Package (MQ) – 5x5x1.0 mm Body [VQFN] With 0.40 mm Contact Length

Note: For the most current package drawings, please see the Microchip Packaging Specification located at http://www.microchip.com/packaging

RECOMMENDED LAND PATTERN

<table>
<thead>
<tr>
<th>Units</th>
<th>MILLIMETERS</th>
</tr>
</thead>
<tbody>
<tr>
<td>Dimension Limits</td>
<td>MIN NOM MAX</td>
</tr>
<tr>
<td>Contact Pitch</td>
<td>E 0.65 BSC</td>
</tr>
<tr>
<td>Optional Center Pad Width</td>
<td>W2 3.35</td>
</tr>
<tr>
<td>Optional Center Pad Length</td>
<td>T2 3.35</td>
</tr>
<tr>
<td>Contact Pad Spacing</td>
<td>C1 4.50</td>
</tr>
<tr>
<td>Contact Pad Spacing</td>
<td>C2 4.50</td>
</tr>
<tr>
<td>Contact Pad Width (X20)</td>
<td>X1 0.40</td>
</tr>
<tr>
<td>Contact Pad Length (X20)</td>
<td>Y1 0.55</td>
</tr>
<tr>
<td>Distance Between Pads</td>
<td>G 0.20</td>
</tr>
<tr>
<td>Thermal Via Diameter</td>
<td>V 0.30</td>
</tr>
<tr>
<td>Thermal Via Pitch</td>
<td>EV 1.00</td>
</tr>
</tbody>
</table>

Notes:
1. Dimensioning and tolerancing per ASME Y14.5M
   BSC: Basic Dimension. Theoretically exact value shown without tolerances.
2. For best soldering results, thermal vias, if used, should be filled or tented to avoid solder loss during reflow process.

Microchip Technology Drawing C04-2139B (MQ)
### 20-Lead Plastic Small Outline (SO) - Wide, 7.50 mm Body [SOIC]

**Note:** For the most current package drawings, please see the Microchip Packaging Specification located at http://www.microchip.com/packaging

![Diagram of MCP2200 package](image)

<table>
<thead>
<tr>
<th>Units</th>
<th>MILLIMETERS</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Dimension Limits</strong></td>
<td><strong>MIN</strong></td>
</tr>
<tr>
<td>Number of Pins</td>
<td>N</td>
</tr>
<tr>
<td>Pitch</td>
<td>e</td>
</tr>
<tr>
<td>Overall Height</td>
<td>A</td>
</tr>
<tr>
<td>Molded Package Thickness</td>
<td>A2</td>
</tr>
<tr>
<td>Standoff</td>
<td>§</td>
</tr>
<tr>
<td>Overall Width</td>
<td>E</td>
</tr>
<tr>
<td>Molded Package Width</td>
<td>E1</td>
</tr>
<tr>
<td>Overall Length</td>
<td>D</td>
</tr>
<tr>
<td>Chamfer (Optional)</td>
<td>h</td>
</tr>
<tr>
<td>Foot Length</td>
<td>L</td>
</tr>
<tr>
<td>Footprint</td>
<td>L1</td>
</tr>
<tr>
<td>Lead Angle</td>
<td>φ</td>
</tr>
<tr>
<td>Foot Angle</td>
<td>φ</td>
</tr>
<tr>
<td>Lead Thickness</td>
<td>c</td>
</tr>
<tr>
<td>Lead Width</td>
<td>b</td>
</tr>
<tr>
<td>Mold Draft Angle Top</td>
<td>α</td>
</tr>
<tr>
<td>Mold Draft Angle Bottom</td>
<td>β</td>
</tr>
</tbody>
</table>

**Notes:**

1. Pin 1 visual index feature may vary, but must be located within the hatched area.
2. § Significant Characteristic
3. Dimension D does not include mold flash, protrusions or gate burrs, which shall not exceed 0.15 mm per end. Dimension E1 does not include interlead flash or protrusion, which shall not exceed 0.25 mm per side.
4. Dimensioning and tolerancing per ASME Y14.5M
   - BSC: Basic Dimension. Theoretically exact value shown without tolerances.
   - REF: Reference Dimension, usually without tolerance, for information purposes only.
5. Datums A & B to be determined at Datum H.

Microchip Technology Drawing No. C04-094C Sheet 2 of 2
MCP2200

20-Lead Plastic Small Outline (SO) - Wide, 7.50 mm Body [SOIC]

**Note:** For the most current package drawings, please see the Microchip Packaging Specification located at http://www.microchip.com/packaging

---

**RECOMMENDED LAND PATTERN**

---

**Table:**

<table>
<thead>
<tr>
<th>Units</th>
<th>MILLIMETERS</th>
</tr>
</thead>
<tbody>
<tr>
<td>Dimension Limits</td>
<td>MIN</td>
</tr>
<tr>
<td>Contact Pitch</td>
<td>E</td>
</tr>
<tr>
<td>Contact Pad Spacing</td>
<td>C</td>
</tr>
<tr>
<td>Contact Pad Width (X20)</td>
<td>X</td>
</tr>
<tr>
<td>Contact Pad Length (X20)</td>
<td>Y</td>
</tr>
<tr>
<td>Distance Between Pads</td>
<td>G</td>
</tr>
</tbody>
</table>

**Notes:**

1. Dimensioning and tolerancing per ASME Y14.5M
   BSC: Basic Dimension. Theoretically exact value shown without tolerances.

Microchip Technology Drawing No. C04-2094A
20-Lead Plastic Shrink Small Outline (SS) – 5.30 mm Body [SSOP]

**Note:** For the most current package drawings, please see the Microchip Packaging Specification located at http://www.microchip.com/packaging

<table>
<thead>
<tr>
<th>Units</th>
<th>MILLIMETERS</th>
</tr>
</thead>
<tbody>
<tr>
<td>Dimension Limits</td>
<td>MIN</td>
</tr>
<tr>
<td>Number of Pins</td>
<td>N</td>
</tr>
<tr>
<td>Pitch</td>
<td>e</td>
</tr>
<tr>
<td>Overall Height</td>
<td>A</td>
</tr>
<tr>
<td>Molded Package Thickness</td>
<td>A2</td>
</tr>
<tr>
<td>Standoff</td>
<td>A1</td>
</tr>
<tr>
<td>Overall Width</td>
<td>E</td>
</tr>
<tr>
<td>Molded Package Width</td>
<td>E1</td>
</tr>
<tr>
<td>Overall Length</td>
<td>D</td>
</tr>
<tr>
<td>Foot Length</td>
<td>L</td>
</tr>
<tr>
<td>Footprint</td>
<td>L1</td>
</tr>
<tr>
<td>Lead Thickness</td>
<td>c</td>
</tr>
<tr>
<td>Foot Angle</td>
<td>ϕ</td>
</tr>
<tr>
<td>Lead Width</td>
<td>b</td>
</tr>
</tbody>
</table>

**Notes:**
1. Pin 1 visual index feature may vary, but must be located within the hatched area.
2. Dimensions D and E1 do not include mold flash or protrusions. Mold flash or protrusions shall not exceed 0.20 mm per side.
3. Dimensioning and tolerancing per ASME Y14.5M.

BSC: Basic Dimension. Theoretically exact value shown without tolerances.

REF: Reference Dimension, usually without tolerance, for information purposes only.

Microchip Technology Drawing C04-072B
20-Lead Plastic Shrink Small Outline (SS) - 5.30 mm Body [SSOP]

Note: For the most current package drawings, please see the Microchip Packaging Specification located at http://www.microchip.com/packaging

<table>
<thead>
<tr>
<th>Units</th>
<th>MILLIMETERS</th>
</tr>
</thead>
<tbody>
<tr>
<td>Dimension Limits</td>
<td>MIN</td>
</tr>
<tr>
<td>Contact Pitch</td>
<td>E</td>
</tr>
<tr>
<td>Contact Pad Spacing</td>
<td>C</td>
</tr>
<tr>
<td>Contact Pad Width (X20)</td>
<td>X1</td>
</tr>
<tr>
<td>Contact Pad Length (X20)</td>
<td>Y1</td>
</tr>
<tr>
<td>Distance Between Pads</td>
<td>G</td>
</tr>
</tbody>
</table>

Notes:
1. Dimensioning and tolerancing per ASME Y14.5M
   BSC: Basic Dimension. Theoretically exact value shown without tolerances.

Microchip Technology Drawing No. C04-2072A
APPENDIX A: REVISION HISTORY

Revision D (March 2017)
The following is the list of modifications:
1. Updated Section 1.8.2 “Power-on Reset (POR)” and added new Figure 1-6.

Revision C (December 2015)
The following is the list of modifications:
1. Added Windows® 8, Windows 8.1 and Windows 10 to Features and Section 1.1 “Supported Operating Systems”.

Revision B (March 2011)
The following is the list of modifications:
1. Added new section Section 1.5.2.
2. Updated entire Section 2.3 “Simple Configuration and I/O DLL”.
3. Added values to parameters TH01 and TH02 for the 20-Lead 5x5 VQFN package in Table 3-2.

Revision A (March 2010)
Original Release of this Document.
# MCP2200

## PRODUCT IDENTIFICATION SYSTEM

To order or obtain information, e.g., on pricing or delivery, contact your local Microchip sales office.

<table>
<thead>
<tr>
<th>PART NO.</th>
<th>[X][(1)]</th>
<th>X</th>
<th>XX</th>
</tr>
</thead>
<tbody>
<tr>
<td>Device</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Tape and Reel Option</td>
<td>Blank = Standard packaging (tube or tray)</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Temperature Range</td>
<td>I = -40°C to +85°C (Industrial)</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Package</td>
<td>MQ = Plastic Quad Flat, No Lead Package 5x5x1 mm Body (VQFN), 20-Lead</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>SO = Plastic Small Outline - Wide, 7.50 mm Body (SO), 20-Lead</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>SS = Plastic Shrink Small Outline - 5.30 mm Body (SS) 20-Lead</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

### Examples:

- **a)** MCP2200- I/MQ: Industrial temperature, 20LD VQFN Package.
- **b)** MCP2200T- I/MQ: Tape and Reel, Industrial temperature, 20LD VQFN Package.
- **c)** MCP2200- I/SO: Industrial temperature, 20LD SOIC Package.
- **d)** MCP2200T- I/SO: Tape and Reel, Industrial temperature, 20LD SOIC Package.
- **e)** MCP2200- I/SS: Industrial temperature, 20LD SSOP Package.
- **f)** MCP2200T- I/SS: Tape and Reel, Industrial temperature, 20LD SSOP Package.

### Note 1:
Tape and Reel identifier only appears in the catalog part number description. This identifier is used for ordering purposes and is not printed on the device package. Check with your Microchip Sales Office for package availability with the Tape and Reel option.
Note the following details of the code protection feature on Microchip devices:

• Microchip products meet the specification contained in their particular Microchip Data Sheet.

• Microchip believes that its family of products is one of the most secure families of its kind on the market today, when used in the intended manner and under normal conditions.

• There are dishonest and possibly illegal methods used to breach the code protection feature. All of these methods, to our knowledge, require using the Microchip products in a manner outside the operating specifications contained in Microchip’s Data Sheets. Most likely, the person doing so is engaged in theft of intellectual property.

• Microchip is willing to work with the customer who is concerned about the integrity of their code.

• Neither Microchip nor any other semiconductor manufacturer can guarantee the security of their code. Code protection does not mean that we are guaranteeing the product as “unbreakable.”

Code protection is constantly evolving. We at Microchip are committed to continuously improving the code protection features of our products. Attempts to break Microchip’s code protection feature may be a violation of the Digital Millennium Copyright Act. If such acts allow unauthorized access to your software or other copyrighted work, you may have a right to sue for relief under that Act.

Information contained in this publication regarding device applications and the like is provided only for your convenience and may be superseded by updates. It is your responsibility to ensure that your application meets with your specifications. MICROCHIP MAKES NO REPRESENTATIONS OR WARRANTIES OF ANY KIND WHETHER EXPRESS OR IMPLIED, WRITTEN OR ORAL, STATUTORY OR OTHERWISE, RELATED TO THE INFORMATION, INCLUDING BUT NOT LIMITED TO ITS CONDITION, QUALITY, PERFORMANCE, MERCHANTABILITY OR FITNESS FOR PURPOSE. Microchip disclaims all liability arising from this information and its use. Use of Microchip devices in life support and/or safety applications is entirely at the buyer's risk, and the buyer agrees to defend, indemnify and hold harmless Microchip from any and all damages, claims, suits, or expenses resulting from such use. No licenses are conveyed, implicitly or otherwise, under any Microchip intellectual property rights unless otherwise stated.

Trademarks

The Microchip name and logo, the Microchip logo, AnyRate, AVR, AVR logo, AVR Freaks, BeaconThings, BitCloud, CryptoMemory, CryptoRF, dsPIC, FlashFlex, flexPWR, Heldo, JukeBlox, Keeloq, Keeloq logo, Kleer, LANCheck, LINK MD, maXSylus, maXTouch, MediaLb, megaAVR, MOST, MOST logo, MPLAB, OptoLyzer, PIC, picoPower, PICSTART, PIC32 logo, Prochip Designer, QTouch, RightTouch, SAM-BA, SpyNic, SST, SST Logo, SuperFlash, tinyAVR, UNI/O, and XMEGA are registered trademarks of Microchip Technology Incorporated in the U.S.A. and other countries.

ClockWorks, The Embedded Control Solutions Company, EtherSynch, Hyper Speed Control, HyperLight Load, IntelliMOS, mTouch, Precision Edge, and Quiet-Wire are registered trademarks of Microchip Technology Incorporated in the U.S.A.


SQTP is a service mark of Microchip Technology Incorporated in the U.S.A.

Silicon Storage Technology is a registered trademark of Microchip Technology Inc. in other countries.

GestIC is a registered trademark of Microchip Technology Germany II GmbH & Co. KG, a subsidiary of Microchip Technology Inc., in other countries.

All other trademarks mentioned herein are property of their respective companies.

© 2011-2017, Microchip Technology Incorporated, All Rights Reserved.

ISBN: 978-1-5224-1447-6

QUALITY MANAGEMENT SYSTEM CERTIFIED BY DNV

ISO/TS 16949
Worldwide Sales and Service

AMERICAS
Corporate Office
2355 West Chandler Blvd.
Chandler, AZ 85224-6199
Tel: 480-792-7200
Fax: 480-792-7277
Technical Support:
http://www.microchip.com/support
Web Address:
www.microchip.com
Atlanta
Duluth, GA
Tel: 678-957-9614
Fax: 678-957-1455
Austin, TX
Tel: 512-257-3370
Boston
Westborough, MA
Tel: 774-760-0087
Fax: 774-760-0088
Chicago
Itasca, IL
Tel: 630-285-0071
Fax: 630-285-0075
Dallas
Addison, TX
Tel: 972-818-7423
Fax: 972-818-2924
Detroit
Novi, MI
Tel: 248-848-4000
Houston, TX
Tel: 281-894-5983
Indianapolis
Noblesville, IN
Tel: 317-773-8323
Fax: 317-773-5453
Tel: 317-536-2380
Los Angeles
Mission Viejo, CA
Tel: 949-462-9523
Fax: 949-462-9608
Tel: 951-273-7800
Raleigh, NC
Tel: 919-844-7510
New York, NY
Tel: 631-435-6000
San Jose, CA
Tel: 408-735-9110
Tel: 408-436-4270
Canada - Toronto
Tel: 905-695-1980
Fax: 905-695-2078
ASIA/PACIFIC
Asia Pacific Office
Suites 3707-14, 37th Floor
Tower 6, The Gateway
Harbour City, Kowloon
Hong Kong
Tel: 852-2943-5100
Fax: 852-2401-3431
Australia - Sydney
Tel: 61-2-9868-6733
Fax: 61-2-9868-6755
China - Beijing
Tel: 86-10-8569-7000
Fax: 86-10-8528-2104
China - Chengdu
Tel: 86-28-8665-5511
Fax: 86-28-8665-7889
China - Chongqing
Tel: 86-23-8960-9588
Fax: 86-23-8980-9500
China - Dongguan
Tel: 86-769-8702-9880
China - Guangzhou
Tel: 86-20-8755-8029
China - Hangzhou
Tel: 86-571-8792-8115
Fax: 86-571-8792-8116
China - Hong Kong SAR
Tel: 852-2943-5100
Fax: 852-2401-3431
China - Nanjing
Tel: 86-25-8473-2460
Fax: 86-25-8473-2470
China - Qingdao
Tel: 86-532-8502-7355
Fax: 86-532-8502-7205
China - Shanghai
Tel: 86-21-3326-8000
Fax: 86-21-3326-8021
China - Shenyang
Tel: 86-24-2334-2829
Fax: 86-24-2334-2393
China - Shenzhen
Tel: 86-755-8664-2200
Fax: 86-755-8203-1760
China - Wuhan
Tel: 86-27-5980-5300
Fax: 86-27-5980-5118
China - Xian
Tel: 86-29-8833-7252
Fax: 86-29-8833-7256
ASIA/PACIFIC
China - Xiamen
Tel: 86-592-2388138
Fax: 86-592-2388130
China - Zhuhai
Tel: 86-756-3210040
Fax: 86-756-3210049
India - Bangalore
Tel: 91-80-3090-4444
Fax: 91-80-3090-4123
India - New Delhi
Tel: 91-11-4160-8631
Fax: 91-11-4160-8632
India - Pune
Tel: 91-20-3019-1500
Japan - Osaka
Tel: 81-6-6152-7160
Fax: 81-6-6152-9310
Japan - Tokyo
Tel: 81-3-6880-3770
Fax: 81-3-6880-3771
Korea - Daegu
Tel: 82-53-744-4301
Fax: 82-53-744-4302
Korea - Seoul
Tel: 82-2-554-7200
Fax: 82-2-558-5932 or 82-2-558-5934
Malaysia - Kuala Lumpur
Tel: 60-3-6201-9857
Fax: 60-3-6201-9859
Malaysia - Penang
Tel: 60-4-227-8870
Fax: 60-4-227-4068
Philippines - Manila
Tel: 63-2-634-9065
Fax: 63-2-634-9069
Singapore
Tel: 65-6334-8870
Fax: 65-6334-8860
Taiwan - Hsin Chu
Tel: 886-3-5778-366
Fax: 886-3-5770-955
Taiwan - Kaohsiung
Tel: 886-7-213-7830
Taiwan - Taipei
Tel: 886-2-2508-8600
Fax: 886-2-2508-0102
Thailand - Bangkok
Tel: 66-2-694-1351
Fax: 66-2-694-1350
EUROPE
Austria - Wels
Tel: 43-7242-2244-39
Fax: 43-7242-2244-393
Denmark - Copenhagen
Tel: 45-4450-2828
Fax: 45-4485-2829
Finland - Espoo
Tel: 358-9-4520-820
France - Paris
Tel: 33-1-69-53-63-20
Fax: 33-1-69-30-90-79
France - Saint Cloud
Tel: 33-1-30-60-70-00
Germany - Garching
Tel: 49-8931-9700
Germany - Haan
Tel: 49-2129-3766400
Germany - Heilbronn
Tel: 49-7131-67-3636
Germany - Karlsruhe
Tel: 49-721-625370
Germany - Munich
Tel: 49-89-627-144-0
Fax: 49-89-627-144-44
Germany - Rosenheim
Tel: 49-8031-354-560
Israel - Ra'anana
Tel: 972-9-744-7705
Italy - Milan
Tel: 39-0331-742611
Fax: 39-0331-466781
Italy - Padova
Tel: 39-049-7626286
Netherlands - Drunen
Tel: 31-416-690399
Fax: 31-416-690340
Norway - Trondheim
Tel: 47-7289-7561
Poland - Warsaw
Tel: 48-22-3325737
Romania - Bucharest
Tel: 40-21-407-87-50
Spain - Madrid
Tel: 34-91-708-08-90
Fax: 34-91-708-08-91
Sweden - Gothenberg
Tel: 46-31-704-60-40
Sweden - Stockholm
Tel: 46-6-5090-4654
UK - Wokingham
Tel: 44-118-921-5800
Fax: 44-118-921-5820

© 2011-2017 Microchip Technology Inc.
11/07/16